Summary of Changes

Dear Editors and Reviewers,

We sincerely thank the Editor and the reviewers for their valuable and constructive comments and suggestions. The review comments are very helpful in further improving the quality of this work. Based on the suggestions, we have carefully revised the manuscript. Our detailed responses are as follows.

Yours sincerely,

Jie Zhu, Qi Li, Cong Wang, Xingliang Yuan, Qian Wang, and Kui Ren

Response to Associate Editor

Response: Thank the editor for paying attention to this paper. This response covers two parts. Each part answers in detail the individual questions raised by the reviewers. All sections and reference numbers below are based on the revised paper, and the major changes in the revised paper are colored in blue.

The major changes are in the following:

- We summarized the shortcomings of the existing schemes and clarified the contributions of our paper in **Section 1**.
- We clarified the problem statement section, i.e., **Section 2**, and added a preliminary section to review Merkle Patricia Tree (MPT), incremental hash, and searchable encryption.
- We rewrote the overview section and clarified the design of our GSSE scheme in **Section 3**.
- We presented the communication overhead of our GSSE scheme, i.e., **Section 6**.
- We added a discuss section, i.e., **Section 7**, to clarify the applications of our scheme in the revised manuscript.
- We modified the related work section, i.e., **Section 8**, and compared our VSSE scheme with authenticated data structure.

Response to reviewer #1

Comment 1.1: *The title lacks a final noun like "data stores". As the title already contains "Verification" the adjective "verifiable" does not have any additional meaning and should be removed.*

Response 1.1: Thank the reviewer for pointing out this issue. We changed the title to “Enabling Generic, Verifiable, and Secure Data Search on Cloud”.

Comment 1.2: *The presentation of related work lacks details. The authors claim that the client in [5], [8], [9] cannot detect, if the server returns an (incorrect) empty result set. This claim has to be supported by further argument (or a citation).*

Response 1.2: Thank the reviewer for raising this question**.** Here, we clarified that the existing schemes (i.e., [5] [8] [9]) fell short of detecting the attack that the server maliciously returned an empty result set.

The scheme proposed by Kamara et al. [5] constructed a Merkle Tree by storing the key and value pairs in its leaf node, where the key was the encrypted keyword and the value was the encrypted data set that contained the keyword. A data user (aka data owner) only kept the root hash of the Merkle Tree. By reconstructing a root hash through the path of the target nodes, the user can check the integrity of the search result. However, they did not clearly state how to handle the situation that the server maliciously returned an empty result set. Unfortunately, their approach

cannot detect such attacks. The key reason is that there is no path that matches a non-existent keyword of the Merkle Tree. One naïve way to address this issue is that they can store the results (including empty sets) of the entire keyword space to the tree, which is not scalable. In addition, this approach is not dynamic friendly. It may require rebuilding the entire tree after data update.

Kurosawa et al. [8] leveraged Message Authenticated Code (MAC) to ensure data integrity and utilizes RSA accumulator to ensure data freshness during data updates. However, the existing study [12] showed that users in [8] required maintaining all keywords in the datasets to detect cheating of servers. It introduces significant overhead for users to maintain a large set of keywords. Therefore, paper [12] proposed a no-dictionary VSSE scheme, which means users do not need to keep the keywords set as a dictionary. However, their scheme only works in the static setting and does not consider data update scenarios.

Stefanov et al. [9] also used MAC and timestamp to ensure data integrity and freshness. However, they were still unable verify an empty search result unless the data owner maintained all keywords locally.

In summary, verification on the three-party model (across multiple users) should allow users to verify an empty result in addition to verification of integrity and freshness of search results. Meanwhile, it is desirable that users do not need to maintain a large set of keywords locally for the verification. We clarified this issue in **Section 1**.

