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

LIHI IDAN, Yale University, USA
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 incompatible 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 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. 2021. PRShare: A Framework for Privacy-Preserving, Interorganizational Data Sharing. 1, 1 (May 2021), 35 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

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 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 both information about the data and information about the organization's work methodologies and role structure.

Authors' addresses: Lihi Idan, lihi.idan@yale.edu, Yale University, New Haven, USA; Joan Feigenbaum, joan.feigenbaum@yale.edu, Yale University, New Haven, USA.

Manuscript submitted to ACM

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, *data intermediaries*, that enrich 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 proprietary, sensitive 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, *e.g.*, the email message's receiver, are sensitive; 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 cataloged 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 in order 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 a different vocabulary VOC_2 to enrich the shared data, and the client uses a third vocabulary VOC_3 to query and process the data, to manage access control, and to 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 any set of organizations that may need to share data to agree on a shared, standard 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 Subsection 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 not known 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 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 users' 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 multivalued 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 description 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 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 used by the owner 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 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 controlled manner that protects the attributes' confidentiality (*attribute privacy*). Furthermore, attributes and policies can be *dynamically reconfigured*, in the sense that updates are done dynamically, without the need for re-encryption, and offline. No previous proposed ABE scheme achieves all of these properties.

We provide a concrete OTABE scheme and prove it 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*. In order to obtain these features, we leverage our OTABE scheme's translation technique. OTABE also enables *division of trust* (multiple independent authorities authorize data access) and *data centricity* (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 encryption time. Furthermore, the owner does not need to authorize or serve clients' data-access queries. Previous ABE schemes achieved some of these desirable properties of our OTABE construction, but 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. Our techniques are useful in scenarios in which there are legally mandated, general principles that govern access to sensitive data, but instantiating those principles in the form of efficiently decidable rules requires data-specific and dynamically changing knowledge. We give two examples in Subsection 2.2.

Paper outline: Section 2 presents related work and two uses cases. In Section 3, we give the definition of attribute translation and the algorithms used in our 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 detai. 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], 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]; they proposed cryptographic protocols that

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	✓

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 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 and key-policy ABE constructions, including those in [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.

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]. Proxies in this scenario 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

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

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 surface 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, she must know AS_1 *and* AS_2. Thus, unlike OTABE, ABPRE does not support multiple vocabularies and can not provide attribute privacy.

In an ABPRE scheme, re-encryption keys are issued to a proxy on a per-access-policy basis. In order 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 in order 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

In order to motivate the notion of OTABE and illustrate its applicability in real-world scenarios, we provide two examples.

Law-enforcement agencies: The Electronic Communications Privacy Act (ECPA) [14] was passed to protect the privacy rights of ISPs' 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 is to deploy a policy that restricts such data sharing to the minimum needed in order to perform 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, which is the content of an email message, encrypted under a set of metadata attributes, *e.g.*, sender's and receiver's email addresses, date, subject line, *etc.* The client is an LE

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

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 both private information about the ESP's customers or proprietary information of the ESP itself. 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.

TAs 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, *etc.* to assess

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 consumers' credit reports. This access is not limited to reports on the company's customers; it may include reports on large sets of potential customers. In order to create pre-screened offers and market them to potential customers, an insurance company is allowed to access consumers' credit reports and to share information with credit-card issuers, banks, other insurance companies, *etc.* However, access rights to credit reports are limited by the FCRA to information for which an insurance company has a *permissible purpose*. 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, credit-utilization ratio, date and time of the report's creation, CRA-internal statistics, *etc.*

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. In order 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. TAs 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, an insurance company does not want to reveal how it makes business decisions regarding which consumers are considered "qualified" for pre-screened offers, *etc.* 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 translates the numeric CREDIT-UTILIZATION-RATIO attribute into a binary attribute IS-CREDIT-RATIO-LESS-THAN-AVERAGE by obliviously comparing the consumer's

[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].

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 hidden to the extent required. Moreover, sensitive information about consumers whose records *are* decrypted 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

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 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, attributes' 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, on the other hand, 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 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 cannot learn the attributes that are attached to a ciphertext but is able to determine which attributes are needed to perform the decryption. 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 some privacy, 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 Subsection. 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 in order to succeed in decrypting, and the data owner can always serve as a TA or delegate to a TA that it trusts not to permit decryptions that it wishes to forbid.

