ISSN: 2366-1313

ENABLING EFFICIENT, SECURE AND PRIVACY-PRESERVING MOBILE CLOUD STORAGE

KALYANI INNAMOORI[1]

CHANDU DELHI POLICE[2]

[1]PG scholar, [2]Associate professor, Department of Computer Science Engineering
Priyadarshini Institute of Technology & Science, Tenali,Guntur

Abstract: Customers may access cloud storage services conveniently using Mobile Cloud Storage (MCS). In this research, we offer a mobile cloud storage method that is efficient, safe, and privacy-preserving. It ensures both data confidentiality and privacy at the same time, particularly the access pattern. More specifically, we suggest that the fundamental building block of the suggested mobile cloud storage system be an oblivious selection and update (OSU) protocol. By generating a small encrypted vector, OSU, which is based on onion additively homomorphic encryption with constant encryption layers, allows the client to silently retrieve an encrypted data item from the cloud and update it with a new value, thereby reducing communication overheads and client computation. Our work is more appropriate for MCS scenarios than prior efforts due to its useful qualities including constant communication overhead, lightweight client-side computation (a few additively homomorphic operations), and fine-grained data structure (small item size). Furthermore, our technique may be verified to withstand malicious cloud by using the "verification chunks" method. According to the comparison and assessment, our plan outperforms current blind storage options in terms of client and cloud workloads, respectively.

Index Terms: Mobile cloud storage, data security, privacy-preserving, efficient, malicious cloud server.

I. Introduction

Mobile cloud storage (MCS) allows users to access data stored on a cloud using mobile devices from any location. Owing to its alluring qualities, MCS is growing in popularity. Certain major corporations, including Apple iCloud, Dropbox, Microsoft OneDrive, and Google Drive, provide MCS services for commercial use. The cloud isn't always seen as completely trustworthy. As a result, the client may use encryption techniques to protect data before sending it to the cloud. Nonetheless, data in MSC-based applications is always connected to some other information, such location data in location-based services. The data item that is being retrieved in this case exposes more information to the cloud server. The cloud may be able to deduce the client's behaviour and even the encrypted data's content by

using this access pattern leak. For instance, using a broad inference attack with access pattern leaks and little prior information, a cloud may detect around 80% of the search queries in a searchable encryption system [1]. Technologies that may safeguard data and access patterns include oblivious random access machines (ORAM) [4], oblivious storage (OS) [3], and oblivious transfer (OT) [2]. These solutions, in general, let a client access its data that is outsourced and kept in an untrusted cloud without disclosing which items have been accessed or even the kind of actions that are requested. These methods have been extensively used in a variety of application scenarios, including searchable encryption [5]–[7], encrypted hidden volumes [8], [9], cloud storage [10]–[13], multi-party computing [14]–[18], etc., because of their high degree of privacy protection.

However, using current unaware systems in an MCS situation presents some difficulties for a number of reasons. First, wireless networks like Wi-Fi, LTE, and ad hoc are often used to link mobile devices to the Internet. This implies that the communication capacity available to mobile devices for data download and upload is constrained. Because of the high communication cost, many techniques that suffer from the well-known communication bandwidth overhead lower bound result O(log N) [4] cannot be used in MCS.1. Second, despite notable advancements in processing power, contemporary mobile devices—such as smartphones and tablets—remain unable to match the capabilities of personal computers or other robust gadgets. Mobile gadgets with complex computing also have shorter battery lives. Because of the complicated client-side encryption and decryption computation involved, some schemes based on fully homomorphic encryption (FHE) [19] or multi-layer onion additively homomorphic encryption [20] are also unsuitable for MCS, even though they get around the communication lower bound and achieve constant communication bandwidth overhead. Thirdly, the larger minimum effective item size also hurts a lot of the current unaware methods. The smallest amount of bits in an effective item of an oblivious system needed to satisfy the predetermined communication complexity (constant or logarithmic) is referred to as the minimum effective item size. The mobile client cannot fine-grained access its own data because to Lager item size. Additionally, it raises the current oblivious techniques' communication or processing overhead even more.