Comment 1.3: *In Section 7, the authors should add more details on multi-user schemes, verifiable SSE, and dynamic SSE, respectively. What are the differences between the cited references and the proposed method?*

Response 1.3: Thanks for the comments. We emphasized that we did not aim to design a multi-user scheme for access control of search. Instead, our design aimed to provide results verification across multiple users. To the best of our knowledge, the existing VSSE schemes were all based on the two-party model and there did not exist a verifiable solution on the three-party model [11]. Specifically, verification on the three-party model faced more challenges than the two-party model. For example, when data was shared among users, a malicious server can easily mounted a data freshness attack because the users cannot detect a data update unless the data owner sent the update information to them. However, such an approach incurred a significant communication cost. In this paper, we proposed a VSSE scheme that is able to not only detect data integrity attack (especially when the server deliberately returns an empty result) but also capture data freshness attack by using a timestamp chain based mechanism. In our scheme, data owner only needed to send update information to the server, and data users can verify the search results by leveraging the information retrieved from the server without interacting with the data owner.

We discussed more verifiable searchable encryption schemes including VSSE scheme and Verifiable PKE (Public Key Encryption) schemes with and without dynamic data update, and compared them with our scheme in **Section 8**. Meanwhile, we also discussed the existing multi-user schemes in this section.

First, we compared several VSSE schemes with our scheme in **Section 8**. The CS2 scheme [5] cannot prevent data integrity attacks when the server maliciously returned an empty result. Besides, their scheme was based on the two-party model, and thus did not support verification on data sharing scenarios. Kurosawa et al. proposed a number of verifiable SSE schemes [6] [8] [12]. However, their schemes either had low search efficiency [6] [8] or did not support file updates [6] [12]. Chai et al. [7] and Cheng et al. [10] proposed VSSE schemes that only handled verification of results on static data. Stefanov et al. [9] achieved verifiability by leveraging message authenticated code, but they did not handle the case when the server intentionally returned an empty result. Bost et al. [11] also presented a verifiable dynamic SSE scheme. However, their scheme required two rounds of communication for result verification and only worked for the two-party model. Generally speaking, our scheme is a completely generic verifiable SSE scheme that can work across multiple users against data freshness attack and data integrity attack. Moreover, it requires only one round of communication, and thus it can be easily applied in practice.

Second, we discussed more related work of the verifiable PKE schemes in **Section 8**. The first verifiable PKE scheme proposed by Zheng et al. [40] leveraged attribute-based encryption to verify search results on static data. Liu et al. [41] improved the efficiency of Zheng et al.'s scheme. Sun et al. [42] also provided a verifiable scheme that enabled conjunctive keyword search. However, the efficiency of these verifiable PKE schemes was much lower than VSSE schemes.

Comment 1.4: *The authors claim that their method can be applied "to various existing SSE schemes [3], [4], [9]". Can their method only be applied to these SSE schemes? If so, what is the difference of [3], [4], [9] to other SSE schemes to which the proposed method cannot be applied? Indeed the cited schemes [3], [4], [9] are all symmetric SSE schemes. Making them multi-user would require appropriate key management and revoking users would require changing the encryption keys and re-encrypting the data. This problem is ignored by the authors. So it is not clear why their approach is more appropriate in a multi-user setting than others.*

Response 1.4: Thanks for the questions. The searchable encryption schemes discussed in [3][4][9] implemented relatively complete SSE functionalities, and they were not particular schemes. Our scheme provided generic verification for various SSE schemes because the verification index in our scheme was separated from the SSE index and did not rely on any specific constructions of SSE.

For the key management issue, as we mentioned above (see Response 1.3), this paper did not focus on designing a multi-user SSE solution with access control. Instead, our scheme can work with these SSE schemes. Therefore, the implementation of the multi-user access control and key management was not discussed in details in this paper. In the revised manuscript, we discussed the implementation of multi-user SSE, which was based on broadcast encryption. Broadcast encryption does not require re-encryption of index or data. We demonstrated that our scheme was complementary to the existing multi-user SSE schemes with access control. Note that, only keys that generate the search tokens need to be updated under user revocation. We clarified these issues in **Section 7**.

