Third-Party DFA Evaluation on Encrypted Files

Lei Wei　　Michael K. Reiter

Department of Computer Science
University of North Carolina at Chapel Hill
{lwei,reiter}@cs.unc.edu

Abstract

We present protocols by which a client can evaluate a deterministic finite automaton (DFA) on an encrypted file stored at a server, once authorized to do so by the file owner. Our protocols provably protect the privacy of the DFA and the file contents from a malicious server and the privacy of the file contents (except for the result of the evaluation) from an honest-but-curious client. One of our protocols additionally protects the privacy of the DFA from the client; this property enables others to outsource execution of the protocol to the client without needing to disclose their DFAs to it. Our protocols are practical for a range of cloud storage scenarios.

1 Introduction

Outsourcing file storage to storage service providers (SSPs) and "clouds" can provide significant savings to file owners in terms of management costs and capital investments (e.g., [35]). However, because cloud storage can heighten the risk of file disclosure, prudent file owners encrypt their cloud-resident files to protect their confidentiality. This encryption introduces difficulties in managing access to these files by third parties, however. For example:

- Third-party service providers who are contracted to analyze files stored in the cloud generally cannot do so if the files are encrypted. For example, periodically "scanning" files to detect new malware, as is common today for PC platforms, cannot presently be performed on encrypted files by a third party.
- With some exceptions (see Section 2), third-party customers generally cannot search the files if they are encrypted. Searches on genome datasets, pharmaceutical databases, document corpora, or network logs are critical for research in various fields, but the privacy constraints of these datasets may mandate their encryption, particularly when stored in the cloud.

These difficulties are compounded when the third party views its queries on the files to be sensitive, as well. New malware signatures may be sensitive since releasing them enables attackers to design malware to evade them (e.g., [47]). Customers of datasets in numerous domains (e.g., pharmaceutical research) may view their research interests, and hence their queries, as private.

As a step toward resolving this tension among file protection, search access by authorized third parties, and privacy for third-party queries, in this paper we introduce protocols by which a third-party (called the "client") can perform private searches on encrypted files (stored at the "server"), once it is authorized to do so by the file owner. The type of searches that our protocols enable is motivated by the scenarios above, which in many cases involve pattern matching a file against one or more regular expressions. Multi-pattern string matching is especially common in analysis of content for malware (e.g., [39, 32]) and also is commonplace in searches on genome data, for example. In fact, there are now a number of available genome databases (e.g., [1, 2]) and accompanying tools for multi-pattern matching against them (e.g., [5]). With the goal of improving privacy in such applications, we develop protocols to evaluate a deterministic finite automaton (DFA) of the client's choice on the plaintext of the encrypted file and to return the final state to the client to indicate which, if any, of the patterns encoded in the DFA were matched. We stress that while there is much work on secure two-party computation including the specific case of private DFA evaluation on a private file (see Section 2), very few works have anticipated the possibility that the file is available only in encrypted form. This setting will become more common as data-storage outsourcing grows.

The security properties we prove for our protocols include privacy of the DFA and file contents against arbitrary server adversaries, and privacy of the file (except what is revealed by the evaluation result) against honest-but-curious client adversaries. We refer to our protocols that provide (only) these properties as *one-sided*, since their protection of the DFA pertains only to server adversaries (even while protecting the file from both clients and servers). We further develop a *two-sided* protocol that hides the DFA from the client, as well, permitting others to outsource DFA evaluations to the client without disclosing the DFAs to it. Though our proofs are limited to only honest-but-curious client adversaries, our protocols appear to be extensible with standard techniques to provably protect file privacy even against arbitrary client adversaries, albeit at substantially greater cost. Here we limit our attention to honest-but-curious client adversaries, however, in light of our motivating scenarios that involve third parties that the file owner must authorize and so presumably trusts to some extent.

We believe our protocols will be efficient enough for many practical scenarios. They support evaluation of any DFA over an alphabet Σ on any file consisting of ℓ symbols drawn from Σ, and require the file to be stored using ℓm ciphertexts where $m = |\Sigma|$. Because m is a multiplicative factor in the storage cost, our protocols are best suited for use with small alphabets Σ, e.g., bits ($m = 2$), bytes ($m = 256$), alphanumeric characters ($m = 36$), or DNA nucleotides ($m = 4$ for "A", "C", "G", and "T"). Specifically, we provide three protocols:

1. In Section 4, we present a one-sided protocol that leverages additively homomorphic encryption (e.g., [36]) and transmits $(nm+3)\ell+3$ ciphertexts to evaluate a DFA of n states.
2. In Section 5, we leverage additively homomorphic encryption that additionally supports a *single* homomorphic multiplication of ciphertexts (e.g., [9]) to develop a two-sided protocol that transmits $(nm+3)\ell+3$ ciphertexts.
3. In Section 6, we again leverage this type of encryption to construct a more efficient one-sided protocol that transmits only $(n+m+1)\ell+3$ ciphertexts.

We stress that our protocols require only additively homomorphic encryption or small extensions thereof, for which much more efficient implementations exist (e.g., [36, 9]) than fully homomorphic schemes [19, 45]. Before describing our protocols, we discuss related work in Section 2 and clarify our goals in Section 3.

2 Related Work

The functionalities offered by our protocols could be implemented with general techniques for "computing on encrypted data" [38] or two-party secure computation [46, 24]. These general techniques tend to yield less efficient protocols than one designed for a specific purpose, and our case will be no exception. In particular, the former achieves computations non-interactively using fully homomorphic encryption, for which existing implementations [19, 45, 41, 43] are dramatically more costly than the techniques we use [20]. The latter utilizes a "garbled circuit" construction that is of size linear in the circuit representation of the function to be computed. Despite progress on practical implementations of this technique [33, 4, 37], this limitation renders it substantially more communication-intensive for the problem we consider.

Two-party private DFA evaluation, in which a server has a file and a client has a DFA to evaluate on that file, has been a topic of recent focus. To our knowledge, Troncoso-Pastoriza et al. [44] were the first to present such a protocol, which they proved secure in the honest-but-curious setting. Frikken [17] presented a protocol for the same setting that improved on the round complexity and computational costs. Gennaro et al. [18] gave a two-party DFA evaluation protocol that they proved secure against arbitrary adversaries. Our work differs from these in that in our protocols, the file is made available to the parties only in ciphertext form, and in our protocol in Section 5, even the DFA is made available only in ciphertext form. In this respect, the protocol of Blanton and Aliasgari [6] is relevant; they adapted the Troncoso-Pasoriza et al. protocol to an "outsourcing" model in which the client and server secret-share the DFA and file, respectively, between two additional hosts that interactively evaluate the DFA on the file without reconstructing either one. While our protocol utilizes secret sharing, as well — in our case, of the file owner's file-decryption key — our protocol shares much less data and does not share the client's DFA (or thus require two parties between which to share it) at all. Pattern-matching and search problems other than DFA evaluation have also been studied in the two-party setting, e.g., by Jha et al. [29], Hazay and Lindell [26], Katz and Malka [30], and Hazay and Toft [27]. Again, these works input the plaintext file to one party and so do not directly apply to our setting.

By two-party secret-sharing the file-decryption key and using this to compute on encrypted data, our protocols are related those of Choi et al. [13]. This work developed a protocol based on garbled circuits by which two parties can evaluate a general function after a private decryption key has been shared between them. This protocol can be used to solve the problem we propose, but inherits the aforementioned limitations of garbled circuits.

Specialized protocols for performing searches on encrypted files or database relations have also been developed. For example, searchable encryption [42, 23, 8, 11, 15] enables a party holding a file-decryption key to search for attribute values in the ciphertext file stored at an untrusted server. These techniques have been generalized to support more complex queries, notably conjunctive [10], disjunctive [31] and range queries [40] and inner products [31]. Searchable encryption schemes typically achieve non-interactive queries on encrypted files, in part by attaching "tag" information to the ciphertext of each file to enable the query operation. However, broadening the supported search attributes typically requires expanding the tags, and so the sizes of the tags are determined by the richness of the supported queries. In contrast, in our work the file ciphertexts are independent of the DFA(s) to be evaluated (assuming a fixed size for the alphabets Σ over which the DFAs are defined), and the computation is performed interactively between the two parties. Besides searchable encryption, other works have explored storing database relations at an untrusted server in a form that hides sensitive attributes or associations between attributes while supporting rich queries, e.g., range queries [28, 12] or SQL queries [25, 14]. The security properties offered by these techniques are usually heuristic, without formal definitions and proofs, and we are unaware of any designed to support DFA searches.

3 Problem Description

Let $[k]$ denote the set $\{0, 1, \ldots, k-1\}$. A deterministic finite automaton M is a tuple $\langle Q, \Sigma, \delta, q_{\mathsf{init}} \rangle$ where Q is a set of $|Q| = n$ *states*; Σ is a set (*alphabet*) of $|\Sigma| = m$ *symbols*; $\delta : Q \times \Sigma \rightarrow Q$ is a transition function; and q_{init} is the initial state. We label the n states in Q simply by $[n]$. We retain Σ in the DFA definition to conform with tradition, though we will generally assume that Σ and m are fixed across DFAs and globally known.

Our goal is to enable a client holding a DFA M to interact with a server holding the ciphertext of a file to evaluate M on the file plaintext. More specifically, the client should output the final state to which the file plaintext drives the DFA, i.e., if the plaintext file is a sequence $\langle \sigma_k \rangle_{k \in [\ell]}$ where each $\sigma_k \in \Sigma$, then the client should output $\delta(\ldots\delta(\delta(q_{\mathsf{init}}, \sigma_0), \sigma_1), \ldots, \sigma_{\ell-1})$. We also permit the client to learn the file length ℓ and the server to learn both ℓ and the number of states n in the client's DFA.[1] The client should learn nothing else about the file, however, and the server should learn nothing else about either the file or the client's DFA; we call such a protocol *one-sided* because the DFA is protected from only one party (while the file is protected from both). We will also present a *two-sided* protocol that additionally prevents the client from learning anything about the DFA except again for n and the final state.

We emphasize that neither the client nor the server is the file owner and so neither possesses the file decryption key. So as to enable DFA evaluation, it is thus necessary for the file owner to provide information related to the private file-decryption key to each one. In our protocols, the client and server each receive a share of the private key; this is the means by which the file owner authorizes the client and server to conduct the protocol. As a result, a client and server that collude could pool their information to decrypt the file. We consider such collusion outside our threat model and prove security against only a client or server acting in isolation.

Our protocols do not retrieve the file based on the DFA evaluation results, e.g., in a way that hides from the server what file is being retrieved, though they can be augmented to do so. That is, once the client learns the final state of the DFA evaluation, it can employ various techniques to retrieve the file privately (e.g., [22]). Moreover, some of our motivating scenarios in Section 1, e.g., malware scans of cloud-resident files by a third party, may not require file retrieval but only that matches be reported to the file owner.

[1]Since the final state exposes $\log_2 n$ bits of the file to the client, presumably the server should learn n so as to monitor for excessive exposure or to charge according to the information learned by the client. Moreover, the client can arbitrarily inflate n by adding unreachable states. As such, we consider disclosing n to the server to be practically necessary but of little threat to the client.

4 A One-Sided Protocol

In this section we present a protocol that meets the goals described in Section 3. We call this protocol "one-sided" because it is one-sided in its protection of the DFA used in the protocol; i.e., the DFA is known to the client but must be hidden from the server. In contrast, our protocol hides the file contents from both the server and the client, except for the final DFA state returned to the client.

4.1 Construction

Let "$\leftarrow$" denote assignment and "$s \overset{\$}{\leftarrow} S$" denote the assignment to s of a randomly chosen element of set S. Let κ denote a security parameter that would be typical for public-key cryptosystems based on the difficulty of factoring, e.g., $\kappa \approx 1200$ at a minimum for most applications today.

Encryption scheme Our scheme is built using an additively homomorphic encryption scheme with plaintext space $\mathbb{R}$ where $\langle \mathbb{R}, +_{\mathbb{R}}, \cdot_{\mathbb{R}} \rangle$ denotes a commutative ring. Specifically, an encryption scheme $\mathcal{E}$ includes algorithms Gen, Enc, and Dec where: Gen is a randomized algorithm that on input 1^κ outputs a public-key/private-key pair $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$; Enc is a randomized algorithm that on input public key pk and plaintext $m \in \mathbb{R}$ (where $\mathbb{R}$ can be determined as a function of pk) produces a ciphertext $c \leftarrow \mathsf{Enc}_{pk}(m)$, where $c \in C_{pk}$ and C_{pk} is the ciphertext space determined by pk; and Dec is a deterministic algorithm that on input a private key sk and ciphertext $c \in C_{pk}$ produces a plaintext $m \leftarrow \mathsf{Dec}_{sk}(c)$ where $m \in \mathbb{R}$. In addition, $\mathcal{E}$ supports an operation $+_{pk}$ on ciphertexts such that for any public-key/private-key pair (pk, sk),

$\mathsf{Dec}_{sk}(\mathsf{Enc}_{pk}(m_1) +_{pk} \mathsf{Enc}_{pk}(m_2)) = m_1 +_{\mathbb{R}} m_2$. Using $+_{pk}$, it is possible to implement $\cdot_{pk}$ for which $\mathsf{Dec}_{sk}(m_2 \cdot_{pk} \mathsf{Enc}_{pk}(m_1)) = m_1 \cdot_{\mathbb{R}} m_2$.

