ISSN: 2366-1313

Safe and Dependable Cloud Storage Allows the Dynamic Nature of Data via the Use of Secure System Coding Methods

Prakash Krishna Shinde

M Tech, Department of, Computer Engineering, Lecturer

Dr. D. Y. Patil Polytechnic Kolhapur Maharastra

Abstract: If you have a little amount of data that has to be stored, you may now utilize distant servers and the cloud. Instead of offering financial incentives, these servers make their customers' data retrievable at any time. A customer may verify the authenticity of their outsourced data thanks to secure cloud storage techniques. Through the use of methods from secure network coding, this study investigates the prospect of building secure cloud storage for dynamic data. We demonstrate that certain secure network coding schemes may be used to build efficient secure cloud storage protocols for dynamic data, and we build one such protocol (DSCS I) on top of a secure network coding protocol. As far as we are aware, DSCS I is the first secure cloud storage protocol for dynamic data built using secure network coding methods, making it safe in the standard model. Append-only data has many practical uses, even though generic dynamic data allows for unrestricted insertions, deletions, and updates. Finally, we present prototype implementations of DSCS I and DSCS II so that their performance may be evaluated. DSCS I is a general-purpose secure cloud storage protocol, therefore we built DSCS II to address the unique challenges of append-only data.

Keywords: Secure cloud storage, network coding, dynamic data, append-only data, public verifiability.

I. Introduction

With the advent of cloud computing, cloud servers provide to their customers (cloud users) numerous services that include delegation of vast amount of compute and outsourcing massive quantity of data. For example, a client with a smart phone with a low-performance CPU or limited storage cannot do intensive calculation or store huge volume of data. Under such conditions, she may outsource her computation/storage to the cloud server. In case of storage outsourcing, the cloud server stores enormous data on behalf of its customers (data owners) (data owners). However, in order to free up space, a malicious cloud server might erase part of the client's data (those are seldom visited). Secure cloud storage protocols (two-party protocols between the client and the server) give a means to check whether the server maintains the client’s data unhampered. Based on the nature of the outsourced data, these protocols are categorised as: secure cloud storage protocols for static data (SSCS) [2], [3], [4]

ISSN: 2366-1313

and for dynamic data (DSCS) [5], [6], [7], [8]. For static data, the customer cannot update her data after the original outsourcing (e.g., backup/archival data). Dynamic data are more general in that the customer may adjust her data as frequently as required. In secure cloud storage protocols, the client may audit the outsourced data without reading the complete data file, and yet be able to identify undesired modifications in data done by a hostile server. While conducting an audit, the client will submit a random challenge to the server, which will then construct proofs of storage (using the data it has stored) that answer the question posed by the client. If an audit can be performed by any third party auditor (TPA) using public parameters, the cloud storage protocol is considered publicly verifiable; otherwise, it is considered privately verifiable, meaning that the auditor needs to know some secret information about the client in order to verify the protocol. The entities involved in a secure cloud storage technique and the interaction among them are represented in Figure 1.

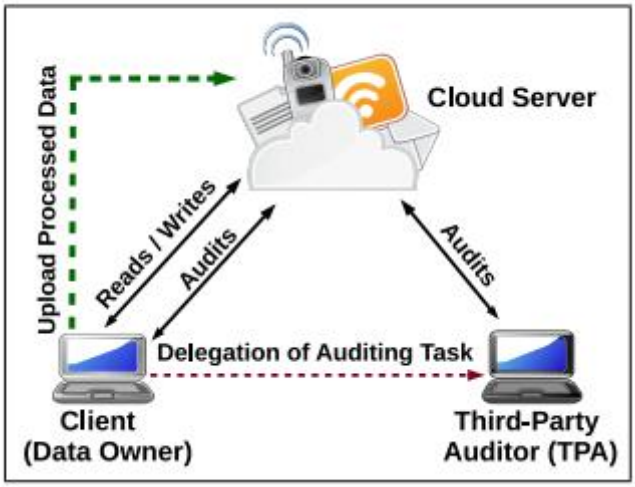

Figure 1: The architecture of a secure cloud storage protocol

Each node along a network route (other than the transmitter and recipient nodes) performs packet fusion in a network coding protocol [9, 10]. In comparison to store-and-forward routing, these protocols have better throughput, efficiency, and scalability; yet, they are vulnerable to pollution attacks in which malevolent intermediary nodes introduce erroneous packets. Because of the propagation of these packets, the receiving node may never be able to decode the original file transmitted by the sending node. To counteract such assaults, cryptographic methods are used by secure network coding (SNC) protocols. In these systems, the sender of a packet verifies its authenticity by appending a unique tag to it. These authentication tags are produced with the use of homomorphic message authentication codes (MACs) [11] or homomorphic signatures [12], [13], [14], [15]. As a result of the