Comment 1.5: *It is not clear, what exactly is the difference between contribution 1) and contribution 2) on page 2. They both sound equivalent.*

Response 1.5: Thanks for the comments. Contribution 1) focused on the generality of our scheme, we proposed a generic framework for result verification based on SSE that separated the index for search result verification from the SSE search index, which can be applied to various SSE schemes. Contribution 2) ensured functional integrity of our scheme. In particular, our scheme ensures the freshness and integrity of the search results on the three-party model, which was not well addressed by the previous schemes [5][8][9]. We clarified these two contributions in **Section 1**.

Comment 1.6: *The design goal 3) Efficiency does not seem to include the retrieval of search results. Otherwise, their intended search efficiency is not achievable. Consider n files containing all only one keyword, which is the same for all files. Search time for this keyword then will be at best, as all n results have to be returned and this takes n time. Search time is not in O(log(|W|)) = O(1), as the authors want to achieve.*

Response 1.6: Thanks for the efficiency question. In the paper, we separately evaluated the computational complexity of SSE and search result verification. The computational complexity here only refers to that of searching the verification results, which is $O(\log|W|)$, where $|W|$ is the number of keywords in the data set. In **Section** 2, we clarified this issue.

Comment 1.7: *The authors' understanding of an incremental hash seems to deviate from the cited papers [22] and [5]. Further explanation or better formulation is needed.*

Response 1.7: Thank the reviewer for pointing out this issue. We reviewed the incremental hash in **Section 2.** An incremental hash function is a collision-resistant function, which was first proposed by Bellare et al. [22]. We revised the description of the incremental hash in **Section 2**.

In our scheme, if a file collection of a data owner is changed, the data owner uses the incremental hash function to quickly calculate a collision-resistance hash value of the changed data, which achieves high efficiency in computing hash values. Another advantage is that, when a file is added or deleted, a collision-resistance hash value ensures the security of our scheme during update.

Comment 1.8:*...is encoded using RLP code, which ensures the cryptographically secure search": What does RLP stand for? Explain.*

Response 1.8: Thanks for pointing out the issue of the RLP code. RLP (Recursive Length Prefix) code is one type of the encoding methods that can encode arbitrary binary data, which was adopted to serialize objects in Ethereum. We clarified this issue in **Section 2**.

Comment 1.9: *In Table 1, the search runtime of [9] is (for search terms with many results) better than the given formula (see last lines of the first page of [9]). Table 1 has to be discussed more in detail in the text: it is not clear what the difference between - and x is. What do the authors mean by the column "Multi User"? Explanation is needed. They seem to mean "Verification in a multi*

user setting". It is not clear what the authors compare in the table. Do they want to compare verifiable SSE schemes or do they want to compare generic schemes which provide verification for SSE schemes? It seems that they mix up both things in their table.

Response 1.9: Thank the reviewer for pointing out the issues. In **Table 1**, 'x' represents the requirements that are not enabled, and '-' represents the requirements that are not required. The reason that we use '-' in data freshness verification of the existing static SSE schemes [6] [7] [10] [12] is that these schemes are not vulnerable to the data freshness attack. We explained these notations of **Table 1 in Section 1**.

Also, multi-user means search result verification across multiple users. In the revised manuscript, we changed the term multi-user to three-party to avoid the confusion, and clarified this in a note in **Table 1.**

Moreover, we used **Table 1** to compare various aspects of various VSSE schemes, e.g., evaluating whether they support verification in dynamic SSE, whether they support verification on the three-party models, and whether they support data freshness and integrity verification. We also compared the search efficiency and generality of these VSSE schemes. We believe that these are important parts that should be enabled in a VSSE scheme.

In this paper, we considered the worst case of the search complexity of that scheme proposed by Stefanov et al. [9]. We corrected it in the revised manuscript (see **Table 1)**.

