USENIX

THE ADVANCED COMPUTING
SYSTEMS ASSOCIATION

PolySys: an Algebraic Leakage Attack Engine

Zachary Espiritu, *MongoDB Research;* Seny Kamara, *Brown University and MongoDB Research;* Tarik Moataz, *MongoDB Research;* Andrew Park, *Carnegie Mellon University and MongoDB Research*

https://www.usenix.org/conference/usenixsecurity25/presentation/espiritu

This paper is included in the Proceedings of the
34th USENIX Security Symposium.

August 13–15, 2025 • Seattle, WA, USA

978-1-939133-52-6

Open access to the Proceedings of the
34th USENIX Security Symposium is sponsored by USENIX.

PolySys: an Algebraic Leakage Attack Engine

Zachary Espiritu
MongoDB Research

Seny Kamara
Brown University &
MongoDB Research

Tarik Moataz
MongoDB Research

Andrew Park
Carnegie Mellon University &
MongoDB Research

Abstract

In this work, we propose a novel framework called PolySys for modeling and designing leakage attacks as constraint-solving algorithms over polynomial systems. PolySys formalizes the design of attacks using invertible encodings, structural and leakage equations, and efficient constraint-solving algorithms including SAT and constraint solvers. It is capable of modeling resolution, known-data, and inference attacks for common leakage patterns.

To demonstrate the practicality of our framework, we implement a `PolySys` attack engine in Python and apply it to state-of-the-art query recovery, data resolution, and query inference attacks on point and range multi-maps. Our results show that `PolySys` outperforms all existing attacks under identical assumptions, achieving up to 60× higher recovery rates in some scenarios. While scalability remains a challenge for larger datasets, `PolySys` represents a promising step toward a general-purpose framework for designing leakage attacks. We believe future work can further enhance its efficiency to scale to larger and more complex workloads.

1 Introduction

Encrypted search algorithms (ESA) are cryptographic primitives that allow one to search over end-to-end encrypted data. ESAs can be designed using a variety of cryptographic primitives including property-preserving encryption (PPE) [5, 16], oblivious RAM (ORAM) [48], secure multi-marty computation (MPC) [47, 101], fully-homomorphic encryption (FHE) [44] and structured encryption (STE) [29].

Leakage regimes. Any ESA with sub-linear search time leaks some information about the data and/or queries. This includes ESAs based on ORAM, STE, PPE and sub-linear MPC. Following [10], sub-linear ESAs can be categorized into the following five leakage regimes:

- *high-leakage* solutions reveal non-trivial information about the secrets from the encrypted data alone;
- *mid-leakage* solutions reveal minimal information about the secrets (e.g., size) when only the encrypted data is available, but can reveal non-trivial information if query and update transcripts are available [26,28,31,34,38,46];
- *low-leakage* solutions reveal minimal information about the secrets given both the encrypted data and any query or update transcripts [8, 10, 43, 66];
- *zero-leakage* solutions hide everything about the secrets beyond the size of the data structure [66] and possibly the maximum response size;
- *subliminal* solutions provide zero-leakage security but also hide metadata such as the number and timing of queries, and even whether any occurred at all [10].

While the security of subliminal and zero-leakage solutions is relatively clear, it is more challenging to evaluate the security of high-, mid-, and low-leakage solutions. To address this, two complementary approaches have been pursued. The first and most recent approach is the development of *leakage analysis frameworks*, i.e., theoretical frameworks that allow one to formally analyze a leakage profile. One such example is the Bayesian Leakage Analysis (BLA) framework of Espiritu, Kamara, and Moataz [36], which can be used to analyze the resistance of a leakage profile to various statistical attacks. Another example is the Quantitative Information Flow (QIF) framework [30] which was adapted to the ESA setting by Jurado and Smith to analyze the resistance of deterministic and order-preserving encryption to certain attacks [58]. We refer the reader to Section 2 of [36] for additional examples and comparisons of leakage analysis frameworks.

Leakage attacks. The second approach is the design of leakage attacks, which are cryptanalytic algorithms aimed at extracting as much information as possible about the secrets from the leakage and, potentially, some auxiliary information. There are many different kinds of leakage attacks, each with distinct goals, assumptions and adversarial models, but they can be broadly categorized as follows:

- *inference attacks* take as input the leakage from an execution and an auxiliary probability distribution over the secrets and attempt to recover information about the secret. If, instead of an auxiliary distribution, the attack takes an auxiliary dataset, we refer to the attack as an *empirical inference attack*;
- *known-data attacks* take as input leakage and knowledge of some fraction of the data and attempt to recover information about the secret;
- *resolution attacks* take as input the leakage and return information about the secrets without requiring additional auxiliary information.

Note that inference attacks are the most general kind of attack since they can capture known-data attacks by setting the auxiliary information to known data and can capture resolution attacks by setting the auxiliary information to nothing, or to the uniform distribution. Leakage cryptanalysis offers insights into the practical security of a leakage profile but a key limitation of this line of work is that attacks must be custom-designed, which is labor-intensive. Consequently, many leakage profiles have not yet been thoroughly cryptanalyzed.

Attack engines. To address this limitation, [36] proposed the notion of *leakage attack engines*; that is, general-purpose cryptanalytic frameworks that can target a broad range of leakage profiles. To demonstrate the feasibility of this concept, they also implemented an attack engine called `Bayle` built on the BLA framework. Cryptanalyzing a leakage profile with `Bayle` requires describing the profile as a Bayesian network which is a class of probabilistic graphical models commonly used in statistics and machine learning. A statistical inference algorithm is then applied to recover information about the secrets. The authors observe that the Bayesian network representations of most leakage profiles exhibit a similar structure, which they call *hidden function networks* (HFNs), and they present a (relatively) optimized statistical estimation algorithm for i.i.d. HFNs.

1.1 Our Contributions

In this work, we continue the research initiated in [36] on leakage attack engines by proposing a new engine called `PolySys`, which is based on an alternative formalization of leakage attacks. We make the following contributions.

The PolySys framework. We show how to model leakage attacks against a leakage profile Λ as constraint solving problems and, specifically, as polynomial systems over $\mathbb{F}_2$. At a high level, our framework consists of the following steps,

- (*secret modeling*) define an efficiently invertible encoding $\mathsf{enc} : \mathbb{S} \to \overline{\mathbb{S}}$, that maps secrets in $\mathbb{S}$ to a subspace $\overline{\mathbb{S}}$ of $\mathbb{F}_2^{n\times m}$, for some $n,m \geq 1$. For example, for data recovery attacks, the secret space $\mathbb{S}$ is the set of all possible data structures under consideration and the $\mathbb{F}_2$-matrices generated by enc is their algebraic representation.
- (*structural equations*) if $\overline{\mathbb{S}} \subset \mathbb{F}_2^{n\times m}$, then define an equation f_0 over $\mathbb{F}_2$ that characterizes $\overline{\mathbb{S}}$; in other words, such that $\overline{\mathbb{S}}$ is the set of solutions to f_0;[1]
- (*leakage modeling*) define a leakage function $\mathsf{leak} : \mathbb{S} \to \overline{\mathbb{K}}$ that maps secrets in $\mathbb{S}$ to a subspace $\overline{\mathbb{K}}$ of $\mathbb{F}_2^{n\times m}$, for some $n,m \geq 1$.
- (*leakage equations*) given observed leakage $\boldsymbol{\ell} \in \overline{\mathbb{K}}$, use the relationships implicitly defined by enc and leak to generate a system $\mathsf{sys}(f_0,\ldots,f_k)$ of polynomial equations over $\overline{\mathbb{S}}$.
- (*solving*) use a solver to search $\mathbb{S}$ for the elements that satisfy $f_0,\ldots,f_k$;
- (*decoding*) use enc^{-1} to decode the solutions back to secrets;

We refer to a collection

$$\chi_\Lambda = (\mathsf{enc}, \mathsf{leak}, \mathsf{sys}(f_0,\ldots,f_k))$$

as Λ's *resolution model*.

Modeling inference attacks. As described so far, a polysys resolution model captures resolution attacks; that is, given some observed leakage one can recovery all the secrets that could have produced the leakage. The framework, however, can be extended to capture inference attacks as well. To do so, we include an objective function obj that takes as input an encoded secret $\mathbf{S}$ and an auxiliary distribution α over the secret space and assigns them a utility. An inference attack $\mathcal{A}$ can then be modeled as

$$\Psi_{\mathcal{A}}(\alpha) = (\alpha, \mathsf{obj}_{\mathcal{A}}, \chi_\Lambda)$$

where χ_Λ is Λ's resolution model and the solving step now consists of solving the optimization problem:

$$\arg\max_{\mathbf{S}\in\overline{\mathbb{S}}} \mathsf{obj}_{\mathcal{A}}(\mathbf{S}, \alpha),$$

subject to the constraints defined by the system of equations in χ_Λ. We refer to $\Psi_{\mathcal{A}}(\alpha)$ as $\mathcal{A}$'s *inference model* and, throughout, we use the term *polysys model* to refer to either a resolution or an inference model.

[1] Note that in certain cases, $\overline{\mathbb{S}}$ can be characterized by more than one equation.

Modeling the MAP adversary. In Section 5, we prove for any deterministic leakage profile, the adversary $\mathcal{A}_{\mathsf{map}}$ that computes the maximum a-posteriori (MAP) estimate has a polysys inference model. More precisely, we show that, for any deterministic profile Λ, the MAP adversary $\mathcal{A}_{\mathsf{map}}$ given auxiliary distribution α can be modeled as

$$\Psi_{\mathcal{A}_{\mathsf{map}}}(\alpha) = (\alpha, \alpha \circ \mathsf{enc}^{-1}, \chi_\Lambda).$$

This theorem is particularly significant because it allows us to construct inference models for various attack settings. In particular, we will use it to demonstrate that, depending on the dependency structure of α, the resulting objective function can have a lower degree, thereby increasing efficiency.

Modeling known-data attacks. Our framework can also capture known-data attacks and, perhaps more interestingly, provides a new perspective on them. In Section 4.4, we show that known-data attacks can be viewed as resolution attacks with a simplified system of equations. In other words, known-data attacks are effective because knowledge of the data reduces the number of variables in the polysys model.

Attacking point and range multi-maps. After formalizing our framework, we use it to generate state-of-the-art query and data recovery attacks on point and range multi-maps. (Specifically, we describe invertible encodings for multi-maps, sequences of point queries, and sequences of range queries.) We show how to model common leakage patterns, including the *query equality pattern*, which reveals if and when two queries are equal; the *size pattern*, which reveals the size of a multi-map; the *response identity pattern*, which reveals the responses to a query sequence; and the *response length* or *volume pattern*, which reveals the length of a query's response.

Based on our encodings and leakage functions, we then show how to build polysys models for the following set of attack settings for point multi-maps:

- query recovery with known data against the response identity
- query inference against the query equality pattern.

We also use our framework to generate polysys models for the following attack settings over range multi-maps:

- data resolution against the response identity
- approximate data attacks against the response identity
- approximate data resolution attacks against the query equality and the response identity.

The `PolySys` engine. The polysys models we build for the settings described above are quadratic, so in principle they can be solved by generating Gröbner basis methods. Unfortunately, this approach did not scale for our purposes since usually the algorithms that generate the basis (such as Buchberger's algorithm) are inefficient and scale exponentially with the number of variables and polynomials degree in the system. For instance, we were unable to scale past 10 queries for query recovery attacks against the response identity pattern using the `solve` method in the `SymPy` Python library [2]. To address this, we instead use SAT solvers, but not in a naive way. A naive approach would transform the model directly into CNF formulas; however, SAT solver efficiency generally decreases as the number of clauses increases, so we cannot do this without care. Instead, we apply a set of optimized transformations tailored to the types of equations found our polysys models. Some transformations are standard, such as the commander-variable encoding [72] and Tseitin decomposition [98], while others are new and of independent interest.

When using `PolySys` to design inference attacks, we did not rely on SAT-solvers, but constraint solvers from `Google OR-Tools` [89]. To do this, we first reformulated the objective function as a polynomial. Notably, we show that the dependency structure of the auxiliary distribution has a significant impact on the degree of the objective function. For the constraints in the system of equations of χ_Λ, we applied a standard technique known as *relinearization*, which reduces the degree of polynomials over $\mathbb{F}_2$ to linear. This reduction comes at the cost of increasing the number of variables, but it allows the system to be more efficiently solved.

