

# Security of Cloud Storage: - Deduplication vs. Privacy



Benny Pinkas - Bar Ilan University

Shai Halevi, Danny Harnik, Alexandra  
Shulman-Peleg - IBM Research Haifa

# Remote storage and security

---

- “Easy” to encrypt data, but
  - Encryption is not so easy in practice
  - There are benefits to storing unencrypted data
    - ⇒ no encryption at the client level
    - ⇒ privacy issues

# A recent headline



The screenshot shows the Enterprise Storage Forum website. At the top left is the logo for Enterprise Storage Forum, featuring a stylized orange and white arrow icon. To the right of the logo is a search bar with the text "Search..." and "EnterpriseStorage" entered. Below the search bar is a navigation menu with links for "Data Backup & Recovery", "Storage Hardware", "Data Storage News", "Storage Networking", "Storage Management", and "Storage Basics". Below the navigation menu is a section for "EnterpriseStorageForum Hot Topics:" with a list of topics: "storage networking", "data storage", and "solid state". Below the hot topics is a breadcrumb trail: "enterprisestorageforum.com" > "Data Storage News" > "Read More in Data Storage News »". The main headline is "Deduplication, Security and Clouds Highlight Data Storage News". Below the headline is the date "May 12, 2010" and the author "By Paul Shread". There are links for "Send Email »" and "More Articles »". The article text begins with "The news in the storage networking industry this week wasn't all VPLEX and EMC World. A number of other data storage vendors had some significant news of their own."

ENTERPRISE  
**Storage**  
FORUM.COM™

Search... EnterpriseStorage

Data Backup & Recovery Storage Hardware Data Storage News Storage Networking Storage Management Storage Basics Special Reports Videos Storage Products Storage Definitions

May 14, 2010 EnterpriseStorageForum Hot Topics:  
◀ storage networking data storage solid state ▶

enterprisestorageforum.com Data Storage News Read More in Data Storage News »

## Deduplication, Security and Clouds Highlight Data Storage News

May 12, 2010  
By Paul Shread  
Send Email »  
More Articles »

The news in the storage networking industry this week wasn't all VPLEX and EMC World. A number of other data storage vendors had some significant news of their own.

# Cloud backup services

---

□ Online file backup and synchronization is huge

□ **Mozy**

- Over one million customers and 50,000 business customers. Over 75 PetaByte stored.



□ **Dropbox**

- Over three million customers.



□ And many more... many services geared towards enterprises.

# Mozy

---

- I use MozyHome
  - You get 2GB backup for free
  - You used to pay only \$4.95 per month for unlimited storage! (until very recently)

