Efficient Oblivious Evaluation Protocol and Conditional Disclosure of Secrets for DFA

Kittiphop Phalakarn[1], Nuttapong Attrapadung[2], and Kanta Matsuura[1]

[1] The University of Tokyo, Tokyo, Japan
{kittipop,kanta}@iis.u-tokyo.ac.jp
[2] National Institute of Advanced Industrial Science and Technology, Tokyo, Japan
n.attrapadung@aist.go.jp

Abstract. In oblivious finite automata evaluation, one party holds a private automaton, and the other party holds a private string of characters. The objective is to let the parties know whether the string is accepted by the automaton or not, while keeping their inputs secret. The applications include DNA searching, pattern matching, and more. Most of the previous works are based on asymmetric cryptographic primitives, such as homomorphic encryption and oblivious transfer. These primitives are significantly slower than symmetric ones. Moreover, some protocols also require several rounds of interaction. As our main contribution, we propose an oblivious finite automata evaluation protocol via conditional disclosure of secrets (CDS), using one (potentially malicious) outsourcing server. This results in a constant-round protocol, and no heavy asymmetric-key primitives are needed. Our protocol is based on a building block called "an oblivious CDS scheme for deterministic finite automata" which we also propose in this paper. In addition, we propose a standard CDS scheme for deterministic finite automata as an independent interest.

Keywords: Finite automata · Conditional disclosure of secrets · Multi-client verifiable computation · Secure multi-party computation.

1 Introduction

In a problem of oblivious finite automata evaluation, one party holds a private automaton, and the other party holds a private string of characters. The objective is to let the parties know whether the string is accepted by the automaton or not, while keeping their inputs secret.

The applications include DNA matching, string searching, password format validation, spam email detection, log files audition, and more. As stated in [42], DNA technology can help us predict a probability that a patient will develop a specific disease, and predict the result of the therapy. However, revealing personal DNA sequence to public can be harmful. An undesired parental relationship can be discovered, or an employee may be rejected to work with a company due to a probability to develop some diseases. Thus, DNA matching should be performed in an oblivious way. This is also applied to other sensitive information such as passwords, email contents, and log files.

As an example, a patient may want to know if there is any anomaly in his or her DNA sequence. Since the DNA sequence can be considered as private information, the patient should not reveal his or her DNA in clear. On the other hand, a doctor has the anomaly pattern modeled with regular language. The pattern can be considered as a valuable research insight, and should also be kept secret. To let one or both of the parties know whether the DNA sequence matches the pattern, oblivious finite automata evaluation is perfectly suitable here. As another example, an email message and a malware pattern can be considered as sensitive information. To obliviously check whether the email contains a malware or not, the oblivious finite automata evaluation can also be used in the same way.

However, almost all of the previous works are constructed based on public key cryptographic primitives, such as homomorphic encryption and oblivious transfer (OT). These asymmetric-key operations (e.g. exponentiation) are some orders of magnitude slower than the symmetric-key operations. In addition, some of the previous works require several rounds of interaction.

On the other hand, generic secure multi-party computation protocol can also be used, but their performance can be worse compared to specifically designed methods. Executing string matching algorithm with Yao's garbled circuit protocol [44, 32] can be inefficient due to the number of comparisons involved in the dynamic programming technique. Using information-theoretic protocol is also possible [21, 11], but the process will be interactive, and the round complexity will depend on the size of the circuit, which can be large in this case.

A verifiable oblivious finite automata evaluation protocol in an outsourced setting is also an open problem stated in [45].

1.1 Our Contributions

In this paper, we propose an oblivious finite automata evaluation protocol via conditional disclosure of secrets (CDS). This results in a constant-round protocol, and no heavy asymmetric-key primitives are needed. We claim three contributions of our work as follows.

Oblivious CDS for DFA. We present the first CDS scheme for the class of deterministic finite automata (DFA). DFA allows to compute satisfiability for regular languages, and therefore is suitable for the aforementioned applications (e.g., DNA matching). Previous work on CDS were proposed for some other classes; for example, equality, inner product predicate [15], and set intersection [7]. To the best of our knowledge, we are the first to consider CDS for DFA.

As a short introduction to (standard) CDS scheme for DFA, the scheme involves two senders and a receiver. One sender has a DFA, the other sender has an input string, and the receiver knows both the DFA and the input string. The two senders also have a common secret and a common randomness which are not known to the receiver. Each sender can send only one message to the receiver without any communication with the other sender. The goal of the CDS scheme for DFA is to let the receiver know the secret if and only if the automaton accepts the input string.

In this paper, we propose an *oblivious* CDS scheme for DFA. The main difference between the oblivious CDS and the standard CDS is the information leaked to the receiver. In the standard CDS, the receiver knows the automaton and the string, and knows whether the automaton accepts the string or not, while the receiver in the oblivious CDS will not know. Thus, oblivious CDS is more suitable for privacy-preserving applications.

Oblivious DFA Evaluation Protocol via CDS. We propose an oblivious DFA evaluation protocol using the oblivious CDS scheme for DFA as a building block. The advantage of our protocol is that it is constant-round and non-interactive. Using CDS as the underlying scheme can be seen as a trade-off between adding one (potentially malicious) outsourcing server and using asymmetric cryptographic primitives.

Standard CDS for DFA. As an independent interest, we also propose a standard CDS scheme for DFA. To the best of our knowledge, converting CDS for other computation classes to CDS for DFA is not straightforward; ours is the first explicit construction for such CDS for DFA.

1.2 Our Approaches

One of our goals is to achieve a constant-round protocol for oblivious finite automata evaluation. Previous protocols [14, 38, 45] that perform in constant rounds for a similar task all use the idea of garbled circuits and require asymmetric-key primitives such as homomorphic encryption or oblivious transfer. Intuitively, these asymmetric-key primitives play an essential role in hiding private inputs from one party to the other party in the two-party settings.

Our approach to mitigate the need for asymmetric-key primitives is to utilize an additional (potentially malicious) outsourcing server. We observe that oblivious CDS [7] fits wells in this context as it allows an outsourcing server to compute a function obliviously without knowing the inputs or the result. Moreover, known oblivious CDS schemes do not require costly asymmetric-key operations. However, all the previous CDS constructions do not support the class of finite automata (even for standard CDS schemes). To this end, we hence propose the first oblivious CDS for DFA.

We adapt the garbled circuit techniques from Frikken [14] to construct our oblivious CDS for DFA. The construction includes a pseudorandom function (PRF). We then use our oblivious CDS as a building block for our oblivious DFA evaluation protocol, based on the multi-client verifiable computation framework of Bhadauria and Hazay [7]. Note that the CDS schemes in [7] consider predicates of equality and set intersection, which are different from DFA.

For the standard CDS, we extend the techniques in the ABE context from [2] that convert DFA into span programs, and adapt to the CDS context. Our construction of standard CDS is information theoretic. (Our oblivious CDS may imply a standard CDS, but that construction will require a PRF.)

1.3 Related Works

While previous studies on oblivious evaluation for DFA were somewhat peaking about 10 years ago or more [8, 14, 38] (with recent improvements such as [45] being somewhat less major), there are some renewed interests very recently (in 2019 – 2021) in secure computation regarding DFA (and a related class, namely, NC1), in the context of ABE in top conference papers such as [1, 2, 22, 23, 29]. These reflect theoretical and practical interests towards secure DFA computations. We hope that our work offers practical improvements for oblivious DFA evaluation as our protocol is the first explicit constant-round protocol that does not require expensive public-key operations.

In this subsection, we briefly describe related works as follows.

Oblivious Finite Automata Evaluation. The problem of oblivious DFA evaluation was first studied by Troncoso-Pastoriza et al. [42]. Their protocol is based on additive secret sharing, homomorphic encryption, and oblivious transfer. At the start of each round (corresponding to each character in the input string), both parties hold shares of the current state of the automaton. Homomorphic encryption and oblivious transfer are then applied in order to compute the shares of the next state. It is obvious that the number of communication rounds is linear in the length of the input string. The protocol also requires $O(|x||Q|)$ modular exponentiations (where $|x|$ is the length of the input string and $|Q|$ is the total number of states of the DFA), which can be a performance drawback.

The second work proposed by Frikken [14] tried to reduce the number of rounds by using the idea of Yao's garbled circuit [44]. They also reduce the number of modular exponentiations to $O(|x|)$. It is shown in [14] that their protocol is 2 to 3 orders of magnitude faster than [42]. However, the protocol is still based on oblivious transfer.

The first protocol that is secure against malicious adversaries is proposed by Gennaro et al. [18]. It is based on public-key encryption and zero knowledge proof of knowledge. The protocol requires several rounds of interactions. Another work that discussed the security in malicious setting is the work of Mohassel et al. [38]. Using similar idea from [14], they proposed an oblivious evaluation protocol for DFA with alphabets $\{0, 1\}$. The protocol is based on OT extension [26] against malicious adversaries.

Laud and Willemson [28] modeled the transition function as a polynomial, and then evaluate it privately using arithmetic black box (ABB) model. This ABB model can be realized by either secret sharing, homomorphic encryption, or other primitives. If information theoretic primitive such as secret sharing is used, the protocol is also information theoretic. However, performing secure multiplication on secret shares requires several rounds of interactions.

More previous works include the work of Di Crescenzo et al. [13] which is based on conditional transfer protocol, and the work of Zhao et al. [45] which considers a setting with additively shared input string.