Comparison to the state of the art. By utilizing the above transformations and techniques, we implement our `PolySys` engine in Python to perform query recovery, data resolution, and query inference attacks against the most common leakage patterns. When compared to prior state-of-the-art attacks on various real-world and synthetic workloads, we find that `PolySys` outperforms *all* attacks operating under identical attack settings. This includes the Count [23], Subgraph-ID [14], IHOP [87], Decoding-Bin [60], LMP-ID and LMP-APP [76], and GLMP [51] attacks. In some cases, `PolySys` achieves a recovery rate that is 60× higher than the best-known attack.

The only cases where `PolySys` underperforms are when competing attacks incorporate additional assumptions into their design. This is true for both the GenKKNO and ApproxValue attacks [52], which assume that queries are sampled uniformly——an assumption that `PolySys` deliberately avoids. Even in such cases, `PolySys` often still outperforms GenKKNO and ApproxValue in other scenarios.

Discussion on results and limitations. As discussed above, `PolySys` performs well against competing attacks operating under the same attack settings. However, leakage engines—and leakage attacks in general——should not be solely evalu-

ated under conditions in which they perform well. It is crucial to assess them across a variety of scenarios, including those where their recovery rates may decline. This is essential for determining whether attacks can be considered practical and for identifying conditions where cryptanalysis can be improved.

For PolySys, the recovery rate against the response identity pattern depends on the underlying query distribution. For instance, recovery rates dropped when the query distribution shifted from uniform to a Zipf distribution with parameter $\alpha = 2$ (see Figures 3c and 3d in Section 7.2). Similarly, the attack is also affected by the data distribution, which is illustrated by a decline in recovery rate when using a real dataset—specifically, the MIMIC dataset from [57]—as opposed to a synthetic dataset (see Figures 3c and 3a also in Section 7.2).

Moreover, as expected, our results show that the less information a pattern reveals, the more difficult it becomes to attack. Specifically, we evaluated PolySys against both the response length pattern (also known as the volume pattern) and the lesser-known total response length in the context of point multi-maps.[2] Our findings showed that, among all the leakage patterns we studied, PolySys had the lowest recovery rates against these two; suggesting that the response length and the total response length are likely very difficult to exploit.

Finally, another limitation of PolySys is its scalability. While our implementation performs well on small- to medium-sized datasets, it does not scale efficiently to larger datasets. For certain leakage patterns, the number of constraints and variables in the SAT-based PolySys engine grows quadratically, see Table 3. This presents a non-trivial practical cost as system variables increase. Nonetheless, we believe that PolySys represents a promising step toward a general framework for designing leakage attacks, and we are confident that future work can enhance its efficiency to handle larger datasets.

2 Related Work

Structured/searchable encryption. Structured encryption (STE) was introduced in [29] as a generalization of index-based searchable symmetric encryption [31, 46, 95]. One of the most widely studied STE primitives are multi-map encryption schemes since they are used as a building block for the majority of sub-linear encrypted search systems, including expressive search [24, 38], encrypted relational and non-relational databases [27, 63, 65], and other end-to-end encrypted applications [35, 67, 93]. Numerous works focus on various dimensions of multi-map encryption schemes including dynamism [26, 33, 68, 69], I/O efficiency [7, 18, 25, 82, 84], expressiveness [24, 38, 39, 62, 63, 88], concurrency [3, 4, 21, 65] and leakage suppression [45, 64, 66].

Leakage attacks. There is a long line of research on leakage attacks [14, 23, 51–53, 56, 60, 70, 73–75, 79, 87, 102] with point and range multi-maps as the main attack targets. This work compares the PolySys engine to state-of-the-art leakage attacks against a single leakage pattern.[3] First, we compare PolySys to attacks against point multi-maps, including to the Count [23] and Subgraph-ID [14] attacks which are query recovery attacks with known data against the response identity pattern rid. We also compare PolySys to the IHOP [87] and Decoding-Bin [60] attacks, which are query inference attacks against the query equality pattern qeq under the assumption that user queries are sampled from a Markov process.

We also consider attacks against range multi-maps. In particular, we compare PolySys to the LMP-ID and LMP-APP [75] range attacks, which are data resolution attacks against the response identity pattern rid that assume the dataset is dense. We also compare to GenKKNO [52] and ApproxValue [52] which are approximate data resolutions against the response identity pattern that require user queries to be sampled independently and uniformly at random. Note that PolySys requires neither assumption to operate. Finally, we compare to [51] which are count resolution attacks against the response length pattern (or the volume pattern) rlen.

The Bayle engine. We could not compare PolySys to the Bayle engine [36] for two reasons: first, the current version of Bayle is only designed to work for attacks against point multi-maps. Second, Bayle only scales to small label spaces (at most size 10); we evaluate on larger spaces in this work.

Gröbner basis. Solving polynomial equations can also be done by computing a Gröbner basis. Various algorithms exist for this computation, including the Buchberger algorithm [22], as well as Faugère's F4 [40] and F5 [41] algorithms. Several computer algebra systems (CASs) implement optimized versions of Buchberger's, F4, and F5 algorithms, such as Macaulay2 [50], Magma [17], Maple [78], Mathematica [100], SageMath [97], and Singular [32]. However, these algorithms do not scale well with respect to the number of equations and variables we handle—on the order of hundreds of thousands of equations and tens of thousands of variables. Roune and Stillman [94] evaluated various algorithms using different CASs, while Berthomieu et al. [12] compared their msolve polynomial system solver against Magma, Maple, and Singular. In particular, solving the Katsura-14 system—comprising 14 quadratic equations with 14 variables—using msolve took 15 days, whereas Maple and Magma were unable to solve it even after six months. This inefficiency is due to the inherent worst-case doubly exponential computational complexity of the Gröbner basis.

[2] While the response length pattern has been analyzed in the context of range multi-maps, as far as we know, neither of these patterns has been studied in the context of point multi-maps.

[3] Note that we do not compare against attacks that leverage more than a single pattern such as SelVol [14] or ARR [74].

SAT and SMT solvers in cryptography. SAT and SMT solvers are used to find solutions to the Boolean satisfiability problem. Modern SAT solvers are complex and use a combination of heuristic and optimizations to solve this problem practically. SAT solvers have been used in cryptography and security. Soos et al. [96] first used SAT solvers in the context of cryptography to attack stream ciphers. SAT solvers have also been used to attack hash functions [83], key schedules [59] and universal hash functions [49, 103], symmetric encryption schemes [11], and optimizing secure MPC [91].

Recently, Zinkus, Cao and Green used SAT solvers to design attacks against the *functionality-inherent leakage* (FIL) of reactive secure computation protocols [104]. In [104], FIL refers to the information about the inputs that are inherently revealed by the output of the functionality being evaluated and the authors consider adaptive attacks where the adversary can generate inputs to the protocol, observe the outputs, and use that to choose inputs for the next round. While related, our focus is on the leakage of sub-linear encrypted search solutions and we restrict ourselves to non-adaptive attacks. To the best of our knowledge, this is the first work that uses SAT solvers in the context of encrypted search.

SAT/SMT solvers in quantitative information flow. Quantitative Information Flow (QIF) is a framework developed to quantify the leakage of secret information in software systems. Given a program and a secret, QIF helps determine how much an adversary can infer about the secret by observing the program's output. There has been significant progress in automating QIF analysis using SAT/SMT solvers, particularly for estimating the number of distinct values that could lead to a set of observations [9, 54, 55, 71, 80, 81, 85, 90, 99]. This count can then be used compute an entropy-based metric. Automated QIF analysis has also been applied in the context of side-channel attacks, where the observations consist of side-channel information such as runtime or program size, and the adversary attempts to infer details about the secret. While related, our `PolySys` framework and engine are not primarily focused on quantifying leakage—though they can be used to do so by, as in QIF, counting the number of solutions—but on explicitly learning information about the secret.

3 Preliminaries

Notation. We use blackboard font $\mathbb{X}$ to denote sets, overlined blackboard font $\overline{\mathbb{X}}$ to denote subspaces of $\mathbb{F}_2^{n\times m}$ for some $n,m \geq 1$, calligraphic font $\mathcal{X}$ to denote probability distributions, uppercase italic X to denote random variables, uppercase bold font $\mathbf{X}$ to denote matrices, lowercase bold italic font $\boldsymbol{x}$ to denote vectors, and lowercase x to denote variables. Given a matrix $\mathbf{X}$, $\boldsymbol{x}_i$ denotes the ith row, $\boldsymbol{x}_i^{\intercal}$ denotes the ith column, and $x_{i,j}$ denotes the component at the ith row and the jth column. For some $m,n \geq 1$, we denote by $\mathbf{1}^{n\times m}$, the matrix with all 1s. The set of all binary strings of length n is denoted as $\{0,1\}^n$, and the set of all finite binary strings as $\{0,1\}^*$. We write $x \leftarrow \mathcal{D}$ to represent an element x being sampled from a distribution $\mathcal{D}$, and $x \xleftarrow{\$} \mathbb{S}$ to represent an element x being sampled uniformly at random from a set $\mathbb{S}$. The output x of an algorithm Alg is denoted by $x \leftarrow \mathsf{Alg}$. Given a set $\mathbb{S}$, we use $\#\mathbb{S}$ to refer to its cardinality.If $\mathbb{S}$ is a set with a total order and $a,b \in \mathbb{S}$, then $[a,b]$ is the set of all elements larger than a and smaller than b. When $\mathbb{S}$ is the set of non-negative integers $\mathbb{N}$, then $[a,b] = \{a, a+1, \cdots, b\}$. If n is a positive integer then $[n]$ is the set $\{1,\ldots,n\}$. We denote by $\mathcal{R}(n)$ the set of all ranges of size at most n; that is, $\mathcal{R}(n) = \{[a,b] : a \leq b \leq N\}$. When considering a totally ordered set $\mathbb{S}$, we will assume the existence of an efficiently-computable order-preserving bijection between $\mathbb{S}$ and the non-negative integers $[\#\mathbb{S}]$ and often identify the elements of $\mathbb{S}$ using their image under this bijection. We write $\mathbb{1}_{\{\mathsf{prop}\}}$ to denote the indicator function that equals 1 if the proposition prop is true and 0 otherwise. Given a random variable X, we write $\Pr[s]$ to denote $\Pr[X = s]$ when the random variable is clear from the context.

Multi-maps. A multi-map MM over a label space $\mathbb{L}$ and a value space $\mathbb{V}$ is a collection of label/tuple pairs $\{(\ell_i, \mathbf{v}_i)_i\}_{i\leq n}$, where for all $i \in [n]$, $\ell_i \in \mathbb{L}$ and for all $j \in \#\mathbf{v}_{i,j}$, $v_{i,j} \in \mathbb{V}$. Multi-maps support 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 assignment of the tuple $\mathbf{v}_i$ to label ℓ_i. We sometimes denote by $\mathbb{L}_{\mathsf{MM}}$ the set of labels stored in MM. We denote by $\mathcal{M}(\mathbb{L},\mathbb{V})$ the set of all multi-maps over label space $\mathbb{L}$ and value space $\mathbb{V}$. Multi-maps are the abstract data type instantiated by hash tables or binary trees.

Range multi-map. A range multi-map is a multi-map RMM over a label space $[N]$, for $N \in \mathbb{N}$, and a value space $\mathbb{V}$. In addition to a Get and Put operations, a range multi-map also supports range queries: given a range $r \in \mathcal{R}(N)$, return the set of values $V := \bigcup_{\ell\in r} \mathsf{RMM}[\ell]$. We write $V := \mathsf{RMM}[r]$ to denote getting the values associated with the range r. Range multi-maps are the abstract data type instantiated by binary trees, b-trees, or segment trees. In this work, we consider range multi-maps indexing a numerical column of a relational table or a numerical field in a non-relational table. With this assumption, the intersection of any two distinct tuples in the range multi-map is an empty set. This assumption captures the fact that a single numerical value exists at most in a given column cell. With this additional assumption, we can compare to all existing range attacks.

