# DataBlinder: A distributed data protection middleware supporting search and computation on encrypted data

Emad Heydari Beni, Bert Lagaisse, Wouter Joosen
{firstname.lastname}@cs.kuleuven.be
imec-DistriNet, KU Leuven, Belgium

Abdelrahaman Aly
abdelrahaman.aly@esat.kuleuven.be
imec-COSIC, KU Leuven, Belgium

Michael Brackx
michael.brackx@unifiedpost.com
UnifiedPost, Belgium

## Abstract

Business application owners want to outsource data storage, including sensitive data, to the public cloud for economical reasons. This is often challenging since these businesses are and remain responsible for regulatory compliance and data protection, even though cloud providers may do their best to offer (data) protection. Meanwhile, data protection techniques evolve and get better because of continuous research and improvement of advanced encryption. Numerous cryptographic tactics have been proposed, e.g., searchable symmetric encryption (SSE) and homomorphic encryption (HE), that support search and aggregation functions on encrypted data. Each of these tactics has a trade-off between *security*, *performance* and *functionality*, but there is no one-size-fits-all solution. For the application developer, the underpinning concepts of these tactics are complex to comprehend, complex to integrate in a distributed application, and prone to implementation mistakes.

In this paper we present DataBlinder, a distributed data access middleware that provides *crypto agility* by means of configurable *fine-grained* data protection at the application level. DataBlinder supports adaptive runtime selection of data protection tactics, and offers a plugin architecture for such tactics based on a key abstraction model for protection level, performance and supported query functionality. We have developed this middleware in close collaboration with businesses that face these challenges and offer cloud-based applications in e-finance, and e-health, by implementing and integrating state-of-the-art cryptographic schemes to DataBlinder. This paper illustrates the case of medical data protection with FHIR-compliant [30] medical data.

***CCS Concepts*** • **Security and privacy → Management and querying of encrypted data**; • **Information systems →** *Middleware for databases.*

***Keywords*** security and privacy, middleware, data protection

**ACM Reference Format:**
Emad Heydari Beni, Bert Lagaisse, Wouter Joosen, Abdelrahaman Aly, and Michael Brackx. 2019. DataBlinder: A distributed data protection middleware supporting search and computation on encrypted data. In *20th International Middleware Conference Industrial Track (Middleware Industry '19), December 9–13, 2019, Davis, CA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3366626.3368132


*Middleware Industry '19, December 9–13, 2019, Davis, CA, USA*

ACM ISBN 978-1-4503-7041-7/19/12…$15.00
https://doi.org/10.1145/3366626.3368132

## 1 Introduction

Software service providers use public cloud computing infrastructure to expand their computational capabilities and storage capacity. Outsourcing customer data to the public cloud is not always feasible for all domains of industry, especially the healthcare sector. The reason lies in the fact that a significant amount of their customer data is sensitive. Data protection regulations trigger companies to further enrich their security countermeasures to protect sensitive data, notably personally identifiable information. For instance, regulations oblige healthcare companies and organizations to notify the authorities regarding any data breach of *unsecured* protected health information (e.g., GDPR [53] and the HITECH Act [45, 46]). They are thus skeptical about employing cloud-based infrastructure and services, in particular, for storage of their critical data.

Healthcare providers would be forced to deploy *data protection mechanisms* that go beyond encryption at rest or transmission. The state-of-practice data protection at rest using standard encryption is insufficient as software services are required to perform operations on encrypted data. They should be able to execute queries like:

- finding the patient with a particular gastric cancer who was admitted to the hospital in 12/05/2012 (boolean search),
- calculating the average heart rate of a patient (aggregate), or
- the number of times that the nurses refilled Doxycycline for a patient (aggregated search).

**Latest advances.** Researchers and practitioners have proposed many searchable encryption (SE) tactics and data protection systems for enabling search and computation on sensitive data in untrusted environments. Followed by Song et. al [54], a prominent body of research has been dedicated to symmetric SE (SSE). The research efforts have focused on defining security notions (e.g., IND-CKA2 [18]), building updatable and scalable schemes [7, 8, 12], optimal locality of encrypted indexes [12, 14, 19], and more complex functionalities such as boolean search [13, 35]. A more practical, albeit less protective mechanisms, are based on property-preserving encryption (PPE), e.g., deterministic encryption (DET) [2], order-preserving encryption (OPE) [1, 4] and order-revealing encryption (ORE) [5] schemes. Furthermore, homomorphic encryption (HE) schemes allow us to operate, i.e., addition and/or multiplication, directly over encrypted data. HE schemes provide either addition or multiplication e.g., Paillier [48] and ElGamal [21]. Somewhat HE (SHE) and Fully HE (FHE) offer some combination of both at the cost of performance e.g., BGV [9] and TFHE [16].

Each of these tactics attempts to find a trade-off between *security*, *performance* and *functionality*. For example, encrypting the whole database (AES128) without searchability in mind provides us with a high degree of security but falls short of performance. Some tactics leak less information than others; among them, there are some with sub-linear search complexity. Each of these schemes offers different functionalities (e.g. equality, (con/dis)junction, etc.). Lastly,

these advanced cryptographic constructions typically have complex designs leading to adoption difficulties by practitioners.