# Mozy

- You can examine your backup history

Start Time	Type	Duration	Result	F...	Size	Files Enco...	Size Enco...	Files Transfer...	Size Transfer...
14/05/2010 00:18	Manual ...	00:03:33	Success	2...	30.0 GB	1	7.9 MB	0	0 bytes
13/05/2010 23:56	Automa...	00:04:01	Success	2...	30.0 GB	5	990.9 KB	5	990.9 KB
13/05/2010 21:49	Automa...	00:03:02	Success	2...	30.0 GB	4	110.9 KB	4	110.9 KB
13/05/2010 19:30	Automa...	00:03:09	Success	2...	30.0 GB	6	848.4 KB	6	848.4 KB
13/05/2010 17:30	Automa...	00:02:06	Success	2...	30.0 GB	0	0 bytes	0	0 bytes
13/05/2010 15:30	Automa...	00:02:05	Success	2...	30.0 GB	0	0 bytes	0	0 bytes
13/05/2010 13:30	Automa...	00:02:12	Success	2...	30.0 GB	0	0 bytes	0	0 bytes
13/05/2010 11:29	Automa...	00:03:12	Success	2...	30.0 GB	0	0 bytes	0	0 bytes
13/05/2010 09:29	Automa...	00:02:10	Success	2...	30.0 GB	0	0 bytes	0	0 bytes
13/05/2010 07:29	Automa...	00:07:39	Success	2...	30.0 GB	29	26.7 MB	22	14.0 MB
12/05/2010 20:15	Automa...	00:06:36	Success	2...	30.0 GB	4	3.1 MB	4	3.1 MB
12/05/2010 18:15	Automa...	00:07:46	Success	2...	30.0 GB	5	4.5 MB	5	4.5 MB
12/05/2010 16:08	Automa...	00:04:08	Success	2...	30.0 GB	3	135.6 KB	3	135.6 KB
12/05/2010 14:08	Automa...	00:04:10	Success	2...	30.0 GB	2	23.6 KB	2	23.6 KB
12/05/2010 11:54	Automa...	00:09:32	Success	2...	30.0 GB	16	266.7 KB	16	266.7 KB
12/05/2010 09:37	Automa...	00:02:28	Success	2...	30.0 GB	0	0 bytes	0	0 bytes
12/05/2010 07:37	Automa...	00:13:41	Success	2...	30.0 GB	27	43.3 MB	26	19.1 MB
10/05/2010 13:07	Automa...	00:04:00	Success	2...	30.0 GB	18	3.1 MB	15	2.6 MB
10/05/2010 08:07	Automa...	00:02:50	Success	2...	30.1 GB	0	0 bytes	0	0 bytes
10/05/2010 05:58	Automa...	00:02:46	Success	2...	30.8 GB	0	0 bytes	0	0 bytes
10/05/2010 02:11	Automa...	03:45:24	Success	2...	30.8 GB	3	701.4 MB	3	701.4 MB
09/05/2010 23:03	Automa...	03:07:34	Success	2...	30.6 GB	6	453.7 MB	6	453.7 MB
09/05/2010 21:36	Automa...	00:01:50	CancelError0	0	0 bytes	0	0 bytes	0	0 bytes
09/05/2010 18:54	Automa...	00:03:14	Success	2...	30.1 GB	0	0 bytes	0	0 bytes
09/05/2010 16:54	Automa...	00:03:33	Success	2...	30.1 GB	0	0 bytes	0	0 bytes
09/05/2010 14:54	Automa...	00:07:06	Success	2...	30.1 GB	0	0 bytes	0	0 bytes
09/05/2010 12:54	Automa...	00:05:14	Success	2...	30.1 GB	0	0 bytes	0	0 bytes
09/05/2010 10:54	Automa...	00:03:43	Success	2...	30.1 GB	0	0 bytes	0	0 bytes
09/05/2010 08:54	Automa...	00:03:42	Success	2...	30.1 GB	0	0 bytes	0	0 bytes
09/05/2010 06:20	Automa...	00:04:18	Success	2...	30.1 GB	0	0 bytes	0	0 bytes
09/05/2010 03:27	Automa...	00:02:24	Success	2...	30.1 GB	3	4.1 KB	3	4.1 KB
08/05/2010 23:16	Automa...	00:04:03	Success	2...	30.1 GB	14	10.2 MB	12	555.7 KB
08/05/2010 20:05	Automa...	00:02:31	Success	2...	30.1 GB	0	0 bytes	0	0 bytes
08/05/2010 19:53	Manual ...	00:05:32	Success	2...	30.1 GB	1	3.1 MB	1	3.1 MB

File	Path	Patch Size	Encoding T...	Transfer Ti...	Transfer R...	Other Details
Copy of my presentation.p...	C:\Documents and Setting...	7.9 MB	00:00:00			File already on MozyHome servers

File already on MozyHome servers

# Mozy

□ But sometimes strange things happen...