Structured encryption. Structured encryption (STE) schemes allow a client to outsource an encrypted data structure to a server while supporting private query and update operations. More precisely, a type-$\mathbf{T}$ structured encryption scheme $\Sigma = (\mathsf{Setup}, \mathsf{Query})$ consists of two efficient algo-

rithms. Setup takes as input a security parameter 1^k and a data structure DS of type $\mathbf{T}$ and outputs a secret key K and an encrypted data structure EDS. Query is a two-party protocol between a client and a server. The client inputs its secret key K and a query q and the server inputs an encrypted data structure EDS. The client receives a response r and the server receives $\perp$. We refer the reader to [29, 66] for more formal details. In this work, we focus on multi-map encryption and range multi-map encryption schemes.

Adversarial models and security. There are different adversarial models that we usually consider in the encrypted search area. The most common are *persistent* and *snapshot* adversaries. The former receives the encrypted data as well as the transcripts of all queries and updates. The latter is weaker and only receives the encrypted data after the execution of every query or update. Here, we study leakage patterns usually revealed at query time to a persistent adversary. STE security definitions are parametrized by a leakage profile, which were introduced by Curtmola et al. [31] and capture the following: given a leakage profile Λ, we say that an STE scheme is $\Lambda-$secure, if a persistent (or a snapshot) adversary cannot learn more than what is captured by the leakage profile, Λ. For formal definitions, we refer the reader to [29, 31].

4 The PolySys Framework

4.1 Secret Modeling

We now show how to encode query sequences, multi-maps and range multi-maps as $\mathbb{F}_2$-matrices but note that similar encodings can also be designed for other data types.

Encoding query sequences. We model a query sequence of length t over a label space $\mathbb{L}$ of size m, for $t, m \geq 1$, as a matrix in $\overline{\mathbb{Q}} \subset \mathbb{F}_2^{t \times m}$ and define the encoding

$$\mathsf{qseq} : \mathbb{L}^t \to \overline{\mathbb{Q}} \subset \mathbb{F}_2^{t \times m}, \quad \mathsf{qseq}(\ell_1, \ldots, \ell_t) = \mathbf{Q}$$

where $\mathbf{Q}$ is a matrix with rows indexed by the queries in the sequence and columns indexed by the labels in $\mathbb{L}$ and such that for all $i \in [t]$ and all $j \in [m]$,

$$q_{i,j} = \begin{cases} 1 & \text{if } q_i = \ell_j \\ 0 & \text{otherwise.} \end{cases}$$

By definition, $\mathbf{Q}$ can only have a single 1 in every row which means that $\overline{\mathbb{Q}}$ is the subspace of $\mathbb{F}_2^{t \times m}$ that satisfies the following t linear equations,

$$\sum_{j=1}^{m} q_{i,j} = 1, \qquad i \in [t]. \tag{1}$$

Encoding multi-maps. We model a multi-map over a label space $\mathbb{L}$ of size m and a value space $\mathbb{V}$ of size v, for $m, v \geq 1$, as a matrix in $\mathbb{F}_2^{m \times v}$ and define the encoding

$$\mathsf{mmap} : \mathcal{M}(\mathbb{L}, \mathbb{V}) \to \mathbb{F}_2^{m \times v}, \quad \mathsf{mmap}(\mathsf{MM}) = \mathbf{M}$$

where $\mathbf{M}$ is a matrix with rows indexed by the labels in $\mathbb{L}$ and the columns indexed by the values in $\mathbb{V}$ and such that for all $i \in [m]$ and all $j \in [v]$,

$$m_{i,j} := \begin{cases} 1 & \text{if } v_j \in \mathsf{MM}[\ell_i] \\ 0 & \text{otherwise.} \end{cases}$$

Encoding range sequences. We model a range query sequence of length t over range space $\mathcal{R}(N)$, where $N \in \mathbb{N}$ as a matrix in $\overline{\mathbb{R}} \subseteq \mathbb{F}_2^{t \times N}$ and use the encoding

$$\mathsf{rseq} : \mathcal{R}(N)^t \to \overline{\mathbb{R}} \subset \mathbb{F}_2^{t \times N}, \quad \mathsf{rseq}(r_1, \ldots, r_t) = \mathbf{R}$$

where $\mathbf{R}$ is a matrix with rows indexed by range queries and columns indexed by labels in $[N]$ and such that for all $i \in [t]$ and all $j \in [N]$,

$$r_{i,j} = \begin{cases} 1 & \text{if } j \in r_i \\ 0 & \text{otherwise.} \end{cases}$$

By definition, every row in $\mathbf{R}$ can only have 1-values in a single, contiguous region in $[N]$. This region would then represent the queried range. This implies that $\overline{\mathbb{R}}$ is the subspace of $\mathbb{F}_2^{t \times m}$ that satisfies the following t quadratic equations,

$$\sum_{j=1}^{N-1} r_{i,j} \cdot r_{i,j+1} = \sum_{j=1}^{N} r_{i,j} + 1, \qquad i \in [t]. \tag{2}$$

Encoding range multi-maps. We model a range multi-map over the label space $[N]$ and a value space $\mathbb{V}$ of size v, for $N, v \geq 1$, as a matrix in $\mathbb{F}_2^{m \times v}$ and define the encoding

$$\mathsf{rmmap} : \mathcal{M}([N], \mathbb{V}) \to \overline{\mathbb{C}} \subset \mathbb{F}_2^{m \times v}, \quad \mathsf{rmmap}(\mathsf{RMM}) = \mathbf{C}$$

where $\mathbf{C}$ is a matrix with rows indexed by the integers in $[N]$ and the columns indexed by the values in $\mathbb{V}$ and such that for all $i \in [N]$ and all $j \in [v]$,

$$c_{i,j} := \begin{cases} 1 & \text{if } v_j \in \mathsf{RMM}[i] \\ 0 & \text{otherwise.} \end{cases}$$

As discussed in Section 3, we consider range multi-maps where a value can only exist in at most one tuple. This implies thta $\mathbf{C}$ can only have a single 1 in every column which means that $\mathbf{C}$ is the subspace of $\mathbb{F}_2^{t \times m}$ that satisfies the following t linear equations,

$$\sum_{i=1}^{N} c_{i,j} = 1, \qquad j \in [v]. \tag{3}$$

Scheme	Type	Leakage Pattern			
		qeq	rid	rlen	trlen
Z-IDX [46]		✓	✓	✓	✓
SSE-1, SSE-2 [31]		✓	✓	✓	✓
π_{bas}, π_{pack}, π_{2Lev} [26]		✓	✓	✓	✓
Sophos [19]		✓	✓	✓	✓
Diana [20]	point	✓	✓	✓	✓
VLH, AVLH [64]		✓	✗	✗	✗
Tethys [18]		✓	✗	✓	✓
FIX [6]		✓	✗	✗	✗
AZL [66]		✗	✗	✗	✓
FJK$^+$ [38]		✓	✓	✓	✓
L-BRC, C-BRC [34]		✓	✓	✓	✓
BPP [15]	range	✓	✓	✓	✓
π_{rq} [77]		✓	✓	✓	✓
POPE [92]		✓	✓	✓	✓

Table 1: A non-exhaustive list of schemes and their leakage. Note that schemes may leak more than the patterns listed above and that rid reveals rlen which in turn reveals trlen.

Remark. We refer to Equation 1 as the structural equation for multi-maps, and Equations 2 and 3 as the structural equations for range multi-maps.

4.2 Leakage Modeling

The most common representation of leakage patterns in literature is the functional representation. Each pattern maps the query and/or the data space of the underlying data type to some output space. Here, we follow the same approach where we additionally view the output of any mapping as a matrix. In the following, we describe the standard leakage patterns and refer the reader to [66] for more details. Note that all the patterns we discuss below are described in the same manner for all multi-maps, including for range multi-maps. The leakage patterns we study in this work are the most common in the encrypted search literature, as highlighted in Table 1.

Multi-map size pattern. The multi-map size reveals the size of a multi-map which is defined as its number of label/value pairs. More precisely, it is defined as

$$\mathsf{mmsize} : \mathcal{M}(\mathbb{L},\mathbb{V}) \to \mathbb{N}^{1\times 1}, \quad \mathsf{mmsize}(\mathsf{MM}) = \mathbf{S}$$

where $\mathbf{S}$ is a 1×1 matrix such that

$$s_{1,1} = \sum_{\ell\in\mathbb{L}} \#\mathsf{MM}[\ell].$$

Query equality pattern. The query equality reveals if and when a query is repeated. More precisely, it is defined as

$$\mathsf{qeq} : \mathbb{Q}^t \to \mathbb{F}_2^{t\times t}, \quad \mathsf{qeq}(q_1,\ldots,q_t) = \mathbf{E}$$

where $\mathbf{E} \in \mathbb{F}_2^{t\times t}$ is such that

$$e_{i,j} = \begin{cases} 1 & \text{if } q_i = q_j, \\ 0 & \text{otherwise.} \end{cases}$$

Response identity pattern. The response identity reveals the responses of a query sequence and is defined as

$$\mathsf{rid} : \mathcal{M}(\mathbb{L},\mathbb{V}) \times \mathbb{L}^t \to \mathbb{F}_2^{t\times v}, \quad \mathsf{rid}(\mathsf{MM},\ell_1,\ldots,\ell_t) = \mathbf{I}$$

$$\mathrm{i}_{i,j} = \begin{cases} 1 & \text{if } v_j \in \mathsf{MM}[\ell_i] \\ 0 & \text{otherwise.} \end{cases}$$

Response length (volume) pattern. The response length pattern, which is also referred to as the volume pattern, reveals the sizes of the responses of a query sequence. More precisely,

$$\mathsf{rlen} : \mathcal{M}(\mathbb{L},\mathbb{V}) \times \mathbb{L}^t \to \mathbb{N}^{t\times 1}, \quad \mathsf{rlen}(\mathsf{MM},\ell_1,\ldots,\ell_t) = \mathbf{L}$$

where $l_{i,1} = \#\mathsf{MM}[\ell_i]$.

Total response length pattern. The total response length pattern reveals the total sizes of the responses of a query sequence. More precisely, it is defined as

$$\mathsf{trlen} : \mathcal{M}(\mathbb{L},\mathbb{V}) \times \mathbb{L}^t \to \mathbb{N}^{1\times 1}, \quad \mathsf{trlen}(\mathsf{MM},\ell_1,\ldots,\ell_t) = \mathbf{T}$$

where $t_{1,1} = \sum_{i=1}^{t} \#\mathsf{MM}[\ell_i]$.

4.3 Leakage Equations

In the following, we show how to derive the leakage polynomial systems for four leakage patterns: the data size pattern, the query equality pattern, the response identity pattern, and the response length pattern, and for two data types: multi-maps and range multi-maps. As highlighted earlier, these patterns and data types are one of the most studied in STE/SSE literature both in terms of design and cryptanalysis.

Size equations. Based on our multi-map encoding mmap and size function mmsize, we can now define the following matrix equation that relates multi-maps to their size. For all $\mathsf{MM} \in \mathcal{M}(\mathbb{L},\mathbb{V})$,

$$\mathbf{1}^{1\times m} \cdot \mathbf{M} \cdot \mathbf{1}^{v\times 1} = \mathbf{S}$$

where $\mathbf{M} = \mathsf{mmap}(\mathsf{MM}) \in \mathbb{F}_2^{m\times v}$ and $\mathbf{S} = \mathsf{mmsize}(\mathsf{MM}) \in \mathbb{N}^{1\times 1}$. This equation, however, can be rewritten as the following linear equation over $\mathbb{F}_2$,

$$\sum_{i=1}^{m}\sum_{j=1}^{v} m_{i,j} = s_{1,1} \tag{4}$$

Similarly, for range multi-maps, we obtain the following linear equation over $\mathbb{F}_2$,

$$\sum_{i=1}^{N}\sum_{j=1}^{v} c_{i,j} = s_{1,1}, \tag{5}$$

where $\mathbf{C} = \mathsf{rmmap}(\mathsf{RMM}) \in \mathbb{F}_2^{N\times v}$.