Certain unaware systems aim to increase efficiency by using data location. Data locality indicates a client's propensity to get its data quickly. Two common forms of reference locality of data access are spatial locality and temporal locality. When a certain item is accessed, the client may access neighbouring data objects, which is referred to as spatial locality. Temporal

locality is the term used to describe the client's frequent short-term reuse of data. When using non-constant communication overhead oblivious techniques that take spatial proximity into account, the amortised communication overhead when accessing many objects is less than when accessing a single item on its own [21]. By using temporal locality, some oblivious techniques may also become much more efficient since, once an object is visited, retrieving it again just needs a little amount of light processing and communication. But to the best of our knowledge, no relevant study has taken temporal location into account.

II. Related Work

In order to protect access pattern privacy, Goldreich and Ostrovsky presented the initial idea, known as the oblivious random access machine (ORAM) [4]. In addition to demonstrating a communication overhead lower-bound blowup, they offered a practical solution, Square Root ORAM ^{SM}(log N). The memory, or cloud in a cloud computing application, functions as a passive storage entity in their context (passive setting) and doesn't do any processing on data. A number of works have been enhanced in terms of theory and efficiency within this context [22]–[33]. Their architecture was first arranged by Shi et al. into a binary tree over buckets [24]. The suggested structure achieves O(log3 N) communication worst-case cost by operating blocks via tree pathways. Stefanov et al. presented Path ORAM [26], which is based on the binary tree ORAM framework. In a passive context, it accomplished the Ω(log N) lower-bound blowup that Goldreich and Ostrovsky [4] had shown. Additionally, it was much less complex than previous designs since it did not need complex cryptographic primitives, and given appropriate settings, it was efficient with a tiny end-to-end latency.

In fact, the cloud of today is thought to possess substantial processing power and be able to do complex calculations. Subsequent efforts [19], [20], and [34] adopted the computation cloud configuration and got around the lower-bound by letting the cloud handle complex calculation for the client. Apon et al. initially formalised the verifiable oblivious storage, which extends the concept of ORAM by enabling the storage medium to execute computation, even though they were not the first to adopt the cloud computing paradigm [19]. An Onion ORAM, or continuous communication bandwidth ORAM, incorporating cloud computing was suggested by Devadas et al. [20]. The data blocks in Onion ORAM were encrypted using a multi-layer encryption scheme that formed a "onion" and was additively homomorphic [35, 36]. Alternatively, the scheme was somewhat homomorphic and allowed the client to retrieve the target blocks and evict blocks through paths with small encrypted

select vectors. For appropriate security settings, Onion ORAM overflowing with minimal probability by combining the reverse lexicographical eviction order technique [16]. Another constant communication bandwidth O RAM, called C-ORAM, was suggested by Moataz et al. [34]. In contrast to Onion ORAM, C-ORAM eliminated multilayer homomorphic encryption in favour of an effective oblivious merging method.

As a result, both the necessary block size and cloud computing costs were greatly decreased. Data locality in OS/ORAM systems was recently the subject of many related publications [21], [37]–[39]. To increase efficiency, the authors of [38] discovered a particular kind of programme locality for recursive ORAM scheme operations. Locality-preserving ORAMs were explicitly explored by Asharov et al. in [39]. These ORAMs maintained the locality of the accessible memory areas and only revealed the lengths of the contiguous memory regions that were accessed. We do not investigate increasing spatial locality in this study since it does not considerably enhance the performance of schemes with constant communication bandwidth overhead.

III. Proposed model

As shown in Fig. 1, there are two entities involved in a mobile cloud storage system, e.g. client (C) and cloud (S). The client C first employs some technologies to protect its data, such as encryption schemes. Then it uploads its encrypted data into a remote cloud server S. After that, the client C can access its data via a mobile device, such as mobile phone, tablet or laptop computer. The client's data is represented as a "key-value" form similar to most cloud storage services. Specifically, the cloud supports the following fundamental access operations. • get(k). If there is an item labeled with the key k (abbr. item k) in the storage, then return the corresponding value to the client, else return ⊥. • put(k, v∗). If the item k is in the storage, then update the value of this item with v ∗, else insert a new item tuple (k, v∗) to the storage. • remove(k). If the item k is in the storage, then delete the item from the storage.