We also require $\mathcal{E}$ to support two-party decryption. Specifically, we assume there is an efficient randomized algorithm Share that on input a private key sk outputs shares $(sk_1, sk_2) \leftarrow \mathsf{Share}(sk)$, and that there are efficient deterministic algorithms Dec^1 and Dec^2 such that $\mathsf{Dec}_{sk}(c) = \mathsf{Dec}^2_{sk_2}(c, \mathsf{Dec}^1_{sk_1}(c))$.

An example of an encryption scheme $\mathcal{E}$ that meets the above requirements is due to Paillier [36] with modifications by Damgård and Jurik [16]. In this scheme, the ring $\mathbb{R}$ is $\mathbb{Z}_N$, the ciphertext space $C_{\langle N,g\rangle}$ is $\mathbb{Z}^*_{N^2}$, and the relevant algorithms are as follows.

$\underline{\mathsf{Gen}(1^\kappa)}$: Choose random $\kappa/2$-bit strong primes p, p'; set $N \leftarrow pp'$ and $g \leftarrow N+1$; compute $d \in \mathbb{Z}_{N\varphi(N)}$ such that $d \equiv 1 \bmod N$ and $d \equiv 0 \bmod \varphi(N)$ where $\varphi(N) = (p-1)(p'-1)$; and return the public key $\langle N, g\rangle$ and private key $\langle N, g, d, \varphi(N)\rangle$.

$\underline{\mathsf{Enc}_{\langle N,g\rangle}(m)}$: Select $x \overset{\$}{\leftarrow} \mathbb{Z}^*_N$ and return $g^m x^N \bmod N^2$.

$\underline{\mathsf{Dec}_{\langle N,g,d,\varphi(N)\rangle}(c)}$: Return $((c^d \bmod N^2) - 1)/N$.

$\underline{c_1 +_{\langle N,g\rangle} c_2}$: Return $c_1 c_2 \bmod N^2$.

$\underline{m \cdot_{\langle N,g\rangle} c}$: Return $c^m \bmod N^2$.

$\underline{\mathsf{Share}(\langle N, g, d, \varphi(N)\rangle)}$: Return $sk_1 = \langle N, g, d_1\rangle$ and $sk_2 = \langle N, g, d_2\rangle$ where $d_1 \overset{\$}{\leftarrow} \mathbb{Z}_{N\varphi(N)}$ and $d_2 \leftarrow d - d_1 \bmod N\varphi(N)$.

$\underline{\mathsf{Dec}^1_{\langle N,g,d_1\rangle}(c)}$: Return $c^{d_1} \bmod N^2$.

$\underline{\mathsf{Dec}^2_{\langle N,g,d_2\rangle}(c_1, c_2)}$: Return $((c_2 c_1^{d_2} \bmod N^2) - 1)/N$.

We henceforth refer to this scheme as "Pai".

We use $\sum^{pk}$ to denote summation using $+_{pk}$; $\sum^{\mathbb{R}}$ to denote summation using $+_{\mathbb{R}}$; and $\prod^{\mathbb{R}}$ to denote the product using $\cdot_{\mathbb{R}}$ of a sequence. In addition, for any operation op, we use t_{op} to denote the time required to perform op; e.g., t_{Dec} is the time to perform a Dec operation.

Encoding δ in a Bivariate Polynomial over $\mathbb{R}$ A second ingredient for our protocol is a method for encoding a DFA $\langle Q, \Sigma, q_{\mathsf{init}}, \delta\rangle$, and specifically the transition function δ, as a bivariate polynomial $f(x, y)$ over $\mathbb{R}$ where x is the variable representing a DFA state and y is the variable representing an input symbol. That is, if we treat each state $q \in Q$ and each $\sigma \in \Sigma$ as distinct elements of $\mathbb{R}$, then we would like $f(q, \sigma) = \delta(q, \sigma)$. We can achieve this by choosing f to be the interpolation polynomial

$$f(x,y) = \sum_{\sigma\in\Sigma}^{\mathbb{R}} \left(f_\sigma(x) \cdot_{\mathbb{R}} \Lambda_\sigma(y)\right) \tag{1}$$

where $f_\sigma(q) = \delta(q, \sigma)$ for each $q \in Q$ and where

$$\Lambda_\sigma(y) = \prod_{\substack{\sigma'\in\Sigma\\ \sigma'\neq\sigma}}^{\mathbb{R}} \frac{y -_{\mathbb{R}} \sigma'}{\sigma -_{\mathbb{R}} \sigma'} \tag{2}$$

is a Lagrange basis polynomial. Note that $\Lambda_\sigma(\sigma) = 1$ and $\Lambda_\sigma(\sigma') = 0$ for any $\sigma' \in \Sigma \setminus \{\sigma\}$.

Calculating (2) requires taking multiplicative inverses in $\mathbb{R}$. While not every element of a ring has a multiplicative inverse in the ring, fortunately the ring $\mathbb{Z}_N$ used in Paillier encryption, for example, has negligibly few elements with no inverses, and so there is little risk of encountering an element with no inverse. So, using (2), we can calculate coefficients $\langle\lambda_{\sigma j}\rangle_{j\in[m]}$ so that

$$\Lambda_\sigma(y) = \sum_{j=0}^{m-1}{}^{\mathbb{R}}\ \lambda_{\sigma j} \cdot_{\mathbb{R}} y^j$$

For our algorithm descriptions, we encapsulate this calculation in the procedure $\langle\lambda_{\sigma j}\rangle_{\sigma\in\Sigma, j\in[m]} \leftarrow \mathsf{Lagrange}(\Sigma)$.

Each f_σ needed to compute (1) can again be determined as an interpolation polynomial in the Lagrange form and then expressed as $f_\sigma(x) = \sum_{i=0}^{\mathbb{R}\,n-1} a_{\sigma i} \cdot_{\mathbb{R}} x^i$. In our pseudocode, we encapsulate this calculation as $\langle a_{\sigma i}\rangle_{\sigma\in\Sigma, i\in[n]} \leftarrow \mathsf{ToPoly}(Q, \Sigma, \delta)$.

Having produced coefficients $\langle a_{\sigma i}\rangle_{\sigma\in\Sigma, i\in[n]}$, our protocol additionally requires a function to "shift" these coefficients to produce a polynomial $f'(x, y)$ satisfying $f'(q +_{\mathbb{R}} r, \sigma) = \delta(q, \sigma)$ for each $q \in Q$ and $\sigma \in \Sigma$, for a specified $r \in \mathbb{R}$. We could construct f' "from scratch" in the same way we constructed f, but our protocols in Sections 5–6 require a different approach. Specifically, if we set

$$f'(x,y) = \sum_{\sigma\in\Sigma}^{\mathbb{R}} \left(f'_\sigma(x) \cdot_{\mathbb{R}} \Lambda_\sigma(y)\right)$$

where $f'_\sigma(x) = \sum_{i=0}^{\mathbb{R}\,n-1} a'_{\sigma i} \cdot_{\mathbb{R}} x^i$, then it suffices if $f'_\sigma(x +_{\mathbb{R}} r) = f_\sigma(x)$ for all $\sigma \in \Sigma$. Note that

$$\begin{aligned} f_\sigma(x -_{\mathbb{R}} r) &= \sum_{i=0}^{n-1}{}^{\mathbb{R}}\ a_{\sigma i} \cdot_{\mathbb{R}} (x -_{\mathbb{R}} r)^i \\ &= \sum_{i=0}^{n-1}{}^{\mathbb{R}}\ a_{\sigma i} \cdot_{\mathbb{R}} \sum_{i'=0}^{i}{}^{\mathbb{R}} \binom{i}{i'} \cdot_{\mathbb{R}} x^{i-i'} \cdot_{\mathbb{R}} (-_{\mathbb{R}} r)^{i'} \qquad (3) \\ &= \sum_{i=0}^{n-1}{}^{\mathbb{R}} \left(\sum_{i'=0}^{n-1-i}{}^{\mathbb{R}}\ a_{\sigma(i+i')} \cdot_{\mathbb{R}} \binom{i+i'}{i'} \cdot_{\mathbb{R}} (-_{\mathbb{R}} r)^{i'} \right) \cdot_{\mathbb{R}} x^i \end{aligned}$$

where (3) follows from the binomial theorem. As such, setting

$$a'_{\sigma i} \leftarrow \sum_{i'=0}^{n-1-i}{}^{\mathbb{R}}\ a_{\sigma(i+i')} \cdot_{\mathbb{R}} \binom{i+i'}{i'} \cdot_{\mathbb{R}} (-_{\mathbb{R}} r)^{i'} \tag{4}$$

```
client($pk, sk_1, pk', \langle Q, \Sigma, \delta, q_{\mathsf{init}}\rangle$)          server($pk, sk_2, \Sigma,$
                                                              $\langle c_{kj}\rangle_{k\in[\ell], j\in[m]}$)

c101. $n \leftarrow |Q|, m \leftarrow |\Sigma|$                     s101. $m \leftarrow |\Sigma|$
c102. $\langle a_{\sigma i}\rangle_{\sigma\in\Sigma, i\in[n]}$                        s102. $\langle \lambda_{\sigma j}\rangle_{\sigma\in\Sigma, j\in[m]}$
        $\leftarrow \mathsf{ToPoly}(Q, \Sigma, \delta)$                         $\leftarrow \mathsf{Lagrange}(\Sigma)$
c103. $\alpha \leftarrow \mathsf{Enc}_{pk}(q_{\mathsf{init}})$
                    m101. $\xrightarrow{n}$
                    m102. $\xleftarrow{\ell}$
c104. for $k \leftarrow 0 \ldots \ell - 1$                      s103. for $k \leftarrow 0 \ldots \ell - 1$
c105.     $r \overset{\$}{\leftarrow} \mathbb{R}$
c106.     $\alpha \leftarrow \alpha +_{pk} \mathsf{Enc}_{pk}(r)$
c107.     $\beta \leftarrow \mathsf{Dec}^1_{sk_1}(\alpha)$
c108.     $\rho \leftarrow \mathsf{Enc}_{pk'}(r)$
                    m103. $\xrightarrow{\alpha,\beta,\rho}$
                                                    s104.     $\gamma \leftarrow \mathsf{Dec}^2_{sk_2}(\alpha, \beta)$
                                                    s105.     for $\sigma \in \Sigma$
                                                    s106.         $\Psi_\sigma \leftarrow \sum_{j=0}^{m-1}{}_{pk}\ \lambda_{\sigma j} \cdot_{pk} c_{kj}$
c109.     $\langle a'_{\sigma i}\rangle_{\sigma\in\Sigma, i\in[n]}$                  s107.         for $i \in [n]$
            $\leftarrow \mathsf{Shift}(r, \langle a_{\sigma i}\rangle_{\sigma\in\Sigma, i\in[n]})$   s108.             $\mu_{\sigma i} \leftarrow \gamma^i \cdot_{pk} \Psi_\sigma$
                                                    s109.         endfor
                                                    s110.     endfor
                    m104. $\xleftarrow{\langle \mu_{\sigma i}\rangle_{\sigma\in\Sigma, i\in[n]}}$
c110.     $\alpha \leftarrow \sum_{\sigma\in\Sigma}{}_{pk} \sum_{i=0}^{n-1}{}_{pk}\ a'_{\sigma i} \cdot_{pk} \mu_{\sigma i}$
c111. endfor                                        s111. endfor
c112. $r \overset{\$}{\leftarrow} \mathbb{R}$
c113. $\alpha \leftarrow \alpha +_{pk} \mathsf{Enc}_{pk}(r)$
c114. $\beta \leftarrow \mathsf{Dec}^1_{sk_1}(\alpha)$
c115. $\rho \leftarrow \mathsf{Enc}_{pk'}(r)$
                    m105. $\xrightarrow{\alpha,\beta,\rho}$
                                                    s112. $\gamma^* \leftarrow \mathsf{Dec}^2_{sk_2}(\alpha, \beta)$
                    m106. $\xleftarrow{\gamma^*}$
c116. return $\gamma^* -_{\mathbb{R}} r$
```

Figure 1. Protocol $\Pi_1(\mathcal{E})$, described in Section 4

ensures $f'_\sigma(x +_{\mathbb{R}} r) = f_\sigma(x)$ and, therefore, $f(x +_{\mathbb{R}} r, \sigma) = f'(x, \sigma)$. In our pseudocode, we encapsulate calculations (4) in the invocation $\langle a'_{\sigma i}\rangle_{\sigma\in\Sigma, i\in[n]} \leftarrow \mathsf{Shift}(r, \langle a_{\sigma i}\rangle_{\sigma\in\Sigma, i\in[n]})$.

Protocol steps Our protocol, denoted $\Pi_1(\mathcal{E})$, is shown in Figure 1. Pseudocode for the client is aligned on the left of the figure and labeled c101–c116; the server pseudocode is on the right of the figure and labeled s101–s112; and messages exchanged between them are aligned in the center and labeled m101–m106. The client receives as input a public key pk under which the file (at the server) is encrypted; a share sk_1 of the private key sk corresponding to pk; another public key pk'; and the DFA $\langle Q, \Sigma, \delta, q_{\mathsf{init}}\rangle$. The server receives as input the public key pk, a share sk_2 of the private key sk, the alphabet Σ, and ciphertexts $c_{kj} \leftarrow \mathsf{Enc}_{pk}((\sigma_k)^j)$ of the k-th file symbol σ_k, for each $j \in [m]$ and for each $k \in [\ell]$ where ℓ denotes the file length in symbols. We assume that sk_1 and sk_2 were generated as $(sk_1, sk_2) \leftarrow \mathsf{Share}(sk)$.