**Challenges.** There are several challenges for the development and application-level integration of such data protection tactics: (i) there is no one-size-fits-all cryptographic scheme which maximizes all three aspects of the security, performance, and functionality trade-off; (ii) integrating data protection tactics in the form of libraries to heterogeneous and polyglot software, e.g., microservice architectures, is prone to mistakes because such systems are developed using various ecosystems of programming languages; (iii) the underpinning concepts and implementation details of cryptographic constructions used in data protection tactics are mostly complex for application developers; in other words, choosing a right scheme as well as a secure and correct implementation are also prone to mistakes; moreover, (iv) developing and incorporating new cryptographic schemes in an existing software stack is not a trivial and straightforward task for cryptographers.

**Contributions.** In this paper we present DataBlinder, a distributed data-access middleware that encapsulates the complexity of data protection tactics. This middleware was developed in the context of an industrial applied research project [32] in close collaboration with software service providers. It enables software service providers to seamlessly outsource sensitive data to the cloud-based services and yet be able to operate on it. The contributions are:

*Adaptive selection of data protection tactics.* We present an abstraction model to reify the data protection concepts, allowing application developers to request for their desired protection level and types of queries. The middleware accordingly selects appropriate tactics satisfying the requirements presented in the policies, and it adaptively loads the right implementation at runtime.

*Extensible and pluggable architecture.* Data protection tactics are subject to change to be more efficient, more secure and/or more expressive. Inspired by the comprehensive categorisation of Fuller et al. [24], we present an abstraction model for data protection tactics to reify their leakage profile, performance metrics and operations. As a result, tactic developers are provided with a set of interfaces based on the required operations using the Service Provider Interface (SPI) pattern, through which they plug in new tactics.

Recent research efforts have attempted to design and build secure database systems, such as CryptDB [52], Blind Seer [49], OSPIR-OXT [12, 13, 22], Arx [3], SisoSPIR [34], EncKV [56], etc. These systems employed different cryptographic constructions, such as SSE, various types of PPE, hardware-assisted approaches based on Trusted Execution Environments (TEEs), Oblivious RAM, and so on. The two key differentiating goals of our contributions in comparison to the prior research are (1) presenting an architecture enabling the software service providers to configure a notion of security with respect to their required operation, and (2) facilitating the current and future tactic extension process, unlike other systems that focused only on application developers and the cloud providers; therefore, our design is extensible and not tied to any specific tactic. More importantly, the adaptive and pluggable architecture take us one step closer to *crypto agility*, i.e., the ability to plug and play cryptographic schemes depending on their evolution in time.

We validated the architecture by implementing several state-of-the-art data tactics leveraging our SPI's, and evaluated on FHIR-compliant [30] medical data. Our performance evaluations show that DataBlinder has limited impact on overall performance.

## 2 Background

Over the past years, researchers and practitioners attempted to make practical SE constructions at the cost of allowing limited and defined information leakage, and yet retaining security in both the snapshot and persistent adversarial model. The snapshot model means the adversary obtains a snapshot of the secure index and the database, a well-motivated model for data breaches in the industry. The persistent model assumes that the adversary can observe all operations of the cloud server but without any interference.

**Searchable encryption (SE).** These schemes generally enable cloud providers to search for user-requested keywords on encrypted data without knowing the search word content and the plaintext data. These constructions are typically built on top secure indexes that reveal no information (or formally defined leakage, also called leakage profile) about the content of search words and data itself. In brief, they typically start with a setup protocol. This protocol generates the required keys, builds the initial index and prepares the cloud and local data stores. Next, the query protocol performs the search query by generating trapdoors (also called tokens) at the application side, which ideally reveal nothing about the search term. Using the trapdoors, the cloud provider can query the secure index by running an algorithm as a part of the search protocol and provide the querier with the encrypted document identifiers. Dynamic schemes include an update protocol for addition, deletion, and modification of the encrypted documents. *An example query can be searching for patients' details such as their health problems.*

**Range queries on encrypted data.** The main goal of such schemes is to allow cloud providers to compare ciphertexts without decryption by applying a *comparison* function. That enables the SE-based systems to build more complex queries such as range queries. Although the practical SE constructions built upon these primitives are recently subject to attacks [28, 29, 37, 43], they are still an ongoing research subject. *An example query can be searching for patients' health problems between particular date ranges.*

**Homomorphic encryption (HE).** Homomorphic encryption schemes encrypt data in a way that their underpinning mathematical properties enable the applications, in our setting the cloud providers, to perform certain operations on encrypted data such as addition or multiplication. Note that any function can be built as an arithmetic circuit, using solely addition and multiplication gates. The downside is that FHE or even SHE schemes that are capable to provide both, report poor performance in terms of computation and storage. Semi Homomorphic schemes, however are considerably faster and have been commonly used in schemes that require arithmetic or geometric aggregation. *An example query can be calculation of the average heart rates or body mass index (BMI).*

## 3 Conceptual abstraction models

In this section, we elaborate on our two conceptual abstraction models: *the data protection tactic model* to abstract and reify different generic concepts found in most tactics, and *the data access model* to enable configurable tactic selection at run-time.

### 3.1 An abstraction model for data protection tactics

Each document is composed of fields. For example, a health record document may contain several fields such as a description, a sickness type and a numeric value indicating a measurable parameter like blood pressure. To protect sensitive values of the fields and

yet offer certain operations to clients, advanced data protection tactics are used. Each tactic typically offers very limited number of operations, and as a result, a field employs multiple tactics to satisfy functional requirements of an application. As depicted in Fig. 1, a tactic has a set of internal operations. Each of these operations comes with a leakage profile and several performance metrics. Our abstraction model is inspired by the recent SoK paper of Fuller et al. [24] published in IEEE Security and Privacy 2017.

