JustMyFriends: Full SQL, Full Transactional Amenities, and Access Privacy

Arthur Meacham
Department of Computer Science
Courant Insitute of Mathematical Sciences
New York University
251 Mercer St
NY, NY 10012
meacham@cs.nyu.edu

Dennis Shasha
Department of Computer Science
Courant Insitute of Mathematical Sciences
New York University
251 Mercer St
NY, NY 10012
shasha@cs.nyu.edu

ABSTRACT

A major obstacle to using Cloud services for many enterprises is the fear that the data will be stolen. Bringing the Cloud in-house is an incomplete solution to the problem because that implies that data center personnel as well as myriad repair personnel must be trusted. An ideal security solution would be to share data among precisely the people who should see it ("my friends") and nobody else.

Encryption might seem to be an easy answer. Each friend could download the data, update it perhaps, and return it to a shared untrusted repository. But such a solution permits no concurrency and therefore no real sharing.

JustMyFriends ensures sharing among friends without revealing unencrypted data to anyone outside of a circle of trust. In fact, non-friends (such as system administrators) see only encrypted blobs being added to a persistent store. JustMyFriends allows data sharing and full transactions. It supports the use of all SQL including stored procedures, updates, and arbitrary queries. Additionally, it provides full access privacy, preventing the host from discovering patterns or correlations in the user's data access behavior.

The demonstration will show how friends in an unnamed government agency can coordinate the management of a spy network in a transactional fashion. Demo visitors will be able to play the roles of station chiefs and/or of troublemakers. As station chiefs, they will write their own transactions and queries, logout, login. As troublemakers, visitors will be able to play the role of a curious observer, kill client processes, and in general try to disrupt the system.

Categories and Subject Descriptors

H.2.0 [**Database Management**]: GeneralSecurity, integrity, and protection; C.2.4 [**Distributed Systems**]: Distributed ApplicationsCloud Computing

General Terms

database outsourcing, privacy

1. MOTIVATION

Despite the growing acceptance of cloud computing, real concerns still linger about the security of data outsourced to the cloud. To use the cloud, one must entrust one's data to an outside party, the cloud provider. Generally, the only assurances of privacy are the word of the provider and the legal and business incentives for the provider not to leak or abuse information.

Even if we trust that the provider is truly scrupulous and motivated to protect user data, there is still the danger of corrupt or sloppy individuals within the provider organization, or of the sort of large-scale data breach that makes the news with alarming frequency[2][1]. Furthermore, data on the cloud is exposed to attacks that did not exist in traditional architectures, as seen in recent, high-profile exploits affecting major infrastructure providers[15][14].

Given the risks, it is understandable that some users might be hesitant or unwilling to move sensitive data to the cloud. JustMyFriends addresses these concerns by enabling the outsourcing of databases in a secure, access-private manner.

2. RELATED WORK

JustMyFriends builds on work first presented in the Blind Stone Tablet[16]. These systems use a client-side approach to privacy, a model also seen in such work as SPORC[4]. In our design, as in the Blind Stone Tablet, the untrusted host provides ACID guarantees and data distribution to clients without ever seeing data in the clear, while all data processing is performed on client machines. Our design also supports all SQL.

There have been a number of other approaches to preserving privacy while accessing data on untrusted servers. These include Private Information Retrieval and 1-out-of-N Oblivious Transfer, strategies that focus on retrieving data without revealing the desired piece of data to the server[3][8][9][12][13]. These solutions are strictly read-only, and can only handle certain types of query.

Another technique that has been explored is Bucketization, in which an encrypted table is partitioned into a number