Oblivious Finite Automata Evaluation with Outsourcing Servers. The protocol proposed by Blanton and Aliasgari [8] generalizes the work of [42] to the

Table 1. Comparison between oblivious DFA evaluation protocols

Protocol	Primitives	Use Asym.	Parties	Round	Comm. Cost	Security
Troncoso et al. [42]	SS, HE, OT	Y	2	$O(\|x\|)$	$O(\|x\|(\|Q\|+\|\Sigma\|))$	SH
Frikken [14]	OT, PRF	Y	2	$O(1)$	$O(\|x\|\|Q\|\|\Sigma\|)$	SH
Gennaro et al. [18]	HE, ZK	Y	2	$O(\|x\|)$	$O(\|x\|\|Q\|\|\Sigma\|)$	M
Mohassel et al. [38]	OT, PRG	Y	2	$O(1)$	$O(\|x\|\|Q\|\|\Sigma\|)$	M
Laud&Willemson [28]	ABB	Y/N	2+	$O(\|Q\|\|\Sigma\|)$	$O(\|x\|\|Q\|\|\Sigma\|)$	SH/M
Crescenzo et al. [13]	PRF	Y	2	$O(\|x\|\|Q\|\|\Sigma\|)$	$O(\|x\|\|Q\|\|\Sigma\|)$	SH
Zhao et al. [45]	HE, PRG	Y	2	$O(1)$	$O(\|x\|\|Q\|\|\Sigma\|)$	SH
Blanton&Aliasgari [8]	SS, OT	Y	4+	$O(\|x\|)$	$O(\|x\|\|Q\|\|\Sigma\|)$	SH
Wei&Reiter [43]	HE	Y	3	$O(\|x\|)$	$O(\|x\|\|Q\|\|\Sigma\|)$	SH, M*
Ours (Section 4)	PRF	N	3	$O(1)$	$O(\|x\|\|Q\|\|\Sigma\|)$	SH, M†

Asym: Asymmetric-key Primitives SS: Secret Sharing HE: Homomorphic Encryption
OT: Oblivious Transfer PRF: Pseudorandom Function PRG: Pseudorandom Generator
ZK: Zero-Knowledge Proof ABB: Arithmetic Black Box (can be implemented from SS or HE)
SH: Semi-honest M: Malicious Y/N and SH/M: Depend on the building block
*Client can be semi-honest, server can be malicious
†String and automaton holders can be semi-honest, outsourcing server can be malicious

outsourced setting. To keep all the inputs private, their work uses a secret sharing technique to outsource the automaton and the input string to two computing servers. These servers are assumed to be semi-honest. Oblivious transfer is used as a building block. In the case that we want to outsource the inputs to more than two servers, threshold homomorphic encryption must be applied.

Another work in outsourced setting is proposed by Wei and Reiter [43]. In their protocol, a client with a DFA wants to execute it on encrypted string stored on a cloud server. They model the transition function as a polynomial, and then evaluate it privately using homomorphic encryption. The decryption key from the string owner is shared between the client and the cloud server.

We note that almost all of the previous works (including both with and without outsourcing servers) are based on asymmetric cryptographic primitives. Some also require several round of communication and interaction. Comparison between the oblivious evaluation protocols is presented in Table 1.

CDS. Conditional disclosure of secrets (CDS) was firstly proposed in [19] as a building block for symmetrically private information retrieval system (SPIR). Their CDS supports the condition equivalent to monotone access structure of a secret sharing scheme. CDS is also used to construct priced oblivious transfer (i.e., SPIR with cost for each item) in [3]. In addition, CDS is used to reduce share size of secret sharing schemes [6, 35, 33, 4, 5]. Some works tried to relate CDS to attribute-based encryption (ABE) [15]. Recently, the work of [7] proposed new variants of CDS, including private CDS and oblivious CDS. The CDS schemes of [7] are for equality and set intersection classes. The main application of these variants is a multi-client verifiable computation protocol. We list some CDS schemes in the literature in Table 2 (note that this list is not exhaustive). To the best of our knowledge, there is no known CDS for DFA until our work.

Multi-client Verifiable Computation. Gennaro et al. [17] was the first to propose the definition of verifiable computation protocol. Their construction, based on Yao's garbled circuit, is only for two parties, a client and an outsourc-

Table 2. Comparison between CDS schemes

CDS Scheme	Type	Functionalities	Security
Gertner et al. [19]	Standard	Monotone access structure	Info. theoretic
Gay et al. [15]	Standard	Equality, Inner product, Index predicate, Prefix, Disjointness	Info. theoretic
Liu et al. [34]	Standard	Index predicate	Info. theoretic
Bhadauria&Hazay [7]	Oblivious	Equality	Info. theoretic
Bhadauria&Hazay [7]	Private	Equality, Inequality, Set intersection cardinality	Computational
Ours (Section 3)	Oblivious	DFA	Computational
Ours (Section 5)	Standard	DFA	Info. theoretic

ing server. The definition was then generalized to multi-client setting by Choi et al. in [9]. Using non-interactive key exchange (NIKE) protocol, their protocol is secure against malicious server, and semi-honest clients. Gordon et al. [24] later strengthened the security guarantee to malicious clients setting, using homomorphic encryption and attribute-based encryption as building blocks. It can be seen that the existing verifiable computation protocols at that time are quite complex. Recently, Bhadauria and Hazay [7] proposed two-client verifiable computation protocol based on various types of CDS. Some of the advantages provided by CDS are simplicity of the verification, and no need for asymmetric-key primitives.

1.4 Organization

After reviewing preliminaries in Section 2, we propose an oblivious CDS scheme for DFA in Section 3. An oblivious DFA evaluation protocol via CDS is presented in Section 4. As an independent interest, we propose a standard CDS scheme for DFA in Section 5. Finally, Section 6 concludes the paper. Proofs are provided in Appendix C.

2 Preliminaries

In this section, we review related background knowledge, including finite automata, conditional disclosure of secrets, and multi-clients verifiable computation. Definitions of PRF, coin-tossing protocol, and monotone span program are standard, and are provided in Appendix A. Matrices are denoted with bold capitals. We denote $\{1, \ldots, n\}$ and $\{a, a+1, \ldots, b\}$ with $[n]$ and $[a, b]$, respectively. The symbol $\approx$ denotes standard indistinguishability between two distributions, which can be information theoretic or computational depending on the case.

2.1 Finite Automata

In this paper, we consider deterministic finite automata (DFA), which is a special case of nondeterministic finite automata (NFA).

A deterministic finite automaton is defined by a 5-tuple $M = (Q, \Sigma, \Delta, q_0, F)$ where Q is a finite set of states, Σ is a finite set of all possible alphabets, $\Delta : Q \times \Sigma \to Q$ is a transition function which outputs the next state from the current state and the given alphabet, $q_0 \in Q$ is an initial state, and $F \subseteq Q$ is a set of accepting states. (In case of a nondeterministic finite automaton, the

transition function is generalized to $\Delta : Q \times \Sigma \to 2^Q$.) In this work, we assume that states are numbered from 1 to $|Q|$ where $q_0 = 1$, and Σ can be numbered from 0 to $|\Sigma| - 1$. In Section 5, we also assume that there is only one accepting state, $F = \{|Q|\}$. Any DFA can be transformed to satisfy these conditions.[3]

A string of alphabets $x = x_0 x_1 \cdots x_{n-1} \in \Sigma^n$ is accepted by the automaton M if there is a sequence of states $q_0 q_1 \cdots q_n$ such that $q_i = \Delta(q_{i-1}, x_{i-1})$ for all $i \in [n]$, and $q_n \in F$. We say $M(x) = 1$ if M accepts x, and $M(x) = 0$ otherwise.

2.2 Conditional Disclosure of Secrets

In our paper, we only focus on 3-party CDS, where Alice and Bob are senders, and Claire is a receiver. Alice has an input a from the domain A, a secret s from $\{0,1\}^\kappa$, and a randomness r from the domain R. Bob has an input b from the domain B, the same secret s, and the same randomness r. For the standard CDS, Claire only knows the inputs a and b. Everyone agrees on a function $f : A \times B \to \{0,1\}$. Each of Alice and Bob can send only one message to Claire without any communication to each other. The goal of the scheme is to let Claire learn the secret s if $f(a,b) = 1$, and let Claire learn nothing otherwise. The definition of CDS is as follows.

Definition 1 (CDS). *Let $f : A \times B \to \{0,1\}$ be a condition, $s \in \{0,1\}^\kappa$ be a secret, and $r \in R$ be a randomness chosen randomly with uniform distribution. Let $\mathsf{Enc_A}$ and $\mathsf{Enc_B}$ be PPT encoding algorithms, and Dec be a deterministic decoding algorithm. The correctness and secrecy properties must hold as follows.*

Correctness: For all inputs $(a,b) \in A \times B$ where $f(a,b) = 1$,

$$Pr[\mathsf{Dec}(a, b, \mathsf{Enc_A}(a,s,r), \mathsf{Enc_B}(b,s,r)) \neq s] \leq \mathsf{negl}(\kappa).$$

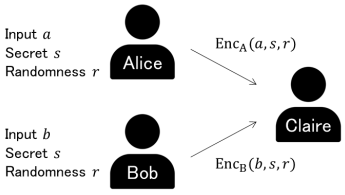
Secrecy: There exists a polynomial time algorithm Sim such that for every input $(a,b) \in A \times B$ where $f(a,b) = 0$ and a secret $s \in \{0,1\}^\kappa$, the following distributions are indistinguishable.

$$\{\mathsf{Sim}(a,b)\}_{a \in A, b \in B} \approx \{\mathsf{Enc_A}(a,s,r), \mathsf{Enc_B}(b,s,r)\}_{a \in A, b \in B}.$$

One useful variant of CDS called as **oblivious CDS** is proposed in [7]. In this setting, Claire does not know a and b. Informally explained, Claire learns a value at the end of the scheme, but Claire will not know whether the condition is satisfied, or whether the decoded value is equal to the secret. The definition of the oblivious CDS from [7] is as follows. For the definition of secrecy of oblivious CDS, we use indistinguishability based definition, which is equivalent to the real-ideal definition in [7].