**Tactic operations.** Tactics include one or several operations [24]. In general, (i) the *init* operation to set up cryptographic primitives and initial provisioning of data structures and databases, (ii) the *update* operation for dynamic tactics to add, update and delete documents, and (iii) the *query* operations to perform tactic-specific functionalities such as boolean search.

**Leakage profile.** Data protection tactics in such systems rely on secure data structures, e.g., an encrypted index, to facilitate data retrieval in an efficient way. These systems, notably their constitutive data structures, are susceptible of leaking (meta-)information. For example, a commonly used structure is encrypted inverted index; a leakage could be the result size of every possible search word. There are a wide range of leakage profiles with different levels of severity. Encrypted indexes usually contain information about searchable keywords and document identifiers. Searchable keywords are derived from the content of the documents. Document identifiers uniquely point to the documents in a database.

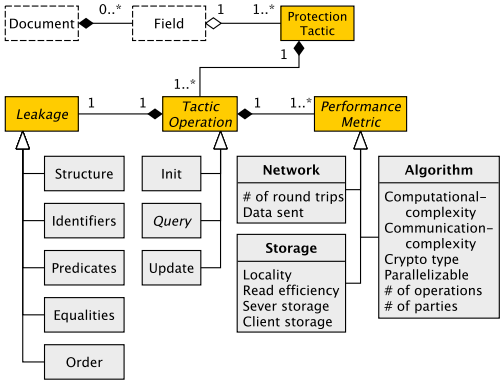


**Figure 1.** Abstraction model of data protection tactics for tactic providers

Prior work presented various formal security definitions for SE [6, 18, 26], and other work categorised the leakage levels [11]. We employed the leakage taxonomy presented by Fuller et al. [24] for the reification of this concept due to its generality and applicability.

To present a middleware solution, having a generic classification of leakage profiles does not capture all specific cases for each operation. For example, the update operations (create, delete, and update) are sensitive. They might leak information about the future or the past; certain leakages occur prior to any query in the setup time (having a snapshot of the database) or at query time such as boolean queries. The pragmatic reification of data protection level in a data-access middleware motivates the idea of presenting the leakage profiles on a per-operation basis.

The leakage level of data protection tactics should be concretely categorised into five levels [24]: (i) *structure*, i.e. nothing is leaked except the size of the entire data structure or things which can be hidden by padding, (ii) *identifiers*, i.e. past and future access patterns of identifiers are leaked, (iii) *predicates*, i.e. complex query predicates leak information such as intersection of a boolean query with a known range, (iv) *equalities*, i.e. which objects have the same value in the system, and (v) *order*, i.e. the numerical and lexicographic order of the objects are leaked. The leakage level of *order* is the highest, and *structure* is the lowest (the most secure one).

**Performance metrics.** Each tactic operation also comes with a performance cost impacting clients' experience. As illustrated in Fig. 1, the performance of data protection tactics can be measured and quantified by certain types of metrics related to the underpinning algorithms, network, and storage overheads. Tactics can employ various algorithmic designs to securely execute operations, which in turn may affect performance differently, e.g., tree based search vs. exhaustive search. Those decisions consequently have impact on networking infrastructure in terms of data sent and received between clients and providers. Besides, tactics may have severe impacts on the locality of objects, read efficiency, the size of data storage both at the client and the server side.

### 3.2 An abstraction model for protected data access

Data protection tactics should be applied to documents with per-field granularity. Fig. 2 illustrates the data access abstraction model for sensitive fields, which primarily includes (i) which data-access and aggregate operations can be performed on a field, and (ii) up to what level sensitive fields are protected.

Each field of a document can be annotated with the model illustrated in Fig. 2. For instance, consider a medical document containing a patient's age. We can select its sensitivity level, and assign a *Class 2* protection level (explained later). We can then configure what operations are needed, in this case *Average* and *Equality Search*. The middleware employs the right implementations at runtime accordingly in order to meet the client's requirements.

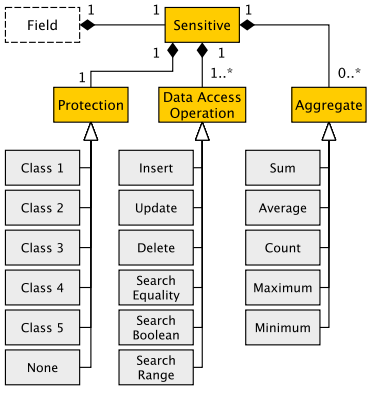


**Figure 2.** Data access model of the sensitive fields for application developers

**Query functionality.** A data access middleware should in general offer all required query functionalities to the client applications. The core functionalities of the query interfaces rely on key operations of associated arrays [27, 42] and basic functions of persistent storage systems [41], which are namely (i) create, (ii) read, (iii) update, and (iv) delete. The read operation goes beyond fetching a document; it comprises more complex search operations with predicates specifying conditions such as (i) equality, (ii) boolean queries

(conjunction, disjunction and negation), (iii) range, and (iv) others. These operations can be combined with aggregate functions such as sum, average, count, maximum, minimum, and so on. Each operation could be mapped to one or more data protection tactics.

**Data protection level.** Unlike the model of data protection tactic in which leakage profiles are specified per operation, the data access model specifies protection level per field. In other words, the protection level of a field is equal to the tactic with the weakest guarantees regardless of the strength of other tactics applied to it (i.e. a chain is only as strong as its weakest link.). We classify data protection tactics into five classes ($Class_1$,..., $Class_5$) of protection guarantees. Each of these classes corresponds to its counterpart in the data protection model. $Class_1$ has the least leakage.

