

Currency and Correctness of Content in Object Storage Networks

Organization:

The Johns Hopkins University
3400 N. Charles St.
Baltimore, MD USA 21218

Technical Contacts:

Randal Burns
224 New Engineering Bldg., Baltimore, MD USA 21218
410-516-7708 (work), 410-493-6312 (cell)
randal@cs.jhu.edu

Giuseppe Ateniese
224 New Engineering Bldg., Baltimore, MD USA 21218
410-516-8051
ateniese@cs.jhu.edu

Administrative Contact:

Laura Graham
224 New Engineering Bldg., Baltimore, MD USA 21218
410-516-8577
lgraham@jhu.edu

Topic areas:

1. Information consistency maintenance in distributed, replicated, storage networks
2. Secure and shareable versioned objects

Author will attend workshop if invited.

Executive Summary

For large-scale replicated and distributed object systems to be useful, they must provide lightweight mechanisms to ensure that applications operate on the correct data—that they are accessing the most recent version and that the data have been stored and reproduced faithfully. This is particularly challenging in object systems in which we perform analysis at the data, because the application never witnesses the object data directly. Rather, it only sees results derived from those objects. At the same time, performing computation at the data is essential to achieve scale, because it avoids transmitting large amounts of data across global-scale networks.

We focus on providing strong guarantees on the currency and correctness of data in adversarial networks in which storage sites and network links may exhibit arbitrary failures and act maliciously in producing and transmitting false data. At the same time, we eschew protocols that require synchronous collaboration among multiple parties, such as Byzantine agreement. Instead, we rely on the lazy, late-binding of data to trusted sites with strong identities (authentication), i.e. establish the correctness and recency of data prior to its use. This decision will allow the system to scale to the thousands of network participants necessary for defense applications.

Provable data possession (PDP) [1] transforms a distributed systems ability to efficiently verify massive data sets on distributed object stores. We use PDP as the fundamental concept to provide data integrity and correctness. PDP allows a client application to verify data objects on remote stores *without transferring data to the client and without having the store access the entire object*. It does so using very small (constant) amounts of metadata and network bandwidth, and by performing little computation at both the application and store.

The interest in PDP is quite new and currently PDP has operational shortcomings that the research community needs to tackle. We will investigate extending PDP to address *dynamic data* in which the content changes and *dynamic replication* in which the number of copies changes. We will also extend PDP to be *publicly verifiable* in which any party can verify the correctness of data and *privacy preserving* in which challenges reveal nothing about the content of the data. As of now, many of these goals can be realized in isolation, but no unifying framework provides all together. Also, verification is efficient, but pre-processing the data for storage is less so and limits the performance of PDP systems. We will engineer PDP pre-processing to improve performance by an order of magnitude through multi-core parallelism and cryptographic optimizations.

To support changing objects and object versions in PDP, we need to efficiently distribute metadata updates and revoke stale metadata. This can be done through authenticated directories [12], but such tree-based indexes perform poorly when they grow larger than available memory and are distributed across many sites. We have prototyped optimizations to authenticated directories that improve the performance by a factor of three. These rely on the optimistic replication of authentication information to clients outside of the directory and fast evaluation of authentication metadata using the replicated information.

In summary, we describe the systems and security research needed to translate the benefits of PDP technology to distributed object systems that perform analysis at the data. PDP makes the verification of massive petascale data sets possible. We extend this to changing and versioned data in large-scale distributed object systems and we will build the practical protocols and systems to realize PDP in global-scale storage networks.

Taking Analysis to the Data

Large-scale distributed data systems are, by necessity, evolving to a model in which computation is performed at the data storage. Data networks are collecting massive amounts of data and running distributed analysis against all of this data. Ten years of negative experience with grid data systems dictates that we cannot transfer multiple large data sources to a single site where we compare and join them. Delivering computation to the data has been a guiding principle in the systems built by the PIs [7, 5] (see Relevant Experience p.5) that perform data-intensive analysis queries against large (30+ terabyte) data sets. The critical need for distributed object storage reflects the same principles; object storage defines a richer computational environment at the stores based on invoking object methods.

