2024 IEEE 44th International Conference on Distributed Computing Systems Workshops (ICDCSW) | 979-8-3503-5471-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICDCSW63686.2024.00007

Tutorial: Information Leakage from Cryptographic Techniques

Komal Kumari,[1] Sharad Mehrotra,[2] and Shantanu Sharma[1]
[1]New Jersey Institute of Technology, USA. [2]University of California, Irvine, USA.

***Abstract*—This tutorial focuses on the research aimed at providing secure data processing at the public cloud. First, we focus on cryptographic (encryption and secret-sharing) techniques and systems developed over the last two decades. Second, we will discuss information leakages from ciphertext (*e.g*., distribution, ordering, and cross-crypto leakages) and query execution (*e.g*. access pattern, volume, and workload skew leakages) and compare the existing techniques and systems based on efficiency and information leakage. Finally, we conclude that cryptographic techniques are not sufficient alone. To provide efficient and secure large-scale data processing at the cloud, a new line of work that combines software and hardware mechanisms is required. We discuss an orthogonal approach designed around the concept of data partitioning, i.e., splitting the data processing into cryptographically secure and non-secure parts.**

I. Goal of the Tutorial

This tutorial delves into three key areas of research aimed at ensuring the secure processing of databases at the public cloud. We will divide this tutorial into three phases as follows:

1) We discuss cryptographic (encryption and secret-sharing) techniques and systems developed over the past two decades and compare these approaches/systems based on their efficiency and operational capabilities. We will discuss encryption-based systems, such as CryptDB [1], HE3DB [2], MongoDB's Queryable Encryption [3], Microsoft Always Encrypted [4], and secret-sharing based systems including Secrecy [5], S^2 [6], Titanium [7], and Obscure [6].
2) We, then, discuss different types of information leakage from ciphertext itself and from query execution. We will illustrate how such leakages can lead to revealing the entire database in cleartext to an adversary. Furthermore, we will discuss techniques to prevent such attacks and compare existing cryptographic techniques and systems against such leakages.
3) Finally, we discuss two new approaches to secure data processing. The first approach will discuss data partitioning methods, where only sensitive data will be encrypted, and non-sensitive data will remain in cleartext. The second approach will discuss how we can use multiple encryption techniques over a table without revealing any additional information to an adversary, using a new concept called secure normal form. Finally, we will conclude the tutorial by showing open problems.

Outcome. Secure data processing is an integral aspect of data management at the cloud. The cloud environment offers unique challenges in implementing security and scalability of data. The tutorial is relevant to the researchers working on security issues in data management. Furthermore, it introduces state-of-the-art technologies to practitioners in protecting the data, while using the cloud for their data management.

Intended audience and duration. Researchers, students, developers, and practitioners interested in data security should be benefited. We cover content for different audiences, as 10% beginner, 40% intermediate, and 50% advanced.

II. Cryptographic Techniques for Data Processing

The rapid rise of cloud technology for data storage and computing has revolutionized the digital landscape. Cloud providers could be located anywhere, under varying legal jurisdictions with varying legal protections; the privacy and confidentiality of the outsourced data can be compromised, thus making it hard to establish trust in the cloud providers. Loss of control over resources (as outsourced to the cloud) coupled with the lack of trust (in the service provider) poses numerous concerns about data integrity, availability, security, privacy, and confidentiality, to mention a few. The problem of trust has become even more profound now given data breaches (HBO [8], Dell [9], Santander Bank [10], Nissan [11] and Equifax [12] data breaches).

To provide security and privacy to client's outsourced data, secure computing emerged as a crucial focus for leveraging cloud services, leading to the development of numerous cryptographic approaches broadly classified into two categories:

- **Encryption-based techniques.** Several encryption-based techniques have been proposed depending on the desired operations. *E.g*., order-preserving encryption (OPE) [13] offers range queries, deterministic encryption (DET) [14] offers equality queries, homomorphic encryption [15] offers addition and/or multiplication over ciphertext, searchable encryption [16], [17] offers search operation over ciphertext, bucketization [18] offers full support for SQL queries, and non-deterministic encryption (NDET) [19], which does not offer any operation over ciphertext.