Query equality equations. Based on our query sequence encoding qseq and our query equality function qeq, we define the following matrix equation which relates query sequences to their query equality. For all $(\ell_1,\ldots,\ell_t) \in \mathbb{L}^t$, we have that

$$\mathbf{Q}\cdot\mathbf{Q}^{\mathsf{T}} = \mathbf{E},$$

where $\mathbf{Q} = \mathsf{qseq}(\ell_1,\ldots,\ell_t) \in \overline{\mathbb{Q}} \subset \mathbb{F}_2^{t\times m}$ and $\mathbf{E} = \mathsf{qeq}(\ell_1,\ldots,\ell_t) \in \mathbb{F}_2^{t\times t}$, which can be rewritten as the following t^2 quadratic equations over $\mathbb{F}_2$,

$$\sum_{k=1}^{t} q_{i,k}\cdot q_{j,k} = e_{i,j}, \qquad i,j \in [t]. \tag{6}$$

In the case of range multi-maps, the formulation of the system is similar but a bit more complex as it involves introducing a new matrix variable, $\overline{\mathbf{R}} \in \mathbb{F}_2^{t\times\lambda}$ that maps a range query to its corresponding range index, i.e., every row in $\overline{\mathbf{R}}$ is a unitary vector, where $\lambda = \#\mathcal{R}(N) = N(N+1)/2$. In particular, based on our range query sequence encoding rseq and our query equality function qeq, we define the following matrix equation that relates range query sequences to their query equality. For all $(r_1,\ldots,r_t) \in \mathcal{R}(N)^t$, we have that

$$\overline{\mathbf{R}}\cdot\overline{\mathbf{R}}^{\mathsf{T}} = \mathbf{E},$$

where $\overline{r}_{i,j} = \prod_{k\in r_j} r_{i,k} \cdot \prod_{k\in[N]\setminus r_j}(1 - r_{i,k})$, $\mathbf{R} = \mathsf{rseq}(r_1,\ldots,r_t) \in \overline{\mathbb{R}} \subset \mathbb{F}_2^{t\times N}$ and $\mathbf{E} = \mathsf{qeq}(r_1,\ldots,r_t) \in \mathbb{F}_2^{t\times t}$. The above matrix product can be rewritten as the following degree-N polynomials over $\mathbb{F}_2$,

$$\sum_{k=1}^{t} \overline{r}_{i,k}\cdot\overline{r}_{j,k} = e_{i,j} \qquad i,j \in [t] \tag{7}$$

Response identity equations. Based on our multi-map encoding mmap, our query sequence encoding qseq and our response identity function rid, we can define the following matrix equation that relates multi-maps and query sequences to their response identity. For all multi-maps $\mathsf{MM} \in \mathcal{M}(\mathbb{L},\mathbb{V})$ and query sequences $(\ell_1,\ldots,\ell_t) \in \mathbb{L}^t$, we have

$$\mathbf{Q}\cdot\mathbf{M} = \mathbf{I},$$

where $\mathbf{Q} = \mathsf{qseq}(\ell_1,\ldots,\ell_t) \in \overline{\mathbb{Q}} \subset \mathbb{F}_2^{t\times m}$, $\mathbf{M} = \mathsf{mmap}(\mathsf{MM}) \in \mathbb{F}_2^{m\times v}$ and $\mathbf{I} = \mathsf{rid}(\mathsf{MM},\ell_1,\ldots,\ell_t) \in \mathbb{F}_2^{t\times v}$. Note, however, that this equation can be rewritten as the following $t\cdot n$ quadratic equations over $\mathbb{F}_2$,

$$\sum_{k=1}^{t} q_{i,k}\cdot m_{k,j} = \mathsf{i}_{i,j}, \qquad i \in [t], j \in [n]. \tag{8}$$

Similarly, using our range sequence encoding rseq, we can write the following matrix equation for all range multi-maps $\mathsf{RMM} \in \mathcal{M}([N],\mathbb{V})$ and query sequences $(r_1,\ldots,r_t) \in \mathcal{R}(N)^t$,

$$\mathbf{R}\cdot\mathbf{C} = \mathbf{I}$$

where $\mathbf{R} = \mathsf{rseq}(r_1,\ldots,r_t) \in \overline{\mathbb{R}} \subset \mathbb{F}_2^{t\times N}$, $\mathbf{C} = \mathsf{rmmap}(\mathsf{RMM}) \in \mathbb{F}_2^{N\times v}$ and $\mathbf{I} = \mathsf{rid}(\mathsf{MM},r_1,\ldots,r_t) \in \mathbb{F}_2^{t\times v}$. This can be rewritten as the following $t\cdot n$ quadratic equations over $\mathbb{F}_2$

$$\sum_{k=1}^{t} r_{i,k}\cdot c_{k,j} = \mathsf{i}_{i,j}, \qquad i \in [t], j \in [n]. \tag{9}$$

Response length (volume) equations. As above, we can use our multi-map encoding mmap, our query sequence encoding qseq and our response length function rlen to define a matrix equation that relates multi-maps and query sequences to their response length. For all $\mathsf{MM} \in \mathcal{M}(\mathbb{L},\mathcal{V})$ and $(\ell_1,\ldots,\ell_t) \in \mathbb{L}^t$, we have

$$\mathbf{Q}\cdot\mathbf{M}\cdot\mathbf{1}^{v\times 1} = \mathbf{L},$$

where $\mathbf{Q} = \mathsf{qseq}(\ell_1,\ldots,\ell_t) \in \overline{\mathbb{Q}} \subset \mathbb{F}_2^{t\times m}$, $\mathbf{M} = \mathsf{mmap}(\mathsf{MM}) \in \mathbb{F}_2^{m\times v}$ and $\mathbf{L} = \mathsf{rlen}(\mathsf{MM},\ell_1,\ldots,\ell_t) \in \mathbb{F}_2^{t\times 1}$. Note, however, that this equation can be rewritten as the following t quadratic equations over $\mathbb{F}_2$,

$$\sum_{j=1}^{v}\left(\sum_{k=1}^{t} q_{i,k}\cdot m_{k,j}\right) = l_i \qquad i \in [t]. \tag{10}$$

Similarly, using our range sequence encoding rseq, we can write the following matrix equation for all range multi-maps $\mathsf{RMM} \in \mathcal{M}([N],\mathbb{V})$ and query sequences $(r_1,\ldots,r_t) \in \mathcal{R}(N)^t$,

$$\mathbf{R}\cdot\mathbf{C}\cdot\mathbf{1}^{v\times 1} = \mathbf{L}$$

where $\mathbf{R} = \mathsf{rseq}(r_1,\ldots,r_t) \in \overline{\mathbb{R}} \subset \mathbb{F}_2^{t\times N}$, $\mathbf{C} = \mathsf{rmmap}(\mathsf{RMM}) \in \mathbb{F}_2^{N\times v}$ and $\mathbf{I} = \mathsf{rid}(\mathsf{MM},r_1,\ldots,r_t) \in \mathbb{F}_2^{t\times}$. This can be rewritten as the following t quadratic equations over $\mathbb{F}_2$

$$\sum_{j=1}^{v}\left(\sum_{k=1}^{t} r_{i,k}\cdot c_{k,j}\right) = l_i \qquad i \in [t]. \tag{11}$$

Total response length equations. As above, we can use our multi-map encoding mmap, our query sequence encoding qseq and our total response length function trlen to define a matrix equation that relates multi-maps and query sequences to their total response length. For all $\mathsf{MM} \in \mathcal{M}(\mathbb{L},\mathcal{V})$ and $(\ell_1,\ldots,\ell_t) \in \mathbb{L}^t$, we have

$$\mathbf{1}^{1\times t}\cdot\mathbf{Q}\cdot\mathbf{M}\cdot\mathbf{1}^{v\times 1} = \mathbf{L},$$

where $\mathbf{Q} = \mathsf{qseq}(\ell_1,\ldots,\ell_t) \in \overline{\mathbb{Q}} \subset \mathbb{F}_2^{t\times m}$, $\mathbf{M} = \mathsf{mmap}(\mathsf{MM}) \in \mathbb{F}_2^{m\times v}$ and $\mathbf{T} = \mathsf{trlen}(\mathsf{MM},\ell_1,\ldots,\ell_t) \in \mathbb{F}_2^{1\times 1}$. Note, however, that this equation can be rewritten as the following quadratic equation over $\mathbb{F}_2$,

$$\sum_{i=1}^{t}\sum_{j=1}^{v}\left(\sum_{k=1}^{t} q_{i,k}\cdot m_{k,j}\right) = t_1. \tag{12}$$

4.4 Handling Known-Data Attacks

As described, our framework addresses resolution attacks. We now show how it can also be used to capture known-data attacks. In particular, we will see that knowledge of the data translates directly to fewer variables in the model of the attack.

The polysys model of a known-data attack parallels that of a resolution attack. However, if we know a subset $\{(\ell_{k_i}, \mathbf{v}_{k_i})\}_{i\in[\beta]}$ of label/tuple pairs in the multi-map, then for all $i \in [\beta]$ and $j \in [n]$, we can set

$$m_{k_i,j} = 1 \quad \text{if} \quad v_j \in \mathbf{v}_{k_i}.$$

If we have complete knowledge of the *entire* data structure, we can further reduce the number of variables by assigning 0 to all values known not to appear in the structure. Formally, for all $i \in [\beta]$ and $j \in [n]$, we can set

$$m_{k_i,j} = \mathbb{1}_{\{v_j \in \mathbf{v}_{k_i}\}}.$$

The system requires $m \cdot n$ (or $N \cdot n$ in the case of a range multi-map) fewer variables. Also, all leakage equations except for the query equality equations become linear, substantially reducing the computational cost of solving the system.

5 Extension to Inference Attacks

In the previous section, we showed how our framework can be used to model and design resolution and known-data attacks. We now show how to extend it to handle inference attacks. At a high level, we do this by formulating an optimization problem based on the resolution model of the leakage profile under consideration. In this problem, the objective function depends on the attack and the adversary's auxiliary distribution. But before we can describe the approach in more detail, we recall the formalization of inference attacks from [36].

Inference attacks. Recall that an inference attack takes as input the observed leakage and an auxiliary distribution over the secret space. Using the formalization of [36], a leakage model is a joint random variable $\mathbf{X} = (\mathbf{S}, \mathbf{H}, \mathbf{L})$, where $\mathbf{S}$ is a multivariate random variable over a secret space $\mathbb{S}$, $\mathbf{H}$ is a multivariate random variable over a hidden space $\mathbb{H}$ and $\mathbf{L}$ is a multivariate random variable over a leakage space $\mathbb{K}$. In this framework, leakage patterns define the likelihood functions $p(\boldsymbol{\ell} \mid \boldsymbol{s})$, for all $\boldsymbol{s} \in \mathbb{S}$, of the joint random variable and leakage attacks are statistical inference algorithms $\mathcal{A}(\boldsymbol{\ell}, \alpha)$ that output an estimate $\hat{\boldsymbol{s}}$ of $\boldsymbol{s}$, where $(\boldsymbol{s}, \boldsymbol{h}, \boldsymbol{\ell}) \leftarrow \mathbf{X}$ and α is an auxiliary distribution over the secret space $\mathbb{S}$. Note that, in the BLA framework, the adversary $\mathcal{A}$ works over the joint random variable $\widetilde{\mathbf{X}} = (\mathbf{A}, \mathbf{H}, \mathbf{L})$, where the secret random variable $\mathbf{S}$ is replaced with an auxiliary random variable $\mathbf{A} \sim \alpha$. In [36], the authors study several adversaries including one based on the maximum a-posteriori (MAP) estimate which outputs

$$\mathsf{map}_{\mathbb{S}}(\boldsymbol{\ell}, \alpha) \stackrel{\text{def}}{=} \operatorname*{arg\,max}_{\boldsymbol{s}\in\mathbb{S}} p(\boldsymbol{\ell} \mid \boldsymbol{s}) \cdot \alpha(\boldsymbol{s}).$$

Modeling the MAP adversary. We now show how the MAP adversary described above can be modeled in our framework when the leakage profile is deterministic.

Theorem 1. *Let $\chi_\Lambda = (\mathsf{enc}, \mathsf{leak}, \mathsf{sys}(f_0, \ldots, f_k))$ be a resolution model for a deterministic leakage profile Λ. Then, for all auxiliary distributions α over the secret space $\mathbb{S}$, the adversary $\mathcal{A}_{\mathsf{map}}$ can be modeled as*

$$\Psi_{\mathcal{A}_{\mathsf{map}}}(\alpha) = (\alpha, \alpha \circ \mathsf{enc}^{-1}, \chi_\Lambda).$$

We use Theorem 1 to formulate inference models of MAP-based query recovery attacks in various settings:

- *arbitrary auxiliary distributions*, where queries can exhibit any form of dependency;
- *Markovian auxiliary distributions*, where a query can depend on at most the query that precedes it;
- *independent auxiliary distributions*, where the queries are independent.