## 4 Architecture and implementation

In this section, we present an overview of the DataBlinder middleware architecture, and we further describe the extensibility and pluggability of the architecture by introducing tactic commonalities and service provider interfaces (SPI). The setting consists of a trusted zone which is the application owner's datacenter, and an untrusted zone composed of external cloud providers (see Fig. 3).

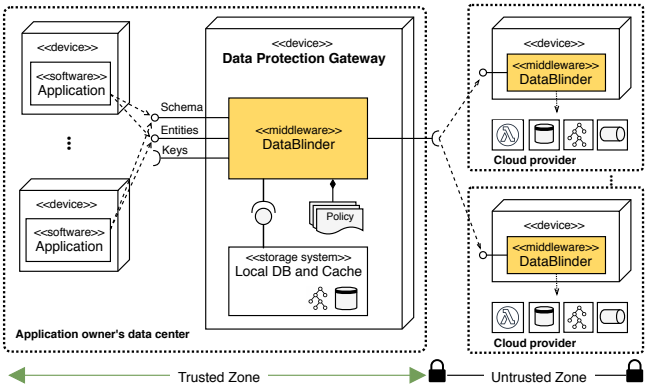


**Figure 3.** Middleware deployment view

In the trusted zone, different types of applications can benefit from external cloud-based storage through a data protection gateway. The DataBlinder middleware along with its required assets such as data protection policies and storage facilities are deployed within the gateway. There are several interfaces exposed to the applications, which are namely a *Schema* interface to enable clients to define and annotate data schemas and data protection metadata, an *Entities* interface to allow regular data-access operations, and a *Keys* interface to allow the system to integrate with on-premise key management systems (e.g., HSM). All data-access operations are trusted, and the inter-application communications within the datacenter follow regular security and access control mechanisms.

The untrusted zone consists of several cloud providers and the communication channels between the application owner's datacenter and these external resources. The middleware is distributed since SE tactics are inherently distributed.

### 4.1 The middleware architecture

Fig. 4 illustrates four subsystems of DataBlinder. Depending on the deployment location, either in the trusted or the untrusted

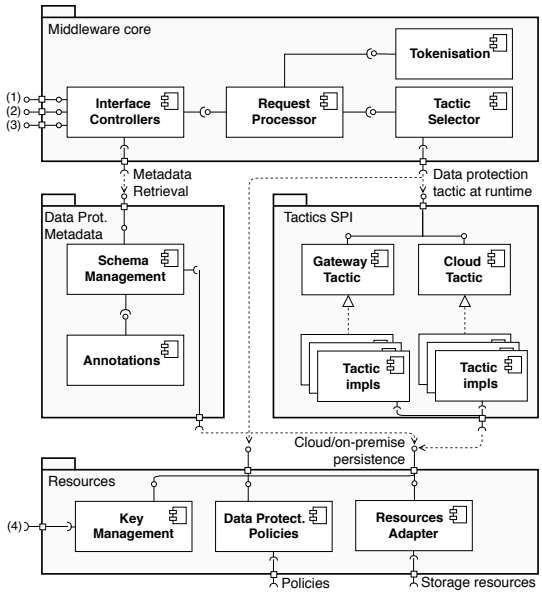


**Figure 4.** Middleware component diagram.

zone, different interfaces and components are employed. (i) *The middleware-core subsystem* is responsible for the abstract execution of the persistence logic, e.g., Create Read Update Delete (CRUD) operations, and adaptive and dynamic tactic selection at run-time. (ii) *The data protection metadata subsystem* is responsible for the persistence and retrieval of per-application database schemas and data protection annotations. The schema management component also validates whether the application documents correspond to the configured schemas. (iii) *The tactics SPI subsystem* is responsible for providing concrete tactic implementations, comprising a set of gateway and cloud implementations. (iv) *The resources subsystem* is responsible for enabling the access to external resources such as cryptographic key management systems, and on-premise or cloud-based resources such as storage systems.

### 4.2 Extensible and pluggable architecture

In this section, we present a more concrete description of the extensibility and pluggability of the DataBlinder architecture.

**Tactic commonalities.** Most tactics share common properties. They are all distributed in the sense that two or more parties are involved in performing a high-level operation such as boolean search. The comprehensive surveys on SE [6, 24, 51] distill their life cycle into three key operations: *setup* for key material generation and initial index provisioning, *update* for dynamic constructions with the operations like deletion, addition and modification, and *query* for constructing tokens and performing the given function.

Each data protection tactic includes a subset of operations. Each of these operations is a distributed protocol. As a result, tactics share a common framework in the DataBlinder architecture supporting: (1) gateway and cloud implementations per operation, (2) cryptographic primitives as building blocks (e.g., PRF), (3) key management integration, (4) communication channels for transferring protocol data, and lastly (5) data repository services available to both the gateway and the cloud implementations to satisfy tactic-specific requirements to construct distributed secure indexes.

| | Gateway Interfaces | Cloud Interfaces |
|---|---|---|
| Insert | Insertion, DocIDGen, SecureEnc | Insertion |
| Update | Update, DocIDGen, Retrieval SecureEnc | Update, Retrieval |
| Delete | Deletion | Deletion |
| Read | Retrieval, SecureEnc | Retrieval |
| Equality Search | EqQuery, EqResolution *<Read>* | EqQuery |
| Boolean Search | BoolQuery, BoolResolution *<Read>* | BoolQuery |
| Aggregate | *<Query>*, AggFunctionResolution | AggFunction |

**Table 1. Service Provider Interface (SPI)**. The implementations of these interfaces get loaded dynamically at runtime. *<Read>* and *<Query>* denote a set of interfaces required for a retrieval and a search operation.