We require that the plaintext space of pk' includes the plaintext space $\mathbb{R}$ of pk. To emphasize this, we assume its generation as $(pk', sk') \leftarrow \mathsf{Gen}(1^{\kappa+2})$, since if Pai were in use, this property would be ensured. That said, note that no information about sk' is provided to either party, and so ciphertexts created using pk' — namely ρ created in c108 and c115 and sent in m103 and m105, respectively — will be indecipherable and are ignored in the protocol. These ciphertexts are included to simplify the proof of privacy against client adversaries (Section 4.3) and can be elided in an actual implementation. We will omit these values from further discussion in this section.

The protocol is structured as matching **for** loops executed by the client (c104–c111) and server (s103–s111). The client begins the k-th loop iteration with an encryption α of the current state of its DFA, which it "blinds" by homomorphically adding to it a random ring element r (c105–c106). The client uses its share sk_1 of sk to create the "partial decryption" β of α (c107) and sends α and β to the server (m103). The server uses its share sk_2 of sk to complete the decryption of α to obtain the blinded state γ (s104). The server then computes, for each $\sigma \in \Sigma$ (s105), a value Ψ_σ such that $\Lambda_\sigma(\sigma_k) = \mathsf{Dec}_{sk}(\Psi_\sigma)$ (s106) by utilizing coefficients $\langle \lambda_{\sigma j}\rangle_{\sigma\in\Sigma, j\in[m]}$ output from Lagrange (s102). The server then returns (in m104) values $\langle \mu_{\sigma i}\rangle_{\sigma\in\Sigma, i\in[n]}$ created so that $\mathsf{Dec}_{sk}(\mu_{\sigma i}) = \gamma^i \cdot_{\mathbb{R}} \Lambda_\sigma(\sigma_k)$ (s108). The client uses these ciphertexts, together with coefficients $\langle a'_{\sigma i}\rangle_{\sigma\in\Sigma, i\in[n]}$ calculated using the Shift operation (c109), to assemble a ciphertext α of the new DFA state (c110).

After ℓ loop iterations, the client interacts with the server one more time in order to decrypt the final state. Specifically, the client blinds α again (c113) and sends α and its partial decryption β to the server (m105), for which the server completes the decryption (s112). The server returns the result (m106), and the client then unblinds the final state and returns it (c116).

For brevity, the description in Figure 1 omits numerous checks that the client and server should perform to confirm that the values each receives are well-formed. For example, the client should confirm that $\mu_{\sigma i} \in C_{pk}$ for each $\sigma \in \Sigma$ and $i \in [n]$, upon receiving these in m104. The server should similarly confirm the well-formedness of the values it receives.

Efficiency The communication complexity of $\Pi_1(\mathcal{E})$ is dominated by $(nm + 3)\ell + 3 = O(\ell nm)$ ciphertexts.

This is higher than, e.g., the DFA evaluation protocol of Troncoso-Pasoriza et al. [44], which sends $O(\ell(n+m))$ ciphertexts, and is asymptotically the same as the protocols of Frikken [17] and Gennaro et al. [18]. (The hidden constants in the big-O complexity of the Gennaro et al. protocol are more substantial, however, due to the protocol's extensive use of zero-knowledge proofs.) The computation of $\Pi_1(\mathcal{E})$ is dominated by $O(\ell nm)$ modular exponentiations (when instantiated with Pai); this is outperformed only by Frikken's protocol, which requires $O(\ell)$ modular exponentiations (but still $O(\ell nm)$ symmetric-key operations). But as discussed in Section 2, none of the those protocols are able to work with an encrypted file.

Our protocol could be made noninteractive by replacing the encryption scheme $\mathcal{E}$ with fully homomorphic encryption [19, 45] — the client could encrypt each $a_{\sigma i}$ under the public key pk and send these ciphertexts to the server, enabling the server to perform calculations analogous to c110 itself. That said, existing fully homomorphic schemes are far less efficient than the additively homomorphic schemes for which $\Pi_1(\mathcal{E})$ is designed. For example, the ciphertexts resulting from the security parameter recommended by Gentry and Halevi [20] (yielding a fully homomorphic scheme roughly as secure as 1024-bit RSA) would be about 1.5 MBytes in size. As such, the communication costs with fully homomorphic encryption could conceivably be competitive with $\Pi_1(\mathcal{E})$ only for files of size $\ell \approx 5{,}000$ or larger, assuming $\mathcal{E}$ produces 300-byte ciphertexts (as would Pai with $\kappa = 1200$). Moreover, the computational costs of fully homomorphic schemes are substantially larger than those in only additively homomorphic ones such as Pai [20].

4.2 Security Against Server Attacks

In this section we prove that the server, by executing this protocol (even arbitrarily maliciously), gains no advantage in either determining the DFA the client is evaluating or the plaintext of the file in its possession. That is, we prove only the *privacy* of the file and DFA inputs against server adversaries. We are not concerned with showing that a client can detect server misbehavior, a property often called *correctness*. In fact, we would argue that correctness is not particularly meaningful in our context. After all, the server can simply ignore its input file and execute the protocol (correctly) with the client using another file. That said, $\Pi_1(\mathcal{E})$ could be augmented using standard tools to enforce correctness, with a commensurate impact on performance; we do not explore these alternatives here.

We formalize our security claims against server compromise by defining two separate server adversaries. The first server adversary $S = (S_1, S_2)$ attacks the DFA $M = \langle Q, \Sigma, \delta, q_{\mathsf{init}}\rangle$ held by the client, as described in experiment $\mathbf{Expt}^{\texttt{s-dfa}}_{\Pi_1(\mathcal{E})}$ in Figure 2(a). S_1 first generates a file $\langle \sigma_k \rangle_{k \in [\ell]}$ and two DFAs M_0, M_1. (Note that we use, e.g., "$M_0.Q$" and "$M_1.Q$" to disambiguate their state sets.) S_2 then receives the ciphertexts $\langle c_{kj}\rangle_{k\in[\ell], j\in[m]}$ of its file and oracle access to $\mathsf{clientOr}(pk, sk_1, pk', M_b)$ for b chosen randomly. S_2 is also passed information ϕ created for it by S_1.

$\mathsf{clientOr}$ responds to queries from S_2 as follows, ignoring malformed queries. The first query (say, consisting of simply "start") causes $\mathsf{clientOr}$ to begin the protocol; $\mathsf{clientOr}$ responds with a message of the form n (i.e., of the form of message m101). The second invocation by S_2 must include a single integer ℓ (i.e., of the form of message m102); $\mathsf{clientOr}$ responds with a message of the form α, β, ρ, i.e., three values as in message m103. The next $\ell - 1$ queries by S_2 must contain nm elements of C_{pk}, i.e., $\langle \mu_{\sigma i}\rangle_{\sigma\in\Sigma, i\in[n]}$ as in m104, to which $\mathsf{clientOr}$ responds with three values as in message m103. The next query to $\mathsf{clientOr}$ again must contain nm elements of C_{pk} as in m104, to which $\mathsf{clientOr}$ responds with three values as in m105. The next (and last) query by S_2 can consist simply of a value in $\mathbb{R}$, as in message m106.

Eventually S_2 outputs a bit b', and $\mathbf{Expt}^{\texttt{s-dfa}}_{\Pi_1(\mathcal{E})}(S) = 1$ only if $b' = b$. We say the *advantage* of S is

$$\mathbf{Adv}^{\texttt{s-dfa}}_{\Pi_1(\mathcal{E})}(S) = 2 \cdot \mathbb{P}\left(\mathbf{Expt}^{\texttt{s-dfa}}_{\Pi_1(\mathcal{E})}(S) = 1\right) - 1$$

and define $\mathbf{Adv}^{\texttt{s-dfa}}_{\Pi_1(\mathcal{E})}(t, \ell, n, m) = \max_S \mathbf{Adv}^{\texttt{s-dfa}}_{\Pi_1(\mathcal{E})}(S)$ where the maximum is taken over all adversaries S taking time t and selecting a file of length ℓ and DFAs containing n states and an alphabet of m symbols.

We reduce DFA privacy against server attacks to the IND-CPA [3] security of the encryption scheme. IND-CPA security is defined using the experiment in Figure 3, in which an adversary U is provided a public key $\hat{pk}$ and access to an oracle $\mathsf{Enc}^{\hat{b}}_{\hat{pk}}(\cdot,\cdot)$ that consistently encrypts either the first of its two inputs (if $\hat{b} = 0$) or the second of those inputs (if $\hat{b} = 1$). Eventually U outputs a guess $\hat{b}'$ at $\hat{b}$, and $\mathbf{Expt}^{\texttt{ind-cpa}}_{\mathcal{E}}(U) = 1$ only if $\hat{b}' = \hat{b}$. The IND-CPA advantage of U is defined as

```
Experiment $\mathbf{Expt}^{\texttt{ind-cpa}}_{\mathcal{E}}(U)$
  $(\hat{pk}, \hat{sk}) \leftarrow \mathsf{Gen}(1^\kappa)$
  $\hat{b} \xleftarrow{\$} \{0,1\}$
  $\hat{b}' \leftarrow U^{\mathsf{Enc}^{\hat{b}}_{\hat{pk}}(\cdot,\cdot)}(\hat{pk})$
  if $\hat{b}' = \hat{b}$
    then return 1
    else return 0
```

Figure 3. $\mathbf{Expt}^{\texttt{ind-cpa}}_{\mathcal{E}}(U)$

$$\mathbf{Adv}^{\texttt{ind-cpa}}_{\mathcal{E}}(U) = 2 \cdot \mathbb{P}\left(\mathbf{Expt}^{\texttt{ind-cpa}}_{\mathcal{E}}(U) = 1\right) - 1$$

and then $\mathbf{Adv}^{\texttt{ind-cpa}}_{\mathcal{E}}(t, w) = \max_U \mathbf{Adv}^{\texttt{ind-cpa}}_{\mathcal{E}}(U)$ where the maximum is taken over all adversaries U executing in time t and making w queries to $\mathsf{Enc}^{\hat{b}}_{\hat{pk}}(\cdot,\cdot)$.

In our theorem statements throughout this paper, we omit terms that are negligible as a function of the security parameter κ.

```
Experiment $\mathbf{Expt}^{\texttt{s-dfa}}_{\Pi_1(\mathcal{E})}(S_1, S_2)$
  $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$
  $(sk_1, sk_2) \leftarrow \mathsf{Share}(sk)$
  $(pk', sk') \leftarrow \mathsf{Gen}(1^{\kappa+2})$
  $(\ell, \langle\sigma_k\rangle_{k\in[\ell]}, M_0, M_1, \phi) \leftarrow S_1(pk, sk_2)$
  if $M_0.Q \neq M_1.Q$ or $M_0.\Sigma \neq M_1.\Sigma$ then return $0$
  $b \xleftarrow{\$} \{0,1\}$
  $m \leftarrow |M_b.\Sigma|$
  for $k \in [\ell], j \in [m]$
    $c_{kj} \leftarrow \mathsf{Enc}_{pk}((\sigma_k)^j)$
  $b' \leftarrow S_2^{\mathsf{clientOr}(pk, sk_1, pk', M_b)}(\phi, \langle c_{kj}\rangle_{k\in[\ell], j\in[m]})$
  if $b' = b$
    then return $1$
    else return $0$
```

(a) Experiment $\mathbf{Expt}^{\texttt{s-dfa}}_{\Pi_1(\mathcal{E})}$

```
Experiment $\mathbf{Expt}^{\texttt{s-file}}_{\Pi_1(\mathcal{E})}(S_1, S_2)$
  $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$
  $(sk_1, sk_2) \leftarrow \mathsf{Share}(sk)$
  $(pk', sk') \leftarrow \mathsf{Gen}(1^{\kappa+2})$
  $(\ell, \langle\sigma_{0k}\rangle_{k\in[\ell]}, \langle\sigma_{1k}\rangle_{k\in[\ell]}, M, \phi) \leftarrow S_1(pk, sk_2)$

  $b \xleftarrow{\$} \{0,1\}$
  $m \leftarrow |M.\Sigma|$
  for $k \in [\ell], j \in [m]$
    $c_{kj} \leftarrow \mathsf{Enc}_{pk}((\sigma_{bk})^j)$
  $b' \leftarrow S_2^{\mathsf{clientOr}(pk, sk_1, pk', M)}(\phi, \langle c_{kj}\rangle_{k\in[\ell], j\in[m]})$
  if $b' = b$
    then return $1$
    else return $0$
```

(b) Experiment $\mathbf{Expt}^{\texttt{s-file}}_{\Pi_1(\mathcal{E})}$

Figure 2. Experiments for proving security of $\Pi_1(\mathcal{E})$ against server **adversaries**

Theorem 1. *For $t' = t + t_{\mathsf{Gen}} + t_{\mathsf{Share}}$,*

$$\mathbf{Adv}^{\texttt{s-dfa}}_{\Pi_1(\mathcal{E})}(t, \ell, n, m) \leq 2\mathbf{Adv}^{\texttt{ind-cpa}}_{\mathcal{E}}(t', \ell+1)$$

Proof. Let S be an adversary meeting the parameters t, ℓ, n, and m. Consider a simulation $\mathbf{Sim}^{\texttt{s-dfa}}_{\Pi_1(\mathcal{E})}$ for $\mathbf{Expt}^{\texttt{s-dfa}}_{\Pi_1(\mathcal{E})}$ that differs only by simulating $\mathsf{clientOr}$ so as to substitute all ciphertexts produced with pk' with encryptions of zero (i.e., $\rho \leftarrow \mathsf{Enc}_{pk'}(0)$ in c108 and c115). Then b is hidden information-theoretically from S in $\mathbf{Sim}^{\texttt{s-dfa}}_{\Pi_1(\mathcal{E})}$, since γ is a random element of $\mathbb{R}$ in s104 (see c106) and since γ^* is a random element of $\mathbb{R}$ (see c113). As a result, $\mathbb{P}\left(\mathbf{Sim}^{\texttt{s-dfa}}_{\Pi_1(\mathcal{E})}(S) = 1\right) = \frac{1}{2}$ and for $\mathbf{Adv}^{\texttt{s-dfa}}_{\Pi_1(\mathcal{E})}(S)$ to be nonzero, S must distinguish $\mathbf{Sim}^{\texttt{s-dfa}}_{\Pi_1(\mathcal{E})}$ from $\mathbf{Expt}^{\texttt{s-dfa}}_{\Pi_1(\mathcal{E})}$.