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 trade-offs 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 the set 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 Subsection 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.

Data users: Data users are employees of org_{client} who need access to data records stored by org_{owner} in order to perform daily tasks. 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}.

Trusted authorities: TAs 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.

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 byproduct 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 finer-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 Subsection 7.2.

4.3 Main flows

The system model consists of an encryption flow, a data flow, 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: In order 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. Lastly, 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 that the ciphertext is associated with.

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, timestamp, *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

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

Data flow: 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 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 paper 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 $MAP : \mathcal{M} \times \mathcal{U}_{owner} \rightarrow \mathcal{U}_{client}$ that maps attributes in org_{owner}'s vocabulary (corresponding to data records' payloads $M \in \mathcal{M}$) to attributes in org_{client}'s vocabulary. 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, the encryptor does not learn $MAP(M, s)$, the translated value of the attribute s with respect to M.
- For every data record's payload M and every attribute $v \in \mathcal{U}_{client}$, if $MAP^{-1}(M, v)$ is mutable, data users and TAs do not learn $MAP^{-1}(M, v)$, the original value of the attribute v with respect to M.

The following observations about our threat model, which considers external adversaries as well as the parties presented in Subsection 4.1, are natural aspects of the security definitions and results presented in Subsection 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 held by each organization that is used for attribute translation. We assume that the owner is honest but curious.

No organization fully trusts its proxy server. CSPs and proxies in our framework, which we assume to be honest but curious, 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 in our system, who are assumed to be malicious, 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.

5.2 Definitions

We start by presenting the definition of selective security for our MA-OTABE 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, the challenger sends PK_i to the adversary. If Aut_i is corrupted, 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 it has MSK_i, 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 $att_{LIMIT} = \langle DATE, <, tl_t \rangle$. In addition, 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 $u \notin RL$, 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 $S \not\models AC'$, 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, 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 $a_j = 1$, 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 $\Pr[b' = b] - 0.5$.

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 [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 $b = 0$, 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 inbspired 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 [38] is single-authority; we extend it here to a multi-authority scheme.

Ciphertext composition in [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 vocabulary and hence needs to undergo translation before it can be used for decryption by u, using 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 $att_4 \in U_{client}$.

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

In order to encrypt M, the owner produces a two-level ciphertext; it is similar to the one in [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 in [38]. In [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 [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 PRF, but, as explained below, OTABE can accommodate a richer set of transformations in order 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 Subsection 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 of attributes S 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 PEKS construction given in [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.

6.2 Other components of PRShare

PRShare combines the MA-OTABE construction in Subsection 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 $F : \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^*$.

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

- Collision-resistant hash function: If the parties wish to use the hidden-access-policy feature, 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 Subsection 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, 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 that 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]. Subsection 7.2 contains concrete examples of attribute translation.

The full construction in Subsection 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 encryption time. 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 encryption time, the owner uses the public keys of all $t \in T$ to create $C0$, which is the data layer. It creates the rest of the ciphertext's components exactly as they are created in Subsection 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, issued by each TA in T' according to the *keygen*() algorithm, and dummy secret keys, 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 key 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 searchable encryption (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, storage and processing of the data records (aka "payloads") is decoupled from 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 her 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, she will learn nothing from the encrypted payloads and attributes that the CSP sends her.

Finally, 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 shared among the relevant entities (owner, CSP, and clients) or a more sophisticated schemes that provides stronger security and privacy guarantees.