Definition 2 (Oblivious CDS). *Let $f : A \times B \to \{0,1\}$ be a condition, $s \in \{0,1\}^\kappa$ be a secret, and $r \in R$ be a randomness chosen randomly with uniform distribution. Let $\mathsf{Enc_A}$ and $\mathsf{Enc_B}$ be PPT encoding algorithms, and Dec be a deterministic decoding algorithm. The properties must hold as follows.*

[3] This can be done by adding one special character marking the end of the string.



	Claire knows a and b	Claire knows $f(a,b)$
Standard CDS	✓	✓
Oblivious CDS	✗	✗

Fig. 1. Standard CDS and oblivious CDS

Correctness: For all inputs $(a, b) \in A \times B$ *where* $f(a, b) = 1$,

$$Pr[\mathsf{Dec}(\mathsf{Enc}_{\mathsf{A}}(a, s, r), \mathsf{Enc}_{\mathsf{B}}(b, s, r)) \neq s] \leq \mathsf{negl}(\kappa).$$

Indistinguishability: There exists a polynomial time algorithm Sim *such that for every input* $(a, b) \in A \times B$ *and a secret* $s \in \{0,1\}^\kappa$*, the following distributions are indistinguishable.*

$$\{\mathsf{Sim}(1^{|a|}, 1^{|b|}, \mathsf{Dec}(\mathsf{Enc}_{\mathsf{A}}(a, s, r), \mathsf{Enc}_{\mathsf{B}}(b, s, r)))\}_{a\in A, b\in B} \approx \{\mathsf{Enc}_{\mathsf{A}}(a, s, r), \mathsf{Enc}_{\mathsf{B}}(b, s, r)\}_{a\in A, b\in B}.$$

The diagram of the standard and oblivious CDS is shown in Figure 1.

2.3 Multi-client Verifiable Computation

Similar to [7, 9], we consider a multi-client verifiable computation (MVC) setting where a set of clients outsources the computation to an untrusted computing server. We focus on a non-interactive setting where clients do not interact with each other after the setup phase. In our work, we consider a setting with semi-honest clients and a malicious server (assume no collusion). The clients should follow the protocol perfectly, while the outsourcing server may try to change the computation result. The definition of MVC from [7, 9] is as follows.

Definition 3 (MVC). *Consider a setting where each client has an input* α_i*, and the goal is to compute* $f(\alpha_1, \ldots, \alpha_m)$*. The MVC protocol consists of four algorithms.*

- $\delta \leftarrow \mathsf{Setup}$: *Generate a common random string* δ *for all clients.*
- $(\widetilde{\alpha_i}, \tau_i) \leftarrow \mathsf{Input}(\alpha_i, \delta, 1^\lambda)$: *For each client, using* α_i*,* δ*, and security parameter* 1^λ *as inputs, this algorithm outputs an encoded input* $\widetilde{\alpha_i}$ *and the decoding secret* τ_i *kept private by the client.*
- $(\beta_1, \ldots, \beta_m) \leftarrow \mathsf{Compute}(f, \widetilde{\alpha_1}, \ldots, \widetilde{\alpha_m})$: *Using the function description and the encoded inputs, the computing server executes this algorithm to generate encoded outputs* β_i*.*
- $y \cup \{\perp\} \leftarrow \mathsf{Verify}(\beta_i, \tau_i)$: *For each client, using* β_i *and* τ_i *as inputs, this algorithm generates an output* y *(which supposes to be* $f(\alpha_1, \ldots, \alpha_m)$*), or outputs a symbol* $\perp$ *in case that the server attempted to cheat.*

We are interested in the protocol that is sound and private.

$\mathsf{PrivClient}^{\gamma}_{A,i}(1^{\lambda})$:
$(\alpha^0_1, \ldots, \alpha^0_m), (\alpha^1_1, \ldots, \alpha^1_m) \leftarrow A_0(1^{\lambda})$
where $\alpha^0_i = \alpha^1_i$
and $f(\alpha^0_1, \ldots, \alpha^0_m) = f(\alpha^1_1, \ldots, \alpha^1_m)$
$\delta \leftarrow \mathsf{Setup}$
$(\widetilde{\alpha_j}, \tau_j) \leftarrow \mathsf{Input}(\alpha^{\gamma}_j, \delta, 1^{\lambda})$ for all $j \in [m]$
$(\beta_1, \ldots, \beta_m) \leftarrow \mathsf{Compute}(f, \widetilde{\alpha_1}, \ldots, \widetilde{\alpha_m})$
$\gamma' \leftarrow A_1(\beta_i, \tau_i)$
return γ'

$\mathsf{PrivServer}^{\gamma}_{A}(1^{\lambda})$:
$(\alpha^0_1, \ldots, \alpha^0_m), (\alpha^1_1, \ldots, \alpha^1_m) \leftarrow A_0(1^{\lambda})$
$\delta \leftarrow \mathsf{Setup}$
$(\widetilde{\alpha_j}, \tau_j) \leftarrow \mathsf{Input}(\alpha^{\gamma}_j, \delta, 1^{\lambda})$ for all $j \in [m]$
$\gamma' \leftarrow A_1(\widetilde{\alpha_1}, \ldots, \widetilde{\alpha_m})$
return γ'

Fig. 2. Privacy for multi-client verifiable computation

Soundness: For all inputs $(\alpha_1, \ldots, \alpha_m)$ *and a malicious server* $\mathcal{A}$*, let* $\delta \leftarrow \mathsf{Setup}$*,* $(\widetilde{\alpha_i}, \tau_i) \leftarrow \mathsf{Input}(\alpha_i, \delta, 1^{\lambda})$*,* $(\beta_1, \ldots, \beta_m) \leftarrow \mathcal{A}(f, \widetilde{\alpha_1}, \ldots, \widetilde{\alpha_m})$*, and* $y \cup \{\perp\} \leftarrow \mathsf{Verify}(\beta_i, \tau_i)$ *for all* $i \in [m]$*. It must hold that*

$$Pr[y \neq f(\alpha_1, \ldots, \alpha_m)] \leq negl(\lambda).$$

Privacy against the clients: We consider a setting with adversarial i-th client. From the security game in Figure 2, the MVC is private against the client if

$$|\mathsf{Pr}[\mathsf{PrivClient}^0_{A,i}(1^{\lambda}) = 1] - \mathsf{Pr}[\mathsf{PrivClient}^1_{A,i}(1^{\lambda}) = 1]| \leq \mathsf{negl}(\lambda).$$

Privacy against the server: We consider a setting with adversarial server. From the security game in Figure 2, the MVC is private against the server if

$$|\mathsf{Pr}[\mathsf{PrivServer}^0_{A}(1^{\lambda}) = 1] - \mathsf{Pr}[\mathsf{PrivServer}^1_{A}(1^{\lambda}) = 1]| \leq \mathsf{negl}(\lambda).$$

3 Oblivious CDS for DFA

In this section, we propose an oblivious CDS scheme for DFA, which is used as a building block to construct an oblivious DFA evaluation protocol in the next section.

In the setting of an oblivious CDS scheme for DFA, Alice has a private DFA M, Bob has a private input string x, and both have a common secret s and a common randomness r. On the other hand, Claire does not have any inputs. The condition is defined as $f(M, x) = 1$ if M accepts x, and $f(M, x) = 0$ otherwise. At the end of the scheme, Claire should learn the secret if and only if M accepts x. Our construction is based on the idea of garbled transition matrix from [14]. The proposed scheme is shown in Figure 3.

As an overview, the scheme can be divided into two parts. In the first part, Alice generates a garbled version of the DFA M, and Bob generates keys corresponding to the input string x. Claire, who receives the garbled transition matrix and keys, can recover an intermediate result embedded inside the garbled matrix. This recovered intermediate result depends on the condition of M and x. We use state permutation functions to hide the automaton structure, and use

Oblivious CDS scheme for DFA

Input:

- Alice has a DFA $M = (Q, \Sigma, \Delta, q_0, F)$, a secret s, and a randomness r.
- Bob has an input string $x = x_0 x_1 \cdots x_{n-1}$, a secret s, and a randomness r.
- Claire has no input.

Algorithm:

1. For each $t \in [0, n-1]$, Alice randomly generates a state permutation function $\pi_t : Q \to Q$ and a character permutation function $\phi_t : \Sigma \to \Sigma$ from the randomness r.
2. For each $t \in [0, n-1]$, Alice extracts a garbled state key $k^{\mathsf{state}}_{t,\pi_t(q)}$ for each $q \in Q$, and a garbled character key $k^{\mathsf{char}}_{t,\phi_t(\sigma)}$ for each $\sigma \in \Sigma$ from the randomness r.
3. For each $t \in [0, n-2]$, $q \in Q$, and $\sigma \in \Sigma$, Alice randomly generates a garbled transition matrix element

$$g_{t,\pi_t(q),\phi_t(\sigma)} = H(k^{\mathsf{state}}_{t,\pi_t(q)}, k^{\mathsf{char}}_{t,\phi_t(\sigma)}) \oplus (k^{\mathsf{state}}_{t+1,\pi_{t+1}(\Delta(q,\sigma))} || \pi_{t+1}(\Delta(q,\sigma))).$$

