PRShare: A Framework for Privacy-preserving, Interorganizational Data Sharing

LIHI IDAN and JOAN FEIGENBAUM, Yale University, USA

We consider the task of interorganizational data sharing, in which data owners, data clients, and data subjects have different and sometimes competing privacy concerns. One real-world scenario in which this problem arises concerns law-enforcement use of phone-call metadata: The data owner is a phone company, the data clients are law-enforcement agencies, and the data subjects are individuals who make phone calls. A key challenge in this type of scenario is that each organization uses its own set of proprietary intraorganizational attributes to describe the shared data; such attributes cannot be shared with other organizations. Moreover, data-access policies are determined by multiple parties and may be specified using attributes that are not directly comparable with the ones used by the owner to specify the data.

We propose a system architecture and a suite of protocols that facilitate dynamic and efficient interorganizational data sharing, while allowing each party to use its own set of proprietary attributes to describe the shared data and preserving the confidentiality of both data records and proprietary intraorganizational attributes. We introduce the novel technique of *Attribute-Based Encryption with Oblivious Attribute Translation (OTABE)*, which plays a crucial role in our solution. This extension of attribute-based encryption uses semi-trusted proxies to enable dynamic and oblivious translation between proprietary attributes that belong to different organizations; it supports hidden access policies, direct revocation, and fine-grained, data-centric keys and queries. We prove that our OTABE-based framework is secure in the standard model and provide two real-world use cases.

CCS Concepts: • **Security and privacy** → **Access control**; **Privacy-preserving protocols**; • **Theory of computation** → **Cryptographic protocols;**

Additional Key Words and Phrases: Attribute-based encryption, privacy-preserving data sharing

ACM Reference format:
Lihi Idan and Joan Feigenbaum. 2022. PRShare: A Framework for Privacy-preserving, Interorganizational Data Sharing. *ACM Trans. Priv. Secur.* 25, 4, Article 29 (July 2022), 38 pages.
https://doi.org/10.1145/3531225

1 INTRODUCTION

As the amount, complexity, and value of data available in both private and public sectors has risen sharply, data management and access control have challenged many organizations. Even more

This work was presented in preliminary form in [IF-WPES2020].
The first author was supported in part by US National Science Foundation grants CNS-1407454 and CNS-1409599 and William and Flora Hewlett Foundation grant 2016-3834. The second author was supported in part by US National Science Foundation grants CNS-1407454 and CNS-1409599 and US Office of Naval Research grant N00014-18-1-2743.
Authors' address: L. Idan and J. Feigenbaum, Computer Science Department, Yale University, PO Box 208285, New Haven, CT, 06520-8285 USA; emails: {lihi.idan, joan.feigenbaum}@yale.edu.

2471-2566/2022/07-ART29 $15.00
https://doi.org/10.1145/3531225

challenging are management and access control in *interorganizational* data sharing. Each organization would like to minimize the amount of sensitive information disclosed to other organizations, including information about the data as well as the organization's work methodologies and role structure.

1.1 Problem Description

We consider scenarios in which multiple organizations need to share data while each organization uses its own set of *proprietary metadata* to describe the shared data. In these scenarios, data records contain a *payload*, which is the actual data, and a set of *metadata attributes* that describe the payload. Although organizations may agree to share the payload, each uses a different set of metadata attributes taken from its own professional domain to describe this payload. Data must be shared in a controlled manner that protects the confidentiality of each organization's proprietary attributes *and* prevents unauthorized users from accessing the payload.

Typically, one organization, the *data owner*, maintains a set of data records that are potentially useful to other organizations, called the *data clients*. Each data record contains sensitive information about an individual, the *data subject*. *Data users*, who are employees of a data client, may need access to data records stored by the data owner to perform their assigned tasks. Each user must have the proper authorization to access the payloads of the specific set of records needed for a given task. Our framework also features a third type of organization, a *data intermediary*, which enriches data with additional information that is needed for the client's tasks but is available only to the intermediary. Each organization ORG_i maintains its own vocabulary VOC_i that contains the overall set of domain-specific, intraorganizational attributes used in its operations. VOC_i includes both sensitive, proprietary attributes and attributes that can be shared with other organizations. ORG_i uses a different set of attributes, $ATT_{i,j} \subseteq VOC_i$, to describe each shared payload p_j.

For example, the data owner may be an **email service provider (ESP)**. The data records represent email messages. Each email record is composed of a payload, which is the content of the email message, and metadata attributes about the payload such as sender, receiver, and date. Some attributes are sensitive, e.g., the email message's receiver; therefore, the ESP will share them with other organizations only when required to do so and only in a controlled manner. Each email message is created by one of the ESP's customers, who are the data subjects; it is then stored and catalogued using attributes that represent the message's metadata as collected by the ESP. Clients may be **law-enforcement (LE)** agencies, in which agents (data users) need access to email records to perform investigations. Intermediaries may include government agencies such as the IRS, which could provide tax records associated with the email addresses that appear in the messages' metadata attributes.

Design goals: Each organization wishes to maintain its proprietary view of the shared data and to keep that view confidential. This means that the set $ATT_{i,j}$ of attributes that ORG_i maintains on each shared payload must be hidden from the other organizations.

Another requirement that must be accommodated is the use of multiple vocabularies. The owner uses vocabulary VOC_1 to store and query the shared data, an intermediary uses vocabulary VOC_2 to enrich the shared data, and the client uses vocabulary VOC_3 to query and process the data, manage access control, and issue data-access authorizations to its employees. Therefore, our framework must provide a mechanism that *dynamically and obliviously* transforms attributes of shared data from one vocabulary to another. Note that that this problem cannot be solved simply by requiring all organizations that may need to share data to agree on a shared vocabulary. Such a standardization effort would require the organizations to know both the names and values of attributes used by other organizations. However, our premise is that the values of many attributes used internally by organizations are sensitive and cannot be exposed to other organizations. Furthermore,

in many natural use cases (see Section 2.2), transformations require auxiliary information, such as up to date statistics or lists. Such information is known only at the point at which a user requests a specific data record and may need to be supplied by an intermediary that is unknown by the data owner at the time that the owner encrypts the data.

Finally, because attributes could reveal sensitive aspects of organizations' activities, regulators and data subjects should expect the sharing of both payloads and attributes to be kept to a minimum. To facilitate minimal exposure of sensitive information, an interorganizational data-sharing framework should offer a data-centric access-control mechanism. Such a mechanism will allow a user to access a payload only if it is essential for the completion of one of her tasks; in addition, it will allow the user to learn only the subset of that payload's attributes that are needed for the task.

1.2 Starting Point: Attribute-based Encryption

Attribute-based encryption (ABE) is a natural starting point in the design of our framework. In our terminology, the encryptor is the data owner, users are data clients' employees (data users), and **trusted authorities (TAs)** both inside and outside the data client determine user access policies. An ABE scheme grants an individual user a key that permits him to decrypt a ciphertext if and only if the key matches certain attributes specified during the ciphertext's creation. ABE enables fine-grained access control, which is essential in a privacy-preserving data-sharing framework. It provides one-to-many encryption, which can significantly increase the scalability of encryption and key management—properties that are necessary for interorganizational data sharing. ABE policy-access formulae are highly expressive, because they can be specified with binary or multi-valued attributes using AND, OR, and threshold gates.

Existing ABE schemes, however, have several properties that make them unsuitable for our framework.

In existing ABE schemes, encryptors, users, and TAs all use the same vocabulary. This means that these schemes cannot be used off-the-shelf in our framework, where a crucial element of the problem is that participating organizations may belong to different business sectors or professional domains and thus use different vocabularies. In particular, a data client's TAs and employees may use a different vocabulary from that of the data owner. In ABE terms, this implies that attributes used in access policies (and keys) issued by the TAs to data users might belong to a different vocabulary from the one that the owner used to encrypt and store ciphertexts. Unless a suitable transformation is made between the keys and the ciphertexts, decryption will fail even if the ciphertext satisfies the user's access policy. Such a transformation must separately consider each attribute in the ciphertext and change it into a valid attribute from the users' keys' vocabulary. To protect both data subjects' privacy and organizations' proprietary views, the original attribute must remain hidden from the user, and the new attribute must remain hidden from the encryptor. Existing ABE schemes cannot support this requirement.

Moreover, existing ABE schemes are generally used for role-based access control and thus have user-centric vocabularies (attributes that describe decryptors' traits) that reflect organizational structure and roles. The use of user-centric attributes, coupled with the single-vocabulary assumption, implies that the encryptor (data owner) must be exposed to the roles of potential decryptors (clients' data users) and the organizational structure that they fit into. Many organizations are reluctant to share such sensitive information.

1.3 Main Contributions

We present a new system architecture and a suite of protocols for interorganizational data sharing that support the privacy of both data (*payload hiding*) and organizational vocabularies (*attribute*

hiding). We introduce **Attribute-Based Encryption with Oblivious Attribute Translation (OTABE)**, in which a semi-trusted proxy *translates* the attributes under which a data record's payload was encrypted into the attributes under which it can be decrypted by authorized users. The proxy performs the translation without learning the underlying plaintext data. Moreover, translation is *oblivious* in the sense that the attributes under which the record is encrypted remain hidden from the proxy. This novel cryptographic technique enables mutually untrusted parties not only to *use different vocabularies* of attributes to describe the shared data but also to share proprietary metadata attributes in a manner that protects the attributes' confidentiality (*attribute privacy*). Furthermore, attributes and policies can be *dynamically reconfigured* in the sense that updates are done dynamically, offline, and without the need for re-encryption. No previously proposed ABE scheme achieves all of these properties.

We provide a concrete OTABE scheme and prove it to be selectively secure in the standard model. We then use it in our design of an efficient, expressive, and flexible interorganizational data-sharing framework that we call PRShare. In addition to the direct benefits of OTABE described above, PRShare provides several other capabilities that are desirable in real-world interorganizational data-sharing applications, including *efficient and direct revocation, protection from key-abuse attacks*, and *hidden access policies*. To obtain these features, we leverage our OTABE scheme's translation technique. OTABE also enables *division of trust* (meaning that multiple independent authorities authorize data access) and *data centricity* (meaning that access policies contain data-related rather, than user-related, attributes), both of which enhance privacy protection in PRShare. Finally, because of the unique structure of OTABE ciphertexts, a single owner's database can serve multiple clients without knowing the clients' identities at the time of encryption. Furthermore, the owner does not need to authorize or serve clients' data-access queries. While previous ABE schemes achieved some of these desirable properties of our OTABE construction, none achieved all of them.

Before proceeding to our technical results, we note that our approach is not suitable for *all* data-sharing applications. For example, it is not intended for scenarios in which the data subject participates directly in the user's request for data about her and could be asked to grant explicit consent. In general, data subjects in the scenarios we consider will not even be aware of the specific uses that are made of data about them. Similarly, our approach is not intended for scenarios in which there are clear, efficiently decidable, and universal rules that govern which users can access which portions of the data; existing access-control mechanisms suffice in such scenarios. Rather, our techniques are useful in scenarios in which there are legally mandated, general principles that govern access to sensitive data, but in which instantiating those principles in the form of efficiently decidable rules requires data-specific and dynamically changing knowledge. We give two examples in Section 2.2.

Article outline: Section 2 presents related work and two use cases. In Section 3, we give the definition of attribute translation and the algorithms used in our **multi-authority OTABE (MA-OTABE)** scheme. Section 4 contains the PRShare system design. Section 5 discusses the security of our proposed scheme. Section 6 contains an overview of our construction, and Section 7 describes the construction and the attribute-translation function in more detail. In Section 8, we give formal statements and proofs of our OTABE results. Section 9 contains implementation details and performance evaluation. Conclusions and open problems are given in Section 10.

2 BACKGROUND AND MOTIVATION

2.1 Related Work

Existing privacy-preserving data-sharing schemes fall into two general categories: centralized and decentralized. The former category includes the works of Dong et al. [13], X. Liu et al. [32],

Table 1. Properties of Proxy-assisted ABE Schemes

Scheme	Type	Multi-Authority	Access policy	Dynamic attribute translation	Proxy's role	Direct revocation	Security model	Hidden policy
[22]	CP,KP-ABE	✗	LSSS	✗	Outsourced decryption	✗	RCCA	✗
[30]	CP-ABE	✗	AND gates	✗	Delegation of decryption rights	✗	Selective CPA	✗
[48]	KP-ABE	✗	LSSS	✗	Revocation management	✗	Selective CPA	✗
[44]	KP-ABE	✗	LSSS	✗	Revocation management	✓	Selective CPA	✗
[49]	CP-ABE	✗	AND gates	✗	Revocation management	✗	Selective CCA	✗
[5]	CP-ABE	✓	LSSS	✗	Outsourced decryption	✗	Selective RCPA	✓
[29]	CP-ABE	✗	LSSS	✗	Delegation of decryption rights	✗	Selective CCA	✗
[27]	CP-ABE	✗	AND gates	✗	Outsourced decryption, encryption	✗	Selective CPA	✗
[26]	CP-ABE	✗	LSSS	✗	Outsourced decryption	✗	Selective CPA	✗
OTABE	KP-ABE	✓	LSSS	✓	Attribute translation	✓	Selective CPA	✓

Popa et al. [36] and Vinayagamurthy et al. [45]. Their major advantage is efficiency; disadvantages include single points of failure and the lack of division of trust. Decentralized solutions can be found in the work of Fabian et al. [15], Froelicher et al. [19], C. Liu et al. [31], and Nayak et al. [33]. They avoid single points of failure but often have limited efficiency or scalability.

The original motivation for PRShare was enhancement of privacy protections in surveillance processes. Previous work in this area includes that of Kamara [24] and Kroll et al. [25], who proposed cryptographic protocols that protect the privacy of known surveillance targets. Segal et al. [42, 43] focused on unknown (i.e., not yet identified) targets and provided cryptographic protocols that protect the privacy of innocent bystanders in two commonly used surveillance operations: set intersection and contact chaining. Frankle et al. [18] used secure, multiparty computation and zero-knowledge protocols to improve the accountability of electronic surveillance.

Attribute-based encryption was introduced by Sahai and Waters [41]. Their work was followed by many **ciphertext-policy ABE (CP-ABE)** and **key-policy ABE (KP-ABE)** constructions, including those in References [4, 7, 20, 35, 38]. Chase [11] introduced multi-authority ABE, and Nishide et al. [34] introduced ABE with hidden access policy. ABE has been applied in a wide range of domains, including fine-grained data-access control in cloud environments [46], health IT [1, 28], and security of blockchains and Internet-Of-Things devices [37, 47].

We now explain some crucial differences between the role of proxies in OTABE and their roles in previous works. A detailed comparison of OTABE to prior proxy-assisted ABE schemes is shown in Table 1. In this table, we use the following acronyms, which are well established in the cryptographic literature: **linear secret-sharing scheme (LSSS)**, **chosen-ciphertext attack (CCA)**, and **chosen-plaintext attack (CPA)**. The R in RCCA and RCPA stands for **replayable**.

An OTABE scheme provides an algorithm, *Translate*(), which allows a semi-trusted proxy to translate one or more of the attributes under which a data record's payload is encrypted without learning the underlying plaintext. Moreover, translation can be done obliviously in the sense that the attributes under which the payload is encrypted remain hidden from the proxy who translates them. The proxy learns only the attributes' new values.

Two common responsibilities of proxies in ABE are *outsourced decryption*, introduced by Green et al. [22], and *revocation management*, which was used by Yu et al. [48, 49]. In both cases, proxies are used for efficiency; they assume much of the computational cost of decryption or revocation and lighten other parties' loads. The attribute-translation protocols in OTABE are *not* designed to reduce the client's or the owner's computational loads. Similarly, outsourced-decryption and revocation-management proxies are not designed to enable oblivious translation between organizational vocabularies or to support dynamically reconfigurable attributes. Simply put, proxies

used for outsourced decryption and revocation management and those in OTABE serve completely different primary purposes.[1]

The use of proxies for *ciphertext delegation* was introduced by Sahai et al. [40]. In this scenario, proxies take ciphertexts that are decryptable under policy P_1 and transform them into ciphertexts that are decryptable under policy P_2. However, P_2 must be stricter than and use the same vocabulary as P_1; here, "stricter" means than P_2 permits the decryption of a subset of the ciphertexts that could be decrypted under the original policy P_1 used by the encryptor. Neither of these restrictions applies to the proxies in OTABE.

In **attribute-based proxy re-encryption (ABPRE)**, which was introduced by Liang et al. [30], a proxy re-encrypts a ciphertext encrypted under access structure AS_1 to one that can be decrypted under access structure AS_2 without learning the plaintext. There is a superficial similarity between ABPRE and OTABE in that proxies in both transform ciphertexts encrypted by data owners under AS_1 into ciphertexts decryptable by clients under AS_2. However, the entity that issues re-encryption keys to proxies in ABPRE requires knowledge of the vocabularies of both owner and client; to create re-encryption keys, this entity must know AS_1 *and* AS_2. Thus, unlike OTABE, ABPRE does not support multiple vocabularies and cannot provide attribute privacy.