ISSN: 2366-1313

homomorphic characteristic, a middle node may merge several incoming packets (and their tags) into a single packet and tag. In this paper, we adopt a fresh approach to the issue of how to build a secure cloud storage protocol for ephemeral data (DSCS). We look at the possibility of employing an SNC protocol as the basis for a robust DSCS protocol. The authors Chen et al. [16] find that safe cloud storage may be tied to secure network coding. In particular, they demonstrate that a secure cloud storage system for static data may be built by using some of the methods involved in an SNC protocol. However, its design is inadequate for applications that need frequent distant data updates (insertion, deletion, or modification) because of the inability to manage dynamic data. More research on an effective DSCS build with a secure network coding (SNC) protocol is required. Distributed storage systems [17, 18] employ network coding methods to disperse client data among numerous servers. However, its primary goal is to lessen the amount of time spent on repairs if several servers go down at once. By contrast, we investigate whether or not we can use the algorithms in an SNC protocol to build a safe and effective cloud storage system for changing data (for a single storage server).

II. Literature survey

In terms of IT infrastructure, cloud computing is seen as the way of the future. In doing so, it shifts the workload away from local machines and onto remote servers and huge data centres, where the security and reliability of data and service management are less certain. Since no one has faced something like this before, there are certain to be numerous unknown security risks associated with it. There is already research on the issue of how to best guarantee the security of data stored in the cloud. In this paper, we focus on the problem of facilitating the cloud client's third-party auditor's (TPA) verification of the authenticity of the cloud's dynamic data. The introduction of TPA removes the need for the customer to verify the security of his cloud-stored data by conducting audits on his behalf.

Since services in Cloud Computing are not restricted to archive or backup data solely, the support for data dynamics through the most common types of data operation, including block modification, insertion, and deletion, is also a key step toward reality. Prior efforts on remote data integrity assurance frequently lack either public audit capabilities or support for dynamic data operations; this study provides both. By first highlighting the challenges and potential security issues of direct extensions with fully dynamic data updates from previous works, the current system demonstrates how to build an elegant verification scheme for the smooth integration of these two prominent features into our protocol design.

ISSN: 2366-1313

In particular, we enhance the pre-existing proof of storage models by modifying the traditional Merkle Hash Tree design for block tag authentication, allowing for more effective data dynamics. In order to expand our primary finding into a multiuser situation, where TPA may carry out numerous auditing jobs concurrently, we investigate the concept of bilinear aggregate signature.

III. Methods and Materials

Distributed storage systems [17, 18] employ network coding methods to disperse client data among numerous servers. However, its primary goal is to lessen the amount of time spent on repairs if several servers go down at once. By contrast, we investigate whether or not we can use the algorithms in an SNC protocol to build a safe and effective cloud storage system for changing data (for a single storage server). Append-only data (where new data related to a data file are placed only at the end of the file) find several uses, even though dynamic data are general in the sense that they permit arbitrary update (insertion, deletion, and modification) actions. These programmes typically preserve historical records while also keeping current records by appending the latter to the former. CCTV footage, financial ledgers, medical records, append-only database information, and so on are all examples of append-only data. Additional log structures may benefit from append-only data storage as well (e.g., certificates are stored using append-only log structures in certificate transparency schemes [39]). Many such uses call for a cloud server to safely store the massive amounts of data, with append as the sole allowed update method. In this case, the system investigates the feasibility of offering a general construction of a DSCS protocol from any SNC protocol, even if secure cloud storage strategies for generic dynamic data work for append-only data as well. We provide a thorough breakdown of the difficulties inherent in a generic design and single out a few SNC protocols well-suited for use in creating effective DSCS procedures.