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}, 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, w randomly from $G1$. It returns the global public key PK as follows:

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

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 $\{d_k = f_k * l_k\}_{k \in [R]}$. The encryptor determines, according to the nature of attributes in S, 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:

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

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

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, as well as 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:

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

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, 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: $att_{LIMIT} =< DATE, <, tl_t >$. Lastly, 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 which associates rows of M_i to 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:

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

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, encrypted under a set of attributes S and retrieved by the CSP, the CSP sends to each proxy, P_{org_p}:

$$C^p = (\{C3_{k,p} | att_k^L \in S_p\}, P, sig_p^{encryptor}, Tok_p)$$

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 Subsection 7.2. It then computes a new value for $E_{k,p}$, $E'_{k,p}$:

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

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 denote the set of obfuscated value corresponding to translated attributes by proxy p as Y_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:

$$C'^p = (C'3_p, sig_p, W_p, Y_p) \qquad C'3_p = \{C'3_{k,p} | att_k^L \in S_p\} = \{E'_{k,p} | att_k^L \in S_p\}$$

Decrypt$(PK, \{SK_{i,u,t}\}, C^u, \{C'^j | j \in ORG_S\}) \Rightarrow M$: The decryption algorithm for data record's payload M, 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: a partial ciphertext, C^u, received directly from the CSP and $|ORG_S| = P - 1$ translated partial ciphertexts $\{C'^j | j \in ORG_S\}$, received from each one of the proxies in ORG_S, $C'^j = (C'3_j, sig_j, W_j, Y_j)$. 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 and updated version of $C3$, $C'3$:

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

The user extracts $C0, C1, C2$ from C^u and merges those 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 $Translate()$ algorithm given in our construction assumes the existence of a set of translation functions, $\{T_j | j \in ORG\}$. Each function is determined separately by each organization org_j and determines how to translate attributes in S_j.

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 only changing its value. A translation may require auxiliary information, 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 (a number, a list *etc*), the "auxiliary information," that is encrypted using the same transformation. Such an oblivious operation can be a comparison, equality test, list membership test, keyword search *etc.* 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 and without learning 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, where entries represent obfuscated labels and values contain the translation logic and auxiliary information used for 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 in its hands. 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), in order to support *oblivious* translation, a function T_j is a two-argument function, neither contains the actual original attribute, but instead, an obfuscated version of both 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 for supporting multiple vocabularies is that, although the encryption of a data record's payload is done by the owner once, the relevance of the data record to the client changes over time. In ABE terms, that means that while the set of attributes under which a ciphertext is encrypted, taken from one vocabulary, does not change, the question whether or not this set satisfies a given access policy, taken from another vocabulary, does change over time. Furthermore, such decision, of whether or not a ciphertext is relevant to the client at a given point in time is made using external information, the "auxiliary information," that is related to one or more of the owner's, client's, and intermediaries' professional domains. Because the auxiliary information changes over time, so does the decision 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, but should rather be dynamically translated, only when a data user needs to access that data record. OTABE supports such dynamic attributes, as shown below.

We consider here two examples, representing common translation operations used for translating 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 Subsection 2.2. Two pieces of metadata that ESPs collect about their customers' email messages are the sender and receiver of the email. 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 such as phone number or email address are known in advance), a raw email address will be meaningless in terms of justification for decryption, and therefore cannot be used for determining 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 as well as protects innocent citizens' privacy and thus can be included in the key in order to determine whether an 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 over time 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:

$for\ l\ in\ watchlist:$

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

Where $E_{client}(watchlist)$ represents the resulting, encrypted watchlist, containing multiple "sender" attributes, and 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 numeric auxiliary-information pieces; each piece usually represents either a 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 ORE scheme presented in [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 consider the insurance-company example discussed in Subsection 2.2. We consider the attribute: "credit utilization ratio," used by the CRA to store credit reports. Such an attribute, however, cannot be used in the secret keys issued by the insurance company to its employees, as a raw number will be meaningless in terms of determining 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 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 as well as protects consumers' privacy to the extent possible. Therefore, it can be included in employees' keys in order to determine whether the insurance company finds the data subject to be a good enough candidate for an insurance offer, *i.e.*, whether access to the data subject's credit report is justified. In this case the translation will be made by the credit card company's proxy, acting as an intermediary, by obliviously comparing the number that represents the 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, constitutes proprietary information of the company and should not be revealed to other organizations.

As in the previous example, the auxiliary information (the utilization ratio's average) may change periodically. Thus, whether or not the utilization ratio of a data subject is above average may change over time. This is why such an attribute can only be translated dynamically, when an employee submits an access request for that specific credit report.