These techniques have led to the development of several systems — CryptDB [1] using OPE, DET, and NDET, HE3DB [2] using homomorphic encryption, MongoDB [3] using a variant of searchable encryption, Microsoft Always Encrypted [4] using DET and NDET, Cypherbase [20] and Vaultree [21] using fully homomorphic encryption. Industrial systems, such as Oracle 12c, Amazon Aurora [22], and MariaDB [23], offer encryption at-rest, but do not offer processing over ciphertext.

- **Secret-sharing (SS) based techniques.** In using SS, the database owner divides a secret value, say S, into c different fragments, called *shares*, and sends each share to a set of c non-colluding servers. To reconstruct the secret, a client needs to collect a subset of c fragments from the server.

979-8-3503-5471-3/24/$31.00 ©2024 IEEE
DOI 10.1109/ICDCSW63686.2024.00007

One of the famous secret-sharing techniques was proposed by Adi Shamir [24]. In Shamir's secret-sharing (SSS) technique [24], the database owner randomly selects a polynomial of degree c' with c' random coefficients, *i.e.*, $f(x) = a_0{+}a_1x{+}a_2x^2{+}\cdots{+}a_{c'}x^{c'}$, where $f(x){\in}\mathbb{F}_p[x]$, p is a prime number, $\mathbb{F}_p$ is a finite field of order p, $a_0{=}S$ (the secret), and $a_i{\in}\mathbb{N}$ ($1{\leq}i{\leq}c'$). The owner distributes S into $c{>}c'$ shares, by computing $f(x)$ for $x{=}1,2,\ldots,c$ and sends an i^{th} share to the i^{th} server. The secret, S, is reconstructed using Lagrange interpolation [25] over any $c'{+}1$ shares. An adversary can construct S, iff they collude with $c'{+}1$ servers. Thus, the degree of a polynomial is set to be c', if an adversary can collude with at most c' servers.

Additive SS [26] is another technique that divides a secret S into c shares such that the addition of all c shares over some modulo p regains S. Other SS techniques are Boolean SS [5], replicated SS [27], distributed point function [28], function SS [29], and accumulating-automata [6], [30].

These techniques have led to the development of systems, *e.g.*, Secrecy [5] using replicated SS, Obscure [6] using SSS, S^2 [6] using SSS and additive SS, Titanium [7] using additive SS, Sharemind [31], [32] using additive SS, Conclave [33] using addtive SS, PDAS [34] using SSS, Waldo [35] using replicated SS, Prism [6] using additive and SSS, Jana [36] using additive SS, and S3ORAM [37] using SSS.

Secure hardware. Secure data processing can also be achieved using trusted hardware, *e.g.*, Intel Software Guard Extensions (SGX) [38] and Trusted Execution Technology (TXT) [39] that allow the owner to create a small trusted execution environment called *enclaves*, which is isolated and protected from the rest of the cloud system. SGX provides encryption of the enclave's memory having the code and data, and the integrity is protected by the CPU as soon as the data leaves the last level of the caching hierarchy. This protects SGX applications from hardware attacks like memory snooping. Several systems including EnclaveDB [40], M2R [41], Opaque [42], ObliDB [43], StealthDB [44], VC3 [45], T-SGX [46], and Oblivate [47] are build using SGX.

Adversary model. Often, these techniques are developed under two types of adversarial cloud models, namely honest-but-curious (HBC) [48] and malicious clouds. Such adversaries have some background knowledge about the data, *e.g.*, data distribution. An HBC cloud, also known as a semi-honest cloud, behaves as per the system protocol; however, an HBC may try to obtain information about the sensitive data by observing the stored data and query execution. In contrast, a malicious adversary deviates from the underlying protocol and thus can tamper with the data stored or query processing.

Evaluation criteria. The above techniques and systems can be evaluated based on several criteria, such as the type of adversary they deal with, the number of used servers, supported operations, availability of indexes for query execution, dealing with malicious client, the use of a trusted proxy, and information leakages from the ciphertext and query execution.

III. Information leakages

The systems mentioned in §II strive to provide data security against respective adversarial models; however, some of them suffer from information leakages from data at-rest and during query processing [49]. Below, we discuss these leakages and present strategies to overcome leakages.