Start Time	Type	Duration	Result	F...	Size	Files Enco...	Size Enco...	Files Transfer...	Size Transfer...
14/05/2010 01:16	Manual ...	00:02:20	Success	2...	30.4 GB	1	175.0 MB	0	0 bytes
14/05/2010 01:13	Manual ...	00:02:23	Success	2...	30.2 GB	1	233.7 MB	0	0 bytes
14/05/2010 00:54	Manual ...	00:02:51	Success	2...	30.0 GB	1	2.4 MB	1	2.4 MB
14/05/2010 00:47	Manual ...	00:05:07	Success	2...	30.0 GB	1	106.6 KB	1	106.6 KB
14/05/2010 00:44	Manual ...	00:02:13	Success	2...	30.0 GB	2	46.9 KB	2	46.9 KB
14/05/2010 00:40	Manual ...	00:02:09	Success	2...	30.0 GB	2	164.0 KB	2	164.0 KB
14/05/2010 00:36	Manual ...	00:02:37	Success	2...	30.0 GB	3	239.1 KB	3	239.1 KB
14/05/2010 00:18	Manual ...	00:03:33	Success	2...	30.0 GB	1	7.9 MB	0	0 bytes
13/05/2010 23:56	Automa...	00:04:01	Success	2...	30.0 GB	5	990.9 KB	5	990.9 KB
13/05/2010 21:49	Automa...	00:03:02	Success	2...	30.0 GB	4	110.9 KB	4	110.9 KB
13/05/2010 19:30	Automa...	00:03:09	Success	2...	30.0 GB	6	848.4 KB	6	848.4 KB
13/05/2010 17:30	Automa...	00:02:06	Success	2...	30.0 GB	0	0 bytes	0	0 bytes
13/05/2010 15:30	Automa...	00:02:05	Success	2...	30.0 GB	0	0 bytes	0	0 bytes
13/05/2010 13:30	Automa...	00:02:12	Success	2...	30.0 GB	0	0 bytes	0	0 bytes
13/05/2010 11:29	Automa...	00:03:12	Success	2...	30.0 GB	0	0 bytes	0	0 bytes
13/05/2010 09:29	Automa...	00:02:10	Success	2...	30.0 GB	0	0 bytes	0	0 bytes
13/05/2010 07:29	Automa...	00:07:39	Success	2...	30.0 GB	29	26.7 MB	22	14.0 MB
12/05/2010 20:15	Automa...	00:06:36	Success	2...	30.0 GB	4	3.1 MB	4	3.1 MB
12/05/2010 18:15	Automa...	00:07:46	Success	2...	30.0 GB	5	4.5 MB	5	4.5 MB
12/05/2010 16:08	Automa...	00:04:08	Success	2...	30.0 GB	3	135.6 KB	3	135.6 KB
12/05/2010 14:08	Automa...	00:04:10	Success	2...	30.0 GB	2	23.6 KB	2	23.6 KB
12/05/2010 11:54	Automa...	00:09:32	Success	2...	30.0 GB	16	266.7 KB	16	266.7 KB
12/05/2010 09:37	Automa...	00:02:28	Success	2...	30.0 GB	0	0 bytes	0	0 bytes
12/05/2010 07:37	Automa...	00:13:41	Success	2...	30.0 GB	27	43.3 MB	26	19.1 MB
10/05/2010 13:07	Automa...	00:04:00	Success	2...	30.0 GB	18	3.1 MB	15	2.6 MB
10/05/2010 08:07	Automa...	00:02:50	Success	2...	30.1 GB	0	0 bytes	0	0 bytes
10/05/2010 05:58	Automa...	00:02:46	Success	2...	30.8 GB	0	0 bytes	0	0 bytes
10/05/2010 02:11	Automa...	03:45:24	Success	2...	30.8 GB	3	701.4 MB	3	701.4 MB
09/05/2010 23:03	Automa...	03:07:34	Success	2...	30.6 GB	6	453.7 MB	6	453.7 MB
09/05/2010 21:36	Automa...	00:01:50	CancelError	0	0 bytes	0	0 bytes	0	0 bytes
09/05/2010 18:54	Automa...	00:03:14	Success	2...	30.1 GB	0	0 bytes	0	0 bytes
09/05/2010 16:54	Automa...	00:03:33	Success	2...	30.1 GB	0	0 bytes	0	0 bytes
09/05/2010 14:54	Automa...	00:07:06	Success	2...	30.1 GB	0	0 bytes	0	0 bytes
09/05/2010 12:54	Automa...	00:05:14	Success	2...	30.1 GB	0	0 bytes	0	0 bytes

File	Path	Patch Size	Encoding T...	Transfer Ti...	Transfer R...	Other Details
30.Rock.S03E20.HDTV.Xvid-LOL.avi	C:\Documents and Setting...	175.0 MB	00:00:22			File already on MozyHome servers

30.Rock.S03E20.HDTV.Xvid-LOL.avi 175MB

# Deduplication

---

- Deduplication = storing and uploading only a single copy of redundant data
  - Applied at the file or block level
  - Saves more than 90% in common business scenarios (90% of 75 PetaBytes...)
  - “most impactful storage technology”
    - July 2009: EMC acquires DataDomain for \$2.1B
    - April 2008: IBM acquires Dilligent for \$200M



# Deduplication and privacy

---

- Our attacks require the following features:
  - Cross-user deduplication
    - If two or more users store the same file, only a single copy is stored.
  - Source-based deduplication
    - Deduplication is performed at the client side.
    - Saves bandwidth as well as storage.
  