**Tactic SPI.** The DataBlinder protection tactics can be extended by leveraging a set of interfaces. Each of these interfaces exposes a high-level operation defined in the data-access abstraction (see Fig. 2), including a gateway and cloud versions as described earlier in this section and Fig. 4. The first interface which is mandatory to implement for all tactics is the *setup* interface. The other major but optional operations include CRUD, various search and aggregate queries. Each implementation receives all dependencies required to perform its protocol as listed in the commonalities. Table 1 lists the interfaces for a large subset of the high-level operations.

**Tactic selection at runtime.** The SPIs are implemented by security experts. The middleware loads the right implementations dynamically at runtime using the strategy design pattern [25].

### 4.3 Proof of concept development

DataBlinder supports two modes of execution: gateway and cloud, implemented using Spring Boot, i.e., ~6,000 lines of Java 8. We employed libraries such as Bouncy Castle for basic cryptographic primitives (e.g., AES/GCM, RSA/OAEP, HMAC-SHA256, etc.). We further leveraged the Clusion [36] project to provide several data protection tactics,and Javallier [55] for the Paillier [48] cryptosystem. The DataBlinder data protection tactics have been developed using these building blocks. We employed document-oriented databases, e.g., MongoDB and Elasticsearch, to store documents and indexes. We also employed a key-value datastore, e.g., Redis, in a semi-persistent durability mode to take advantage of basic constructions such as persistent sets, maps, and so on, to build custom indexes.

## 5 Use case validation and evaluation

We implemented and integrated several tactics using the proposed achitecture based on the SPI pattern (see Table 2). The implementation covers a broad range of tactics, having various properties, such as different protection levels, forward privacy (e.g. Mitra and Sophos), deterministic and probabilistic encryption (e.g. DET and RND), read and space efficiency (e.g. BIEX-2Lev and BIEX-ZMF), data order (e.g. ORE and OPE), and HE (e.g. Paillier).

### 5.1 Healthcare use case

To validate the middleware, we present a real-world example of the industry-standard FHIR-compliant [30] medical documents. We annotate the schema based on some assumptions regarding protection level and functionalities.

**Example.** Observations are measurements and assertions about patients [30]. In the following document, the amount of Glucose observed in a blood test is illustrated. Most of these fields are assumed to be sensitive since they can be the indicators of diabetes.

```
{
 id: f001,
 identifier: 6323,
 status: final,
 code: Glucose,
 subject: John Doe,
 effective: 1359966610
 issued: 1362407410,
 performer: John Smith,
 value: 6.3,
 interpretation: High
}
```

| Sensitives | Annotations |
|---|---|
| status | C3, op [I, EQ, BL] |
| code | C3, op [I, EQ, BL] |
| subject | C2, op [I, EQ] |
| effective | C5, op [I, EQ, BL, RG] |
| issued | C5, op [I, EQ, BL, RG] |
| performer | C1, op [I] |
| value | C3, op [I, EQ, BL], agg [avg] |

| Sensitives | Tactic Selection | Reasons |
|---|---|---|
| status | BIEX-2Lev | Boolean & cross-field |
| code | BIEX-2Lev | Boolean & cross-field |
| subject | Mitra | Identifier protection level |
| effective | DET, OPE | Range queries |
| issued | DET, OPE | Range queries |
| performer | RND | Structure protection level |
| value | BIEX-2Lev, Paillier | Cloud-side averages |

C is a class; op is a list of operations; I,EQ, BL and RG are insertion, equality, boolean and range queries; agg is the list of aggregate functions; and avg is an average operation. DataBlinder enforces data protection policies to perform tactic selection, and it abstracts away the complexity of underpinning cryptographic protocols. Therefore, software developers only require the necessary knowledge about the data-access abstraction model. Moreover, the middleware decouples the applications built on top, in the sense that evolvability of the tactics has limited impact on the applications with respect to their functionality and data protection requirements.

### 5.2 Performance evaluation

We evaluate the overall performance overhead of DataBlinder in comparison to the scenarios where: the application only does data operations and does not use the middleware or any tactic ($S_A$); the data protection tactics are implemented hard-coded into the application without using the middleware ($S_B$); and the application uses DataBlinder to enforce the required data protection tactics ($S_C$).

**Set-up.** To evaluate the performance overhead of DataBlinder, we developed the middleware as presented in §4.3, and we deployed an instance of it on the Openstack private cloud in *gateway mode* and another instance on a public cloud provider in *cloud mode*. Our underpinning Openstack compute node comes with 2.60 GHz Intel Xeon E5-2660 processors and 128GB DDR3 memory. The gateway VM instance has 8 vCPU cores and 16GB of RAM. The cloud VM instance has 4 vCPU cores and 16GB RAM. We deployed an instance of Redis in a semi-durability mode on both sides and an instance of MongoDB on the cloud. We deployed an instance of Locust [31] load generation and benchmarking framework in a third VM instance on Openstack in the trusted zone of the experiment.

