A Decentralized and Encrypted National Gun Registry

Seny Kamara [*] Brown University
Tarik Moataz [†] Aroki Systems
Andrew Park [‡] Brown University
Lucy Qin [§] Brown University

Abstract

Gun violence results in a significant number of deaths in the United States. Starting in the 1960's, the US Congress passed a series of gun control laws to regulate the sale and use of firearms. One of the most important but politically fraught gun control measures is a national gun registry. A US Senate office is currently drafting legislation that proposes the creation of a voluntary national gun registration system. At a high level, the bill envisions a decentralized system where local county officials would control and manage the registration data of their constituents. These local databases could then be queried by other officials and law enforcement to trace guns. Due to the sensitive nature of this data, however, these databases should guarantee the confidentiality of the data.

In this work, we translate the high-level vision of the proposed legislation into technical requirements and design a cryptographic protocol that meets them. Roughly speaking, the protocol can be viewed as a decentralized system of locally-managed end-to-end encrypted databases. Our design relies on various cryptographic building blocks including structured encryption, secure multi-party computation and secret sharing. We propose a formal security definition and prove that our design meets it. We implemented our protocol and evaluated its performance empirically at the scale it would have to run if it were deployed in the United States. Our results show that a decentralized and end-to-end encrypted national gun registry is not only possible in theory but feasible in practice.

1 Introduction

Gun violence accounts for a considerable number of deaths in the United States. $36,000$ Americans are killed by guns every year and another $100,000$ are injured. Around $2/3$ of gun deaths are suicides and $1/3$ are homicides. Among high-income countries, 93% of children (14 and under) that are killed by guns are American. Each year 600 Women are shot and killed by an intimate partner and 4.5 million Women have been threatened with a gun. Black people are 10 times more likely to be killed by a gun than Whites and Black men account for 52% of gun deaths [18].

Gun control. In the US, firearms are regulated by a set of laws, regulations and policies commonly referred to as *gun control* laws. At the national level, the most prominent gun control laws are the *Omnibus Crime Control and Safe Streets Act* of 1968, which prohibited the interstate sale of handguns and increased the minimum age to purchase a gun to 21; the *Gun Control Act* of 1968 which established the Federal Firearms License system which requires gun sellers to be licensed; and the *Brady Act* of 1993 which instituted the system of background checks, requiring sellers to check the criminal history of buyers. Another important gun control law is the *Firearm Owner Protection Act* (FOPA) of 1986 which amends some of the provisions of the Gun Control Act. One of the main provisions of FOPA was to prohibit the Bureau of Alcohol, Tobacco, Firearms and Explosives (ATF) from keeping a registry that maps guns to their owners. More precisely, the Act states:

[*] seny@brown.edu
[†] tarik@aroki.com
[‡] andrew_park1@alumni.brown.edu
[§] lucyq@brown.edu

No such rule or regulation...may require that records...be recorded at or transferred to a facility owned, managed, or controlled by the United States or any State or any political subdivision thereof, nor that any system of registration of firearms, firearms owners, or firearms transactions or dispositions be established.

Gun tracing. Gun tracing is the tracking of guns recovered by law enforcement. In the US, it is conducted by the ATF's National Tracing Center in West Virginia. When a gun is recovered by law enforcement, a trace can be requested based on the gun's serial number and characteristics. Because FOPA prohibits the existence of databases that map owners to guns and prevents data on firearms from being searchable [37], gun tracing is done manually by searching through physical stacks of paper. This requires, on average, 4 to 7 business days [36]. Furthermore, searches cannot be run on the text of a record or using specific tags or identifiers. On average there are $1,500$ traces a day and about $370,000$ a year and only 65% of search requests are successfully answered. FOPA even requires gun dealers to get a special exemption by the ATF to use electronic or cloud-based computing systems to store their data [26].

A decentralized national registry. US Senator Wyden's office is currently drafting legislation that proposes the creation of a *voluntary* national system of firearm licensing and registration. The core idea behind the bill is to provide financial and legislative incentives for US counties that choose to participate in the system. Two crucial aspects of the proposed design are to guarantee: (1) the confidentiality of the data; and (2) that local officials maintain complete control of their constituents' data. Control, here, means the ability to "pull" the data from the system at any point in time. From a technical perspective, these requirements roughly translate to designing a distributed and decentralized system of locally-managed end-to-end encrypted databases that would allow for efficient gun tracing without compromising the privacy of gun owners.

At a high level, the legislation would require all gun owners in a participating US county to register their firearms with a local official by providing information about the make, model, caliber or gauge, and serial number along with the owner's registration number. This information would be stored in an end-to-end encrypted database whose key is known only to the local official. The encrypted database itself would be hosted on state servers or the cloud to guarantee a higher degree of availability. A county's encrypted database would be queryable by law enforcement agents and other officials but the query process would be overseen by the county's official. In addition, the system needs to remain functional even if a county's local official is offline. As mentioned above, county officials should also have the ability to pull their database from the system entirely at any point in time.

Impact of the proposed system. The system envisioned by this legislation would be part of a broader set of existing gun control laws and background check system. It would allow for faster and more accurate gun tracing which, in turn, would help in violent criminal cases and possibly act as a deterrent from illegally transferring firearms. A possible critique of the system is the expense of running it. Another possible critique is that allowing law enforcement to obtain and query this data could lead to the Federal government confiscating people's firearms and, therefore, violating the second amendment. The legislation and envisioned system addresses these concerns by requiring that the databases that store registration records be end-to-end encrypted with a key held and managed by a local county official.

Our contributions. The purpose of this work is to ascertain whether the high-level design goals of the proposed legislation are technically feasible. Towards this end we make the following contributions:

- *(Cryptographic design)* as a first step, we translated the high-level requirements of the legislation to a set of technical requirements. We then designed a novel cryptographic protocol to satisfy them. The protocol can be roughly viewed as a decentralized collection of end-to-end multi-user encrypted databases. It makes use of a variety of cryptographic building blocks including secure two-party and multi-party computation, structured encryption and secret sharing. At a very high level, the system is composed of a encrypted global directory that allows authorized parties to find the county associated to a serial number; and of a set of local encrypted databases that store the full records and that are owned and managed by a local county official. As far as we know, this kind of protocol and system

has never been considered in the past.

- *(Formal security definition)* we formulate a security definition for such a decentralized registry and show that our protocol satisfies it. Our definition is in the ideal/real-world paradigm which is standard in cryptography.
- *(Deployment considerations)* in our setting, there are many real-world considerations that need to be taken into account that are not captured by our abstract protocol. We identify these considerations and describe how our cryptographic protocol could be deployed in practice.
- *(Prototype & evaluation)* we implement our protocol and evaluate it empirically. Our evaluation shows that the protocol is practical at the scale of the US. More precisely, assuming the system stores 400 million records, where the largest county has 50 million records [1] it takes 300 ms to identify the county that a gun is registered in and at most 1 minute to query the county's local database on a query that matches 100 records. Adding a batch of $10,000$ records to the system takes 45 minutes.

Though our work was motivated directly by the legislation mentioned above and our solution is relatively unique, we believe that our design could prove useful for other decentralized systems that need to store sensitive data.

2 Related Work

Gun registries exists at the state level in the United States and in other countries. Canada implemented a national firearms registry through its Firearms Act in 1995, which was later dismantled in 2012. Since then, Quebec has implemented its own Firearms Registry. In 2019, New Zealand proposed legislation to create a national firearms registry, in response to the Christchurch shooting. Within the United States, California, Connecticut, Delaware, Hawaii, Maryland, New Jersey, and New York currently have firearm registration requirements for some subset of firearms, depending on the state [19].

Cryptographic building blocks. Our encrypted registry system relies on secure computation, secret sharing and structured encryption. Secure two-party computation was introduced by Yao [52] and secure multi-party computation by by Goldwasser, Micali and Wigderson [28]. Formal definitions of security for MPC in the standalone setting were given by Canetti in [13]. Secret sharing was introduced by Shamir in [45]. We also make use of structured encryption and, specifically, of dictionary, multi-map and (NoSQL) database encryption schemes. Structured encryption was introduced by Chase and Kamara as a generalization of indexed-based symmetric searchable encryption (SSE) constructions [22]. SSE was introduced by Song, Wagner and Perrig [46] and formalized by Curtmola et al. [22]. In the standard setting of structured encryption (STE), the client encrypts its data, stores it in on an untrusted server, and performs queries on the encrypted structure. In this work, however, the client also needs the ability to allow other parties to query its encrypted data. This multi-user setting was considered in [22, 29, 42, 43].

Federated encrypted databases. Our system has some superficial similarities to federated encrypted databases like Conclave [49] and SMSQL [6]. These are systems that also leverage MPC to privately query multiple databases. Their goal, however, is to support private queries on the union of disjoint databases, each of which is held by different parties. On the other hand, in our setting, queries are executed over a single local database (after it has been found using the global dictionary). Furthermore, in our system, 2PC and MPC are not used to process the databases but, to generate tokens that, in turn, are used to query the STE-encrypted databases. To summarize, our system is designed to enable a party to efficiently find the local database it needs to query, whereas federated encrypted databases are designed to query the union of multiple databases owned by different parties.

[1] The US is estimated to have $393,347,000$ guns. The largest county in the US is Los Angeles county with a population of 10 million. Assuming an average of 5 guns per person, this county would have 50 million guns

3 Preliminaries

Notation. The set of all binary strings of length n is denoted as $\{0,1\}^n$, and the set of all finite binary strings as $\{0,1\}^*$. The output y of an algorithm $\mathcal{A}$ on input x is denoted by $x \leftarrow \mathcal{A}(x)$. When we wish to make $\mathcal{A}$'s random coins r explicit, we write $y \leftarrow \mathcal{A}(x;r)$. Given a sequence $\mathbf{s}$ of n elements, we refer to its ith element as s_i. If S is a set then $\#S$ refers to its cardinality. Throughout, k will denote the security parameter.

Dictionaries & multi-maps. A dictionary DX with capacity n is a collection of n label/value pairs $\{(\ell_i, v_i)\}_{i \leq n}$ and supports get and put operations. We write $v_i := \mathsf{DX}[\ell_i]$ to denote getting the value associated with label ℓ_i and $\mathsf{DX}[\ell_i] := v_i$ to denote the operation of associating the value v_i in DX with label ℓ_i. A multi-map MM with capacity n is a collection of n label/tuple pairs $\{(\ell_i, \mathbf{v}_i)_i\}_{i \leq n}$ that supports Get and Put operations. We write $\mathbf{v}_i := \mathsf{MM}[\ell_i]$ to denote getting the tuple associated with label ℓ_i and $\mathsf{MM}[\ell_i] := \mathbf{v}_i$ to denote operation of associating the tuple $\mathbf{v}_i$ to label ℓ_i. Multi-maps are the abstract data type instantiated by an inverted index. In the encrypted search literature multi-maps are sometimes referred to as indexes, databases or tuple-sets (T-sets).

Document databases. A document database DB of size n holds n records $\{\mathbf{r}_1, \ldots, \mathbf{r}_n\}$ each of which is a collection of field/value pairs $\mathsf{field{:}value}$. Here, we consider databases that support boolean queries, i.e., queries of the form $\varphi = \big(\mathsf{field}_1 = \mathsf{value}_1 \wedge \mathsf{field}_2 = \mathsf{value}_2 \vee \mathsf{field}_3 = \mathsf{value}_3\big)$.

Basic cryptographic primitives. A symmetric-key encryption scheme is a set of three polynomial-time algorithms $\mathsf{SKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ such that Gen is a probabilistic algorithm that takes a security parameter k and returns a secret key K; Enc is a probabilistic algorithm that takes a key K and a message m and returns a ciphertext c; Dec is a deterministic algorithm that takes a key K and a ciphertext c and returns m if K was the key under which c was produced. Informally, a private-key encryption scheme is secure against chosen-plaintext attacks (CPA) if the ciphertexts it outputs do not reveal any partial information about the plaintext even to an adversary that can adaptively query an encryption oracle. In addition to encryption schemes, we also make use of pseudo-random functions (PRF), which are polynomial-time computable functions that cannot be distinguished from random functions by any probabilistic polynomial-time adversary. We refer the reader to [35] for formal security definitions.

Secret sharing. A threshold secret sharing scheme $\mathsf{SS} = (\mathsf{Share}, \mathsf{Recover})$ consists of two efficient algorithms [45]. Share takes as input a secret s, a threshold t and total number of shares n and outputs n shares $s_1, \ldots, s_n$. $\mathsf{Recover}$ takes as input t out of n shares and outputs s. A secret sharing scheme SS is secure if no efficient adversary can learn any partial information about the secret s given any set of $r < t$ shares. We refer the reader to [25] for formal definitions.

Secure multi-party computation. Secure multi-party computation [52] allows n parties to securely compute a function over their joint inputs without revealing any information about their inputs beyond what can inferred from the output. In our work, we make use of two-party secure computation (2PC) and of multi-party secure computation (MPC). For modularity and conciseness, we describe our protocol in the $(\mathcal{F}^f_{\mathsf{2PC}}, \mathcal{F}^f_{\mathsf{MPC}})$-hybrid model which functions like a real-world protocol execution except that all parties also have access to ideal 2PC and MPC functionalities denoted by $\mathcal{F}^f_{\mathsf{2PC}}$ and $\mathcal{F}^f_{\mathsf{MPC}}$. We only consider security against semi-honest adversaries so, in practice, these ideal functionalities can be instantiated with standard semi-honest two-party and multi-party protocols. We refer the reader to [25] for an overview of MPC and standard security definitions.

3.1 Structured Encryption

A (non-interactive, response-hiding and semi-dynamic) structured encryption scheme $\Sigma_{\mathsf{DS}} = (\mathsf{Init}, \mathsf{Token}, \mathsf{Query}, \mathsf{AddToken}, \mathsf{Add}, \mathsf{Resolve})$ for data structures DS consists of six efficient algorithms. Init takes as input a security parameter 1^k and outputs an encrypted dictionary EDS and a secret key K. Token takes as input a key K and a query q and outputs a token tk. Query takes as input an encrypted structure EDS and a token tk and outputs a ciphertext ct. $\mathsf{AddToken}$ takes as input a key K and an update u and outputs an add token

atk. Add takes as input an encrypted structure EDS and an add token atk and outputs a new encrypted structure EDS′. Resolve takes as input a key K and a ciphertext ct and outputs a value v.

In this work, we also rely on STE schemes that include a ResKey algorithm that takes as input a secret key K and a query q and outputs a restricted key K_q which can be used to resolve the encryption of v. We refer to such schemes as STE schemes with restricted resolve.

Security. There are two adversarial models for STE: persistent adversaries and snapshot adversaries. A persistent adversary observes: (1) the encrypted data; and (2) the transcripts of the interaction between the client and the server when a query is made. A snapshot adversary, on the other hand, only receives the encrypted data after a query has been executed. Persistent adversaries capture situations in which the server is completely compromised whereas snapshot adversaries capture situations where the attacker recovers only a snapshot of the server's memory.

The security of STE is formalized using "leakage-parameterized" definitions following [20, 22]. In this framework, a design is proven secure with respect to a security definition that is parameterized with a specific leakage profile. Leakage-parameterized definitions for persistent adversaries were given in [20, 22] and for snapshot adversaries in [5].[2] The leakage profile of a scheme captures the information an adversary learns about the data and/or the queries. Each operation on the encrypted data structure is associated with a set of *leakage patterns* and this collections of sets forms the scheme's *leakage profile*. We recall the informal security definition for STE and refer the reader to [5, 20, 22] for more details.