   For each $q \in Q$, and $\sigma \in \Sigma$, Alice generates a garbled transition matrix element of the last step. If $\Delta(q,\sigma) \in F$, then

$$g_{n-1,\pi_{n-1}(q),\phi_{n-1}(\sigma)} = H(k^{\mathsf{state}}_{n-1,\pi_{n-1}(q)}, k^{\mathsf{char}}_{n-1,\phi_{n-1}(\sigma)}) \oplus (w || mw + s)$$

   where w and m are generated from the randomness r. If $\Delta(q,\sigma) \notin F$, then

$$g_{n-1,\pi_{n-1}(q),\phi_{n-1}(\sigma)} = H(k^{\mathsf{state}}_{n-1,\pi_{n-1}(q)}, k^{\mathsf{char}}_{n-1,\phi_{n-1}(\sigma)}) \oplus (c || d)$$

   where (c, d) is a random point not on the line $P(\chi) = m\chi + s$. This point (c, d) is not known to Bob.
4. Alice sends the garbled transition matrix $\{g_{t,\pi_t(q),\phi_t(\sigma)}\}_{t,q,\sigma}$ and the garbled state key of the first state $k^{\mathsf{state}}_{0,\pi_0(q_0)}$ together with $\pi_0(q_0)$ to Claire.
5. For each $t \in [0, n-1]$, Bob generates the character permutation function $\phi_t : \Sigma \to \Sigma$, and extracts the garbled character key $k^{\mathsf{char}}_{t,\phi_t(x_t)}$ from the randomness r in the same way as Alice.
6. Bob randomly chooses a point $(z, mz + s)$ where $z \neq w$ is totally random, and m is generated from the randomness r. Note that Alice does not know this point, and this point is different from Alice's point.
7. Bob sends $(\phi_t(x_t), k^{\mathsf{char}}_{t,\phi_t(x_t)})$ for each $t \in [0, n-1]$, and $(z, mz + s)$ to Claire.
8. For each $t \in [0, n-1]$, Claire, with current permuted state $\pi_t(q)$, permuted character $\phi_t(x_t)$, garbled state key $k^{\mathsf{state}}_{t,\pi_t(q)}$, and garbled character key $k^{\mathsf{char}}_{t,\phi_t(x_t)}$, computes the next permuted state and state key

$$(k^{\mathsf{state}}_{t+1,\pi_{t+1}(\Delta(q,x_t))} || \pi_{t+1}(\Delta(q,x_t))) = g_{t,\pi_t(q),\phi_t(x_t)} \oplus H(k^{\mathsf{state}}_{t,\pi_t(q)}, k^{\mathsf{char}}_{t,\phi_t(x_t)}),$$

   and in the last round discovers

$$(i || j) = g_{n-1,\pi_{n-1}(q),\phi_{n-1}(x_{n-1})} \oplus H(k^{\mathsf{state}}_{n-1,\pi_{n-1}(q)}, k^{\mathsf{char}}_{n-1,\phi_{n-1}(x_{n-1})}).$$

9. Claire interpolates the points (i, j) from the garbled matrix and $(z, mz + s)$ from Bob to find the y-intercept. Claire outputs this value as s'.

Fig. 3. Oblivious CDS scheme for DFA

character permutation functions to hide the input string. The suitable state and character permutations can be as simple as $\pi_t(i) = ((i + u_t) \bmod |Q|) + 1$ and $\phi_t(i) = (i + v_t) \bmod |\Sigma|$ where u_t and v_t are random shift values. Since Claire can only decode the value corresponding to the state at each step, and cannot know whether the final state is an accepting state or not, Claire does not know anything from the intermediate result of the garbled matrix.

In the second part, Claire decodes the secret from the intermediate result. The intermediate result that Claire can recover will be a point on a 2D plane. Before sending messages to Claire, Alice and Bob agree on the same linear equation $P(\chi) = m\chi + s$ where m is generated from the common randomness. If the automaton accepts the input string, Claire will recover a random point on this line. If not, Claire will recover a random point not on this line. Bob also sends the other random point on this line to Claire. Finally, Claire uses two points from the garbled matrix and from Bob to decode the secret via interpolation.

The scheme is secure in the sense that Claire cannot learn anything about the inputs and the result. See the following theorem.

Theorem 1. *Assume that H is a secure PRF. The oblivious CDS for DFA in Figure 3 satisfies correctness and indistinguishability as per definition 2.*

4 Oblivious DFA Evaluation via CDS

In this section, we present an oblivious DFA evaluation protocol via CDS. The protocol shown in Figure 4 is based on the multi-client verifiable computation framework from [7]. In short, we execute two oblivious CDS schemes for DFA M and $\overline{M}$, where $\overline{M}$ is the complement DFA of M. Since exactly one condition must be satisfied, Claire can recover the secret for that condition (but Claire will not know which one). Security then follows from the underlying oblivious CDS. The protocol is secure against semi-honest Alice and Bob, and malicious Claire. Here, Claire can be considered as an untrusted outsourcing server.

There is a reason why we have to execute two oblivious CDS schemes for DFA M and $\overline{M}$. If Alice and Bob execute only one oblivious CDS scheme for the DFA M, Claire may output a random value, and then Alice and Bob may conclude that the DFA does not accept the input string. When the oblivious CDS schemes for both M and $\overline{M}$ are executed, it is difficult for Claire to change the result, since either $s_1 = s'_1$ or $s_2 = s'_2$ must be satisfied, but not both. The protocol in Figure 4 satisfies the following theorem.

Theorem 2. *Assume that H in Figure 3 is a secure PRF. The oblivious DFA evaluation protocol in Figure 4 is a sound and private MVC as per definition 3.*

Application. It is not difficult to see how this protocol can be used for DNA matching and other applications. In this case, Alice holds a pattern modeled with a DFA[4], and Bob holds a DNA sequence. Firstly, they generate common

[4] The method in [41] can be used to transform a pattern p to a finite automaton $LEV_d(p)$ accepts the language $L_d(p)$ contains all strings with Levenshtein distance at

Oblivious DFA evaluation protocol via CDS

Input:

- Alice has a DFA $M = (Q, \Sigma, \Delta, q_0, F)$.
- Bob has an input string $x = x_0 x_1 \cdots x_{n-1}$.
- Claire has no input.

Algorithm:

1. $\delta = (s_1, s_2, r_1, r_2) \leftarrow \mathsf{Setup}$: Alice and Bob get common secrets (s_1, s_2) and common randomness (r_1, r_2) from a coin-tossing protocol or a public source of randomness. These values are not known to Claire.
2. $(\widetilde{\alpha_i}, \tau_i) \leftarrow \mathsf{Input}(\alpha_i, \delta, 1^\lambda)$: Alice and Bob execute the oblivious CDS scheme (Figure 3) for the DFA M and the input string x using the secret s_1 and the randomness r_1. They also execute the oblivious CDS scheme (Figure 3) for the DFA $\overline{M} = (Q, \Sigma, \Delta, q_0, Q - F)$ and the input string x using the secret s_2 and the randomness r_2. Here, $\widetilde{\alpha_1}$ and $\widetilde{\alpha_2}$ are CDS messages, and $\tau_1 = \tau_2 = (s_1, s_2)$.
3. $(\beta_1, \beta_2) \leftarrow \mathsf{Compute}(f, \widetilde{\alpha_1}, \widetilde{\alpha_2})$: Claire computes the output values s_1' and s_2' from both CDS schemes. Here, $\beta_1 = \beta_2 = (s_1', s_2')$.
4. $y \cup \{\perp\} \leftarrow \mathsf{Verify}(\beta_i, \tau_i)$: If $s_1 = s_1'$, both Alice and Bob output "$M(x) = 1$". If $s_2 = s_2'$, both of them output "$M(x) = 0$". Otherwise, output $\perp$.

Fig. 4. Oblivious DFA evaluation protocol via CDS

secrets and randomness. Next, Alice generates the garbled transition matrices of the DFA and its complement, while Bob transforms the DNA sequence into garbled character keys, according to the oblivious CDS scheme for DFA. After the oblivious CDS schemes for DFA are executed, Alice and Bob conclude their result based on Claire's outputs.

4.1 Complexity Analysis

We now briefly analyze round complexity, communication complexity, and computational complexity of our protocol in Figure 4. See Table 1 for more details.

Round Complexity. In Figure 4, two CDS can be executed in parallel. Each of Alice and Bob then send one message to Claire, and Claire sends the results back. The total number of rounds is 2, which is a constant. Our protocol can also be considered as non-interactive.

Communication Complexity. The largest part of the communication is the garbled transition matrices. Thus, the communication complexity is $O(|x||Q||\Sigma|)$.

Computational Complexity. We trade-off a usage of asymmetric-key operations with one outsourcing server. Hence, the protocol can be more efficient

most d from p. It is shown in [42] that a finite automaton for a language $\Sigma^* L_d(p) \Sigma^*$ will not have too many states.

Table 3. Numbers of operations for oblivious DFA evaluation protocols

	Automaton Holder		String Holder		Outsourcing Server	
Protocol	Asym. Comp.	Sym. Comp.	Asym. Comp.	Sym. Comp.	Asym. Comp.	Sym. Comp.
Troncoso et al. [42]	$O(\|x\|\|Q\|\|\Sigma\|)$	$O(\|x\|\|\Sigma\|+\|Q\|)$	$O(\|x\|\|Q\|)$	$O(\|x\|)$	-	-
Frikken [14]	$O(\|x\|)$	$O(\|x\|\|Q\|\|\Sigma\|)$	$O(\|x\|)$	$O(\|x\|)$	-	-
Gennaro et al. [18]	$O(\|Q\|(\|x\|+\|\Sigma\|))$	-	$O(\|x\|\|Q\|)$	-	-	-
Mohassel et al. [38]	$O(\|x\|)$	$O(\|x\|(\|Q\|+\|\Sigma\|))$	$O(\|x\|)$	$O(\|x\|)$	-	-
Laud&Willemson [28]	-	$O(\|x\|\|Q\|\|\Sigma\|)$	-	$O(\|x\|)$	-	-
Crescenzo et al. [13]	$O(\|x\|\|Q\|\|\Sigma\|)$	$O(\|Q\|)$	$O(\|x\|\|Q\|\|\Sigma\|)$	-	-	-
Zhao et al. [45]	$O(\|x\|)$	$O(\|x\|\|Q\|)$	$O(\|x\|)$	$O(\|x\|)$	-	-
Blanton&Aliasgari [8]	-	-	-	-	$O(\|x\|)$	$O(\|x\|\|Q\|\|\Sigma\|)$
Wei&Reitor [43]	$O(\|x\|\|Q\|\|\Sigma\|)$	-	$O(\|x\|\|\Sigma\|(\|Q\|+\|\Sigma\|))$	-	-	-
Ours	-	$O(\|x\|\|Q\|\|\Sigma\|)$	-	-	-	$O(\|x\|)$

- : Negligible compare to other operations

Table 4. Numbers of operations for protocols with outsourcing servers

Protocol	Automaton Holder	Outsourcing Server
Blanton&Aliasgari [8]	-	1-out-of-$\|\Sigma\|$-OT $\times 2$
		1-out-of-$\|Q\|\|\Sigma\|$-OT $\times 2\|x\|$
		1-out-of-$\|Q\|$-OT $\times 2$
Ours	PRF $\times 2\|x\|\|Q\|\|\Sigma\|$	PRF $\times 2\|x\|$

compared to the previous works with heavy usage of homomorphic encryption and oblivious transfer (OT).