**Results.** We performed 3 experiments using the medical document application introduced in §5.1. There were in total 8 tactics involved in the benchmarks, namely Mitra, RND, Paillier, and five times DET. Figure 5 illustrates *insert* and *equality search* operations, as well as the overall throughput. There is 44% overall throughput loss by employing data protection tactics. Adding our middleware to this setting causes only 1.4% additional overall throughput loss in comparison to the scenario where tactics are inflexibly integrated into the application. The following table further shows the overall average latency, and 50th, 75th and 99th percentile latency.

| Operation | Scheme | Protection level | | SPI* | | | Implementation |
|---|---|---|---|---|---|---|---|
| | | Class(#) | Leakage | Gateway | Cloud | Challenge | |
| Equality Search | DET | 4 | Equalities | 9 | 6 | - | ○ |
| | Mitra [15] | 2 | Identifiers | 7 | 5 | Local storage | ○ |
| | Sophos [7] | 2 | Identifiers | 6 | 4 | Key management | ○ |
| | RND | 1 | Structure | 6 | 4 | Inefficiency | ○ |
| Boolean Search | BIEX-2Lev [35] | 3 | Predicate | 8 | 5 | Storage impl. complexity | [36] ☽ |
| | BIEX-ZMF [35] | 3 | Predicate | 8 | 5 | Storage impl. complexity | [36] ☽ |
| Range Query | OPE [4] | 5 | Order | 3 | 3 | | [40] ☽ |
| | ORE [5] | 5 | Order | 3 | 3 | | [38] ☽ |
| Sum | Paillier [48] | - | - | 3 | 3 | Key management | [55] ☽ |
| Average | Paillier [48] | - | - | 3 | 3 | Key management | [55] ☽ |

**Table 2.** These cryptographic constructions have been implemented and integrated to DataBlinder using the tactic interfaces. ∗ denotes the number of service interfaces required in the implementation (Table 1); ○ denotes that we implemented the construction; ☽ denotes that the implementation is slightly modified.

Based on our observation, the execution of aggregate protocols, namely the Paillier partially homomorphic encryption (PHE), had a considerable impact on these numbers.

| Scenario (latency) | 50th | 75th | 99th | Average |
|---|---|---|---|---|
| $S_A$ | 62ms | 81ms | 140ms | 85ms |
| $S_C$ | 110ms | 2500ms | 13000ms | 828ms |
| Aggregate | 105ms | 1700ms | 5600ms | 1114ms |

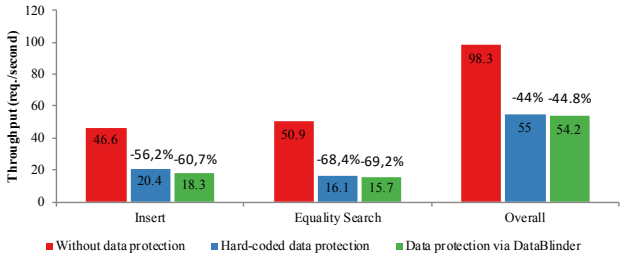


**Figure 5.** Per-operation and overall throughput comparison. Each experiment included ~151k requests, ~50k documents, ~350k secure index operations, and 1,000 benchmark users with a balance between *read* (equality search protocols), *write* (insertions and secure indexing) and *aggregate* operations (search and homomorphic calculation of averages). In addition to the computational and search complexity of each tactic, the Paillier queries were executed ~50k times per run, having a considerable impact on the throughput of the experiments involving data protection tactics (blue and green).

## 6 Related work

**Encrypted databases.** Designing protected search systems has been an active research area over the past years. CryptDB[52] is one of the seminal works, using onion of encryption that encrypt data in a layered approach for queries with different functionalities. The main goal is to keep the underlying legacy database unchanged. Pappas et al. [49] present Blindseer which is, unlike CryptDB, a custom database based on an approach using encrypted bloom filter trees as a storage mechanism. Fuhry et al. present the HardIDX [23] secure index system which leveraged Intel SGX to perform relatively efficient queries. Towards building secure NoSQL database systems, ARX[50] aims at presenting a protected system on top of MongoDB to provide the functionalities necessary to support associative arrays. EncKV [56] proposed a secure key-value store with a focus on secure and efficient partitioning of encrypted data and distributing data evenly across a cluster.

**Middleware solutions.** Diallo et al. present CloudProtect [20], a middleware to enable users transparently encrypt sensitive data within various cloud applications. Their main goal is to support application functionalities while protecting sensitive data. CloudProtect uses deterministic encryption for search purposes, and on top of that, it has a policy-based protocol to expose senesitive data in plaintext for a limited duration on the server if some operation or function execution requires access to the data in plaintext. Alves et al. [39] present a framework for searching encrypted databases. They use ORE and HE for the range and aggregate queries.

There are several commercial encryption products such as Skyhigh Networks[44] and CipherCloud[17]. Most of these solutions employ legacy-friendly SE constructions to ensure that the existing applications can operate as before. Recently, Ionic [33] presented an encrypted search system with an advanced query construction mechanism based on EC-OPRF [10].

Most of these systems either proposed new SE constructions or employed fixed data protection tactics to provide their functionalities. However, the key goals of DataBlinder are to present a middleware solution allowing software developers to configure a notion of security with respect to their required operations, and it is not dependent on any particular database. Moreover, it facilitates the future tactic extensions via its architecture.

## 7 Conclusion

We presented DataBlinder, a distributed data access middleware that supports fine-grained data protection configuration on application data towards *crypto agility*. Our performance evaluations showed that DataBlinder offers this flexibility at the cost of 1.4% overall throughput loss in comparison to a scenario where the tactics are inflexibly integrated into an application without the middleware.