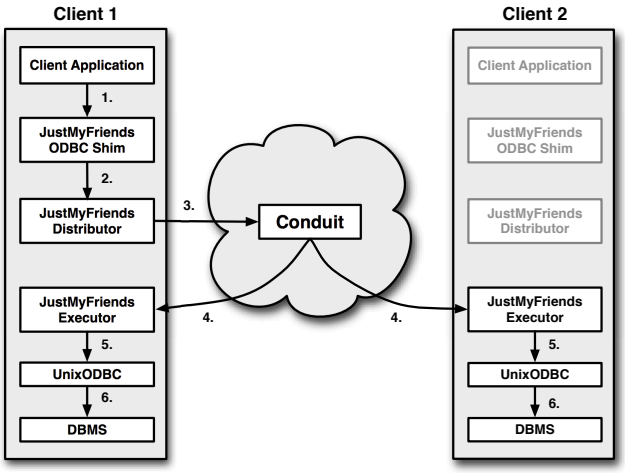

Figure 1: Query Flow: The life of a query from the initiating client to all other clients. (1) The Client application begins a database access via the ODBC API. (2) The ODBC call is captured by JustMyFriends and passed to the Distributor. (3) The Distributor marshalls and encrypts the statement and any arguments or bound data. The encrypted statement description is then forwarded to the Conduit. (4) The Conduit assigns a sequence number to the statement description, appends it to a log, then forwards the sequence number and description to all connected JustMyFriends clients. (5) On each client, the statement description is received by the Executor, where it is decrypted and enqueued for processing. (6) Statements are unmarshalled, data is bound, and the actual call to the unixODBC layer is made. Multistep transactions are executed concurrently, with commits happening in order of sequence number. (7) The actual statement is executed on the local replica of the database. When conflicts occur, the Executor detects and prevents potential inconsistencies. On the initiating client, any results and return values are given to the calling application.

of "buckets", and queries on the encrypted data determine which buckets may contain matching rows[6][7]. Bucketization supports all of SQL, but has a tradeoff between performance (small buckets) and privacy (large buckets), and weakens security in able to support full SQL on encrypted data.

Finally, breakthroughs in fully-homomorphic encryption techniques have been touted as a potential solution to the dangers of hosting data on untrusted servers[5]. While fully-homomorphic encryption is still not practical from a performance standpoint, others have proposed weakening the requirement to somewhat-fully-homomorphic encryption to achieve performance gains[11]. Even if a practical FHE solution is achieved, however, it will not provide access privacy.

3. TECHNICAL OVERVIEW

Following the Blind Stone Tablet, our solution takes advantage of the low cost of disk storage on client machines to

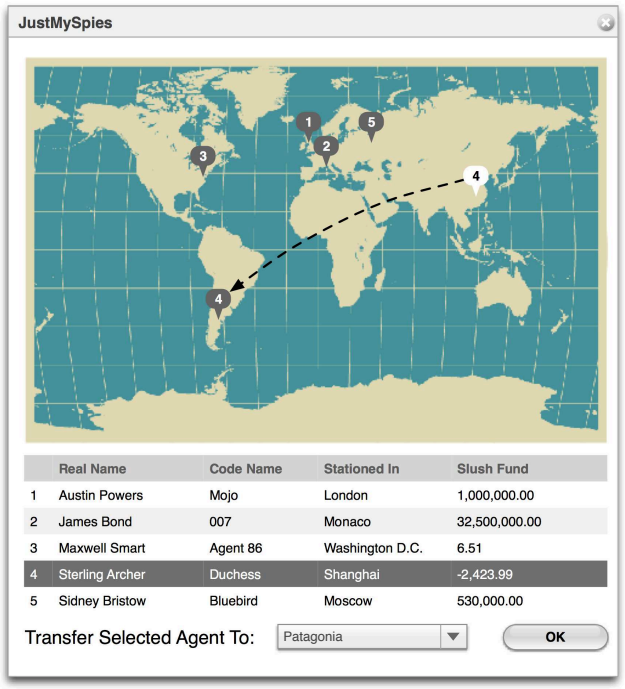

Figure 2: JustMySpies GUI: A location window in which Demo Viewers can see positions of spies and transfer them among stations.

maintain replicas of the database on each trusted client. Before being executed, all database transactions are encrypted and forwarded to a service residing on the cloud, referred to as the conduit. The conduit records the encrypted message in a backup log on the cloud (achieving durability), assigns a sequence number to the transaction, then forwards it to all trusted clients. The clients are able to decrypt the message and apply it to their local replicas of the database. If a new client joins the system, or a client has to recover from a failure, the client can replay the transactions from the log on the cloud. Using ideas presented in SUNDR[10], the system includes mechanisms to detect misbehavior by the conduit, such as fork attacks or data corruption.

For the current implementation, we have modified unixODBC, a standard open source ODBC driver manager. This allows JustMyFriends to work with unmodified application binaries, and with a wide range of database products. When a client program uses the ODBC API to communicate with a DBMS, the call is captured by the driver manager, encrypted, and forwarded to the conduit. Once a database operation has been distributed to the clients by the conduit, it is passed on to each local DBMS for processing.