We briefly compare numbers of operations of the oblivious DFA evaluation protocols in Table 3. Some of the works have at least $O(|x||Q||\Sigma|)$ asymmetric computation, and some have a bit lower asymmetric computation as at least $O(|x|)$. Ours requires zero asymmetric computation. Although [28] also requires zero asymmetric computation (depend on the building block), it requires $O(|Q||\Sigma|)$ rounds (see Table 1), while ours has constant rounds.

We compare protocols with outsourcing servers in Table 4. Blanton et al. [8] uses $2|x|$ applications of 1-out-of-$|Q||\Sigma|$-OT, while our protocol uses $2|x||Q||\Sigma|$ applications of PRF. Since OT typically uses public-key operations such as modular exponentiations (e.g., [10, 16]), while PRF can be based on symmetric ones such as block ciphers or keyed one-way hash functions (e.g., [12, 20]) which are several orders of magnitude more efficient than public-key operations [39], this suggests that ours should be fundamentally faster than [8]. In more details, according to the state-of-the-art schemes for fast 1-out-of-n OT in [10, 16, 36, 37], running m applications of 1-out-of-n OT requires $O(m)$ modular exponentiations and $O(nm)$ overall time. This suggests that [8] requires $O(|x|)$ asymmetric-key operations and $O(|x||Q||\Sigma|)$ symmetric operations. On the other hand, ours uses $2|x||Q||\Sigma|$ applications of PRF.

We note also that even looking at less-dominant computation part in OT (besides its public-key operations), running one OT application itself typically already requires more than one applications of PRF. Theoretically justified by [25], OT is an expensive operation compared to the evaluation of a PRF or a PRG. The OT protocol in [39] also uses a PRF as a building block. This also confirms that ours protocol should be fundamentally faster than [8].

5 CDS for DFA

As an independent interest, we propose a standard CDS for DFA in this section. In the setting of a standard CDS scheme for DFA, Alice has a DFA M, a secret s, and a randomness r, while Bob has an input string x, the same secret s, and the same randomness r. Claire knows both M and x, but does not know s or r. Claire can learn s if and only if $M(x) = 1$, and learn nothing else if $M(x) = 0$. As an overview of our construction, we transform the automaton and the input string into monotone span programs using a method from [2], and then construct a CDS scheme for those span programs. The transformations are applied to both automaton and input string in order to polynomially bound the size of the span program. Note that the standard CDS for DFA in this section does not require a PRF.[5] Complexity analysis of our standard CDS for DFA is in Appendix B.

5.1 Transform an Input String to a MSP

The following transformation from an input string to a MSP is from [2]. We extend it in order to support any size of alphabets. An input string x is transformed to a MSP $(\boldsymbol{L}_x, \rho_x)$, and a DFA M is transformed to a set of attributes S_M. A universe of attributes for DFA is denoted as

$$\mathcal{U}_M = \{(\sigma, i, j) : i, j \in [Q_{max}], \sigma \in \Sigma\} \cup \{\text{“}\mathsf{Size} = i\text{”} : i \in [Q_{max}]\} \cup \{\text{“}\mathsf{Dummy}\text{”}\}$$

where Q_{max} is a maximum number of states that all parties agree on. Each attribute can be represented by an integer using the following mapping.

$$\text{“}\mathsf{Dummy}\text{”} \mapsto 0, \quad (\sigma, i, j) \mapsto 2|\Sigma|((i+j)^2 + j) + 2\sigma, \quad \text{“}\mathsf{Size} = i\text{”} \mapsto 2i + 1$$

A DFA $M = (Q, \Sigma, \Delta, q_0, F)$ is transformed into a set of attributes

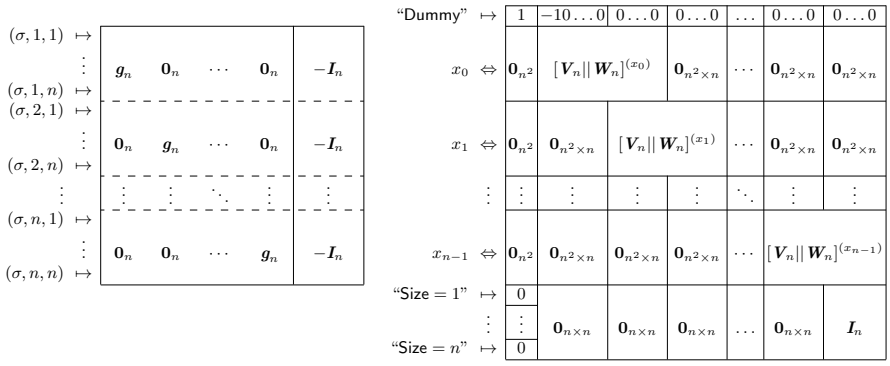
$$S_M = \{\text{“}\mathsf{Dummy}\text{”}\} \cup \{(\sigma, i, j) \in \Sigma \times Q^2 : j = \Delta(\sigma, i)\} \cup \{\text{“}\mathsf{Size} = |Q|\text{”}\}.$$

To transform an input string $x = x_0 x_1 \dots x_{n-1}$ with length $|x| = n$ to a MSP, we define the following matrices as in [2].

- $\boldsymbol{I}_n$ denotes $n \times n$ identity matrix.
- $\boldsymbol{g}_n$ and $\boldsymbol{0}_n$ denote column vectors $(1, \dots, 1)^\top$ and $(0, \dots, 0)^\top$ of size n.
- $\boldsymbol{0}_{m \times n}$ denotes a zero matrix of size $m \times n$.
- $\boldsymbol{V}_n = \boldsymbol{I}_n \otimes \boldsymbol{g}_n = \begin{bmatrix} \boldsymbol{g}_n & \boldsymbol{0}_n & \cdots & \boldsymbol{0}_n \\ \boldsymbol{0}_n & \boldsymbol{g}_n & \cdots & \boldsymbol{0}_n \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0}_n & \boldsymbol{0}_n & \cdots & \boldsymbol{g}_n \end{bmatrix}$ of size $n^2 \times n$.
- $\boldsymbol{W}_n = -\boldsymbol{g}_n \otimes \boldsymbol{I}_n = [-\boldsymbol{I}_n || \dots || -\boldsymbol{I}_n]^\top$ of size $n^2 \times n$.
- For each $\sigma \in \Sigma$, define $[\boldsymbol{V}_n || \boldsymbol{W}_n]^{(\sigma)}$ associated with σ as shown in Table 5 (left). Each row is corresponding to (σ, i, j) for all $i, j \in [n]$.

[5] In practice, PRF can be used to reduce the size of the common randomness.

Table 5. Submatrix $[\boldsymbol{V}_n||\boldsymbol{W}_n]^{(\sigma)}$ associated with σ (refer to Table 4 in [2]) and a MSP $(\boldsymbol{L}_x, \rho_x)$ from an input string x (refer to Table 5 in [2])



The MSP matrix $\boldsymbol{L}_x$ with labeling function ρ_x is shown in Table 5 (right). We refer to the following theorem proved in [2].

Theorem 3. *Using the method above, let $(\boldsymbol{L}_x, \rho_x)$ be a MSP constructed from an input string x, and S_M be a set of attributes constructed from a DFA M. We have $(\boldsymbol{L}_x, \rho_x)$ accepts S_M if and only if $M(x) = 1$ and $|Q| \leq n$.*

5.2 Transform a DFA to a MSP

Similar to the previous subsection, we also transform a DFA to a MSP. We refer to the transformation from a DFA to a MSP from [2] with an extension to support any size of alphabets. A universe of attributes for input strings is denoted as

$$\mathcal{U}_x = \{(i, \sigma) : i \in [0, n_{max}], \sigma \in \Sigma\} \cup \{\text{“}\mathsf{Length} = i\text{”} : i \in [n_{max}]\} \cup \{\text{“}\mathsf{Dummy}\text{”}\}$$

where n_{max} is a maximum length of input string that all parties agree on. Each attribute can be represented by an integer using the following mapping.

$$\text{“}\mathsf{Dummy}\text{”} \mapsto 0, \quad (i, \sigma) \mapsto (|\Sigma| + 1)i + \sigma + 1, \quad \text{“}\mathsf{Length} = i\text{”} \mapsto (|\Sigma| + 1)(i + 1)$$

An input string $x = x_0 x_1 \dots x_{n-1}$ is transformed into a set of attributes

$$S_x = \{\text{“}\mathsf{Dummy}\text{”}\} \cup \{(i, x_i) : i \in [0, n-1]\} \cup \{\text{“}\mathsf{Length} = n\text{”}\}.$$

To transform a DFA $M = (Q, \Sigma, \Delta, q_0, F)$ to a MSP matrix, we define a submatrix $\boldsymbol{Y}^{(\sigma)}$ for each $\sigma \in \Sigma$ with size $|Q| \times |Q|$ in the same way as [2]. The cell of $\boldsymbol{Y}^{(\sigma)}$ at position (i, j) is -1 if $j = \Delta(i, \sigma)$, and is 0 otherwise. The MSP matrix $\boldsymbol{L}_M$ with a labeling function ρ_M is shown in Table 6. We refer to the following theorem proved in [2].