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 timestamp, 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 a data user's GID, u, who sent a data retrieval request, $P_{org_{client}}$ performs the following: it first checks whether u appears in org_{client}'s revocation list. If so, 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, 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. Thus, the proxy only needs to know the original attributes' obfuscated labels in order to perform the translation.

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 3, it is enough to prove that (q, n, n)-DBTDH $\Rightarrow$ q-DBDH.

Given a distinguisher D_1 which 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 which 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:

$$\Omega_2' = \{h1, h2\} \cup \{h3, h4, h5 || \forall l \in [q]\} \cup \{h6, h7, \Psi'_{l,f} | \forall l, f \in [q], l \neq f\}$$

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 which is able to tell q-DBDH term from a random term with the 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^*, which 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^*, 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. Then it may issue any polynomial number of private key queries, for tuples of (access structure, GID, task identifier), and sends those 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, as it has MSK_i, 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:

[5]For simplicity, we prove our attribute-secrecy related claims separately, in Theorem 3. We also omit the signatures that are attached to some of the messages in our construction

- If $S \models AC'$ and $u \notin RL$, the simulator will abort.
- If $S \models AC'$ and $u \in RL$, 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 $S \not\models AC'$, 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.

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

The simulator then chooses uniformly at random $n-1$ random elements in Z_p, v_i, 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:

$$\lambda_r =< M_r^{AC^*}, (MSK_h\beta + (0, v_2, \ldots, v_n)^{\perp}) >= xy < M_r^{AC^*}, \beta > + x \sum_{i \in Aut_c} n_i < M_r^{AC^*}, \beta > + < M_r^{AC^*}, (0, v_2, \ldots, v_n)^{\perp} >$$

$$= xy < M_r^{AC^*}, \beta > + x \sum_{i \in Aut_c} n_i < M_r^{AC^*}, \beta > + \lambda'_r$$

Now, let us see how the simulator computes the secret key for $r \in A$: From 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, SK^3 as in the $KeyGen$ algorithm:

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

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

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

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: From 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:

$$SK^1 = g^{\lambda_r} w^{t_r} = g^{\lambda'_r} \prod_{l\in[n]} (g^{x^2 z b_l})^{<M_r^{AC^*},\beta>/(\rho(r)-att_l)} \cdot g^{xt'_r}$$

$$SK^2 = (\theta^{\rho(r)}h)^{-t_r} = g^{\sum_{i\in Aut_c}(n_i)<M_r^{AC^*},\beta>(\rho(r)u^*+h^*)} (g^y)^{<M_r^{AC^*},\beta>(\rho(r)u^*+h^*)} \cdot \prod_{l\in I_S} (g^{xzb_l})^{-(\rho(r)u^*+h^*)<M_r^{AC^*},\beta>/(\rho(r)-att_l)}$$

$$\cdot \prod_{(l,f)\in I_S} (g^{(xz)^2 b_f/b_l e_l})^{-<M_r^{AC^*},\beta>/(\rho(r)-att_f)} \cdot \prod_{l\in I_S} (g^{y^2/b_l^2})^{<M_r^{AC^*},\beta>(\rho(r)-att_l)} \cdot \prod_{l\in I_S} (g^{xz/b_l e_l})^{\sum_{i\in Aut_c}(n_i)<M_r^{AC^*},\beta>}$$

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

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

Therefore, in both cases the simulator can reply to the adversary's query with the entire secret key. Note, that since $AC, AC' \subseteq \mathcal{U}_{client}$, and $S_{GID}, S_{LIMIT} \in \mathcal{U}_{client}$, $AC^* \subseteq \mathcal{U}_{client}$, and so does the secret key given to the adversary. In addition, all the secret key's terms, both for A and for 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 $C^j = (\{C3_{k,j} | att^L_k \in S_j\}, P, Tok_j)$ 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 $att^L_k \in S_j$, the simulator has two attributes in its hands: 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 $C'3_j = \{C'3_{k,j} | att^L_k \in S_j\}$ 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:

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

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

From the construction,

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

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

$$D_{k,j} = ((g^{u^*} \prod_{l \in I_S} (g^{y/b_l^2}))^{att_k} (g^{h^*} \prod_{l \in I_S} (g^{xz/b_l e_l}) \cdot \prod_{l \in I_S} (g^{y/b_l^2})^{-att_l}))^{b_k e_k} g^{-xz_j} =$$

$$g^{b_k e_k (u^* att_k + h^*)} \prod_{l \in I_S} g^{xz b_k e_k / b_l e_l} \prod_{l \in I_S} g^{y b_k e_k (att_k - att_l)/b_l^2}) g^{-xz_j} =$$

$$g^{b_k e_k (u^* att_k + h^*)} \prod_{l \in I_S} \prod_{c \in [P]} g^{xz_c b_k e_k / b_l e_l} \prod_{l \in I_S} g^{y b_k e_k (att_k - att_l)/b_l^2} \cdot g^{-xz_j} =$$

$$g^{b_k e_k (u^* att_k + h^*)} \prod_{\substack{l \in I_S \\ (l,c) \neq (k,j)}} \prod_{c \in [P]} g^{xz_c b_k e_k / b_l e_l} \cdot \prod_{l \in I_S} (g^{y b_k e_k / b_l^2})^{att_k - att_l}$$

$$E_{k,j} = ((g^{u^*} \prod_{l \in I_S} (g^{y/b_l^2}))^{F_p(K_{org(k)}, att_k)} (g^{h^*} \prod_{l \in I_S} (g^{xz/b_l e_l}) \cdot \prod_{l \in I_S} (g^{y/b_l^2})^{-att_l}))^{b_k e_k} g^{-xz_j} =$$

$$g^{b_k e_k (u^* F_p(K_{org(k)}, att_k) + h^*)} \prod_{l \in I_S} g^{xz b_k e_k / b_l e_l} \cdot \prod_{l \in I_S} g^{y b_k e_k (F_p(K_{org(k)}, att_k) - att_l)/b_l^2}) g^{-xz_j} =$$

$$g^{b_k e_k (u^* F_p(K_{org(k)}, att_k) + h^*)} \prod_{l \in I_S} \prod_{c \in [P]} g^{xz_c b_k e_k / b_l e_l} \cdot \prod_{l \in I_S} g^{y b_k e_k (F_p(K_{org(k)}, att_k) - att_l)/b_l^2} \cdot g^{-xz_j} =$$

$$g^{b_k e_k (u^* F_p(K_{org(k)}, att_k) + h^*)} \prod_{\substack{l \in I_S \\ (l,c) \neq (k,j)}} \prod_{c \in [P]} g^{xz_c b_k e_k / b_l e_l} \cdot \prod_{l \in I_S} (g^{y b_k e_k / b_l^2})^{F_p(K_{org(k)}, att_k) - att_l}$$

$$E'_{k,j} = (\theta^{(Patt_k' - (P-1)F_p(K_{org(k)}, att_k))} h)^{d_k} w^{-s_p} =$$

$$g^{b_k e_k (u^* (Patt_k' - (P-1)F_p(K_{org(k)}, att_k)) + h^*)} \prod_{l \in I_S} g^{xz b_k e_k / b_l e_l} \cdot \prod_{l \in I_S} g^{y b_k e_k ((Patt_k' - (P-1)F_p(K_{org(k)}, att_k)) - att_l)/b_l^2} g^{-xz_j} =$$

$$g^{b_k e_k (u^* (Patt_k' - (P-1)F_p(K_{org(k)}, att_k)) + h^*)} \cdot \prod_{l \in I_S} \prod_{c \in [P]} g^{xz_c b_k e_k / b_l e_l} \cdot \prod_{l \in I_S} g^{y b_k e_k ((Patt_k' - (P-1)F_p(K_{org(k)}, att_k)) - att_l)/b_l^2} g^{-xz_j} =$$

$$g^{b_k e_k (u^* (Patt_k' - (P-1)F_p(K_{org(k)}, att_k)) + h^*)} \cdot \prod_{\substack{l \in I_S \\ (l,c) \neq (k,j)}} \prod_{c \in [P]} g^{xz_c b_k e_k / b_l e_l} \cdot \prod_{l \in I_S} (g^{y b_k e_k / b_l^2})^{Patt_k' - (P-1)F_p(K_{org(k)}, att_k) - att_l}$$

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

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

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

Phase 2: Phase 1 is repeated.