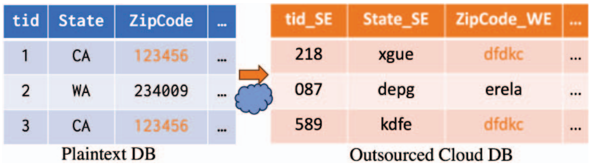
A. *Data at-rest leakages*

These leakages can be grouped into three types: (*i*) distribution leakage, (*ii*) ordering leakage, and (*iii*) cross-crypto leakage.

1) Distribution leakage: reveals data distribution from the ciphertext. Deterministic encryption [14], offering efficient execution of equality testing for selection and join queries, creates an identical ciphertext for more than two occurrences of a value and, hence, reveals data distribution from the ciphertext. In contrast, non-deterministic encryption or secret sharing do not reveal data distribution from ciphertext.

2) Ordering leakage: reveals the ordering (*e.g.*, $<$, $>$, $\leq$, $\geq$, $=$) of two or more ciphertexts. For example, if two values have a relationship in cleartext, say $x_1{<}x_2$, then ordering leakage will reveal the same relationship from the ciphertext, i.e., $E(x_1){<}E(x_2)$, where E refers to an encryption technique. Order-preserving encryption (OPE) [50], which offers efficient execution of range queries, reveals ordering information. [51]–[53] has shown that distribution leakage mixed with ordering leakage can reveal the entire data in cleartext to an adversary.

3) Cross-crypto leakage: occurs when using different encryption techniques on different parts of the data. Figure 1 shows a table with three columns, *e.g.*, tuple/row id (tid), State, and ZipCode; Suppose we encrypt tid and State columns with a strong encryption technique, such as non-deterministic encryption, and ZipCodes with a weaker encryption technique, such as deterministic encryption, to enable the equality test. NDET reveals nothing about ciphertext, while DET reveals data distribution. Thus, the distribution of the ZipCode column is revealed. Note that ZipCode and State columns are functionally dependent. An adversary server can thus learn more than what is allowed about State data, *e.g.*, the first and third rows have the same state. This toy example illustrates how functional dependency between columns can lead to additional leakage. A recent paper of ICDE 2024 [6] proposes a technique called secure normal form to partition a table into multiple tables such that a single table does not reveal any additional leakage from the ciphertext.



tid	State	ZipCode	...
1	CA	123456	...
2	WA	234009	...
3	CA	123456	...

Plaintext DB

tid_SE	State_SE	ZipCode_WE	...
218	xgue	dfdkc	...
087	depg	erela	...
589	kdfe	dfdkc	...

Outsourced Cloud DB

Figure 1: An example of showing cross-crypto leakage.

B. *Query processing leakages*

These leakages refer to the disclosure of information during query execution. Query processing leakages can be broadly classified into three categories: (*i*) access pattern leakage, (*ii*) workload leakage, and (*iii*) volume leakage.

1) Access pattern leakage: reveals the identity or the sequence in which data is accessed, *e.g*., the rows in the database, the file-ids, or the storage locations. Access pattern leakage also results in search pattern (or query pattern) leakage that allows an adversary to establish linkability among queries by identifying which queries in a sequence are the same or different [54]–[56]. An adversary observing the access patterns can potentially reconstruct the original search query and even the underlying plaintext data, effectively nullifying the purpose of storing the data in cipher form, as discussed in [57]–[61].

Techniques, *e.g*., oblivious random access memory (ORAM) [62]–[64] and its improved version — Path-ORAM [65], private information retrieval (PIR) [66], distributed point function (DPF) [28], and function secret-sharing (FSS) [29], hide access patterns.

Path ORAM organizes data blocks into a logical binary tree, where each block is randomly assigned to a specific path within the tree. The client stores the mapping between blocks and paths into a position map. For a single read/write operation, Path ORAM retrieves the entire path containing the desired data block at the client side and then rewrites the path by reassigning the fetched blocks to new, randomly chosen paths within the tree. Each request incurs a bandwidth cost of O($\log$ N), where N is the number of data blocks. Despite its conceptual simplicity, the overhead of O($\log$ N) makes Path ORAM impractical for many applications.Moreover, the throughput and support for concurrent clients are also limited in Path ORAM. Since the introduction of Path ORAM, works such as [67]–[74] have been carried out to address such limitations. Systems, *e.g*., Metal [75] and Titanium [7] use ORAM to build secure file systems.