In an ABPRE scheme, re-encryption keys are issued to a proxy on a per-access-policy basis. To perform re-encryption, the entire access policy must be changed so that the new policy contains no attributes that appear in the original policy. Neither of these restrictions applies to OTABE, in which re-encryption-key issuing and re-encryption itself can be done on a per-attribute basis. The responsibility for determining the new attribute set and performing the re-encryption is divided among multiple parties from different trust domains. Each party performs a partial re-encryption that uses only the attributes that belong to its trust domain and does so in a controlled manner that results in a final, full re-encryption that satisfies the data owner's requirements. This decentralized approach allows OTABE to support multiple vocabularies, provide attribute privacy, and enable dynamically reconfigurable translation policies that do not require re-initialization of the system or re-encryption of records by the owner.

Finally, in ABPRE, the proxy must know the ciphertext's original access policy to perform the re-encryption. OTABE proxies, by contrast, perform *oblivious* translation and re-encryption; they do not learn the original set of attributes or the original access structure under which the plaintext was encrypted.

2.2 Use Cases

To motivate the introduction of OTABE and illustrate its applicability in real-world scenarios, we provide two examples.

Law-enforcement agencies (LEAs): The **Electronic Communications Privacy Act (ECPA)** [14] was passed to protect the privacy rights of ISP customers with respect to disclosure of their personal information. The ECPA limits LE access to email and other communication records in a manner that is consistent with the Fourth Amendment. However, it has several "loopholes." For example, the ECPA classifies an email message that is stored on a third party's server for more than 180 days as "abandoned." As a result, LE agencies can request that both the metadata and the content of those email messages be turned over without the need for judicial review.

Unrestrained government access to communication data is clearly undesirable. However, given national-security and public-health concerns, expecting LE and intelligence agencies never to access *any* data held by communication companies such as ESPs is unrealistic. A more realistic goal

[1] A direct-revocation mechanism, partially managed by the proxy, is a natural by-product of attribute translation, as described in Section 4.2, but it is not the primary goal of OTABE.

is to deploy a policy that restricts such data sharing to the minimum needed for the task at hand as defined by multiple trusted entities. OTABE provides a mechanism that can enforce such policies and protect the confidential information of all organizations and agencies that participate in the data-sharing protocols.

In OTABE terms, the data owner is the ESP, and the data subjects are people who send and receive email messages. The data are email records. Each email record contains a payload (the content of an email message), which is encrypted under a set of metadata attributes, e.g., the sender's and receiver's email addresses, date, subject line, and so on. The client is an LE agency, such as the FBI or a municipal police department, and the intermediaries may be other LE agencies, non-LE government agencies, or private companies. The data users are LE agents employed by the client.

Clearly, email records can be useful to LE agencies, but an agent should be able to decrypt only those records whose metadata attributes constitute *probable cause* in the context of a specific investigation. The entities who determine probable cause on a per-investigation basis are the TAs. Each TA is motivated by a different set of interests and goals. A TA may be part of the judicial branch, the ESP, the LE agency, or another external entity.

Not all of the attributes used by the ESP to store email records can be shared with the LE agency, because some of them reveal private information about the ESP's customers or proprietary information belonging to the ESP. Similarly, the attributes used by the LE agency to access and process records and to issue access policies cannot be shared with the ESP, because they reveal confidential information about the LE agency's investigations. Furthermore, some of the attributes that are used by the parties do not belong to the same vocabulary. For instance, the attribute "appears-in-watchlist" is frequently used in keys issued to LE agents, but it is meaningless to the ESP. Such attributes must undergo dynamic adaptation to ensure that agents' keys match an email message's attributes. OTABE allows the ESP and LE agency to use their own vocabularies while keeping the email messages' content and metadata confidential.

Trusted authorities are likely to grant an agent who is investigating a crime access to email records in which either the sender or the receiver is on the agency's watchlist. The LE agency's proxy can translate the ESP's sender and receiver attributes into the LE agency's "on-watchlist" attribute in an oblivious fashion, thus maintaining both the confidentiality of the watchlist and the privacy of data subjects' email addresses. In addition, an agent might want to check whether the sender or receiver appears on other agencies' lists, e.g., a list of investigations ongoing at LEA-2, which is another LE agency. Because details of LEA-2's ongoing investigations cannot be shared with the client, the translation of the attributes sender and receiver will be done obliviously by LEA-2's intermediary proxy.

Similarly, the access policy of an agent investigating cyber fraud may enable access to email records whose subject lines match a "suspicious" pattern. The definition of "suspicious" may be determined by a dynamically updated list of keywords. Using this keyword list, the client's proxy can obliviously translate the attribute "subject line," maintained by the ESP, into the attribute "is-suspicious-subject," maintained by the client and used in the agent's access policy. Neither the agent nor the proxy is able to read the actual subject line, and the data subject's privacy is maintained.

Note that, in both of these investigations, dynamic translations are needed, because watchlists and lists of suspicious keywords change over time. They enforce the requirement that an agent cannot access payloads without probable cause, but they do not reveal to the ESP confidential information about watchlists and ongoing investigations.

Insurance companies: Consumer reporting agencies (CRAs) collect and share credit-related information about consumers. This information is used by credit-card issuers, mortgage lenders, insurance companies, and other organizations to assess the creditworthiness of consumers. The

three largest CRAs in the US are Experian, TransUnion, and Equifax.[2] The **Fair Credit Reporting Act (FCRA)** [16] regulates the collection, dissemination, and use of credit-related information. The FCRA gives companies the right to access consumer credit reports. This access is not limited to reports on the company's customers and may include reports on large sets of *potential* customers. To create pre-screened offers and market them to potential customers, an insurance company is allowed to access consumer credit reports and share information with credit-card issuers, banks, other insurance companies, and so on. However, the FCRA limits credit-report access to *only* the information that serves a *permissible purpose* for the insurance company. OTABE can be used to formalize and enforce this vague concept in a manner that protects both consumers' privacy and proprietary information of insurance companies and CRAs.

In OTABE terms, the data owner is a CRA, and the data subjects are consumers. Data records are credit reports owned by the CRA. Each record is encrypted under the set of attributes that describe the report, e.g., the phone number, credit score, and **driver's license number (DLN)** of the data subject as well as the credit-utilization ratio, date, and time of the report's creation and CRA-internal statistics, and so on.

Insurance companies are the data clients. Data users are insurance-company employees who use credit reports to make decisions about which insurance products to offer consumers and how to price them. To comply with the FCRA's "permissible-purpose" requirement, employees should only access credit reports on a "need-to-know" basis. An employee can only access those records whose associated attributes are relevant to her task, as determined by a set of TAs. Trusted authorities may include the CRA, a government entity, or various parties within the insurance company. Other organizations, such as credit-card issuers, government entities, banks, and other insurance companies may serve as intermediaries by "enriching" data supplied by a CRA in a privacy-preserving manner.

As in the LE scenario, each organization wants to protect its proprietary information. For instance, the CRA does not want to reveal unnecessary identifying information about its customers, and an insurance company does not want to reveal how it decides which consumers qualify for pre-screened offers. Also as in LE, different organizations may use different vocabularies. Consider the attribute "number of accidents," which is used by insurance companies to screen potential customers. This attribute cannot be used by CRAs, because they do not maintain such information in their credit reports. OTABE supports all of these requirements.

Assume that each report is encrypted under these attributes: CREDIT-UTILIZATION-RATIO, CREDIT-SCORE, PHONE-NUMBER, DLN, and DATE. Employee U in the car-insurance department is assigned the task of finding qualified potential customers and tailoring pre-screened offers using information found in their credit reports.

The TAs determine that, for this task, a qualified customer is defined by the following policy:

CREDIT-SCORE>X $\wedge$ #ACCIDENTS<Y $\wedge$ IS-BLACKLISTED=FALSE $\wedge$ IS-CREDIT-RATIO-LESS-THAN-AVERAGE=TRUE

The intermediaries in this case are financial business partners of the insurance company, e.g., banks and credit-card issuers, and the **Department of Motor Vehicles (DMV)**.

To complete her task, U submits to the CRA a query that requests the reports of all consumers whose credit scores are greater than X. The CRA then sends each matching record to two intermediaries: the DMV and a credit-card issuer.

For each record, the DMV's proxy obliviously translates the DLN attribute into #ACCIDENTS, which is found in the subject's driving record. The credit-card issuer's proxy obliviously

[2]In September of 2017, Equifax announced a data breach that exposed the personal information of 147 million people and cost the company hundreds of millions of dollars in compensation to affected people [6, 17].

translates the numeric CREDIT-UTILIZATION-RATIO attribute into a binary attribute IS-CREDIT-RATIO-LESS-THAN-AVERAGE by obliviously comparing the consumer's utilization ratio with the average utilization ratio of the issuer's customers. The insurance company's proxy obliviously translates the PHONE-NUMBER attribute into the attribute IS-BLACKLISTED using a dynamically updated list of individuals who were blacklisted by the insurance company or one of its business associates for, e.g., failure to pay.

When U receives a record, she will be able to decrypt the credit report, read its contents, and learn the subjects' identifying information if and only if the record's post-translation attributes satisfy her access policy.

Data privacy is achieved, because only authorized users can decrypt a given credit report. Attribute privacy is achieved, because attributes used by each organization remain sufficiently hidden. Moreover, sensitive information from decrypted consumer records is also protected. For example, a user may learn that a consumer's number of accidents is below a certain threshold but not learn the exact number. Finally, these translations demonstrate OTABE proxies' ability to translate *dynamically*, because the list and the average change over time, and *obliviously*, because neither the attributes nor the data are revealed to them.

3 ATTRIBUTE-BASED ENCRYPTION WITH OBLIVIOUS ATTRIBUTE TRANSLATION

A summary of the notation and symbols used in this article is given in Table 2.

3.1 Terminology

Attributes: Our scheme uses multi-valued attributes denoted by $\langle label, operator, value \rangle$. Note that this representation is different from the ones found in typical ABE schemes, which use "descriptive" (essentially binary) attributes. We denote by att_k^L and att_k^V the label and value, respectively, of an attribute att_k. Translation of an attribute can be done either by changing the attribute's value (i.e., replacing $value$ with $value^*$) or by replacing both the attribute's label and its value with $label^*$ and $value^*$, respectively.

In PRShare, attribute labels are partitioned into two sets: mutable, denoted S_m, and immutable, denoted S_{im}. Immutable attributes are ones that cannot be translated by any party in the system. Intuitively, they are the attributes that are shared by the owner and the client. Mutable attributes, however, are ones that can be translated by a semi-trusted proxy at some point after their initialization by the owner.

Hidden access policy: We introduce an OTABE scheme with hidden access policy by ensuring that the set of attributes used to encrypt a message is hidden from the **cloud-service provider (CSP)**, the proxies, and the data users. We use the term "hidden access policy" for compatibility with the terminology used in existing CP-ABE work, in which access policies are attached to the ciphertexts.

In such a scenario, a data user is able to determine which attributes are needed to perform the decryption but cannot learn the attributes that are attached to a ciphertext. The hidden-access-policy feature is used to enhance privacy. However, if the owner and client wish to reveal the ciphertexts' attributes to the users or wish to speed up decryption at the expense of privacy, then they can turn off this feature without having to alter the encryption, translation, and decryption operations. This follows from the modular design of the system, as discussed in Section 5.1. Note that the hidden-access-policy feature does not enable the creation of trivial policies (i.e., those that always allow a user to decrypt every record she receives). This is because a key must satisfy *all* TAs' policies to succeed in decrypting, and the data owner can always serve as a TA or delegate authority to a TA that it trusts not to permit decryptions that it wishes to forbid.

Table 2. Summary of Notations and Symbols

Notation	Description	Notation	Description
$(M)_S$	encryption of M under a set S of attributes	$[x]_y$	encryption of x under the key y
ORG_S	set of proxies involved in translation of $C = (M)_S$	DEC_S	set of parties involved in decryption of $C = (M)_S$
P_{org_j}	the proxy operating on behalf of organization org_j	S_m	mutable attributes
S_{im}	immutable attributes	S_p	the set of attributes' labels that P_{org_p} is allowed to translate
$pub_\Pi(x)$	the public key of entity x, created by a public-key scheme Π	K_x	a symmetric shared key between org_{owner} and organization org_x
$org(k)$	the organization who is allowed to translate attribute att_k	$E_j(L)$	encryption of auxiliary information L by organization org_j
$F(K, x)$	pseudorandom function keyed with symmetric key K	$F(x)[0]$	the first argument of the output of the evaluation of F on x

In general, PRShare is designed to achieve a high level of privacy while allowing flexible and expressive data-sharing protocols. In real-world scenarios, however, organizations have different priorities. Some may favor privacy, but others may favor functionality and thus prefer to allow their data users broader access to information about the shared data at the expense of privacy. PRShare is able to support both approaches: It is highly modular, and each privacy guarantee relies on a different low-level feature that can be removed or changed to fit the organization's privacy-functionality tradeoffs while maintaining the rest of the privacy guarantees.

(Informal) Definition: Let M be a data record's payload encrypted under a set $S \subseteq \mathcal{U}_1$ of attributes, resulting in a ciphertext C. We refer to S as the set of original attributes under which M is encrypted. Let $T : \mathcal{U}_1 \rightarrow \mathcal{U}_2$ be a translation function from the universe $\mathcal{U}_1$ of attributes to the universe $\mathcal{U}_2$ of attributes, and let Q_j be the set of original attributes that a semi-trusted proxy j is allowed to translate. An ABE scheme supports *oblivious attribute translation by semi-trusted proxy* j if, given C, Q_j, and T, for all $s \in Q_j$, the proxy is able to compute $T(s)$ without:

- learning anything about M,
- learning anything about the attributes in $S \setminus Q_j$, or
- learning the labels or the values of attributes in $S \cap Q_j$.

Formal security definitions are given in Section 5.2.

3.2 Algorithms

An MA-OTABE scheme consists of the following algorithms:

GlobalSetup$(\lambda) \Rightarrow (PK)$: The global-setup algorithm takes as input a security parameter λ and outputs global parameters PK.

AuthoritySetup$(PK) \Rightarrow (PK_i, MSK_i)$: Each authority runs the authority-setup algorithm with PK as input to produce its own public key PK_i and master secret key MSK_i.

Encrypt$(M, PK, S, \{PK_i\}_{i \in Aut}) \Rightarrow (CT)$: The encryption algorithm takes as input a message M, a set S of attributes, and the public parameters. It outputs the ciphertext CT.

KeyGen$(PK, MSK_i, A_i, u, t) \Rightarrow (SK_{i,u,t})$: The key-generation algorithm takes as input the global parameters, an access structure A_i, a master secret key MSK_i, the global identifier u of a data user who issued the key-generation request, and a task t. It outputs a decryption key $SK_{i,u,t}$.

Distribute$(I) \Rightarrow (\{C^j | j \in DEC_S\})$: This algorithm takes as input a set I of ciphertexts' ids. It outputs a set of partial ciphertexts, $\{C^j | j \in DEC_S\}$.

Translate$(PK, j = p, C^p, \{PK_i\}_{i \in Aut}) \Rightarrow (C'^p)$: The translation algorithm takes as input the global public parameters and the authorities' public parameters, a proxy's index $j = p$, and a partial ciphertext C^p. It outputs a translated partial ciphertext C'^p.

Decrypt$(PK, \{SK_{i,u,t}\}, C^u, \{C'^j | j \in ORG_S\}) \Rightarrow (M)$: The decryption algorithm takes as input the global parameters, a set of secret keys $\{SK_{i,u,t}\}_{i \in Aut}$, a partial ciphertext C^u, and a set of translated partial ciphertexts $\{C'^j | j \in ORG_S\}$. It outputs the plaintext M.

4 SYSTEM MODEL

Definition of attributes: We define two sets of attributes' labels: S_{owner} represents the set of attributes that the owner uses to encrypt, store, and access data records that it owns. This set is determined by the data owner. S_{client} represents the set of attributes under which keys are generated; those are the attributes that the client uses to access and process the shared data records, and they are chosen by org_{client}. Note that $S_{owner} \cap S_{client} \neq \emptyset$; this means that some attributes are shared by the client and the owner. This enables the users to retrieve data records of potential interest from the CSP using queries that are composed of shared attributes and also enables the data owner, if it wishes, to be one of the TAs. We denote the universes of attributes comprising each set by $\mathcal{U}_{owner}$ and $\mathcal{U}_{client}$.

For each data intermediary org_j in the system, we define a set of attributes' labels $S_j \subseteq S_m$. It represents the set of attributes that is governed by org_j and hence can be translated by the proxy P_{org_j} that acts on behalf of org_j.

4.1 System Participants

Data owner: org_{owner} is responsible for encrypting each of its data records using the set $S \subseteq \mathcal{U}_{owner}$ of attributes that are most likely to appear in future queries. Examples of owners include telecoms, ESPs, credit-card companies, CRAs, and so on. Note that some organizations can function as both owners and clients with respect to different types of data records.

Data users in data clients: Data users are employees of a data clients org_{client} who need access to data records stored by org_{owner} to perform daily tasks. Examples of data clients can be credit-card companies, CRAs, insurance companies, government agencies, LEAs, and so on. Note that PRShare is currently designed for data sharing between organizations (and not between individual users). Hence, data users are only considered in the context of a client organization. Each user is assigned a unique global identifier and a list of tasks. Each task t has a well-defined time limit tl_t. The list is dynamic in the sense that tasks can be removed or added to it during the system run. A user issues two types of queries. A *key request* is used to obtain a key that corresponds to a specific access policy. A *data query* is used to obtain data records owned by org_{owner} that are relevant to a specific task in the user's task list.