Guess: The adversary outputs a guess b' of b. If $b = b'$ 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 2 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, see [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 random 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 random chosen element, known only to the owner. Furthermore, In both $E_{k,j}$, $Tok3_{k,j}$ 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, or to any proxy. Lastly, 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 by defining the term "terminal attributes." Terminal attributes include either immutable attributes or attributes that are the result of an attribute translation performed by one of the proxies. Intuitively, those are the attributes that the data user will eventually receive with the ciphertext, and thus must be kept hidden from the user, in such 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 encryption time by an obfuscated value of that attribute, $e((g^{\beta_{aut(k)}})^c, H(att_k))$, derived from the PEKS construction in [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, and only that obfuscated value is attached to the translated partial ciphertext that 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 [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 the user has retrieved from the CSP (we 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, its tokens, and auxiliary information in order to perform a translation of an attribute att_k. We claim that neither of the items above reveals the attribute att_k.

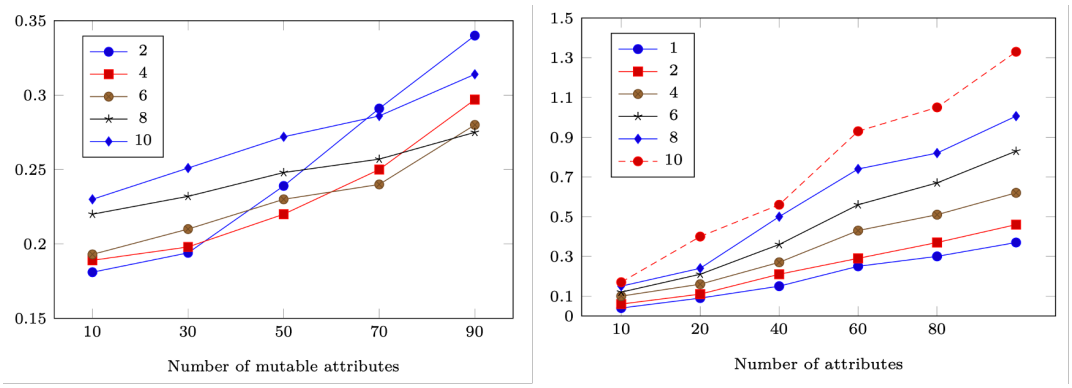
Within the partial ciphertext, an attribute att_k such that $att_k^L \in S_j$ can only appear in the attribute components $\{C3_{k,j}\}$ to which the attribute att_k corresponds, within $E_{k,j}$, as the exponent of θ inside the local-randomness part. However, each local-randomness part within an element $E_{k,j}$ in which the attribute appears is blinded by a local, uniformly random 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. Auxiliary information pieces are also encrypted using the same keyed PRFs, with the same key used for encryption of the attribute to be translated by the proxy. Hence, if F and F_p are secure, att_k remains hidden from the proxy. As discussed in Subsection 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, using a key that is unknown to the proxy, the original attribute will remain hidden from the proxy as well.

Lastly, we would like to note that though the translation is done obliviously and the proxy does not learn the original attribute, it does leak some information about the original attribute, as well as the auxiliary information, to the proxy. For instance, the proxy is able to know, because of the deterministic encryption, which attributes in S_j are used in different ciphertexts, as equality can be determined based on the PRF-encrypted value. However, at least some sort of leakage appears to be inherent, as this is exactly what enables the proxy to perform the functionality required from it in our scheme. Also note, 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 our proposed scheme as discussed in Theorem 1 or on traits of Π; no system entities except the translator can 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" encryption that enables it to perform the translation.