Definition 3.1 (Security vs. persistent adversary (Informal))**.** *Let* $\Lambda = \big(\mathcal{L}_{\mathsf{S}}, \mathcal{L}_{\mathsf{Q}}, \mathcal{L}_{\mathsf{U}}\big) = \big(\mathsf{patt}_1, \mathsf{patt}_2, \mathsf{patt}_3\big)$ *be a leakage profile. A structured encryption scheme* STE *for data structures* DS *is* Λ*-secure if there exists a* PPT *simulator that, given* $\mathsf{patt}_1(\mathsf{DS})$ *for an adversarially-chosen structure* DS*,* $\mathsf{patt}_2(\mathsf{DS}, q_1, \ldots, q_t)$ *for adaptively-chosen queries* $(q_1, \ldots, q_t)$*, and* $\mathsf{patt}_3(\mathsf{DS}, u_1, \ldots, u_t)$ *for adaptively-chosen updates* $(u_1, \ldots, u_t)$ *can simulate the view of any* PPT *adversary. Here, the view includes the encrypted data structure and the tokens of the queries.*

Encrypted dictionaries & multi-maps. When the data structure DS in the definitions above is a dictionary, then Σ_{DX} is a dictionary encryption scheme. Similarly, if DS above is a multi-map then Σ_{MM} is a multi-map encryption scheme. Also, note that dictionary encryption schemes are a special case of multi-map encryption schemes since dictionaries are just multi-maps with single-item tuples. There are many well-known practical multi-map encryption schemes that achieve different tradeoffs between query and storage complexity, leakage and efficiency [5, 10, 11, 16, 17, 17, 22, 24, 27, 31–34, 44, 47]. We note that all these constructions either implicitly have a ResKey algorithm or can be trivially modified to have one. From a security point of view, we require that, given a value v, one can simulate a ciphertext ct and a key K_R such that $\mathsf{Resolve}(K_R, \mathrm{ct})$ outputs v and that the ciphertexts output by Query be computationally indistinguishable form random. Again, these properties are trivially achievable by the mentioned schemes.

Encrypted document databases. Encrypted multi-maps can be combined with standard symmetric encryption to yield an encrypted document database. This is equivalent to the notion of *index-based* searchable symmetric encryption (SSE). For completeness we recall the details in Appendix A.

4 Overview of Legislation

Our design is based directly on legislation that is currently being drafted by Sen. Wyden's office. This proposal envisions a national firearm registry instantiated as a system of distributed, decentralized, and locally managed encrypted databases. The following details and requirements strictly come from drafted legislation.

Under this system, each county in the United States stores license and registration data in its own database. Each State then operates a server, which stores and maintains the availability of all of county

[2] Even though parameterized definitions were introduced in the context of SSE and STE, they can be (and have been) applied to other primitives, including to fully-homomorphic encryption, property-preserving encryption, oblivious RAM, secure multi-party computation and functional encryption.

databases corresponding to the state. The state server should not have any other responsibility outside of this role and does not act as a authorized user of the system nor should it have access to any of the data it stores.

In order to ensure the security and privacy of the data, each county designates a local official who is responsible for an encryption key. This key is required in order to view, query, update, and encrypt any registration data corresponding to the county. No additional parties may access any licensing or registration data from a county without this key. The local official must also upload any new licensing or registration data to the system.

Authorized users of the system include, but are not limited to, other local county officials, law enforcement personnel, and firearm distributors. These individuals are permitted to query the system for registration and licensing data but may not delete or update this data, unless they are the county's designated local official. At minimum the data collected includes basic personal information about a licensed firearm owner, including their license number, and information about their individual firearms (specifically, the make, model, caliber, and serial number). The legislation dictates that bulk queries (in the sense of being overly broad) and attempts at collecting large amounts of information from the system must be prohibited and reported. The system must therefore have some means of rate limiting queries while protecting the privacy of the data.

Given that different regions in the United States differ in infrastructure and have varying levels of Internet connectivity, the legislation makes an explicit requirement that authorized users must be able to make queries, even if the key held by the county's local official is offline. Offline access, however, must be bounded by some predetermined time. Once the key has been offline beyond this time, any data pertaining to this county should be entirely inaccessible. Another important feature of the system is that counties should have the ability to retract their database from the system at any point.

The system is voluntary in the sense that States can elect to participate by operating a server and counties can elect to participate by storing a database within the State server. This allows for local laws to dictate participation and accommodates for changes to those laws, if a state or county's constituents choose to later opt out of the system, without affecting the ability for other states or counties to participate.

To understand the type of data collected, we additionally reviewed existing firearm registration forms from New York [41], Washington D.C. [39], and the ATF [12]. Ultimately in our empirical evaluation, we based our sample record off of the minimum data requirements from this draft legislation since it contained the intersection of common attributes among these forms. Throughout this process, we have been in conversation with Sen. Wyden's office in order to understand the priorities of the legislation and the infrastructure constraints that must be accounted for when designing a system across counties in all 50 states. In our evaluation, we specifically chose virtual machines that would reflect the different resources available at the state and county level. Based on feedback from their office, we also explicitly included a cost analysis in our evaluation to understand the monetary feasibility of running a system like this at scale.

5 Our Protocol

In this section we describe the design of our encrypted registry system and its usage. The protocol $\Omega = (\mathsf{InitGlobal}, \mathsf{InitLocal}, \mathsf{Add}, \mathsf{Find}, \mathsf{Query}, \mathsf{OfflineQuery})$ consists of six protocols which we describe at a high-level below. We stress that the protocol we propose is one of many possible ways to instantiate the requirements outlined in Section 4 but that this design received positive feedback from members of the team responsible for drafting the legislation.

Parties. The system is designed to be executed among: a large (constant) number of parties $\mathbf{P}_1, \ldots, \mathbf{P}_\theta$, three backups $\mathbf{B}_1^{(i)}$, $\mathbf{B}_2^{(i)}$ and $\mathbf{B}_3^{(i)}$ for each party $\mathbf{P}_i$, two custodians $\mathbf{C}_1$ and $\mathbf{C}_2$, and a server $\mathbf{S}$. Figure 1 describes the parties. While the legislation envisions that every state hosts a server for its counties, in our model we only include a single server $\mathbf{S}$ for ease of exposition and because it captures the worst-case scenario from a security point of view (i.e., it captures the case where all state-level servers are corrupted and collude). The parties $\mathbf{P}_1$ through $\mathbf{P}_\theta$ correspond to the local county officials in the Bill who are responsible for registering gun owners. The parties $\mathbf{C}_1$ and $\mathbf{C}_2$ are *custodians* who we assume do not collude. The server

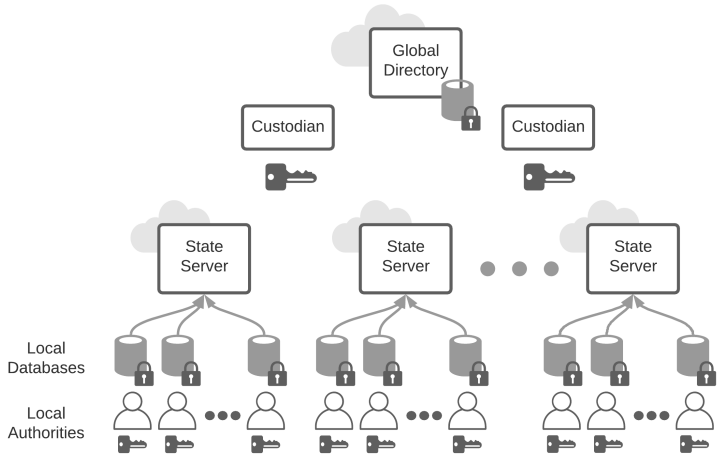


Figure 1: At a high level, the system consists of an encrypted global directory that stores the mapping of serial numbers to county identifiers, custodians who hold a share of the global directory's key, encrypted local databases managed by local officials, and state servers that store the local encrypted databases.

S is untrusted. We stress that while we chose to use three backups per party, two custodians and one server; the protocol can be trivially extended to handle a different number of custodians, backups and servers.

We sometimes refer to a party $\mathbf{P}_i$ by the role it plays during a particular operation. For example, if $\mathbf{P}_i$ queries $\mathbf{P}_j$'s local database, we refer to $\mathbf{P}_i$ as the *querier* and denote it by $\mathbf{Q}$ and to $\mathbf{P}_j$ as the *local official* and denote it by $\mathbf{L}$.

Initializing the global directory. To initialize the system, the two custodians $\mathbf{C}_1$ and $\mathbf{C}_2$ and the server execute $\Omega.\mathsf{InitGlobal}$. This sets up an encrypted dictionary EDX on the server which we call the *global directory* and provides each custodian with a share of its key. The global directory maps serial numbers to county identifiers and its purpose is to enable a querier to find the county a given gun is registered in. With this information, it can then interact directly with that county's local official to query its local database.

Initializing the local databases. After the global directory is initialized, each party $\mathbf{P}_i$ initializes its own local encrypted database by executing $\Omega.\mathsf{InitLocal}$. This results in a secret key K_i and an empty encrypted database EDB_i which it sends to the server. In addition, $\mathbf{P}_i$ splits its key into three shares and sends a share to each of its backups.

Adding a new record. A new record $\mathbf{r}$ is added by a local official $\mathbf{L} \in \{\mathbf{P}_1, \ldots, \mathbf{P}_\theta\}$ by executing the $\Omega.\mathsf{Add}$ protocol with the custodians and the server. At a high-level, the protocol works as follows. First, $\mathbf{L}$ needs to add the label/value pair $(\mathbf{r}.\text{SN}, \mathbf{r}.\text{CID})$ to the global dictionary. To do this, it splits the pair into two shares p_1 and p_2 which it sends to the custodians. This guarantees that neither custodian will learn the pair. The custodians, then use 2PC to securely compute a function that: (1) recovers the key K to the global directory from their key shares; (2) recovers the serial/id pair from their pair shares; (3) computes an add token for the pair; and then (4) splits that token into two shares. The custodians then send the shares of the add token to the server which reconstructs it and uses to update the global directory.

After updating the global directory, the official updates its local (but remotely stored) encrypted database by generating an add token for the local database and sending it to the server.

Querying the global directory. When a querier $\mathbf{Q} \in \{\mathbf{P}_1, \ldots, \mathbf{P}_\theta\}$ wants to query the registry with a serial number SN but does not know in which county the gun is registered, it first queries the global directory EDX by executing the $\Omega.\mathsf{Find}$ protocol with the custodians $\mathbf{C}_1$ and $\mathbf{C}_2$ and the server. This results in $\mathbf{Q}$ recovering the identifier of the county that registered the gun. At a high level, the protocol works as follows. $\mathbf{Q}$ splits SN into two shares which it sends to the custodians. The custodians then use 2PC to securely

Let $\mathbf{P}_1, \ldots, \mathbf{P}_\theta$ be θ parties, each of which has 3 designated backups $\mathbf{B}_1^{(i)}, \mathbf{B}_2^{(i)}, \mathbf{B}_3^{(i)}$. Let $\mathbf{C}_1$ and $\mathbf{C}_2$ be two custodians, $\mathbf{S}$ be a server, $\mathbf{Q} \in \{\mathbf{P}_1, \ldots, \mathbf{P}_\theta\}$ be a querier and $\mathbf{L} \in \{\mathbf{P}_1, \ldots, \mathbf{P}_\theta\}$ be a local official. Let $\mathsf{SS} = (\mathsf{Share}, \mathsf{Recover})$ be a secret sharing scheme, $\Sigma_{\mathsf{DX}} = (\mathsf{Init}, \mathsf{Token}, \mathsf{Query}, \mathsf{AddToken}, \mathsf{ResKey}, \mathsf{Resolve})$ be a response-hiding dictionary encryption scheme with restricted resolve and $\Sigma_{\mathsf{DB}} = (\mathsf{Init}, \mathsf{Token}, \mathsf{Query}, \mathsf{AddToken}, \mathsf{Resolve})$ be a response-hiding database encryption scheme. Consider the multi-party protocol $\Omega = (\mathsf{InitGlobal}, \mathsf{InitLocal}, \mathsf{Add}, \mathsf{Find}, \mathsf{Query}, \mathsf{OfflineQuery})$ defined as follows in the $(\mathcal{F}_{\mathsf{2PC}}, \mathcal{F}_{\mathsf{MPC}})$-hybrid model:

- $\mathsf{InitGlobal}_{\mathbf{C}_1,\mathbf{C}_2,\mathbf{S}}(1^k, 1^k, \perp)$:
  1. $\mathbf{C}_1$ samples $r_1 \xleftarrow{\$} \{0,1\}^k$ and $\mathbf{C}_2$ samples $r_2 \xleftarrow{\$} \{0,1\}^k$;
  2. $\mathbf{C}_1$ and $\mathbf{C}_2$ execute $(K_1, (K_2, \mathsf{EDX})) \leftarrow \mathcal{F}_{\mathsf{2PC}}^f(r_1, r_2)$ where $f(r_1, r_2)$:
     (a) $(K, \mathsf{EDX}) \leftarrow \Sigma_{\mathsf{DX}}.\mathsf{Init}(1^k, r_1 \oplus r_2)$;
     (b) $(K_1, K_2) \leftarrow \mathsf{SS}.\mathsf{Share}(K, 2, 2)$;
     (c) output K_1 to $\mathbf{C}_1$ and (K_2, EDX) to $\mathbf{C}_2$;
  3. $\mathbf{C}_2$ sends EDX to $\mathbf{S}$;
- $\mathsf{InitLocal}_{\mathbf{L},\mathbf{S},\mathbf{B}_1,\mathbf{B}_2,\mathbf{B}_3}(1^k, \perp, \perp, \perp, \perp)$:
  1. $\mathbf{L}$ computes $(K_{\mathbf{L}}, \mathsf{EDB}_{\mathbf{L}}) \leftarrow \Sigma_{\mathsf{DB}}.\mathsf{Init}(1^k)$ and sends $\mathsf{EDB}_{\mathbf{L}}$ to $\mathbf{S}$;
  2. $\mathbf{L}$ computes $(K_{\mathbf{L},1}, K_{\mathbf{L},2}, K_{\mathbf{L},3}) \leftarrow \mathsf{SS}.\mathsf{Share}(K_{\mathbf{L}}, 2, 3)$ and sends $K_{\mathbf{L},i}$ to $\mathbf{B}_i$;
- $\mathsf{Add}_{\mathbf{L},\mathbf{C}_1,\mathbf{C}_2,\mathbf{S}}((K_{\mathbf{L}}, \mathbf{r}), K_1, K_2, \mathsf{EGR})$:
  1. $\mathbf{S}$ parses EGR as $(\mathsf{EDX}, \mathsf{EDB}_1, \ldots, \mathsf{EDB}_\theta)$;
  2. $\mathbf{L}$ computes $\mathsf{atk} \leftarrow \Sigma_{\mathsf{DB}}.\mathsf{AddToken}(K_{\mathbf{L}}, \mathbf{r})$ and sends it to $\mathbf{S}$;
  3. $\mathbf{S}$ computes $\mathsf{EDB}'_{\mathbf{L}} \leftarrow \Sigma_{\mathsf{DB}}.\mathsf{Add}(\mathsf{EDB}_{\mathbf{L}}, \mathsf{atk})$;
  4. let $\text{SN} = \mathbf{r}.\text{SN}$ and $\text{CID} = \mathbf{r}.\text{CID}$;
  5. $\mathbf{L}$ computes $(p_1, p_2) \leftarrow \mathsf{SS}.\mathsf{Share}(\text{SN}\|\text{CID}, 2, 2)$ and sends p_1 to $\mathbf{C}_1$ and p_2 to $\mathbf{C}_2$;
  6. $\mathbf{C}_1$ samples $r_1 \xleftarrow{\$} \{0,1\}^k$ and $\mathbf{C}_2$ samples $r_2 \xleftarrow{\$} \{0,1\}^k$;
  7. $\mathbf{C}_1$ and $\mathbf{C}_2$ execute $(\mathsf{atk}_1, \mathsf{atk}_2) \leftarrow \mathcal{F}_{\mathsf{2PC}}^f((K_1, p_1, r_1), (K_2, p_2, r_2))$ where $f((K_1, p_1, r_1), (K_2, p_2, r_2))$:
     (a) computes $K \leftarrow \mathsf{SS}.\mathsf{Recover}(K_1, K_2)$;
     (b) computes $\text{SN}\|\text{CID} \leftarrow \mathsf{SS}.\mathsf{Recover}(p_1, p_2)$;
     (c) computes $\mathsf{atk} \leftarrow \Sigma_{\mathsf{DX}}.\mathsf{AddToken}(K, (\text{SN}, \text{CID}))$;
     (d) computes $(\mathsf{atk}_1, \mathsf{atk}_2) \leftarrow \mathsf{SS}.\mathsf{Share}(\mathsf{atk}, 2, 2; r_1 \oplus r_2)$;
     (e) outputs atk_1 to $\mathbf{C}_1$ and atk_2 to $\mathbf{C}_2$
  8. $\mathbf{C}_1$ sends atk_1 to $\mathbf{S}$ and $\mathbf{C}_2$ sends atk_2 to $\mathbf{S}$;
  9. $\mathbf{S}$ computes $\mathsf{atk} \leftarrow \mathsf{SS}.\mathsf{Recover}(\mathsf{atk}_1, \mathsf{atk}_2)$ and $\mathsf{EDX}' \leftarrow \Sigma_{\mathsf{DX}}.\mathsf{Add}(\mathsf{EDX}, \mathsf{atk})$;

Figure 2: An encrypted registry (part 1).