We construct an IND-CPA adversary U that, on input $\hat{pk}$, sets $pk' \leftarrow \hat{pk}$ and uses its own oracle $\mathsf{Enc}^{\hat{b}}_{\hat{pk}}$ to choose between running $\mathbf{Expt}^{\texttt{s-dfa}}_{\Pi_1(\mathcal{E})}$ and $\mathbf{Sim}^{\texttt{s-dfa}}_{\Pi_1(\mathcal{E})}$ for S by setting $\rho \leftarrow \mathsf{Enc}^{\hat{b}}_{\hat{pk}}(0, r)$ in c108 and c115. (Aside from this, U performs $\mathbf{Expt}^{\texttt{s-dfa}}_{\Pi_1(\mathcal{E})}$ faithfully, using $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$ and $(sk_1, sk_2) \leftarrow \mathsf{Share}(sk)$ it generates itself.) U then returns $\hat{b}' = 1$ if S_2 outputs $b' = b$ and $\hat{b}' = 0$, otherwise. Then,

$$\begin{aligned}
&\mathbb{P}\left(\mathbf{Expt}^{\texttt{ind-cpa}}_{\mathcal{E}}(U) = 1\right) \\
&= \frac{1}{2}\mathbb{P}\left(\mathbf{Expt}^{\texttt{s-dfa}}_{\Pi_1(\mathcal{E})}(S) = 1\right) + \frac{1}{2}\mathbb{P}\left(\mathbf{Sim}^{\texttt{s-dfa}}_{\Pi_1(\mathcal{E})}(S) = 0\right) \\
&= \frac{1}{2}\left(\frac{1}{2} + \frac{1}{2}\mathbf{Adv}^{\texttt{s-dfa}}_{\Pi_1(\mathcal{E})}(S)\right) + \frac{1}{4} \\
&= \frac{1}{2} + \frac{1}{4}\mathbf{Adv}^{\texttt{s-dfa}}_{\Pi_1(\mathcal{E})}(S)
\end{aligned}$$

and so $\mathbf{Adv}^{\texttt{ind-cpa}}_{\mathcal{E}}(U) = \frac{1}{2}\mathbf{Adv}^{\texttt{s-dfa}}_{\Pi_1(\mathcal{E})}(S)$.

Note that U makes $\ell + 1$ oracle queries and runs in time $t' = t + t_{\mathsf{Gen}} + t_{\mathsf{Share}}$, due to the need to generate (pk, sk) and sk_2. □

The second server adversary $S = (S_1, S_2)$ attacks the file for which it holds the per-symbol ciphertexts $\langle c_{kj}\rangle_{k\in[\ell], j\in[m]}$ as in experiment $\mathbf{Expt}^{\texttt{s-file}}_{\Pi_1(\mathcal{E})}$ shown in Figure 2(b). Here, S_1 produces two separate, equal-length plaintext files $\langle\sigma_{0k}\rangle_{k\in[\ell]}$, $\langle\sigma_{1k}\rangle_{k\in[\ell]}$ and a DFA M. S_2 then receives the ciphertexts $\langle c_{kj}\rangle_{k\in[\ell], j\in[m]}$ for file $\langle\sigma_{bk}\rangle_{k\in[\ell]}$ where b is chosen randomly. S_2 is also given oracle access to $\mathsf{clientOr}(pk, sk_1, pk', M)$. Eventually S_2 outputs a bit b', and $\mathbf{Expt}^{\texttt{s-file}}_{\Pi_1(\mathcal{E})}(S) = 1$ iff $b' = b$. We say the *advantage* of S is

$$\mathbf{Adv}^{\texttt{s-file}}_{\Pi_1(\mathcal{E})}(S) = 2 \cdot \mathbb{P}\left(\mathbf{Expt}^{\texttt{s-file}}_{\Pi_1(\mathcal{E})}(S) = 1\right) - 1$$

and then $\mathbf{Adv}^{\texttt{s-file}}_{\Pi_1(\mathcal{E})}(t, \ell, n, m) = \max_S \mathbf{Adv}^{\texttt{s-file}}_{\Pi_1(\mathcal{E})}(S)$ where the maximum is taken over all adversaries $S = (S_1, S_2)$ taking time t and producing (from S_1) files of ℓ symbols and a DFA of n states and alphabet of size m.

Theorem 2. *For $t' = t + t_{\mathsf{Gen}} + t_{\mathsf{Share}}$,*

$$\begin{aligned}
&\mathbf{Adv}^{\texttt{s-file}}_{\Pi_1(\mathsf{Pai})}(t, \ell, n, m) \\
&\leq 2\mathbf{Adv}^{\texttt{ind-cpa}}_{\mathsf{Pai}}(t', \ell+1) + \mathbf{Adv}^{\texttt{ind-cpa}}_{\mathsf{Pai}}(t', \ell m)
\end{aligned}$$

Proof. Let $\mathbf{Expt}^{\texttt{s-file-0}}_{\Pi_1(\mathsf{Pai})}$ denote experiment $\mathbf{Expt}^{\texttt{s-file}}_{\Pi_1(\mathsf{Pai})}$ with b fixed at $b = 0$, and let $\mathbf{Expt}^{\texttt{s-file-1}}_{\Pi_1(\mathsf{Pai})}$ denote the experiment $\mathbf{Expt}^{\texttt{s-file}}_{\Pi_1(\mathsf{Pai})}$ with b fixed at $b = 1$. Consider a simulation $\mathbf{Sim}^{\texttt{s-file-0}}_{\Pi_1(\mathsf{Pai})}$ for $\mathbf{Expt}^{\texttt{s-file-0}}_{\Pi_1(\mathsf{Pai})}$ that differs only by simulating $\mathsf{clientOr}$ so as to substitute all ciphertexts produced with pk' with encryptions of zero (i.e., $\rho \leftarrow \mathsf{Enc}_{pk'}(0)$ in c108 and c115). Proceeding as in the proof of Theorem 1, we construct an IND-CPA adversary U_0 that uses its own oracle $\mathsf{Enc}^{\hat{b}}_{\hat{pk}}$ to choose between running $\mathbf{Expt}^{\texttt{s-file-0}}_{\Pi_1(\mathsf{Pai})}$ and $\mathbf{Sim}^{\texttt{s-file-0}}_{\Pi_1(\mathsf{Pai})}$ for S, i.e., by setting $pk' \leftarrow \hat{pk}$ and $\rho \leftarrow \mathsf{Enc}^{\hat{b}}_{\hat{pk}}(r, 0)$ in c108 and c115. (Aside from this, U_0 performs $\mathbf{Expt}^{\texttt{s-file-0}}_{\Pi_1(\mathsf{Pai})}$ faithfully, using $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$

and $(sk_1, sk_2) \leftarrow \mathsf{Share}(sk)$ it generates itself.) U_0 returns $\hat{b}' = 0$ if $b' = b$ and $\hat{b}' = 1$, otherwise. Then,

$$1 + \mathbf{Adv}^{\texttt{ind-cpa}}_{\mathsf{Pai}}(U_0) = 2 \cdot \mathbb{P}\left(\mathbf{Expt}^{\texttt{ind-cpa}}_{\mathsf{Pai}}(U_0) = 1\right) = \mathbb{P}\left(\mathbf{Expt}^{\texttt{s-file-0}}_{\Pi_1(\mathsf{Pai})}(S) = 1\right) + \mathbb{P}\left(\mathbf{Sim}^{\texttt{s-file-0}}_{\Pi_1(\mathsf{Pai})}(S) = 0\right) \tag{5}$$

Now consider a simulation $\mathbf{Sim}^{\texttt{s-file-1}}_{\Pi_1(\mathsf{Pai})}$ for $\mathbf{Expt}^{\texttt{s-file-1}}_{\Pi_1(\mathsf{Pai})}$ that again differs only by simulating $\mathsf{clientOr}$ so as to substitute all ciphertexts produced with pk' with encryptions of zero. As above, we construct an IND-CPA adversary U_1 that uses its own oracle $\mathsf{Enc}^{\hat{b}}_{\hat{pk}}$ to choose between running $\mathbf{Expt}^{\texttt{s-file-1}}_{\Pi_1(\mathsf{Pai})}$ and $\mathbf{Sim}^{\texttt{s-file-1}}_{\Pi_1(\mathsf{Pai})}$ for S, i.e., by setting $pk' \leftarrow \hat{pk}$ and $\rho \leftarrow \mathsf{Enc}^{\hat{b}}_{\hat{pk}}(0, r)$ in c108 and c115. U_1 returns $\hat{b}' = 1$ if $b' = b$ and $\hat{b}' = 0$, otherwise. Then,

$$1 + \mathbf{Adv}^{\texttt{ind-cpa}}_{\mathsf{Pai}}(U_1) = 2 \cdot \mathbb{P}\left(\mathbf{Expt}^{\texttt{ind-cpa}}_{\mathsf{Pai}}(U_1) = 1\right) = \mathbb{P}\left(\mathbf{Sim}^{\texttt{s-file-1}}_{\Pi_1(\mathsf{Pai})}(S) = 0\right) + \mathbb{P}\left(\mathbf{Expt}^{\texttt{s-file-1}}_{\Pi_1(\mathsf{Pai})}(S) = 1\right) \tag{6}$$

Finally, consider an adversary U that uses its oracle $\mathsf{Enc}^{\hat{b}}_{\hat{pk}}$ to choose between running $\mathbf{Sim}^{\texttt{s-file-0}}_{\Pi_1(\mathsf{Pai})}$ and $\mathbf{Sim}^{\texttt{s-file-1}}_{\Pi_1(\mathsf{Pai})}$ for S. Specifically, on input $\hat{pk} = \langle N, g\rangle$, U generates $d_2 \xleftarrow{\$} \mathbb{Z}_{N^2}$ and invokes $S_1(\hat{pk}, sk_2)$ where $sk_2 = \langle N, g, d_2\rangle$. Upon receiving $\langle \sigma_{0k}\rangle_{k\in[\ell]}$ and $\langle \sigma_{1k}\rangle_{k\in[\ell]}$ from S_1, U sets $c_{kj} \leftarrow \mathsf{Enc}^{\hat{b}}_{\hat{pk}}((\sigma_{0k})^j, (\sigma_{1k})^j)$. Additionally, in the simulation of $\mathsf{clientOr}$, U sets $\alpha \leftarrow \mathsf{Enc}_{\hat{pk}}(r)$ in c106 and c113 and $\beta \leftarrow g^r \alpha^{-d_2} \bmod N^2$ in c107 and c114, so that $\alpha^{d_2}\beta \equiv g^r \bmod N^2$. ($U$ also generates pk' itself and constructs all encryptions for pk' as encryptions of zero.) When S_2 outputs b', U outputs b' as $\hat{b}'$. Then,

$$1 + \mathbf{Adv}^{\texttt{ind-cpa}}_{\mathsf{Pai}}(U) = 2 \cdot \mathbb{P}\left(\mathbf{Expt}^{\texttt{ind-cpa}}_{\mathsf{Pai}}(U) = 1\right) = 2 \cdot \mathbb{P}\left(\mathbf{Sim}^{\texttt{s-file}}_{\Pi_1(\mathsf{Pai})}(S) = 1\right) = \mathbb{P}\left(\mathbf{Sim}^{\texttt{s-file-0}}_{\Pi_1(\mathsf{Pai})}(S) = 1\right) + \mathbb{P}\left(\mathbf{Sim}^{\texttt{s-file-1}}_{\Pi_1(\mathsf{Pai})}(S) = 1\right) \tag{7}$$