By starting with an SNC protocol, the proposed approach generates a publicly verifiable DSCS protocol (DSCS I) [15]. With DSCS I, clients may easily make changes (including insertions, deletions, and modifications) to their data that has been outsourced. The benefits and drawbacks of DSCS I are discussed, both in terms of its (asymptotic) performance and its practical applicability. The formal security specification of a DSCS protocol and proof of the security of DSCS I am provided by the proposed s system. DSCS I (which is based on [15]) may be used to append-only data since it treats them as a particular form of generic dynamic data. While we do find a few SNC protocols that aren't well suited to constructing secure

ISSN: 2366-1313

cloud storage for general dynamic data, we also show that they can be used to create efficient secure cloud storage protocols for append-only data. Using an SNC protocol described by Boneh et al. [13], we develop a publicly verifiable secure cloud storage protocol (DSCS II) for append-only data.

IV. Security Model

DSCS I offer the guarantee of dynamic provable data possession (DPDP) [5]. The untrusted server (acting as a PPT adversary A) can be malicious exhibiting Byzantine behavior and corrupt the client's data arbitrarily, i.e., it can apply updates of its choice. The data possession game between a challenger C and A is as follows. • C runs KeyGen to generate (sk, pk) and gives pk to A. A selects a file F associated with the identifier fid to store. C processes F to form another file F 0 with the help of sk and returns F 0 to A. C stores only some metadata to verify the future updates. • A adaptively chooses a sequence of operations defined by {opi}1≤i≤q1 (q1 is polynomial in the security parameter λ), where opi is an authenticated read, an authenticated update (write) or an audit. C executes these operations on the file stored by A. For an update operation defined by (updtype, info), C verifies the proof (sent by A) by running VerifyUpdate and updates her metadata if and only if the proof passes verification. A is notified about the result of verification for each opi . A can corrupt the file in an arbitrary way during the execution of these operations, i.e., it can update any part of the file that need not be the same as those specified in {opi}1≤i≤q1 . • Let F ∗ be the final state of the file after q1 operations. C has the latest metadata for the file F ∗ . C challenges A with a random challenge set Q, and A returns a proof T = (T1, T2) to C. A wins the game if the proof passes verification. C can challenge A for q2 (polynomial in λ) times to extract (at least) the challenged vectors of F ∗

V. Performance Analysis of DSCS I

DSCS Effectiveness Exponentiation costs account for the bulk of DSCS I's computational price tag (modulo N). The client must execute a multi-exponentiation [38] and take the e-th root of the result in order to create x in the tag for a vector. There are two multiplications by exponents needed by the server in order to compute x. A verifier using Verify must do both a multi-exponentiation and a single-exponentiation in order to validate a proof (see Eqn. 4). The features of a skip list [34] mean that it is very likely that the time needed to construct a proof of concept (connected to a rank-based authenticated skip list) and the time needed to

ISSN: 2366-1313

validate that proof of concept are both O(log m). To further understand how DSCS I stack up against other PDP schemes, we compared it using audit-related metrics in Table 1.

Table 1: Comparison among secure cloud storage protocols achieving PDP guarantees

Secure cloud storage	Type of Data	Computation for verifier	Computation for server	Communication complexity	Public verifiability	Security model
P [2]	Static	O(1)	O(1)	O(1)	Yes	Random oracle model [36]
Scalable PDP [23]	Dynamic	O(1)	O(1)	O(1)	No	Random oracle model
DPDP I [5]	Dynamic	O(log m~)	O(log m~)	O(log m~)	Yesx	Standard model
DPDP II [5]	Dynamic	O(log m~)	O(m~ log m~)?	O(log m~)	Yesx	Standard model
Wang et al. [6]	Dynamic	O(log m~)	O(log m~)	O(log m~)	Yes	Random oracle model
Wang et al. [24]	Dynamic	O(log m~)	O(log m~)	O(log m~)	Yes	Random oracle model
FlexDPDP [37]	Dynamic	O(log m~)	O(log m~)	O(log m~)	Yesx	Standard model
Chen et al. [16]	Static	O(1)	O(1)	O(1)	Yes	Standard model
DSCS I (in this work)	Dynamic	O(log m)	O(log m)	O(log m)	Yes	Standard model
DSCS II (in this work)	Dynamic{	O(1)	O(1)	O(1)	Yes	Random oracle model

We highlight that the current secure cloud storage technique for static data [16], which is based on the same SNC protocol [15], also has the first two drawbacks. This work, however, investigates the possibility that an SNC protocol may be converted into a DSCS protocol. Better DSCS protocols may be derived from more space-efficient SNC protocols. Following this, we propose a new, more efficient DSCS protocol (DSCS II) for append-only data.