Cloud-service provider: The CSP stores the ciphertexts outsourced by org_{owner} and responds to queries submitted by data users in org_{client}. A CSP can be implemented using a cloud-computing service such as **Amazon Simple Storage Service (Amazon S3)**. We assume the existence of a CSP to create a general framework that provides flexibility and efficiency to data owners of all sizes, including small- and medium-sized organizations. However, a large data owner, such as a telecommunications company, that wishes to use its own data centers instead of a CSP can do so straightforwardly in our framework.

Trusted authorities: Trusted authorities are the entities that determine the decryption policy of org_{client} and issue secret keys that are used by data users. They use attributes from $\mathcal{U}_{client}$. There must be at least two TAs, and they may be entities in org_{owner}, org_{client}, or external organization. We assume that at least one TA belongs to org_{client} and that at least one TA does not.

Proxies: Each proxy P_{org_j} represents a different organization org_j (either an intermediary or a client) and operates on behalf of that organization. The role of a proxy P_{org_j} is to translate a subset of attributes in $\mathcal{U}_{owner}$ under which a ciphertext was encrypted to the corresponding attributes in $\mathcal{U}_{client}$. To do this, the proxy uses both a generic translation algorithm that is used by all proxies in the system and an organization-specific translation function that is determined by org_j and may involve auxiliary information provided by the organization to its proxy. The generic translation algorithm is public, but the organization-specific translation function and auxiliary information are

considered private to org_j and P_{org_j}. We assume that every MA-OTABE scheme includes at least one proxy (the "client proxy") that is responsible for managing org_{client}'s user-level revocation mechanism and for performing vocabulary translations. Returning to the Amazon example, while data storage and management (CSP) are provided by AWS S3, a proxy can be deployed as a service in Amazon Elastic Compute Cloud. Note that the proxy and the CSP do not have to be hosted on the same platform. Of course, the same holds for different proxies that represent different clients; we expect, however, that different proxies representing the same client will be implemented using the same platform.

Data subjects: Each data record owned by org_{owner} is linked to a certain individual, the data subject. A data record's payload contains personal information about the data subject, including content produced by the data subject. We assume that every data subject has a **user id (UID)** that varies based on the type of data used in the system. Examples of UIDs include phone numbers and email addresses.

4.2 Revocation Mechanism

One major by-product of OTABE is the ability to implement an efficient and direct revocation mechanism, in which revoking the keys of a set U of users does not affect the keys of users not in U. Using the translation technique, a semi-trusted mediator can transform a ciphertext that was encrypted under a set of data-centric attributes at point A into a "personalized" ciphertext reflecting a specific data query made by a user at point B. The main idea of our revocation mechanism is the addition of **global-identifier (GID)** and time attributes to each key. In addition, we add a dummy GID and dummy times during encryption. These dummy attributes will be translated to suit the specific data query's time and requester only if a certain criterion is met. This creates an efficient mechanism in which most revocations are enforced automatically.

We assume that every data user receives a unique GID. The data client maintains a revocation list that contains revoked GIDs. Users whose GIDs are on the revocation list are not allowed to access any data record. Revocation-list updates are infrequent and happen only when a user completely leaves the organization. Furthermore, GIDs can be removed from the revocation list after a relatively short time, because the key-level revocation mechanism ensures that secret keys become invalid within a well-known and controlled length of time from the date they were issued.

For the key-level revocation mechanism, we leverage a basic trait of an organizational task: It has a well-defined time limit. This time limit is determined by the user's manager and may change while the user is working the task. In our case, the entities who choose the time limit are the TAs; this is an integral part of the per-task "probable-cause" approach. The time limit given to a specific task performed by a user becomes an attribute in the user's key. In addition, the encryptor adds to each ciphertext a dummy "time" attribute. That dummy attribute is translated by the client proxy to the current time at which the data query is submitted by the user, thus making a key-level revocation check an automatic part of any decryption attempt. In our construction, we view a "time limit" as a date. This can easily be extended to include more fine-grained notions of time.

We also leverage our attribute-translation technique for the user-level revocation mechanism. It enables us to include a user-specific component in the ciphertext; this component is adjusted according to the specific data user by the client proxy in the data-retrieval phase. Note that we treat the GID as an *additional attribute*. We incorporate the user's GID as an attribute in the user's secret keys and, in parallel, add a "placeholder" GID attribute to each ciphertext. When a user submits a data query, the placeholder attribute is translated to that specific user's GID only if she does not appear in the revocation list. This mechanism provides an efficient user-level revocation mechanism and protects the scheme from collusion attempts and key-abuse attacks.

Details of the translations used in our revocation mechanism are provided in Section 7.2.

4.3 Main Flows

The system model consists of an encryption flow, a dataflow, and a key-generation flow. We assume that the system has already been set up, resulting in the global public parameters PK and a public-key, master-secret-key pair (PK_i, MSK_i) for each trusted authority Aut_i.

Encryption flow: To encrypt a data record's payload M, org_{owner} first determines the set S of attributes under which M will be encrypted. $S \subseteq \mathcal{U}_{owner}$ is composed of $|S| - 2$ data-centric attributes that describe the record's metadata and two attributes that serve as "placeholders." The placeholders att_{GID} and att_{TIME} are initialized with random, "dummy" values by org_{owner} and receive their actual values from org_{client}'s proxy. Based on the attributes in S, the encryptor determines the set DEC_S of decryption parties. DEC_S contains all parties involved in the decryption of the ciphertext, i.e., a data user and the set ORG_S of organizations that are allowed to translate attributes in S (represented by their proxies). ORG_S includes the client's proxy and any number of data intermediaries' proxies. After determining DEC_S, org_{owner} encrypts M under S by calling $Encrypt(M, PK, S, \{PK_i\}_{i \in Aut})$ and receives a set $\{C^j\}$ of $|DEC_S|$ partial ciphertexts. $|DEC_S| - 1$ of the partial ciphertexts correspond to proxies and contain only attribute components. One corresponds to the data user and contains both attribute components and a data component; the latter contains the payload M itself. Note that, for each C^j, $U(C^j) \subseteq \mathcal{U}_{owner}$, where $U(C)$ is the vocabulary of attributes under which a ciphertext C is encrypted. Last, org_{owner} computes $Y = \{Obf(att_k) \mid att_k \in S\}$, a set of obfuscated values for immutable attributes in S, and uploads to the cloud the preprocessed ciphertext and the UID with which the ciphertext is associated.

Key-generation flow: A user u who belongs to org_{client} sends a key request to the TAs in each of the following cases: Either a new task is inserted to u's task list, or the time limit for an existing task in u's task list has expired, and her existing secret key for that task is no longer valid. The request contains a description of the task and the "ideal" access policy that u would like to obtain in the context of that task. Each authority Aut_i creates an access policy A_i based on an examination of the user's request and the nature of the specific task. It creates a GID attribute att_{GID} that contains the user's GID u. Finally, it determines tl_t, which is either a new time limit for t (if t is a new task) or an extended time limit (if t is an existing task and its time limit has expired) and uses tl_t to create a time-limit attribute att_{LIMIT}. The time-limit attribute that is embedded in a secret key must be expressed using the same units (date, month, time stamp, etc.) used in the time attribute att_{TIME} that is attached to the ciphertext. It then creates its secret key $SK_{i,u,t}$ by calling $KeyGen(PK, MSK_i, A'_i, u, t)$, where

$$A'_i = A_i \wedge att_{GID} \wedge att_{LIMIT} = A_i \wedge (GID == u) \wedge (TIME < tl_t).$$

Dataflow: A data user u sends a data query to the CSP. It contains a conjunctive query ψ on attributes from $\mathcal{U}_{owner} \cap \mathcal{U}_{client}$. The CSP retrieves the ciphertexts that satisfy the query. For each ciphertext C, it sends $C^{j=u}$ to u and each $C^{j=p}$ to a proxy P_{org_p}. At that point, because u received only a partial ciphertext, she cannot yet use her key for decryption. Each proxy P_{org_p} in ORG_S translates each attribute att_k such that $(att_k \in S) \wedge (att_k^L \in S_p)$ by calling $Translate(PK, j = p, C^p, \{PK_i\}_{i \in Aut})$ and computes an obfuscated value for each new attribute $att_{k'}$ that it added, creating $Y_p = \{Obf(att_{k'})\}$. The client organization's proxy also manages the user-level mechanism by performing a correct translation of att_{GID} and att_{TIME} only if u does not appear in the revocation list. Each proxy P_{org_p} then sends the translated partial ciphertext $C'^{j=p}$ and Y_p to the user. At this point, $U(C'^j)$ has changed from $\mathcal{U}_{owner}$ to $\mathcal{U}_{client}$. Because each partial ciphertext is, from the proxy's view, independent of the data component inside the ciphertext, each proxy is able to perform the translations without learning M. Moreover, the structure of each partial ciphertext ensures that P_{org_j} learns nothing about the attributes with labels that

do not belong to S_j. All attribute components that correspond to attributes that the proxy can translate contain obfuscations of the attributes rather than the attributes themselves; thus, each attribute att_k such that $(att_k \in S) \wedge (att_k^L \in S_p)$ remains hidden from the proxy, while the obfuscated value can still be used for various translation operations. The user gathers all the translated partial ciphertexts $\{C'^j | j \in ORG_S\}$ and her partial ciphertext C^u to create an aggregated ciphertext that she can decrypt using her secret key. Finally, u decrypts the payload by calling $Decrypt(PK, \{SK_{i,u,t}\}_{i \in Aut}, C^u, \{C'^j | j \in ORG_S\})$. The decryption succeeds if and only if the following three conditions hold:

- $\forall i \in Aut, TR(S) \models A_i$, where $TR(S) = Y \cup \{Y_j\}_{j \in ORG_S}$ represents the set of translated attributes, created based on the original set S of attributes.
- tl_t, the time limit for task t, has not expired. (Otherwise, att_{LIMIT} cannot be satisfied.)
- u has not been revoked, and no collusion or key-abuse attempt has been made. (Otherwise, att_{GID} cannot be satisfied.)

5 SECURITY DEFINITIONS

5.1 Goals and Trust Relationships

An OTABE-based framework should satisfy three security goals with respect to all **probabilistic polynomial-time (PPT)** adversaries.

Selective security against chosen-plaintext attacks: The adversary cannot learn (in the selective-security model) the plaintext of either an original ciphertext or an aggregated, translated ciphertext.

Security against colluding parties: Let $C = (M)_S$ be a valid MA-OTABE ciphertext. No coalition of at most $|DEC_S| - 1$ parties can learn anything about M.

Attribute secrecy: The trust model that we consider in this article is different from the standard ABE trust model. Unlike the plaintext, for which we have a single security notion that applies to all the participants, we cannot apply a uniform security criterion to the attributes. Because each party plays a distinct role in the protocol, the set of attributes to which it is allowed to be exposed differs from the sets to which other parties are allowed to be exposed. We define three security requirements to ensure the secrecy of ciphertexts' attributes: hidden access policy, oblivious translation, and attribute privacy.

Hidden access policy: The set of attributes used to encrypt a message cannot be learned by the CSP, the proxies, or the data users.

Oblivious translation: The original attributes that each proxy P_{org_j} translates remain hidden from the proxy. That is, for every attribute s such that $s^L \in S_j$, the proxy P_{org_j} is able to translate s into a new attribute $s' \in \mathcal{U}_{client}$ without learning s.

Attribute privacy: Informally, the attribute-privacy requirement states that organizations that share data must be able to maintain separate views of the data that they share.

Definition 5.1. Given a payload space $\mathcal{M}$, a universe $\mathcal{U}_{owner}$ of attributes used by the encryptor (org_{owner}) to describe data records it owns, and a universe $\mathcal{U}_{client}$ of attributes used by org_{client} for data usage and authorization management, we define a function $T_j^M : \mathcal{U}_{owner} \rightarrow \mathcal{U}_{client}$ that maps attributes in org_{owner}'s vocabulary to attributes in org_{client}'s vocabulary with respect to a data record's payload $M \in \mathcal{M}$. An OTABE scheme achieves *attribute privacy* if and only if

- For every data record's payload M and every attribute $s \in \mathcal{U}_{owner}$, if s is mutable, then the owner (encryptor) does not learn $T_j^M(s)$, the translated value of the attribute s with respect to M, where $j = org(s)$.

- For every data record's payload M and every attribute $v \in \mathcal{U}_{client}$, if $(T_j^M(v))^{-1}$ is mutable, then the client (decryptor) does not learn $(T_j^M(v))^{-1}$, the original value of the attribute v with respect to M, where $j = org(s)$.

The following observations about our threat model, which considers external adversaries as well as the parties presented in Section 4.1, are natural aspects of the security definitions and results presented in Section 5.2.

No organization fully trusts the other organizations. Our framework protects the owner's data records, attributes of the data held by each organization, and auxiliary information that is held by each organization and used for attribute translation. We assume that the owner is honest but curious.

No organization fully trusts its proxy server. CSPs and proxies are assumed to be honest but curious and are only given encrypted attributes and encrypted auxiliary information. Note that the use of honest but curious proxies is well established in the design of cryptographic protocols [3, 8, 10, 21, 23, 49].

The client organization does not fully trust its data users. Data users are assumed to be malicious and can only access records that are relevant to their assigned tasks as determined by the TAs. We assume that at least one TA is honest. Data users also cannot learn attributes of the shared data records that are held by organizations other than the data client.

Knowledge of plaintext. For each $M \in \mathcal{M}$, only the data owner and authorized, non-revoked data users from the data client are able to learn M.

Knowledge of attributes. Using terms given in Definition 5.1, we make the following observations about T_j^M for every $s \in \mathcal{U}_{owner}$ such that s is a mutable attribute and every $v \in \mathcal{U}_{client}$ such that $(T_j^M(v))^{-1}$ is a mutable attribute.

The encryptor (data owner), the CSP, all proxy servers besides proxy j (P_{org_j}), and all the TAs (except for the owner when it chooses to serve as a TA) do not learn $T_j^M(s)$.

The decryptor (data users), servers (proxy, CSP), and TAs (except for those that belong to the client organization) do not learn $(T_j^M(v))^{-1}$.

It should be noted that, in most real-world settings, not all attributes are considered "proprietary information." Attributes that are not considered proprietary can be learned by both the owner and the client. Note that a translation mechanism might still be needed in such scenarios. For instance, it might be needed to perform dynamic updates of an attribute or to obtain additional information about an attribute that is necessary for decryption but can only be provided by an intermediary or extracted from a proprietary information piece belonging to the intermediary.

5.2 Definitions

We start by presenting the definition of selective security for our scheme.

Let $E = (Setup, AuthoritySetup, Encrypt, Distribute, KeyGen, Translate, Decrypt)$ be an OTABE scheme for a set of authorities Aut, $|Aut| = K$. Consider the following OTABE game for a PPT adversary $\mathcal{A}$, a challenger $\mathcal{B}$, a security parameter λ, an attribute universe $\mathcal{U}_{owner}$, and an attribute universe $\mathcal{U}_{client}$.

Init: The adversary chooses the challenge attribute set S, where $S \subseteq \mathcal{U}_{owner}$. Based on S, the adversary chooses the challenge decryption-parties set DEC_S^*, where $DEC_S^* \subseteq DEC_S$. The adversary also chooses a subset of corrupted authorities Aut_c. We assume that all authorities but one are corrupted and denote the honest authority by Aut_h; thus, $Aut = Aut_c \cup \{Aut_h\}$. The adversary sends Aut_c, Aut_h, S, and DEC_S^* to the challenger.

Setup: The challenger runs the *Setup* algorithm to produce the public parameters PK and, for each authority Aut_i, runs the *AuthoritySetup* algorithm to produce PK_i and MSK_i. If Aut_i is honest, then the challenger sends PK_i to the adversary. If Aut_i is corrupted, then the challenger sends both PK_i and MSK_i to the adversary.

Phase 1: The adversary chooses a revocation list RL and sends it to the challenger. It may then issue any polynomial number of key requests for tuples of the form (access structure, GID, task identifier) and send them to the challenger.

Given a request (access structure=$AC \in \mathcal{U}_{client}$, GID=$u$, task=$t$), the adversary proceeds as follows. For requests issued for a corrupted authority Aut_i, the adversary runs $SK_{iut} = KeyGen(PK, MSK_i, AC, u, t)$ itself, because MSK_i was given to it in the setup phase. For requests issued for the honest authority Aut_h, the challenger provides the answer. It extracts the time limit tl_t from the description of task t and creates a time-limit attribute $att_{LIMIT} = \langle DATE, <, tl_t \rangle$. Given the GID u in the request, the challenger creates a GID attribute $att_{GID} = \langle GID, ==, u \rangle$. It then creates $AC' = AC \wedge att_{LIMIT} \wedge att_{GID}$, which is an updated version of AC, and performs:

- If $S \models AC'$ and $u \notin RL$, then the challenger aborts.
- If $S \models AC'$ and $u \in RL$, then S must contain $S_{GID} = u$. The challenger picks GID u', $u' \neq u$, and generates the secret key using $SK_{hu't} = KeyGen(PK, MSK_h, AC, u', t)$.
- If $S \not\models AC'$, then the challenger generates the secret key using $SK_{hut} = KeyGen(PK, MSK_h, AC, u, t)$.