Due to space constraints, we only describe the inference model for arbitrary auxiliary distribution and defer the last two cases to the full version of the paper. One of the main observations is that, the less dependency the queries exhibit the simpler the objective function of the inference model will be and the more computationally efficient the attack will be.

Remark. While we focus on point multi-maps and query recovery attacks, the same approach can be used for other structures and other types of recovery.

5.1 Arbitrary Query Sequences

Theorem 1 allows us to translate the MAP adversary to a polysys inference model but, for computational purposes, the objective function in the model has to formulated as a polynomial with some fixed degree. In the following corollary, we show that this is the case. Due to space constraints, the proof of the theorem is in the full version.

Corollary 1. *Let Λ be a deterministic leakage profile and $\mathcal{A}_{\mathsf{map}}$ be the query recovery adversary that outputs the MAP estimate. Then, for all auxiliary distributions α over the query sequence space $\mathbb{L} = \mathbb{L}_1 \times \cdots \times \mathbb{L}_t$, $\mathcal{A}_{\mathsf{map}}$ can be modeled as*

$$\Psi_{\mathcal{A}_{\mathsf{map}}}(\alpha) = \left(\alpha, \alpha \circ \mathsf{enc}^{-1}, \chi_\Lambda\right)$$

where the objective function $\alpha \circ \mathsf{enc}^{-1}$ is a degree-t polynomial over $\mathbb{F}_2$. Specifically,

$$\alpha\left(\mathsf{enc}^{-1}(\mathbf{Q})\right) = \sum_{i=1}^{t}\left(\sum_{j_1,\ldots,j_i\in[m]} q_{1,j_1}\cdots q_{i,j_i} \cdot \log\left(\alpha\left(\ell_{j_i} \mid \ell_{j_{i-1}}, \ldots, \ell_{j_1}\right)\right)\right).$$

Relinearization. Given that the objective function as well as the system of equations composing the constraints are polynomials over $\mathbb{F}_2$, we make use of a standard technique called *relinearization* that reduces the degree of polynomials at the cost of increasing the number of variables. Due to space limitations, we discuss the details in the full version.

6 The `PolySys` Engine

Various methods can be used to solve systems of polynomial equations. In this work, we use SAT solvers, which are designed to solve the Boolean satisfiability problem (SAT). Given a set of clauses in conjunctive normal form (CNF), a SAT-solver determines whether there is a truth-value assignment to the variables that makes the formula true.

Our main observation is that finding a solution to a system of polynomial equations can be reduced to transforming the system into an equivalent CNF and checking whether it is satisfiable. The main technical challenge lies in converting each polynomial equation into a set of Boolean clauses, which are then expressed in CNF. The CNF of the entire system is simply the conjunction of the CNFs corresponding to each individual equation.

Because SAT-solver performance depends on the size of the formula—both in terms of variables and clauses—our transformations aim to minimize these quantities. The following section details how to transform polynomial equations into CNF. We categorize these equations into four types, which are described in the subsections that follow.

Binary sums. This category includes the linear equations that sum to one. In particular, both Equation 1 and Equation 3 have the following format $\sum_{i=1}^{\gamma} x_i = 1$ where $x_i \in \{0,1\}$ for all $i \in [\gamma]$ and $\gamma \in \mathbb{N}$. Such a summation can be represented as the following Boolean clause

$$\left(\bigvee_{i\in[\gamma]} x_i\right) \wedge \left(\bigwedge_{i\neq j} \overline{x}_i \vee \overline{x}_j\right)$$

where the binary variables, x_i, can also be considered to be propositional variables in $\{\mathsf{false}, \mathsf{true}\}$. The first term captures that at least one variable is equal to 1 while the latter captures that at most one variable equals 1. While the formulation is already in its CNF, the number of constraints in the second term is $O(\gamma^2)$, which is quite prohibitive. We instead use a standard encoding, called *commander-variable encoding* [72], where we can define a recursive function that enforces that at most one variable in a given sequence is equal to one, and which we denote by AMO, defined as,

$$\mathsf{AMO}(x_1,\ldots,x_\gamma) = \mathsf{ND}(x_1,\ldots,x_3,y) \wedge \mathsf{AMO}(\overline{y},x_4,\ldots,x_\gamma),$$

where y is a new propositional variable and where ND is the naive Boolean expression for a conjunction of disjunctions defined as,

$$\mathsf{ND}(x_1,\ldots,x_3,y) = \left(\bigwedge_{\substack{i,j\in[3]\\ i\neq j}} \overline{x}_i \vee \overline{x}_j\right) \wedge \left(\bigwedge_{i\in[3]} \overline{x}_i \vee \overline{y}\right).$$

Using the commander-variable encoding, the number of Boolean clauses composing the CNF becomes $O(\gamma)$ and with $O(\gamma)$ additional variables. The final Boolean formula is :s

$$\left(\bigvee_{i\in[\gamma]} x_i\right) \wedge \mathsf{AMO}(x_1,\ldots,x_\gamma).$$

Quadratic sums. This category includes equations with quadratic terms such that their sum is binary. In particular, Equations 6, 8 and 9 have the following format,

$$\sum_{i\in[\gamma]} x_i y_i = b \tag{13}$$

where $x_i, y_i \in \{0,1\}$ for all $i \in [\gamma]$ and $\gamma \in \mathbb{N}$, and $b \in \{0,1\}$. In the following, we analyze the cases when $b = 0$ and $b = 1$.

Case 1. When $b = 0$, then we can reformulate the sum in Equation 13 as, $\bigwedge_{i\in[\gamma]} \overline{x}_i \vee \overline{y}_i$, which is already a CNF format with $O(\gamma)$ clauses.

Case 2. When $b = 1$, then we can reformulate the sum in Equation 13 as the following Boolean formula

$$\left(\bigvee_{i\in[\gamma]} x_i \wedge y_i\right) \wedge \left(\bigwedge_{i\neq j} \overline{x}_i \vee \overline{y}_i \vee \overline{x}_j \vee \overline{y}_j\right),$$

where the first term captures the fact that at least one product $x_i y_i$, for $i \in [\gamma]$, is equal to 1, whereas the second term captures the fact that at most, one term is equal to 1. Notice that the first term is not a CNF, and using the (naive) De Morgan's distribution laws, the resulting CNF would be composed of $O(2^\gamma)$ clauses, which is extremely prohibitive. Instead, we use Tseitin decomposition [98], which works as follows. We introduce γ new propositional variables z_i such that $z_i \implies x_i \wedge y_i$, and we can rewrite the disjunctions of conjunctions:

$$\bigvee_{i\in[\gamma]} x_i \wedge y_i \equiv \left(\bigvee_{i\in\gamma} z_i\right) \wedge \left(\bigwedge_{i\in[\gamma]} (\overline{z}_i \vee x_i) \wedge (\overline{z}_i \vee y_i)\right)$$

which is in CNF with $O(\gamma)$ clauses and $O(\gamma)$ additional variables. The second term of our Boolean clause has $O(\gamma^2)$ clauses, but by leveraging commander-variable encoding, we can reduce it to $O(\gamma)$ using the recursive AMO function:

$$\bigwedge_{i\neq j} \overline{x}_i \vee \overline{y}_i \vee \overline{x}_j \vee \overline{y}_j \equiv \mathsf{AMO}(x_1 \wedge y_1,\ldots,x_\gamma \wedge y_\gamma).$$

To sum up, the resulting Boolean clause has $O(\gamma)$ clauses and $O(\gamma)$ variables and is equal to

$$\left(\bigvee_{i\in\gamma} z_i\right) \wedge \left(\bigwedge_{i\in[\gamma]} (\overline{z}_i \vee x_i) \wedge (\overline{z}_i \vee y_i)\right) \wedge \mathsf{AMO}(x_1 \wedge y_1, \ldots, x_\gamma \wedge y_\gamma).$$

The z_i variables provide us an equisatisfiable formula such that an assignment of the x_i and y_i variables to the rewritten formula is also a satisfying instance to the original.

Mixed sums. This category includes equations that have both quadratic and linear terms. In our case, only Equation 2 fits in this category and has the following format,

$$\sum_{i\in[\gamma-1]} x_i x_{i-1} = \sum_{i\in[\gamma]} x_i + 1.$$

A naive Boolean representation of the equation above is to list all contiguous ranges such that,

$$\bigvee_{i\in[\gamma],j\in[i,\gamma]} \left(\left(\bigwedge_{k\in[i,j]} x_k\right) \wedge \left(\bigwedge_{k\notin[i,j]} \overline{x}_k\right)\right)$$

This Boolean expression has $O(\gamma^2)$ clauses and is in a DNF. It can be reformulated naively into a CNF form, resulting in $O(2^{2\gamma})$ clauses. We can transform this expression into one that only requires $O(\gamma)$ clauses and $O(\gamma)$ additional variables. We first introduce the variables $a_1, \ldots, a_\gamma$ and $b_1, \ldots, b_\gamma$ where $a_i = \mathsf{true}$ if there exists $j \geq i$ such that $x_j = \mathsf{true}$; and $b_i = \mathsf{true}$ if there exists $j \leq i$ such that $x_j = \mathsf{true}$. We then reformulate these two definitions as the following Boolean clauses,

$$x_i \implies a_i \equiv \overline{x}_i \vee a_i \quad \text{and} \quad a_{i+1} \implies a_i \equiv \overline{a}_{i+1} \vee a_i$$

$$x_i \implies b_i \equiv \overline{x}_i \vee b_i \quad \text{and} \quad b_i \implies b_{i+1} \equiv \overline{b}_i \vee b_{i+1}$$

We finally ensure that there is no index $i \in [\gamma]$ such that $x_i = \mathsf{false}$ and $a_i = \mathsf{true}$ and $b_i = \mathsf{true}$. We capture this by,

$$x_i \vee \overline{a}_i \vee \overline{b}_i, \quad i \in [\gamma].$$

The final Boolean expression is the conjunction of all these clauses, which is in a CNF such that,

$$\left(\bigwedge_{i\in[\gamma]} \overline{x}_i \vee a_i\right) \wedge \left(\bigwedge_{i\in[\gamma]} \overline{x}_i \vee b_i\right) \wedge \left(\bigwedge_{i\in[\gamma-1]} \overline{a}_{i+1} \vee a_i\right) \wedge \left(\bigwedge_{i\in[\gamma-1]} \overline{b}_i \vee b_{i+1}\right) \wedge \left(\bigwedge_{i\in[\gamma]} x_i \vee \overline{a}_i \vee \overline{b}_i\right).$$

Non-binary linear sums. This category includes equations composed of quadratic terms such that their sum is an integer strictly larger than 0. In particular, Equations 11 and 12 have the following format,

$$\sum_{j\in[n]} \sum_{i\in[\gamma]} x_i y_{i,j} = c,$$

where $c \in [n]$ and $\sum_{i\in[\gamma]} x_i y_{i,j} \in \{0,1\}$. A naive representation of the equation above as a Boolean expression is,

$$\bigvee_{(j_1,\ldots,j_c)\in C_n} \left(\left(\bigwedge_{i\in[c]} B_{j_i}\right) \wedge \left(\bigwedge_{i\in[n]\setminus j_1,\ldots,j_c} \overline{B}_i\right)\right)$$

where B_j is the Boolean expression corresponding to the sum $\sum_{i\in[\gamma]} x_i y_{i,j} \in \{0,1\}$. The expression above captures the fact that $\binom{n}{c}$ possible combinations, C_n, of indexes in $\{1,\ldots,n\}$ can result in a sum equal to c. The above Boolean expression is also very costly to solve as it is written in a DNF, and a naive CNF representation would result in $2^{\gamma \cdot n}$ clauses. Instead, we leverage the *sequential counter encoding* result by Frisch and Giannaros [42] where we can generate the Boolean expression with $O((c+\gamma)\cdot n)$ clauses and $O((c+\gamma)\cdot n)$ additional variables, where $c \in [n]$. We denote by AMC_c the function that takes as input n truth variables B_i and generates at most c-out-of-n combinations. The resulting CNF is defined as the conjunction of the following five clauses,

$$\bigwedge_{i=2}^{c} \overline{R}_{1,j} \quad \text{and} \quad \bigwedge_{i=2}^{n-1} \bigwedge_{j\in[c]} \overline{R}_{i-1,j} \vee R_{i,j} \quad \text{and} \quad \bigwedge_{i\in[n-1]} \overline{B}_i \vee R_{i,1}$$

$$\bigwedge_{i=2}^{n-1} \bigwedge_{j=2}^{c} \overline{B}_i \vee \overline{R}_{i-1,j-1} \vee R_{i,j} \quad \text{and} \quad \bigwedge_{i\in[n]} \overline{B}_i \vee R_{i-1,c},$$

where $R_{i,j}$ are truth variables (called registers in [42]), for all $i \in [n]$ and $j \in [c]$. In our case, to get the Boolean expression for *exactly* c-out-of-n, we take the conjunction of both at most and at least c-out-of-n defined as,

$$\mathsf{AMC}_c(B_1,\ldots,B_n) \wedge \mathsf{AMC}_c(\overline{B}_1,\ldots,\overline{B}_n). \tag{14}$$