Comment 1.10: I*n the performance evaluation, it is not clear what portion of the Enron email dataset is used. Moreover, benchmarks only report MPT generation and update. No SSE scheme is benchmarked. Figure 5 is hardly readable and should be enlarged.*

Response 1.10: Thanks for the questions. The part of the Enron email dataset that we used in our experiments is between "allen-p" and "kaminski-v". In particular, we not only measured the initialization and update delays of MPT, but also evaluated the search delays (see in **Fig. 7**) and the overhead of generating MPT root node (see in **Fig. 9**). Note that, the prove operation is essentially a search operation performed on the MPT structure.

We used a dynamic SSE scheme proposed by Cash et al. [4] as a baseline, and enabled result verification for it. As we can see in **Fig. 13** and **Table 3**, the extra overhead incurred by our scheme is acceptable. We revised the figures and make them clearer in **Section 6**.

Comment 1.11: *Moreover, the authors should also survey authenticated data structures in general (for example, the skip list proposed by Goodrich/Tamassia) and state why the Merkle Patricia tree is better. In this sense, the paper just focuses on the verification part and hence should be compared to other authenticated data structures but not to searchable encryption schemes.*

Response 1.11: Thanks for the suggestions about related work. In the revised manuscript, we discussed more related work on authenticated data structures including Merkle Tree [5] [36],

authenticated hash table [37] and skip list [38][39]. We selected the MPT structure to implement our scheme due to the following reasons: First, Merkle Tree is not a balanced data structure, and search delay of the authenticated hash table leveraging the RSA accumulator is lower than Merkle Patricia Tree (MPT). The search delay of the authenticated hash table is in millisecond level, while that of MPT is in microsecond level. Second, the current Merkle Tree scheme proposed by Kamara et al. [5] did not store additional information such as keyword in its intermediate node, so that it cannot enable empty result validation (as we discussed in Respons1.2).

Furthermore, skip list [38][39] could be an alternate to implement our scheme. Unfortunately, the storage cost of skip list is higher than a tree structure since the keyword information needs to be stored in the path.

Note that, SSE result verification is different from verification used in proof of data possession (PDP) and proof retrievability (POR). An SSE scheme needs to not only verify the integrity of a particular file block but also the integrity of all the files that contain the specific keyword, which cannot be easily achieved by the authenticated data structures. We clarified these issues in **Section 8**.

Comment 1.12: *There are several formal problems in the article:*

1) *Notations keep changing throughout the article; e.g., is the curly W equal to the italic W? Is the curly A equal to the italic A?*
2) *It is confusing that the author first state the algorithm names for MSSE schemes but then use slightly different algo names in their definition 3 and 4, e.g. KeyGen -> KGen, AddUser -> EnrollUser.*
3) *Furthermore the names of parameters are changed between Def. 3 and Def. 4 e.g. the first argument of Init looses its index "O" without any apparent reason.*
4) *The amount of keys needed is not clear: The authors sometimes speak of the symmetric key K and sometimes of the symmetric keys* $K = K_1, K_2$. *This is at least confusing.*
5) *The amount of spelling mistakes is unacceptably high. For example: "Bottem", "marcobenchmark", "all tokens was encrypted", "sciphers","data which can be treat as", "tempers", "bandwidth cost increase","substaction operation", "we compares", "exits" instead of exist, " the server need to return", lots of missing "a", "an" or "the"*

Response 1.12: Thanks for pointing out the grammar errors. We proofread the manuscript and fixed them in the revised manuscript.

Response to reviewer #2

Comment 2.1: *I think the paper can be made more readable by leveraging examples. For instance, the paper contains a very condensed description of an MPT without going into details. I would suggest the authors expand on the description possibly with an example.*

Response 2.1: Thanks for the suggestion. We explained the MPT structure in details by using a simple example in **Fig. 1**. Specifically, the simple example illustrated insertion of Merkle Patricia Tree (MPT), and explained the node changes during the MPT insertion processes (see **Section 2**).

Comment 2.2: *A significant majority of the approach is first provided at an abstract level which is then expanded with a little example. I would say for the sake saving space, setting up the abstract notions separate from the details requires the reader to jump around the text. A natural flow will be to provide the abstract notions and then instantiate them right away.*

Response 2.2: Thank the reviewer for pointing out this problem. In the revised manuscript, we added a notation table (i.e., **Table 2**) in **Section 3** and revised the definition to enable a natural flow for readers in **Section 4**.

