Remote Data Checking: Auditing the Preservation Status of Massive Data Sets on Untrusted Stores

Randal Burns
randal@cs.jhu.edu
www.cs.jhu.edu/~randal/

Department of Computer Science, *Johns Hopkins University*
What’s Remote Data Checking?

Auditing protocols that verify the correctness of data objects on remote, untrusted stores

- Without transferring data to the client
  - Constant network complexity per audit per object
  - Constant amount of metadata per object

- That do not require the store to access the entire file
  - Constant amount of I/O per audit per object
Service-Oriented Architectures

- Outsourced storage commoditized and ubiquitous
  - Cloud computing
  - Amazon S3/EC
  - SDSC Storage Resource Broker

- And, the associated security problems
  - How much trust must we place in services?
  - For data, auditing services for correctness, availability, preservation
Why Remote Data Checking?

- Verifying integrity/content on retrieval is insufficient
  - Too late, data are already damaged.
  - Identifying damaged data quickly is critical to repair

- Data are too large to retrieve and check
  - I/O burden on servers
  - Lots of network traffic
  - Expensive! Services charge by byte of I/O and byte transferred

- Exposure
  - Data are held for long periods of time
  - Much data are accessed infrequently or never
Don’t I Trust Service Providers?

- Financial motivations to cheat
  - Charge for terabytes and store gigabytes
  - Discard unaccessed data (based on statistical analysis)
  - Keep fewer replicas than promised

- Reputation
  - Hide data loss incidents

- Latent errors
  - Of which service providers are unaware

- There’s a history of data-loss incidents
RDC Storage Protocol

- Data owner preprocesses file for RDC protocol
  - May modify file (add bytes, tags, etc.)
  - Generate a constant amount of (public or private) metadata
RDC Audit Protocol

- Verify that an untrusted store retains the correct data
  - Without transferring the data to the verifier (homomorphism)
Many RDC Protocols

- Hot topic of recent industrial and academic research
  - Security [AB+08, BJO08, SW08]
  - Others that don’t quite fit our definition [JK07, KAD07, SM06]
  - Related concepts and extensions [CK+08, CKB08, SS+08]

- Several core principles have emerged
  - Compact signature for multiple blocks: homomorphic tags
  - Probabilistic audits via spot checking
  - Redundancy in storage with forward error correction

- I will explain with Provable Data Possession (PDP)
  - Our system [AB+08, CK+08, CKB08]
PDP’s Spot Checking

Auditor randomly selects a set of blocks to challenge
- A constant number for files of any size
PDP’s Homomorphic Tags

Server computes function of random block set

Small constant amount of data verifies possession of all blocks

- Server processing is I/O bound
  - Single exponentiation per challenge
Forward Error Correcting (FEC) Codes

- Integrating data checking with redundancy
  - Improves possession guarantee realized from spot checking

- Attacker cannot effectively delete data
  - Big attacks are easy to detect
  - Small attacks are recoverable

- Use systematic codes [BJO08,CKB08]
  - To preserve sequential file layout for read performance
PDP with Reed-Solomon Coding

- Systematic RS code keeps original file sequential
  - Practical RS codes fixed/limited widths
- Layout must conceal coding constraints among blocks
  - Random selection of input blocks
  - Encryption and permutation of redundancy blocks

\[
\begin{align*}
\text{randomly select groups of } k \text{ blocks from } b \\
6,4 \text{ RS Encoder} \\
\begin{array}{cccccccc}
b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 & b_8 \\
r_1 & r_2 & r_3 & r_4 \\
\end{array}
\end{align*}
\]

\[
\text{encrypt and permute} \\
\begin{array}{cccc}
b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 & b_8 \\
\end{array}
\begin{array}{cccc}
e_2 & e_4 & e_3 & e_1 \\
\end{array}
\]

= Output File
PDP+FEC: An Attacker’s Perspective

- Successful attack probability < 0.00001
  - 10% redundancy, checking 500kb of a 600MB file
Additional Desirable Properties

- Dynamic (or incremental)
  - Can modify file contents without exposure to replay attacks
- Publicly verifiable
  - No secret material needed to conduct audits
- Efficient (for pre-processing files)
  - Auditing is already quite fast (I/O bound)
- Multiple-replica
- Privacy preserving [SS+08]
  - RDC protocol reveals nothing about the content to the verifier
- No single protocol provides all
  - Notably, publicly verifiable conflicts with dynamic and efficient
PDP: Observations about RSA

- **Provably secure**
- **Allows for public-verifiability**
  - Anyone can check possession (even if they can’t access content)
  - Metadata easy to manage. It’s not secret and can be replicated widely or published.
- **Supports multi-replica protocols**
  - Differentiate copies of the same data in network
  - Dynamic creation of new copies
- **Performance limitations for storage**
  - Must exponentiate every block: to generate the tag
  - Suitable for archival data (store once)
  - Good audit performance
Multiple-Replica PDP [CB+08]

- Multiple copies in untrusted networks to protect data
- For storing/auditing replicas
  - Ensure system stores $t$ unique copies
  - Create new replicas on demand
  - Need efficient techniques to define replicas, i.e. better than PDP $t$ times
- MR-PDP (Multiple-replica)
  - All the above and
  - Verify all replicas with single set of signatures, i.e. $O(1)$ metadata
Other Interesting Ideas

- **Commitment schemes (Safestore [KAD07])**
  - Have a server provide fresh signatures for many files
  - Check a few files among the fresh signatures
  - Spot checking across files can be used in conjunction with RDC

- **Symmetric key homomorphisms [MS06, SW08]**
  - Makes pre-processing fast
  - Supports dynamic RDC
  - Not publicly verifiable and metadata must be kept secret

- **Hierarchical redundancy encoding**
  - Split redundancy across multiple servers and within file [KAD07]
  - Use redundancy in challenge protocol and within file [BJO08]
Conclusions

- Remote data checking supports the outsourced storage model of service-oriented architectures

- PDP and other RDC schemes provide secure and efficient (in both I/O and network) auditing
  - We have yet to get all the desirable features in a single system

- Important systems issues remain
  - File layouts and redundancy
  - Distributed implementations
  - Many usable security schemes
References