- $\mathsf{Find}_{\mathbf{Q},\mathbf{C}_1,\mathbf{C}_2,\mathbf{S}}\,(\text{SN}, K_1, K_2, \mathsf{EDX})$:
  1. $\mathbf{Q}$ computes $(\text{SN}_1, \text{SN}_2) \leftarrow \mathsf{SS.Share}(\text{SN}, 2, 2)$ and sends SN_1 to $\mathbf{C}_1$ and SN_2 to $\mathbf{C}_2$;
  2. $\mathbf{C}_1$ samples $r_1, r_1' \xleftarrow{\$} \{0,1\}^k$ and $\mathbf{C}_2$ samples $r_2, r_2' \xleftarrow{\$} \{0,1\}^k$
  3. $\mathbf{C}_1$ and $\mathbf{C}_2$ execute $((\mathsf{tk}_1, K_{U_1}), (\mathsf{tk}_2, K_{U_2})) \leftarrow \mathcal{F}_{\mathsf{2PC}}^{f}\left((K_1, \text{SN}_1, r_1, r_1'), (K_2, \text{SN}_2, r_2, r_2')\right)$ where $f\left((K_1, \text{SN}_1, r_1, r_1'), (K_2, \text{SN}_2, r_2, r_2')\right)$:
     (a) recovers $K \leftarrow \mathsf{SS.Recover}(K_1, K_2)$;
     (b) computes $\text{SN} \leftarrow \mathsf{SS.Recover}(\text{SN}_1, \text{SN}_2)$;
     (c) computes $\mathsf{tk} \leftarrow \Sigma_{\mathsf{DX}}.\mathsf{Token}(K, \text{SN})$;
     (d) computes $K_R := \Sigma_{\mathsf{DX}}.\mathsf{ResKey}(K, \text{SN})$;
     (e) computes $(\mathsf{tk}_1, \mathsf{tk}_2) \leftarrow \mathsf{SS.Share}(\mathsf{tk}, 2, 2; r_1 \oplus r_2)$;
     (f) computes $(K_{R,1}, K_{R,2}) \leftarrow \mathsf{SS.Share}(K_R, 2, 2; r_1' \oplus r_2')$;
     (g) outputs $(\mathsf{tk}_1, K_{R,1})$ to $\mathbf{C}_1$ and $(\mathsf{tk}_2, K_{R,2})$ to $\mathbf{C}_2$
  4. $\mathbf{C}_1$ sends tk_1 to $\mathbf{S}$ and $K_{R,1}$ to $\mathbf{Q}$;
  5. $\mathbf{C}_2$ sends tk_2 to $\mathbf{S}$ and $K_{R,2}$ to $\mathbf{Q}$;
  6. $\mathbf{S}$ computes $\mathsf{tk} \leftarrow \mathsf{SS.Recover}(\mathsf{tk}_1, \mathsf{tk}_2)$ and $\text{ct} \leftarrow \mathsf{Query}(\mathsf{EDX}, \mathsf{tk})$ and sends ct to $\mathbf{Q}$;
  7. $\mathbf{Q}$ computes $K_R \leftarrow \mathsf{SS.Recover}(K_{R,1}, K_{R,2})$;
  8. $\mathbf{Q}$ computes $\text{CID} \leftarrow \Sigma_{\mathsf{DX}}.\mathsf{Resolve}(K_R, \text{ct})$;
- $\mathsf{Query}_{\mathbf{L},\mathbf{Q},\mathbf{S}}\,(K_{\mathbf{L}}, \varphi, \mathsf{EDB}_{\mathbf{L}})$
  1. $\mathbf{L}$ and $\mathbf{Q}$ execute $(\perp, \mathsf{tk}) \leftarrow \mathcal{F}_{\mathsf{2PC}}^{f}(K_{\mathbf{L}}, \varphi)$ where $f(K_{\mathbf{L}}, \varphi)$ and outputs $\mathsf{tk} \leftarrow \Sigma_{\mathsf{DB}}.\mathsf{Token}(K_{\mathbf{L}}, \varphi)$ to $\mathbf{Q}$.
  2. $\mathbf{Q}$ sends tk to $\mathbf{S}$;
  3. $\mathbf{S}$ computes $(\text{ct}_1, \ldots, \text{ct}_m) \leftarrow \Sigma_{\mathsf{DB}}.\mathsf{Search}(\mathsf{EDB}_{\mathbf{L}}, \mathsf{tk})$ and sends $(\text{ct}_1, \ldots, \text{ct}_m)$ to $\mathbf{Q}$;
  4. $\mathbf{L}$ and $\mathbf{Q}$ compute $(\perp, (\mathbf{r}_1, \ldots, \mathbf{r}_m)) \leftarrow \mathcal{F}_{\mathsf{2PC}}^{f}(K_{\mathbf{L}}, (\text{ct}_1, \ldots, \text{ct}_m))$ where $f(K_{\mathbf{L}}, (\text{ct}_1, \ldots, \text{ct}_m))$:
     (a) for all $1 \le i \le m$, computes $\mathbf{r}_i \leftarrow \Sigma_{\mathsf{DB}}.\mathsf{Resolve}(K_{\mathbf{L}}, \text{ct}_i)$
     (b) outputs $(\mathbf{r}_1, \ldots, \mathbf{r}_m)$ to $\mathbf{Q}$;
- $\mathsf{OfflineQuery}_{\mathbf{B}_1,\mathbf{B}_2,\mathbf{Q},\mathbf{S}}\,(K_1, K_2, \varphi, \mathsf{EDB}_{\mathbf{L}})$
  1. $\mathbf{B}_1$ samples $r_1 \xleftarrow{\$} \{0,1\}^k$, $\mathbf{B}_2$ samples $r_2 \xleftarrow{\$} \{0,1\}^k$ and $\mathbf{Q}$ samples $r_3 \xleftarrow{\$} \{0,1\}^k$;
  2. $\mathbf{B}_1$, $\mathbf{B}_2$ and $\mathbf{Q}$ execute $(K_1', \perp, K_2') \leftarrow \mathcal{F}_{\mathsf{MPC}}^{f}\big((K_1, r_1), (K_2, r_2), (\perp, r_3)\big)$ where f:
     (a) computes $K \leftarrow \mathsf{SS.Recover}(K_1, K_2)$;
     (b) computes $(K_1', K_2') \leftarrow \mathsf{SS.Share}(K, 2, 2; r_1 \oplus r_2 \oplus r_3)$;
     (c) outputs K_1' to $\mathbf{B}_1$ and K_2' to $\mathbf{Q}$;
  3. $\mathbf{B}_1$ and $\mathbf{Q}$ execute $(\perp, \mathsf{tk}) \leftarrow \mathcal{F}_{\mathsf{2PC}}^{f}\big(K_1', K_2'\big)$ where f:
     (a) computes $K \leftarrow \mathsf{SS.Recover}(K_1', (\varphi, K_2'))$;
     (b) computes $\mathsf{tk} \leftarrow \Sigma_{\mathsf{DB}}.\mathsf{Token}(K, \varphi)$;
     (c) outputs $\perp$ to $\mathbf{B}_1$ and tk to $\mathbf{Q}$;
  4. $\mathbf{Q}$ sends tk to $\mathbf{S}$;
  5. $\mathbf{S}$ computes $(\text{ct}_1, \ldots, \text{ct}_m) \leftarrow \Sigma_{\mathsf{DB}}.\mathsf{Query}(\mathsf{EDB}_{\mathbf{L}}, \mathsf{tk})$ and returns $(\text{ct}_1, \ldots, \text{ct}_m)$ to $\mathbf{Q}$;
  6. $\mathbf{B}_1$ and $\mathbf{Q}$ execute $(\perp, (\mathbf{r}_1, \ldots, \mathbf{r}_m)) \leftarrow \mathcal{F}_{\mathsf{2PC}}^{f}\big(K_1', (K_2', (\text{ct}_1, \ldots, \text{ct}_m))\big)$ where f:
     (a) computes $K \leftarrow \mathsf{SS.Recover}(K_1', K_2')$;
     (b) for all $1 \le i \le m$,
        i. computes $\mathbf{r}_i \leftarrow \Sigma_{\mathsf{DB}}.\mathsf{Resolve}(K, \text{ct}_i)$;
     (c) outputs $\perp$ to $\mathbf{B}_1$ and $(\mathbf{r}_1, \ldots, \mathbf{r}_m)$ to $\mathbf{Q}$;

Figure 3: An encrypted registry (part 2).

compute a function that: (1) recovers the key K to the EDX from their key shares; (2) recovers SN from their serial number pairs; (3) computes a token for the serial number; (4) generates the resolve key for the serial number; and (5) splits both the token and the resolve key into two shares, outputting a share of each to each custodian. The custodians then send their shares to the server and the querier who recovers the token and the resolve key, respectively. It then uses the token to query the global directory EDX and returns the encrypted result to $\mathbf{Q}$ who can then recover the county identifier CID using the resolve key.

Querying a local database. To query the database of a specific county on a boolean formula φ, such as (`(first:Jon or first:John) and last:Smith`), $\mathbf{Q}$ executes the $\Omega.\mathsf{Query}$ protocol with the local official $\mathbf{L} \in \{\mathbf{P}_1, \ldots, \mathbf{P}_\theta\}$ and the server. This results in $\mathbf{Q}$ recovering the plaintext records that match its query. At a high level, the protocol works as follows. $\mathbf{L}$ and $\mathbf{Q}$ use 2PC to securely compute a function that generates a token for φ using the key $K_{\mathbf{L}}$ for $\mathbf{L}$'s local database. $\mathbf{Q}$ sends the token to the server who uses it to query $\mathbf{L}$'s database and returns an encrypted response to $\mathbf{Q}$. $\mathbf{Q}$ and $\mathbf{L}$ then execute another 2PC to resolve the encrypted response into a set of plaintext records for $\mathbf{Q}$.

Offline querying. If $\mathbf{Q}$ wishes to query $\mathbf{L}$'s local database when $\mathbf{L}$ is offline, it interacts with $\mathbf{L}$'s backups, $\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3$, and the server. Here, we refer to $\mathbf{B}_1$ as the *designated* backup because it will play a slightly different role than the other backups but we note that any of $\mathbf{B}_1$, $\mathbf{B}_2$ or $\mathbf{B}_3$ could be the designated backup. First, $\mathbf{Q}$ and the backups use MPC to securely compute a function that: (1) recovers the local official's key from the backups' shares; (2) splits the key into two new shares; and (3) outputs one share to the designated backup $\mathbf{B}_1$ and one to $\mathbf{Q}$. $\mathbf{B}_1$ and $\mathbf{Q}$ then use 2PC to securely compute a function that: (1) recovers the key from the new shares; (2) computes a token for $\mathbf{Q}$'s query φ. $\mathbf{Q}$ then sends the token to the server who uses it to recover and return a set of encrypted records that match the query. $\mathbf{Q}$ and $\mathbf{B}_1$ then use 2PC to securely resolve the encrypted response.

5.1 Deployment Considerations

As described, our protocol does not capture many real-world considerations that would have to be taken into account to deploy it. Here, we discuss some of those issues.

Local infrastructure. One can expect local county officials to manage only a minimal computing infrastructure. Specifically, a consumer-level desktop computer or laptop but not a server-level machine. In addition, especially in rural regions, one should expect poor Internet connectivity and intermittent access. These challenges motivate two important features of our protocol: (1) a relatively lightweight amount of computation for the officials; (2) no storage requirements for officials (besides the secret key); and (3) an offline query protocol in case the official is disconnected from the Internet.

Licensing. In practice, our protocol would be used to store two kinds of records: (1) licenses, which are issued to individuals who wish to own a gun; and (2) firearm registrations, which are issued when a gun is purchased. In our system, both licenses and registrations can be stored as records so we do not differentiate between them.

Rate limiting. To prevent a querier from making excessive queries, rate limiting can be achieved by both the server and the official. The server can rate limit when it receives search tokens since it can keep track of the number of encrypted records it has returned from the official's encrypted database. Furthermore, it can be set to cap the maximum number of records it returns per query. Similarly, the local official can also rate limit during the execution of the query protocol; specifically during the second 2PC execution where the encrypted results are resolved. Here, the official could simply refuse to execute the 2PC if the number of records returned by the search exceeds some threshold.

Moving & history. When a gun owner moves from county A to county B, it is expected to re-register the gun in county B. Note that, in our protocol, the new registration would overwrite the old serial number and county ID pair in the global directory and insert a new record in county B's local database. The old record in county A would still persist but this is by design so that a history of the gun can be recovered. The new record in county B's database would include the old county ID so that the two records are linked. If, on the other hand, one needed to support deletion from county A's database it would suffice to instantiate Σ_{DB} with

a dynamic database encryption scheme instead of a semi-dynamic one. There are many such constructions one could choose from [5, 10, 11].

Sales. If a gun is sold by owner A to owner B then a new serial number/county identifier pair will be created in the global directory and a new record will be added to a local database. To keep history, the new record will contain the license number of owner A. Note that if multiple records are found for the same serial number in the same county, they can be ordered using the previous owner's license number, e.g., if record Y's previous owner is the owner of record X then record Y was created after record X.

Custodians. An important deployment consideration for our protocol is the choice of custodians. The security of the protocol relies on the custodians not colluding so they should be picked carefully. One could imagine choosing, for example, gun rights and civil liberties organizations like the National Rifle Association (NRA) and the ACLU under the assumption that local officials would trust that the NRA would not collude with the ACLU in order to subvert the system and recover the private information of gun owners. Also, we note that in our protocol description and prototype we use two custodians but this number can be easily increased to any number in the natural way (i.e., increasing the number of shares used throughout the protocol and using MPC instead of 2PC).

Batch updates. For ease of exposition, we describe the Add operation of our protocol as taking a single record to add to the database. In practice, however, local officials may prefer to add a batch of new records (e.g., one per day or week). The naive way to handle this is to execute the Add protocol on each record in the batch but a more efficient approach (which we implement) is to process the entire batch of new records at once and to execute the 2PC in Add over multiple records.

Number of backups. In our protocol description and prototype, we chose to use three backups but this can be trivially extended to any number.

Removing local databases. As discussed in Section 4, an important requirement is that local officials have the ability to remove their database from the system at any point in time. This feature is easy to achieve in our protocol since the local databases are all end-to-end encrypted. In fact, to remove a database from the system it suffices to erase the secret key. A more usable approach could be to store the secret key on a hardware token like a Yubikey that remains connected to the official's device and to physically remove it in order to pull the database. Note that to removing the database should include asking the backups to erase their shares. If enough backups are honest and erase their shares then no keying material will remain. Alternatively, one could augment our design to include a form of key rotation so that the shares become useless or a revocation mechanism to revoke shares.

6 Security Definition and Proof

We formalize the security of our design in the ideal/real-world paradigm [13]. Roughly speaking, we require that an execution of the protocol in the real-world is indistinguishable from an ideal gun registry functionality which we define below.

Parties. The two executions take place between an environment $\mathcal{Z}$, an adversary which we denote $\mathcal{A}$ in the real-world execution and $\mathcal{S}$ in the ideal-world execution, $\theta = \mathsf{poly}(k)$ parties $\mathbf{P}_1, \ldots, \mathbf{P}_\theta$, 3θ backup parties $\mathbf{B}_1^{(1)}, \mathbf{B}_2^{(1)}, \mathbf{B}_3^{(1)}, \ldots, \mathbf{B}_1^{(\theta)}, \mathbf{B}_2^{(\theta)}, \mathbf{B}_3^{(\theta)}$, two custodians $\mathbf{C}_1$ and $\mathbf{C}_2$ and a server $\mathbf{S}$. While the legislation proposes that every state manage a server for its counties, in our model we only include a single server $\mathbf{S}$ for ease of exposition and because it captures the worst-case scenario from a security point of view (i.e., the case where all state-level servers are corrupted and collude).