Adding (5), (6) and (7), we get

$$\begin{aligned}
&3 + \mathbf{Adv}^{\texttt{ind-cpa}}_{\mathsf{Pai}}(U_0) + \mathbf{Adv}^{\texttt{ind-cpa}}_{\mathsf{Pai}}(U) + \mathbf{Adv}^{\texttt{ind-cpa}}_{\mathsf{Pai}}(U_1) \\
&= \mathbb{P}\left(\mathbf{Expt}^{\texttt{s-file-0}}_{\Pi_1(\mathsf{Pai})}(S) = 1\right) + \mathbb{P}\left(\mathbf{Sim}^{\texttt{s-file-0}}_{\Pi_1(\mathsf{Pai})}(S) = 0\right) \\
&+ \mathbb{P}\left(\mathbf{Sim}^{\texttt{s-file-0}}_{\Pi_1(\mathsf{Pai})}(S) = 1\right) + \mathbb{P}\left(\mathbf{Sim}^{\texttt{s-file-1}}_{\Pi_1(\mathsf{Pai})}(S) = 1\right) \\
&+ \mathbb{P}\left(\mathbf{Sim}^{\texttt{s-file-1}}_{\Pi_1(\mathsf{Pai})}(S) = 0\right) + \mathbb{P}\left(\mathbf{Expt}^{\texttt{s-file-1}}_{\Pi_1(\mathsf{Pai})}(S) = 1\right) \\
&= 2 \cdot \mathbb{P}\left(\mathbf{Expt}^{\texttt{s-file}}_{\Pi_1(\mathsf{Pai})}(S) = 1\right) + 2 \\
&= 3 + \mathbf{Adv}^{\texttt{s-file}}_{\Pi_1(\mathsf{Pai})}(S)
\end{aligned}$$

```
Experiment $\mathbf{Expt}^{\texttt{c-file}}_{\Pi_1(\mathcal{E})}(C_1, C_2)$
  $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$
  $(sk_1, sk_2) \leftarrow \mathsf{Share}(sk)$
  $(pk', sk') \leftarrow \mathsf{Gen}(1^{\kappa+2})$
  $(\ell, \langle \sigma_{0k}\rangle_{k\in[\ell]}, \langle \sigma_{1k}\rangle_{k\in[\ell]}, M, \phi) \leftarrow C_1(pk, sk_1, pk')$
  if $M(\langle \sigma_{0k}\rangle_{k\in[\ell]}) \neq M(\langle \sigma_{1k}\rangle_{k\in[\ell]})$ then return 0
  $b \xleftarrow{\$} \{0, 1\}$
  $m \leftarrow |M.\Sigma|$
  for $k \in [\ell], j \in [m]$
    $c_{kj} \leftarrow \mathsf{Enc}_{pk}((\sigma_{bk})^j)$
  $b' \leftarrow C_2^{\mathsf{serverOr}(pk, sk_2, M.\Sigma, \langle c_{kj}\rangle_{k\in[\ell], j\in[m]})}(\phi)$
  if $b' = b$
    then return 1
    else return 0
```

Figure 4. Experiment $\mathbf{Expt}^{\texttt{c-file}}_{\Pi_1(\mathcal{E})}$

The result then follows because each of U_0 and U_1 makes $\ell + 1$ oracle queries and runs in time $t' = t + t_{\mathsf{Gen}} + t_{\mathsf{Share}}$ due to the need to generate (pk, sk) and sk_2, and because U makes ℓm oracle queries and runs in time $t + t_{\mathsf{Gen}} + t_{\mathsf{Share}}$ due to the need to generate pk' and d_2, the latter of which requires time similar to t_{Share}. □

4.3 Security Against Client Attacks

The proof of security against client attacks is easier in the sense that the client has the DFA in its possession, and so security of the DFA is not a concern. (In Section 5 we will introduce a protocol to hide the DFA from the client, as well, and so we will need to revisit this issue later.) The proof of security against the client therefore is concerned with the privacy of only the file. However, by the nature of what the protocol computes for the client — i.e., the final state of a DFA match on the file — it naturally exposes $\log_2 n$ bits about the file to the client. As such, this permits the client to distinguish two files of its choosing, simply by running the protocol correctly using a DFA that distinguishes between the two files it chose.

For this reason, it is necessary to adapt the notion of indistinguishability to apply only to files that produce the same final state for the client's DFA. So, in the experiment $\mathbf{Expt}^{\texttt{c-file}}_{\Pi_1(\mathcal{E})}$ (Figure 4) that we use to define file security against client adversaries, the adversary $C = (C_1, C_2)$ succeeds (i.e., $\mathbf{Expt}^{\texttt{c-file}}_{\Pi_1(\mathcal{E})}(C)$ returns 1) only if the two files $\langle \sigma_{0k}\rangle_{k\in[\ell]}$ and $\langle \sigma_{1k}\rangle_{k\in[\ell]}$ output by C_1 both drive the DFA M, also output by C_1, to the same final state (denoted $M(\langle \sigma_{0k}\rangle_{k\in[\ell]}) = M(\langle \sigma_{1k}\rangle_{k\in[\ell]})$).

This caveat aside, the experiment $\mathbf{Expt}^{\texttt{c-file}}_{\Pi_1(\mathcal{E})}$ is straightforward: C_1 is provided a public key pk, private-key share sk_1, and another public key pk', and returns the two ℓ-symbol files (for ℓ of its choosing) $\langle \sigma_{0k}\rangle_{k\in[\ell]}$ and $\langle \sigma_{1k}\rangle_{k\in[\ell]}$ and a DFA M. Depending on how b is then chosen, one of these files is encrypted using pk and then provided to the server, to which C_2 is given oracle access (denoted $\mathsf{serverOr}(pk, sk_2, M.\Sigma, \langle c_{kj}\rangle_{k\in[\ell],j\in[m]})$).

Adversary C_2 can invoke serverOr first with a message containing an integer n (i.e., with a message of the form m101), to which serverOr returns ℓ (message m102). C_2 can then invoke serverOr up to $\ell+1$ times. The first ℓ such invocations take the form α, β, ρ and correspond to messages of the form m103. Each such invocation elicits a response $\langle \mu_{\sigma i}\rangle_{\sigma\in\Sigma,i\in[n]}$ (i.e., of the form m104). The last client invocation is of the form α, β, ρ and corresponds to m105. This invocation elicits a response γ^* (i.e., of the form m106). Malformed or extra queries are rejected by serverOr.

As discussed in Section 1, we prove file privacy against *honest-but-curious* client adversaries. A client adversary (C_1, C_2) is honest-but-curious if C_2 invokes serverOr exactly as $\Pi_1(\mathcal{E})$ prescribes, using DFA M output by C_1. The advantage $\mathbf{hbcAdv}^{\texttt{c-file}}_{\Pi_1(\mathcal{E})}(C)$ of $C=(C_1,C_2)$ is

$$\mathbf{hbcAdv}^{\texttt{c-file}}_{\Pi_1(\mathcal{E})}(C) = 2\cdot\mathbb{P}\left(\mathbf{Expt}^{\texttt{c-file}}_{\Pi_1(\mathcal{E})}(C)=1\right)-1$$

and $\mathbf{hbcAdv}^{\texttt{c-file}}_{\Pi_1(\mathcal{E})}(t,\ell,n,m) = \max_C \mathbf{Adv}^{\texttt{c-file}}_{\Pi_1(\mathcal{E})}(C)$ where the maximum is taken over honest-but-curious client adversaries C running in total time t and producing files of length ℓ and a DFA of n over an alphabet of m symbols.

Theorem 3. *For* $t' = t + t_{\mathsf{Gen}} + (\ell+1)\cdot t_{\mathsf{Dec}}$,

$$\mathbf{hbcAdv}^{\texttt{c-file}}_{\Pi_1(\mathsf{Pai})}(t,\ell,n,m) \le \mathbf{Adv}^{\texttt{ind-cpa}}_{\mathsf{Pai}}(t',\ell m(1+n))$$

Proof. Given an adversary $C=(C_1,C_2)$ running in time t and selecting files of length ℓ symbols and a DFA of n states over an alphabet of m symbols, we construct an IND-CPA adversary U that demonstrates the theorem as follows. On input $\hat{pk} = \langle N, g\rangle$, U generates $(pk', sk') \leftarrow \mathsf{Gen}(1^{\kappa+2})$ and $d_1 \overset{\$}{\leftarrow} \mathbb{Z}_{N^2}$, and invokes $C_1(\hat{pk}, sk_1, pk')$ where $sk_1 = \langle N, g, d_1\rangle$ to obtain $(\ell, \langle\sigma_{0k}\rangle_{k\in[\ell]}, \langle\sigma_{1k}\rangle_{k\in[\ell]}, M, \phi)$, where $M = \langle Q,\, \Sigma,\, q_{\mathsf{init}},\, \delta\rangle$ is a DFA. Note that d_1 is chosen from a distribution that is statistically indistinguishable from that from which d_1 is chosen in the real system. For $k\in[\ell]$ and $j\in[m]$, U sets $c_{kj} \leftarrow \mathsf{Enc}^{\hat{b}}_{\hat{pk}}((\sigma_{0k})^j, (\sigma_{1k})^j)$.

U then invokes $C_2(\phi)$ and simulates responses to C_2's queries to serverOr as follows (ignoring malformed invocations). In response to the initial query n, the adversary U returns ℓ and, in preparation for the subsequent serverOr invocations by C_2, sets $q_0 \leftarrow q_{\mathsf{init}}$ and $q_1 \leftarrow q_{\mathsf{init}}$. For the k-th query of the form α,β,ρ $(0\le k<\ell)$, the adversary U sets $r \leftarrow \mathsf{Dec}_{sk'}(\rho)$, $\gamma_0 \leftarrow q_0 +_{\mathbb{R}} r$, and $\gamma_1 \leftarrow q_1 +_{\mathbb{R}} r$, and then sets $\mu_{\sigma i} \leftarrow \mathsf{Enc}^{\hat{b}}_{\hat{pk}}(((\gamma_0)^i \cdot_{\mathbb{R}} \Lambda_\sigma(\sigma_{0k}), ((\gamma_1)^i \cdot_{\mathbb{R}} \Lambda_\sigma(\sigma_{1k}))$ for $\sigma\in\Sigma$ and $i\in[n]$. After this, U updates $q_0 \leftarrow \delta(q_0,\sigma_{0k})$ and $q_1 \leftarrow \delta(q_1,\sigma_{1k})$, and returns $\langle\mu_{\sigma i},\rangle_{\sigma\in\Sigma,k\in[n]}$ to C_2. For the last query α,β,ρ, adversary U computes $r \leftarrow \mathsf{Dec}_{sk'}(\rho)$ and returns $\gamma^* = q_0 +_{\mathbb{R}} r$ $(= q_1 +_{\mathbb{R}} r)$ to C_2. When C_2 outputs b', U outputs b', as well.

This simulation is statistically indistinguishable from the real system provided that C is honest-but-curious, and so ignoring terms that are negligible in κ, $\mathbf{hbcAdv}^{\texttt{c-file}}_{\Pi_1(\mathsf{Pai})}(C) = \mathbf{Adv}^{\texttt{ind-cpa}}_{\mathsf{Pai}}(U)$. Note that U runs in $t' = t + t_{\mathsf{Gen}} + (\ell+1)\cdot t_{\mathsf{Dec}}$ where t_{Gen} denotes the time to generate (pk', sk') and t_{Dec} denotes the time to do one Pai decryption. U makes nm oracle queries in order to respond to each of the ℓ oracle queries following the first, plus an additional ℓm queries to create $\langle c_{kj}\rangle_{k\in[\ell],j\in[m]}$. □

5 A Two-Sided Protocol

In this section we extend the protocol $\Pi_1(\mathcal{E})$ to protect the secrecy of the DFA $\langle Q, \Sigma, q_{\mathsf{init}}, \delta\rangle$ from the client. As such, this modification enables the client to execute the protocol on behalf of others who do not trust it with knowledge of the DFA. One scenario in which this protection is desirable is if the client is executing the protocol on behalf of another party (possibly the actual owner of the file) who does not have the bandwidth or processing available for performing the evaluation herself. If that party wants to evaluate a DFA on the cloud-resident files but is currently constrained to doing so from a bandwidth-limited device (e.g., a mobile cellular device), the owner can "encrypt" the DFA as prescribed by our protocol and send the result to the proxy (i.e., client). The client can then conduct the evaluation and return to the requesting party the state in which each searched file left the DFA.

5.1 Construction

Encryption scheme The primary conceptual difference between our protocol in this section, denoted $\Pi_2(\mathcal{E})$, and the protocol $\Pi_1(\mathcal{E})$ developed in Section 4 is that we are unable to provide client with the DFA itself; rather, we give it encryptions of the coefficients $\langle a_{\sigma i}\rangle_{\sigma\in\Sigma,i\in[n]} \leftarrow \mathsf{ToPoly}(Q,\Sigma,\delta)$. The implications of this to the protocol are far-reaching, however, due to the operations that the client needs to perform using these coefficients: both Shifting and especially combining coefficients with ciphertexts as was done in line c110 in Figure 1. For this reason, we need to expand the properties we require of the encryption system we use, to include the ability to homomorphically "multiply" ciphertexts *once*. We emphasize that we do *not* require fully homomorphic encryption. Our construction can be instantiated with any additively homomorphic encryption scheme that allows a single homomorphic multiplication of two ciphertexts [9, 21], provided that it also supports two-party decryption. Here we build from the more well-studied scheme of Boneh, Goh and Nissim [9], which we denote by BGN.