Theorem 4. *Using the method above, let $(\boldsymbol{L}_M, \rho_M)$ be a MSP constructed from a DFA M, and S_x be a set of attributes constructed from an input string x. We have $(\boldsymbol{L}_M, \rho_M)$ accepts S_x if and only if $M(x) = 1$ and $n \leq |Q|$.*

Table 6. A MSP $(\boldsymbol{L}_M, \rho_M)$ from a DFA M (refer to Table 2 in [2])

"Dummy" $\mapsto$	1	$-10\dots0$	$0\dots0$	$0\dots0$	$\dots$	$0\dots0$	$0\dots0$
"$x_0 = 0$" $\mapsto$	$\mathbf{0}_{\lvert Q\rvert}$	$\boldsymbol{I}_{\lvert Q\rvert}$	$\boldsymbol{Y}^{(0)}$	$\mathbf{0}_{\lvert Q\rvert\times\lvert Q\rvert}$	$\cdots$	$\mathbf{0}_{\lvert Q\rvert\times\lvert Q\rvert}$	$\mathbf{0}_{\lvert Q\rvert\times\lvert Q\rvert}$
"$x_0 = 1$" $\mapsto$	$\mathbf{0}_{\lvert Q\rvert}$	$\boldsymbol{I}_{\lvert Q\rvert}$	$\boldsymbol{Y}^{(1)}$	$\mathbf{0}_{\lvert Q\rvert\times\lvert Q\rvert}$	$\cdots$	$\mathbf{0}_{\lvert Q\rvert\times\lvert Q\rvert}$	$\mathbf{0}_{\lvert Q\rvert\times\lvert Q\rvert}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
"$x_0 = \lvert\Sigma\rvert - 1$" $\mapsto$	$\mathbf{0}_{\lvert Q\rvert}$	$\boldsymbol{I}_{\lvert Q\rvert}$	$\boldsymbol{Y}^{(\lvert\Sigma\rvert-1)}$	$\mathbf{0}_{\lvert Q\rvert\times\lvert Q\rvert}$	$\cdots$	$\mathbf{0}_{\lvert Q\rvert\times\lvert Q\rvert}$	$\mathbf{0}_{\lvert Q\rvert\times\lvert Q\rvert}$
"$x_1 = 0$" $\mapsto$	$\mathbf{0}_{\lvert Q\rvert}$	$\mathbf{0}_{\lvert Q\rvert\times\lvert Q\rvert}$	$\boldsymbol{I}_{\lvert Q\rvert}$	$\boldsymbol{Y}^{(0)}$	$\cdots$	$\mathbf{0}_{\lvert Q\rvert\times\lvert Q\rvert}$	$\mathbf{0}_{\lvert Q\rvert\times\lvert Q\rvert}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
"$x_{\lvert Q\rvert-1} = \lvert\Sigma\rvert - 1$" $\mapsto$	$\mathbf{0}_{\lvert Q\rvert}$	$\mathbf{0}_{\lvert Q\rvert\times\lvert Q\rvert}$	$\mathbf{0}_{\lvert Q\rvert\times\lvert Q\rvert}$	$\mathbf{0}_{\lvert Q\rvert\times\lvert Q\rvert}$	$\cdots$	$\boldsymbol{I}_{\lvert Q\rvert}$	$\boldsymbol{Y}^{(\lvert\Sigma\rvert-1)}$
"Length $= 1$" $\mapsto$	0	$0\dots0$	$0\dots01$				
"Length $= 2$" $\mapsto$	0		$0\dots0$	$0\dots01$			
$\vdots$	$\vdots$				$\ddots$		
"Length $= \lvert Q\rvert$" $\mapsto$	0					$0\dots0$	$0\dots01$

5.3 CDS for MSP

A CDS scheme for MSP is proposed in Figure 5. In this setting, Alice has a MSP $(\boldsymbol{L}, \rho)$, a secret s, and a randomness r. Bob has a set of attributes S, the same secret s, and the same randomness r. Claire has only $(\boldsymbol{L}, \rho)$ and S. The idea is that Alice performs a dot product between the MSP and a secret vector, masks the results with random values, and then sends to Claire. At the same time, Bob sends the random values corresponding to the set of attributes. If the set of attributes satisfies the MSP, masked random values can be cancelled out, and the secret can be recovered. The method can be considered as a linear secret sharing based on MSP.

From the scheme, Claire will only learn the values from rows corresponding to the set of attributes. Correctness and security then follow from linear secret sharing of MSP. We have the following theorem.

Theorem 5. *The CDS scheme for MSP proposed in Figure 5 is correct and secure. That is when $(\boldsymbol{L}, \rho)$ accepts S, we have $Pr[Dec((\boldsymbol{L}, \rho), S, \mathsf{Enc}_\mathsf{A}((\boldsymbol{L}, \rho), s, r), \mathsf{Enc}_\mathsf{B}(S, s, r)) = s] = 1$. And when $(\boldsymbol{L}, \rho)$ does not accept S, there exists a simulator Sim such that $\{\mathsf{Sim}((\boldsymbol{L}, \rho), S)\} \approx \{\mathsf{Enc}_\mathsf{A}((\boldsymbol{L}, \rho), s, r), \mathsf{Enc}_\mathsf{B}(S, s, r)\}$.*

5.4 CDS for DFA

We are now ready to use the building blocks from previous subsections to construct a CDS scheme for DFA. The scheme is shown in Figure 6. Alice and Bob first transform a DFA and an input string into MSPs and sets of attributes. After that, they execute two CDS schemes for MSP in parallel. If the automaton accepts the input string, Claire will learn the secret of the CDS scheme. Correctness and security follow from the schemes in previous subsections. We have the following theorem.

CDS scheme for MSP

Input:

- Alice has a MSP $(\boldsymbol{L} \in \mathbb{Z}_p^{\ell \times m}, \rho)$, a secret s, and a randomness r.
- Bob has a set of attributes $S = \{u_1, \ldots, u_n\}$, a secret s, and a randomness r.
- Claire has the MSP $(\boldsymbol{L}, \rho)$, and the set of attributes S.

Algorithm:

1. Alice randomly generates an m-element vector $v = (v_1, \ldots, v_m)$ such that the first element is $v_1 = s$.
2. For each $i \in [\ell]$, let $\boldsymbol{L}_i$ (the i-th row of $\boldsymbol{L}$) be the j-th row that is associated with the attribute u. Alice calculates $w_i = \boldsymbol{L}_i \cdot v + r_{u,j}$ where $r_{u,j}$ is extracted from the randomness r. Alice then sends $\{w_i\}_{i \in [\ell]}$ to Claire.
3. For each $u \in S$, Bob extracts $\{r_{u,j}\}_{j \in [\ell_{max}]}$ from the randomness r in the same way as Alice, where ℓ_{max} is the maximum number of rows that can be associated to an attribute. Bob then sends $\{r_{u,j}\}_{u \in S, j \in [\ell_{max}]}$ to Claire.
4. For each $\boldsymbol{L}_i$ that can map to an attribute in S, Claire computes $(\boldsymbol{L}_i \cdot v)$ from $w_i - r_{u,j}$. If $(\boldsymbol{L}, \rho)$ accepts S, then there exists a vector ω such that $\omega \cdot \boldsymbol{L}_S = (1, 0, \ldots, 0)$. Claire can calculate s from $\omega \cdot (\boldsymbol{L}_S \cdot v)$.

Fig. 5. CDS scheme for MSP

Theorem 6. *The CDS scheme for DFA proposed in Figure 6 is correct and secure. That is when a DFA M accepts an input string x, it holds that $Pr[Dec(M, x, \mathsf{Enc_A}(M, s, r), \mathsf{Enc_B}(x, s, r)) = s] = 1$. And when M does not accept x, there exists a simulator Sim such that $\{\mathsf{Sim}(M, x)\} \approx \{\mathsf{Enc_A}(M, s, r), \mathsf{Enc_B}(x, s, r)\}$.*

6 Concluding Remarks

In this paper, we propose an oblivious CDS scheme for DFA. Then we use it as a building block to construct an oblivious DFA evaluation protocol. We also propose a standard CDS scheme for DFA as an independent interest.

Some of the previous works considered oblivious CDS schemes for NFA, including [28, 40]. We believe that our work could be extended for those situations. For the works considered NFA, although there exists an algorithm to convert a NFA into a DFA, the number of states in the result DFA can be exponentially large. Thus, these schemes can have an advantage in this case.

At this point, we do not know how to construct an oblivious (or even a standard) CDS scheme for NFA without using asymmetric-key operations. Using the method in Section 3 can leak the structure of the NFA to Claire. This is because at each step, Claire will have more than one state keys, and may know which states have transitions to the same state. Extending the method from [2] to the NFA setting is also not trivial. We left this problem as future work.

In addition, it is interesting to extend our protocol to a setting with malicious Alice and Bob. It is also interesting to try constructing oblivious evaluation protocols for pushdown automata and Turing machine. The difficulty is how to

CDS scheme for DFA

Input:

- Alice has a DFA $M = (Q, \Sigma, \Delta, q_0, F)$, a secret s, and randomness (r_1, r_2).
- Bob has an input string $x = x_0 x_1 \dots x_{n-1}$, a secret s, and randomness (r_1, r_2).
- Claire has the finite automaton M, and the input string x.