Corruptions. We consider two classes of corruptions: *external* corruptions and *internal* corruptions. External corruptions can include: (1) the server; (2) either $\mathbf{C}_1$ or $\mathbf{C}_2$ but not both; and (3) for all parties $\mathbf{P}_i$, at most 1 of $\mathbf{P}_i$'s backup parties. Internal corruptions can include at most one party in $\{\mathbf{P}_1, \ldots, \mathbf{P}_\theta\}$.

Hybrid-world execution. In the hybrid-world execution every party has access to ideal $\mathcal{F}_{\mathsf{2PC}}$ and $\mathcal{F}_{\mathsf{MPC}}$ functionalities. The environment $\mathcal{Z}$ takes as input a string $z \in \{0,1\}^*$ and starts by choosing a set of parties

Functionality $\mathcal{F}_{\mathsf{GR}}^{\Lambda}$

The functionality is parameterized with a leakage profile $\Lambda = (\mathcal{L}_{\mathsf{IG}}, \mathcal{L}_{\mathsf{IL}}, \mathcal{L}_{\mathsf{A}}, \mathcal{L}_{\mathsf{F}}, \mathcal{L}_{\mathsf{Q}}, \mathcal{L}_{\mathsf{O}})$ and interacts with θ parties $\{\mathbf{P}_1, \ldots, \mathbf{P}_\theta\}$, 3θ backups $\mathbf{B}_1^{(1)}, \mathbf{B}_2^{(1)}, \mathbf{B}_3^{(1)}, \ldots, \mathbf{B}_1^{(\theta)}, \mathbf{B}_2^{(\theta)}, \mathbf{B}_3^{(\theta)}$, two custodians $\mathbf{C}_1$ and $\mathbf{C}_2$ and an ideal adversary $\mathcal{S}$. It stores and manages a gun registry $\mathsf{GR} = (\mathsf{DX}, \mathsf{DB}_1, \ldots, \mathsf{DB}_\theta)$ using the following operations:

- upon receiving ($\mathsf{initglobal}$) from $\mathbf{C}_1$, $\mathbf{C}_2$ and $\mathbf{S}$, initialize and store an empty dictionary DX and send the message ($\mathsf{globalinitialized}$) to $\mathbf{C}_1$, $\mathbf{C}_2$ and $\mathbf{S}$ and the message $\mathcal{L}_{\mathsf{IG}}(1^k)$ to $\mathcal{S}$;
- upon receiving ($\mathsf{initlocal}$) from $\mathbf{P}_i$, initialize and store an empty database EDB and send the message ($\mathsf{localinitialized}$) to $\mathbf{P}_i$ and $\mathbf{S}$ and the message $(i, \mathcal{L}_{\mathsf{IL}}(1^k))$ to $\mathcal{S}$;
- upon receiving $(\mathsf{add}, \mathbf{r})$ from $\mathbf{L} = \mathbf{P}_i$ store $\mathbf{r}$ in DB_i and add the pair $(\mathbf{r}.\text{SN}, \mathbf{r}.\text{CID})$ to DX. Send the message (add, i) to $\mathbf{C}_1$, $\mathbf{C}_2$ and $\mathbf{S}$ and the message $\big(i, \mathcal{L}_{\mathsf{A}}(\mathsf{GR}, \mathbf{r})\big)$ to the ideal adversary $\mathcal{S}$;
- upon receiving $(\mathsf{find}, \text{SN})$ from $\mathbf{Q} = \mathbf{P}_i$ compute $\text{CID} := \mathsf{DX}[\text{SN}]$. Return CID to $\mathbf{Q}$ and send the message (find, i) to $\mathbf{C}_1$, $\mathbf{C}_2$ and $\mathbf{S}$ and the message $\big(i, \mathcal{L}_{\mathsf{F}}(\mathsf{GR}, \varphi)\big)$ to the ideal adversary $\mathcal{S}$ if the corruptions are external and $\big(i, \mathcal{L}_{\mathsf{F}}(\mathsf{GR}, \varphi), \text{SN}, \text{CID}\big)$ if the corruptions are internal.
- upon receiving $(\mathsf{query}, \text{CID}, \varphi)$ from $\mathbf{Q} = \mathbf{P}_i$ return the records $(\mathbf{r}_1, \ldots, \mathbf{r}_m)$ in DB_{CID} that match φ to $\mathbf{Q}$. Send the message (query, i) to $\mathbf{S}$ and the message $\big(i, \mathcal{L}_{\mathsf{Q}}(\mathsf{GR}, \varphi)\big)$ to the ideal adversary $\mathcal{S}$ if the corruptions are external and $\big(i, \mathcal{L}_{\mathsf{Q}}(\mathsf{GR}, \varphi), \varphi, (\mathbf{r}_1, \ldots, \mathbf{r}_m)\big)$ if the corruptions are internal.
- upon receiving $(\mathsf{offline}, \text{CID}, \varphi)$ from $\mathbf{Q} = \mathbf{P}_i$ return the records $(\mathbf{r}_1, \ldots, \mathbf{r}_n)$ in DB_{CID} that match φ to $\mathbf{Q}$. Send the message (query, i) to $\mathbf{B}_1^{(\text{CID})}$, $\mathbf{B}_2^{(\text{CID})}$, $\mathbf{B}_3^{(\text{CID})}$ and $\mathbf{S}$ and the message $\big(i, \mathcal{L}_{\mathsf{O}}(\mathsf{GR}, \varphi)\big)$ to the ideal adversary $\mathcal{S}$ if the corruptions are external and $\big(i, \mathcal{L}_{\mathsf{O}}(\mathsf{GR}, \varphi), \varphi, (\mathbf{r}_1, \ldots, \mathbf{r}_m)\big)$ if the corruptions are internal.

Figure 4: $\mathcal{F}_{\mathsf{GR}}^{\Lambda}$: The registry functionality parameterized with leakage profile Λ.

I for the adversary to corrupt, where I is either external or internal. $\mathcal{Z}$ sends I to $\mathcal{A}$ which corrupts all the parties in I. After the parties in I have been corrupted, $\mathbf{C}_1$, $\mathbf{C}_2$ and $\mathbf{S}$ execute $\Omega.\mathsf{InitGlobal}$ and each party $\mathbf{P}_i$ executes $\Omega.\mathsf{InitLocal}$ with $\mathbf{S}$ and its backup parties $\mathbf{B}_1^{(i)}$, $\mathbf{B}_2^{(i)}$ and $\mathbf{B}_3^{(i)}$.

$\mathcal{Z}$ then adaptively chooses a polynomial number of commands $(\mathsf{comm}_1, \ldots, \mathsf{comm}_m)$ of the form $\mathsf{comm}_j = (\mathbf{P}_j, \mathsf{op}_j)$, where op_j is either an add operation $(\mathsf{add}, \mathbf{r}_i)$, a find operation $(\mathsf{find}, \text{SN})$, a query operation $(\mathsf{query}, \varphi)$ or an offline query operation $(\mathsf{offline}, \varphi)$. More precisely, for $1 \leq j \leq m$, $\mathcal{Z}$ sends comm_j to $\mathbf{P}_j$. If op_j is an add, $\mathbf{P}_j$ executes $\Omega.\mathsf{Add}$ with the custodians and the server. If op_j is a find, $\mathbf{P}_j$ executes $\Omega.\mathsf{Find}$ with the custodians and the server. If op_j is a query, $\mathbf{P}_j$ executes $\Omega.\mathsf{Query}$ with a local official $\mathbf{L} \in \{\mathbf{P}_1, \ldots, \mathbf{P}_\theta\}$ and the server. If op_j is an offline query, $\mathbf{Q}$ executes $\Omega.\mathsf{OfflineQuery}$ with the local official's backup parties and the server. In all cases, $\mathbf{Q}$ returns its output to $\mathcal{Z}$. At then end of the execution, $\mathcal{A}$ sends an arbitrary message to $\mathcal{Z}$ which outputs a bit b. We denote this bit $\mathbf{Hybrid}_{\mathcal{Z},\mathcal{A}}(k)$.

Ideal-world execution. In the ideal-world execution every party has access to an ideal functionality $\mathcal{F}_{\mathsf{GR}}$ described in Figure 4. The environment $\mathcal{Z}$ takes as input a string $z \in \{0,1\}^*$ and starts by choosing a set of parties I for the adversary to corrupt, where I is either external or internal.

$\mathcal{Z}$ sends I to $\mathcal{S}$ and $\mathcal{F}_{\mathsf{GR}}$ and adaptively chooses a polynomial number of commands $(\mathsf{comm}_1, \ldots, \mathsf{comm}_m)$ of the above form. More precisely, for $1 \leq j \leq m$, $\mathcal{Z}$ sends comm_j to $\mathbf{P}_j$ who, in turn, forwards the operation op_j to $\mathcal{F}_{\mathsf{GR}}$ and returns its output to $\mathcal{Z}$. At the end of the execution, $\mathcal{S}$ sends an arbitrary message to $\mathcal{Z}$ which outputs a bit b. We denote this bit $\mathbf{Ideal}_{\mathcal{Z},\mathcal{S}}^{\Lambda}(k)$.

Definition. We can now state our security definition based on the above experiments.

Definition 6.1 (Security)**.** *We say that Ω is a Λ-secure registry if for all* PPT *semi-honest adversaries $\mathcal{A}$, there exists a* PPT *ideal adversary $\mathcal{S}$ such that for all* PPT *standalone environments $\mathcal{Z}$, for all $z \in \{0,1\}^*$,*

$$\Big|\Pr\big[\,\mathbf{Hybrid}_{\mathcal{Z},\mathcal{A}}(k) = 1\,\big] - \Pr\Big[\,\mathbf{Ideal}_{\mathcal{Z},\mathcal{S}}^{\Lambda}(k) = 1\,\Big]\Big| \leq \mathsf{negl}(k).$$

7 Security Analysis

We conduct a leakage analysis of our protocol and formalize its leakage profile. We first provide a black-box leakage analysis and then a concrete one.

Black-box leakage analysis. Black-box leakage analysis, introduced in [30], is a way to describe the leakage profile of a scheme or protocol as a function of the leakage profiles of its underlying building blocks. The value of such an analysis is that it remains useful even when the protocol's building blocks are replaced or instantiated with different concrete schemes. In particular, this means that as new schemes are developed with more desirable tradeoffs the protocol's leakage profile can be easily updated. Suppose the leakage profile of Σ_{DX} is

$$\Lambda_{\mathsf{DX}} = \left(\mathcal{L}_{\mathsf{I}}^{\mathsf{dx}}, \mathcal{L}_{\mathsf{Q}}^{\mathsf{dx}}, \mathcal{L}_{\mathsf{A}}^{\mathsf{dx}}\right) = \left(\mathsf{patt}_{\mathsf{I}}^{\mathsf{dx}}, \mathsf{patt}_{\mathsf{Q}}^{\mathsf{dx}}, \mathsf{patt}_{\mathsf{A}}^{\mathsf{dx}}\right)$$

and that the leakage profile of Σ_{DB} is

$$\Lambda_{\mathsf{DB}} = \left(\mathcal{L}_{\mathsf{I}}^{\mathsf{db}}, \mathcal{L}_{\mathsf{Q}}^{\mathsf{db}}, \mathcal{L}_{\mathsf{A}}^{\mathsf{db}}\right) = \left(\mathsf{patt}_{\mathsf{I}}^{\mathsf{db}}, \mathsf{patt}_{\mathsf{Q}}^{\mathsf{db}}, \mathsf{patt}_{\mathsf{A}}^{\mathsf{db}}\right)$$

then the leakage profile of Ω is

$$\begin{aligned}\Lambda_{\Omega} &= \left(\mathcal{L}_{\mathsf{IG}}^{\Omega}, \mathcal{L}_{\mathsf{IL}}^{\Omega}, \mathcal{L}_{\mathsf{A}}^{\Omega}, \mathcal{L}_{\mathsf{F}}^{\Omega}, \mathcal{L}_{\mathsf{Q}}^{\Omega}, \mathcal{L}_{\mathsf{OQ}}^{\Omega}\right) \\ &= \left(\mathsf{patt}_{\mathsf{I}}^{\mathsf{dx}}, \mathsf{patt}_{\mathsf{I}}^{\mathsf{db}}, \left(\mathsf{patt}_{\mathsf{A}}^{\mathsf{dx}}, \mathsf{patt}_{\mathsf{A}}^{\mathsf{db}}\right), \mathsf{patt}_{\mathsf{Q}}^{\mathsf{dx}}, \mathsf{patt}_{\mathsf{Q}}^{\mathsf{db}}, \mathsf{patt}_{\mathsf{Q}}^{\mathsf{db}}\right).\end{aligned}$$

Theorem 7.1. If SS is secure, Σ_{DX} is Λ_{DX}-secure and Σ_{DB} is Λ_{DB}-secure, then the registry Ω described in Figures 2 and 3 is Λ_{Ω}-secure.

Proof. Let $\mathbf{Sim}_{\mathsf{DX}}$ be the simulator guaranteed to exist by the Λ_{DX}-security of Σ_{DX} and $\mathbf{Sim}_{\mathsf{DB}}$ be the simulator guaranteed to exist by the Λ_{DB} of Σ_{DB}. Consider the simulator $\mathcal{S}$ that simulates $\mathcal{A}$ and works as follows in the context of external corruptions:

- **(simulating** InitGlobal**)** simulate $\mathcal{F}_{\mathsf{2PC}}^{f}$ and compute $\mathsf{EDX} \leftarrow \mathbf{Sim}_{\mathsf{DX}}(\mathcal{L}_{\mathsf{I}}^{\mathsf{dx}}(1^k))$ and $(K_1, K_2) \leftarrow \mathsf{SS.Share}(0^k, 2, 2)$. Send K_1 to $\mathbf{C}_1$ and and K_2 and EDX to $\mathbf{C}_2$;
- **(simulating** InitLocal**)** compute $\mathsf{EDB} \leftarrow \mathbf{Sim}_{\mathsf{DB}}(\mathcal{L}_{\mathsf{I}}^{\mathsf{db}}(1^k))$, sample $K_i \leftarrow \{0,1\}^k$ and compute $(K_{i,1}, K_{i,2}, K_{i,3}) \leftarrow \mathsf{SS.Share}(K_i, 2, 3)$. Send EDB to $\mathbf{S}$, and the shares $K_{i,1}$, $K_{i,2}$ and $K_{i,3}$ to $\mathbf{B}_1$, $\mathbf{B}_2$ and $\mathbf{B}_3$, respectively.
- **(simulating** Add**)** compute $\mathsf{atk} \leftarrow \mathbf{Sim}_{\mathsf{DB}}(\mathcal{L}_{\mathsf{A}}^{\mathsf{db}}(\mathsf{DB}_i, \mathbf{r}))$ and send it to $\mathbf{S}$. Compute $(p_1, p_2) \leftarrow \mathsf{SS.Share}(0^{|\mathrm{SN}|+|\mathrm{CID}|}, 2, 2)$ and send p_1 and p_2 to $\mathbf{C}_1$ and $\mathbf{C}_2$, respectively. Simulate $\mathcal{F}_{\mathsf{2PC}}^{f}$ and compute $\mathsf{atk} \leftarrow \mathbf{Sim}_{\mathsf{DX}}(\mathcal{L}_{\mathsf{A}}^{\mathsf{dx}}(\mathsf{DX}, \mathrm{SN} \| \mathrm{CID}))$. Generate the shares $(\mathsf{atk}_1, \mathsf{atk}_2) \leftarrow \mathsf{SS.Share}(\mathsf{atk}, 2, 2)$ and output atk_1 to $\mathbf{C}_2$ and atk_2 to $\mathbf{C}_2$.
- **(simulating** Find**)** compute $(\mathrm{SN}_1, \mathrm{SN}_2) \leftarrow \mathsf{SS.Share}(0^{|\mathrm{SN}|}, 2, 2)$ and send SN_1 and SN_2 to $\mathbf{C}_1$ and $\mathbf{C}_2$, respectively. Simulate $\mathcal{F}_{\mathsf{2PC}}^{f}$ and compute $\mathsf{tk} \leftarrow \mathbf{Sim}_{\mathsf{DX}}(\mathcal{L}_{\mathsf{Q}}^{\mathsf{dx}}(\mathsf{DX}, \mathrm{SN}))$. Generate the shares $(\mathsf{tk}_1, \mathsf{tk}_2) \leftarrow \mathsf{SS.Share}(\mathsf{tk}, 2, 2)$ and $(K_{R,1}, K_{R,2}) \leftarrow \mathsf{SS.Share}(0^k, 2, 2)$ and output $(\mathsf{tk}_1, K_{R,1})$ to $\mathbf{C}_1$ and $(\mathsf{tk}_2, K_{R,2})$ to $\mathbf{C}_2$.
- **(simulating** Query**)** compute $\mathsf{tk} \leftarrow \mathbf{Sim}_{\mathsf{DB}}(\mathcal{L}_{\mathsf{Q}}^{\mathsf{db}}(\mathsf{DB}_{\mathbf{L}}, \varphi))$ and send tk to $\mathbf{S}$.
- **(simulating** OfflineQuery**)** simulate the first $\mathcal{F}_{\mathsf{2PC}}^{f}$ interaction and compute $(K_1', K_2') \leftarrow \mathsf{SS.Share}(0^k)$ and send K_1' to $\mathbf{B}_1$ and K_2' to $\mathbf{Q}$. Simulate the second $\mathcal{F}_{\mathsf{2PC}}^{f}$ execution and compute $\mathsf{tk} \leftarrow \mathbf{Sim}_{\mathsf{DB}}(\mathcal{L}_{\mathsf{Q}}^{\mathsf{db}}(\mathsf{DB}_{\mathbf{L}}, \varphi))$ and output tk to $\mathbf{Q}$ and $\perp$ to $\mathbf{B}_1$. Finally, simulate the $\mathcal{F}_{\mathsf{MPC}}^{f}$ execution and output $\perp$ to $\mathbf{B}_1$.