PIR allows clients to fetch an item from outsourced data without revealing to the server which is the item of interest. The client creates a query for the server such that the query holds the identifier of the data to be retrieved. There are two types of PIR: computational PIR (CPIR) [76] and information-theoretic PIR (ITPIR) [77]. CPIR is secure under a bounded computational capabilities of an adversary, while ITPIR is secure regardless of the computational power of an adversary.

DPF allows the client to create a share of the query in the form of a point function and send shares to servers. These shares evaluate to one in share form when the query matches the data; otherwise zero in share form. S^2 [6] and Dory [78] use DPF. FSS [29] is a generalization of DPF.

Other techniques, different from previously discussed methods, have also been proposed, *e.g*., Durashift [79], Nemo [80], Waffle [81], Pancake [82], S3ORAM [37], and [83]–[85],

2) Workload-skew leakage: reveals an estimate of which encrypted tuples potentially satisfy the frequent selection queries to an adversary, knowing frequent selection queries. Except for access pattern hiding techniques, all cryptographic techniques are prone to workload-skew attacks. Panda [6] prevents workload attacks by creating bins over the data.

3) Volume or output-size leakage: reveals the number of records returned for a query. Volume leakage can be exploited by an adversary having knowledge about the number of records for each data element, as discussed in [6], [49], [60], [86]–[89]. A common strategy to prevent volume leakage is to return to the maximum number of records, say L, associated with an element to answer any query. With each query result padded to L, the result must ensure that the records satisfying the query are returned, together with the additional padded records, which can be discarded on the client side. Path ORAM [65] can prevent volume but will incur a significant cost by returning O($\log N \times L$) items, where N is the number of outsourced records. Veil [6], HybrIdx [90], and [91]–[97] hides volume. These techniques store more data at servers than the actual amount of data and fetch more than L elements to answer a query.

IV. Biographies

Komal Kumari is pursuing PhD degree in computer science at New Jersey Institute of Technology, USA. She obtained her MTech degree in computer science from Indraprastha Institute of Information Technology, Delhi, India, in 2021. Her primary research focus is secure data processing.

Sharad Mehrotra received his PhD degree in computer science from the University of Texas, Austin, in 1993. He is a distinguished professor with the Department of Computer Science, University of California, Irvine. Previously, he was a professor with the University of Illinois at Urbana Champaign. He has received numerous awards and honors, including the 2011 SIGMOD Best Paper Award, 2007 DASFAA Best Paper Award, SIGMOD test of time award, 2012, DASFAA ten year best paper awards for 2013 and 2014, 1998 CAREER Award from the US National Science Foundation (NSF), and ACM ICMR best paper award for 2013. His primary research interests include the area of database management, distributed systems, secure databases, and the Internet of Things. He is an IEEE Fellow and an ACM Fellow.

Shantanu Sharma received his PhD degree in computer science from Ben-Gurion University, Israel, in 2016. He is an assistant professor with the Department of Computer Science, New Jersey Institute of Technology, USA. Before joining NJIT, he worked as a postdoctoral fellow at UC Irvine. His research interests include secure and privacy-preserving database systems and trustworthy smart spaces.

Acknowledgement

The work of Shantanu Sharma was supported by NSF under Grant 2245374. The work of Sharad Mehrotra was supported by NSF under Grant 2420846, Grant 2245372, Grant 2133391, Grant 2008993, and Grant 1952247.

References

[1] R. A. Popa *et al.*, “CryptDB: processing queries on an encrypted database,” *CACM*, vol. 55, no. 9, pp. 103–111, 2012.

[2] S. Bian *et al.*, “HE3DB: An efficient and elastic encrypted database via arithmetic-and-logic fully homomorphic encryption,” in *CCS*, 2023.

[3] Mongo, “MongoDB Atlas: Cloud Document Database — MongoDB.” https://tinyurl.com/2rrfn88m.

[4] “Microsoft Always Encrypted.” https://tinyurl.com/2e5pcvep.

[5] J. Liagouris *et al.*, “Secrecy: Secure collaborative analytics in untrusted clouds,” in *NSDI 23*, pp. 1031–1056, 2023.

[6] “List of publications.” https://web.njit.edu/~ss797/publications.html.

[7] W. Chen *et al.*, “Titanium: A metadata-hiding file-sharing system with malicious security,” in *NDSS*, 2022.