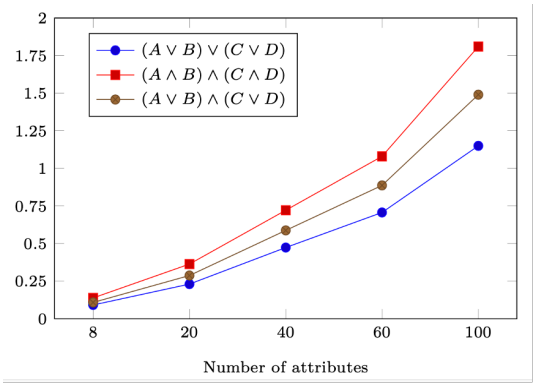
Attribute privacy: We consider the data owner and the data client:

Data client: Given a translated attribute $v \in \mathcal{U}_{client}$, such that $MAP^{-1}(M, v)$ is a mutable attribute, $MAP^{-1}(M, v)$ 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, $MAP^{-1}(M, v)$ 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



(a) Data query

(b) Encryption



(c) Key generation

Fig. 1. Typical running times in seconds

part inside each element $E_{k,j}$ in which $MAP^{-1}(M, v)$ appears, is blinded by a local, uniformly random chosen element, known only to the owner. Lastly, 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 $MAP^{-1}(M, v)$ is a mutable attribute, org_{client} does not learn $MAP^{-1}(M, v)$.

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

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

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 existing ABE schemes, including that of Rouselakis and Waters [38]. We instantiated our implementation using a 256-bit Barreto-Naehrig (BN) 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.

We consider a setting with three authorities and policies of size ten, where the decryption is always successful, and use oblivious list membership as our translation operation. We present benchmarks for three operations. The first is the overall turnaround time of a data query, *i.e.*, the total time between a user's initiation of a query and her receiving 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.

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 [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. In order 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 314ms 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 trade-off 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.

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 other ABE schemes: it only takes 0.46s to encrypt a ciphertext that contains 100 attributes if the number of decrypting entities is 2, and 0.81s if the number of decrypting entities is 6. Bear in mind, that encryption is done once per record, and offline.

Finally, Figure 1(c) shows the average time taken by the key generation algorithm for various policies. The times are all under 1.81s. This means that, within less than two seconds from 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 that PRShare relies on; for example, we will explore the use of malicious proxies.

11 ACKNOWLEDGMENTS

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

REFERENCES

[1] Joseph A. Akinyele et al. 2011. Securing Electronic Medical Records Using Attribute-based Encryption on Mobile Devices. In *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. *Journal of Cryptographic Engineering* 3, 2 (2013), 111–128.
[3] Giuseppe Ateniese et al. 2005. Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage. In *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 *14th International Conference on Practice and Theory in Public-Key Cryptography*. Springer LNCS volume 6571, 90–108.
[5] Sana Belguith et al. 2018. PHOABE: Securely outsourcing multi-authority attribute based encryption with policy hidden for cloud assisted IoT. *Computer Networks* 133 (2018), 141–156.
[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 *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 *17th EUROCRYPT*. Springer LNCS volume 1403, 127–144.
[9] Dan Boneh et al. 2004. Public-Key Encryption with Keyword Search. In *23rd EUROCRYPT*. Springer LNCS volume 3027, 506–522.
[10] Dan Boneh, Xuhua Ding, and Gene Tsudik. 2004. Fine-grained control of security capabilities. *ACM Transactions on Internet Technology* 4, 1 (2004), 60–82.
[11] Melissa Chase. 2007. Multi-authority Attribute Based Encryption. In *4th Theory of Cryptography Conference*. Springer LNCS volume 4392, 515–534.
[12] Nathan Chenette et al. 2016. Practical Order-Revealing Encryption with Limited Leakage. In *23rd International Conference on Fast Sofware Encryption*. Springer LNCS volume 9783, 474–493.
[13] Xin Dong et al. 2013. Achieving an Effective, Scalable and Privacy-preserving Data Sharing Service in Cloud Computing. *Computers and Security* 42 (2013), 151–164.
[14] ECPA 1986. Electronic Communications Privacy Act, Public law 99-508. 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. *Information Systems* 48 (2015), 132–150.
[16] FCRA 1970. Fair Credit Reporting Act, Public law 91-508. https://www.consumer.ftc.gov/articles/pdf-0111-fair-credit-reporting-act.pdf.
[17] Federal Trade Commission. 2017. Equifax Data Breach Settlement. https://www.ftc.gov/enforcement/cases-proceedings/refunds/equifax-data-breach-settlement.
[18] Jonathan Frankle et al. 2018. Practical Accountability of Secret Processes. In *27th USENIX Security Symposium*. 657–674.
[19] David Froelicher et al. 2017. UnLynx: A Decentralized System for Privacy-Conscious Data Sharing. *Proceedings on Privacy Enhancing Technologies* 2017, 4 (2017), 232–250.
[20] Vipul Goyal et al. 2006. Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data. In *13th ACM Conference on Computer and Communications Security*. 89–98.
[21] Matthew Green and Giuseppe Ateniese. 2007. Identity-Based Proxy Re-encryption. In *5th International Conference on Applied Cryptography and Network Security*. Springer LNCS volume 4521, 288–306.