As a second step, we need to compute the concrete Boolean expression of B_i, for $i \in [n]$. For this, we start by leveraging Tseitin's decomposition [98] where we introduce $\gamma \cdot n$ variables $z_{i,j}$ such that $z_{i,j} \Leftrightarrow x_i \wedge y_{i,j}$. This equivalence can be represented as, for all $i \in [\gamma]$ and $j \in [n]$,

$$(\overline{z}_i \vee x_i) \wedge (\overline{z}_i \vee y_{i,j}) \wedge (z_i \vee \overline{x}_i \vee \overline{y}_{i,j}). \tag{15}$$

Given the $z_{i,j}$'s we can rewrite B_j as $B_j \Leftrightarrow \bigwedge_{i\in[\gamma]} z_{i,j}$. Using Tseitin's decomposition a second time, we can rewrite this equivalence as the conjunction of the following three clauses, for all $j \in [n]$,

$$z_{1,j} \vee \ldots \vee z_{\gamma,j} \vee \overline{B}_j \quad \text{and} \quad \mathsf{AMO}(z_{1,j} \vee \ldots \vee z_{\gamma,j}) \vee \overline{B}_j \quad \text{and} \quad \bigwedge_{i\in[\gamma]} \overline{z}_{i,j} \vee B_j. \tag{16}$$

Finally, the resulting CNF is the conjunction of the clauses in Equations 14, 15 and 16, for all $i \in [\gamma]$ and $j \in [n]$.

Attack	Target	Leakage Profile	Auxiliary Data		Assumptions		Avg. Runtime
			Queries	Data	Queries	Data	
Count [23]	K	co	–	Partial	Non-rep.	–	< 1s
Subgraph-ID [14]	K	rid	–	Partial	Non-rep.	–	< 1s
IHOP [86]	K	qeq	**P**	-	-	-	< 1s
Decoding-Bin [60]	K	qeq	**P**	-	-	-	< 1s
LMP-ID [76]	RV	rid	–	–	–	Dense	< 1s
LMP-APP [76]	RV	rid	–	–	–	Dense	< 1s
GenKKNO [52]	RV	rid	–	–	Uniform	–	< 1s
ApproxValue [52]	RV	rid	–	–	Uniform	Specific	< 1s
GLMP [51]	RC	rlen	–	–	$\mathcal{A}$	–	< 1s
GJW-Basic [53]	RC	rlen	B	–	$\mathcal{A}_B$	–	DNR
GJW-Missing [53]	RC	rlen	B,k	–	$\mathcal{A}_{B,k}$	–	DNR

Table 2: List of attacks in evaluation. The target is either Keyword query (K) or Range (Value/Count, RV/RC) reconstruction. B is the maximum width and k the amount of missing queries per width. $\mathcal{A}$ denotes that all possible response lengths occur (only within all widths $\leq B$ for $\mathcal{A}_B$, or k missing therein for $\mathcal{A}_{B,k}$). **P** represents a known Markov transition matrix. The Avg. Runtime represents the average time the attack takes to run across all trials in the evaluation. We use < 1s to represent the fact that the attacks run on the order of ms, and DNR means that the attack fails on the parameter regime considered in this work.

7 Evaluation

We implement the `PolySys` engine in Python and make use of `cadical` [13] as the underlying SAT-solver when using `PolySys` as a resolution or known-data attack, and the `Google OR-tools` [89] as the underlying optimization problem library when using `PolySys` as an inference attack. `PolySys` is written in 3,700 lines of code. For comparison, we use the Python implementations of prior attacks in LEAKER [61].

Setup. We evaluate `PolySys` on an AWS `m6i.2x` large instance with 32GB of memory. All runs are performed on a single thread, and the results are reported as an average over 10 trials along with the 25th and 75th percentiles.

Comparison. We compare `PolySys` against state-of-art point and range attacks as described in Section 2. Table 2 lists all attacks we considered and their runtimes. The attacks are evaluated against real-world datasets, which are described below. Additionally, in cases where an attack cannot be evaluated on real-world data due to some assumption on data distribution or scalability limitations, we compare the attacks on synthetic datasets. We assess the set of attacks on both uniform and Zipf-distributed query sequences, with $\alpha = 2$. For known data attacks (Figure 1), we vary the known data rate where we uniformly sample the data records known to the adversary. For other attacks, we vary the number of queries t to demonstrate the efficacy of the attacks.

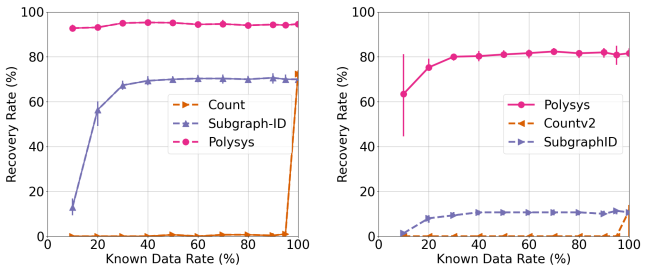
Datasets. We evaluate the attacks on two real-world datasets: the Enron dataset [1] for keyword attacks and the MIMIC dataset [57] for range attacks:

- Enron [1]: *Enron* is an email dataset for various employees of the Enron Corporation. We use the emails of allen_p and select the top 200 most common keywords to create a dataset for the evaluation. Selecting the top 200 keywords creates a dataset of 602 emails.
- MIMIC [57]: *Medical Information Mart for Intensive Care* is an open-source medical dataset that contains patient readings of medical metrics. In this evaluation, we use the MIMIC-T4 dataset, which measures patients' free thyroxine levels. We use the first 500 readings for the sake of this evaluation. This creates a sparse dataset with domain size $N = 64$.

7.1 Attacks Against Point Multi-Maps

We evaluated `PolySys` as a query recovery attack in two settings: first, as a known data attack against the response identity pattern rid; and then, as an inference attack against the query equality pattern qeq. For both these cases, `PolySys` *significantly* outperforms all existing attacks, sometimes with a recovery rate $60\times$ higher than the best-known attack.

Known data attacks against rid**.** Figure 1a and Figure 1b show the recovery rates of `PolySys`, Count, and Subgraph-ID attacks when varying the known data rate on the Enron dataset. We also consider that user queries are sampled uniformly or following a Zipf distribution. The query sequence length was set to 150, and the recovery rate is computed as the fraction of queries (out of 150) that are correctly matched. Note that



(a) Uniform (b) Zipf($\alpha = 2$)

Figure 1: Recovery rates for known data attacks against rid on the Enron database for a query sequence of length $t = 150$.

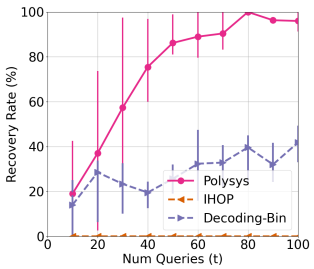


Figure 2: Attack against qeq with Markovian queries.

although the Count attack uses the co-occurrence pattern, this is implied via the response identity pattern.

PolySys outperforms all other attacks in low and high known data rates. Even with a 10% known data rate, PolySys achieves a recovery rate of 92% and 79% with uniform and Zipf-distributed queries respectively, whereas Subgraph-ID has a low recovery rate of 9% and 1%. This nontrivial gap is probably because PolySys does exceptionally well in extracting all the information from the leakage as it fully describes it as a system of equations. In contrast, the Subgraph-ID attack and (even worse) the Count attacks are custom attacks that do not leverage all leaked information. We note that the recovery rate of PolySys (and all other attacks) dropped when the query distribution was changed from uniform to Zipf.

Inference attack against qeq**.** Figure 2 shows the recovery rates of PolySys, IHOP and Decoding-Bin attacks when varying the lenght of the query sequence. For these attacks, we assume that the query sequence was sampled according to a Markov process with transition matrix $\mathbf{P}$ and original distribution μ. We also consider the setting where the adversary has exact knowledge of the stochastic process, i.e., both $\mathbf{P}$ and μ. Due to scalability limitations, we evaluate the attacks on a small synthetic multi-map with a number of labels equal to $m = 10$. Every row in the transition matrix $\mathbf{P}$ is equal to a random permutation of the pmf of a Zipf with parameter 2, and the initial distribution is uniform, i.e., $\mu = (1/m, \ldots, 1/m)$.

PolySys significantly outperforms IHOP and Decoding-Bin and achieves nearly perfect query reconstruction for a query sequence of length 100. In contrast, IHOP's recovery is extremely low, whereas Decoding-Bin can only recover a small percentage. We recognize that a downside of PolySys is its scalability, which we discuss at the end of this section.

7.2 Attacks against Range Multi-Maps

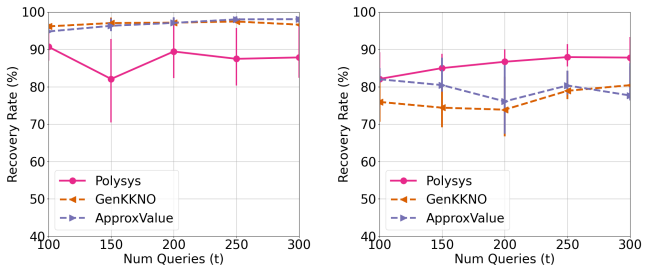
In this section, we evaluate PolySys as a data resolution attack against the response identity pattern and as a count resolution attack against the response length pattern, where the latter only attempts to recover the number of records/values that map to each numerical value in the range multi-map. As above, PolySys outperforms all known range attacks in all cases, with the exception of one case we discuss below.

Data resolution against rid**.** Figure 3a and Figure 3b show the recovery rate of the GenKKNO, ApproxValue, LMP-ID, and PolySys on the MIMIC dataset for uniform and Zipf-distributed queries, respectively. The recovery rate is the mean absolute error accounting for reflection. As expected, PolySys slightly underperforms prior work in the uniform case while outperforming them in the Zipf case as the number of queries increases. The slight underperformance in the uniform case is due to hardcoded distributional knowledge in the attacks; an assumption PolySys does not use. Specifically, GenKKNO and ApproxValue were designed assuming the queries were uniformly distributed. This explains the difference in performance between the uniform and Zipf case.

Because LMP-ID and LMP-APP are only usable in the case where the dataset is dense (e.g., there exists at least one record for each value in the domain), we additionally test PolySys and the other attacks on a synthetic dense dataset for the same size as the MIMIC dataset ($N = 64,\ v = 500$). We do this by fixing N records/values to be each value in the domain $[N]$ and then uniformly sampling the remainder. The results of the attacks for uniform and Zipf-distributed query sequences on this synthetic dataset are shown in Figure 3c and Figure 3d. In the dense case, PolySys again outperforms all prior work. In particular, PolySys can obtain $> 99\%$ data reconstruction and $> 85\%$ reconstruction rate in the uniform and Zipf case, respectively, with 300 queries. Similar to the point attacks, varying the query distribution and, in this case, the data distribution can impact the recovery rate, as illustrated in Figure 3.

