Outsourcing Durability

Peter Williams
Stony Brook Network Security and Applied Cryptography Lab
Stony Brook, NY 11794-4400
petertw@cs.sunysb.edu

Radu Sion
Stony Brook Network Security and Applied Cryptography Lab
Stony Brook, NY 11794-4400
sion@cs.sunysb.edu

Rimmi Devgan
Stony Brook Network Security and Applied Cryptography Lab
Stony Brook, NY 11794-4400
rdevgan@cs.sunysb.edu

ABSTRACT

We introduce a new paradigm for outsourcing the durability property of a multi-client transactional database to an untrusted service provider. Specifically, we enable untrusted service providers to support transaction serialization, backup and recovery for clients, with full data confidentiality and correctness. Moreover, providers learn nothing about transactions (except their size and timing), thus achieving read and write access pattern privacy.

We build a proof-of-concept implementation of this protocol for the MySQL database management system, achieving tens of transactions per second in a two-client scenario with full transaction privacy and guaranteed correctness. This shows the method is ready for production use, creating a novel class of secure database outsourcing models.

1. INTRODUCTION

Increasingly, data management is outsourced to third parties. This trend is driven by growth and advances in cheap, high-speed communication infrastructures as well as by the fact that the total cost of data management is 5–10 times higher than the initial acquisition costs [11]. Outsourcing has the potential to minimize client-side management overheads and benefit from a service provider's global expertise consolidation and bulk pricing. Providers such as Yahoo [9], Amazon [2, 3, 1], Google [4], Sun [8] and others – ranging from corporate-level services such as IBM Data Center Outsourcing Services [5] to personal level database hosting [7, 6] – are rushing to offer increasingly complex storage and computation services.

Yet, significant challenges lie in the path of a successful large-scale adoption. In business, health care and government frameworks, clients are reluctant to place sensitive data under the control of a remote, third-party provider, without practical assurances of *privacy* and *confidentiality*, yet these services are fundamentally insecure and vulnerable to illicit behavior.

Existing research addresses several important outsourcing aspects, including direct data retrieval with access privacy, searches on encrypted data, and techniques for querying remotely-hosted encrypted structured data in a unified client model [10, 13]. These efforts are based on the assumption that, to achieve confidentiality, data will need to be encrypted before outsourcing to an untrusted provider. Once encrypted however, inherent limitations in the types of primitive operations that can be performed on encrypted data by untrusted hosts lead to fundamental query expressiveness constraints. Specifically, reasonably practical mechanisms exist only for simple selection and range queries or variants thereof.

In this poster, we introduce an orthogonal thesis spurred by the advent of cheap and fast disks and CPUs. We believe that in many deployments, individual data clients' systems are in a position to host and access local large data sets with little difficulty at no additional cost.

Holding data locally may appear to contradict the spirit of outsourcing, yet we argue that there is one essential data management aspect that cannot be hosted as such: *transaction processing for multiple concurrent clients.* In a world where client machines have become powerful enough to run local databases, we predict data management outsourcing markets will converge on supplying the transactional, network intensive services and availability assurances which are not trivially achievable locally.

We thus introduce a novel paradigm for solving the data management outsourcing desiderata: *a mechanism for collaborative transaction processing with durability guarantees supported by an untrusted service provider under assurances of confidentiality and access privacy.* In effect, we achieve the cost benefits of standard outsourcing techniques (durability, transaction processing, availability) while preserving the privacy guarantees of local data storage. This is accomplished by enabling data clients to collaboratively perform runtime transaction processing and interact through an untrusted service provider that offers durability and transaction serializability support.

In this context, data outsourcing becomes a setting in which all permanent data is hosted securely encrypted off-site, yet clients access it through their locally-run database effectively acting as a data cache. If local data is lost, it can be retrieved from the offsite repository. Inter-client interaction

and transaction management is intermediated by the untrusted provider who also ensures durability by maintaining a client-encrypted and authenticated transaction log with full confidentiality.