Current architecture can be deployed as a *cloud-native* service, where the gateway is a stateless data access middleware (e.g., ORM [47]). However, there exist some secure SE tactics, e.g. Sophos [7], requiring keeping the state at the gateway. A challenging research direction towards secure cloud-native systems is to design efficient stateless SE schemes. The current architecture does not take other classes of constructions, e.g., MPC, Oblivious RAM, and TEE, into consideration. It is interesting to explore and abstract their new tradeoffs, different trust models and various execution frameworks.

## References


[1] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. 2004. Order preserving encryption for numeric data. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. ACM, 563–574.

[2] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. 2006. Deterministic and Efficiently Searchable Encryption. Cryptology ePrint Archive, Report 2006/186. https://eprint.iacr.org/2006/186.

[3] Alex Biryukov, Vesselin Velichkov, and Yann Le Corre. 2016. Automatic Search for the Best Trails in ARX: Application to Block Cipher Speck. Cryptology ePrint Archive, Report 2016/409. https://eprint.iacr.org/2016/409.

[4] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. 2012. Order-Preserving Symmetric Encryption. Cryptology ePrint Archive, Report 2012/624. https://eprint.iacr.org/2012/624.

[5] Dan Boneh, Kevin Lewi, Mariana Raykova, Amit Sahai, Mark Zhandry, and Joe Zimmerman. 2015. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. , 563–594 pages.

[6] Christoph Bösch, Pieter Hartel, Willem Jonker, and Andreas Peter. 2015. A survey of provably secure searchable encryption. *ACM Computing Surveys (CSUR)* 47, 2 (2015), 18.

[7] Raphael Bost. 2016. âĹŚ oφoç: Forward Secure Searchable Encryption. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1143–1154.

[8] Raphael Bost, Brice Minaud, and Olga Ohrimenko. 2017. Forward and Backward Private Searchable Encryption from Constrained Cryptographic Primitives. Cryptology ePrint Archive, Report 2017/805. https://eprint.iacr.org/2017/805.

[9] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2014. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)* 6, 3 (2014), 13.

[10] Jonathan Burns, Daniel Moore, Katrina Ray, Ryan Speers, and Brian Vohaska. 2017. EC-OPRF: Oblivious Pseudorandom Functions using Elliptic Curves. *IACR Cryptology ePrint Archive* 2017 (2017), 111.

[11] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. 2015. Leakage-abuse attacks against searchable encryption. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. ACM, 668–679.

[12] David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. 2014. Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation. Cryptology ePrint Archive, Report 2014/853. https://eprint.iacr.org/2014/853.

[13] David Cash, Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel Rosu, and Michael Steiner. 2013. Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries. Cryptology ePrint Archive, Report 2013/169. https://eprint.iacr.org/2013/169.

[14] David Cash and Stefano Tessaro. 2014. The Locality of Searchable Symmetric Encryption. Cryptology ePrint Archive, Report 2014/308. https://eprint.iacr.org/2014/308.

[15] Javad Ghareh Chamani, Dimitrios Papadopoulos, Charalampos Papamanthou, and Rasool Jalili. 2018. New constructions for forward and backward private symmetric searchable encryption. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 1038–1055.

[16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. 2018. Tfhe: Fast fully homomorphic encryption over the torus. *Journal of Cryptology* (2018), 1–58.

[17] CipherCloud. 2014. CipherCloud. Online. https://www.ciphercloud.com/.

[18] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. 2011. Searchable symmetric encryption: improved definitions and efficient constructions. *Journal of Computer Security* 19, 5 (2011), 895–934.

[19] Ioannis Demertzis and Charalampos Papamanthou. 2017. Fast searchable encryption with tunable locality. In *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 1053–1067.

[20] Mamadou H Diallo, Bijit Hore, Ee-Chien Chang, Sharad Mehrotra, and Nalini Venkatasubramanian. 2012. Cloudprotect: managing data privacy in cloud applications. In *2012 IEEE Fifth International Conference on Cloud Computing*. IEEE, 303–310.

[21] Taher ElGamal. 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory* 31, 4 (1985), 469–472.

[22] Sky Faber, Stanislaw Jarecki, Hugo Krawczyk, Quan Nguyen, Marcel Rosu, and Michael Steiner. 2015. Rich Queries on Encrypted Data: Beyond Exact Matches. Cryptology ePrint Archive, Report 2015/927. https://eprint.iacr.org/2015/927.

[23] Benny Fuhry, Raad Bahmani, Ferdinand Brasser, Florian Hahn, Florian Kerschbaum, and Ahmad-Reza Sadeghi. 2017. HardIDX: Practical and secure index with SGX. In *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 386–408.

[24] Benjamin Fuller, Mayank Varia, Arkady Yerukhimovich, Emily Shen, Ariel Hamlin, Vijay Gadepally, Richard Shay, John Darby Mitchell, and Robert K Cunningham. 2017. Sok: Cryptographically protected database search. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 172–191.

[25] Erich Gamma. 1995. *Design patterns: elements of reusable object-oriented software*. Pearson Education India.

[26] Eu-Jin Goh et al. 2003. Secure indexes. *IACR Cryptology ePrint Archive* 2003 (2003), 216.

[27] Michael T Goodrich, Roberto Tamassia, and Michael H Goldwasser. 2014. *Data structures and algorithms in Java*. John Wiley & Sons.