We now show that $\mathcal{A}$'s simulated view in an $\mathbf{Ideal}_{\mathcal{Z},\mathcal{S}}(k)$ experiment is indistinguishable from its view in a $\mathbf{Hybrid}_{\mathcal{Z},\mathcal{A}}(k)$ experiment. We do this using the following sequence of games:

- Game_0: is an execution of a $\mathbf{Hybrid}(k)$ experiment;
- Game_1: is the same as Game_0 except that the $\mathcal{F}_{\mathsf{2PC}}^f$ functionalities are simulated. Clearly, the adversary's view is not affected by this change.
- Game_2: is the same as Game_1 except that the $\mathcal{F}_{\mathsf{MPC}}^f$ functionality in $\mathsf{OfflineQuery}$ is simulated. Clearly, the adversary's view is not affected by this change.
- Game_3: is the same as Game_2 except that in step 2-a of $\mathsf{InitGlobal}$, we replace $r_1 \oplus r_2$ with a random k-bit string r. Since $\mathcal{A}$ corrupts at most one of $\mathbf{C}_1$ or $\mathbf{C}_2$ it follows that $r_1 \oplus r_2$ is distributed exactly as r and therefore $\mathcal{A}$'s view is (perfectly) indistinguishable from its view in Game_1.
- Game_4: is the same as Game_3 except that in step 7-d of Add, we replace $r_1 \oplus r_2$ with a random k-bit string r. As above, this does not affect $\mathcal{A}$'s view.
- Game_5: is the same as Game_4 except that in step 3-e and 3-f of Find, we replace $r_1 \oplus r_2$ and $r'_1 \oplus r'_2$ with random k-bit strings r and r'. As above, this does not affect $\mathcal{A}$'s view.
- Game_6: is the same as Game_5 except that in step 2-b of $\mathsf{OfflineQuery}$, we replace $r_1 \oplus r_2 \oplus r_3$ with a random k-bit string r. As above, this does not affect $\mathcal{A}$'s view.
- Game_7: is the same as Game_6 except that the global directory EDX is replaced with a simulated global directory as follows:
  - in step 2-a of $\mathsf{InitGlobal}$, we replace K and EDX with 0^k and $\mathsf{EDX} \leftarrow \mathbf{Sim}_{\mathsf{DX}}(\mathcal{L}_{\mathsf{I}}^{\mathsf{dx}}(1^k))$, respectively.
  - in step 7-c of Add, we replace atk with $\mathsf{atk} \leftarrow \mathbf{Sim}_{\mathsf{DX}}(\mathcal{L}_{\mathsf{A}}^{\mathsf{dx}}(\mathsf{DX}, \text{SN} \| \text{CID}))$
  - in step 3-c of Find, we replace tk with $\mathsf{tk} \leftarrow \mathbf{Sim}_{\mathsf{DX}}(\mathcal{L}_{\mathsf{Q}}^{\mathsf{dx}}(\mathsf{DX}, \text{SN}))$;

  Since $\mathcal{A}$ corrupts at most one of $\mathbf{C}_1$ and $\mathbf{C}_2$, the security of SS guarantees that we can replace K with 0^k without affecting $\mathcal{A}$'s view. Furthermore, the Λ_{DX} security of Σ_{DX} guarantees that the simulated global directory EDX and all the simulated query and add tokens are computationally indistinguishable from a real global directory and tokens.
- $\mathsf{Game}_{8,\ldots,8+\theta}$: Game_i, for $8 \leq i \leq 8+\theta$, is the same as Game_{i-1} except that we replace party $\mathbf{P}_i$'s encrypted database EDB_i with a simulated encrypted database as follows:
  - in step 1 of $\mathsf{InitLocal}$, we replace $K_{\mathbf{L}}$ and EDB_i with 0^k and $\mathsf{EDB}_i \leftarrow \mathbf{Sim}_{\mathsf{DB}}(\mathcal{L}_{\mathsf{I}}^{\mathsf{db}}(1^k))$;
  - in step 2 of Add, we replace atk with $\mathsf{atk} \leftarrow \mathbf{Sim}_{\mathsf{DB}}(\mathcal{L}_{\mathsf{A}}^{\mathsf{db}}(\mathsf{DB}_i, \mathbf{r}))$;
  - in step 1-a of Query, we replace tk with $\mathsf{tk} \leftarrow \mathbf{Sim}_{\mathsf{DB}}(\mathcal{L}_{\mathsf{Q}}^{\mathsf{db}}(\mathsf{DB}_{\mathbf{L}}, \varphi))$;
  - in step 3-b of $\mathsf{OfflineQuery}$, we replace tk with $\mathsf{tk} \leftarrow \mathbf{Sim}_{\mathsf{DB}}(\mathcal{L}_{\mathsf{Q}}^{\mathsf{db}}(\mathsf{DB}_{\mathbf{L}}, \varphi))$.

  Since $\mathcal{A}$ corrupts at most one of $\mathbf{C}_1$ and $\mathbf{C}_2$, the security of SS guarantees that we can replace K with 0^k without affecting $\mathcal{A}$'s view. Furthermore, the Λ_{DB} security of Σ_{DB} guarantees that the simulated encrypted database EDB and all the simulated query and add tokens are computationally indistinguishable from a real encrypted database and tokens.

Finally, note that $\mathsf{Game}_{8+\theta}$ is equivalent to an $\mathbf{Ideal}_{\mathcal{Z},\mathcal{S}}(k)$ execution.

We now turn to internal corruptions. Note that if $\mathbf{L} = \mathbf{P}_i$ is corrupted then its view only includes messages from Query. In particular, it can be simulated by simulating the two $\mathcal{F}_{\mathsf{2PC}}^f$ execution and returning $\perp$ to $\mathbf{Q}$ in each case. The indistinguishability of $\mathcal{A}$'s view in this case is trivial. If $\mathbf{Q} = \mathbf{P}_i$ is corrupted, consider the simulator $\mathcal{S}$ that simulates $\mathcal{A}$ and works as follows:

- **(simulating Find)** recall that in this setting, $\mathcal{S}$ receives CID from the functionality. Compute $(\mathrm{ct}, K_R) \leftarrow \mathbf{Sim}_{\mathsf{DX}}(\mathrm{CID})$ and $(K_{R,1}, K_{R,2}) \leftarrow \mathsf{SS.Share}(K_R, 2, 2)$. Send $K_{R,1}$, $K_{R,2}$ and ct to $\mathbf{Q}$.
- **(simulating Query)** recall that in this setting, $\mathcal{S}$ receives $(\mathbf{r}_1, \ldots, \mathbf{r}_m)$ from the functionality. Simulate the first $\mathcal{F}^f_{\mathsf{2PC}}$ execution, compute $\mathsf{tk} \leftarrow \mathbf{Sim}_{\mathsf{DB}}(\mathcal{L}^{\mathsf{DB}}_{\mathsf{Q}}(\mathsf{DB}_{\mathbf{L}}, \varphi))$ and output tk to $\mathbf{Q}$. Sample $\mathrm{ct}_1, \ldots, \mathrm{ct}_m \xleftarrow{\$} \{0,1\}^{|\mathbf{r}|}$ and send $(\mathrm{ct}_1, \ldots, \mathrm{ct}_m)$ to $\mathbf{Q}$. Simulate the second $\mathcal{F}^f_{\mathsf{2PC}}$ execution and output $(\mathbf{r}_1, \ldots, \mathbf{r}_m)$ to $\mathbf{Q}$
- **(simulating OfflineQuery)** recall that, as above, in this setting $\mathcal{S}$ receives $(\mathbf{r}_1, \ldots, \mathbf{r}_m)$ from the functionality. Simulate the $\mathcal{F}^f_{\mathsf{MPC}}$ execution, compute $(K'_1, K'_2) \leftarrow \mathsf{SS.Share}(0^k, 2, 2)$ and output K'_2 to $\mathbf{Q}$. Simulate the first $\mathcal{F}^f_{\mathsf{2PC}}$ execution, compute $\mathsf{tk} \leftarrow \mathbf{Sim}_{\mathsf{DB}}(\mathcal{L}^{\mathsf{db}}_{\mathsf{Q}}(\mathsf{DB}_{\mathbf{L}}, \varphi))$ and output tk to $\mathbf{Q}$. Sample $\mathrm{ct}_1, \ldots, \mathrm{ct}_m \xleftarrow{\$} \{0,1\}^{|\mathbf{r}|}$ and send $(\mathrm{ct}_1, \ldots, \mathrm{ct}_m)$ to $\mathbf{Q}$. Finally, simulate the second $\mathcal{F}^f_{\mathsf{2PC}}$ execution and output $(\mathbf{r}_1, \ldots, \mathbf{r}_m)$ to $\mathbf{Q}$.

We now show that $\mathcal{A}$'s simulated view in an $\mathbf{Ideal}_{\mathcal{Z},\mathcal{S}}(k)$ experiment is indistinguishable from its view in a $\mathbf{Hybrid}_{\mathcal{Z},\mathcal{A}}(k)$ experiment. We do this using the following sequence of games:

- Game_0: is an execution of a $\mathbf{Hybrid}_{\mathcal{Z},\mathcal{A}}(k)$ experiment;
- Game_1: is the same as Game_0 except that all the $\mathcal{F}^f_{\mathsf{2PC}}$ functionalities are simulated. Clearly, the adversary's view is not affected by this change.
- Game_2: is the same as Game_1 except that the $\mathcal{F}^f_{\mathsf{MPC}}$ functionalities are simulated. Clearly, the adversary's view is not affected by this change.
- Game_3: is the same as Game_2 except for the following. We compute $(\mathrm{ct}', K'_R) \leftarrow \mathbf{Sim}_{\mathsf{DX}}(\mathrm{CID})$ and $(K'_{R,1}, K'_{R,2}) \leftarrow \mathsf{SS.Share}(K'_R, 2, 2)$. In steps 4 and 5 of Find we replace $K_{R,1}$ and $K_{R,2}$ with $K'_{R,1}$ and $K'_{R,2}$. In step 6 we replace ct with ct′. We store a mapping between CID and (ct', K'_R) so that we can reuse the latter consistently for CID. The security of schemes with restricted resolve guarantees that this change does not affect $\mathcal{A}$'s view.
- Game_4: is the same as Game_3 except for the following. In step 1-a of Query we replace tk with $\mathsf{tk} \leftarrow \mathbf{Sim}_{\mathsf{DB}}(\mathcal{L}^{\mathsf{DB}}_{\mathsf{Q}}(\mathsf{EDB}_{\mathbf{L}}, \varphi))$. In step 3 we replace $\mathrm{ct}_1, \ldots, \mathrm{ct}_m$ with $\mathrm{ct}_1, \ldots, \mathrm{ct}_m \xleftarrow{\$} \{0,1\}^{|\mathbf{r}|}$ and in step 4-b we return $(\mathbf{r}_1, \ldots, \mathbf{r}_m)$. The pseudo-randomness of the ciphertexts guarantees that they are computationally indistinguishable from real ciphertexts and the Λ_{DB} security of Σ_{DB} guarantee that the token is computationally indistinguishable from a real token.
- Game_5: is the same as Game_4 except for the following. In step 2-b of OfflineQuery we replace K'_1 and K'_2 with $(K'_1, K'_2) \leftarrow \mathsf{SS.Share}(0^k, 2, 2)$. In step 3-$b$ we replace tk with $\mathsf{tk} \leftarrow \mathbf{Sim}_{\mathsf{DB}}(\mathcal{L}^{\mathsf{DB}}_{\mathsf{Q}}(\mathsf{EDB}_{\mathbf{L}}, \varphi))$. In step 5 we replace $\mathrm{ct}_1, \ldots, \mathrm{ct}_m$ with $\mathrm{ct}_1, \ldots, \mathrm{ct}_m \xleftarrow{\$} \{0,1\}^{|\mathbf{r}|}$ and in step 6-c we return $(\mathbf{r}_1, \ldots, \mathbf{r}_m)$. Since $\mathbf{Q}$ is the only corrupted party, K'_2 is indistinguishable from a real share. The pseudo-randomness of the ciphertexts guarantees that they are computationally indistinguishable from real ciphertexts and the Λ_{DB} security of Σ_{DB} guarantee that the token is computationally indistinguishable from a real token.

Finally, note that Game_5 is equivalent to an $\mathrm{IDEAL}_{\mathcal{Z},\mathcal{S}}(k)$ experiment.

□

Concrete leakage analysis. In our implementation (detailed in Section 8) we instantiate Σ_{DX} with a forward-secure variant of Pibase [17] and Σ_{DB} with the scheme that results from applying standard techniques from [22] (outlined in Appendices A and B) to the BIEX construction of [30]. We provide below a high-level description of the leakage profile of our registration system, with a more formal description in Appendix A. For each pattern that the system reveals, we provide some high-level intuition of what the disclosure implies

from a real-world perspective. We stress that all end-to-end encrypted solutions that are sub-linear reveal some leakage; even ORAM-based systems. For more on concrete leakage attacks we refer the reader to [8].

- *(global directory)* our concrete instantiation of Σ_{DX} has no init leakage. The query leakage is composed solely of the *query equality pattern* which reveals if and when the same serial number was been queried in the past. Note that this is only disclosed to the server and not to the two custodians. Similarly, the add leakage reveals if and when the county id was modified for some serial number. With both these patterns, a server can learn the frequency that a serial number is accessed or modified. Note that the server does not learn the value but only the frequency of the serial number.
- *(local databases)* our concrete instantiation of Σ_{DB} has no init leakage. The add leakage only reveals the size of the sub-EMMs which itself discloses the number of added records. Depending on the complexity of the query, the query leakage will be different. If the query is for a single keyword, then the server will learn the *query equality pattern* and the *response identity pattern*. Concretely, the former reveals if and when the same query has been made, while the latter reveals the identifiers of the matching records. If the query is a disjunction, then the query equality will reveal the query equality pattern on all the keywords that compose the query. It will also reveal the response identity pattern of the query and of a subset of keyword pairs that compose the query. Finally for a boolean query in CNF form, the server learns the same information mentioned above for the first disjunction, the query equality of all the keywords that compose the other disjunctions, and the response identity pattern on all pairs of keywords in the disjunctions. As a consequence, the server can learn the number of records that share a specific sets of keywords, the frequency they are accessed with, but not the queries or the records' content. Finally, we note that even though we instantiated the database encryption scheme so that it could support boolean queries, this level of expressivity may not be necessary in practice; at the very least not for all queries.