ISSN: 2366-1313

DSCS II's Effectiveness The client must execute a multi-exponentiation in order to create the value of an authentication tag for every vector t while performing the Outsource algorithm (see Eqn. 5). To determine t, the server needs to do a single multiplication by exponent (see Eqn. 7 in the algorithm Prove). In order to verify a proof using the method Verify, the verifier must do two multi-exponentiations and two pairing operations (see Eqn. 8). It is important to note that in DSCS II, all three parameters are fixed (independent of m): the proof size, the time needed to create a proof, and the time needed to verify a proof. Table 1 displays the results of a variety of audits using DSCS II and various efficiency metrics.

VI. Evaluation Methodology

We use a 2.5 GHz Intel i5 CPU and 8 GB of RAM to quantify server-side storage costs, client-side communication costs, and calculation costs. OpenSSL 1.0.2 [40] is used for DSCS I cryptographic operations, whereas the PBC 0.5.14 [41] library is used for DSCS II cryptographic operations. DSCS I and DSCS II employ different libraries in their implementations; hence the time it takes to perform cryptographic operations that are otherwise comparable (such as random number generation) differs across the two protocols. Our tests show that m vectors (or data blocks) of size n 0 are stored in a single file. One thing to keep in mind is that under DSCS I and DSCS II, each data block is made up of n data segments. Thus, n 0 = n × sseg, where sseg is the size of each data segment. In our tests, we always use a block size of 500 KB (n 0); hence the number of blocks changes with file size (m). We experiment with file sizes 1, 10, 50, 200 and 500 MB, except in the comparison with [16] given. We take the security parameter $\lambda = 112$ which is same as that considered in [16]. Each experiment was repeated 50 times to get the average findings presented below. Here, we just account for the number of tags and the amount of proofs when calculating the total cost of communication during an update (and audit). We point out that these algorithms must also share a block of information (of constant size). We ignore the block size for the sake of simplicity in making comparisons.

1. Experimental Results for DSCS I

N = pq is a 2048-bit RSA modulus that gives 112-bit security, where p and q are 1024-bit primes. The client incurs a continuous storage cost to keep her private key and certain metadata (the root-digest of the rank-based authenticated skip list) in memory. However, the server must also save the data file and all the authentication data, such as the skip list and the

ISSN: 2366-1313

tags. This extra data storage space is shown. We see that the proportion of extra storage is essentially constant as the file size rises, assuming a fixed block size n0. As a result, even after accommodating dynamic activities, there is a negligible storage burden on the server. Communication Cost: During an audit, the communication cost relies on the number of challenged blocks (i.e., jQj) which is a tiny constant. When a client makes a query, the server will respond with proofs. The proof size is proportional to the sum of the sizes of the blocks being searched and the size of a constant aggregated block (together with the size of an aggregated tag). Due to the second component, the proof size might shift depending on the number of blocks that were queried. For the sake of our comparison of DSCS I's performance to that of [16], we do not include the block size into the total cost (similar to [16]). We see that DSCS I uses a very little amount of bandwidth despite its data-dynamic nature.

The amount of money spent on messages during updates varies with the kind of update being performed. For an insertion, the client transmits to the server an index, a public parameter h, the new data block together with its tag. For a modification, client just has to submit an index, the changed block and its tag. For a deletion, communication comprises the index of the block to be erased. The server provides the client with a validated update at regular intervals. On each iteration, the communication cost is recorded. The Price of Computing: We present the calculation cost for the following stages of DSCS I: outsourcing (client), challenge creation (client), proof generation (server), proof verification (client), and updates on the outsourced file (client and server) (client and server). The time for outsourcing comprises breaking the file into chunks, tag calculation and generating a skip list.

It's clear that the first time you outsource a data file; it's going to cost you a lot of processing resources. It expands as the file size rises. It relies on numerous aspects including the block size n0, the number of blocks m in a file and the number of segments n in each block. If we have a data file with segments of varying sizes, and assume that n0 is big, then the amount of time required to generate a single tag will grow proportionally with the number of segments in the block (i.e., more components in each vector). However, if n0 is assumed to be small, the time required to compute all the tags (and create the skip list) grows exponentially with the rise in the number of blocks m and the number of tags. Accordingly, a good value strikes a balance between them.