Challenge: The adversary submits two messages m_0 and m_1 to the challenger. In addition, for every proxy j in DEC_S^*, it sends a bit a_j to the challenger. (By default, if j represents the user, then we assume $a_j = 0$.) The challenger flips a fair coin b and encrypts m_b under S: $CT = Encrypt(m_b, PK, S, \{PK_i\}_{i \in Aut})$. Assuming I_{CT} is the index corresponding to the ciphertext CT, the challenger computes a set $\{C^j | j \in DEC_S^*\}$ of partial ciphertexts using $Distribute(I_{CT})$. For each proxy $j \in DEC_S^*$, if $a_j = 1$, then the challenger performs a translation of the corresponding partial ciphertext, $C'^j = Translate(PK, j, C^j, \{PK_i\}_{i \in Aut})$, resulting in a translated partial ciphertext C'^j. Finally, it sends the ciphertext C^* to the adversary:

$$C^* = \bigcup_{j \in DEC_S^*} c_j^* \qquad c_j^* = \begin{cases} C'^j & \text{if } a_j = 1 \\ C^j & \text{if } a_j = 0 \end{cases}.$$

Phase 2: Phase 1 is repeated.

Guess: The adversary outputs a guess b' of b. The advantage of the adversary in this game is defined as $\Pr[b' = b] - 1/2$.

Definition 5.2. An MA-OTABE scheme is *selectively secure* if all PPT adversaries have negligible advantage with respect to λ in the selective-security game.

In the proof that our MA-OTABE construction is secure, we use a q-type assumption about prime-order bilinear groups: the *decisional q-Bilinear (t, n)-threshold Diffie-Hellman assumption ((q, t, n)-DBTDH).* It is similar to the **Decisional q-Bilinear Diffie-Hellman assumption (q-DBDH)** used in Reference [38].

The assumption is parameterized by a security parameter λ, a suitably large prime p, two prime-order bilinear groups $G1$ and $G2$, a bilinear map $e : G1 \rightarrow G2$, and integers q, t, and n, where $n \geq 1$ is polynomial in λ, and $t \leq n$. It is defined by a game between a challenger and an attacker. The attacker chooses a subset $V \subseteq [n]$ of t indices and sends it to the challenger. The challenger picks a group element g uniformly at random from $G1$, $q+3$ exponents $x, y, z, b_1, b_2, \ldots, b_q$ independently and uniformly at random from Z_p, and $n-1$ additional exponents $z_1, \ldots, z_{n-1}$ independently and

uniformly at random from Z_p. It sets $z_n = z - \sum_{c=1}^{n-1} z_c$. Then it sends $(p, G1, G2, e)$ and the following terms to the attacker:

$$g, g^x, g^y, g^z, g^{(xz)^2}$$
$$\forall l \in [q] : g^{b_l}, g^{xzb_l}, g^{xz/b_l}, g^{x^2zb_l}, g^{y/b_l^2}, g^{y^2/b_l^2}$$
$$\forall l, f \in [q], l \neq f : g^{yb_l/b_f^2}, g^{xyzb_l/b_f^2}, g^{(xz)^2b_l/b_f}, \Psi_{l,f},$$

where

$$\Psi_{l,f} = \{g^{xz_c(b_l/b_f)} | c \in V\}.$$

The challenger flips a fair coin b. If $b = 0$, then it gives the term $e(g,g)^{xyz}$ to the attacker. Otherwise, it gives the attacker a term R chosen uniformly at random from $G2$. Finally, the attacker outputs its guess b' for the value of b.

Definition 5.3. We say that *the* (q, t, n)*-DBTDH assumption holds* if all PPT attackers have at most a negligible advantage in λ in the above security game, where the advantage is defined as $\Pr[b' = b] - 1/2$.

6 CONSTRUCTION OVERVIEW

6.1 Main OTABE Techniques

Before presenting our construction in full detail, we present a simplified version that is inspired by the large-universe ABE scheme of Rouselakis and Waters [38] and that illustrates basic techniques that are new to our construction. Note that the scheme in Reference [38] is single-authority; we extend it here to a multi-authority scheme. Note that, although [39] also presents a multi-authority ABE scheme, it is a ciphertext-policy ABE scheme rather than a key-policy ABE scheme like OTABE. Furthermore, the multi-authority scheme presented in Reference [39] is not a mere adaptation of the scheme in Reference [38] to the multi-authority case but rather a separate scheme with a different construction and different security assumptions. The OTABE scheme that we present in our construction could not have been built using the scheme in Reference [39]. However, we would like to stress that the general translation technique offered in this article is not limited to the scheme in Reference [38] and can potentially be extended to other types of ABE schemes as well.

Ciphertext composition in Reference [38] is given by these equations:

$$C0 = Me(g,g)^{s\alpha} \qquad C1 = g^s \qquad C2_k = g^{f_k} \qquad C3_k = (\theta^{att_k} h)^{f_k} (w)^{-s}.$$

The ciphertext is composed of a data layer and an attribute layer. We refer to $C0$ and $C1$ as *data-layer components*, $C2$ and $C3$ as *attribute-layer components*, and each element in $C3$ as an *attribute component*. The data-layer component $C0$ in [38] contains the message M masked by the public key $e(g,g)^\alpha$ of the (single) TA. Assuming that M is encrypted under a set S of attributes, the attribute layer contains $2|S|$ components, i.e., two ($C2_k$ and $C3_k$) for each attribute att_k in the ciphertext. Each pair contains a uniform, randomly chosen term f_k that is local to the specific attribute att_k. $C3_k$ also contains the attribute att_k itself. The two layers are connected by the binder term s.

The basic idea of our construction is as follows. Assume that we have a data owner, a data client, two authorities (denoted Aut_1 and Aut_2), a client proxy, and a data user u.[3] Assume that the keys given to u by Aut_1 and Aut_2 are based on the access structures $att_1 \vee att_2$ and att_4, respectively.

The data owner wishes to encrypt a record M under a set $S = \{att_1, att_3\}$ of attributes, where $att_1 \in U_{owner} \cap U_{client}$, but $att_3 \notin U_{owner} \cap U_{client}$. That is, att_3 does not belong to the client's

[3]For clarity, we do not use intermediaries in this simplified construction.

vocabulary and hence needs to undergo translation before u can use it for decryption with the keys she received from the authorities. In this example, we assume that $T(att_3) = att_4$; that is, a correct translation of the attribute $att_3 \in U_{owner}$ is $att_4 \in U_{client}$.

To encrypt M, the owner produces a two-level ciphertext; it is similar to the one in Reference [38] but differs in several respects.

First, instead of creating $|S|$ attribute components $C3_k$, one for each attribute, the owner creates $|S| * |DEC_S|$ attribute components $C3_{k,j}$, one for each pair (attribute, decryption party), where DEC_S represents the set of parties that participate in the decryption of the ciphertext (*decryption-parties* set). In this example, $|DEC_S| = 2$, because there are two decryption parties: the user and the client proxy.

Second, we use the binder term s differently from the way it is used by Rouselakis and Waters [38]. In Reference [38], the binder term is used in the data layer and in each attribute component. By contrast, we use secret sharing to break s into $|DEC_S|$ shares: Each attribute component $C3_{k,j}$ contains only one share of the binder term, the one that corresponds to the decryption party j. In this example, there are two decryption parties: the user and the client proxy.

Third, recall that each attribute component in Reference [38] contains the actual attribute to which it corresponds. In our OTABE scheme, however, each attribute component contains the output of a given transformation that is applied to the attribute. This enables the proxy to translate the attribute *obliviously* without knowing its label or value. In our construction, the transformation is a keyed **pseudorandom function (PRF)**, but, as explained below, OTABE can accommodate a richer set of transformations to better serve each organization's business logic.

Fourth, we use another uniformly randomly chosen term, l_k. Like f_k, l_k is local to the attribute component in which it appears. It is used to double blind the attribute part ($\theta^{att_k} h$) of each attribute component, using $d_k = f_k * l_k$ as a blinding factor; in this way, f_k can be used by the proxy as a token for oblivious translation.

Because of the composition of the ciphertext, the proxy is able to translate the attribute $att_3 \in U_{owner}$ into a new attribute $att_4 \in U_{client}$. The proxy uses the attribute component $C3_{att_3,proxy}$, an obfuscated version of the original attribute att_3, the tokens given to it by the $Encrypt()$ algorithm, and Equation (1) in the $Translate()$ algorithm, where $att_k{}'$ corresponds to the new attribute (in our case, att_4). In general, determination of the new attribute is done obliviously based on the obfuscated original attribute's label and value; this determination is explained fully in Section 7.2.

When the user receives the translated record from the proxy, she combines it with her own attribute-layer components and data-layer components to create the final aggregated ciphertext. She uses the keys that she received from Aut_1 and Aut_2 to decrypt the aggregated ciphertext.[4] Decryption with this equation uses secret sharing and the unique structure of the translated attribute component received from the proxy, which includes both an obfuscated version of the original attribute att_3 and the new attribute att_4.

Finally, to enable hidden access policy, we do not attach the actual set S of attributes to the ciphertext. Instead, both the data owner and the proxy compute an obfuscated value of each attribute they add to the ciphertext, based on the **Public-key Encryption with Keyword Search (PEKS)** construction given in Reference [9]. Using trapdoors received from the TAs, u is able to perform a "blind intersection" of the obfuscated values received with the ciphertext and her own obfuscated access structure's attributes received from the TAs. Thus, u is able to determine which attributes are required for decryption without learning their values.

[4]Decryption of aggregated ciphertexts is done using Equation (2), which is given (along with the rest of the full construction) in Section 7.1).

6.2 Other Components of PRShare

PRShare combines the MA-OTABE construction in Section 7.1 with the following building blocks:

- Pseudorandom functions: The data owner and each organization org_j agree on two random k-bit keys $K_{org(j)}$ and $K1_{org(j)}$ for the PRFs $F_p : \{0,1\}^k \times \mathcal{U} \rightarrow \mathcal{U}$ and $F : \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^*$.
- Collision-resistant hash function: If the parties wish to use the hidden-access-policy feature, then they agree on a collision-resistant hash function H.
- **Searchable-encryption (SE)** scheme, Λ: The input to the *Distribute*() algorithm is a set I of ciphertexts' ids. I is the output of Search_Λ, an SE scheme's Search protocol executed by the CSP and a data user u. I contains the ids of ciphertexts whose associated attributes satisfy the conjunctive query ψ sent by u to the CSP.
- Translation function: In the setup phase, each organization org_j provides to its proxy the translation function T_j and the encrypted auxiliary information $E_j(L)$ according to which it should perform attribute translation.

Section 7 provides detailed descriptions of our MA-OTABE scheme and the associated attribute-translation procedure. However, for ease of exposition, it does not present these contributions in their maximum generality or explain all of their features. We briefly discuss some natural generalizations and interesting features here.

One essential feature of PRShare is *oblivious translation* of attributes in S_m by a semi-trusted proxy. Oblivious translation is accomplished by applying a transformation to the attribute inside each attribute component; this allows translation without disclosing the attributes' values to the proxy. The version of the full construction given in Section 7.1 applies the same transformation to each attribute in the ciphertext using two PRFs. This version demonstrates a specific translation operation in which the proxy performs oblivious equality tests and set-membership tests to determine the new attribute. However, PRShare supports a more flexible approach in which different transformations are applied to different attributes in the ciphertext based on the attributes' types and sensitivities. For example, if $att_k \in \mathcal{U}_{owner}$ is a numeric attribute, then the proxy can translate it into a descriptive attribute $att'_k \in \mathcal{U}_{client}$ by comparing att_k with a threshold that was provided to it by the organization it represents. It determines the value of the new, descriptive attribute according to the result of that comparison. In such a case, we would choose an order-preserving transformation instead of an equality-preserving transformation. Based on this modular approach and other PRF-based transformations, PRShare enables a broader set of translation operations that better suit organizations' translation logic. These operations include oblivious addition, keyword search, and numeric comparison [12]. Section 7.2 contains concrete examples of attribute translation.

The full construction in Section 7.1 involves just one data client. In fact, a data owner in PRShare can encrypt its data records once for use by multiple data clients, and it need not know who the data clients are at the time of encryption. What it does need to know is the universe T of TAs from which each data client chooses the set of TAs that it will use.

At the time of encryption, the owner uses the public keys of all $t \in T$ to create $C1$, which is the data layer. It creates the rest of the ciphertext's components exactly as they are created in Section 7.1. Now consider a client c that uses TAs $T' \subseteq T$. In the key-generation phase, data users associated with c will receive two types of keys: regular secret keys, which are issued by each TA in T' according to the *keygen*() algorithm, and dummy secret keys, which are issued by TAs in $T \setminus T'$. Each dummy key represents a "decrypt all" policy and thus has no effect when combined with the actual decryption policies represented by keys issued by TAs in T'.

Dummy keys are issued to each data user once during the setup phase, and the total number of TAs in the system is small. Furthermore, the attribute-layer components, which constitute the longer part of the ciphertext, remain the same under this generalization. Therefore, the performance of this generalized construction will be reasonable.

Query and retrieval of encrypted records in PRShare use a SE scheme Λ. There is a CSP that stores ciphertext records that the data owner has created using the *Encrypt*() algorithm and receives from data users requests that contain conjunctive queries on attributes in $\mathcal{U}$. In PRShare, the storage and processing of data records ("payloads") is decoupled from the storage and processing of their metadata. The SE scheme can be chosen independently of the OTABE scheme according to specific needs or privacy requirements of the client or owner. The only functionality that the SE scheme must provide is

(1) The data user can submit to the CSP a conjunctive query that contains attributes in $\mathcal{U}_{owner} \cap \mathcal{U}_{client}$.
(2) The CSP is able to retrieve all the records that match the query without learning the query's contents or the attributes associated with each record. Furthermore, the data user cannot learn the attributes that are associated with each record except for those that appear in the query.

Upon receiving a query from a data user, the CSP searches for all the ciphertexts that satisfy this query; for each one, it performs the *Distribute*() algorithm. Importantly, *the CSP need not perform any type of authentication or authorization of users*. Each payload and its associated attributes are stored in encrypted form according to the OTABE scheme, and *only users with suitable keys are able to decrypt the payload and the attributes*. If a user does not belong to a client organization that uses TAs in T, or if she *does* belong to such an organization but has not been issued the necessary decryption keys for the records that match her queries, then she will learn nothing from the encrypted payloads and attributes that the CSP sends her.

Note that the choice of SE scheme is highly flexible. One may choose a very simple scheme, in which tags are created using PRFs with keys that are shared among the relevant entities (owner, CSP, and clients), or a more sophisticated scheme that provides stronger security and privacy guarantees.

Finally, a distinction should be made between an access policy and a query. Access policies, on which secret keys are built, can include both "and" and "or" gates. This follows directly from Reference [38]. A query, however, has nothing to do with key management. A query is sent from a data user to the CSP and requests that the CSP retrieve certain data records. The query's structure depends on the underlying searchable encryption scheme used in PRShare. As mentioned, PRShare decouples the encryption scheme from the storage and search scheme. As long as it supports the requirements mentioned above, the latter can be chosen independently of the OTABE scheme according to the specific needs or privacy requirements of the client or owner. Thus, the owner may wish to use an SE scheme that supports general query types or one that only supports conjunctive queries (because it provides better security guarantees or better performance, for example). For simplicity, our construction assumes that queries are conjunctions of attributes.

7 DETAILED CONSTRUCTION AND TRANSLATION FUNCTION

7.1 Construction

We denote by $org(k)$ the organization that governs the attribute att_k. We denote by $pub_{\Pi}(P_{org_j})$ the public key of a proxy P_{org_j} that was created using a standard public-key encryption scheme Π.

Our MA-OTABE scheme consists of the following algorithms:

GlobalSetup$(\lambda) \Rightarrow (PK)$: This algorithm takes as input a security parameter λ. It defines bilinear groups $G1$,$G2$ of prime order p and a bilinear map $e : G1 \times G1 \rightarrow G2$. The attribute universe is $\mathcal{U} = Z_p$. Finally, the algorithm selects θ, h, and w randomly from $G1$. It returns the following global public key PK:

$$PK = (G1, G2, p, \theta, w, h, g, e).$$

AuthoritySetup$(PK) \Rightarrow (PK_i, MSK_i)$: Each authority Aut_i chooses random numbers $\alpha_i, \beta_i \in Z_p$. It sets $PK_i = (e(g,g)^{\alpha_i}, g^{\beta_i})$ as its public key and $MSK_i = (\alpha_i, \beta_i)$ as its master secret key.