8 Empirical Evaluation

In this section, we describe and evaluate our prototype implementation of the encrypted registry system of Section 5. In particular, we evaluate: (1) the time it takes to add a record; (2) the time it takes to find the county id of a serial number; and (3) the time it takes to query a local database both when the official is online and offline; (4) the size of the global directory and of the local databases; and (5) the impact of our optimizations. First, we describe our implementation and testing environment.

8.1 Implementation

Our implementation is written in C++, Java, Javascript and Python. It is 3261 lines of code in total. In addition, it includes 451 lines for experimental testing and 71 lines to load and generate records, all calculated using CLOC [2]. The prototype has a client-server architecture. All communication between the parties uses the Node.js framework. The same framework is used to run child processes on the server and call the needed cryptographic libraries.

Testing environment. We conducted our experiments on Amazon Elastic Compute Cloud (EC2) [3] in the East region (Ohio). Given the distributed nature of our system, we set up our testing environment over the public network. We used three different types of EC2 instances: `t2.micro`, which has 1 virtual CPU and 1GB of RAM; `t3.micro` which has 2 virtual CPUs and 1GB of RAM; and `t3.xlarge`, which has 4 virtual CPUs and 16GB of RAM to which we associated 500GB of Elastic Block Store with generic SSDs for disk storage. We used a different instance depending on the computation and memory requirements of each role: `t2.micro` for the local official's backups, a `t3.micro` for the local officials, `t3.xlarge` for the custodians and server. All the instances are running a 64-bit Ubuntu Server 18.04 LTS.

Data generation. For each of our experiments involving the global directory, we randomly generated serial/id pairs based on the test case. The maximum number of pairs generated was 400 million in order to reflect the estimate of the number of firearms in the United States [1]).

For the experiments involving local databases, we created sample records, based on the test case, using the Python library `Faker`. Each record is a collection of field/value pairs. In particular, we use the following fields: *first name*, *last name*, *address*, *birth date*, *license number*, *model*, *make*, *caliber*, *serial number*, and *history*. In order to control selectivity for our query experiments (i.e., the number of matching records) we partially forge some records by choosing the values of some of their attributes.

Query generation. For our evaluation of the Query and the OfflineQuery protocols, we consider three scenarios: (1) high selectivity; (2) medium selectivity; and (3) low selectivity. This categorization helps us assess the time it takes to query a local database. We consider the following queries:

- *(low selectivity)*: `license:123456789`;
- *(medium selectivity)*: `address:987 Xyz St.  OR address:987 Xyz Street`;
- *(high selectivity)*: `last:Smith`.

For these test scenarios, the low selectivity query returns 1 record, the medium selectivity query returns 10 records, and the high selectivity query returns 100 records. These queries were selected to demonstrate the performance of the system but as mentioned previously, bulk queries are not permitted. Therefore a query that returns 10 or 100 records may not be a valid query, depending on the county.

8.2 Cryptographic primitives

Our encrypted registry makes use of several cryptographic primitives as building blocks. In the following, we discuss our instantiations along with the libraries used. We stress that that our protocols make black-box use of these primitives so the instantiations and libraries can be changed.

Multi-party computation. For all 2PCs, our prototype uses the `EMP toolkit` library [50]. In particular, we use `emp-sh2pc` which is an implementation of Yao's 2PC protocol [51] in the semi-honest setting. For our purposes, we wrote an EMP-compatible implementation of HMAC-SHA256 on variable size inputs based on an existing EMP-compatible implementation of SHA256 for fixed-size inputs [9]. For MPC, we used the the `JIFF` library [23].

Encrypted data structures. We used the `Clusion` library [40] to implement all the encrypted data structures; including the global directory and the encrypted multi-maps needed for the encrypted databases. In particular, we implemented the global directory with `Clusion`'s Pibas implementation [17]. The EMMs of the local database were implemented with `Clusion`'s BIEX-2Lev implementation [17, 30]. We made two changes to `Clusion`: (1) we changed the underlying PRFs from AES-CMAC to HMAC-SHA256; and (2) we replaced the use of AES in counter mode with HMAC-SHA256 in counter mode (i.e., we encrypt each bloc by XORing it with the output of HMAC-SHA256 on a counter). [3] These two modifications were needed so that our 2PC-based decryption of records in the Query and OfflineQuery protocols would be compatible with the Clusion-based encrypted structures stored, updated and queried at the server.

Unfortunately, `Clusion` only implements of the static variant of BIEX, whereas we need a dynamic variant. To handle this, we implemented a dynamic variant of BIEX using the approach outlined in Appendix B.

Secret sharing. Our protocol uses both threshold and 2-out-of-3 secret sharing. We instantiated the former with Shamir secret sharing [45] and the latter with XOR secret sharing.

Overview of our results. Here, we summarize the main takeaways from our empirical evaluation and provide a more detailed analysis in Section 8.2.

- *(add efficiency)* our experiments show that during an add operation, the time to add the serial/id pair to the global directory dominates the time needed to add the record to the encrypted database. For example, to add a batch of $10,000$ records, it takes $2,627$ seconds to add the corresponding $10,000$

[3] This construction was formally analyzed by Bellare et al. in [7].

serial/id pairs to the global directory, whereas it only takes 14.8 seconds to add the 10,000 records to the local official's encrypted database. In total, this is about 264 milliseconds per record.

- *(query efficiency)* our experiments show that querying the global directory to identify the county of a serial number takes less than 300 milliseconds. They also show that the time to query an official's encrypted database mainly depends on the selectivity of the query. Interestingly, the number of records, as well as the number of updates performed in the past, have limited impact on the query time. In particular the time is dominated by the 2PCs required to decrypt the response. As an example, it takes about 1 minute to query and retrieve 100 matching records. For offline queries, the protocol has similar behavior. The MPC needed to reconstruct and re-share the local official's secret key is negligible compared to the total time of offline queries which is dominated by the 2PCs required to decrypt the response.

- *(storage overhead)* our results show that the size of the global directory is linear in the number of pairs stored. Recall that the encrypted databases are composed of a BIEX-2Lev EMM and a set of encrypted records. The size of the EMM has quadratic behavior as a function of the number of records. Moreover, the size of the encrypted records is negligible compared to the size of the EMM. In particular, for 400 million pairs, the size of the global directory is 110GB. For 100,000 records, the size of the encrypted database's EMM is 1.7GB and the size of its encrypted records is less than 40MB.

8.3 Add Time

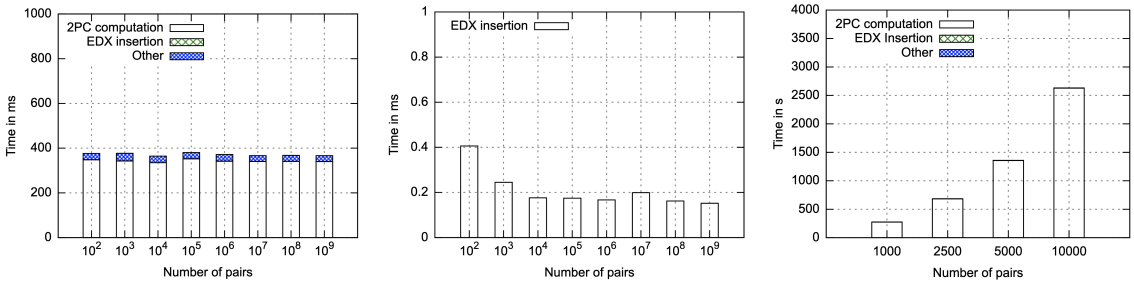
We evaluate the time of the Add protocol as a function of the number of records inserted in the system. For this, we evaluate each step of the protocol including: (1) the time to add a serial/id pair to the global directory as a function of its size; (2) the time to add a batch of serial/id pairs; and (3) the time to create a new EMM as a function of the number of records.

Adding pairs. The goal of this experiment is to assess how the time of adding a new serial/id pair behaves as a function of the size of the global directory, which includes the number of registered guns in the US. In Figure 5a, we vary the size from 100 up to 1 million pairs and then extrapolate the results to reach 1 billion. We used a logistic regression to extrapolate the results for when 1 million and 1 billion are already stored in the global directory. Our results demonstrate that adding a new pair is independent of the size of the global directory and takes 366 milliseconds. For completeness, we tested the time to add a new pair on a global directory containing 400 million pairs. The total time was 377 milliseconds, with 349 milliseconds spent on the 2PC. The time required for the 2PC computation dominates the other tasks. In Figure 5b, we show that the time to add the pair in the global directory takes less than 1 millisecond.

Adding batches of pairs. In this experiment, we measure the time it takes to add multiple pairs at once to the global directory. In particular, we want to know whether inserting multiple pairs affects the overall time of the protocol. In Figure 5c, our experiment shows that the amortized time to insert a single pair in a batch is around 269 milliseconds which is 27% more efficient than adding a single pair at a time. For this set of experiments, we used a batch size of 100 pairs. Increasing the batch size beyond this did not lead to better execution times mainly because of the increasing communication overhead incurred by the 2PC needed to accommodate for a higher circuit size. Like above, the 2PC computation dominates the execution time. In Figure 5d, we show the execution time without the 2PC. In particular, inserting 10,000 pairs into the global directory requires less than 0.5 milliseconds.

Setting up the encrypted database. In this experiment, we measure the time to encrypt the multi-map and the records. Figure 7a shows that the setup of the EMM dominates the overall execution time. In particular, setup takes around 220 seconds while the encryption of the records takes only 22 seconds.

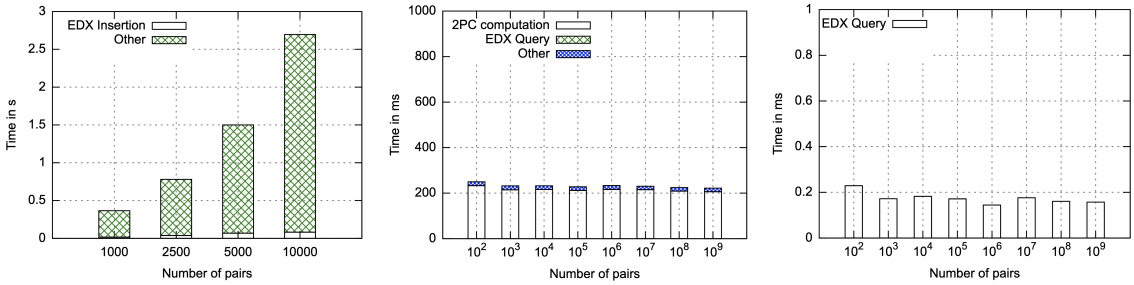
Total add time. The time to execute the Add protocol is the sum over the time do a batch update, the time to prepare and the time to encrypt the local database (which itself consists of the time prepare the EMM and encrypt the records). For clarity, we consider a concrete example in which the local official has an add rate of 10,000 records per week. The time to add 10,000 serial/id pairs is 2,627 seconds while the



(a) Global directory add time for a single pair.

(b) Global directory add time for a single pair (without 2PC).

(c) Global directory add time for batched pairs.



(d) Global directory add time for batched pairs (without 2PC).

(e) Global directory find time for a single county.

(f) Global directory find time for a single county (without 2PC).

Figure 5: Performance of operations with varying number of pairs in the global directory. In (a), (b), (e), (f), the values for when 10^8 and 10^9 pairs are in the global directory are extrapolated.

time to add a new batch of records to the encrypted database is either of the following (depending on the update number). If the update number is a power of 2, then the execution time is simply the time to create a new EMM along with the encrypted records which is 14.58 seconds. Otherwise, if the update number is not a power of 2, then the execution time is less than 220.42 seconds, which is the time to create an EMM for 100,000 records. [4] For $n = 10,000$ and $u = 53$, the largest EMM has a size less than 100,000. In summary, the Add protocol requires less than 48 minutes.

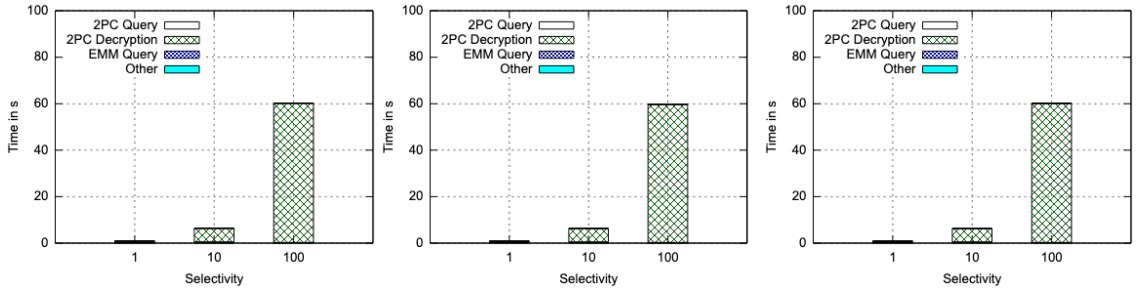
8.4 Find Time

We are interested in measuring the time it takes a querier to identify the county a given gun is registered in. As above, we vary the number of pairs in the global directory from 100 to 1 million and then extrapolate to 1 billion. We used a logistic regression to extrapolate the results for when 1 million and 1 billion are already stored in the global directory. In Figure 5e, we show that the time is independent of the size of the directory. In particular, it takes around 230 milliseconds to retrieve the county identifier. For completeness, we tested the time to query for a county ID a global encrypted directory containing 400 million pairs. The total time was 247 milliseconds, with 230 milliseconds spent on the 2PC. Similar to the Add protocol, the 2PC computation takes the most time. Figure 5f shows that the server requires less than 1 millisecond to retrieve the encrypted pair.

8.5 Query Time

In this experiment, we measure the time it takes for a querier to retrieve the matching records from a local database. There are two dimensions we varied in the experiment: (1) the selectivity of the query; and (2)

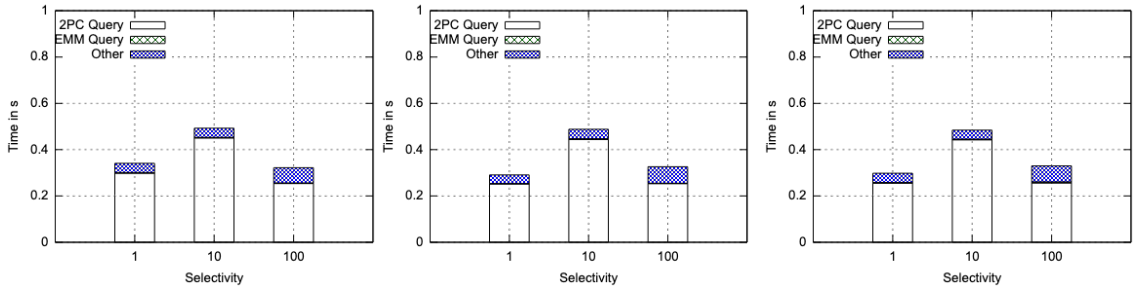
[4]Note that one can show that the smallest structure on the server has a size $O(u \cdot n/\log(u))$, where u is the number of updates and n the number of records, respectively.



(a) Query time with 1,000 records in local database.

(b) Query time with 10,000 records in local database.

(c) Query time with 100,000 records in local database.



(d) Query time with 1,000 records in local database (without 2PC de-cryption).

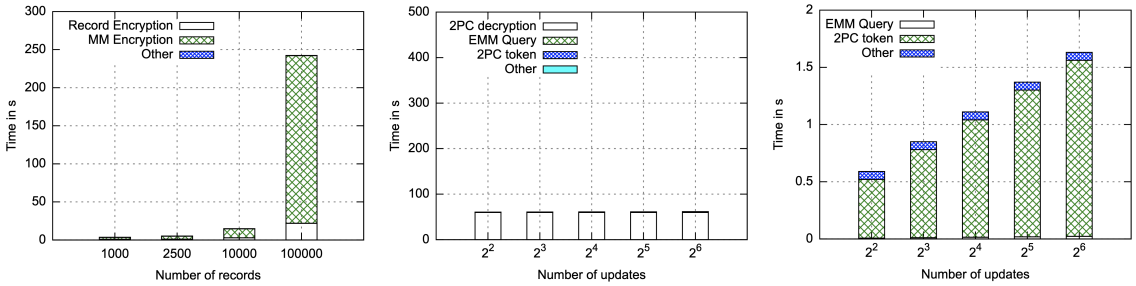
(e) Query time with 10,000 records in local database (without 2PC Decryption).