Specifically, BGN uses an algorithm BGNInit that, on input 1^κ, outputs $(p,\, p',\, \mathbb{G},\, \mathbb{G}',\, e)$ where p, p' are random $\kappa/2$-bit primes, $\mathbb{G}$ and $\mathbb{G}'$ are cyclic groups of order $N = pp'$, and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}'$ is a bilinear map. In this encryption scheme, the ring $\mathbb{R}$ is $\mathbb{Z}_N$, the ciphertext space $C_{\langle N,\mathbb{G},\mathbb{G}',e,g,h,\hat{g}\rangle}$ is $\mathbb{G} \cup \mathbb{G}'$, and the relevant algorithms are defined as follows. Note that we assume that elements of $\mathbb{G}$ and $\mathbb{G}'$ are encoded distinctly.

$\underline{\mathsf{Gen}(1^\kappa)}$: Generate $(p, p', \mathbb{G}, \mathbb{G}', e) \leftarrow \mathsf{BGNInit}(1^\kappa)$; select random generators $g, u \overset{\$}{\leftarrow} \mathbb{G}$; set $N \leftarrow pp'$, $h \leftarrow u^{p'}$, and $\hat{g} \leftarrow e(g,g)^p$; and return public key $\langle N, \mathbb{G}, \mathbb{G}', e, g, h, \hat{g}\rangle$ and private key $\langle N, \mathbb{G}, \mathbb{G}', e, g, \hat{g}, p\rangle$.

$\underline{\mathsf{Enc}_{\langle N,\mathbb{G},\mathbb{G}',e,g,h,\hat{g}\rangle}(m)}$: Select $x \overset{\$}{\leftarrow} \mathbb{Z}_N$ and return $g^m h^x$.

$\underline{\mathsf{Dec}_{\langle N,\mathbb{G},\mathbb{G}',e,g,\hat{g},p\rangle}(c)}$: If $c \in \mathbb{G}$, then return the discrete logarithm of $e(c,g)^p$ with respect to base $\hat{g}$. If $c \in \mathbb{G}'$, then return the discrete logarithm of c^p with respect to base $\hat{g}$.

$\underline{c_1 +_{\langle N,\mathbb{G},\mathbb{G}',e,g,h,\hat{g}\rangle} c_2}$: If c_1 and c_2 are in the same group (i.e., both are in $\mathbb{G}$ or both are in $\mathbb{G}'$), then return c_1c_2. Otherwise, if $c_1 \in \mathbb{G}$ and $c_2 \in \mathbb{G}'$, then return $e(c_1, g)c_2$.

$\underline{m \cdot_{\langle N,\mathbb{G},\mathbb{G}',e,g,h,\hat{g}\rangle} c}$: Return c^m.

$\underline{c_1 \odot_{\langle N,\mathbb{G},\mathbb{G}',e,g,h,\hat{g}\rangle} c_2}$: If $c_1, c_2 \in \mathbb{G}$, then return $e(c_1, c_2)$. Otherwise, return $\perp$.

$\underline{\mathsf{Share}(\langle N, \mathbb{G}, \mathbb{G}', e, g, \hat{g}, p\rangle)}$: Return $sk_1 = \langle \mathbb{G}, \mathbb{G}', d_1\rangle$ and $sk_2 = \langle \mathbb{G}, \mathbb{G}', e, g, \hat{g}, d_2\rangle$ where $d_1 \overset{\$}{\leftarrow} \mathbb{Z}_N$ and $d_2 \leftarrow p - d_1 \bmod N$.

$\underline{\mathsf{Dec}^1_{\langle \mathbb{G},\mathbb{G}',d_1\rangle}(c)}$: Return c^{d_1}.

$\underline{\mathsf{Dec}^2_{\langle \mathbb{G},\mathbb{G}',e,g,\hat{g},d_2\rangle}(c_1, c_2)}$: If $c_1, c_2 \in \mathbb{G}$, then return the discrete logarithm of $e(c_2c_1^{d_2}, g)$ with respect to base $\hat{g}$. If $c_1, c_2 \in \mathbb{G}'$, then return the discrete logarithm of $c_2c_1^{d_2}$ with respect to base $\hat{g}$.

Note the new operator $\odot_{pk}$ that homomorphically multiplies two ciphertexts in $\mathbb{G}$. Since the result is in $\mathbb{G}'$, it is not possible to use the result as an argument to $\odot_{pk}$. This is the sense in which this scheme permits homomorphic multiplication "once". Also note that though the basic scheme of Boneh et al. did not include $\hat{g} = e(g,g)^p$ in the public key, Boneh et al. proposed an extension supporting multiparty threshold decryption [9, Section 5] that did so[2]; it is this extension that we adopt here.

A complication of using BGN is the need to compute a discrete logarithm to decrypt in $\mathsf{Dec}_{\langle N,\mathbb{G},\mathbb{G}',e,g,\hat{g},p\rangle}$ and $\mathsf{Dec}^2_{\langle \mathbb{G},\mathbb{G}',e,g,\hat{g},d_2\rangle}$. We thus need to design our protocol so that any ciphertext that a party attempts to decrypt should hold a plaintext from a small range $0 \ldots L$. Then, Pollard's lambda method [34, p. 128] enables recovery of the plaintext in $O(\sqrt{L})$ time. Alternatively, a precomputed table that maps $\hat{g}^m$ to the plaintext $m \in \{0 \ldots L\}$ enables decryption to be performed by table lookup.

[2]The exact construction supporting threshold decryption was left implicit by Boneh et al. [9], but we have confirmed that including $\hat{g} = e(g,g)^p$ in the public key is what they intended [7].

Protocol steps Protocol $\Pi_2(\mathcal{E})$ is shown in Figure 5. It has a similar structure to $\Pi_1(\mathcal{E})$, but differs in many respects.

- Rather than taking the DFA $\langle Q, \Sigma, q_{\mathsf{init}}, \delta\rangle$ as input, the client takes $\alpha \leftarrow \mathsf{Enc}_{pk}(q_{\mathsf{init}})$ and encrypted coefficients $\langle \hat{a}_{\sigma i}\rangle_{\sigma\in\Sigma, i\in[n]}$ as input. Specifically, Figure 5 presumes that these coefficients are created by performing $\langle a_{\sigma i}\rangle_{\sigma\in\Sigma, i\in[n]} \leftarrow \mathsf{ToPoly}(Q, \Sigma, \delta)$ and then encrypting each $a_{\sigma i}$ using pk, i.e., $\hat{a}_{\sigma i} \leftarrow \mathsf{Enc}_{pk}(a_{\sigma i})$ for each $\sigma \in \Sigma$ and $i \in [n]$.
- Because server decrypts the (blinded) DFA state in line s204, the plaintext should be adequately small so that decryption — which as discussed above, involves computing (or looking up) a discrete logarithm if BGN encryption is in use — is not too costly. For this reason, and assuming $\mathbb{R} = \mathbb{Z}_N$ (as it is in BGN) and $Q = [n]$, the blinding term is drawn from $\{0,1\}^{\kappa'}$ instead of $\mathbb{R}$, where $\kappa' \ll \kappa$ is another security parameter. Then, the statistical distance between the distribution of γ seen by server in line s204 when the blinded state is q (i.e., when $\gamma = q +_{\mathbb{R}} r$) and uniformly random choices from $\{0,1\}^{\kappa'}$ is

$$\sum_x \left| \begin{array}{l} \mathbb{P}(q + r = x \mid r \overset{\$}{\leftarrow} \{0,1\}^{\kappa'}) \\ \quad -\mathbb{P}(r = x \mid r \overset{\$}{\leftarrow} \{0,1\}^{\kappa'}) \end{array} \right|$$
$$= \sum_{0\le x<q} \frac{1}{2^{\kappa'}} + \sum_{2^{\kappa'} \le x < q+2^{\kappa'}} \frac{1}{2^{\kappa'}} = \frac{q}{2^{\kappa'-1}}$$

  Since $q \in [n]$ and since the server sees $\ell + 1$ "samples" of γ in a protocol execution, we anticipate setting $\kappa' \approx \log_2 n + \log_2 \ell + 20$ to achieve a reasonable balance between decryption cost and security for moderately sized n and ℓ (e.g., $\ell n < 2^{20}$). It is important to note, however, that generally κ' will need to grow with n and ℓ (though only logarithmically so).
- The fact that each $\hat{a}_{\sigma i}$ is a ciphertext imposes several changes to the protocol. First, it is necessary to employ the "one-time multiplication" operator $\odot_{pk}$ in line c207 to produce the ciphertext of the new state, versus $\cdot_{pk}$ as in line c110. Second, the Shift operation must be altered to work on ciphertexts, i.e., changing (4) to

$$\hat{a}'_{\sigma i} \leftarrow \sum_{i'=0}^{n-1-i}{}_{pk} \left(\binom{i+i'}{i'} \cdot_{\mathbb{R}} (-_{\mathbb{R}} r)^{i'} \right) \cdot_{pk} \hat{a}_{\sigma(i+i')} \quad (8)$$

5.2 Security

The theorems in Section 4.2 that pertain to security against server adversaries for $\Pi_1(\mathcal{E})$ carry over more-or-less directly for $\Pi_2(\mathcal{E})$. That is, if we adapt the definitions of $\mathbf{Expt}^{\texttt{s-dfa}}_{\Pi_1(\mathcal{E})}$ and $\mathbf{Expt}^{\texttt{s-file}}_{\Pi_1(\mathcal{E})}$ in the natural way to $\Pi_2(\mathcal{E})$ —

```
client($pk, sk_1, pk', \Sigma, n,$                      server($pk, sk_2, \Sigma,$
       $\alpha, \langle \hat{a}_{\sigma i} \rangle_{\sigma \in \Sigma, i \in [n]}$)          $\langle c_{kj} \rangle_{k \in [\ell], j \in [m]}$)

                                              s201. $m \leftarrow |\Sigma|$
                                              s202. $\langle \lambda_{\sigma j} \rangle_{\sigma \in \Sigma, j \in [m]}$
                                                        $\leftarrow$ Lagrange($\Sigma$)
                    m201. $\xrightarrow{\quad n \quad}$
                    m202. $\xleftarrow{\quad \ell \quad}$
c201. for $k \leftarrow 0 \ldots \ell - 1$                  s203. for $k \leftarrow 0 \ldots \ell - 1$
c202.   $r \overset{\$}{\leftarrow} \{0,1\}^{\kappa'}$
c203.   $\alpha \leftarrow \alpha +_{pk} \mathsf{Enc}_{pk}(r)$
c204.   $\beta \leftarrow \mathsf{Dec}^1_{sk_1}(\alpha)$
c205.   $\rho \leftarrow \mathsf{Enc}_{pk'}(r)$
                    m203. $\xrightarrow{\quad \alpha,\beta,\rho \quad}$
                                              s204.   $\gamma \leftarrow \mathsf{Dec}^2_{sk_2}(\alpha, \beta)$
                                              s205.   for $\sigma \in \Sigma$
                                              s206.     $\Psi_\sigma \leftarrow \sum^{pk}_{j=0}{}^{m-1} \lambda_{\sigma j} \cdot_{pk} c_{kj}$
c206.   $\langle \hat{a}'_{\sigma i} \rangle_{\sigma \in \Sigma, i \in [n]}$          s207.     for $i \in [n]$
          $\leftarrow$ Shift($r, \langle \hat{a}_{\sigma i} \rangle_{\sigma \in \Sigma, i \in [n]}$) s208.       $\mu_{\sigma i} \leftarrow \gamma^i \cdot_{pk} \Psi_\sigma$
                                              s209.     endfor
                                              s210.   endfor
                    m204. $\xleftarrow{\quad \langle \mu_{\sigma i} \rangle_{\sigma \in \Sigma, i \in [n]} \quad}$
c207.   $\alpha \leftarrow \sum^{pk}_{\sigma \in \Sigma} \sum^{pk}_{i=0}{}^{n-1} a'_{\sigma i} \odot_{pk} \mu_{\sigma i}$
c208. endfor                                  s211. endfor
c209. $r \overset{\$}{\leftarrow} \{0,1\}^{\kappa'}$
c210. $\alpha \leftarrow \alpha +_{pk} \mathsf{Enc}_{pk}(r)$
c211. $\beta \leftarrow \mathsf{Dec}^1_{sk_1}(\alpha)$
c212. $\rho \leftarrow \mathsf{Enc}_{pk'}(r)$
                    m205. $\xrightarrow{\quad \alpha,\beta,\rho \quad}$
                                              s212. $\gamma^* \leftarrow \mathsf{Dec}^2_{sk_2}(\alpha, \beta)$
                    m206. $\xleftarrow{\quad \gamma^* \quad}$
c213. return $\gamma^* -_{\mathbb{R}} r$
```

Figure 5. Protocol $\Pi_2(\mathcal{E})$, described in Section 5

in particular, to modify the behavior of clientOr to conform to $\Pi_2(\mathcal{E})$ — then ignoring terms negligible in κ and κ', we can prove:

Theorem 4. *For* $t' = t + t_{\mathsf{Gen}} + t_{\mathsf{Share}}$,

$$\mathbf{Adv}^{\texttt{s-dfa}}_{\Pi_2(\mathcal{E})}(t, \ell, n, m) \le 2\mathbf{Adv}^{\texttt{ind-cpa}}_{\mathcal{E}}(t', \ell + 1)$$

Theorem 5. *For* $t' = t + t_{\mathsf{Gen}} + t_{\mathsf{Share}}$,

$$\mathbf{Adv}^{\texttt{s-file}}_{\Pi_2(\mathsf{BGN})}(t, \ell, n, m) \\ \le 2\mathbf{Adv}^{\texttt{ind-cpa}}_{\mathsf{BGN}}(t', \ell + 1) + \mathbf{Adv}^{\texttt{ind-cpa}}_{\mathsf{BGN}}(t', \ell m)$$

The proofs of these theorems are analogous their counterparts for $\Pi_1(\mathcal{E})$, and so we omit them here.