The model of remote computation makes it much more difficult to achieve guarantees about the currency and correctness of the data. An application that performs a function against a file or database never witnesses the data directly. Thus, it cannot verify the data using standard techniques, such as checking a digital signature.

We describe techniques for remote data checking that allow clients and applications to provably verify the correctness of data remotely, i.e. without transferring the data back to the client. These methods scale efficiently to large datasets, leading to practical protocols and systems implementations. Our design matches well the constraints of distributed object storage networks in that they avoid the transfer of large amounts of data across networks, performing the bulk of the computation at the data store.

Provable Data Possession

Provable data possession (PDP) is a remote data checking scheme. It allows a data producer to outsource the storage of a data object to a (possibly untrusted) server. Later, an application (not necessarily the producer) issues challenges to the server in which the server proves that it possesses the correct data. Remote describes that PDP allows an application to verify the data without retrieving the information.

PDP employs a novel combination of random spot checking of data and composite, homomorphic signatures. Figure 1 shows the operation of PDP's remote data checking during the challenge phase. In spot checking, an application (client) selects a random subset of data from an object. The server computes and returns a proof that shows it possesses that random subset. The proof of possession extends probabilistically to the entire file. The server cannot delete any substantial fraction of data without the client detecting the deletion. In fact, we use erasure coding to strengthen this guarantee. We perform PDP on an erasure coded file so that that the server cannot effectively damage any data in the original file without damaging much more data in the erasure encoded file. This strengthens the possession guarantee to arbitrarily high probabilities, e.g. 0.99999 is achievable with practical parameters.

PDP introduced the notion of homomorphic verifiable tags, which we call signatures (s_i associated with each block b_i). The homomorphic signatures make it so that a single composite signature can verify the possession of any subset of blocks. The store combines the data (through modular addition) and builds a composite signature (through multiplication and a single exponentiation) that verifies the entire random subset.

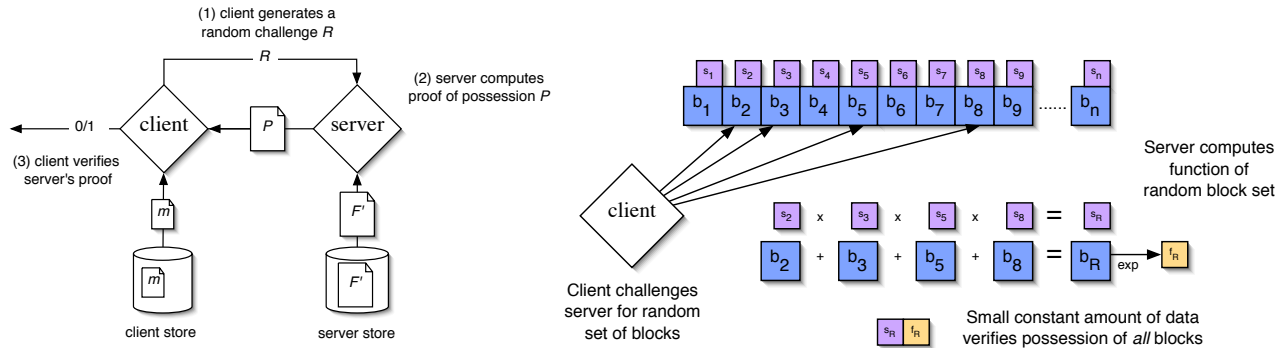


Figure 1: PDP verification network protocol (left) and computation (right)

The combination of homomorphic signatures and spot checking makes PDP scale to massive data sets. Homomorphic signatures are only marginally more expensive than signing a single data block, because they use a single exponentiation for all data blocks. They also make the proof small, a constant size for any number of blocks and the proof can be verified with a constant amount of metadata. Both the proof and metadata are 200 bytes. Spot checking limits the amount of I/O that the store performs: a constant amount for any file size.

While quite counter-intuitive, PDP shows that public-key cryptography can be used efficiently with massive data sets. PDP's signatures work in the RSA security model, but variants can be implemented with discrete log or bi-linear maps [4]. Efficiency lies in amortizing the public-key operations across many blocks through the homomorphism (composition of data). PDP also realizes the benefits of public-key cryptography, such as provable security, ease of key management, and public verifiability.