(f) Query time with 100,000 records in local database (without 2PC Decryption).

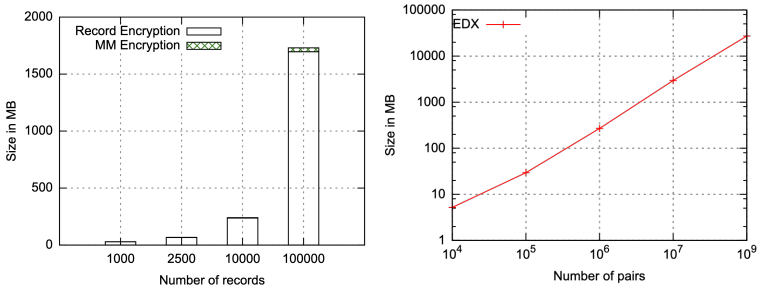
Figure 6: Local database query times based on the selectivity of the query with varying records stored. The x-axis represents low, medium, and high selectivity, which return 1, 10, 100 records respectively.

the number of records stored in the server. Figure 6 summarizes our results. As an example, retrieving 100 records from the database, requires about 1 minute. We noticed that the query time is independent of the number of records in the database. However, the selectivity of the query does impact query time; especially the time to do the decryption in the 2PC which represents 99.5% of the overall execution time, and sharply increases with more records to decrypt. Note that in our current prototype, each decryption requires a new 2PC. And since the decryptions occur sequentially, the query time is almost proportional to the selectivity of the query. In Figure 6, we also notice that the time to query the EMM on the server side is negligible compared to the time required to generate the token using 2PC. In particular, the former takes 3 milliseconds which accounts for 0.006% of the overall time, while the latter takes 260 milliseconds which accounts for 0.4% of the overall time. Note that the 2PC computation for selectivity 10 is slightly larger as we need to compute a token for a more complex query, refer to Section 8.1 for more details on our queries.

Query time with dynamic databases. We assessed the query and offline query times while dynamically expanding the database and, in particular, the EMM. Note that the main effects of using the dynamic BIEX-2Lev EMM are that: (1) we need to generate tokens that increase linearly as a function of the number of sub-EMMs; and (2) the query algorithm needs to query all the sub-EMMs. In Figure 7b and 7c, we provide a simulation based on the empirical numbers from the previous experiment. Specifically, we use the average time for querying an EMM and multiply it by the number of EMMs that exist after x updates have been made, which is $log_2(x)$. Recall that the number of sub-EMMs, as well as the token size, grows logarithmically as a function of the number of adds. In particular, our simulation demonstrates that dynamism has a little to no impact on the execution time of the query protocols. This was expected since the 2PC decryption step greatly dominates the other tasks.



(a) Local database setup time. (b) Simulated local database query time after updates (c) Simulated local database query time after updates (without 2PC decryption).



(d) Size of local database. (e) Size of the global directory.

Figure 7: Performance of various operations on a local database and the storage sizes of a local database and global directory. Record encryption refers to the amount of time taken to encrypt records. MM Encryption refers to the amount of time taken to setup the EMM used to make queries over the local database.

8.6 Offline Query Time

We show in Figure 8 (in Appendix 8.8) the overall time spent by the OfflineQuery protocol as a function of the selectivity, but also the number of records in the dataset. This protocol is similar to the Query protocol. The main differences are that the backups need to: (1) reconstruct the local official's key; and (2) generate new shares for the key. In particular, the time for reconstruction is around 5.68 seconds, while the time for generating the new shares is around 450 milliseconds.

8.7 Storage Overhead

In this experiment, we are interested in assessing the size of the encrypted databases (including its EMM and encrypted records) and the size of the global directory. Figure 7d summarizes our results. We observe that the size of the EMM dominates the overall size of the encrypted databases. The encrypted records only accounts for 0.2% of the overall storage. With respect to the global directory, we varied the number of pairs it holds from 1000 up to 100 million. The storage overhead is summarized in Figure 7e. We notice that the size of the global directory grows linearly as a function of the number of pairs. As an example, storing 100 million serial/id pairs requires 28GB.

8.8 Cost Analysis

We estimated the yearly cost of running the system in order to provide a sense of its financial feasibility. Our numbers are based on the AWS pricing calculator [4] and our own AWS usage data. We assume that the yearly cost would be paid upfront. Note that these costs are based on regular billing rates while government entities would be billed at a discounted rate. We only consider the servers that store the global directory

and the encrypted databases and the custodians since a standard laptop is sufficient for local officials and queriers. The base cost reflects the price of running a single AWS instance with no additional storage and no data transfer. Depending on a server's role, it incurs different storage and data transfer costs, which we describe. For the custodians and the global directory server, we assume 400 million adds and finds are executed. This captures the estimated number of gun ownership in the US that was previously mentioned. For the encrypted database servers, which would store the databases of each county in a state, our estimate is based on the population of California since it is the largest state with around 40 million people as of 2019 [48].

Base costs. As mentioned, we used a `t3.xlarge` AWS instance for our empirical evaluation, which has a yearly cost of $853.22 when considering no additional storage and upfront payment. If a more powerful instance were to be used, such as the `m5.metal`, which has 96 vCPUs and 384GB of RAM, the yearly cost with no additional storage and upfront payment would be $23,739.60.

Global directory server. Given that an EDX of 400 million pairs is around 110GB, the directory server would require at least this much storage. An attachment of 200GB of SSD EBS storage would cost $461.28 a year. The data transfer cost is assumed to be negligible since the server receives tokens that are either 96 or 64 bytes. If 400 million updates were made in a single year, the amount of data transferred would still be less than 1 TB, which amounts to less than $10.

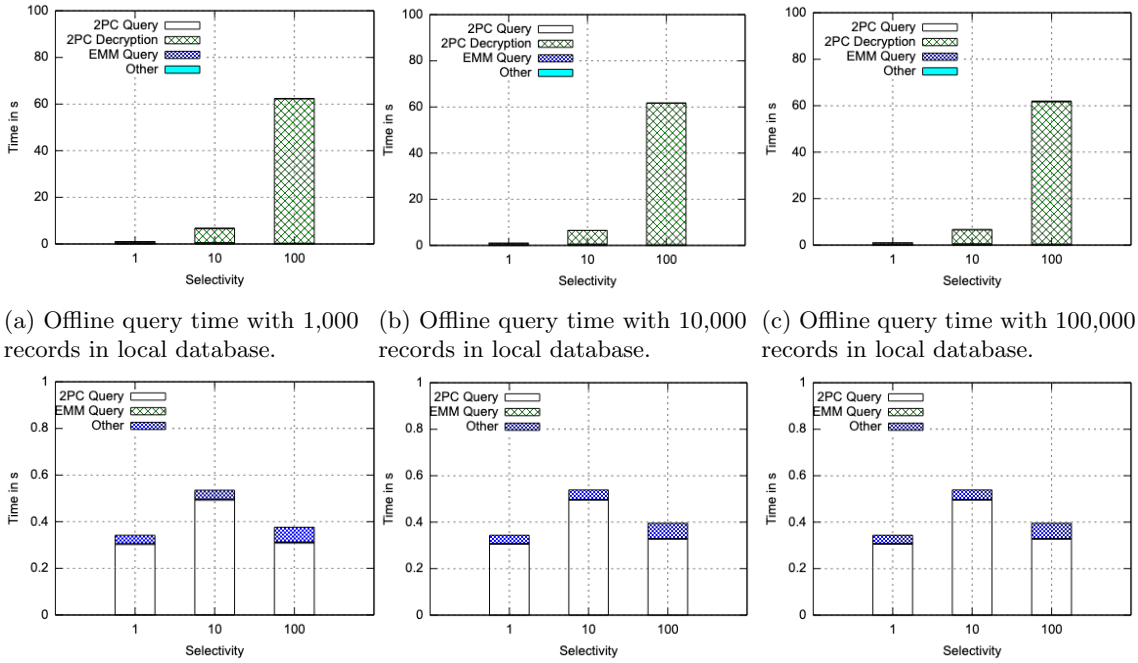
Custodians. The custodians have relatively low storage costs but high data transfer costs due to the 2PCs they have to execute as part of the add and find operations. We observed that around 0.5 GB of data was transferred between the two custodians for the 2PCs of 25 adds and the 2PCs of 25 finds. Based on this number, at a price of $10 per TB, it would cost roughly $80,000 to execute the 2PCS of 400 million adds and 400 million finds.

Encrypted database servers. Given that an EMM that holds 100,000 records is around 1.75 GB and that the encrypted records are 384 bytes each, if the state of California were to store a record for every person, it would need around 680 GB to store the EMMs and 15.36 GB to store the encrypted records. An attachment 700 GB of SSD EBS storage would cost $1586.28 a year.

The data transfer cost for a state-level encrypted database server is negligible since the server only receives data in the form of query tokens and encrypted records. Each encrypted record is 384 bytes and a query token for the high selectivity query used is 380 bytes. If each individual in California were to be queried in a year, this would result in less than 40 GB of data transferred, which would cost less than $10. These estimates show an approximate range of costs. This does not imply that the various government entities would choose to use these specific machines or even a cloud provider such as AWS, but rather it demonstrates that the financial cost of operating our system would not be a significant barrier to deployment.

9 Responsible Design

We believe that gun control laws, stricter licensing, and firearm registries have a positive affect on public health. We acknowledge, however, that introducing a protocol designed specifically for data collection by law enforcement has the potential for abuse and we recognize and strongly oppose existing forms of non-consensual data collection by law enforcement. We are also aware that giving law enforcement access to data on citizens can enable the abuse of that data beyond the original intent of its collection. Although driver license and license plate databases are—like gun registries—intended to increase public safety, unfettered access to this data by various law enforcement agencies has lead to abuse. In 2019, it was reported that US Immigration and Customs Enforcement officers were given access to license plate databases which were used to profile "foreign-looking" drivers, record their license plates, look up personal information, and target them for deportation [38]. Furthermore, it is known that the Federal Bureau of Investigation has access to driver license photos in multiple states and that it uses the data with facial recognition algorithms to detect suspected criminals [21]. In registering for a driver's license, citizens do not explicitly consent to these uses of their data and photos. This is exacerbated by the fact that this data can misidentify them as criminals



(a) Offline query time with 1,000 records in local database.

(b) Offline query time with 10,000 records in local database.

(c) Offline query time with 100,000 records in local database.

(d) Offline query time with 1,000 records in local database (without 2PC decryption).

(e) Offline query time with 10,000 records in local database (without 2PC decryption).

(f) Offline query time with 100,000 records in local database (without 2PC decryption).

Figure 8: Local database offline query times based on the selectivity of the query with varying records stored. The x-axis represents low, medium, and high selectivity, which return 1, 10, 100 records respectively.

which is particularly harmful to Black license holders, for whom facial recognition algorithms have a lower accuracy rate.

In designing this system, we kept in mind how the data it stores could be misused; particularly against marginalized groups. We emphasize that people should consent to how their registration data is used and by whom. The draft legislation explicitly states that attempts to use the registry to collect large amounts of data is prohibited and will be reported. We support this rate limiting feature (and instantiate it via our 2PC-based decryption) and urge that that access to the system (e.g., by officials, sellers, police agencies, government agencies) be clearly and transparently defined prior to deployment.

We also understand that once the technical infrastructure is in place, firearm data could be be replaced and used for other purposes beyond our intentions. While we cannot control how this protocol is deployed, we *strongly* advocate for policies and regulations that limit personal data collection by Law Enforcement. While new developments in cryptography can enhance privacy, the use of privacy-enhancing technologies can also be used as cover to request access to more data. Policies and laws that specifically outline what kind of data can and cannot be collected by law enforcement *before* a system such as ours is built and deployed would help curb the potential for abuse.

10 Conclusion

In this work, we designed, implemented and evaluated an end-to-end encrypted national gun registry. In designing the system, we were in conversation with the Senate office that drafted the legislation and followed their requirements explicitly. We implemented the protocol and evaluated at the scale it would have to run if it were deployed in the US. Our empirical results confirmed that the design is not only possible but practical.

The purpose of this work is to demonstrate the feasibility of such a system, so our design can be improved by future work in several respects. As designed (and implemented), the system is secure in the standalone setting. Future work could improve the protocol to be secure in the universal composability model [14]. As mentioned in Section 5, our protocol makes black-box use of its underlying primitives. As such, different instantiations of these building blocks yield different tradeoffs between efficiency and leakage. Future work could explore how different instantiations (e.g., using techniques from [34]) improve the leakage profile of the system at what cost to performance.

Acknowledgments

We would like the thank Senator Wyden' s office for motivating this work, sharing the draft of their legislation, and providing valuable feedback throughout the project. We would also like to thank Samuel Boger for his implementation assistance in the initial stages of this project.

References

[1] Aaron Karp. Estimating Global Civilian-Held Firearms Numbers. Technical report, Small Arms Survey, Geneva, Switzerland, June 2018.

[2] Al Danial. cloc, May 2020.

[3] Amazon Web Services. Amazon Elastic Compute Cloud Documentation, 2020.

[4] Amazon Web Services. AWS Pricing Calculator, 2020.

[5] Ghous Amjad, Seny Kamara, and Tarik Moataz. Breach-resistant structured encryption. In *Proceedings on Privacy Enhancing Technologies (Po/PETS '19)*, 2019.

[6] Johes Bater, Gregory Elliott, Craig Eggen, Satyender Goel, Abel Kho, and Jennie Rogers. SMCQL: secure querying for federated databases. *Proceedings of the VLDB Endowment*, 10(6):673–684, February 2017.

[7] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *Symposium on Foundations of Computer Science (FOCS '97)*, pages 394–405. IEEE Computer Society, 1997.

[8] Laura Blackstone, Seny Kamara, and Tarik Moataz. Revisiting leakage abuse attacks. In *Network and Distributed System Security Symposium (NDSS '20)*, 2020.

[9] Bolt Labs. emp-sh2pc, January 2020.

[10] R. Bost. Sophos - forward secure searchable encryption. In *ACM Conference on Computer and Communications Security (CCS '16)*, 20016.

[11] R. Bost, B. Minaud, and O. Ohrimenko. Forward and backward private searchable encryption from constrained cryptographic primitives. In *ACM Conference on Computer and Communications Security (CCS '17)*, 2017.

[12] Bureau of Alcohol, Tobacco, Firearms and Explosives. ATF Form 4473 - Firearms Transaction Record Revisions, August 2020.

[13] R. Canetti. Security and composition of multi-party cryptographic protocols. *Journal of Cryptology*, 13(1), 2000.

[14] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *IEEE 42^{nd} Annual Symposium on the Foundations of Computer Science (FOCS 2001)*, pages 111–126. IEEE, 2001.

[15] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner. Highly-scalable searchable symmetric encryption with support for boolean queries. In *Advances in Cryptology - CRYPTO '13*. Springer, 2013.

[16] D. Cash and S. Tessaro. The locality of searchable symmetric encryption. In *Advances in Cryptology - EUROCRYPT 2014*, 2014.

[17] David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel Rosu, and Michael Steiner. Dynamic searchable encryption in very-large databases: Data structures and implementation. In *Network and Distributed System Security Symposium (NDSS '14)*, 2014.

[18] Giffords Law Center. Giffords law center to prevent gun violence, 2018.

[19] Giffords Law Center. Registration, 2018.

[20] M. Chase and S. Kamara. Structured encryption and controlled disclosure. In *Advances in Cryptology - ASIACRYPT '10*, volume 6477 of *Lecture Notes in Computer Science*, pages 577–594. Springer, 2010.

[21] Clare Garvie, Alvaro M. Bedoya, and Jonathan Frankle. The Perpetual Line-Up. Technical report, Center on Privacy & Technology at Georgetwon Law, Georgetown, Washington D.C., October 2016.