Transactions run concurrently on each local machine, with commits executed in the order of sequence number. The local DBMS is relied upon to handle deadlock detection. Once a commit has been received, JustMyFriends determines whether a transaction on an individual client has reached a different state than that of the initiating client, generally as a result of a local deadlock caused by a race condition. If such a "disobedient" state is detected, then

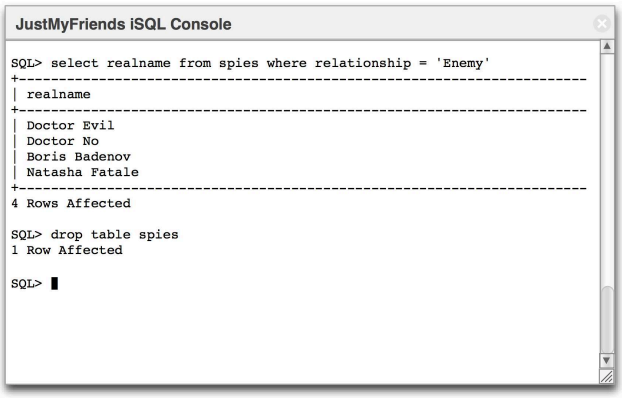

Figure 3: Console: The demo visitor will be able to find spies, move them, change the slush fund value, and perform arbitrary SQL

all affected transactions on the disobedient client are rolled back and executed serially. Once disobedience has been corrected, concurrent processing is resumed.

4. DEMO EXPERIENCE

The demonstration will consist of an application in which various local station chiefs in a fictional spy agency use a shared database to manage agents and resources. Because they want to avoid leaks, but trust each other, they use JustMyFriends to host their shared database.

Multiple client machines will be available to the user, each representing a different country in the spy network. There will also be a machine acting as untrusted conduit, on which the user will be able to examine the data traffic as seen by the outside host.

The demo visitor as a station chief will have the ability to execute pre-written transactions to perform tasks such as transferring spies to new locations or allocating slush funds. Additionally, the demo visitor will be able to write and execute arbitrary SQL, including data definition statements. Finally, the user as troublemaker will have the ability to cause failures or wipe out the local state on some or all of the clients, observe untrusted storage, and then perform a recovery on the affected machines.

5. REFERENCES

[1] Citi breach may have compromised customer data -ft, 2011.

[2] Baker, L. B. Sony suffers second major user data theft, May 2011.

[3] Chor, B., Goldreich, O., Kushilevitz, E., and Sudan, M. Private information retrieval. *Foundations of Computer Science, Annual IEEE Symposium on 0* (1995), 41.

[4] Feldman, A. J., Zeller, W. P., Freedman, M. J., and Felten, E. W. Sporc: group collaboration using untrusted cloud resources. In *Proceedings of the 9th USENIX conference on Operating systems design and implementation* (Berkeley, CA, USA, 2010), OSDI'10, USENIX Association, pp. 1–.

[5] Gentry, C. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st annual ACM symposium on Theory of computing* (New York, NY, USA, 2009), STOC '09, ACM, pp. 169–178.

[6] Hacigümüş, H., Iyer, B., Li, C., and Mehrotra, S. Executing sql over encrypted data in the database-service-provider model. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data* (New York, NY, USA, 2002), SIGMOD '02, ACM, pp. 216–227.

[7] Hore, B., Mehrotra, S., and Tsudik, G. A privacy-preserving index for range queries. In *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30* (2004), VLDB '04, VLDB Endowment, pp. 720–731.

[8] Kushilevitz, E., and Ostrovsky, R. Replication is not needed: single database, computationally-private information retrieval. In *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on* (oct 1997), pp. 364 –373.

[9] Kushilevitz, E., and Ostrovsky, R. One-way trapdoor permutations are sufficient for non-trivial single-server private information retrieval, 2000.

[10] Li, J., Krohn, M., Mazières, D., and Shasha, D. Secure untrusted data repository (sundr). In *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6* (Berkeley, CA, USA, 2004), USENIX Association, pp. 9–9.

[11] Naehrig, M., Lauter, K., and Vaikuntanathan, V. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop* (New York, NY, USA, 2011), CCSW '11, ACM, pp. 113–124.

[12] Naor, M., and Pinkas, B. Oblivious transfer and polynomial evaluation. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing* (New York, NY, USA, 1999), STOC '99, ACM, pp. 245–254.

[13] Naor, M., and Pinkas, B. Efficient oblivious transfer protocols. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms* (Philadelphia, PA, USA, 2001), SODA '01, Society for Industrial and Applied Mathematics, pp. 448–457.

[14] Ristenpart, T., Tromer, E., Shacham, H., and Savage, S. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM conference on Computer and communications security* (New York, NY, USA, 2009), CCS '09, ACM, pp. 199–212.

[15] Somorovsky, J., Heiderich, M., Jensen, M., Schwenk, J., Gruschka, N., and Lo Iacono, L. All your clouds are belong to us: security analysis of cloud management interfaces. In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop* (New York, NY, USA, 2011), CCSW '11, ACM, pp. 3–14.

[16] Williams, P., Sion, R., and Shasha, D. The blind stone tablet: Outsourcing durability tountrusted parties. In *Proceedings of the 16th Annual Network and Distributed System Security Symposium* (2009), NDSS'09.