Algorithm:

1. Alice generates $(\boldsymbol{L}_M, \rho_M)$ and S_M from M.
2. Bob generates $(\boldsymbol{L}_x, \rho_x)$ and S_x from x.
3. Alice and Bob execute two CDS schemes with inputs $((\boldsymbol{L}_M, \rho_M), S_x, s, r_1)$ and inputs $((\boldsymbol{L}_x, \rho_x), S_M, s, r_2)$ at the same time.
4. If M accepts x, then Claire can recover the secret s.

Fig. 6. CDS scheme for DFA

keep track of the memory obliviously. Moreover, we would like to try constructing oblivious protocols via CDS for Moore machine and Mealy machine as well. Although, we can embed the output information into the garbled transition matrix in the same way as [14], the output from this revised protocol will not be verifiable and not be secure against malicious Claire anymore.

Acknowledgments. Nuttapong Attrapadung was partly supported by JSPS KAKENHI Kiban-A Grant Number 19H01109. Kanta Matsuura was partially supported by JSPS KAKENHI Grant Number 17KT0081.

References

1. Agrawal, S., Maitra, M., Yamada, S.: Attribute based encryption (and more) for nondeterministic finite automata from LWE. In: Annual International Cryptology Conference. pp. 765–797. Springer (2019)
2. Agrawal, S., Maitra, M., Yamada, S.: Attribute based encryption for deterministic finite automata from DLIN. In: Theory of Cryptography Conference. pp. 91–117. Springer (2019)
3. Aiello, B., Ishai, Y., Reingold, O.: Priced oblivious transfer: How to sell digital goods. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 119–135. Springer (2001)
4. Applebaum, B., Beimel, A., Farras, O., Nir, O., Peter, N.: Secret-sharing schemes for general and uniform access structures. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 441–471. Springer (2019)
5. Applebaum, B., Beimel, A., Nir, O., Peter, N.: Better secret sharing via robust conditional disclosure of secrets. In: Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing. pp. 280–293 (2020)
6. Beimel, A., Peter, N.: Optimal linear multiparty conditional disclosure of secrets protocols. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 332–362. Springer (2018)

7. Bhadauria, R., Hazay, C.: Multi-clients verifiable computation via conditional disclosure of secrets. In: International Conference on Security and Cryptography for Networks. pp. 150–171. Springer (2020)
8. Blanton, M., Aliasgari, M.: Secure outsourcing of DNA searching via finite automata. In: IFIP Annual Conference on Data and Applications Security and Privacy. pp. 49–64. Springer (2010)
9. Choi, S.G., Katz, J., Kumaresan, R., Cid, C.: Multi-client non-interactive verifiable computation. In: Theory of Cryptography Conference. pp. 499–518. Springer (2013)
10. Chou, T., Orlandi, C.: The simplest protocol for oblivious transfer. In: International Conference on Cryptology and Information Security in Latin America. pp. 40–58. Springer (2015)
11. Cramer, R., Damgård, I., Maurer, U.: General secure multi-party computation from any linear secret-sharing scheme. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 316–334. Springer (2000)
12. Desai, A., Hevia, A., Yin, Y.L.: A practice-oriented treatment of pseudorandom number generators. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 368–383. Springer (2002)
13. Di Crescenzo, G., Coan, B., Kirsch, J.: Privacy-preserving deterministic automata evaluation with encrypted data blocks. In: Data Privacy Management, Cryptocurrencies and Blockchain Technology, pp. 275–294. Springer (2017)
14. Frikken, K.B.: Practical private DNA string searching and matching through efficient oblivious automata evaluation. In: IFIP Annual Conference on Data and Applications Security and Privacy. pp. 81–94. Springer (2009)
15. Gay, R., Kerenidis, I., Wee, H.: Communication complexity of conditional disclosure of secrets and attribute-based encryption. In: Annual Cryptology Conference. pp. 485–502. Springer (2015)
16. Genç, Z.A., Iovino, V., Rial, A.: "the simplest protocol for oblivious transfer" revisited. Information Processing Letters **161**, 105975 (2020)
17. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In: Annual Cryptology Conference. pp. 465–482. Springer (2010)
18. Gennaro, R., Hazay, C., Sorensen, J.S.: Text search protocols with simulation based security. In: International Workshop on Public Key Cryptography. pp. 332–350. Springer (2010)
19. Gertner, Y., Ishai, Y., Kushilevitz, E., Malkin, T.: Protecting data privacy in private information retrieval schemes. In: Proceedings of the thirtieth annual ACM symposium on Theory of computing. pp. 151–160 (1998)
20. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. Journal of the ACM (JACM) **33**(4), 792–807 (1986)
21. Goldwasser, S., Ben-Or, M., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computing. In: Proc. of the 20th STOC. pp. 1–10 (1988)
22. Gong, J., Waters, B., Wee, H.: ABE for DFA from k-lin. In: Annual International Cryptology Conference. pp. 732–764. Springer (2019)
23. Gong, J., Wee, H.: Adaptively secure ABE for DFA from k-lin and more. In: EUROCRYPT 2020-International Conference on Theory and Applications of Cryptographic Techniques. pp. 278–308 (2020)
24. Gordon, S.D., Katz, J., Liu, F.H., Shi, E., Zhou, H.S.: Multi-client verifiable computation with stronger security guarantees. In: Theory of Cryptography Conference. pp. 144–168. Springer (2015)

25. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: Proceedings of the twenty-first annual ACM symposium on Theory of computing. pp. 44–61 (1989)
26. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Annual International Cryptology Conference. pp. 145–161. Springer (2003)
27. Karchmer, M., Wigderson, A.: On span programs. In: Proceedings of the Eighth Annual Structure in Complexity Theory Conference. pp. 102–111. IEEE (1993)
28. Laud, P., Willemson, J.: Universally composable privacy preserving finite automata execution with low online and offline complexity. IACR Cryptol. ePrint Arch. **2013**, 678 (2013)
29. Lin, H., Luo, J.: Compact adaptively secure ABE from k-lin: Beyond NC1 and towards NL. In: EUROCRYPT (2020)
30. Lindell, Y.: Parallel coin-tossing and constant-round secure two-party computation. In: Annual International Cryptology Conference. pp. 171–189. Springer (2001)
31. Lindell, Y.: Parallel coin-tossing and constant-round secure two-party computation. Journal of Cryptology **16**(3) (2003)
32. Lindell, Y., Pinkas, B.: A proof of security of Yao's protocol for two-party computation. Journal of cryptology **22**(2), 161–188 (2009)
33. Liu, T., Vaikuntanathan, V.: Breaking the circuit-size barrier in secret sharing. In: Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing. pp. 699–708 (2018)
34. Liu, T., Vaikuntanathan, V., Wee, H.: Conditional disclosure of secrets via non-linear reconstruction. In: Annual International Cryptology Conference. pp. 758–790. Springer (2017)
35. Liu, T., Vaikuntanathan, V., Wee, H.: Towards breaking the exponential barrier for general secret sharing. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 567–596. Springer (2018)
36. Mansy, D., Rindal, P.: Endemic oblivious transfer. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 309–326 (2019)
37. McQuoid, I., Rosulek, M., Roy, L.: Minimal symmetric PAKE and 1-out-of-n OT from programmable-once public functions. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. pp. 425–442 (2020)
38. Mohassel, P., Niksefat, S., Sadeghian, S., Sadeghiyan, B.: An efficient protocol for oblivious DFA evaluation and applications. In: Cryptographers' Track at the RSA Conference. pp. 398–415. Springer (2012)
39. Naor, M., Pinkas, B.: Computationally secure oblivious transfer. Journal of Cryptology **18**(1), 1–35 (2005)
40. Sasakawa, H., Harada, H., duVerle, D., Arimura, H., Tsuda, K., Sakuma, J.: Oblivious evaluation of non-deterministic finite automata with application to privacy-preserving virus genome detection. In: Proceedings of the 13th Workshop on Privacy in the Electronic Society. pp. 21–30 (2014)
41. Schulz, K.U., Mihov, S.: Fast string correction with Levenshtein automata. International Journal on Document Analysis and Recognition **5**(1), 67–85 (2002)
42. Troncoso-Pastoriza, J.R., Katzenbeisser, S., Celik, M.: Privacy preserving error resilient DNA searching through oblivious automata. In: Proceedings of the 14th ACM conference on Computer and communications security. pp. 519–528 (2007)

43. Wei, L., Reiter, M.K.: Third-party private DFA evaluation on encrypted files in the cloud. In: European Symposium on Research in Computer Security. pp. 523–540. Springer (2012)
44. Yao, A.C.: Protocols for secure computations. In: 23rd Annual Symposium on Foundations of Computer Science. pp. 160–164. IEEE (1982)
45. Zhao, C., Zhao, S., Zhang, B., Jing, S., Chen, Z., Zhao, M.: Oblivious DFA evaluation on joint input and its applications. Information Sciences **528**, 168–180 (2020)

A Definitions

A.1 Pseudorandom Function

We refer to the definition of pseudorandom function (PRF) in [20].

Definition 4. *Let λ be the security parameter, and S_1, S_2, and S_3 be collections of sets indexed by λ. A pseudorandom function $H : S_1 \times S_2 \to S_3$ is a function such that for all probabilistic polynomial time algorithm Λ, the following probability is negligible in λ.*

$$| Pr[s \leftarrow S_1; \Lambda(H(s,\cdot)) = 1] - Pr[a \text{ random map } g \text{ from } S_2 \text{ to } S_3; \Lambda(g(\cdot)) = 1] |$$

A.2 Coin-Tossing Protocol

In a setting of 2-party coin-tossing protocol, Alice and Bob want to generate a common random bit. From the work of Lindell [30, 31], there exists a constant-round protocol for coin-tossing of polynomially many coins, assuming the existence of one-way functions. Both Alice and Bob input 1^λ, and get the same random bit string r of size $poly(\lambda)$.