[22] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. In *ACM Conference on Computer and Communications Security (CCS '06)*, pages 79–88. ACM, 2006.

[23] Kinan Dak Albab, Rawane Issa, Andrei Lapets, Peter Flockhart, Lucy Qin, and Ira Globus-Harris. Tutorial: Deploying Secure Multi-Party Computation on the Web Using JIFF. In *2019 IEEE Cybersecurity Development (SecDev)*, pages 3–3, Tysons Corner, VA, USA, September 2019. IEEE.

[24] Mohammad Etemad, Alptekin Küpçü, Charalampos Papamanthou, and David Evans. Efficient dynamic searchable encryption with forward privacy. *PoPETs*, 2018(1):5–20, 2018.

[25] David Evans, Vladimir Kolesnikov, and Mike Rosulek. *A pragmatic introduction to secure multi-party computation*, volume 2. Foundations and Trends in Privacy and Security, 2017.

[26] Dan Friedman. The ATFs Nonsensical Non-Searchable Gun Databases, Explained. *The Trace*, August 2016.

[27] S. Garg, P. Mohassel, and C. Papamanthou. TWORAM: efficient oblivious RAM in two rounds with applications to searchable encryption. In *Advances in Cryptology - CRYPTO 2016*, pages 563–592, 2016.

[28] O. Goldreich, S. Micali, and A. Wigderson. How to play ANY mental game. In *ACM Symposium on the Theory of Computation (STOC '87)*, pages 218–229. ACM, 1987.

[29] S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner. Outsourced symmetric private information retrieval. In *ACM Conference on Computer and Communications Security (CCS '13)*, pages 875–888, 2013.

[30] S. Kamara and T. Moataz. Boolean searchable symmetric encryption with worst-case sub-linear complexity. In *Advances in Cryptology - EUROCRYPT '17*, 2017.

[31] S. Kamara and T. Moataz. Computationally volume-hiding structured encryption. In *Advances in Cryptology - Eurocrypt' 19*, 2019.

[32] S. Kamara and C. Papamanthou. Parallel and dynamic searchable symmetric encryption. In *Financial Cryptography and Data Security (FC '13)*, 2013.

[33] S. Kamara, C. Papamanthou, and T. Roeder. Dynamic searchable symmetric encryption. In *ACM Conference on Computer and Communications Security (CCS '12)*. ACM Press, 2012.

[34] Seny Kamara, Tarik Moataz, and Olya Ohrimenko. Structured encryption and leakae suppression. In *Advances in Cryptology - CRYPTO '18*, 2018.

[35] J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. Chapman & Hall/CRC, 2008.

[36] Jeanne Laskas. Inside the Federal Bureau Of Way Too Many Guns. *GQ*, August 2016.

[37] James McClure. Firearms Owners' Protection Act, May 1986.

[38] McKenzie Funk. How ICE Picks Its Targets in the Surveillance Age. *New York Times Magazine*, October 2019.

[39] Metropolitan Police Department. Application for Firearms Registration Certificate (PD-219), August 2020.

[40] T. Moataz and S. Kamara. Clusion. https://github.com/encryptedsystems/Clusion.

[41] New York State Police. Pistol Revolver License Recertification.

[42] V. Pappas, F. Krell, B. Vo, V. Kolesnikov, T. Malkin, S.-G. Choi, W. George, A. Keromytis, and S. Bellovin. Blind seer: A scalable private dbms. In *Security and Privacy (SP), 2014 IEEE Symposium on*, pages 359–374. IEEE, 2014.

[43] S. Patel, G. Persiano, and K. Yeo. Symmetric searchable encryption with sharing and unsharing. Cryptology ePrint Archive, Report 2017/973, 2017. http://eprint.iacr.org/2017/973.

[44] Sarvar Patel, Giuseppe Persiano, Kevin Yeo, and Moti Yung. Mitigating leakage in secure cloud-hosted data structures: Volume-hiding for multi-maps via hashing. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 79–93. ACM, 2019.

[45] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[46] D. Song, D. Wagner, and A. Perrig. Practical techniques for searching on encrypted data. In *IEEE Symposium on Research in Security and Privacy*, pages 44–55. IEEE Computer Society, 2000.

[47] E. Stefanov, C. Papamanthou, and E. Shi. Practical dynamic searchable encryption with small leakage. In *Network and Distributed System Security Symposium (NDSS '14)*, 2014.

[48] United States Census Bureau. 2019 National and State Population Estimates, December 2019.

[49] Nikolaj Volgushev, Malte Schwarzkopf, Ben Getchell, Mayank Varia, Andrei Lapets, and Azer Bestavros. Conclave: secure multi-party computation on big data. In *Proceedings of the Fourteenth EuroSys Conference 2019 - EuroSys '19*, pages 1–18, Dresden, Germany, 2019. ACM Press.

[50] Xiao Wang, Alex J. Malozemoff, and Jonathan Katz. EMP-toolkit: Efficient MultiParty computation toolkit, 2016.

[51] A. Yao. Protocols for secure computations. In *IEEE Symposium on Foundations of Computer Science (FOCS '82)*, pages 160–164. IEEE Computer Society, 1982.

[52] A. Yao. How to generate and exchange secrets. In *IEEE Symposium on Foundations of Computer Science (FOCS '86)*, pages 162–167. IEEE Computer Society, 1986.

A Encrypted Document Databases

We recall the standard approach to designing an encrypted document database from a symmetric key encryption $\mathsf{SKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and a multi-map encryption scheme $\Sigma_{\mathsf{MM}} = (\mathsf{Init}, \mathsf{Token}, \mathsf{Query}, \mathsf{AddToken}, \mathsf{Add}, \mathsf{Resolve})$. This is equivalent to the index-based approach to designing SSE schemes from Curtmola et al. [22]. Let Index be an algorithm that takes as input a record and outputs label/tuple pairs and $\Sigma_{\mathsf{DB}} = (\mathsf{Init}, \mathsf{Token}, \mathsf{Query}, \mathsf{AddToken}, \mathsf{Add}, \mathsf{Resolve})$ be a (document) database encryption scheme that works as follows:

- $\mathsf{Init}(1^k)$: takes as input a security parameter 1^k and computes $(K_1, \mathsf{EMM}) \leftarrow \Sigma_{\mathsf{MM}}.\mathsf{Init}(1^k)$ and $K_2 \leftarrow \mathsf{SKE.Gen}(1^k)$. It sets $K = (K_1, K_2)$ and $\mathsf{EDB} = \mathsf{EMM}$ and outputs (K, EDB).
- $\mathsf{Token}(K, q)$: takes as input a key K and a query q. It parses K as (K_1, K_2) and outputs $\mathsf{tk} \leftarrow \Sigma_{\mathsf{MM}}.\mathsf{Token}(K_1, q)$.
- $\mathsf{Query}(\mathsf{EDB}, \mathsf{tk})$: takes as input an encrypted database EDB and a token tk. It parses EDB as $(\mathsf{EMM}, \mathrm{ct}_1, \ldots, \mathrm{ct}_n)$, computes $I \leftarrow \Sigma_{\mathsf{MM}}.\mathsf{Query}(\mathsf{EMM}, \mathsf{tk})$ and outputs $\{\mathrm{ct}_i\}_{i \in I}$.
- $\mathsf{AddToken}(K, \mathbf{r})$: takes as input a secret key K and a record $\mathbf{r}$. It parses K as (K_1, K_2) and computes $\{(\ell_1, \mathbf{v}_1), \ldots, (\ell_m, \mathbf{v}_m)\} \leftarrow \mathsf{Index}(\mathbf{r})$. Then, for all $1 \leq i \leq m$, it computes $\mathsf{atk}_i \leftarrow \Sigma_{\mathsf{MM}}.\mathsf{AddToken}(K_1, (\ell_i, \mathbf{v}_i))$. Finally, it outputs $\mathsf{atk} = (\mathsf{atk}_1, \ldots, \mathsf{atk}_m, \mathrm{ct})$, where $\mathrm{ct} \leftarrow \mathsf{SKE.Enc}(K_2, \mathbf{r})$.
- $\mathsf{Add}(\mathsf{EDB}, \mathsf{atk})$: takes as input an encrypted database EDB and an add token atk. It parses EDB as $(\mathsf{EMM}_0, \mathrm{ct}_1, \ldots, \mathrm{ct}_n)$ and atk as $(\mathsf{atk}_1, \ldots, \mathsf{atk}_m, \mathrm{ct})$. For all $1 \leq i \leq m$, it computes $\mathsf{EMM}_i \leftarrow \Sigma_{\mathsf{MM}}.\mathsf{Add}(\mathsf{EDB}_{i-1}, \mathsf{atk}_i)$. It outputs $\mathsf{EDB} = (\mathsf{EMM}_m, \mathrm{ct}_1, \ldots, \mathrm{ct}_n, \mathrm{ct})$.
- $\mathsf{Resolve}(K, \mathbf{ct})$: takes as input a secret key K and a set of ciphertexts $\mathbf{ct}$. It parses K as (K_1, K_2) and $\mathbf{ct}$ as $(\mathrm{ct}_1, \ldots, \mathrm{ct}_t)$. For all $1 \leq i \leq t$, it computes $\mathbf{r}_i \leftarrow \mathsf{SKE.Dec}(K_2, \mathrm{ct}_i)$.

Concrete instantiations. In this work, we instantiate Σ_{MM} with the BIEX-2Lev construction of [30]. The indexing algorithm we use takes a record $\mathbf{r}$ as input and, for every field/value pair $f{:}v$ in $\mathbf{r}$, outputs a label/value pair $(f \| v, \mathbf{r}.\mathsf{id})$. To search for all records with $f = v$, it then suffices to query for $q = f \| v$.

We now describe the leakage profile of BIEX-2Lev. Let

$$\varphi = \Big(\big(w_{1,1} \vee \cdots \vee w_{1,q}\big) \wedge \cdots \wedge \big(w_{\ell,1} \vee \cdots \vee w_{\ell,q}\big) \Big)$$

where $w_{i,j} \stackrel{\text{def}}{=} f_{i,j} \| v_{i,j}$. Then the query leakage of BIEX-2Lev is

$$\begin{aligned}
&\mathcal{L}^{\mathsf{biex}}_{\mathsf{Q}}(\mathsf{DB}, \varphi) = \\
&\Bigg(\Big(\mathsf{qeq}_g(w_{1,i}), \mathsf{trsize}(\mathsf{LMM}_{1,i}), \mathsf{qeq}_d(w_{1,i}), \\
&\quad \mathsf{qeq}_{1,i}(w_{1,i+1}), \ldots, \mathsf{qeq}_{1,i}(w_{1,q}), \mathsf{tag}_{1,i}(w_{1,1}, \ldots, w_{1,q}) \Big)_{1 \leq i \leq q-1}, \\
&\quad \Big(\mathsf{qeq}_{1,1}(w_{j,i}), \ldots, \mathsf{qeq}_{1,q}(w_{j,i}), \\
&\qquad \mathsf{tag}_{j,i}(w_{1,i}, \cdots, w_{1,q}) \Big)_{2 \leq j \leq \ell, 1 \leq i \leq q} \Bigg)
\end{aligned}$$

where $\mathsf{tag}_{i,j}(w_1, \ldots, w_q) = (f_{i,j}(w_1), \ldots, f_{i,j}(w_1))$ with $f_{i,j}$ a random function, $\mathsf{trsize}(\mathsf{MM})$ is a stateful leakage pattern that reveals the size of the multi-map MM, and $\mathsf{qeq}_X(w)$ is a stateful leakage pattern that reveals if and when w was queried in the past. Note that qeq_X and qeq_Y for $X \neq Y$ are not correlated in the sense that given $\mathsf{qeq}_X(w)$ and $\mathsf{qeq}_Y(w)$ one cannot tell that the leakages are for the same w. The leakage profile of BIEX-2Lev is complex and can be difficult to understand but we note that, currently, BIEX is the boolean scheme with the smallest leakage profile. To achieve a better leakage profile, one could use the leakage suppression techniques of [34].

B Semi-Dynamic Encrypted Multi-Maps

As described in Section 5, our protocol relies on a semi-dynamic database encryption scheme. Using the approach described in Appendix A one can construct such a scheme from a semi-dynamic multi-map encryption scheme and symmetric key encryption. Since, in our setting, we want support for boolean queries, we need the multi-map encryption scheme to also support boolean queries. As far as we know, there are three sub-linear boolean multi-map encryption schemes: the scheme that underlies the Blind Seer construction [42], the boolean variant of OXT [15] and the BIEX construction of [30]. The latter is the most efficient but is only static so we describe here how to make it semi-dynamic using a technique described in [22]. More precisely, given a static multi-map encryption scheme $\Sigma_{\mathsf{MM}} = (\mathsf{Setup}, \mathsf{Token}, \mathsf{Query})$, we define a new semi-dynamic scheme $\Delta_{\mathsf{MM}} = (\mathsf{Init}, \mathsf{Token}, \mathsf{Query}, \mathsf{Add})$ that works as follows:

- $\mathsf{Init}(1^k)$: takes as input a security parameter 1^k, sets $K = \perp$, $st = 1$ and $\mathsf{EMM} = \perp$ and outputs (K, st, EMM).
- $\mathsf{Add}\big((K, st, \{(\ell_i, \mathbf{v}_i)_i\}), \mathsf{EMM}\big)$: is a two-party protocol between the client and server where the client inputs a key K, a state st and a set of label/tuple pairs $\{(\ell_1, \mathbf{v}_1), \ldots, (\ell_t, \mathbf{v}_t)\}$ and the server inputs EMM. The client first creates a new multi-map MM that stores the pairs $\{(\ell_1, \mathbf{v}_1), \ldots, (\ell_t, \mathbf{v}_t)\}$. It then parses st into some positive integer i. If i a power of 2, it computes $(K_i, \mathsf{EMM}_i) \leftarrow \mathsf{Setup}(1^k, \mathsf{MM})$, and sends EMM_i to the server who sets $\mathsf{EMM} := (\mathsf{EMM}, \mathsf{EMM}_i)$. The client then sets $K := (K, K_i)$. On the other hand, if i is not a power of 2, the client downloads the smallest EMM at the server, decrypts it to recover $\mathsf{MM}^\star$ and computes $(K_i, \mathsf{EMM}_i) \leftarrow \mathsf{Setup}(1^k, \mathsf{MM}_i)$, where MM_i is a multi-map that holds the pairs in both $\mathsf{MM}^\star$ and MM. It then sends EMM_i to the server who sets $\mathsf{EMM} := (\mathsf{EMM}, \mathsf{EMM}_i)$. The client then updates $K := (K, K_i)$ and increments st by 1.
- $\mathsf{Token}(K, q)$: takes as input the secret key K and a query q and uses each key in $K = (K_1, \ldots, K_m)$ to generate a token $\mathsf{tk}_i \leftarrow \mathsf{Token}(K_i, \varphi)$ for each EMM stored on the server. It outputs $\mathsf{tk} = (\mathsf{tk}_1, \ldots, \mathsf{tk}_m)$.
- $\mathsf{Query}(\mathsf{EMM}, \mathsf{tk})$: takes as input $\mathsf{EMM} = (\mathsf{EMM}_1, \ldots, \mathsf{EMM}_m)$ and a token $\mathsf{tk} = (\mathsf{tk}_1, \ldots, \mathsf{tk}_m)$. For all $1 \leq i \leq m$, it computes $\mathbf{R}_i \leftarrow \mathsf{Query}(\mathsf{EDB}_i, \mathsf{tk}_i)$ and outputs $(\mathbf{R}_1, \ldots, \mathbf{R}_m)$.

The resulting semi-dynamic scheme Δ_{MM} has $O(\mathbf{time}(\mathsf{Q}) \cdot \log u)$ query time, where $\mathbf{time}(\mathsf{Q})$ is the complexity of $\Sigma_{\mathsf{MM}}.\mathsf{Query}$ and u is the number of add operations. It has the same asymptotic storage as Σ_{MM}. It has $O(\log u)$ token size and Add has communication complexity $O(u/\log u)$. The leakage profile of Δ_{MM} is the same as Σ_{MM} with the addition of add leakage $\mathcal{L}_{\mathsf{A}}$ which reveals the size of $\mathbf{R}$.