[8] Available at: https://tinyurl.com/4nb984s6.

[9] PCWorld, “Dell hack: Personal info of 49 million customers allegedly breached — PCWorld.” https://tinyurl.com/yc4ebk3k.

[10] EM360, "Santander Customer Data Swiped in Cyber Attack — Enterprise Tech News EM360Tech." https://tinyurl.com/ycynjzks.

[11] CBSNews, "Nissan data breach exposed Social Security numbers of thousands of employees - CBS News." https://tinyurl.com/yc582bm8.

[12] Available at: https://tinyurl.com/4x6hvyp7.

[13] R. Agrawal *et al.*, "Order-preserving encryption for numeric data," in *SIGMOD*, pp. 563–574, 2004.

[14] M. Bellare *et al.*, "Deterministic and efficiently searchable encryption," in *CRYPTO*, pp. 535–552, 2007.

[15] C. Gentry, *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

[16] D. X. Song *et al.*, "Practical techniques for searches on encrypted data," in *S&P*, pp. 44–55, 2000.

[17] R. Curtmola *et al.*, "Searchable symmetric encryption: Improved definitions and efficient constructions," *JCS*, vol. 19, pp. 895–934, 2011.

[18] H. Hacigümüs *et al.*, "Executing SQL over encrypted data in the database-service-provider model," in *SIGMOD*, pp. 216–227, 2002.

[19] S. Goldwasser *et al.*, "Probabilistic encryption," *JCSS*, vol. 28, no. 2, pp. 270–299, 1984.

[20] A. Arasu *et al.*, "Orthogonal security with Cipherbase," in *CIDR*, 2013.

[21] "Data-in-use encryption - vaultree." https://www.vaultree.com/.

[22] Amazon Aurora, available at:https://aws.amazon.com/rds/aurora/.

[23] MariaDB, available at:https://mariadb.com/.

[24] A. Shamir, "How to share a secret," *CACM*, vol. 22, 1979.

[25] R. M. Corless *et al.*, "A graduate introduction to numerical methods," *AMC*, vol. 10, p. 12, 2013.

[26] R. Cramer *et al.*, *Secure multiparty computation*. Cambridge University Press, 2015.

[27] T. Araki *et al.*, "High-throughput semi-honest secure three-party computation with an honest majority," in *CCS*, pp. 805–817, 2016.

[28] N. Gilboa *et al.*, "Distributed point functions and their applications," in *EUROCRYPT*, pp. 640–658, 2014.

[29] E. Boyle *et al.*, "Function secret sharing," in *EUROCRYPT*, 2015.

[30] S. Dolev *et al.*, "Accumulating automata and cascaded equations automata for communicationless information theoretically secure multi-party computation," *TCS*, vol. 795, pp. 81 – 99, 2019.

[31] Cybernetica's Sharemind. Available at: https://tinyurl.com/2x6t8sat.

[32] D. Bogdanov *et al.*, "Sharemind: A framework for fast privacy-preserving computations," in *ESORICS*, vol. 5283, pp. 192–206, 2008.

[33] N. Volgushev *et al.*, "Conclave: secure multi-party computation on big data," in *EuroSys*, pp. 3:1–3:18, 2019.

[34] B. Thompson *et al.*, "Privacy-preserving computation and verification of aggregate queries on outsourced databases," in *PETS*, 2009.

[35] E. Dauterman *et al.*, "Waldo: A private time-series database from function secret sharing," in *S&P*, pp. 2450–2468, 2022.

[36] D. W. Archer *et al.*, "From keys to databases - real-world applications of secure multi-party computation," *IACR*, 2018.

[37] T. Hoang *et al.*, "S3ORAM: A computation-efficient and constant client bandwidth blowup oram with shamir secret sharing," in *CCS*, 2017.

[38] M. Hoekstra *et al.*, "Using innovative instructions to create trustworthy software solutions," in *Workshop on HASP*, pp. 1–8, 2013.

[39] "Intel® trusted execution technology hardware-based technology for enhancing server platform security." https://tinyurl.com/5b3mjf7b.

[40] C. Priebe *et al.*, "EnclaveDB: A secure database using SGX," in *S&P*, pp. 264–278, 2018.