[28] Paul Grubbs, Marie-Sarah Lacharité, Brice Minaud, and Kenneth G Paterson. 2019. Learning to Reconstruct: Statistical Learning Theory and Encrypted Database Attacks. In *IEEE Symposium on Security and Privacy (S&P) 2019*.

[29] Paul Grubbs, Kevin Sekniqi, Vincent Bindschaedler, Muhammad Naveed, and Thomas Ristenpart. 2017. Leakage-abuse attacks against order-revealing encryption. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 655–672.

[30] Health Level Seven International (HL7) health-care standards organization. 2011. Fast Healthcare Interoperability Resources. http://hl7.org/fhir/ (Last visit: 2019-05-16).

[31] Jonatan Heyman et al. 2016. Locust Load Testing Framework. https://github.com/locustio/locust (Last visit: 2019-05-16).

[32] imec.icon. 2016-2018. SeClosed: Secure, Cloud-based Storage and Processing of Sensitive Documents. https://www.imec-int.com/en/what-we-offer/research-portfolio/seclosed (Last visit: 2019-11-16).

[33] Ionic. 2014. Introducing Ionic Encrypted Search. Online. https://www.ionic.com/blog/introducing-ionic-encrypted-search/.

[34] Yuval Ishai, Eyal Kushilevitz, Steve Lu, and Rafail Ostrovsky. 2016. Private Large-Scale Databases with Distributed Searchable Symmetric Encryption. In *Topics in Cryptology - CT-RSA 2016*, Kazue Sako (Ed.). Springer International Publishing, Cham, 90–107.

[35] Seny Kamara and Tarik Moataz. 2017. Boolean searchable symmetric encryption with worst-case sub-linear complexity. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 94–124.

[36] Encrypted Systems Lab. 2016. Clusion. https://github.com/encryptedsystems/Clusion (Last visit: 2019-04-30).

[37] Marie-Sarah Lacharité, Brice Minaud, and Kenneth G Paterson. 2018. Improved reconstruction attacks on encrypted data using range query leakage. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 297–314.

[38] Lewi et al. 2016. An Implementation of Order-Revealing Encryption. https://github.com/kevinlewi/fastore (Last visit: 2019-05-16).

[39] Pedro G. M. R. Alves and Diego F. Aranha. 2018. A framework for searching encrypted databases. *Journal of Internet Services and Applications* 9, 1 (03 Jan 2018), 1. https://doi.org/10.1186/s13174-017-0073-0

[40] Ayman Madkour. 2018. Order-Preserving Encryption. https://github.com/aymanmadkour/ope (Last visit: 2019-05-16).

[41] James Martin. 1981. Managing the data base environment. (1981).

[42] Kurt Mehlhorn and Peter Sanders. 2008. *Algorithms and data structures: The basic toolbox*. Springer Science & Business Media.

[43] Muhammad Naveed, Seny Kamara, and Charles V Wright. 2015. Inference attacks on property-preserving encrypted databases. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 644–655.

[44] SkyHigh Networks. 2014. SkyHigh Networks. Online. https://www.skyhighnetworks.com/cloud-encryption/.

[45] U.S. Department of Health and Human Services. [n. d.]. Breach Portal: Notice to the Secretary of HHS Breach of Unsecured Protected Health Information. https://ocrportal.hhs.gov/ocr/breach/breach_report.jsf. Accessed: 2019-04-13.

[46] US Department of Health, Human Services, et al. 2009. HITECH Act enforcement interim final rule. *US Department of* (2009).

[47] Elizabeth J O'Neil. 2008. Object/relational mapping 2008: hibernate and the entity data model (edm). In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 1351–1356.

[48] Pascal Paillier. 1999. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 223–238.

[49] Vasilis Pappas, Fernando Krell, Binh Vo, Vladimir Kolesnikov, Tal Malkin, Seung Geol Choi, Wesley George, Angelos Keromytis, and Steve Bellovin. 2014. Blind seer: A scalable private dbms. In *2014 IEEE Symposium on Security and Privacy*. IEEE, 359–374.

[50] Rishabh Poddar, Tobias Boelter, and Raluca Ada Popa. 2016. Arx: A strongly encrypted database system. *IACR Cryptology ePrint Archive* 2016 (2016), 591.

[51] Geong Sen Poh, Ji-Jian Chin, Wei-Chuen Yau, Kim-Kwang Raymond Choo, and Moesfa Soeheila Mohamad. 2017. Searchable symmetric encryption: designs and challenges. *ACM Computing Surveys (CSUR)* 50, 3 (2017), 40.

[52] Raluca Ada Popa, Catherine Redfield, Nickolai Zeldovich, and Hari Balakrishnan. 2011. CryptDB: protecting confidentiality with encrypted query processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. ACM, 85–100.

[53] General Data Protection Regulation. 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46. *Official Journal of the European Union (OJ)* 59, 1-88 (2016), 294.

[54] Dawn Xiaoding Song, David Wagner, and Adrian Perrig. 2000. Practical techniques for searches on encrypted data. In *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*. IEEE, 44–55.

[55] Brian Thorne et al. 2017. Javallier. https://github.com/n1analytics/javallier (Last visit: 2019-05-15).
[56] Xingliang Yuan, Yu Guo, Xinyu Wang, Cong Wang, Baochun Li, and Xiaohua Jia. 2017. Enckv: An encrypted key-value store with rich queries. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. ACM, 423–435.