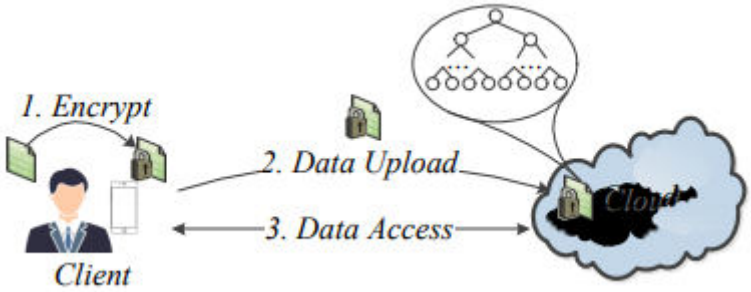


Figure 1: System Model

In the system, the communication channel between the client and the cloud is considered to be secure. That means the confidentiality and integrity of all messages in the channel are guaranteed. This condition can be easily achieved with standard cryptographic tools, such as public key encryption, digital signature and public key infrastructure (PKI). This helps to simplify the description of our scheme.

Additively Homomorphic Encryption

Homomorphic encryption is a form of public key encryption. It allows anyone with the public key to manipulate ciphertexts to generate a new ciphertext, which is encrypted of corresponding operation result of original plaintexts. Compared with fully homomorphic encryption, additively homomorphic encryption scheme only supports homomorphic additive operation, but has higher efficiency. In our scheme, we employ a special additively homomorphic encryption scheme, i.e. Damgard-Jurik construction [˚ 35]. In Damgard-Jurik construction, ciphertexts are hierarchical, ˚ which means the lower layer ciphertext can be encrypted into a higher layer. Additionally, for two ciphertexts in the same layer, anyone with the public key can execute homomorphic additive operation on them without decrypting. Specifically, there are three algorithms in Damgard-Jurik ˚ construction.

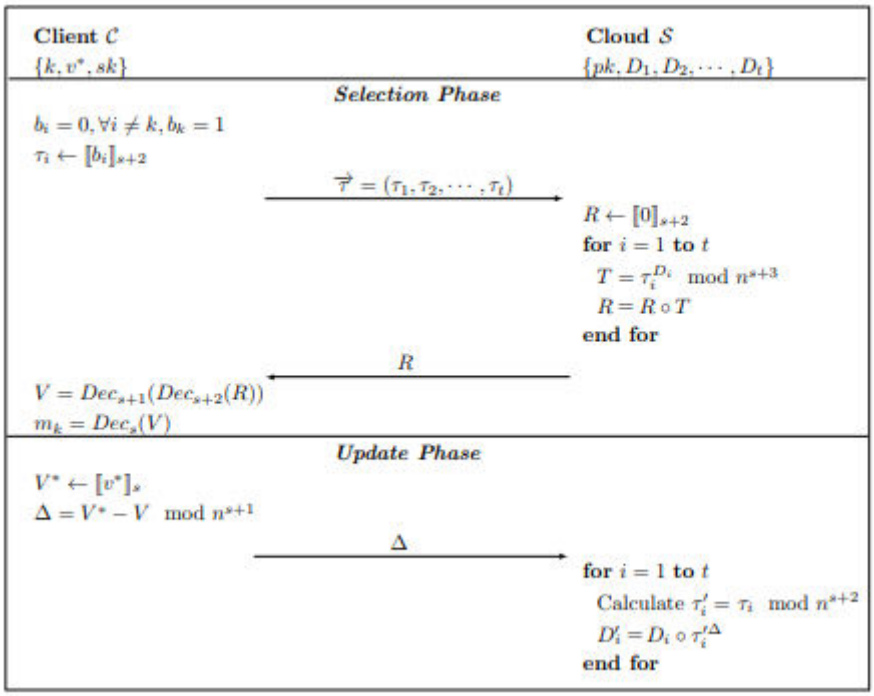
• Key Generation. The algorithm takes input a security parameter κ and generates an RSA modulus n = pq, where |p| = |q| = κ. Then the public key is n while the secret key is the least common multiple λ = Lcm(p − 1, q − 1). • Encryption. To encrypt a message m ∈ Zns , the algorithm first chooses a random number r ∈ Z ∗ ns+1 , and sets the ciphertext as c = (1 + n) m · r n s mod n s+1 . 3 • Decryption. To decrypt a ciphertext c ∈ Z ∗ ns+1, the algorithm first computes a number d, such that d = 1 mod n s and d = 0 mod λ by the Chinese Remainder Theorem.