In our model, each client maintains its own cache of (portions of) the database in client-local storage, allowing it to perform efficient reads with privacy, while relieving local system administrators of backup obligations. The key benefit thus becomes achieving data and transaction privacy while (1) avoiding the requirement for persistent client storage (clients are now allowed to fail or be wiped out at any time), and (2) avoiding the need to keep any single client-side machine online as a requirement for availability.

2. MODEL

Provider/Server. The provider owns durable storage, and would like to provide use of this storage for a fee. The provider, being hosted in a well-managed data center, also has high availability. We will investigate ways that clients can make use of these attributes.

Since the provider has different motivations than the clients, we assume an actively malicious provider. However, we do not try to prevent denial of service behavior from the provider.

Clients. In our model, the clients are a set of trusted parties who must run transactions on a shared database with full ACID guarantees. Since storage is cheap, each client has a local hard disk to use as working space; however, due to the fragile nature of hard disks, we do not assume this storage is permanent. Additionally, the clients would like to perform read queries as efficiently as possible without wasting network bandwidth or paying network latency costs. Each of the trusted parties would also like to be able to continue running transactions even when the others are offline, possibly making use of the provider's high availability.

The clients would like to take advantage of the durability of the provider's storage, but they do not trust the provider with the privacy or integrity of their data. Specifically, the provider should be prevented from observing any of the distributed database contents. We define a notion of consistency between the clients' database views to address integrity. It is not imperative that all clients see exactly the same data as the other clients at exactly the same time, however, they need to agree on the sequence of updates applied. We define $trace_{c,i}$ to be the series of the first i transactions applied by client c to its local database copy. Clients c and d are considered i-**trace consistent** if $trace_{c,i} = trace_{d,i}$.

In some scenarios the provider might be able to partition the set of clients, and maintain separate versions of the database for each partition. This partitioning attack has been examined in previous literature; if there are non-intercommunicating asynchronous clients, the best that can be guaranteed is fork consistency, first defined by Mazières and Shasha in [12]. Any adopted solution should guarantee that the data repository is fork consistent; that is, all clients *within each partition* agree on the repository history. This guarantee is not as weak it may appear to be on the surface, as once the provider has created a partition, the provider must block all future communication between partitioned clients, or else the partition will be immediately detected.

We assume that clients do not leak information through transaction timing and transaction size. Clients in real life may vary from this with only minimal loss of privacy, but we use a timing and size side-channel free model for illustration purposes.

3. REFERENCES

[1] Amazon Elastic Compute Cloud. Online at `http://aws.amazon.com/ec2`.

[2] Amazon Simple Storage Service. Online at `http://aws.amazon.com/s3`.

[3] Amazon Web Services. Online at `http://aws.amazon.com`.

[4] Google App Engine. Online at `http://code.google.com/`.

[5] IBM Data Center Outsourcing Services. Online at `http://www-1.ibm.com/services/`.

[6] Inetu.net Managed Database Hosting. Online at `http://www.inetu.net`.

[7] Opendb.com Web Database Hosting. Online at `http://www.opendb.com`.

[8] Sun Utility Computing. Online at `http://www.sun.com/service/sungrid/index.jsp`.

[9] Yahoo Briefcase. Online at `http://briefcase.yahoo.com`.

[10] Premkumar T. Devanbu, Michael Gertz, Chip Martel, and Stuart G. Stubblebine. Authentic third-party data publication. In *IFIP Workshop on Database Security*, pages 101–112, 2000.

[11] Gartner, Inc. Server Storage and RAID Worldwide. Technical report, Gartner Group/Dataquest, 1999. `www.gartner.com`.

[12] David Mazières and Dennis Shasha. Building secure file systems out of byzantine storage. In *PODC '02: Proceedings of the 21st Annual Symposium on Principles of Distributed Computing*, pages 108–117, New York, NY, USA, 2002. ACM Press.

[13] E. Mykletun, M. Narasimha, and G. Tsudik. Authentication and integrity in outsourced databases. In *ISOC Symposium on Network and Distributed Systems Security NDSS*, 2004.