PDP Extensions

PDP and related constructs [8, 10, 11] have sparked an interest in the research community and many works have explored how to enhance the semantics or performance of remote data checking. We highlight those extensions that make PDP more usable in replicated distributed object systems.

Dynamic PDP supports incremental updates to a PDP encoded file. The updates need to both validate the new version of the data and invalidate the old version. The challenge lies in supporting these updates succinctly, i.e. without having the file size or metadata grow linearly in the number of updates. We have developed one such scheme in a symmetric key variant of PDP [2].

Multiple-replica PDP extends the possession guarantee to cover multiple replicas, ensuring that there are k unique replicas that occupy kn storage for files of size n [6]. It does so with a single set of homomorphic tags, which reduces metadata and requires signatures to be generated only once. Multiple-replica PDP prevents colluding servers from storing only a single copy.

Public verification makes all metadata publicly available so that any party can verify that stores possess data. This feature is essential in distributed systems because it allows for the public distribution of metadata. Without this feature, PDP metadata must be kept secret. In distributed systems,

secrets are fragile and difficult to manage [9]. They require trust relationships to be pre-established and long lasting, which inhibits re-configurability and adaptation.

Privacy preserving PDP makes challenges reveal nothing about the content of the data to parties involved. Shah et al. [11] have explored this feature in data checking protocols that are not remote, i.e. that transfer data back to the client. Initial evidence suggests that their techniques translate to PDP protocols. This is a nice complement to public verifiability, but may be unnecessary if other privacy constructs are in place, e.g. cryptography.

No single PDP system meets all of these requirements. For example, most extensions leverage properties of public-key (in our case RSA) cryptosystems. However, dynamic data support to date relies on symmetric key PDP-like constructs. It remains an open question to find a dynamic, publicly verifiable PDP scheme. For now, we are actively seeking a dynamic version of PDP in the RSA model, because public verifiability cannot be achieved outside of public-key cryptosystems.

Engineering Fast PDP Systems

Given the fast verification of PDP, pre-processing files represents the performance limitation. In pre-processing, the client computes the signatures associated with each block of data. Each block has its own tag, which requires an exponentiation. Currently, pre-processing rates are hundreds of KBs to a few MBs per second per processor [1], depending upon the parameters of the encoding. The improvements we will undertake will allow PDP systems to generate data at network bandwidth speeds in high-performance systems, i.e on the order of 100Mbps to 1Gbps.

We will improve the performance of pre-processing by orders of magnitude through the use of processor-level parallelism and cryptography optimizations. Through the poor use of multiple cores and vector processing, modern processors realize only a fraction of their potential. Database applications often use 3-10% of available cycles [3]. PDP pre-processing can be done in a SIMD model compatible with processor vector architectures and stream processors in order to better utilize available cycles. Also, in pre-processing the data producer knows the RSA field and can implement exponentiation using the Chinese remainder theorem, which should provide a 2-3 times speedup.

Optimized Authenticated Directories

With dynamic and publicly verifiable PDP, a distributed replicated object store will create multiple versions of data and modify existing data, and all of these objects will be remotely verifiable. This introduces new challenges in that the system needs to ensure that applications are accessing valid and current data. Distributing updated currency information every time an object changes is a tremendously complex task.

Authenticated directories offer a solution to the currency problem, reducing the scale of the information that needs to be kept current. An authenticated directory associates current versions of objects in a Merkle (hierarchical-hash) tree so that the root of the tree uniquely and securely identifies all subordinate objects. To achieve currency, the system needs only keep a single piece of metadata, the authenticated directory root, consistent and up to date. Applications ensure the currency of individual objects by acquiring the root and then performing membership queries to

verify that objects are part of the authenticated directory defined by that root. Proof of membership is a hash chain or a Merkle-tree *path* from a leaf node in the tree to the root

The problem with authenticated directories lies in their performance as they become large. Tree based indexes perform poorly when they are larger than system memory [13], because lookups at each level of the tree require I/O. For large authenticated directories, the system may perform multiple I/Os to authenticate a data object and then perform only a single I/O to access the object.