$$\begin{aligned} c^d &= \left((1+n)^m \cdot r^{n^s}\right)^d \\ &= (1+n)^{md} \bmod n^s \cdot \left(r^{n^s}\right)^d \bmod \lambda \\ &= (1+n)^m \bmod n^{s+1} \end{aligned}$$

After that, invokes the algorithm described in [35] to retrieve $m \bmod n^s$ from$(1+n)^m \bmod n^{s+1}$.

Note that, when encrypting a message m ∈ Zns into the s layer to get a ciphertext c ∈ Zns+1 , the ciphertext is also in the s + 1 layer plaintext space and can be encrypted again to get

another ciphertext c 0 ∈ Zns+2 . Accordingly, to retrieve the message m from the ciphertext c 0 , the decryption algorithm should be executed twice. For convenience, we use the notation JmKl to denote the AHE ciphertext directly encrypted m ∈ Znl into the l layer. We also use ◦ to represent the homomorphic addition operation: Jm1Kl ◦ Jm2Kl = Jm1 + m2Kl . Moreover, we independently explore the following useful property of Damgard-Jurik construction. In this cryp- ˚ tosystem, if a ciphertext is encrypted of a message from a lower layer message space into a higher level, this ciphertext can be reduced to the lower layer without decrypting it. For example, a message m ∈ Znu can be encrypted in w-th layer to get the ciphertext JmKw, where w > u. Then we can easily transform the ciphertext JmKw into the v layer to get a new ciphertext JmKv, where w > v ≥ u, by computing JmKv = JmKw mod n v+1. The detail of this property is shown in Appendix A.



Oblivious Selection and Update Protocol

Here, we first describe a notion named oblivious selection and update (OSU) protocol. In this two-party protocol, a client C securely stores a set of data in a cloud S. Then, the client C can obliviously retrieve a target block from the cloud and update it with another block with small client computation. We believe this protocol will be of independent interest in many secure multi-party computation scenarios. Formally, the oblivious selection and update protocol is defined as follows. Definition 2 (Oblivious Selection and Update Protocol). A two-phase oblivious selection and update protocol is an interactive protocol between a client and a cloud

with two phases i.e. selection phase and update phase. In the protocol, the cloud takes input a set of protected data set D = {D1, D2, ..., Dt} and public information, e.g. public key pk, and the client takes input an index k ($1 \leq k \leq t$), a fresh value v $*$ and its secret information, e.g. secret key sk. At the end of the selection phase, the client will get Dk. At the end of the update phase, Dk will be updated by the fresh value v $*$ and Di will remains unchanged for all i 6= k. During the protocol, the cloud learns nothing about the data set D or the index k. formally, the following notations are used to denote the two phases of OSU.

$$(D_k; \perp) \leftarrow OSU^1(k, sk; \mathcal{D}, pk)$$
$$(\perp; \mathcal{D}') \leftarrow OSU^2(D_k, k, v^*, sk; \mathcal{D}, pk)$$

It easy to know that data Di should be protected by probabilistic method, such as probabilistic encryption. Otherwise, the cloud will learn the index k while updating Dk with the fresh value v $*$. Besides, the fresh value v $*$ is not necessary until the update phase. Thus, the client can determine the fresh value v $*$ after obtaining Dk.