Due to the lack of computing power required, the time required to generate challenges is minimal. The number of blocks searched rather than the size of the file is what determines how long it takes to generate a proof (jQj). The time required to generate an aggregated block

ISSN: 2366-1313

(and tag) and the time required to compute the skip-list evidence for each challenged block are both factored in. Time spent confirming proofs include checking the skip-list proof for each challenged block, as well as comparing the aggregated block with the aggregated tag.

VII. Conclusion

The work presented here proposes a secure cloud storage protocol for dynamic data (DSCS I) that makes use of secure network coding (SNC). We believe this to be the first publicly verifiable and standard-model-secure SNC-based DSCS protocol. We have covered some of the difficulties encountered in developing a functional DSCS protocol from a simple SNC one. We have also discovered some shortcomings of a secure cloud storage mechanism for changing data that is based on SNC. Nonetheless, the underlying SNC protocol does impose some of these constraints. It is possible that a more effective SNC protocol will lead to a more effective DSCS protocol. We have also created an effective DSCS protocol (DSCS II) for append-only data and found certain SNC methods that work well with this kind of data. As we've seen, DSCS II is able to remedy a few problems with the original DSCS. Finally, to demonstrate their viability, we have presented prototype implementations of DSCS I and DSCS II and compared the performance of DSCS I to that of an SNC-based secure cloud storage for static data and that of DPDP I.

VIII. References

[1] B. Sengupta and S. Ruj, “Publicly verifiable secure cloud storage for dynamic data using secure network coding,” in ACM Asia Conference on Computer and Communications Security, 2016, pp. 107– 118.

[2] G. Ateniese, R. C. Burns, R. Curtmola, J. Herring, L. Kissner, Z. N. J. Peterson, and D. X. Song, “Provable data possession at untrusted stores,” in ACM Conference on Computer and Communications Security, 2007, pp. 598–609.

[3] A. Juels and B. S. Kaliski, “PORs: Proofs of retrievability for large files,” in ACM Conference on Computer and Communications Security, 2007, pp. 584–597.

[4] H. Shacham and B. Waters, “Compact proofs of retrievability,” Journal of Cryptology, vol. 26, no. 3, pp. 442–483, 2013.

[5] C. C. Erway, A. Kupc¸ ¨ u, C. Papamanthou, and R. Tamassia, ¨ “Dynamic provable data possession,” ACM Transactions on Information and System Security, vol. 17, no. 4, pp. 15:1–15:29, 2015.

ISSN: 2366-1313

[6] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 5, pp. 847–859, 2011.

[7] D. Cash, A. Kupc¸ ¨ u, and D. Wichs, "Dynamic proofs of ¨ retrievability via oblivious RAM," in EUROCRYPT, 2013, pp. 279–295.

[8] E. Shi, E. Stefanov, and C. Papamanthou, "Practical dynamic proofs of retrievability," in ACM Conference on Computer and Communications Security, 2013, pp. 325–336.

[9] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, "Network information flow," IEEE Transactions on Information Theory, vol. 46, no. 4, pp. 1204–1216, 2000.

[10] S. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," IEEE Transactions on Information Theory, vol. 49, no. 2, pp. 371–381, 2003.

[11] S. Agrawal and D. Boneh, "Homomorphic MACs: MAC-based integrity for network coding," in International Conference on Applied Cryptography and Network Security, 2009, pp. 292–305.

[12] D. X. Charles, K. Jain, and K. E. Lauter, "Signatures for network coding," International Journal of Information and Coding Theory, vol. 1, no. 1, pp. 3–14, 2009.

[13] D. Boneh, D. M. Freeman, J. Katz, and B. Waters, "Signing a linear subspace: Signature schemes for network coding," in International Conference on Practice and Theory in Public Key Cryptography, 2009, pp. 68–87.

[14] R. Gennaro, J. Katz, H. Krawczyk, and T. Rabin, "Secure network coding over the integers," in International Conference on Practice and Theory in Public Key Cryptography, 2010, pp. 142–160.

[15] D. Catalano, D. Fiore, and B. Warinschi, "Efficient network coding signatures in the standard model," in International Conference on Practice and Theory in Public Key Cryptography, 2012, pp. 680–696.