We have optimized a distributed authenticated directory service to reduce or eliminate I/O entirely.

1. We replicate authentication paths from the Merkle tree in the object metadata so that clients can often validate objects without consulting the index.
2. Clients use these embedded paths in part, consulting only the top levels of the index when the index has not changed at lower levels.
3. We then dynamically cache the upper levels of the Merkle-tree at the client, again to avoid consulting the index.
4. At the directory server, we make the Merkle-tree index cache friendly by isolating updates to a “hot” portion of the Merkle tree that represents a small amount of data that fits in cache.
5. Data in the cold portion of the Merkle tree changes infrequently, which makes the client optimizations more likely to succeed.
6. Finally, when clients do consult the authenticated directory, because the embedded path is stale, they update the embedded path for future currency queries.

These optimizations are never incorrect; they reduce I/O when a partial path does not change and, when the optimizations fail, the client consults the authoritative index at the directory service.

In our prototype, these optimizations improve the performance of authenticated directories by a factor of three, reducing the average query time from 1.1 ms to 400 μ s.

Relevant Experience in Building Large Object Stores

Randal Burns has been building the distributed data systems that allow scientists to make discoveries through the exploration of vast amounts of data and the archival systems that preserve critical scientific data for the future. The challenges in managing large scientific data include the size of the data (terabytes to petabytes), the global distribution of storage sites, the longevity of the data in archival stores, and ensuring correctness when storing data at untrusted sites. Examples include the JHU Turbulence Database Cluster [7], which provides community access to high-resolution turbulence simulations. The service is used by scientists internationally and has serviced more than 10 billion particle tracking queries initiated from three continents. Burns’ group has also built the Chesapeake Bay Environmental Observatory testbed [5], which allows users to compare and join more than 15 model-generated and observational data sources. These systems are built in close collaboration with the scientists that use them, which improves the relevance and timeliness of our research results and guarantees that our efforts are practical, usable, and benefit the intended users.

References

- [1] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song. Provable data possession at untrusted stores. In *ACM Computer and Communications Security*, 2007.
- [2] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik. Scalable and efficient provable data possession. *ePrint Archive Report*, (2008/114), 2008.
- [3] P. Boncz, M. Zukowski, and N. Nes. Monetdb/x100: Hyper-pipelining query execution. In *Conference on Innovative Data Systems Research*, 2007.
- [4] K. Bowers, A. Juels, and A. Oprea. Proofs of retrievability: Theory and implementation. *ePrint Archive Report*, (2008/175), 2008.
- [5] The Chesapeake Bay Environmental Observatory. <http://ccmp.chesapeake.org/CBEO/>, accessed 04/05/2008.
- [6] R. Curtmola, O. Khan, R. Burns, and G. Ateniese. Mr-pdp: Multiple-replica provable data possession. In *International Conference on Distributed Computing Systems*, 2008.
- [7] The JHU Turbulence Database Cluster. <http://turbulence.pha.jhu.edu>, accessed 04/05/2008.
- [8] A. Juels and B. S. Kaliski. PORs: Proofs of retrievability for large files. In *ACM Computer and Communications Security*, 2007.
- [9] P. Maniatis, M. Roussopoulos, T. Giuli, D. Rosenthal, M. Baker, and Y. Muliadi. Preserving peer replicas by rate-limited sampled voting. In *Symposium on Operating Systems Principles*, 2003.
- [10] T. S. J. Schwarz and E. L. Miller. Store, forget, and check: Using algebraic signatures to check remotely administered storage. In *International Conference on Distributed Computing Systems*, 2006.
- [11] M. A. Shah, R. Swaminathan, and M. Baker. Privacy-preserving audit and extraction of digital contents. *ePrint Archive Report*, (2008/186), 2008.
- [12] A. Y. Yumerefendi and J. Chase. Strong accountability for network storage. In *Conference on File and Storage Technologies*, 2007.
- [13] B. Zhu, K. Li, and H. Patterson. Avoiding the disk bottleneck in the data domain deduplication file system. In *Conference on File and Storage Technologies*, 2008.