IV. Implementation and Evaluation

We first compare the proposed mobile cloud storage scheme with other two oblivious random access machine schemes, which also have constant communication bandwidth overhead, i.e. Onion ORAM [20] and C-ORAM [34]. The comparison is shown in the Table 2. Since there are omitted large constants in Onion ORAM and C-ORAM, our scheme has significantly smaller item (block) size than other two schemes.

Then we implement the proposed mobile cloud storage scheme with Java and conduct the evaluation environment with workstations to simulate the client and the cloud, respectively. The workstations are all installed with Ubuntu 16.06 LTS system with 3.4 GHz Intel Core i5-7500 processor.

For security, we set the parameters of AHE scheme with |p| = |q| = 1024 and |n| = 2048. We also partially implement Onion ORAM and C-ORAM and set parameters of Onion ORAM and C-ORAM following [41]. Specifically, the real block number of a bucket and the eviction factor are both set as 333 and λ is set as 80. All experiments are run in four-threaded method and semi-honest setting. The Table 3 indicates the performances of per access in three schemes on a reasonable database (1 GiB). Due to the smaller item size and constant parameters, our scheme is much more efficient than Onion ORAM and C-ORAM in terms of client computation, cloud computation and communication. Since the runtime of Onion ORAM is enormous, the experimental results of Onion ORAM are estimated based on the

evaluation performances of core operations in Onion ORAM. In our evaluation environment, even in the best case and omitting the amortized cost of eviction, the cloud computation of per access in Onion ORAM still takes about 3.5 days. Note that, if we repeatedly run access operation 18 times in our scheme to retrieve same size data (81 KiB) as in C-ORAM, our scheme has almost the same client computation (25.38 s) as the C-ORAM scheme, but the cloud computation is still much smaller than C-ORAM, i.e. about 10 times smaller.

V. Conclusion

In this work, we suggest a mobile cloud storage system (MCS) that is effective, safe, and privacy-preserving. The suggested strategy may simultaneously secure access patterns and data. Our approach offers lower item size, lightweight client-side computation, and consistent communication overhead when compared to previous techniques. Additionally, we examine temporal locality to enhance the scheme's efficiency even more. We can verify that our technique can withstand malevolent cloud by integrating extra methods. We also provide an obliging selection and update protocol, which is a fundamental component of the proposed MCS system. A client may use a short vector to obligingly pick and update one of its encrypted data items that are outsourced to the cloud. We think our protocol could be of independent relevance for additional secure multi-party computing application situations because of its modest client computation and communication. Our plan accomplishes data confidentiality and a suitable degree of privacy preservation, according to security and privacy proofs and evaluations. Finally, we thoroughly estimate our structure in a simulation environment and compare our scheme with other two oblivious storage strategies. The results show that our strategy performs well and is notably efficient.

VI. References

[1] M. S. Islam, M. Kuzu, and M. Kantarcioglu, “Access pattern disclosure on searchable encryption: Ramification, attack and mitigation,” in 19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012, 2012. [Online].

[2] J. Kilian, “Founding cryptography on oblivious transfer,” in Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA, 1988, pp. 20–31. [Online].

[3] D. Boneh, D. Mazieres, and R. A. Popa, “Remote oblivious storage: Making oblivious ram practical,” pp. 1–18, 2011.

[4] O. Goldreich and R. Ostrovsky, “Software protection and simulation on oblivious rams,” J. ACM, vol. 43, no. 3, pp. 431–473, 1996. [Online].

[5] J. Camenisch, M. Kohlweiss, A. Rial, and C. Sheedy, “Blind and anonymous identity-based encryption and authorised private searches on public key encrypted data,” in Public Key Cryptography - PKC 2009, 12th International Conference on Practice and Theory in Public Key Cryptography, Irvine, CA, USA, March 18-20, 2009. Proceedings, 2009, pp. 196–214. [Online].

[6] T. Hoang, A. A. Yavuz, F. B. Durak, and J. Guajardo, “Oblivious dynamic searchable encryption via distributed PIR and ORAM,” IACR Cryptology ePrint Archive, vol. 2017, p. 1158, 2017. [Online].