[16] F. Chen, T. Xiang, Y. Yang, and S. S. M. Chow, "Secure cloud storage meets with secure network coding," in IEEE International Conference on Computer Communications, 2014, pp. 673–681.

ISSN: 2366-1313

[17] A. G. Dimakis, B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," IEEE Transactions on Information Theory, vol. 56, no. 9, pp. 4539–4551, 2010.

[18] K. Omote and T. T. Phuong, "DD-POR: Dynamic operations and direct repair in network coding-based proof of retrievability," in International Computing and Combinatorics Conference, 2015, pp. 713–730.

[19] "Secure cloud storage," 2018. https://github.com/akankshadixit/ Secure Cloud Storage

[20] S. D. Galbraith, K. G. Paterson, and N. P. Smart, "Pairings for cryptographers," Discrete Applied Mathematics, vol. 156, no. 16, pp. 3113–3121, Sep. 2008.

[21] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, J. Shi, ´ and B. Leong, "A random linear network coding approach to multicast," IEEE Transactions on Information Theory, vol. 52, no. 10, pp. 4413–4430, 2006.

[22] T. Ho, R. Koetter, M. Medard, D. R. Karger, and M. Effros, "The ´ benefits of coding over routing in a randomized setting," in IEEE International Symposium on Information Theory, 2003, p. 442.

[23] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in International Conference on Security and Privacy in Communication Networks, 2008, p. 9.

[24] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," IEEE Transactions on Computers, vol. 62, no. 2, pp. 362–375, 2013.

[25] F. J. MacWilliams and N. J. A. Sloane, The Theory of Error-Correcting Codes. North-Holland Publishing Company, 1977.

[26] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," Journal of the Society for Industrial and Applied Mathematics, vol. 8, no. 2, pp. 300–304, 1960.

[27] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of retrievability: Theory and implementation," in ACM Cloud Computing Security Workshop, 2009, pp. 43–54.

ISSN: 2366-1313

[28] Y. Dodis, S. P. Vadhan, and D. Wichs, “Proofs of retrievability via hardness amplification,” in Theory of Cryptography Conference, 2009, pp. 109–127.

[29] E. Stefanov, M. van Dijk, A. Juels, and A. Oprea, “Iris: A scalable cloud file system with efficient integrity checks,” in Annual Computer Security Applications Conference, 2012, pp. 229–238.

[30] N. Chandran, B. Kanukurthi, and R. Ostrovsky, “Locally updatable and locally decodable codes,” in Theory of Cryptography Conference, 2014, pp. 489–514.

[31] F. Armknecht, J. Bohli, G. O. Karame, Z. Liu, and C. A. Reuter, “Outsourced proofs of retrievability,” in ACM Conference on Computer and Communications Security, 2014, pp. 831–843.

[32] K. D. Bowers, A. Juels, and A. Oprea, “HAIL: A high-availability and integrity layer for cloud storage,” in ACM Conference on Computer and Communications Security, 2009, pp. 187–198.

[33] R. C. Merkle, “A digital signature based on a conventional encryption function,” in CRYPTO, 1987, pp. 369–378.

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

[35] N. Baric and B. Pfitzmann, “Collision-free accumulators and fail- ´ stop signature schemes without trees,” in EUROCRYPT, 1997, pp. 480–494.

[36] M. Bellare and P. Rogaway, “Random oracles are practical: A paradigm for designing efficient protocols,” in ACM Conference on Computer and Communications Security, 1993, pp. 62–73.

[37] E. Esiner, A. Kachkeev, S. Braunfeld, A. Kupc¸ ¨ u, and ¨ O. ¨ Ozkasap, ¨ “FlexDPDP: FlexList-based optimized dynamic provable data possession,” 2013. http://eprint.iacr.org/2013/645

[38] B. Moller, “Algorithms for multi-exponentiation,” in ¨ International Conference on Selected Areas in Cryptography, 2001, pp. 165–180.

ISSN: 2366-1313

[39] B. Laurie, A. Langley, and E. Kasper, "Certificate transparency," June 2013. https://tools.ietf.org/html/rfc6962

[40] OpenSSL, "Cryptography and SSL/TLS toolkit," 2018. https:// www.openssl.org/

[41] PBC Library, "The pairing-based cryptography library," 2006. https://crypto.stanford.edu/pbc/