[22] Matthew Green, Susan Hohenberger, and Brent Waters. 2011. Outsourcing the Decryption of ABE Ciphertexts. In *20th USENIX Security Symposium*. 523–538.

[23] Luan Ibraimi et al. 2009. Mediated Ciphertext-Policy Attribute-Based Encryption and Its Application. In *10th international conference on information security applications*. 309–323.

[24] Seny Kamara. 2014. Restructuring the NSA Metadata Program. In *2nd Financial Cryptography Workshop on Applied Homomorphic Cryptography and Encrypted Computing*. Springer LNCS volume 8438, 235–247.

[25] Joshua A. Kroll, Edward W. Felten, and Dan Boneh. 2014. Secure protocols for accountable warrant execution. https://www.cs.princeton.edu/~felten/warrant-paper.pdf.

[26] Junzuo Lai et al. 2013. Attribute-Based Encryption With Verifiable Outsourced Decryption. *IEEE Transactions on Information Forensics and Security* 8, 8 (2013), 1343–1354.

[27] Jiguo Li et al. 2017. Flexible and Fine-Grained Attribute-Based Data Storage in Cloud Computing. *IEEE Transactions on Services Computing* 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 *6th International ICST Conference on Security and Privacy in Communication Networks*. Springer LNICST volume 50, 89–106.

[29] Kaitai Liang et al. 2013. A Ciphertext-Policy Attribute-Based Proxy Re-Encryption with Chosen-Ciphertext Security. In *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 *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 *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 Transactions on Parallel and Distributed Systems* 24, 6 (2013), 1182–1191.

[33] Kartik Nayak et al. 2015. GraphSC: Parallel Secure Computation Made Easy. In *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 *6th International Conference on Applied Cryptography and Network Security*. Springer LNCS volume 5037, 111–129.

[35] Rafail Ostrovsky, Amit Sahai, and Brent Waters. 2007. Attribute-Based Encryption with Non-Monotonic Access Structures. In *14th ACM Conference on Computer and Communications Security*. 195–203.

[36] Raluca Popa et al. 2011. CryptDB: Protecting Confidentiality with Encrypted Query Processing. In *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 *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 *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 *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 *32nd CRYPTO*. Springer LNCS volume 7417, 199–217.

[41] Amit Sahai and Brent Waters. 2005. Fuzzy Identity-Based Encryption. In *24th EUROCRYPT*. Springer LNCS volume 3494, 457–473.

[42] Aaron Segal, Joan Feigenbaum, and Bryan Ford. 2016. Open, privacy-preserving protocols for lawful surveillance. *CoRR* abs/1607.03659 (2016). http://arxiv.org/abs/1607.03659

[43] Aaron Segal, Joan Feigenbaum, and Bryan Ford. 2016. Privacy-Preserving Lawful Contact Chaining [Preliminary Report]. In *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. *Information Sciences* 295 (2015), 221–231.

[45] Dhinakaran Vinayagamurthy, Alexey Gribov, and Sergey Gorbunov. 2019. StealthDB: a Scalable Encrypted Database with Full SQL Query Support. *Proceedings on Privacy Enhancing Technologies* 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 *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. *Future Generation Computer Systems* 49 (2015), 104–112.

[48] Shucheng Yu et al. 2010. Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing. In *29th IEEE Conference on Computer Communications*. 534–542.

[49] Shucheng Yu et al. 2010. Attribute-based data sharing with attribute revocation. In *5th ACM Symposium on Information, Computer, and Communications Security*. 261–270.