[7] S. Garg, P. Mohassel, and C. Papamanthou, “TWORAM: efficient oblivious RAM in two rounds with applications to searchable encryption,” in Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III, 2016, pp. 563–592. [Online].

[8] E. Blass, T. Mayberry, G. Noubir, and K. Onarlioglu, “Toward robust hidden volumes using write-only oblivious RAM,” in Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014, 2014, pp. 203–214. [Online].

[9] D. S. Roche, A. J. Aviv, S. G. Choi, and T. Mayberry, “Deterministic, stash-free write-only ORAM,” in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017, 2017, pp. 507–521. [Online].

[10] E. Stefanov and E. Shi, “Oblivistore: High performance oblivious cloud storage,” in 2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013, 2013, pp. 253–267. [Online].

[11] D. Cash, A. Kupc¸ ¨ u, and D. Wichs, “Dynamic proofs of ¨ retrievability via oblivious RAM,” in Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International

Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings, 2013, pp. 279–295. [Online].

[12] E. Stefanov and E. Shi, "Multi-cloud oblivious storage," in 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013, 2013, pp. 247–258. [Online].

[13] B. Carbunar and R. Sion, "Write-once read-many oblivious RAM," IEEE Trans. Information Forensics and Security, vol. 6, no. 4, pp. 1394–1403, 2011.

[14] X. S. Wang, Y. Huang, T. H. Chan, A. Shelat, and E. Shi, "SCORAM: oblivious RAM for secure computation," in Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014, 2014, pp. 191–202. [Online].

[15] E. Boyle, K. Chung, and R. Pass, "Large-scale secure computation: Multi-party computation for (parallel) RAM programs," in Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II, 2015, pp. 742–762. [Online].

[16] C. Gentry, K. A. Goldman, S. Halevi, C. S. Jutla, M. Raykova, and D. Wichs, "Optimizing ORAM and using it efficiently for secure computation," in Privacy Enhancing Technologies - 13th International Symposium, PETS 2013, Bloomington, IN, USA, July 10-12, 2013. Proceedings, 2013, pp. 1–18. [Online].

[17] S. Lu and R. Ostrovsky, "Distributed oblivious RAM for secure two-party computation," in Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings, 2013, pp. 377–396. [Online].

[18] S. Zahur, X. Wang, M. Raykova, A. Gascon, J. Doerner, D. Evans, ´ and J. Katz, "Revisiting square-root ORAM: efficient random access in multi-party computation," in IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016, 2016, pp. 218–234. [Online].

[19] D. Apon, J. Katz, E. Shi, and A. Thiruvengadam, "Verifiable oblivious storage," in Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26- 28, 2014. Proceedings, 2014, pp. 131–148. [Online].

[20] S. Devadas, M. van Dijk, C. W. Fletcher, L. Ren, E. Shi, and D. Wichs, "Onion ORAM: A constant bandwidth blowup oblivious RAM," in Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II, 2016, pp. 145–174. [Online].

[21] A. Chakraborti, A. J. Aviv, S. G. Choi, T. Mayberry, D. S. Roche, and R. Sion, "roram: Efficient range ORAM with o(log2 N) locality," in 26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019, 2019. [Online]. Available: https://www.ndss-symposium.org/ ndss-paper/roram-efficient-range-oram-with-olog2-n-locality/

[22] B. Pinkas and T. Reinman, "Oblivious RAM revisited," in Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings, 2010, pp. 502–519. [Online].

[23] I. Damgard, S. Meldgaard, and J. B. Nielsen, "Perfectly secure ˚ oblivious RAM without random oracles," in Theory of Cryptography - 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings, 2011, pp. 144–163. [Online].

[24] E. Shi, T. H. Chan, E. Stefanov, and M. Li, "Oblivious RAM with O((logN) 3) worst-case cost," in Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings, 2011, pp. 197–214. [Online].

[25] E. Stefanov, E. Shi, and D. X. Song, "Towards practical oblivious RAM," in 19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012, 2012. [Online].