Comment 2.3: *Although I sincerely appreciate the author using an example to explain their verifiability approach, because of the lack of details on the underlying SSE schemes make it very difficult for a reader to relate to it. I would suggest giving a small preliminary on the underlying SSE scheme; not the abstract functionality.*

Response 2.3: Thanks for the suggestions. We explained the secure searchable encryption schemes in **Section 2**. Searchable encryption allows the server to perform search operations without seeing plaintext data. It empowers the server an ability to search over ciphertext and ensures the security of data on the server. In this revised manuscript, we discussed the existing categories of searchable encryption in **Section 2**.

Comment 2.4: *I had the impression that the adversary can be fully malicious; in light of my understanding the following sentence does not make sense to me: the cloud servers do not have motivation to fake clocks*

Response 2.4: Thanks for the question. A malicious server can indeed fake clocks, but it cannot fake the timestamp chain. If the server faked the clock, the timestamp will not allow a server to bypass the verification performed by users. Therefore, a server does not have any incentive to fake clocks. We clarified this issue in **Section 4**.

Comment 2.5: *The paper has a lot of typos and incoherent sentences which should be fixed with a thorough editorial pass. For instance, macro-benchmark is written as marcobenchmark in a few places including section heading; I was thinking it was some special kind of benchmark, only later did I figure out it is supposed to be macro-benchmark.*

Response 2.5: Sorry for the spelling and grammar mistakes, we fixed these issues in the revised manuscript.

Comment 2.6: *In the evaluation, I am not sure why the authors use simulation for macro-benchmarks. Why cannot one carry out the real experiment? As the scheme requires interaction, I would request the authors to consider network bandwidth in their evaluation. If verifiability can be achieved with low network bandwidth, this will be a strength for the paper. I would have also liked to see the index sizes in the experiment.*

Response 2.6: Thanks for pointing out this issue. Actually, we did not use a traditional "simulation" approach to perform experiments. Instead, we built a real system to carry out our experiments on a local machine and used the real dataset to feed into the system to measure the performance. Sorry for the confusion. We reported the average bandwidth consumption of our scheme after 50,000 rounds of keyword searching in **Section 6**. As shown in **Table 3**, **Fig. 7** and **Fig 8**, the average proof size of our scheme is lower than 3KB, while the average size of the search results of the SSE scheme [4] is about 53KB. Moreover, the size of the search tokens is 32Byte, and the size of tokens used in the SSE scheme on average is 390Bytes. The above results demonstrate that the overhead introduced by our scheme is acceptable. Moreover, the storage space required by our MPT structure is given in **Fig. 10.** For a 590MB database with 1,000,000 keywords, the storage cost of the MPT is only about 82MB. As the data set (aka, the Enron email) contains a large number of keywords, the storage space of MPT is relatively large. Note that, if a data user stores various media types of data set (e.g., images or music) with fewer keywords or attributes, the storage overhead of MPT is negligible compared with the size of the data set itself. We revised the manuscript correspondingly in **Section 6**.

References

[2] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, “Searchable symmetric encryption: improved definitions and efficient constructions,” Journal of Computer Security, vol. 19, no. 5, pp. 895–934, 2011.

[3] S. Kamara, C. Papamanthou, and T. Roeder, “Dynamic searchable symmetric encryption,” in Proceedings of the 2012 ACM conference on Computer and communications security. ACM, 2012, pp. 965–976.

[4] D. Cash, J. Jaeger, S. Jarecki, C. S. Jutla, H. Krawczyk, M.-C. Rosu, and M. Steiner, “Dynamic searchable encryption in verylarge databases: Data structures and implementation.” IACR Cryptology ePrint Archive, vol. 2014, p. 853, 2014.