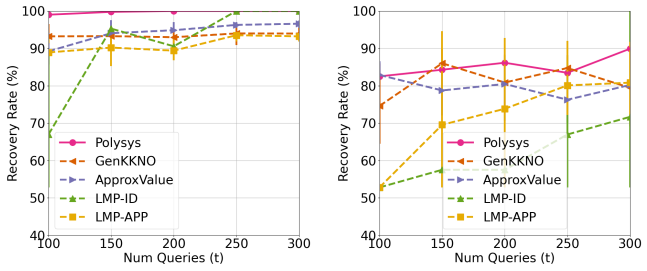
Count resolution against rlen**.** Figure 4a and Figure 4b show the recovery rate of the GLMP attack and PolySys on the MIMIC dataset for uniform and Zipf-distributed query sequences, respectively. We note that the GJW attacks are missing from the figures because they do not successfully run on the MIMIC dataset due to infeasible search space size.

PolySys outperforms GLMP in both scenarios. This is because GLMP assumes that all response lengths are observed in



(a) MIMIC, Uniform

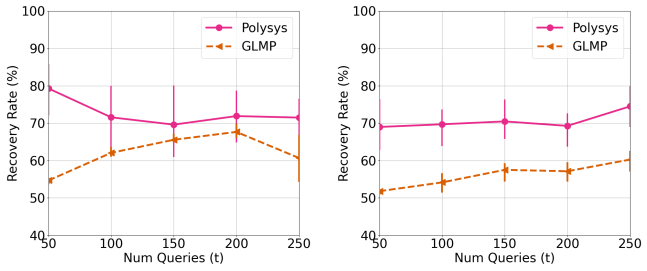
(b) MIMIC, Zipf($\alpha = 2$)



(c) Dense, Uniform

(d) Dense, Zipf($\alpha = 2$)

Figure 3: Recovery rates for data resolution attacks against rid for both the MIMIC dataset and a dense synthetic dataset.



(a) Uniform

(b) Zipf($\alpha = 2$)

Figure 4: Recovery rates for data resolution attacks against rlen for the MIMIC database.

the query sequence, which might not necessarily be true. Note that we compare PolySys and GLMP on histogram reconstruction because GLMP is unable to recover the underlying values, while PolySys has the capability to do so.

8 Limitations

Scalability. The scalability limitation arises in two places: the runtime of the attacks and the number of variables/constraints required to represent the SAT program. Table 3 shows the average runtime and the average program size for the various attacks considered in this work. As seen in Table 3, the runtime of PolySys is on the order of seconds/minutes, while the specialized attacks in prior work run much more quickly (on the order of ms - see Table 2). This is due to the fact that PolySys is a much more general framework and that prior works are often specially tailored a specific leakage pattern. Despite this, we believe that insights provided by PolySys are sufficient to draw meaningful conclusions.

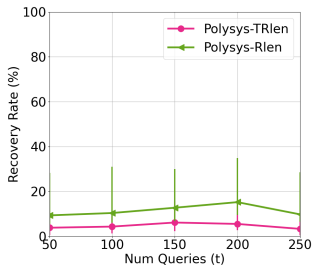


Figure 5: Recovery rates for count resolution attacks against rlen and trlen for a dense, synthetic database.

First, a high recovery rate against a given leakage pattern under specific attack settings that increases with the size of the data and parameters is an indication that the leakage profile might not be secure enough for that setting—even if the results are for small datasets and parameters. Conversely, if the recovery rate is low, it may suggest that the leakage is more challenging to exploit.

Extending to more secure patterns. It is important to evaluate attacks in settings where they may not do well. To this end, we evaluated PolySys against the response length (also known as volume) and the total response length which, intuitively, seem very hard to exploit. Note that we consider the *easier* setting of a count resolution attack against dense uniformly-sampled data (as opposed to sparse data) with a domain size of $N = 100$. Table 3 presents PolySys' recovery rate. One can see that it recovers at most 15% of the data when attacking rlen and at most 6% when attacking trlen.

Leakage Pattern	Type	Avg. Runtimes	# Vars	# Clauses (CNF)
rid	point	2.3s – 9.9s	0.45m – 2.65m	4.73m – 15.81m
	range	49.9s – 280.8s	1.26m – 2.25m	5.08m – 8.06m
qeq*	point	14.0s – 497.5s	6.7k – 42.7k	19.6k – 129.0k
rlen	point	29.9s – 65.9s	0.84m – 1.67m	2.25m – 6.83m
	range	141.6s – 3283.0s	0.26m – 0.52m	1.05m – 2.09m
trlen	point	4.6s – 10.2s	0.23m – 0.45m	0.93m – 2.29m

Table 3: Middle quartile range for runtime, number of CNF variables, and number of CNF constraints across all runs for the attacks evaluated in this work. *Note that the attack against qeq is an inference attack that makes use of `PolySys-AUX`, which is implemented using an SMT solver rather than a SAT solver, hence the lower number of variables and clauses.

9 Open Science

The artifact used in the experimental evaluation is available on Zenodo at [37].

10 Ethics Considerations

The evaluation in our paper makes use of two public datasets that potentially contain sensitive information (e.g. email logs, medical history) of the original parties in order to evaluate the efficacy of our encrypted search attacks.

The first dataset is Enron and is publicly available online and is explicitly made available for academic purposes. The second data is MIMIC which is a medical history dataset. Use of the MIMIC dataset is access-controlled and was only handled via approved authors who completed the required training and data use agreements laid out on the corresponding website [57].

Before the datasets were used in our experiments, all of the information was tokenized and anonymized, since the only information required for our work are the keywords and the numerical values of the data entries. We stress that we did not attempt to de-anonymize any of the relevant parties in the used datasets.

Our research builds an attack framework for analyzing encrypted search schemes on real-world datasets. While this furthers the landscape of attacks on encrypted search databases, we note that this could have a potential negative impact because it could be theoretically used by an attacker to recover sensitive information from a confidential dataset. However, we believe the benefits from advancing the encrypted search attack landscape outweighs the small risk of an attacker using this framework in practice.

References

[1] Enron Email Dataset. https://www.cs.cmu.edu/~enron/.

[2] Sympy. https://www.sympy.org/en/index.html.

[3] Archita Agarwal and Zachary Espiritu. Sequentially consistent concurrent encrypted multimaps. In *Proceedings of the 10th IEEE European Symposium on Security and Privacy*. IEEE, 2025.

[4] Archita Agarwal, Seny Kamara, and Tarik Moataz. Concurrent encrypted multimaps. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 169–201. Springer, 2024.

[5] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order preserving encryption for numeric data. In *ACM SIGMOD International Conference on Management of Data*, pages 563–574, 2004.

[6] Ghous Amjad, Seny Kamara, and Tarik Moataz. Injection-secure structured and searchable symmetric encryption. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023*, pages 232–262, Singapore, 2023. Springer Nature Singapore.

[7] G. Asharov, M. Naor, G. Segev, and I. Shahaf. Searchable symmetric encryption: Optimal locality in linear space via two-dimensional balanced allocations. In *ACM Symposium on Theory of Computing (STOC '16)*, STOC '16, pages 1101–1114, New York, NY, USA, 2016. ACM.

[8] Léonard Assouline and Brice Minaud. Weighted oblivious ram, with applications to searchable symmetric encryption. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023*, volume 14004 of *Lecture Notes in Computer Science*, pages 426–455. Springer, 2023.

[9] Michael Backes, Boris Köpf, and Andrey Rybalchenko. Automatic discovery and quantification of information leaks. In *Proceedings of the 2009 IEEE Symposium on Security and Privacy (S&P)*, pages 141–153. IEEE Computer Society, 2009.

[10] Amine Bahi, Seny Kamara, Tarik Moataz, and Guevara Noubir. Subliminal encrypted multi-maps and blackbox leakage absorption. Cryptology ePrint Archive, Paper 2024/1708, 2024.

[11] Gabrielle Beck, Maximilian Zinkus, and Matthew Green. Automating the development of chosen ciphertext attacks. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1821–1837, 2020.

[12] Jérémy Berthomieu, Christian Eder, and Mohab Safey El Din. Msolve: A library for solving polynomial systems. In *Proceedings of the 2021 International Symposium on Symbolic and Algebraic Computation*, pages 51–58, 2021.

[13] Armin Biere, Tobias Faller, Katalin Fazekas, Mathias Fleury, Nils Froleyks, and Florian Pollitt. CaDiCaL 2.0. In *Computer Aided Verification - 36th International Conference*, volume 14681 of *Lecture Notes in Computer Science*, pages 133–152. Springer, 2024.

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

[15] Tobias Boelter, Rishabh Poddar, and Raluca Ada Popa. A secure one-roundtrip index for range queries. Cryptology ePrint Archive, Paper 2016/568, 2016.

[16] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. Order-preserving symmetric encryption. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009*, pages 224–241, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[17] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system I: The user language. *Journal of Symbolic Computation*, 24(3–4):235–265, 1997.

[18] Angèle Bossuat, Raphael Bost, Pierre-Alain Fouque, Brice Minaud, and Michael Reichle. SSE and SSD: page-efficient searchable symmetric encryption. In *Advances in Cryptology–CRYPTO 2021*, pages 157–184. Springer, 2021.

[19] R. Bost. Sophos - forward secure searchable encryption. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016.

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

[21] Théophile Brézot and Chloé Hébant. Findex: A Concurrent and Database-Independent Searchable Encryption Scheme, 2024. Cryptology ePrint Archive, Paper 2024/1541.

[22] Bruno Buchberger. An algorithm for finding the basis elements of the residue class ring of a zero-dimensional polynomial ideal. *PhD Thesis, University of Innsbruck*, 1965.

[23] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart. Leakage-abuse attacks against searchable encryption. In *ACM Conference on Communications and Computer Security (CCS '15)*, pages 668–679. ACM, 2015.

[24] 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.

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

[26] 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.

[27] David Cash, Ruth Ng, and Adam Rivkin. Improved structured encryption for sql databases via hybrid indexing. In *Applied Cryptography and Network Security: 19th International Conference, ACNS 2021, Kamakura, Japan, June 21–24, 2021, Proceedings, Part II*, pages 480–510. Springer, 2021.

[28] Y. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *Applied Cryptography and Network Security (ACNS '05)*, volume 3531 of *Lecture Notes in Computer Science*, pages 442–455. Springer, 2005.

[29] 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.

[30] David Clark, Sebastian Hunt, and Pasquale Malacaria. Quantitative analysis of the leakage of confidential data. In *Proceedings of the 2002 Workshop on Specification and Verification of Component-Based Systems (SAVCBS)*, 2002.

[31] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 79–88. ACM, 2006.

[32] Wolfram Decker, Gert-Martin Greuel, Gerhard Pfister, and Hans Schönemann. SINGULAR 4-4-0 — A computer algebra system for polynomial computations. http://www.singular.uni-kl.de, 2024.

[33] Ioannis Demertzis, Javad Ghareh Chamani, Dimitrios Papadopoulos, and Charalampos Papamanthou. Dynamic searchable encryption with small client storage. In *Proceedings of the Network and Distributed System Security Symposium 2019*. Internet Society, 2019.

[34] Ioannis Demertzis, Stavros Papadopoulos, Odysseas Papapetrou, Antonios Deligiannakis, and Minos Garofalakis. Practical private range search revisited. In *Proceedings of the 2016 International Conference on Management of Data*, pages 185–198, 2016.

[35] Zachary Espiritu, Marilyn George, Seny Kamara, and Lucy Qin. Synq: Public Policy Analytics Over Encrypted Data. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 146–165, San Francisco, CA, USA, May 2024. IEEE.

[36] Zachary Espiritu, Seny Kamara, and Tarik Moataz. Bayesian Leakage Analysis. *IACR Communications in Cryptology*, 2(1), April 2025.

[37] Zachary Espiritu, Seny Kamara, Tarik Moataz, and Andrew Park. Artifact for "PolySys: an algebraic leakage attack engine". http://doi.org/10.5281/zenodo.15610354.

[38] Sky Faber, Stanislaw Jarecki, Hugo Krawczyk, Quan Nguyen, Marcel Rosu, and Michael Steiner. Rich queries on encrypted data: Beyond exact matches. In *Computer Security–ESORICS 2015: 20th European Symposium on Research in Computer Security*, pages 123–145. Springer, 2015.