The case of client adversaries in $\Pi_2(\mathcal{E})$ differs more substantially from that in $\Pi_1(\mathcal{E})$. For one, we need to formalize and prove a result about the degree to which the DFA is protected from the client. Such an experiment for defining this type of security is shown in Figure 6. This experiment is analogous to that of Figure 4, though is longer due to the need to prepare the input arguments to C_2. In this experiment, C_2 is invoked with encrypted coefficients $\langle \hat{a}_{\sigma i} \rangle_{\sigma \in \Sigma, i \in [n]}$ and the encrypted initial state α for one of two DFAs output by C_1 (determined by random selection of b). C_2 can invoke serverOr first with an integer n (as in m201), in response to which serverOr returns ℓ (as in m202). C_2 can then invoke serverOr ℓ times, each time with ciphertexts α, β, ρ (as in m203), and receive ciphertexts $\langle \mu_{\sigma i} \rangle_{\sigma \in \Sigma, i \in [n]}$ in response (as in m204). After this, C_2 can invoke serverOr with ciphertexts α, β, ρ and receive a value γ^* in response. Finally, C_2 outputs a bit b', and $\mathbf{Expt}^{\texttt{c-dfa}}_{\Pi_2(\mathcal{E})}(C) = 1$ only if $b' = b$. As usual, for any honest-but-curious C we define

$$\mathbf{hbcAdv}^{\texttt{c-dfa}}_{\Pi_2(\mathcal{E})}(C) = 2 \cdot \mathbb{P}\left(\mathbf{Expt}^{\texttt{c-dfa}}_{\Pi_2(\mathcal{E})}(C) = 1\right) - 1$$

and then $\mathbf{hbcAdv}^{\texttt{c-dfa}}_{\Pi_2(\mathcal{E})}(t, \ell, n, m) = \max_C \mathbf{hbcAdv}^{\texttt{c-dfa}}_{\Pi_2(\mathcal{E})}(C)$ where the advantage is taken over all honest-but-curious adversaries C taking time t, producing a file of ℓ symbols, and producing DFAs with n states and an alphabet of m symbols.

```
Experiment $\mathbf{Expt}^{\texttt{c-dfa}}_{\Pi_2(\mathcal{E})}(C_1, C_2)$
  $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$
  $(sk_1, sk_2) \leftarrow \mathsf{Share}(sk)$
  $(pk', sk') \leftarrow \mathsf{Gen}(1^{\kappa+2})$
  $(\ell, \langle \sigma_k \rangle_{k \in [\ell]}, M_0, M_1, \phi) \leftarrow C_1(pk, sk_1, pk')$
  if $M_0.Q \ne M_1.Q$ or $M_0.\Sigma \ne M_1.\Sigma$
     or $M_0(\langle \sigma_k \rangle_{k \in [\ell]}) \ne M_1(\langle \sigma_k \rangle_{k \in [\ell]})$ then return 0
  $b \overset{\$}{\leftarrow} \{0,1\}$
  $\langle Q, \Sigma, \delta, q_{\mathsf{init}} \rangle \leftarrow M_b$
  $n \leftarrow |Q|, m \leftarrow |\Sigma|$
  for $k \in [\ell], j \in [m]$
    $c_{kj} \leftarrow \mathsf{Enc}_{pk}((\sigma_k)^j)$
  $\langle a_{\sigma i} \rangle_{\sigma \in \Sigma, i \in [n]} \leftarrow \mathsf{ToPoly}(Q, \Sigma, \delta)$
  for $\sigma \in \Sigma, i \in [n]$
    $\hat{a}_{\sigma i} \leftarrow \mathsf{Enc}_{pk}(a_{\sigma i})$
  $\alpha \leftarrow \mathsf{Enc}_{pk}(q_{\mathsf{init}})$
  $b' \leftarrow C_2^{\mathsf{serverOr}(pk, sk_2, \langle c_{kj} \rangle_{k \in [\ell], j \in [m]})}(\phi, \alpha, \langle \hat{a}_{\sigma i} \rangle_{\sigma \in \Sigma, i \in [n]})$
  if $b' = b$
    then return 1
    else return 0
```

Figure 6. Experiment $\mathrm{Expt}^{\texttt{c-dfa}}_{\Pi_2(\mathcal{E})}(C_1, C_2)$

Theorem 6. *For* $t' = t + t_{\mathsf{Gen}} + (\ell + 1) \cdot t_{\mathsf{Dec}}$,

$$\mathbf{hbcAdv}^{\texttt{c-dfa}}_{\Pi_2(\mathsf{BGN})}(t, \ell, n, m) \le \mathbf{Adv}^{\texttt{ind-cpa}}_{\mathsf{BGN}}(t', (\ell + 1)nm)$$

Proof. Given an honest-but-curious adversary C for $\Pi_2(\mathsf{BGN})$ that runs in time t, produces a file of length

ℓ, and produces DFAs of n states over an alphabet of m symbols, we construct an IND-CPA attacker U for $\mathcal{E}$ to demonstrate the theorem as follows. On input $\hat{pk} = \langle N, \mathbb{G}, \mathbb{G}', e, g, h, \hat{g}\rangle$, U generates $(pk', sk') \leftarrow \mathsf{Gen}(1^{\kappa+2})$ and $d_1 \overset{\$}{\leftarrow} \mathbb{Z}_N$, and invokes $C_1(\hat{pk}, sk_1, pk')$ where $sk_1 = \langle \mathbb{G}, \mathbb{G}', d_1\rangle$ to obtain $(\ell, \langle\sigma_k\rangle_{k\in[\ell]}, M_0, M_1, \phi)$. Note that d_1 is chosen from a distribution that is perfectly indistinguishable from that from which d_1 is chosen in the real system. If $M_0.Q \neq M_1.Q$, $M_0.\Sigma \neq M_1.\Sigma$, or $M_0(\langle\sigma_k\rangle_{k\in[\ell]}) \neq M_1(\langle\sigma_k\rangle_{k\in[\ell]})$, then U aborts the simulation, since $\mathbf{Expt}^{\texttt{c-dfa}}_{\Pi_2(\mathcal{E})}(C) = 0$ in this case. Otherwise, letting $\Sigma = M_0.\Sigma$, $Q = M_0.Q$, $m = |\Sigma|$ and $n = |Q|$, U computes $\langle a_{0\sigma i}\rangle_{\sigma\in\Sigma, i\in[n]} \leftarrow \mathsf{ToPoly}(Q, \Sigma, M_0.\delta)$ and $\langle a_{1\sigma i}\rangle_{\sigma\in\Sigma, i\in[n]} \leftarrow \mathsf{ToPoly}(Q, \Sigma, M_1.\delta)$, and then sets $\hat{a}_{\sigma i} \leftarrow \mathsf{Enc}^{\hat{b}}_{\hat{pk}}(a_{0\sigma i}, a_{1\sigma i})$ for $\sigma \in \Sigma$ and $i \in [n]$.

U then invokes $C_2(\phi, \alpha, \langle\hat{a}_{\sigma i}\rangle_{\sigma\in\Sigma, i\in[n]})$ and simulates responses to C_2's queries to serverOr as follows (ignoring malformed invocations). In response to the initial query n, the adversary U returns ℓ and, in preparation for the subsequent serverOr invocations by C_2, sets $q_0 \leftarrow M_0.q_{\mathsf{init}}$, and $q_1 \leftarrow M_1.q_{\mathsf{init}}$. For the k-th query of the form α, β, ρ $(0 \leq k < \ell)$, the adversary U sets $r \leftarrow \mathsf{Dec}_{sk'}(\rho)$, $\gamma_0 \leftarrow q_0 +_{\mathbb{R}} r$, and $\gamma_1 \leftarrow q_1 +_{\mathbb{R}} r$; sets $\mu_{\sigma i} \leftarrow \mathsf{Enc}^{\hat{b}}_{\hat{pk}}(((\gamma_0)^i \cdot_{\mathbb{R}} \Lambda_\sigma(\sigma_k)), ((\gamma_1)^i \cdot_{\mathbb{R}} \Lambda_\sigma(\sigma_k)))$ for $\sigma \in \Sigma$ and $i \in [n]$; updates $q_0 \leftarrow M_0.\delta(q_0, \sigma_k)$ and $q_1 \leftarrow M_1.\delta(q_1, \sigma_k)$; and returns $\langle\mu_{\sigma i}\rangle_{\sigma\in\Sigma, k\in[n]}$ to C_2. After ℓ such invocations, U responds to the next invocation α, β, ρ by computing $r \leftarrow \mathsf{Dec}_{sk'}(\rho)$ and returning $q_0 +_{\mathbb{R}} r$ $(= q_1 +_{\mathbb{R}} r)$. Finally, when C_2 outputs b', U outputs b', as well.

U's simulation is perfectly indistinguishable from the real system to an honest-but-curious adversary C, and so $\mathbf{Adv}^{\texttt{ind-cpa}}_{\mathsf{BGN}}(U) = \mathbf{hbcAdv}^{\texttt{c-dfa}}_{\Pi_2(\mathsf{BGN})}(C)$. Note that U runs in $t' = t + t_{\mathsf{Gen}} + (\ell+1)\cdot t_{\mathsf{Dec}}$, where t_{Gen} is incurred to generate (pk', sk'). U makes nm oracle queries in order to respond to each oracle query except the first and last, of which there are ℓ, and makes nm additional oracle queries to create $\langle\hat{a}_{\sigma i}\rangle_{\sigma\in\Sigma, i\in[n]}$. □

The second way in which the argument for security against client adversaries for $\Pi_2(\mathcal{E})$ differs from that for $\Pi_1(\mathcal{E})$ is that the argument for file security needs to be adapted accordingly. If we again alter $\mathbf{Expt}^{\texttt{c-file}}_{\Pi_1(\mathcal{E})}$ in the natural way to produce $\mathbf{Expt}^{\texttt{c-file}}_{\Pi_2(\mathcal{E})}$, then it is straightforward to prove the following theorem:

Theorem 7. *For* $t' = t + t_{\mathsf{Gen}} + (\ell+1)\cdot t_{\mathsf{Dec}}$,

$$\mathbf{hbcAdv}^{\texttt{c-file}}_{\Pi_2(\mathcal{E})}(t, \ell, n, m) \leq \mathbf{Adv}^{\texttt{ind-cpa}}_{\mathsf{BGN}}(t', \ell m(1+n))$$

6 A More Communication-Efficient One-Sided Protocol

The last protocol we present has the same goals as $\Pi_1(\mathcal{E})$ — i.e., it is "one-sided" in the sense that DFA privacy is offered against only server adversaries — but incurs less communication costs. Specifically, whereas the communication cost of $\Pi_1(\mathcal{E})$ is dominated by sending $(nm+3)\ell+3$ ciphertexts, the protocol we present in this section, called $\Pi_3(\mathcal{E})$, sends only $(n+m+1)\ell+3$ ciphertexts. $\Pi_3(\mathcal{E})$ accomplishes this in part by exploiting a cryptosystem that is additively homomorphic and that offers the ability to homomorphically "multiply" ciphertexts once, as we employed for $\Pi_2(\mathcal{E})$ in Section 5. BGN is thus an appropriate encryption scheme to instantiate $\mathcal{E}$, and this observation will again inform our design.

Protocol $\Pi_3(\mathcal{E})$ is shown in Figure 7. Note that the input arguments to both the client and the server are identical to those in $\Pi_1(\mathcal{E})$, as are the client and server operations before (c301–c303, s301–s302, and m301–m302) their main loops (c304–c314, s303–s313). Moreover, the beginning (c305–c308, m303) and end (c313) of the client's main loop, and the steps following the client and server main loops (c315–c319, s314, m305–m306), are inherited directly from $\Pi_2(\mathcal{E})$. The primary novelties of $\Pi_3(\mathcal{E})$ can be summarized as follows:

- As in $\Pi_2(\mathcal{E})$, after the k-th invocation of the form m303, the server constructs Ψ_σ to be an encryption of $\Lambda_\sigma(\sigma_k)$ (s306). Rather than constructing each $\mu_{\sigma i} \leftarrow \gamma^i \cdot_{pk} \Psi_\sigma$, however, the server sends $\langle\Psi_\sigma\rangle_{\sigma\in\Sigma}$ to the client in message m304. Each $\mu_{\sigma i}$ is then constructed at the client, instead (c310–c312).
- Because each $\mu_{\sigma i}$ is constructed at the client, it is necessary for the server to send γ to the client (m304). So as to not divulge the current DFA state to the client — recall that the client possesses the blinding term r, see c305 — the server blinds γ again with a random r' (s308–s309) before returning it.
- A consequence of this additional blinding by server is that the client's Shift operation must be adapted to account for it. To enable this, the server also sends in m304 the ciphertext ν_i of $(r')^i$, for each $i \in [n]$ (see s311). The client can then construct a ciphertext s_i of $(r +_{\mathbb{R}} r')^i$ for each $i \in [n]$, using the binomial theorem:

$$s_i \leftarrow \sum_{i'=0}^{i}{}_{pk} \left(\binom{i}{i'} \cdot_{\mathbb{R}} r^{i'} \right) \cdot_{pk} \nu_{i-i'} \qquad (9)$$

The client can then calculate a ciphertext $\hat{a}'_{\sigma i}$ of the coefficient of x^i in f'_σ:

$$\hat{a}'_{\sigma i} \leftarrow \sum_{i'=0}^{n-1-i}{}^{pk} \left(a_{\sigma(i+i')} \cdot_{\mathbb{R}} \binom{i+i'}{i'} \cdot_{\mathbb{R}} (-_{\mathbb{R}}1)^{i'} \right) \cdot_{pk} s_{i'} \tag{10}$$

In our pseudocode, the calculations (9) and (10) are encapsulated within the operation $\langle \hat{a}'_{\sigma i} \rangle_{\sigma\in\Sigma, i\in[n]} \leftarrow$ Shift$(r, \langle \nu_i \rangle_{i\in[n]}, \langle a_{\sigma i} \rangle_{\sigma\in\Sigma, i\in[n]})$ on line c309.