[41] T. T. A. Dinh *et al.*, "M2R: enabling stronger privacy in mapreduce computation," in *USENIX*, pp. 447–462, 2015.

[42] W. Zheng *et al.*, "Opaque: An oblivious and encrypted distributed analytics platform," in *NSDI*, pp. 283–298, 2017.

[43] S. Eskandarian and M. Zaharia, "Oblidb: Oblivious query processing for secure databases," *VLDB*, vol. 13, no. 2, pp. 169–183, 2019.

[44] D. Vinayagamurthy *et al.*, "StealthDB: a scalable encrypted database with full SQL query support," *PoPETs*, pp. 370–388, 2019.

[45] F. Schuster *et al.*, "VC3: trustworthy data analytics in the cloud using SGX," in *S&P*, pp. 38–54, 2015.

[46] M.-W. Shih *et al.*, "T-SGX: Eradicating Controlled-Channel Attacks Against Enclave Programs.," in *NDSS*, 2017.

[47] A. Ahmad *et al.*, "OBLIVIATE: A data oblivious filesystem for Intel SGX.," in *NDSS*, 2018.

[48] R. Canetti *et al.*, "Adaptively secure multi-party computation," in *STOC*, pp. 639–648, 1996.

[49] G. Kellaris *et al.*, "Generic attacks on secure outsourced databases," in *CCS*, 2016.

[50] X. Cao *et al.*, "Frequency-revealing attacks against frequency-hiding order-preserving encryption," *VLDB*, vol. 16, no. 11, 2023.

[51] V. Bindschaedler *et al.*, "The tao of inference in privacy-protected databases," *IACR*, 2017.

[52] P. Grubbs *et al.*, "Leakage-abuse attacks against order-revealing encryption," in *S&P*, pp. 655–672, 2017.

[53] M. Naveed *et al.*, "Inference attacks on property-preserving encrypted databases," in *CCS*, pp. 644–655, 2015.

[54] S. Oya *et al.*, "Hiding the access pattern is not enough: Exploiting search pattern leakage in searchable encryption," in *USENIX Security*, 2021.

[55] E. M. Kornaropoulos *et al.*, "The state of the uniform: Attacks on encrypted databases beyond the uniform query distribution," in *S&P*, pp. 1223–1240, 2020.

[56] E. A. Markatou *et al.*, "Reconstructing with less: Leakage abuse attacks in two dimensions," in *CCS*, pp. 2243–2261, 2021.

[57] G. Kellaris *et al.*, "Generic attacks on secure outsourced databases," in *CCS*, pp. 1329–1340, 2016.

[58] M. S. Islam *et al.*, "Access pattern disclosure on searchable encryption: ramification, attack and mitigation.," in *NDSS*, vol. 20, p. 12, 2012.

[59] D. Cash *et al.*, "Leakage-abuse attacks against searchable encryption," in *CCS*, pp. 668–679, 2015.

[60] P. Grubbs *et al.*, "Pump up the volume: Practical database reconstruction from volume leakage on range queries," in *CCS*, 2018.

[61] P. Grubbs *et al.*, "Learning to reconstruct: Statistical learning theory and encrypted database attacks," in *S&P*, pp. 1067–1083, 2019.

[62] O. Goldreich, "Towards a theory of software protection and simulation by oblivious rams," in *STOC*, pp. 182–194, 1987.

[63] R. Ostrovsky, "Efficient computation on oblivious RAMs," in *STOC*, pp. 514–523, 1990.

[64] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious RAMs," *JACM*, vol. 43, no. 3, pp. 431–473, 1996.

[65] E. Stefanov *et al.*, "Path ORAM: An extremely simple oblivious RAM protocol," in *CCS*, pp. 299–310, 2013.

[66] B. Chor *et al.*, "Private information retrieval," *JACM*, vol. 45, no. 6, pp. 965–981, 1998.

[67] S. Devadas *et al.*, "Onion ORAM: A constant bandwidth blowup oblivious RAM," in *TCC*, pp. 145–174, 2016.

[68] A. Chakraborti and R. Sion, "ConcurORAM: High-throughput stateless parallel multi-client oram," *NDSS*, 2019.

[69] B. H. Falk *et al.*, "3-party distributed ORAM from oblivious set membership," in *SCN*, pp. 437–461, 2022.