A.3 Monotone Span Program

A monotone span program (MSP) over $\mathbb{Z}_p$ is defined by $(\boldsymbol{L}, \rho)$ where $\boldsymbol{L} \in \mathbb{Z}_p^{\ell \times m}$ is a matrix, and $\rho : [\ell] \to \mathbb{Z}$ is a labeling function that maps the i-th row to an attribute $\rho(i)$.

Consider a set of attributes $S = \{u_1, \ldots, u_n\} \subseteq \mathbb{Z}$. We construct a set of row indexes $I_S = \{i \in [\ell] : \rho(i) \in S\}$. Let $\boldsymbol{L}_S$ be a submatrix of $\boldsymbol{L}$ restricted to set of rows indexed by I_S. A monotone span program $(\boldsymbol{L}, \rho)$ accepts S if and only if $(1, 0, \ldots, 0)$ is in the row span of $\boldsymbol{L}_S$.

MSP is proved to be equivalent to perfectly secure linear secret sharing [27].

B Complexity of Our Proposed Standard CDS for DFA

Refer to the scheme in Figure 6, Alice and Bob will generate MSP matrices and sets of attributes from the input automaton and the input string, respectively. From the scheme in Figure 5, the communication complexity of the scheme depends on the number of rows of the MSP matrix and the size

of the set of attributes. In Subsection 5.1, the communication complexity is $O(|x|^3 + n_{max}|Q||\Sigma|)$, and in Subsection 5.2, the communication complexity is $O(|\Sigma||Q|^2 + Q_{max}|x|)$. Thus, the total communication complexity is equal to the summation of the two subsections.

Note that it is difficult to compare this complexity to oblivious CDS in Sections 3 and 4, since the definitions and security settings are different.

C Proofs

C.1 Proof of Theorem 1

Proof. (Sketch) For correctness, Claire can recover the state key of the next (permuted) state one by one, using the current (permuted) state key and the corresponding (permuted) character key. If the last state is an accepting state, Claire will recover the point $(w, mw+s)$ from the garbled matrix. Together with the point $(z, mz+s)$ from Bob, Claire can recover $s' = s$. On the other hand, if the last state is not an accepting state, Claire will recover a random point (c, d) from the garbled matrix. The y-intercept of the interpolated line will not be s.

For indistinguishability, Claire will learn only the state keys corresponding to the state sequence induced by the given character keys and the garbled transition matrix, and this sequence of states will look random. Other keys are masked by random values, and will not be recovered by Claire if H is a secure PRF. In the second part, Claire will not know whether the last state is accepting or not. All Claire sees are two random points. Thus, Claire does not know whether the output value s' is equal to the secret s.

More formally, we can construct a simulator Sim_1 according to Definition 2 as follows. The inputs for Sim_1 are the number of states of the DFA $|Q|$, the length of the input string n, and the decoded value s'. Firstly, Sim_1 randomly picks a DFA M' with $|Q|$ states and a string x' with length n. Then Sim_1 generates state permutation functions, character permutation functions, garbled state keys, garbled character keys, and garbled transition matrix elements in the same way as shown in Figure 3. Assume that the point discovered in the accepting case is p_1, the point discovered in the rejecting case is p_2, and the point from Bob is p_3. Sim_1 randomly chooses a random line $P'(\chi)$ with y-intercept at s'. In case that M' accepts x', Sim_1 randomly chooses p_1 and p_3 on $P'(\chi)$, and randomly chooses p_2 not on $P'(\chi)$. On the other hand, if M' does not accept x', Sim_1 randomly chooses p_2 and p_3 on $P'(\chi)$, and randomly chooses p_1 not on $P'(\chi)$. It can be seen that the simulated messages give the output s'. From the security of the PRF, the garbled transition matrices from Sim_1 and from the real execution must be indistinguishable. □

C.2 Proof of Theorem 2

Proof. (Sketch) Soundness of the protocol in Figure 4 is based on the correctness of oblivious CDS scheme for DFA in Theorem 1. Since the DFA M cannot accept

and reject x at the same time, we have $s_1 = s'_1$ or $s_2 = s'_2$, but not both. Thus, only one output value is same as the secret, and the result of oblivious DFA evaluation then follows.

In the case that Claire is malicious, assume that the secrets s_1 and s_2 are λ bits, where λ is the security parameter. If Claire tries to change the result by choosing s'_1 or s'_2 to match s_1 or s_2, the success probability will be $1/2^\lambda$, which is negligible.

Privacy against the server is achieved from the underlying oblivious CDS. For privacy against the clients, we consider the information that Alice and Bob may know from the returned values. In case that Claire returns the secret s, Alice and Bob do not learn anything new, since they already agree on the same linear equation $P(\chi) = m\chi + s$. In case that Claire returns a random value, Alice can learn the point that Bob chose, and Bob can learn a line that Alice's random point is on. From the property of the underlying oblivious CDS, and assume that we use prime field operations, the points that they learn are uniformly distributed on the line $P(\chi)$ and on the plane, respectively. In any case, knowing these points does not affect the privacy of the protocol, since these points are not related to the input automaton or the input string. Thus, there is no advantage for both clients in the security game in Figure 2. □

C.3 Proof of Theorem 3

Proof. For the "if" direction, assume that $M(x) = 1$ and $|Q| \leq n$. Since M accepts x, there exists a sequence of states $q_0 q_1 \cdots q_n$ such that $q_1 = 1$, $q_n = |Q|$, and $q_i = \Delta(q_{i-1}, x_{i-1})$ for all $i \in [n]$.

After choosing rows in $(\boldsymbol{L}_x, \rho_x)$ with attributes from S_M, the remaining submatrix $\boldsymbol{L}_{x,M}$ contains rows equivalent to all possible transitions corresponding to each character in x. To show that rows of $\boldsymbol{L}_{x,M}$ span $(1, 0, \ldots, 0)$, we consider the rows corresponding to the transitions from q_{i-1} to q_i for each x_{i-1}. That is, there exists a row with label (x_{i-1}, q_{i-1}, q_i) between the $((i-1)^2 + 2)$-th to $(i^2 + 2)$-th rows of $\boldsymbol{L}_x$, which also appears in $\boldsymbol{L}_{x,M}$. Adding these rows together with rows corresponding to attributes "Dummy" and "$\mathsf{Size} = |Q|$", all 1 and -1 will be cancelled out except the first 1 in the top left corner. Thus, we show that rows of $\boldsymbol{L}_{x,M}$ span $(1, 0, \ldots, 0)$ as desired.

For the "only if" direction, assume that $M(x) = 0$ and $|Q| \leq n$. Since M does not accept x, there does not exist a sequence of states $q_0 q_1 \cdots q_n$ such that $q_1 = 1$, $q_i = \Delta(q_{i-1}, x_{i-1})$ for all $i \in [n]$, and $q_n = |Q|$. Since $q_n \neq |Q|$, the positions of -1 from the row corresponding to q_n and 1 from the row with attribute "$\mathsf{Size} = |Q|$" are not matched. Thus, $(1, 0, \ldots, 0)$ is not in the span.

In the other case where $|Q| > n$, $\boldsymbol{L}_{x,M}$ will not include the row with "$\mathsf{Size} = |Q|$" attribute. It is impossible to cancel out the last -1 from the row corresponding to q_n. Thus, $(1, 0, \ldots, 0)$ is not in the span. □

C.4 Proof of Theorem 4

Proof. After choosing rows in $(\boldsymbol{L}_M, \rho_M)$ with attributes from S_x, the remaining submatrix $\boldsymbol{L}_{M,x}$ contains rows equivalent to all possible transitions corresponding to each character in x. This is the same as the submatrix $\boldsymbol{L}_{x,M}$ constructed in Theorem 3. Thus, the same proof is applied. □

C.5 Proof of Theorem 5

Proof. For correctness, when $(\boldsymbol{L}, \rho)$ accepts S, there exists a vector ω such that $\boldsymbol{L}_S \cdot \omega = (1, 0, \ldots, 0)$. Claire can calculate $(\boldsymbol{L}_S \cdot v) \cdot \omega = (\boldsymbol{L}_S \cdot \omega) \cdot v = (1, 0, \ldots, 0) \cdot (s, v_2, \ldots, v_m) = s$.

For secrecy, Sim_2 does as follows. After receiving $(\boldsymbol{L}, \rho)$ and S, it randomly generates a vector $v' = (v'_1, \ldots, v'_m)$. Similar to Figure 5, it calculates $w'_i = \boldsymbol{L}_i \cdot v' + r'_{u,j}$ for each $i \in [\ell]$ using randomly generated $r'_{u,j}$. The simulator then outputs $\{w'_i\}_{i \in [\ell]}$ and $\{r'_{u,j}\}_{u \in S, j \in [\ell_{max}]}$.

Since $(\boldsymbol{L}, \rho)$ does not accept S, the value w'_i that Claire recovered are not enough to solve the equations for v'_1, and Claire will not learn the secret s. This is similar to the perfect security of linear secret sharing. Thus, it is impossible to distinguish between an output from Sim_2 and output messages from Alice and Bob. □

C.6 Proof of Theorem 6

Proof. The scheme first transforms the DFA and the input string into MSPs and attributes. Then, we can execute CDS schemes for MSP. Thus, the correctness of the scheme in Figure 6 follows from Theorems 3, 4, and 5.

For secrecy, Sim_3 does as follows. After receiving M and x, it transforms the DFA and the input string into MSPs. After that, it calls a simulator Sim_2 in the proof of Theorem 5 as a sub-function with inputs $((\boldsymbol{L}_M, \rho_M), S_x)$ and $((\boldsymbol{L}_x, \rho_x), S_M)$. Sim_3 then outputs the simulated values from Sim_2. Secrecy of the scheme then follows from Theorem 5. □