Encrypt$(M, PK, S, \{PK_i\}_{i \in Aut}) \Rightarrow (CT)$: This algorithm takes as input a data record's payload M, the public keys for all authorities $\{PK_i\}_{i \in Aut}$, and a set of attributes S, $|S| = R$. It adds two attributes to S: $att_{DATE} =< DATE, ==, rand_1 >$, $att_{GID} =< GID, ==, rand_2 >$. Both are randomly initialized. It then chooses $2|S| + 2$ random exponents $s, a, \{f_k\}_{k \in [R]}, \{l_k\}_{k \in [R]} \in Z_p$ and computes $\{d_k = f_k * l_k\}_{k \in [R]}$. According to the nature of attributes in S, the encryptor determines the subset ORG_S of organization proxies that are able to perform translations of the ciphertext. The set of parties involved in decryption of C will include the set of proxies in ORG_S and the final decryptor, i.e., the user. Hence $|DEC_S| = |ORG_S| + 1 = P$. The encryptor chooses another P random elements $s_j \in Z_p$, $\sum_{j \in DEC_S} s_j = s$. It then encrypts M under S. The resulting ciphertext is composed of four elements: $C0, C1, C2, C3$ and a set Tok of tokens:

$$W = g^a \qquad C0 = M \prod_{i \in Aut} e(g,g)^{s\alpha_i} \qquad C1 = g^s \qquad C2_k = \{g^{d_k} | att_k \in S\} \qquad C3_{k,j} = \bigcup_{\substack{att_k \in S, \\ j \in DEC_S}} c3_{k,j}$$

$$c3_{k,j} = \begin{cases} D_{k,j} & \text{if } att_k^L \in S_{im} \\ E_{k,j} & \text{if } att_k^L \in S_m \end{cases},$$

where

$$D_{k,j} = (\theta^{att_k} h)^{d_k} (w)^{-s_j} \qquad E_{k,j} = (\theta^{F_p(K_{org(k)}, att_k)} h)^{d_k} (w)^{-s_j}$$

$$C2 = \{C2_k | att_k \in S\} \qquad C3 = \{C3_{k,j} | att_k \in S, j \in DEC_S\}$$

$$Tok_j = \{Tok_{k,j} | att_k^L \in S_j\} \qquad Tok_{k,j} = [Tok1_{k,j} || Tok1_{k,j} || Tok1_{k,j} || Tok1_{k,j}]_{pub_\Pi(P_{org(k)})}$$

$$Tok1_{k,j} = \theta^{l_k} \qquad Tok2_{k,j} = f_k \qquad Tok3_{k,j} = F(K1_{org(k)}, att_k^L) \qquad Tok4_{k,j} = F_p(K_{org(k)}, att_k)$$

$$C = (W, C0, C1, C2, C3, Tok = \{Tok_j | j \in ORG_S\}).$$

For each attribute att_k where $att_k^L \in S_{im}$, the encryptor computes an obfuscated value as follows: $Y^k = e((g^{\beta_{aut(k)}})^a, H(att_k))$, where $aut(k)$ denotes the authority that may use the attribute att_k in its access structure. The encryptor computes its signature $sig_u^{encryptor}$ on each element in C and on the number of attributes that each proxy in ORG_S is allowed to translate. In addition, for each proxy, it computes a signature $sig_p^{encryptor}$ on each element in $\{C3_{k,p} | att_k^L \in S_p\}$, on each element in Tok_j, and on the size of both sets. The encryptor then uploads the following record to the cloud server:

$$CT = \left(C, UID, P, Y = \left\{Y^k \mid \forall att_k^L \in S_{im}\right\}, sig_u^{encryptor}, \left\{sig_p^{encryptor} | p \in ORG_S\right\}\right).$$

KeyGen$(PK, MSK_i, A_i, u, t) \Rightarrow (SK_{i,u,t})$: The key generation algorithm for Aut_i, user u, and task t takes as input the master secret key MSK_i and access structure A_i, which are determined by the authority based on the combination of data-centric attributes that it considers to be a sufficient

justification for decrypting a data record's payload in the context of task t and the role of user u. The authority determines a new or updated time limit for task t, tl_t, and creates a time-limit attribute: $att_{LIMIT} =< DATE, <, tl_t >$. Last, given the user's GID, u, the authority creates a GID attribute, $att_{GID} =< GID, ==, u >$. Aut_i then creates $A'_i = A_i \wedge att_{LIMIT} \wedge att_{GID}$, an updated version of A_i. To ensure the hidden-access-policy feature, the authority replaces each attribute att_x in the access structure with a trapdoor $H(att_x)^{\beta_i}$ and transforms the resulting access structure into an LSSS access structure $(M_i; \rho)$, where M_i is an $ni \times mi$ matrix and ρ is a function that associates rows of M_i with attributes' trapdoors. The algorithm chooses random $y_2, \ldots, y_{mi} \in Z_p^n$ and creates a vector $vi = (\alpha_i; y_2, \ldots, y_{mi})$. For $c = 1, \ldots, ni$, it calculates: $\lambda_{i,c} = M_i(c) \cdot vi$, where $M_i(c)$ is the vector corresponding to the c'th row of the matrix M_i. In addition, the algorithm chooses ni random exponents $r_1, \ldots, r_{ni} \in Z_p$. For each $x \in [ni]$, it sets the private key $SK_{i,u,t}$ as

$$SK^1_{x,i,u,t} = g^{\lambda_{i,x}}(w)^{r_x} \qquad SK^2_{x,i,u,t} = (\theta^{\rho(x)}h)^{-r_x} \qquad SK^3_{x,i,u,t} = g^{r_x}.$$

Each authority Aut_i then sends

$$SK_{i,u,t} = \left\{SK^1_{x,i,u,t}, SK^2_{x,i,u,t}, SK^3_{x,i,u,t}\right\}_{x\in[ni]}$$

to u. The user's secret keys for task t are $\{SK_{i,u,t}\}_{i\in Aut}$.

$Distribute(I) \Rightarrow (\{C^j | j \in DEC_S\})$: The input to the $Distribute()$ algorithm is a set of ciphertexts' ids, I. The cloud first retrieves all the ciphertexts that are associated with ids in I. For a ciphertext CT that is encrypted under a set of attributes S and retrieved by the CSP, the CSP sends to each proxy P_{org_p} the following:

$$C^p = \left(\left\{C3_{k,p} | att^L_k \in S_p\right\}, P, sig_p^{encryptor}, Tok_p\right)$$

and sends to user u,

$$C^u = \{W, C0, C1, C2, C3_u, P, Y, sig_u^{encryptor}),$$

where:

$$C3_u = \{C3_{k,u} | att_k \in S\} \cup \{C3_{k,p} | att_k \in S, org(k) \neq p\}.$$

$Translate(PK, j = p, C^p, \{PK_i\}_{i\in Aut}) \Rightarrow C'^p$: the $Translate()$ algorithm for a proxy P_{org_p} and a data record's payload M encrypted under attribute S receives as input a partial ciphertext C^p. For each attribute component $C3_{k,p}$ that corresponds to an attribute att_k to be translated, the proxy first verifies the encryptor's signature. It then decrypts its tokens using its private key and extracts each of them. It computes $T_j(Tok3_{k,j}, Tok4_{k,j}) = att_k{}'$, thus obliviously translating the attribute att_k into a new attribute, att'_k. The function T_j is determined separately by each organization; see Section 7.2. It then computes a new value for $E_{k,p}$, $E'_{k,p}$:

$$E'_{k,p} = E_{k,p} \cdot \left(Tok1_{k,p}^{-PTok4_{k,p}+Patt_k{}'}\right)^{Tok2_{k,p}} = \left(\theta^{(Patt_k{}'-(P-1)F_p(K_{org(k)},att_k))}h\right)^{d_k} w^{-s_p}. \quad (1)$$

Finally, the proxy chooses a random exponent, $c \in Z_p$, where $W_p = g^c$, and computes, for each new attribute $att_k{}'$ that it created, an obfuscated value as follows: $Y^{att_{k'}} = e((g^{\beta_{aut(k')}})^c, H(att_{k'}))$. We use Y_p to denote the set of obfuscated values corresponding to attributes translated by proxy p. It then signs the new elements it added as well as the number of attributes it translated. It sends those signatures, sig_p, and the translated partial ciphertext to the user u. The record that is sent to the user is

$$C'^p = (C'3_p, sig_p, W_p, Y_p) \qquad C'3_p = \left\{C'3_{k,p} | att^L_k \in S_p\right\} = \left\{E'_{k,p} | att^L_k \in S_p\right\}.$$

Decrypt$(PK, \{SK_{i,u,t}\}, C^u, \{C'^j | j \in ORG_S\}) \Rightarrow M$: The decryption algorithm for a data record's payload M, which was encrypted under a set of attributes S and a user u, takes as input the global parameters, K secret keys $\{SK_{i,u,t}\}$ representing access structures $\{A'_i\}$, and two types of ciphertexts: C^u, a partial ciphertext received directly from the CSP, and $|ORG_S| = P - 1$ translated partial ciphertexts $\{C'^j | j \in ORG_S\}$, where $C'^j = (C'3_j, sig_j, W_j, Y_j)$, received from each of the proxies in ORG_S. After verifying both the encryptor's signatures and the proxies' signatures, the user aggregates all the translated partial ciphertexts she received from the proxies, extracts $C3_u$ from her partial ciphertext C^u, and creates an updated version $C'3$ of $C3$,

$$C'3 = \{C3_{k,u} | att_k \in S\} \cup \{C3_{k,p} | att_k \in S, org(k) \neq p\} \cup \{C'3_j | j \in ORG_S\}.$$

The user extracts $C0, C1$, and $C2$ from C^u and merges them with $C'3$. The final ciphertext is

$$C_f = (C0, C1, C2, C'3).$$

The user then determines the attributes that are needed for decryption, as well as their corresponding rows in the LSSS matrix of each authority. For a given access policy, represented by (M_i, ρ), the user uses W, received from the CSP and $\{W_j\}_{j \in ORG_S}$, received from each proxy and computes the following set:

$$S_i^* = \bigcup_{i \in [ni]} s_i^* \qquad s_i^* = \begin{cases} e(W, \rho(i)) & \text{if } att_k^L \in S_{im} \\ e(W_{org(k)}, \rho(i)) & \text{else} \end{cases}.$$

The user collects both the original attributes of the ciphertext and the ciphertext's translated attributes, to create the final set of attributes $TR(S) = Y \cup \{Y_j\}_{j \in ORG_S}$. By performing $\hat{S}_i = S_i^* \cap TR(S)$, she receives the (obfuscated) set of attributes $\hat{S}_i$ that are needed for decryption, I_i. This process is performed for each access policy $(M_i, \rho)_{i \in Aut}$, resulting in K obfuscated attribute sets, I_i and corresponding index-sets, Ind_i such that

- For all $c \in Ind_i, \rho(c) \in I_i$.
- Exist constants, $\{w_{c,i} \in Z_p\}_{c \in Ind_i}$, such that $\sum_{c \in Ind_i} w_{c,i} M_i(c) = (1, 0, \ldots, 0)$.

The algorithm now recovers M by computing

$$\frac{C0}{B},$$

where

$$B = \prod_{i \in Aut} \prod_{c \in I_i} (e(C1, SK^1_{c,i,u,t})(e(C2_c, SK^2_{c,i,u,t}))^P \prod_{j \in [P]} e(C'3_{c,j}, SK^3_{c,i,u,t}))^{w_{c,i}}. \tag{2}$$

7.2 Translation

The translation mechanism is composed of two parts: the public translation algorithm and the private, attribute-specific translation function. The public algorithm is specified by the method $Translate()$ (Equation (1)). It holds for all the proxies from all the organizations and for all attributes. This generic algorithm provides a semi-trusted server the ability to transform an attribute component in a multilayer ABE ciphertext that corresponds to attribute att into a new attribute component that corresponds to another attribute att' without learning the underlying plaintext or att. This algorithm assumes that att' is known to the translating server.

The private translation function (denoted by T_j) is organization-specific and attribute-specific. Its goal is to determine the new attribute, att', so that it can be used as input to the public, generic translation algorithm we described in the previous paragraph. Each organization org_j determines

on its own how it wants to translate each attribute that it governs (using the function T_j) and provides its own proxy with this information as well as any auxiliary information needed to perform that translation.

To conclude, $Translate()$ is a public, generic algorithm used by all the organizations' proxies that updates an attribute component in a multilayer ABE cipherext with respect to a given new attribute att'. The function T_j is an organization-specific, private translation function that determines how each attribute in the organization's vocabulary can be translated into attribute att', which belongs to the client's vocabulary.

In this section, we discuss only the organization-specific translation functions $\{T_j | j \in ORG_S\}$ and give concrete examples of such functions.

The translation of an attribute can be done in two ways: either by changing both the label and the value of the attribute or by keeping the attribute's label and changing only its value. A translation may require auxiliary information that is provided to the proxy by its organization. In such a case, the translation is done by performing an oblivious operation on the attribute that is encrypted using a certain transformation and on another object (e.g., a number or a list) that is encrypted using the same transformation; this other object is the "auxiliary information." Such an oblivious operation can be a comparison, equality test, list-membership test, keyword search, and so on. Since both the attribute inside the ciphertext and the organization-specific auxiliary information are encrypted using the same keyed transformation, with a key that is unknown to the proxy, the proxy can perform the translation without learning the attribute's value or the contents of the private auxiliary information provided by the organization.

On a high level, the transformation applied by organization org_j to a data structure, L, that contains multiple auxiliary information items, $l \in L$, works by treating each item l as the value of the corresponding attribute's label in S_{owner}, mapping the resulting attribute to an element in $\mathcal{U}$, and using the transformation to encrypt that element. The result, the encryption of auxiliary information L that belongs to an organization org_j, is denoted by $E_j(L)$. A similar process is used for auxiliary information that includes only one element, l, such as a threshold or a descriptive statistic.

Each organization prepares a lookup table in which entries represent obfuscated labels and values contain the translation logic and auxiliary information used for the translation of attributes with that label. Using the same obfuscated label received from the owner, the proxy knows what logic and auxiliary information it should use for the translation of the attribute it holds. It then uses the obfuscated value (in our construction, a PRF-encrypted value) of the attribute that the proxy obtained from the owner to compute the new attribute using the translation logic and auxiliary information.

We now present three important examples. For simplicity, in the following examples, we fix a specific translation function and refer to it as T. In addition, we use T as if it takes one argument, namely the original attribute. In practice (as shown in our construction), to support *oblivious* translation, a function T_j takes two arguments. Neither argument contains the actual original attribute; rather they contain obfuscated versions of the label and the value. In our construction, we use two PRFs for that purpose.

Dynamic translation between vocabularies: As discussed, translation of an attribute from org_{owner}'s vocabulary to org_{client}'s vocabulary is done according to the specific attribute being translated as well as the specific needs and work methodologies of the client organization.

One of the main reasons that attribute translation is essential to support multiple vocabularies is that, while the encryption of a data record's payload is done only once by the owner, the relevance of the data record to the client changes over time. In ABE terms, this means that, while the set of

attributes under which a ciphertext is encrypted in one vocabulary does not change, the question of whether this set satisfies a given access policy in another vocabulary *does* change over time. Furthermore, the decision about whether or not a ciphertext is relevant to the client at a given point in time is made using the "auxiliary information" that is related to one or more of the owner's, client's, or intermediaries' professional domains. Because the auxiliary information changes over time, so does the decision about whether or not the set of attributes of a given data record should satisfy a given access policy. Values of such attributes with respect to a data record cannot be fully determined at encryption time; they should be dynamically translated only when a data user needs to access that data record. OTABE supports such dynamic attributes, as shown below.

We consider two examples that represent common operations used to translate attributes from $\mathcal{U}_{owner}$.

The first operation is determining the new attribute according to the original attribute's membership in a list provided by the client organization or an intermediary. Since both the attribute and the list items are encrypted using the output of a PRF, such translation can be done obliviously.

To illustrate, we continue with the watchlist example given in Section 2.2. Two pieces of metadata that ESPs collect about their customers' email messages are the sender and receiver of the email message. Such attributes, however, cannot be used in the secret keys issued by the LE agency to its employees. Unless the investigation is targeted (and therefore the data subject's UID, e.g., phone number or email address, is known in advance), a raw email address will be meaningless as a justification for decryption; it thus cannot be used to determine the relevance of a certain ciphertext to one of the LE agency's investigations. Furthermore, exposing raw sender's and receiver's email addresses to agents in the LE agency will violate the privacy of data subjects that do not appear on any watchlist. Hence, the translation of the attribute "sender" is a Boolean attribute that indicates whether the sender of the email appears on an existing watchlist. Such an attribute better suits the daily activity of the LE agency and protects innocent citizens' privacy; thus, it can be included in the key to determine whether access to an email address is justified. Clearly, such a list cannot be revealed to an external entity, including the ESP.

Note that the raw email address's relevance to a given investigation may vary over time. This is because the auxiliary information, i.e., the watchlist, may change periodically, and thus the membership of a data subject associated with a given email address in the watchlist may change as well. This is why such attributes can only be translated dynamically when an agent submits an access request for that specific email record.

We now show how the data client, the LE agency, encrypts the watchlist. The watchlist, L, contains multiple items, $l \in L$, that represent data subjects' ids (for example, email addresses). In order for the watchlist to be compatible with the "sender" attribute under which email messages are encrypted, the LE agency performs the following preprocessing step on the watchlist:

$$for\ l\ in\ watchlist:$$

$$E_{j=client}(watchlist).add(F_p(K_{org_{j=client}}, < label = SENDER, operator = \text{“}==\text{''}, value = l >)),$$

where $E_{client}(watchlist)$ represents the resulting, encrypted watchlist containing multiple "sender" attributes and is thus compatible with the "sender" attribute used by the ESP.

Assuming $att_k =< SENDER, ==, c >$ represents a sender's email address, c, $att_k{}' =< ON - WATCHLIST, ==, b >$ is a Boolean attribute that represents whether the sender appears on a watchlist, and $E_{org_j}(L)$ represents an encryption of the watchlist L as described above, the value of b is determined by the proxy as follows:

$$Contains(E_{org_j}(L), Tok4_{k,j}) = b.$$

The second operation is determining the new attribute by comparing it to one or more numerical pieces of auxiliary information that usually represent either a certain threshold that is related to the attribute's value or aggregated statistics about other data records that share the same attribute. In this case, instead of an equality-preserving transformation, we will use an order-revealing transformation such as the **order-revealing encryption (ORE)** scheme presented in Reference [12], denoted by Π_{ORE} (which also makes use of a PRF). Since both the attribute and the threshold or the descriptive statistic with which the attribute is to be compared are encrypted using Π_{ORE}, such translation can be done obliviously.