[39] Francesca Falzon, Evangelia Anna Markatou, Zachary Espiritu, and Roberto Tamassia. Range search over encrypted multi-attribute data. *Proceedings of the VLDB Endowment*, 16(4):587–600, December 2022.

[40] Jean-Charles Faugere. A new efficient algorithm for computing gröbner bases (f4). *Journal of pure and applied algebra*, 139(1-3):61–88, 1999.

[41] Jean Charles Faugere. A new efficient algorithm for computing gröbner bases without reduction to zero (f 5). In *Proceedings of the 2002 international symposium on Symbolic and algebraic computation*, pages 75–83, 2002.

[42] Alan M Frisch and Paul A Giannaros. Sat encodings of the at-most-k constraint: Some old, some new, some fast, some slow. In *Proceedings of the International Workshop of Constraint Modeling and Reformulation*, 2010.

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

[44] C. Gentry. Fully homomorphic encryption using ideal lattices. In *ACM Symposium on Theory of Computing (STOC '09)*, pages 169–178. ACM Press, 2009.

[45] Marilyn George, Seny Kamra, and Tarik Moataz. Structured encryption and dynamic leakage suppression. In *Advances in Cryptology - EUROCRYPT 2021*, 2021.

[46] E-J. Goh. Secure indexes, 2003. Cryptology ePrint Archive, Paper 2003/216.

[47] 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.

[48] O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious RAMs. *Journal of the ACM*, 43(3):431–473, 1996.

[49] Carla P. Gomes, Ashish Sabharwal, and Bart Selman. Model counting: A new strategy for obtaining good bounds. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1 (AAAI'06)*, pages 54–61. AAAI Press, 2006.

[50] Daniel R. Grayson and Michael E. Stillman. Macaulay2, a software system for research in algebraic geometry. http://www2.macaulay2.com.

[51] Paul Grubbs, Marie-Sarah Lacharité, Brice Minaud, and Kenneth G. Paterson. Pump up the volume: Practical database reconstruction from volume leakage on range queries. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 315–331. ACM, 2018.

[52] Paul Grubbs, Marie-Sarah Lacharité, Brice Minaud, and Kenneth G. Paterson. Learning to reconstruct: Statistical learning theory and encrypted database attacks. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 1067–1083. IEEE, 2019.

[53] Zichen Gui, Oliver Johnson, and Bogdan Warinschi. Encrypted databases: New volume attacks against

range queries. In *Proceedings of the 2019 ACM Conference on Computer and Communications Security*, CCS '19, pages 361–378, New York, NY, USA, 2019. ACM.

[54] Jonathan Heusser and Pasquale Malacaria. Applied quantitative information flow and statistical databases. In *Formal Aspects in Security and Trust*, volume 5983 of *Lecture Notes in Computer Science*, pages 96–110, Berlin, Heidelberg, 2010. Springer-Verlag.

[55] Jonathan Heusser and Pasquale Malacaria. Quantifying information leaks in software. In *Proceedings of the 26th Annual Computer Security Applications Conference*, pages 261–269. ACM, 2010.

[56] M. Saiful Islam, M. Kuzu, and M. Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *Network and Distributed System Security Symposium (NDSS '12)*, 2012.

[57] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-Wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

[58] Mireya Jurado and Geoffrey Smith. Quantifying information leakage of deterministic encryption. In *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop*, CCSW'19, pages 129–139, New York, NY, USA, 2019. ACM.

[59] Abdel Alim Kamal and Amr M. Youssef. Applications of SAT solvers to AES key recovery from decayed key schedule images, 2010. Cryptology ePrint Archive, Paper 2010/324.

[60] Seny Kamara, Abdelkarim Kati, Tarik Moataz, Jamie DeMaria, Andrew Park, and Amos Treiber. Maple: Markov process leakage attacks on encrypted search. *Proceedings on Privacy Enhancing Technologies*, 2024.

[61] Seny Kamara, Abdelkarim Kati, Tarik Moataz, Thomas Schneider, Amos Treiber, and Michael Yonli. Sok: Cryptanalysis of encrypted search with leaker–a framework for leakage attack evaluation on real-world data. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pages 90–108. IEEE, 2022.

[62] Seny Kamara and Tarik Moataz. Boolean searchable symmetric encryption with worst-case sub-linear complexity. In *Advances in Cryptology - EUROCRYPT '17*, 2017.

[63] Seny Kamara and Tarik Moataz. SQL on structurally-encrypted databases. In *Advances in Cryptology–ASIACRYPT 2018*, pages 149–180. Springer, 2018.

[64] Seny Kamara and Tarik Moataz. Computationally volume-hiding structured encryption. In *Advances in Cryptology - EUROCRYPT '19*, 2019.

[65] Seny Kamara and Tarik Moataz. Design and analysis of a stateless encrypted document database. Technical report, 2023. https://www.mongodb.com/resources/products/capabilities/stateless-document-database-encryption-scheme.

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

[67] Seny Kamara, Tarik Moataz, Andrew Park, and Lucy Qin. A decentralized and encrypted national gun registry. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1520–1537. IEEE, 2021.

[68] Seny Kamara and Charalampos Papamanthou. Parallel and dynamic searchable symmetric encryption. In *Financial Cryptography and Data Security*, 2013.

[69] Seny Kamara, Charalampos Papamanthou, and Tom Roeder. Dynamic searchable symmetric encryption. In *Proceedings of the ACM Conference on Computer and Communications Security*. ACM, 2012.

[70] G. Kellaris, G. Kollios, K. Nissim, and A. O' Neill. Generic attacks on secure outsourced databases. In *Proceedings of the ACM Conference on Computer and Communications Security*, 2016.

[71] Vladimir Klebanov. Precise quantitative information flow analysis using symbolic model counting. In Fabio Martinelli and Flemming Nielson, editors, *Proceedings of the International Workshop on Quantitative Aspects in Security Assurance (QASA)*, 2012. Extended version available as arXiv:1208.3800.

[72] Will Klieber and Gihwon Kwon. Efficient cnf encoding for selecting 1 from n objects. In *Proc. International Workshop on Constraints in Formal Verification*, page 14, 2007.

[73] Evgenios M Kornaropoulos, Charalampos Papamanthou, and Roberto Tamassia. The state of the uniform: Attacks on encrypted databases beyond the uniform query distribution. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1223–1240. IEEE, 2020.

[74] Evgenios M Kornaropoulos, Charalampos Papamanthou, and Roberto Tamassia. Response-hiding encrypted ranges: Revisiting security via parametrized leakage-abuse attacks. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1502–1519. IEEE, 2021.

[75] Marie-Sarah Lacharité, Brice Minaud, and Kenneth G. Paterson. Improved reconstruction attacks on encrypted data using range query leakage. *IACR Cryptology ePrint Archive*, 2017:701, 2017.

[76] Marie-Sarah Lacharité, Brice Minaud, and Kenneth G. Paterson. Improved reconstruction attacks on encrypted data using range query leakage. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 297–314. IEEE Computer Society, 2018.

[77] Kevin Lewi and David J. Wu. Order-revealing encryption: New constructions, applications, and lower bounds. In *Proceedings of the ACM Conference on Computer and Communications Security*, 2016.

[78] Maplesoft, a division of Waterloo Maple Inc. *Maple 2023*. Maplesoft, Waterloo, Ontario, 2023. https://www.maplesoft.com.

[79] Evangelia Anna Markatou, Francesca Falzon, Zachary Espiritu, and Roberto Tamassia. Attacks on Encrypted Response-Hiding Range Search Schemes in Multiple Dimensions. *Proceedings on Privacy Enhancing Technologies*, 2023(4):204–223, October 2023.

[80] Stephen McCamant and Michael D. Ernst. Quantitative information flow as network flow capacity. In *Proceedings of the 29th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pages 193–205. ACM, 2008.

[81] Zhendong Meng and Geoffrey Smith. Calculating bounds on information leakage using two-bit patterns. In *Proceedings of the 2011 ACM Workshop on Programming Languages and Analysis for Security (PLAS)*, pages 1–12. ACM, 2011.

[82] Brice Minaud and Michael Reichle. Hermes: I/o-efficient forward-secure searchable symmetric encryption. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023*, pages 263–294, Singapore, 2023. Springer Nature Singapore.

[83] Ilya Mironov and Lintao Zhang. Applications of sat solvers to cryptanalysis of hash functions. In *Proceedings of the International Conference on Theory and Applications of Satisfiability Testing (SAT 2006)*, volume 4121 of *Lecture Notes in Computer Science*, pages 102–115. Springer, August 2006.

[84] Priyanka Mondal, Javad Ghareh Chamani, Ioannis Demertzis, and Dimitrios Papadopoulos. I/O-Efficient dynamic searchable encryption meets forward & backward privacy. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 2527–2544, Philadelphia, PA, August 2024. USENIX Association.

[85] James Newsome, Stephen McCamant, and Dawn Song. Measuring channel capacity to distinguish undue influence. In *Proceedings of the 2009 ACM Workshop on Programming Languages and Analysis for Security (PLAS)*, pages 73–85, New York, NY, USA, 2009. ACM.

[86] Simon Oya and Florian Kerschbaum. Hiding the access pattern is not enough: Exploiting search pattern leakage in searchable encryption. In *Proceedings of the 30th USENIX Security Symposium*, pages 127–142, 2021.

[87] Simon Oya and Florian Kerschbaum. IHOP: Improved statistical query recovery against searchable symmetric encryption through quadratic optimization. In *Proceedings of the 31st USENIX Security Symposium*, pages 2407–2424, 2022.

[88] Sarvar Patel, Giuseppe Persiano, Joon Young Seo, and Kevin Yeo. Efficient boolean search over encrypted data with reduced leakage. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 577–607. Springer, 2021.

[89] Laurent Perron and Frédéric Didier. Cp-sat.

[90] Quang-Sang Phan, Pasquale Malacaria, Olga Tkachuk, and Corina S. Păsăreanu. Symbolic quantitative information flow. In Peter Mehlitz, Nitin Rungta, and Willem Visser, editors, *Proceedings of the Java Pathfinder Workshop*, pages 1–5, 2012.

[91] Vaibhav Rastogi, Andrew C. Myers, and Cormac Flanagan. Knowledge inference for optimizing secure multiparty computation. In *Proceedings of the 20th ACM Conference on Computer and Communications Security (CCS)*, pages 521–532. ACM, 2013.

[92] Daniel S Roche, Daniel Apon, Seung Geol Choi, and Arkady Yerukhimovich. Pope: Partial order preserving encoding. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1131–1142, 2016.

[93] Leah Namisa Rosenbloom. *Cryptography for Grassroots Organizing*. PhD thesis, Brown University, May 2024.

[94] Bjarke Hammersholt Roune and Michael Stillman. Practical gröbner basis computation. In *Proceedings of the 37th International Symposium on Symbolic and Algebraic Computation*, pages 203–210, 2012.

[95] 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.

[96] Mate Soos, Karsten Nohl, and Claude Castelluccia. Extending sat solvers to cryptographic problems. In *Proceedings of the International Conference on Theory and Applications of Satisfiability Testing (SAT 2009)*, pages 244–257, June 2009.

[97] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 10.2)*. The Sage Development Team, 2023. https://www.sagemath.org.

[98] Grigori S Tseitin. On the complexity of derivation in propositional calculus. *Automation of reasoning: 2: Classical papers on computational logic 1967–1970*, pages 466–483, 1983.

[99] Andreas Weigl. Efficient sat-based pre-image enumeration for quantitative information flow in programs. In *Hardware and Software: Verification and Testing (HVC)*, volume 10028 of *Lecture Notes in Computer Science*, pages 294–310. Springer, 2016.

[100] Wolfram Research, Inc. *Mathematica, Version 13.3*. Wolfram Research, Inc., Champaign, Illinois, 2023. https://www.wolfram.com/mathematica.

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

[102] Y. Zhang, J. Katz, and C. Papamanthou. All your queries are belong to us: The power of file-injection attacks on searchable encryption. In *USENIX Security Symposium*, 2016.

[103] Shengjia Zhao, Sorathan Chaturapruek, Ashish Sabharwal, and Stefano Ermon. Closing the gap between short and long XORs for model counting. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI 2016)*, December 2016.

[104] Maximilian Zinkus, Yinzhi Cao, and Matthew D Green. McFIL: Model counting functionality-inherent leakage. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 7001–7018, 2023.