[5] S. Kamara, C. Papamanthou, and T. Roeder, “Cs2: A semantic cryptographic cloud storage system,” Tech. Rep. MSR-TR-2011-58, Microsoft Technical Report (May 2011), http://research. microsoft.com/apps/pubs, Tech. Rep., 2011.

[6] K. Kurosawa and Y. Ohtaki, “Uc-secure searchable symmetric encryption,” in International Conference on Financial Cryptography and Data Security. Springer, 2012, pp. 285–298.

[7] Q. Chai and G. Gong, “Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers,” in 2012 IEEE International Conference on Communications (ICC). IEEE, 2012, pp. 917–922.

[8] K. Kurosawa and Y. Ohtaki, “How to update documents verifiably in searchable symmetric encryption,” in International Conference on Cryptology and Network Security. Springer, 2013, pp. 309–328.

[9] E. Stefanov, C. Papamanthou, and E. Shi, “Practical dynamic searchable encryption with small leakage.” in NDSS, vol. 14, 2014, pp. 23–26.

[10] R. Cheng, J. Yan, C. Guan, F. Zhang, and K. Ren, “Verifiable searchable symmetric encryption from indistinguishability obfuscation,” in Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security. ACM, 2015, pp. 621–626.

[11] R. Bost, P.-A. Fouque, and D. Pointcheval, “Verifiable dynamic symmetric searchable encryption: Optimality and forward security,” Cryptology ePrint Archive: Report 2016/062, Tech. Rep., 2016.

[12] W. Ogata and K. Kurosawa, “Efficient no-dictionary verifiable SSE,” Cryptology ePrint Archive: Report 2016/981, Tech. Rep., 2016.

[21] https://github.com/ethereum/wiki/wiki/RLP..

[22] M. Bellare, O. Goldreich, and S. Goldwasser, “Incremental cryptography: The case of hashing and signing,” in Annual International Cryptology Conference. Springer, 1994, pp. 216–233.

[31] Y. Yang, F. Bao, X. Ding, and R. H. Deng, “Multiuser private queries over encrypted databases,” International Journal of Applied Cryptography, vol. 1, no. 4, pp. 309–319, 2009.

[32] S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner, “Outsourced symmetric private information retrieval,” in Proceedings of the 2013 ACM SIGSAC conference on Computer &communications security. ACM, 2013, pp. 875–888.

[36] R. C. Merkle, “A digital signature based on a conventional encryption function,” in Conference on the Theory and Application of Cryptographic Techniques. Springer, 1987, pp. 369–378.

[37] C. Papamanthou, R. Tamassia, and N. Triandopoulos, “Authenticated hash tables,” in Proceedings of the 15th ACM conference on Computer and communications security. ACM, 2008, pp. 437–448.

[38] W. Pugh, “Skip lists: a probabilistic alternative to balanced trees,” Communications of the ACM, vol. 33, no. 6, pp. 668–676, 1990.

[39] M. T. Goodrich, R. Tamassia, and A. Schwerin, “Implementation of an authenticated dictionary with skip lists and commutative hashing,” in DARPA Information Survivability Conference & Exposition II, 2001. DISCEX’01. Proceedings, vol. 2. IEEE, 2001, pp. 68–82.

[40] Q. Zheng, S. Xu, and G. Ateniese, “Vabks: verifiable attributebased keyword search over outsourced encrypted data,” in IEEE INFOCOM 2014-IEEE Conference on Computer Communications. IEEE, 2014, pp. 522–530.

[41] P. Liu, J. Wang, H. Ma, and H. Nie, “Efficient verifiable public key encryption with keyword search based on kp-abe,” in Broadband and Wireless Computing, Communication and Applications (BWCCA), 2014 Ninth International Conference on. IEEE, 2014, pp. 584–589.

[42] W. Sun, X. Liu, W. Lou, Y. T. Hou, and H. Li, “Catch you if you lie to me: Efficient verifiable conjunctive keyword search over large dynamic encrypted cloud data,” in Computer Communications (INFOCOM), 2015 IEEE Conference on. IEEE, 2015, pp. 2110–2118.