To illustrate, we use the insurance-company example discussed in Section 2.2. We consider the attribute "credit utilization ratio" used by the CRA to store credit reports. Such an attribute cannot be used in the secret keys issued by the insurance company to its employees, because a raw number will be meaningless in the determination of whether a consumer is a good candidate for an insurance offer and therefore cannot be used to determine the relevance of a certain credit report to an employee's task. Furthermore, exposing the exact utilization ratio to the insurance company's employees will violate consumers' privacy. Hence, the translation of the numeric attribute "credit utilization ratio" is a Boolean attribute that indicates whether that ratio is below the average ratio. Such an attribute better suits the daily activity of the insurance company and protects consumers' privacy to the fullest extent possible. Therefore, it can be included in employees' keys to determine whether the insurance company considers the data subject a good enough candidate for an insurance offer. In this case, the translation will be made by the credit-card company's proxy, which acts as an intermediary by obliviously comparing the number representing a given consumer's credit-utilization ratio to the average utilization ratio of its customers. Clearly, the average utilization ratio that is calculated by each credit-card company based on its own customers' utilization constitutes proprietary information of the company and should not be revealed to other organizations.

Assuming $att_k =< CREDIT-UTILIZATION-RATIO, ==, c >$ represents the credit utilization ratio, c, $att_k' =< IS-CREDIT-RATIO-LESS-THAN-AVERAGE, ==, b >$ is a Boolean attribute that represents whether the credit utilization ratio is above the current average, as calculated by the credit-card company, and $E_{org_j}(l)$ represents an encryption of the average l using Π_{ORE}, the value of b is determined by the proxy as follows:

$$\Pi_{ORE}.COMPARE(E_{org_j}(l), Tok4_{k,j}) = b.$$

Note that in both cases, both the label and the value of the attributes are being translated.

Key-level revocation: $P_{org_{client}}$ translates the value of the attribute $att_k =< DATE, ==, rand >$ from the c-bit random value (its default value given at encryption time by the data owner) to the current date (or the current time stamp, if a time limit is expressed using time instead of dates), $date_{cur}$, and so $T(att_k) = att_k' =< DATE, ==, date_{cur} >$. In this case, only the value of the attribute is being translated. Note that access structures in our system contain a time-limit attribute of the form $< DATE, <, tl_t >$, where tl_t is the per-task time limit assigned by the TAs.

User-level revocation: Given the GID u of the data user who sent a data retrieval request, $P_{org_{client}}$ performs the following actions: It first checks whether u appears in org_{client}'s revocation list. If so, then it aborts. Otherwise, the proxy translates the value of the attribute $att_k =< GID, ==, rand >$ from the c-bit random value (its default value given at encryption time by the data owner) to the data user's GID, u, and so $T(att_k) = att_k' =< GID, ==, u >$. Note, that if the revocation list contains the data user's GID, then the partial ciphertext $C'^{P_{org_{client}}}$ will not be sent to the data user who initiated the retrieval request. Furthermore, in such a case, att_{GID} will remain with its default random value assigned by org_{owner}.

In both revocation events, only the attribute's value is being translated, as the original attributes serve as placeholders.

8 RESULTS

We now give the formal statements and full proofs of the properties of the scheme presented in Section 7.

Lemma 8.1. *If $n \geq 2$ and $t \leq n$, then (q,t,n)-DBTDH $\Rightarrow$ q-DBDH.*

Proof. From Definition 5.3, it is enough to prove that (q,n,n)-DBTDH $\Rightarrow$ q-DBDH.

Given a distinguisher D_1 that is able to tell a (q,n,n)-DBTDH term from a random term with non-negligible probability, we want to show that there exists a polynomial distinguisher D_2 that is able to tell a q-DBDH term from a random term with non-negligible advantage. We are given the terms:

$$\Omega_1 = \{g, g^x, g^y\} \cup \{g^{b_l}, g^{y/b_l^2}, g^{y^2/b_l^2} | \forall l \in [q]\} \cup \{g^{yb_l/b_f^2} | \forall l, f \in [q], l \neq f\}$$

$$\Omega_2 = \{g^z, g^{(xz)^2}\} \cup \{g^{xzb_l}, g^{xz/b_l}, g^{x^2zb_l} | \forall l \in [q]\} \cup \{g^{xyzb_l/b_f^2}, g^{(xz)^2b_l/b_f}, g^{xzb_l/b_f} | \forall l, f \in [q], l \neq f\}$$

and R, where R is either a q-DBDH term $e(g,g)^{xyz}$ or a random term. We choose a set $A = \{a_i\}$, where each a_i is randomly selected from Z_p (note that for the (q,n,n)-DBTDH, V is uniquely determined, as $V = [n]$) and compute, for each element in Ω_2, a new term:

$$h1 = (g^z)^{\sum_{i=1}^n a_i} = g^{(\sum_{i=1}^n a_i z)}$$

$$h2 = (g^{(xz)^2})^{(\sum_{i=1}^n a_i)^2} = g^{(x(\sum_{i=1}^n a_i z))^2}$$

$$h3 = (g^{xzb_l})^{\sum_{i=1}^n a_i} = g^{x(\sum_{i=1}^n a_i z)b_l}$$

$$h4 = (g^{xz/b_l})^{\sum_{i=1}^n a_i} = g^{x(\sum_{i=1}^n a_i z)/b_l}$$

$$h5 = (g^{x^2zb_l})^{\sum_{i=1}^n a_i} = g^{x^2(\sum_{i=1}^n a_i z)b_l}$$

$$h6 = (g^{xyzb_l/b_f^2})^{\sum_{i=1}^n a_i} = g^{xy(\sum_{i=1}^n a_i z)b_l/b_f^2}$$

$$h7 = (g^{(xz)^2b_l/b_f})^{(\sum_{i=1}^n a_i)^2} = g^{(x(\sum_{i=1}^n a_i)z)^2b_l/b_f},$$

$$\Psi'_{l,f} = \{(g^{xzb_l/b_f})^{a_i} | i \in [n]\} = \{g^{x(a_i z)b_l/b_f} | i \in [n]\}.$$

We set Ω'_2 as follows:

$$\Omega'_2 = \{h1, h2\} \cup \{h3, h4, h5 || \forall l \in [q]\} \cup \{h6, h7, \Psi'_{l,f} | \forall l, f \in [q], l \neq f\}.$$

Note that if R is a q-DBDH term, then $R' = R^{(\sum_{i=1}^n a_i)} = e(g,g)^{xy(\sum_{i=1}^n a_i z)}$ is a (q,n,n)-DBTDH term, and, if R is a random term, then R' is a random term. We then view $(\Omega_1, \Omega'_2, R')$ as input to the oracle D_1 to obtain correct value $b \in \{0,1\}$ ($b = 0$ if the answer of D_1 is (q,n,n)-DBTDH term, and 1 otherwise). Therefore, we have a polynomial distinguisher D_2 that is able to tell q-DBDH term from a random term with same non-negligible advantage. □

Theorem 8.2. *If (q,n,n)-DBTDH holds, then our MA-OTABE scheme achieves selective security against all PPT adversaries with a challenge attribute set S of size W, where $W \leq q$, and a challenge decryption-parties set DEC_S^* of size P, where $P \leq n$.*

Proof. To prove the theorem, we will assume that there exists a PPT adversary $\mathcal{A}$ with a challenge attribute set S and a challenge decryption-parties set DEC_S^* that has a non-negligible advantage in selectively breaking our MA-OTABE scheme. Using $\mathcal{A}$, we will build a PPT simulator $\mathcal{B}$ that attacks the (q, n, n)-DBTDH assumption with a non-negligible advantage.[5]

Init: The simulator receives the given terms from the assumption. The adversary chooses the challenge attribute set S, where $|S| = W$. Based on S, the adversary chooses the challenge decryption-parties set DEC_S^*, where $DEC_S^* \subseteq DEC_S$ and $|DEC_S^*| = P$. The adversary chooses a subset of corrupted authorities Aut_c. We assume all authorities but one are corrupted and denote the honest authority by Aut_h. The adversary sends Aut_c, Aut_h, S, and DEC_S^* to the simulator.

Setup: We denote S as $\{att_1, \ldots, att_W\}$ and the set of indexes of attributes in S as I_S.

The simulator chooses h^* and u^* randomly from Z_p. For each attribute att_l, it chooses e_l randomly from Z_p. It then computes the global public parameters:

$$w = g^x$$

$$\theta = g^{u^*} \prod_{l \in I_S} (g^{y/b_l^2})$$

$$h = g^{h^*} \prod_{l \in I_S} (g^{xz/b_l e_l}) \prod_{l \in I_S} (g^{y/b_l^2})^{-att_l}.$$

Based on the global public parameters, the simulator creates the parameters for authority Aut_i as follows:

For every $Aut_i \in Aut_c$, the simulator chooses random $n_i \in Z_p$ and sets $MSK_i = -xn_i$. It computes $PK_i = e(g,g)^{MSK_i} = e(g^x, g^{-(n_i)})$. The simulator sends MSK_i and PK_i to the adversary. For Aut_h, the simulator sets $MSK_h = xy + x \sum_{i \in Aut_c} n_i$. It computes $PK_h = e(g^x, g^y) \prod_{i \in Aut_c} e(g^x, g^{n_i})$. The simulator sends only PK_h to the adversary.

Phase 1: The adversary chooses a revocation list RL and sends it to the simulator. It may then issue any polynomial number of private key queries for tuples of the form (access structure, GID, task identifier) and send them to the simulator.

For a query: (access structure=AC, GID=u, task=t), the simulator does the following:

For queries issued for a corrupted authority $Aut_i \in Aut_c$, the adversary runs $SK_{iut} = KeyGen(PK, MSK_i, AC, u, t)$ itself, because MSK_i was given to it in the setup phase. For queries issued for the honest authority Aut_h, the simulator provides the answer. The simulator determines a time limit for task t, tl_t, and creates a time-limit attribute: $att_{LIMIT} =< DATE, <, tl_t >$. In addition, given the GID in the query, u, the simulator creates a GID attribute, $att_{GID} =< GID, ==, u >$. It then creates an updated version of AC, $AC' = AC \wedge att_{LIMIT} \wedge att_{GID}$ and performs the following:

- If $S \models AC'$ and $u \notin RL$, then the simulator will abort.
- If $S \models AC'$ and $u \in RL$, then S must contain $S_{GID} = u$. The simulator picks a GID u', $u' \neq u$, and generates the secret key using $SK_{hu't} = KeyGen(PK, MSK_h, AC, u', t)$
- If $S \not\models AC'$, then the simulator generates the secret key using $SK_{hut} = KeyGen(PK, MSK_h, AC, u, t)$.

We will now show how the simulator produces the secret keys in the last two cases.

[5]For simplicity, we prove our attribute-secrecy related claims separately, in Theorem 8.5. We also omit the signatures that are attached to some of the messages in our construction.

- In the second case, the simulator first creates $att'_{GID} =< GID, ==, u' >$. Then it needs to create a key for $AC^* = AC \wedge att'_{GID} \wedge att_{LIMIT}$.
- In the third case, the simulator needs to create a key for $AC^* = AC'$.

Those access policies are represented by an LSSS matrix M^{AC^*} with dimensions $l \times n$ and a row-mapping function ρ. Note that, in both cases, S is not authorized for AC^*. Hence, we can split M^{AC^*}'s rows into two sets:

$$A = \{r | r \in [l], \rho(r) \in S\} \qquad B = \{r |\ r \in [l], \rho(r) \notin S\},$$

where $A, B \neq \emptyset$. Because S is not authorized for M^{AC^*}, the properties of LSSS imply that we can find a vector $\beta \in Z_p^n$ with $\beta_1 = 1$ such that $\forall r \in A, M_r^{AC^*}\beta = 0$.

The simulator then chooses $n-1$ elements $\{v_i\}_{2 \leq i \leq n}$ uniformly at random from Z_p and sets the shares of MSK_h as

$$\lambda_r =< M_r^{AC*}, \Theta >,$$

where

$$\Theta = MSK_h\beta + (0, v_2, \ldots, v_n)^{\perp}.$$

Hence, row's r share is

$$\begin{aligned}\lambda_r &=< M_r^{AC^*}, (MSK_h\beta + (0, v_2, \ldots, v_n)^{\perp}) >= xy < M_r^{AC^*}, \beta > + x \sum_{i \in Aut_c} n_i < M_r^{AC^*}, \\ &\beta > + < M_r^{AC^*}, (0, v_2, \ldots, v_n)^{\perp} > \\ &= xy < M_r^{AC^*}, \beta > + x \sum_{i \in Aut_c} n_i < M_r^{AC^*}, \beta > + \lambda'_r.\end{aligned}$$

Now, let us see how the simulator computes the secret key for $r \in A$: By definition, $r \in A \rightarrow \rho(r) \in S$. From LSSS properties, $< M_r^{AC^*}, \beta >= 0$. Thus, in this case,

$$\lambda_r = \lambda'_r =< M_r^{AC^*}, (0, v_2, \ldots, v_n)^{\perp} >,$$

and, hence, its value is known to the simulator. The simulator can then compute the key components SK^1, SK^2, and SK^3 as in the *KeyGen* algorithm:

$$SK^1 = g^{\lambda_r} w^{t_r} = g^{\lambda'_r} g^{x a_r}$$

$$SK^2 = (\theta^{\rho(r)} h)^{-t_r} = \left(\left(g^{u^*} \prod_{l \in I_S} (g^{y/b_l^2})\right)^{\rho(r)} \left(g^{h^*} \prod_{l \in I_S} (g^{xz/b_l e_l}) \cdot \prod_{l \in I_S} (g^{y/b_l^2})^{-att_l}\right)\right)^{-a_r}$$

$$SK^3 = g^{t_r} = g^{a_r},$$

where $t_r = a_r$ are randomly selected from Z_p by the simulator and $\lambda_r = \lambda'_r$.

Finally, for $r \in B$, the simulator will compute the secret key in the following way: By definition, $r \in B \rightarrow \rho(r) \notin S$. In this case, the simulator will define

$$t_r = -\sum_{i \in Aut_c} (n_i) < M_r^{AC^*}, \beta > -y < M_r^{AC^*}, \beta > + \sum_{l \in I_S} \frac{xzb_l < M_r^{AC^*}, \beta >}{\rho(r) - att_l} + t'_r,$$

where t'_r is randomly selected from Z_p. Hence the key components can be computed as

$$SK^1 = g^{\lambda_r} w^{t_r} = g^{\lambda'_r} \prod_{l \in [n]} (g^{x^2 z b_l})^{< M_r^{AC^*}, \beta > / (\rho(r) - att_l)} \cdot g^{x t'_r}$$

$$SK^2 = (\theta^{\rho(r)}h)^{-t_r} = g^{\sum_{i\in Aut_c}(n_i)<M_r^{AC^*},\beta>(\rho(r)u^*+h^*)}(g^y)^{<M_r^{AC^*},\beta>(\rho(r)u^*+h^*)}$$
$$\cdot \prod_{l\in I_S}(g^{xzb_l})^{-(\rho(r)u^*+h^*)<M_r^{AC^*},\beta>/(\rho(r)-att_l)} \cdot \prod_{(l,f)\in I_S}(g^{(xz)^2 b_f/b_l e_l})^{-<M_r^{AC^*},\beta>/(\rho(r)-att_f)}$$
$$\cdot \prod_{l\in I_S}(g^{y^2/b_l^2})^{<M_r^{AC^*},\beta>(\rho(r)-att_l)} \cdot \prod_{l\in I_S}(g^{xz/b_l e_l})^{\sum_{i\in Aut_c}(n_i)<M_r^{AC^*},\beta>}$$
$$\cdot \prod_{l\in I_S}(g^{y/b_l^2})^{\sum_{i\in Aut_c}(n_i)<M_r^{AC^*},\beta>(\rho(r)-att_l)}$$
$$\cdot \prod_{\substack{(l,f)\in I_S\\ l\neq f}}(g^{xzy(b_f/b_l^2)})^{-<M_r^{AC^*},\beta>(\rho(r)-att_l)/(\rho(r)-att_f)} \cdot (\theta^{\rho(r)}h)^{-t'_r}$$

$$SK^3 = g^{t_r} = (g)^{-\sum_{i\in Aut_c}(n_i)<M_r^{AC^*},\beta>}(g^y)^{-<M_r^{AC^*},\beta>} \cdot \prod_{l\in I_S}(g^{xzb_l})^{<M_r^{AC^*},\beta>/(\rho(r)-att_l)} \cdot g^{t'_r}.$$

Therefore, in both cases, the simulator can reply to the adversary's query with the entire secret key. Because AC and AC' are subsets of $\mathcal{U}_{client}$, and S_{GID} and S_{LIMIT} are elements of $\mathcal{U}_{client}$, $AC^* \subseteq \mathcal{U}_{client}$, and so is the secret key given to the adversary. In addition, all the secret key's terms for A and B can be calculated by the simulator using terms from the assumption, the challenge set S (chosen by the adversary), and the access structure AC (chosen by the adversary).