The privacy of the file and DFA from server adversaries and the privacy of the file from client adversaries can be proved for $\Pi_3(\mathcal{E})$ very similarly to how they are proved for $\Pi_1(\mathcal{E})$. In fact, Theorems 1–3 hold for $\Pi_3(\mathcal{E})$ unchanged except for replacing Pai with BGN where appropriate.

7 Conclusion

With the growth of cloud storage due to the cost savings it offers, it is imperative that we develop efficient techniques for enabling the same sorts of third-party access to cloud-resident files that is commonplace today for privately stored files — e.g., malware scans or searches by authorized partners. The fact that cloud-resident files are generally at greater risk of exposure, however, mandates their encryption, hindering these sorts of third-party access.

In this paper, we have developed a family of protocols for enabling DFA evaluation on encrypted files by third parties authorized by the file owner. Our *one-sided* protocols provably protect the privacy of the DFA from an arbitrarily malicious server holding the ciphertext file, as well as the privacy of the file from the server and from an honest-but-curious client performing the DFA evaluation. We also presented a *two-sided* protocol that additionally protects the privacy of the DFA from the client, enabling others to outsource their DFA evaluations to it without divulging their DFAs. Our protocols employ additively homomorphic cryptosystems or small extensions thereof, for which practical implementations exist. The costs of our protocols in terms of storage, communication and computation suggest that they are sufficiently practical for many domains, particularly ones where files consist of symbols from a limited alphabet, and are more practical than protocols that would result from the application of general private two-party computation or fully homomorphic encryption to this problem.

Acknowledgements

We are grateful to Dan Boneh for helpful clarifications about the Boneh-Goh-Nissim cryptosystem [9]. This research was supported in part by NSF award 0910483 and by a gift from NEC.

```
client($pk, sk_1, pk', \langle Q, \Sigma, \delta, q_{\mathsf{init}} \rangle$)          server($pk, sk_2, \Sigma, \langle c_{kj} \rangle_{k\in[\ell], j\in[m]}$)

c301. $n \leftarrow |Q|, m \leftarrow |\Sigma|$                        s301. $m \leftarrow |\Sigma|$
c302. $\langle a_{\sigma i} \rangle_{\sigma\in\Sigma, i\in[n]}$                   s302. $\langle \lambda_{\sigma j} \rangle_{\sigma\in\Sigma, j\in[m]}$
        $\leftarrow$ ToPoly$(Q, \Sigma, \delta)$                          $\leftarrow$ Lagrange$(\Sigma)$
c303. $\alpha \leftarrow$ Enc$_{pk}(q_{\mathsf{init}})$
                  m301. $\xrightarrow{n}$
                  m302. $\xleftarrow{\ell}$
c304. for $k \leftarrow 0 \ldots \ell - 1$                 s303. for $k \leftarrow 0 \ldots \ell - 1$
c305.     $r \overset{\$}{\leftarrow} \{0,1\}^{\kappa'}$
c306.     $\alpha \leftarrow \alpha +_{pk}$ Enc$_{pk}(r)$
c307.     $\beta \leftarrow$ Dec$^1_{sk_1}(\alpha)$
c308.     $\rho \leftarrow$ Enc$_{pk'}(r)$
                  m303. $\xrightarrow{\alpha,\beta,\rho}$
                                                  s304.     $\gamma \leftarrow$ Dec$^2_{sk_2}(\alpha, \beta)$
                                                  s305.     for $\sigma \in \Sigma$
                                                  s306.         $\Psi_\sigma \leftarrow \sum_{j=0}^{m-1}{}^{pk} \lambda_{\sigma j} \cdot_{pk} c_{kj}$
                                                  s307.     endfor
                                                  s308.     $r' \overset{\$}{\leftarrow} \mathbb{R}$
                                                  s309.     $\gamma \leftarrow \gamma +_{\mathbb{R}} r'$
                                                  s310.     for $i \in [n]$
                                                  s311.         $\nu_i \leftarrow$ Enc$_{pk}((r')^i)$
                                                  s312.     endfor
                  m304. $\xleftarrow{\gamma, \langle \Psi_\sigma \rangle_{\sigma\in\Sigma}, \langle \nu_i \rangle_{i\in[n]}}$
c309.     $\langle \hat{a}'_{\sigma i} \rangle_{\sigma\in\Sigma, i\in[n]}$
            $\leftarrow$ Shift$(r, \langle \nu_i \rangle_{i\in[n]},$
                    $\langle a_{\sigma i} \rangle_{\sigma\in\Sigma, i\in[n]})$
c310.     for $\sigma \in \Sigma, i \in [n]$
c311.         $\mu_{\sigma i} \leftarrow \gamma^i \cdot_{pk} \Psi_\sigma$
c312.     endfor
c313.     $\alpha \leftarrow \sum_{\sigma\in\Sigma}{}^{pk} \sum_{i=0}^{n-1}{}^{pk} \hat{a}'_{\sigma i} \odot_{pk} \mu_{\sigma i}$
c314. endfor                                      s313. endfor
c315. $r \overset{\$}{\leftarrow} \{0,1\}^{\kappa'}$
c316. $\alpha \leftarrow \alpha +_{pk}$ Enc$_{pk}(r)$
c317. $\beta \leftarrow$ Dec$^1_{sk_1}(\alpha)$
c318. $\rho \leftarrow$ Enc$_{pk'}(r)$
                  m305. $\xrightarrow{\alpha,\beta,\rho}$
                                                  s314. $\gamma^* \leftarrow$ Dec$^2_{sk_2}(\alpha, \beta)$
                  m306. $\xleftarrow{\gamma^*}$
c319. return $\gamma^* -_{\mathbb{R}} r$
```

Figure 7. Protocol $\Pi_3(\mathcal{E})$, described in Section 6

References

[1] GenBank. `http://www.ncbi.nlm.nih.gov/genbank/`.

[2] United Kingdom National DNA Database. `http://www.npia.police.uk/en/8934.htm`.

[3] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption

schemes. In *Advances in Cryptology – CRYPTO '98*, pages 26–45, Aug. 1998.

[4] A. Ben-David, N. Nisan, and B. Pinkas. FairplayMP: A system for secure multi-party computation. In *15th ACM Conference on Computer and Communications Security*, pages 257–266, 2008.

[5] D. Betel and C. Hogue. Kangaroo – a pattern-matching program for biological sequences. *BMC Bioinformatics*, 3, 2002.

[6] M. Blanton and M. Aliasgari. Secure outsourcing of DNA searching via finite automata. In *Data and Applications Security and Privacy XXIV*, pages 49–64, June 2010.

[7] D. Boneh. Personal communication, July 2011.

[8] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Advances in Cryptology – EUROCRYPT 2004*, pages 506–522, 2004.

[9] D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In *2nd Theory of Cryptography Conference*, pages 325–342, 2005.

[10] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *4th Theory of Cryptography Conference*, pages 535–554, Feb. 2007.

[11] Y.-C. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *Applied Cryptography and Network Security, 3rd International Conference*, pages 442–455, 2005.

[12] K. Chen, R. Kavuluru, and S. Guo. RASP: Efficient multidimensional range query on attack-resilient encrypted databases. In *1st ACM Conference on Data and Application Security and Privacy*, Feb. 2011.

[13] S. G. Choi, A. Elbaz, A. Juels, T. Malkin, and M. Yung. Two-party computing with encrypted data. In *Advances in Crypotology – ASIACRYPT 2007*, pages 298–314, 2007.

[14] V. Ciriani, S. D. C. D. Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Combining fragmentation and encryption to protect privacy in data storage. *ACM Transactions on Information and System Security*, 13(3), July 2010.

[15] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. In *13th ACM Conference on Computer and Communications Security*, pages 79–88, 2006.

[16] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *Public Key Cryptography, 4th International Workshop on Practice and Theory in Public Key Cryptosystems*, pages 119–136, Feb. 2001.

[17] K. B. Frikken. Practical private DNA string searching and matching through efficient oblivious automata evaluation. In *Data and Applications Security XXIII*, pages 81–94, July 2009.

[18] R. Gennaro, C. Hazay, and J. S. Sorensen. Text search protocols with simulation based security. In *Public Key Cryptography – PKC 2010*, pages 332–350, 2010.

[19] C. Gentry. Fully homomorphic encryption using ideal lattices. In *41st ACM Symposium on Theory of Computing*, pages 169–178, 2009.

[20] C. Gentry and S. Halevi. Implementing Gentry's fully-homomorphic encryption scheme. In *Advances in Cryptology – EUROCRYPT 2011*, May 2011.

[21] C. Gentry, S. Halevi, and V. Vaikuntanathan. A simple BGN-type cryptosystem from LWE. In *Advances in Cryptology – EUROCRYPT 2010*, pages 506–522, May 2010.

[22] C. Gentry and Z. Ramzan. Single-database private information retrieval with constant communication rate. In *Automata, Languages and Programming, 32nd International Colloquium*, pages 803–815, July 2005.

[23] E.-J. Goh. Secure indexes. Cryptology ePrint Archive, Report 2003/216, 2003. `http://eprint.iacr.org/`.

[24] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *19th ACM Symposium on Theory of Computing*, pages 218–229, 1987.

[25] H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra. Executing SQL over encrypted data in the database-service-provider model. In *2002 ACM SIGMOD International Conference on Management of Data*, pages 216–227, June 2002.

[26] C. Hazay and Y. Lindell. Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. *Journal of Cryptology*, 23(3):422–456, 2010.

[27] C. Hazay and T. Toft. Computationally secure pattern matching in the presence of malicious adversaries. In *Advances in Cryptology – ASIACRYPT 2010*, pages 195–212, 2010.

[28] B. Hore, S. Mehrotra, and G. Tsudik. A privacy-preserving index for range queries. In *30th International Conference on Very Large Data Bases*, pages 720–731, Aug. 2004.

[29] S. Jha, L. Kruger, and V. Shmatikov. Towards practical privacy for genomic computation. In *29th IEEE Symposium on Security and Privacy*, pages 216–230, 2008.

[30] J. Katz and L. Malka. Secure text processing with applications to private DNA matching. In *17th ACM Conference on Computer and Communications Security*, pages 485–492, 2010.

[31] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Advances in Cryptology – EUROCRYPT 2008*, pages 146–162, Apr. 2008.

[32] T. Kojm. *ClamAV*. `http://www.clamav.net`.

[33] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay – a secure two-party computation system. In *13th USENIX Security Symposium*, pages 287–302, Aug. 2004.

[34] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[35] D. Needle. Cloud storage poised to save enterprises money: Report. `http://itmanagement.earthweb.com/datbus/article.php/3896116/Cloud-Storage-Poised-to-Save-Enterprises-Money-Report.htm`, July 30, 2010.

[36] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – EUROCRYPT '99*, pages 223–238, May 1999.

[37] B. Pinkas, T. Schneider, N. Smart, and S. Williams. Secure two-party computation is practical. In *Advances in Cryptology – ASIACRYPT 2009*, pages 250–267, Dec. 2009.

[38] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, pages 169–177, 1978.

[39] M. Roesch. Snort – lightweight intrusion detection for networks. In *13th USENIX Conference on System Administration*, pages 229–238, 1999.

[40] E. Shi, J. Bethencourt, T.-H. Chan, D. Song, and A. Perrig. Multi-dimensional range query over encrypted data. In *2007 IEEE Symposium on Security and Privacy*, pages 350–364, May 2007.

[41] N. P. Smart and F. Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Public Key Cryptography – PKC 2010*, pages 420–443, May 2010.

[42] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *2000 IEEE Symposium on Security and Privacy*, 2000.

[43] D. Stehlé and R. Steinfeld. Faster fully homomorphic encryption. In *Advances in Cryptology – ASIACRYPT 2010*, pages 377–394, Dec. 2010.

[44] J. R. Troncoso-Pastoriza, S. Katzenbeisser, and M. Celik. Privacy preserving error resilient DNA searching through oblivious automata. In *14th ACM Conference on Computer and Communications Security*, pages 519–528, 2007.

[45] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology – EUROCRYPT 2010*, pages 24–43, 2010.

[46] A. C. Yao. Protocols for secure computations. In *23rd IEEE Symposium on Foundations of Computer Science*, pages 160–164, 1982.

[47] J. Zhuge, T. Holz, C. Song, J. Guo, X. Han, and W. Zou. Studying malicious websites and the underground economy on the Chinese web. In *Proceedings of the Workshop on the Economics of Information Security*, June 2008.