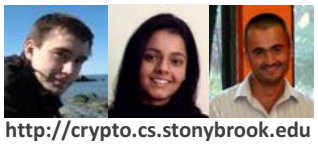
ODP: A System for Outsourcing Durability to Untrusted Parties

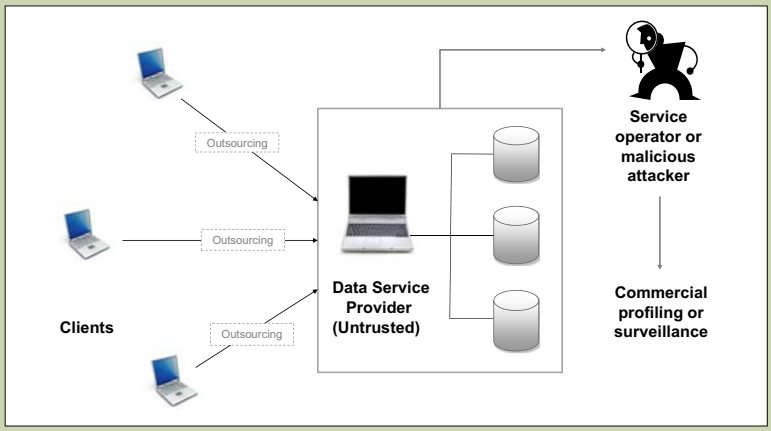
Peter Williams Rimmi Devgan Radu Sion
{petertw,rdevgan,sion}@cs.stonybrook.edu



http://crypto.cs.stonybrook.edu

NSAC
Network Security and Applied Cryptography Lab

Scenario: Outsourcing Data Management to Untrusted Service Providers



Outsourcing minimizes client data management overhead, allowing the client to leverage the service provider's expertise and bulk pricing.

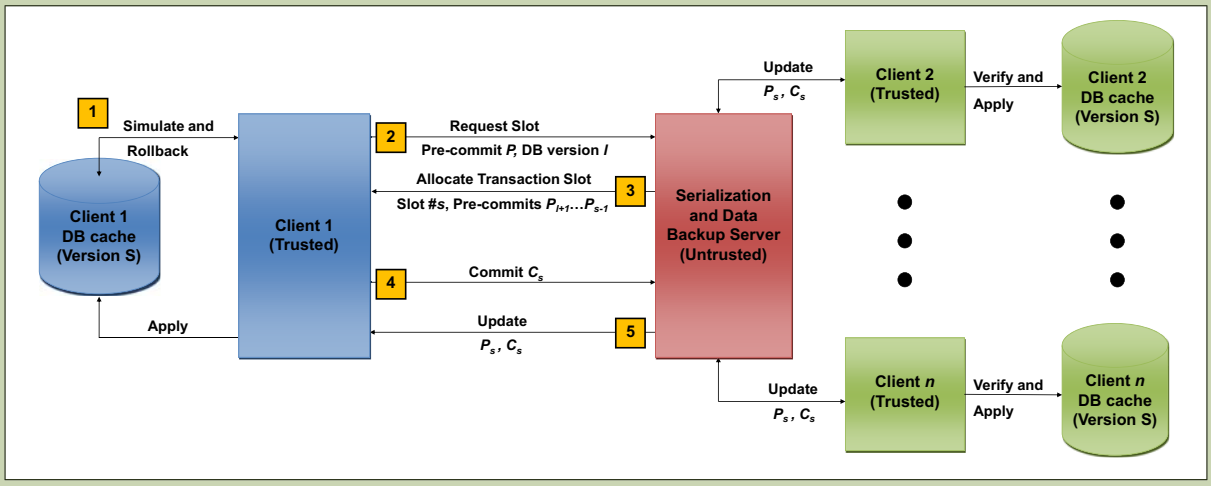
However, clients may not wish to place sensitive data under the control of a remote, third-party provider without practical assurances of *privacy* and *confidentiality*.

With advent of cheap and fast disks and CPUs individual data client's system can host local large data sets. Thus, we predict that in context of a "multi-client transactional database" data management markets will converge on providing the following while guaranteeing data and transaction privacy:

1. persistent client storage
2. availability assurances

Idea: Outsourced Serialization and Durability

Mechanism for collaborative transaction processing with durability guarantees hosted by an untrusted service provider under assurances of confidentiality and access privacy. (NDSS 09, with Dennis Shasha)



The server maintains a definitive representation of the database called an "encrypted transaction log" and allows clients to reserve slots in the log i.e. allows clients to append to this log.

1. The client simulates the intended transaction on its local database copy, then undoes it.
2. Clients issue an encrypted notification of their pending transaction and the slot ID which is the latest the client knows about.
3. The server reserves a slot and returns its ID and a list of all new pre-commit descriptions up to this ID to the client, and relays the notification to the other clients. This pre-commit contains enough information to allow other clients to determine whether it might cause a conflict with their own pending transactions.
4. After this notification ("pre-commit"), clients then check to see if their pending transaction might conflict with any pending transactions scheduled to run before theirs. If not, they commit the slot; otherwise they retry with a new request.
5. The server commits by logging the encrypted transaction to permanent storage. It informs all other clients about the new transaction.

NSF
CNS 0627554
CNS 0708025

16th ACM CCS
November 9-13, 2009
Chicago, IL

STONY BROOK UNIVERSITY