Challenge: $\mathcal{A}$ submits two messages, m_0 and m_1, to the simulator. In addition, for every proxy in DEC_S^*, j, it sends a bit a_j to the simulator. The simulator then flips a random coin b and encrypts m_b under S: $CT = Encrypt(m_b, PK, S, \{PK_i\}_{i\in Aut})$, by implicitly setting $s = z$, $\{l_k = b_k | \forall k \in I_S\}$, $\{f_k = e_k | \forall k \in I_S\}$, $\{d_k = b_k e_k | \forall k \in I_S\}$ and $\{s_j = z_j | \forall j \in DEC_S^*\}$. For each proxy $j \in DEC_S^*$, the simulator creates a partial ciphertext $C^j = (\{C3_{k,j} | att_k^L \in S_j\}, P, Tok_j)$ using the $Distribute$ algorithm and, if $a_j = 1$, performs $C'^j = Translate(PK, j, C^j, \{PK_i\}_{i\in Aut})$. Note that, for every proxy j such that $a_j = 1$, if an attribute $att_k^L \in S_j$, the simulator holds two attributes: the original attribute, att_k, and the translated attribute, $att_{k'}$. Finally, the simulator extracts $C0, C1, C2, C3 = \{C3_{k,j} | att_k \in S, j \in DEC_S^*\}, Tok$ from CT and extracts $C'3_j = \{C'3_{k,j} | att_k^L \in S_j\}$ from each translated partial ciphertext, C'^j. The simulator then sends the translated ciphertext C^* to $\mathcal{A}$. Note that each element in C^* can be computed using terms from the assumption:

$$C^* = \{C0, C1, C2, C^*3, Tok\},$$

where

$$C0 = m_b \cdot e(g,g)^{xys} = m_b \cdot R \qquad C1 = g^s = g^z \qquad C2 = \{g^{d_k} | att_k \in S\} = \{g^{b_k e_k} | att_k \in S\}$$

$$C^*3 = \bigcup_{\substack{att_k\in S,\\ j\in DEC_S^*}} c^*3_{k,j} \qquad c^*3_{k,j} = \begin{cases} C'3_{k,j} & \text{if } att_k^L \in S_j \ \wedge \ a_j = 1 \\ C3_{k,j} & \text{otherwise} \end{cases}.$$

From the construction,

$$c^*3_{k,j} = \begin{cases} D_{k,j} & \text{if } att_k^L \in S_{im} \\ E_{k,j} & \text{if } (att_k^L \in S_m \wedge att_k^L \notin S_j) \vee (att_k^L \in S_j \wedge a_j = 0). \\ E'_{k,j} & \text{if } att_k^L \in S_j \ \wedge \ a_j = 1 \end{cases}$$

Now, the simulator can compute the following terms using terms from the assumption:

$$\begin{aligned}
D_{k,j} &= \left(\left(g^{u^*}\prod_{l\in I_S}(g^{y/b_l^2})\right)^{att_k}\left(g^{h^*}\prod_{l\in I_S}(g^{xz/b_le_l})\cdot\prod_{l\in I_S}(g^{y/b_l^2})^{-att_l}\right)\right)^{b_ke_k} g^{-xz_j}\\
&= g^{b_ke_k(u^*att_k+h^*)}\prod_{l\in I_S}g^{xzb_ke_k/b_le_l}\prod_{l\in I_S}g^{yb_ke_k(att_k-att_l)/b_l^2)}g^{-xz_j}\\
&= g^{b_ke_k(u^*att_k+h^*)}\prod_{l\in I_S}\prod_{c\in[P]}g^{xz_cb_ke_k/b_le_l}\prod_{l\in I_S}g^{yb_ke_k(att_k-att_l)/b_l^2}\cdot g^{-xz_j}\\
&= g^{b_ke_k(u^*att_k+h^*)}\prod_{l\in I_S}\prod_{\substack{c\in[P]\\(l,c)\neq(k,j)}}g^{xz_cb_ke_k/b_le_l}\cdot\prod_{l\in I_S}(g^{yb_ke_k/b_l^2})^{att_k-att_l}
\end{aligned}$$

$$\begin{aligned}
E_{k,j} &= \left(\left(g^{u^*}\prod_{l\in I_S}(g^{y/b_l^2})\right)^{F_p(K_{org(k)},att_k)}\left(g^{h^*}\prod_{l\in I_S}(g^{xz/b_le_l})\cdot\prod_{l\in I_S}(g^{y/b_l^2})^{-att_l}\right)\right)^{b_ke_k} g^{-xz_j}\\
&= g^{b_ke_k(u^*F_p(K_{org(k)},att_k)+h^*)}\prod_{l\in I_S}g^{xzb_ke_k/b_le_l}\cdot\prod_{l\in I_S}g^{yb_ke_k(F_p(K_{org(k)},att_k)-att_l)/b_l^2)}g^{-xz_j}\\
&= g^{b_ke_k(u^*F_p(K_{org(k)},att_k)+h^*)}\prod_{l\in I_S}\prod_{c\in[P]}g^{xz_cb_ke_k/b_le_l}\cdot\prod_{l\in I_S}g^{yb_ke_k(F_p(K_{org(k)},att_k)-att_l)/b_l^2}\cdot g^{-xz_j}\\
&= g^{b_ke_k(u^*F_p(K_{org(k)},att_k)+h^*)}\prod_{l\in I_S}\prod_{\substack{c\in[P]\\(l,c)\neq(k,j)}}g^{xz_cb_ke_k/b_le_l}\cdot\prod_{l\in I_S}(g^{yb_ke_k/b_l^2})^{F_p(K_{org(k)},att_k)-att_l}
\end{aligned}$$

$$\begin{aligned}
E'_{k,j} &= (\theta^{(Patt_k'-(P-1)F_p(K_{org(k)},att_k))}h)^{d_k}w^{-s_p}\\
&= g^{b_ke_k(u^*(Patt_k'-(P-1)F_p(K_{org(k)},att_k))+h^*)}\prod_{l\in I_S}g^{xzb_ke_k/b_le_l}\\
&\quad\cdot\prod_{l\in I_S}g^{yb_ke_k((Patt_k'-(P-1)F_p(K_{org(k)},att_k))-att_l)/b_l^2}g^{-xz_j}\\
&= g^{b_ke_k(u^*(Patt_k'-(P-1)F_p(K_{org(k)},att_k))+h^*)}\cdot\prod_{l\in I_S}\prod_{c\in[P]}g^{xz_cb_ke_k/b_le_l}\\
&\quad\cdot\prod_{l\in I_S}g^{yb_ke_k((Patt_k'-(P-1)F_p(K_{org(k)},att_k))-att_l)/b_l^2}g^{-xz_j}\\
&= g^{b_ke_k(u^*(Patt_k'-(P-1)F_p(K_{org(k)},att_k))+h^*)}\cdot\prod_{l\in I_S}\prod_{\substack{c\in[P]\\(l,c)\neq(k,j)}}g^{xz_cb_ke_k/b_le_l}\\
&\quad\cdot\prod_{l\in I_S}(g^{yb_ke_k/b_l^2})^{Patt_k'-(P-1)F_p(K_{org(k)},att_k)-att_l}
\end{aligned}$$

$$Tok = \bigcup_{\substack{att_k\in S,\\ att_k^L\in S_j}} Tok_{k,j} = \bigcup_{\substack{att_k\in S,\\ att_k^L\in S_j}} (Tok1_{k,j}, Tok2_{k,j}, Tok3_{k,j}, Tok4_{k,j})$$

$$Tok1_{k,j} = \left(g^{u^*}\prod_{l\in I_S}(g^{y/b_l^2})\right)^{b_k} = (g^y)^{u^*}\prod_{l\in I_S, l\neq k}(g^{yb_k/b_l^2}) \qquad Tok2_{k,j} = e_k$$

$$Tok3_{k,j} = F(K1_{org(k)}, att_k^L) \qquad Tok4_{k,j} = F_p(K_{org(k)}, att_k).$$

Phase 2: Phase 1 is repeated.

Guess: The adversary outputs a guess b' of b. If $b = b'$, then the challenger outputs 0, i.e., it claims that the challenge term is $R = e(g,g)^{xyz}$. Otherwise, it outputs 1 to indicate that it believes R is a random group element.

If $R = e(g,g)^{xyz}$, then $\mathcal{A}$ played the proper security game, because $C = m_b \cdot R = m_b \cdot e(g,g)^{xys}$. On the other hand, if R is a random term, then all information about the message m_b is lost in the challenge ciphertext. Therefore, the advantage of $\mathcal{A}$ is exactly 0. As a result, if $\mathcal{A}$ breaks the proper security game with a non negligible advantage, then $\mathcal{B}$ has a non negligible advantage in breaking the (q,n,n)-DBTDH assumption. □

Theorem 8.3. *Let $C = (M)_S$ be a MA-OTABE ciphertext. No coalition of at most $|DEC_S| - 1$ parties can learn anything about M.*

The proof of Theorem 8.3 is straightforward and is omitted because of space limitations. It consists of showing that no information about the data-layer components can be inferred from the translation tokens or shares of attribute-layer components that are held by the colluding decryption parties. This conclusion follows from the fact that, in the construction given in Section 7, the local random strings f_k and l_k are chosen independently and uniformly at random, as are the binder-term exponents s_j.

Lemma 8.4. *Let $C = (M)_S$ be a MA-OTABE ciphertext. The proxies in an MA-OTABE scheme cannot learn anything about M, even if they all collude.*

Proof. The proof follows from Theorem 8.3, since every colluding set of at most $|DEC_S| - 1$ parties cannot learn any information about M and, by definition, only $|DEC_S|-1$ proxies participate in each ciphertext's translation. □

Theorem 8.5. *Let F and F_p be two PRFs used in the construction of our MA-OTABE scheme. If F and F_p are secure, then the scheme achieves attribute secrecy.*

Proof. Consider a message M encrypted under a set of attributes S, resulting in a ciphertext C.

Hidden access policy: We consider both the servers and the data users:

CSP, proxies: The set of attributes Y that is stored with the ciphertext on the CSP includes only the obfuscated values of immutable attributes from S. In addition, neither the CSP nor the proxies are given any trapdoors for attributes in S. Thus, Y is hidden from the servers (for more details, the reader is referred to Reference [9]).

Apart from Y, an attribute $att_k \in S$ may appear in the ciphertext only within the attribute components $\{C3_{k,j}\}$ to which the attribute corresponds. Immutable attributes can only appear within $D_{k,j}$, as the exponent of θ inside the local-randomness part. Because each local-randomness part in which the attribute itself appears is blinded by a local, uniformly randomly chosen element known only to the owner, the attribute remains hidden. Mutable attributes in S can appear within $E_{k,j}$ or $E'_{k,j}$ as the exponent of θ inside the local-randomness part or in $Tok3_{k,j}, Tok4_{k,j}$. In both $E_{k,j}$ and $E'_{k,j}$, each local-randomness part in which the attribute itself appears is blinded by a local, uniformly randomly chosen element known only to the owner. Furthermore, in $E_{k,j}$, $Tok3_{k,j}$, and $Tok4_{k,j}$, either att_k or att_k^L only appear in their encrypted form using a keyed PRF with a key that is unknown to the CSP and to all proxies. Last, each PRF-encrypted term inside $Tok3_{k,j}$ and $Tok4_{k,j}$ is encrypted using the public key of the proxy who is allowed to translate att_k ("the translator"). Hence, mutable attributes inside the ciphertext remain hidden as well.

Data users: We start with a definition. "Terminal attributes" are either immutable or the result of an attribute translation performed by one of the proxies. Intuitively, these are the attributes

that the data user will eventually receive with the ciphertext; thus, they must be kept hidden from the user in a manner that enables her to know which attributes she should use for decryption.

Each immutable attribute in S is replaced by the owner at the time of encryption by an obfuscated value of that attribute,
$e((g^{\beta_{aut(k)}})^c, H(att_k))$ (derived from the PEKS construction in Reference [9]), where c is a random number, creating the set Y.

When a proxy P_{org_j} performs a translation of an attribute, it computes an obfuscated value of the new attribute that it created. Only that obfuscated value is attached to the translated partial ciphertext, which it sends to the user as Y_j.

The data user never receives the actual S. Instead, it receives $TR(S) = Y \cup \{Y_j\}_{j \in ORG_S}$ where Y represents immutable attributes in S and $\{Y_j\}$ represent the set of mutable attributes in S. Hence, all the *terminal attributes* are obfuscated and therefore remain hidden from the user. (For more details, see Reference [9].) Note, however, that unlike the servers, data users do hold trapdoors for attributes that appear in their access policies, $H(att_k)^{\beta_{aut(k)}}$, and those trapdoors do leak some information about the attributes in S. Such leakage to the data user is limited to those attributes in S that also appear in the user's access policy; that is, the user learns nothing about attributes in S that are not in her access policy. Such leakage includes, for instance, the ability of the user to know whether an attribute in S that also appears in the user's access policy appeared in previous ciphertexts that she has retrieved from the CSP. Note that the source of such leakage is the transitivity of the equality operation, not the attributes' actual values. The user is not able to learn any of the attributes in S, even for those attributes that appear in her access policy.

Oblivious translation: A proxy P_{org_j} uses its partial ciphertext, tokens, and auxiliary information to perform a translation of an attribute att_k. We claim that neither of the items above reveals the attribute att_k.

Within the partial ciphertext, an attribute att_k such that
$att_k^L \in S_j$ can only appear in the attribute components $\{C3_{k,j}\}$ to which the attribute att_k corresponds, within $E_{k,j}$, as the exponent of θ inside the local-randomness part. However, each local-randomness part within an element $E_{k,j}$ in which the attribute appears is blinded by a local, uniformly randomly chosen element known only to the owner.

Tokens include f_k, θ^{l_k}, which cannot provide any information about att_k. Tokens also include the label and the value of att_k, each encrypted using a different keyed PRF (F and F_p). The keys of both F and F_p are shared between the owner and org_j and are unknown to the proxy. Pieces of auxiliary information are also encrypted using the same keyed PRFs, with the same key used to encrypt the attribute to be translated by the proxy. Hence, if F and F_p are secure, then att_k remains hidden from the proxy. As discussed in Section 6.2, the PRF can be replaced with other transformations that better suit the translation logic of each organization, e.g., order-preserving transformations. Often, such transformations also use PRFs to some extent. In this case, because both the original attribute and the auxiliary information will be encrypted using the same transformation and a key that is unknown to the proxy, the original attribute will remain hidden from the proxy as well.

Last, we would like to note that, although the translation is done obliviously and the proxy does not learn the original attribute, it does leak some information about the original attribute and the auxiliary information to the proxy. For instance, because deterministic encryption is used, the proxy knows which attributes in S_j are used in different ciphertexts—equality can be determined based on the PRF-encrypted value. However, at least some sort of leakage appears to be inherent, because this is exactly what enables the proxy to perform the functionality required by our scheme. Note also that such leakage is limited to the translator. This is because each attribute

component that is meant to undergo translation by a proxy has two encryption layers: In the outer layer, we use strong encryption based on traits of the proposed scheme discussed in Theorem 8.2 or on traits of Π; all system entities except the translator will be unable to decrypt this layer. Only the translator is able to decrypt the outer layer and access the inner layer, which contains the actual attribute encrypted in a "weaker" fashion that enables it to perform the translation.

Attribute privacy: We consider the data owner and the data client:

Data client: Given a translated attribute $v \in \mathcal{U}_{client}$, such that $(T_j^M(v))^{-1}$ is a mutable attribute and $j = org((T_j^M(v))^{-1})$, $(T_j^M(v))^{-1}$ may appear either within the attribute components $\{C3_{k,j}\}$ to which the attribute v corresponds, inside an element $E_{k,j}$, or within $Tok3_{k,j}, Tok4_{k,j}$. In both the ciphertext and the tokens, $(T_j^M(v))^{-1}$ only appears in its encrypted form using a keyed PRF with a key that is unknown to any member of org_{client}. Furthermore, each local-randomness part inside each element $E_{k,j}$ in which $(T_j^M(v))^{-1}$ appears is blinded by a local, uniformly randomly chosen element known only to the owner. Last, PRF-encrypted terms inside $Tok3_{k,j}, Tok4_{k,j}$ are encrypted using the translator's public key and can only be decrypted by the translator.

Hence, for every attribute $v \in \mathcal{U}_{client}$ such that $(T_j^M(v))^{-1}$ is a mutable attribute, org_{client} does not learn $(T_j^M(v))^{-1}$.

Data owner: For every mutable attribute $s \in S$, the *Encrypt*() algorithm given in our construction does not require any knowledge about $T_j^M(s)$, where $j = org(s)$. Furthermore, for each $s \in S$, the resulting ciphertext, C (including both ciphertext's elements and translation tokens), does not contain $T_j^M(s)$. Last, for every mutable attribute $s \in S$, data owners participating in PRShare receive neither terms that include $T_j^M(s)$ nor terms that provide any information about the value of $T_j^M(s)$.