[70] X. S. Wang *et al.*, "SCORAM: oblivious RAM for secure computation," in *CCS*, pp. 191–202, 2014.

[71] S. Patel *et al.*, "Panorama: Oblivious RAM with logarithmic overhead," in *FOCS*, pp. 871–882, 2018.

[72] X. Yu *et al.*, "PRO-ORAM: dynamic prefetcher for oblivious RAM," in *ISCA*, 2015.

[73] I. Komargodski *et al.*, "OptORAMa: optimal oblivious RAM," *JACM*, 2023.

[74] C. Sahin *et al.*, "Taostore: Overcoming asynchronicity in oblivious data storage," in *S&P*, pp. 198–217, 2016.

[75] W. Chen and R. A. Popa, "Metal: A metadata-hiding file-sharing system," in *NDSS*, 2020.

[76] B. Chor *et al.*, "Computationally private information retrieval (extended abstract)," in *STOC*, pp. 304–313, 1997.

[77] A. Beimel and Y. Stahl, "Robust information-theoretic private information retrieval," in *SCN*, vol. 2576 of *Lecture Notes in Computer Science*, pp. 326–341, Springer, 2002.

[78] E. Dauterman *et al.*, "DORY: An encrypted search system with distributed trust," in *OSDI*, pp. 1101–1119, 2020.

[79] B. H. Falk *et al.*, "Durasift: A robust, decentralized, encrypted database supporting private searches with complex policy controls," in *WPES*, pp. 26–36, 2019.

[80] J. Li *et al.*, "Nemo: Practical distributed boolean queries with minimal leakage," *TIFS*, 2024.

[81] S. Maiyya *et al.*, "Waffle: An online oblivious datastore for protecting data access patterns," *PACMMOD*, vol. 1, no. 4, pp. 1–25, 2023.

[82] P. Grubbs *et al.*, "Pancake: Frequency smoothing for encrypted data stores," in *USENIX Security*, pp. 2451–2468, 2020.

[83] L. Xu *et al.*, "Interpreting and mitigating leakage-abuse attacks in searchable symmetric encryption," *TIFS*, vol. 16, 2021.

[84] J. Ghareh Chamani *et al.*, "New constructions for forward and backward private symmetric searchable encryption," in *CCS*, 2018.

[85] S. Cui *et al.*, "Privacy-preserving dynamic symmetric searchable encryption with controllable leakage," *TOPS*, vol. 24, no. 3, 2021.

[86] R. Poddar *et al.*, "Practical volume-based attacks on encrypted databases," in *EuroS&P*, 2020.

[87] Z. Gui *et al.*, "Encrypted databases: New volume attacks against range queries," in *CCS*, 2019.

[88] J. Yao *et al.*, "Sok: A systematic study of attacks in efficient encrypted cloud data search," in *Workshop on SBC*, pp. 14–20, 2020.

[89] S. Lambregts *et al.*, "Val: Volume and access pattern leakage-abuse attack with leaked documents," in *ESORICS*, Springer, 2022.

[90] K. Ren *et al.*, "Hybridx: New hybrid index for volume-hiding range queries in data outsourcing services," in *ICDCS*, pp. 23–33, 2020.

[91] G. Amjad *et al.*, "Dynamic volume-hiding encrypted multi-maps with applications to searchable encryption," *PETS*, no. 1, pp. 417–436, 2023.

[92] S. Patel *et al.*, "Mitigating leakage in secure cloud-hosted data structures: Volume-hiding for multi-maps via hashing," in *CCS*, 2019.

[93] M. George *et al.*, "Structured encryption and dynamic leakage suppression," in *Eurocrypt*, pp. 370–396, 2021.

[94] S. Kamara *et al.*, "Computationally volume-hiding structured encryption," in *EUROCRYPT*, pp. 183–213, 2019.

[95] S. Kamara *et al.*, "Structured encryption and leakage suppression," in *CRYPTO*, pp. 339–370, 2018.

[96] J. Wang *et al.*, "Practical volume-hiding encrypted multi-maps with optimal overhead and beyond," in *CCS*, pp. 2825–2839, 2022.

[97] A. Bienstock *et al.*, "Near-optimal oblivious key-value stores for efficient PSI, PSU and volume-hiding multi-maps," in *USENIX Security*, pp. 301–318, 2023.