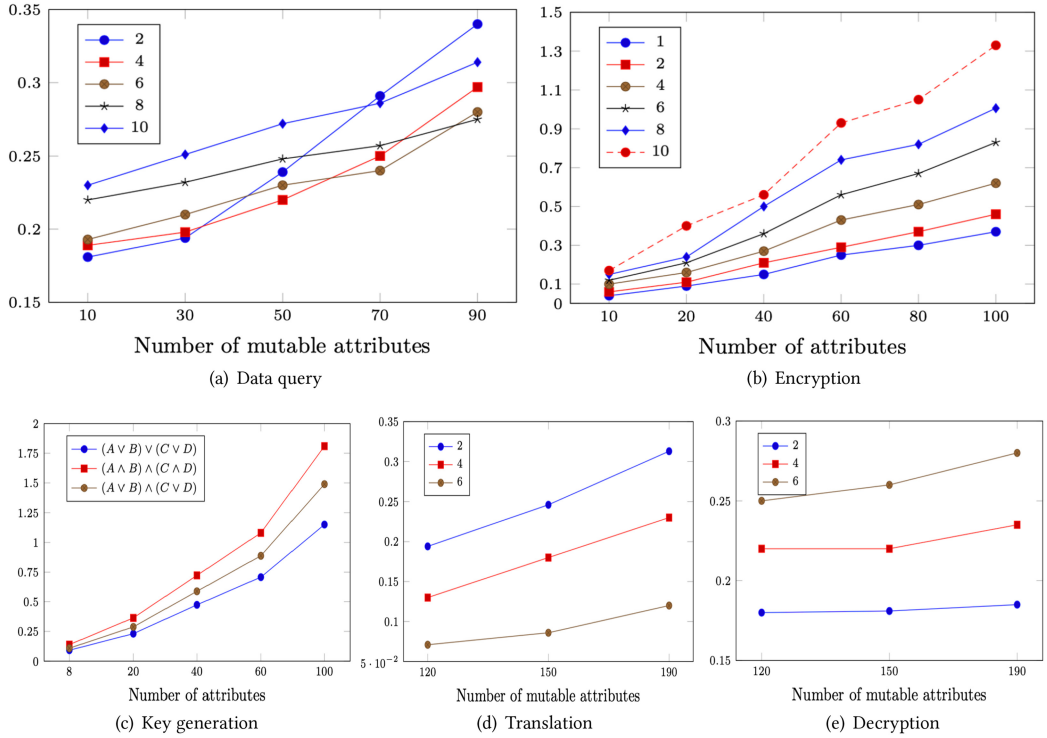
Hence, for every attribute $s \in \mathcal{U}_{owner}$ such that s is a mutable attribute, org_{owner} does not learn $T_j^M(s)$. □

9 IMPLEMENTATION AND EVALUATION

To assess the feasibility of our framework, we implemented the full version of our OTABE scheme using Charm, a framework developed for rapidly prototyping advanced cryptosystems [2]. Charm was used to develop multiple, prominent ABE schemes, including that of Rouselakis and Waters [38, 39]. We instantiated our implementation using a 256-bit Barreto-Naehrig curve. Note that, in our implementation, we translated our scheme to the asymmetric setting, as Charm uses formally asymmetric groups. The assumptions and the security proofs can be translated to the asymmetric setting in a generic way. All our benchmarks were executed on a MacBook Pro laptop with an Intel Core i7 CPU with 16 GB RAM.

We consider a setting with three authorities and policies of size ten where the decryption is always successful, and we use oblivious list membership as our translation operation. We present benchmarks for the overall turnaround time of a data query, i.e., the total time between a user's initiation of a query and her receipt of the *plaintext* records that satisfy it. We also provide benchmarks for the encryption algorithm and the key-generation algorithm, despite the fact that encryptions are done offline, and key requests are significantly less frequent than data queries. The overall runtime, as shown in Figure 1, includes computation, communication, and I/O time. Note that the hidden-access-policy feature is turned off in our experiments.

Recall that each data query entails the following steps. A query is sent to the CSP. The CSP searches for all of the records that satisfy the query. For each ciphertext returned by the search, the CSP sends its partial ciphertexts to the relevant proxies. Each proxy obliviously translates the partial ciphertext it received. The user aggregates all partial ciphertexts and decrypts the result to obtain the plaintext.



(a) Data query (b) Encryption (c) Key generation (d) Translation (e) Decryption

Fig. 1. Typical running times in seconds.

To enable adequate comparison of our OTABE scheme and other ABE schemes, results are given for a single-record data query. Indeed, our running times are similar to other multi-authority ABE schemes, such as Reference [39]. When generalizing our results to the multi-record case, it is important to note that our scheme is highly parallelizable. No TA or proxy needs to coordinate its computation with any other TA or proxy; thus, they can all proceed in parallel. To decrypt, a data user must perform a separate computation for each TA, and all of these computations can be done in parallel. Finally, partial ciphertexts that correspond to different attributes can be translated in parallel.

Figure 1(a) compares the average time of a data query that contains 100 attributes for different numbers of mutable attributes and various sizes of ORG_S. The runtimes are relatively small: It takes only 314 ms to perform a 90-translation data query when $|ORG_S| = 10$. Although there is an increase in runtime as the number of mutable attributes increases, this increase is significantly more noticeable when ORG_S contains fewer proxies. Figure 1(a) also demonstrates an inherent tradeoff between the translation and decryption algorithms: A larger number of proxies results in better load balancing of translation operations, but it also results in more expensive decryption.

Figures 1(d) and (e) show translation and decryption times for varying levels of mutable attributes and sizes of ORG_S. Note that we show results for a high number of mutable attributes to emphasize the translation-decryption tradeoff. However, it is very unlikely that real data records will contain so many mutable attributes; we expect the average number of *mutable* attributes to be around 10–50.

Figure 1(b) shows the average time taken by the encryption algorithm for different numbers of attributes in the ciphertext and various sizes of DEC_S. As expected, encryption time increases

as the number of attributes in the ciphertext increases and as the number of organizations that participate in the decryption increases. Yet, as can be seen, all times are very reasonable compared to the times of other ABE schemes: It only takes 0.46 seconds to encrypt a ciphertext that contains 100 attributes if the number of decrypting entities is 2 and 0.81 seconds if the number of decrypting entities is 6. Bear in mind that encryption is done offline and once per record.

Finally, Figure 1(c) shows the average time taken by the key generation algorithm for various policies. The times are all under 1.81 seconds. This means that, within 2 seconds of a data user's request for a task-related key, she will receive from each authority a key that supports a policy of size 100. Bear in mind that key requests are significantly less frequent than data queries and only occur once per time-limited task.

10 CONCLUSIONS AND OPEN PROBLEM

We have proposed PRShare, an interorganizational data-sharing framework that protects the privacy of data owners, data clients, and data subjects. In designing PRShare, we have introduced the novel concept of *Attribute-Based Encryption With Oblivious Attribute Translation*, which may be of independent interest. In future work, we will consider relaxing one or more assumptions on which PRShare relies. For example, we will explore the use of malicious proxies. Modeling all proxies as malicious instead of semi-trusted would protect our system against proxies who actively deviate from the PRShare protocols. One example of such a deviation is collusion between a user and the proxy servers that may bypass PRShare's revocation mechanism; they could intentionally replace the current time stamp with a future time stamp or intentionally translate the GID attribute into an incorrect GID that belongs to an unauthorized colluding user.

ACKNOWLEDGMENTS

We thank Babis Papamanthou and Satyanarayana Vusirikala for their helpful comments.

REFERENCES

[1] Joseph A. Akinyele et al. 2011. Securing electronic medical records using attribute-based encryption on mobile devices. In *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*. 75–86.

[2] Joseph A. Akinyele et al. 2013. Charm: A framework for rapidly prototyping cryptosystems. *J. Cryptogr. Eng.* 3, 2 (2013), 111–128.

[3] Giuseppe Ateniese et al. 2005. Improved proxy re-encryption schemes with applications to secure distributed storage. In *Proceedings of the 12th Network and Distributed System Security Symposium*. 29–43.

[4] Nuttapong Attrapadung, Benoît Libert, and Elie de Panafieu. 2011. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In *Proceedings of the 14th International Conference on Practice and Theory in Public-Key Cryptography*. Springer LNCS, Vol. 6571, 90–108.

[5] Sana Belguith et al. 2018. PHOABE: Securely outsourcing multi-authority attribute based encryption with policy hidden for cloud assisted IoT. *Comput. Netw.* 133 (2018), 141–156. https://www.sciencedirect.com/science/article/pii/S1389128618300495.

[6] Tara Siegel Bernard, Tiffany Hsu, Nicole Perlroth, and Ron Lieber. 2017. Equifax says cyberattack may have affected 143 million in the U.S. *The New York Times* (Sept. 7, 2017).

[7] John Bethencourt, Amit Sahai, and Brent Waters. 2007. Ciphertext-policy attribute-based encryption. In *Proceedings of the 28th IEEE Symposium on Security and Privacy*. 321–334.

[8] Matt Blaze, Gerrit Bleumer, and Martin Strauss. 1998. Divertible protocols and atomic proxy cryptography. In *Proceedings of the 17th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'98)*. Springer LNCS, Vol. 1403, 127–144.

[9] Dan Boneh et al. 2004. Public-key encryption with keyword search. In *Proceedings of the 23rd Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'04)*. Springer LNCS, Vol. 3027, 506–522.

[10] Dan Boneh, Xuhua Ding, and Gene Tsudik. 2004. Fine-grained control of security capabilities. *ACM Trans. Internet Technol.* 4, 1 (2004), 60–82.

[11] Melissa Chase. 2007. Multi-authority attribute based encryption. In *Proceedings of the 4th Theory of Cryptography Conference*. Springer LNCS, Vol. 4392, 515–534.
[12] Nathan Chenette et al. 2016. Practical order-revealing encryption with limited leakage. In *Proceedings of the 23rd International Conference on Fast Sofware Encryption*. Springer LNCS, Vol. 9783, 474–493.
[13] Xin Dong et al. 2013. Achieving an effective, scalable and privacy-preserving data sharing service in cloud computing. *Comput. Secur.* 42 (2013), 151–164. https://www.sciencedirect.com/science/article/pii/S0167404813001703.
[14] ECPA 1986. Electronic Communications Privacy Act, Public law 99-508. Retrieved from https://it.ojp.gov/PrivacyLiberty/authorities/statutes/1285.
[15] Benjamin Fabian, Tatiana Ermakova, and Philipp Junghanns. 2015. Collaborative and secure sharing of healthcare data in multi-clouds. *Inf. Syst.* 48 (2015), 132–150. https://www.sciencedirect.com/science/article/pii/S030643791400088X.
[16] FCRA 1970. Fair Credit Reporting Act, Public law 91-508. Retrieved from https://www.consumer.ftc.gov/articles/pdf-0111-fair-credit-reporting-act.pdf.
[17] Federal Trade Commission. 2017. Equifax Data Breach Settlement. Retrieved from https://www.ftc.gov/enforcement/cases-proceedings/refunds/equifax-data-breach-settlement.
[18] Jonathan Frankle et al. 2018. Practical accountability of secret processes. In *Proceedings of the 27th USENIX Security Symposium*. 657–674.
[19] David Froelicher et al. 2017. UnLynx: A decentralized system for privacy-conscious data sharing. *Proc. Priv. Enhanc. Technol.* 2017, 4 (2017), 232–250.
[20] Vipul Goyal et al. 2006. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*. 89–98.
[21] Matthew Green and Giuseppe Ateniese. 2007. Identity-based proxy re-encryption. In *Proceedings of the 5th International Conference on Applied Cryptography and Network Security*. Springer LNCS, Vol. 4521, 288–306.
[22] Matthew Green, Susan Hohenberger, and Brent Waters. 2011. Outsourcing the decryption of ABE ciphertexts. In *Proceedings of the 20th USENIX Security Symposium*. 523–538.
[23] Luan Ibraimi et al. 2009. Mediated ciphertext-policy attribute-based encryption and its application. In *Proceedings of the 10th International Conference on Information Security Applications*. 309–323.
[24] Seny Kamara. 2014. Restructuring the NSA metadata program. In *Proceedings of the 2nd Financial Cryptography Workshop on Applied Homomorphic Cryptography and Encrypted Computing*. Springer LNCS, Vol. 8438, 235–247.
[25] Joshua A. Kroll, Edward W. Felten, and Dan Boneh. 2014. Secure Protocols for Accountable Warrant Execution. Retrieved from https://www.cs.princeton.edu/~felten/warrant-paper.pdf.
[26] Junzuo Lai et al. 2013. Attribute-based encryption with verifiable outsourced decryption. *IEEE Trans. Inf. Forens. Secur.* 8, 8 (2013), 1343–1354.
[27] Jiguo Li et al. 2017. Flexible and fine-grained attribute-based data storage in cloud computing. *IEEE Trans. Serv. Comput.* 10, 5 (2017), 785–796.
[28] Ming Li et al. 2010. Securing personal health records in cloud computing: Patient-centric and fine-grained data access control in multi-owner settings. In *Proceedings of the 6th International ICST Conference on Security and Privacy in Communication Networks*. Springer LNICST, Vol. 50, 89–106.
[29] Kaitai Liang et al. 2013. A ciphertext-policy attribute-based proxy re-encryption with chosen-ciphertext security. In *Proceedings of the 5th IEEE International Conference on Intelligent Networking and Collaborative Systems*. 552–559.
[30] Xiaohui Liang et al. 2009. Attribute based proxy re-encryption with delegating capabilities. In *Proceedings of the 4th ACM Symposium on Information, Computer, and Communications Security*. 276–286.
[31] Chang Liu et al. 2015. ObliVM: A programming framework for secure computation. In *Proceedings of the 36th IEEE Symposium on Security and Privacy*. 359–376.
[32] Xuefeng Liu et al. 2013. Mona: Secure multi-owner data sharing for dynamic groups in the cloud. *IEEE Trans. Parallel Distrib. Syst.* 24, 6 (2013), 1182–1191.
[33] Kartik Nayak et al. 2015. GraphSC: Parallel secure computation made easy. In *Proceedings of the 36th IEEE Symposium on Security and Privacy*. 377–394.
[34] Takashi Nishide, Kazuki Yoneyama, and Kazuo Ohta. 2008. Attribute-based encryption with partially hidden encryptor-specified access structures. In *Proceedings of the 6th International Conference on Applied Cryptography and Network Security*. Springer LNCS, Vol. 5037, 111–129.
[35] Rafail Ostrovsky, Amit Sahai, and Brent Waters. 2007. Attribute-based encryption with non-monotonic access structures. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*. 195–203.
[36] Raluca Popa et al. 2011. CryptDB: Protecting confidentiality with encrypted query processing. In *Proceedings of the 23rd ACM Symposium on Operating Systems Principles*. 85–100.
[37] Yogachandran Rahulamathavan et al. 2017. Privacy-preserving blockchain based IoT ecosystem using attribute-based encryption. In *Proceedings of the 11th IEEE International Conference on Advanced Networks and Telecommunications Systems*.

[38] Yannis Rouselakis and Brent Waters. 2013. Practical constructions and new proof methods for large universe attribute-based encryption. In *Proceedings of the 20th ACM Conference on Computer and Communications Security*. 463–474.
[39] Yannis Rouselakis and Brent Waters. 2015. Efficient statically-secure large-universe multi-authority attribute-based encryption. In *Proceedings of the 19th International Conference on Financial Cryptography and Data Security*. 315–332.
[40] Amit Sahai, Hakan Seyalioglu, and Brent Waters. 2012. Dynamic credentials and ciphertext delegation for attribute-based encryption. In *Proceedings of the 32nd Annual International Cryptology Conference on Advances in Cryptology (CRYPTO'12)*. Springer LNCS, Vol. 7417, 199–217.
[41] Amit Sahai and Brent Waters. 2005. Fuzzy identity-based encryption. In *Proceedings of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'05)*. Springer LNCS, Vol. 3494, 457–473.
[42] Aaron Segal, Joan Feigenbaum, and Bryan Ford. 2016. Open, privacy-preserving protocols for lawful surveillance. https://arxiv.org/abs/1607.03659.
[43] Aaron Segal, Joan Feigenbaum, and Bryan Ford. 2016. Privacy-preserving lawful contact chaining [preliminary report]. In *Proceedings of the 15th ACM Workshop on Privacy in the Electronic Society*. 185–188.
[44] Yanfeng Shi et al. 2015. Directly revocable key-policy attribute-based encryption with verifiable ciphertext delegation. *Inf. Sci.* 295 (2015), 221–231. https://www.sciencedirect.com/science/article/pii/S0020025514010020.
[45] Dhinakaran Vinayagamurthy, Alexey Gribov, and Sergey Gorbunov. 2019. StealthDB: A scalable encrypted database with full SQL query support. *Proc. Priv. Enhanc. Technol.* 2019, 3 (2019), 370–388.
[46] Guojun Wang, Qin Liu, and Jie Wu. 2010. Hierarchical attribute-based encryption for fine-grained access control in cloud-storage services. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*. 735–737.
[47] Xuanxia Yao, Zhi Chen, and Ye Tian. 2015. A lightweight attribute-based encryption scheme for the Internet of Things. *Fut. Gener. Comput. Syst.* 49 (2015), 104–112. https://www.sciencedirect.com/science/article/pii/S0167739X14002039.
[48] Shucheng Yu et al. 2010. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *Proceedings of the 29th IEEE Conference on Computer Communications*. 534–542.
[49] Shucheng Yu et al. 2010. Attribute-based data sharing with attribute revocation. In *Proceedings of the 5th ACM Symposium on Information, Computer, and Communications Security*. 261–270.
[50] Lihi Idan and Joan Feigenbaum. 2020. PRShare: A Framework for Privacy-Preserving, Inter organizational Data Sharing. In *Proceedings of the 19th ACM Workshop on Privacy in the Electronic Society*. 137–149.

Received May 2021; revised March 2022; accepted April 2022