- It is easy to check whether your storage service uses these features.

# Deduplication and privacy

---

- The storage service is an oracle, which answers the following query
  - “Has any other user previously uploaded this file?”
- Rather limited
  - Does not tell who uploaded the file
  - The attacker can only ask this query once – afterwards the file is always deduped (*but this issue can be solved!*)
- Still, many attacks are possible

# Repeating the attack

---

- Naively, the attacker can only check once whether a file has been previously uploaded
  - Attempt to backup the file.
  - If no upload occurs, then someone must have previously uploaded the file.
  - If an upload occurs, then no one has uploaded the file before. But now the file is uploaded and it won't be possible to repeat the test ☹
- **Solution:** When the actual upload begins, terminate the communication.

# File Identification Attacks



# Attack I – Identifying files

---

- ❑ Alice gives Bob a file, and swears him not to copy it to his home machine (which uses MozyHome/Dropbox/etc.)
- ❑ Alice can check if Bob followed this request
- ❑ Relevant to the Wikileaks case.
- ❑ There is no need for Alice to upload entire sensitive files.
- ❑ Easy to implement.

# Attack II – Learning contents of files

---

- ❑ Alice and Bob work for the same company, which uses online storage for backup.
- ❑ All employees receive a standard form listing their yearly bonus.
- ❑ Alice knows that Bob's bonus is a multiple of \$500, and is in the range \$0 - \$100K.
- ❑ She generates 201 documents, and runs a backup...
  
- ❑ Essentially a brute force attack. Applicable when the range is of medium size ( $10^6$ ?)

# Attack II – Learning contents of files

---

- Essentially a side channel
  
- Possible scenarios
  - Online banking – learning PIN codes or details of transactions
  - Learning results of medical tests
  - Learning bids in auctions
  - Many others examples...

# Attack III – covert channel

---

- Alice installs a malicious software on Bob's machine.
- Bob runs a firewall, blocking network access.
- Bob uses an online storage service.
  
- To transfer a bit to Alice, the software saves one of two versions of a file.
  - Much more efficient coding is possible.



# Consequences

---

- These are simple attacks
- But no company would be happy if they could be applied to its data

# Solution I

---

- ❑ Global policy: Do not perform deduplication.
- ❑ Local hack: Bob encrypts his files with a personal key.
  - Then it is impossible for the service to check whether Alice's file is identical to Bob's.

# Solution I

---

- ❑ Global policy: Do not perform deduplication.
- ❑ Local hack: Bob encrypts his files with a personal key.
  - Then it is impossible for the service to check whether Alice's file is identical to Bob's.
- ❑ The cost is too high (both for dedup and for support for lost keys).
  - Services which support encryption with personal keys do not have an all-you-can-eat pricing option.

# Side note - encryption

---

- ❑ All online storage services encrypt data
  - But in order to support dedup they do not encrypt stored data with personal keys.
- ❑ Dropbox: “Dropbox uses [modern encryption](#)... All transmission of data occurs over an encrypted channel ([SSL](#)). All files stored on Dropbox servers are encrypted ([AES-256](#)) and are inaccessible without your account password.”
- ❑ Mozy enable users to use personal keys
  - But this is not the default option, and users are strongly advised against using it.
  - Personal key is susceptible to [offline](#) brute force attack.

# Side note - costs

---

- Suppose that
  - Deduplication currently provides a saving of 95%
  - A solution reduces the savings to 93%

# Side note - costs

---

- Suppose that
  - Deduplication currently provides a saving of 95%
  - A solution reduces the savings to 93%
  
- Costs are increased by 40% !

# Solution II

---

- Perform deduplication at the server.
- Files are always uploaded
  - Users do not notice whether dedup occurs 😊
  - But, high communication costs 😞 (at Amazon S3, cost of uploading is  $\approx$ cost of two months of storage).

# Solution II

---

- ❑ Perform deduplication at the server.
- ❑ Files are always uploaded
  - Users do not notice whether dedup occurs 😊
  - But, high communication costs 😞 (at Amazon S3, cost of uploading is  $\approx$ cost of two months of storage).
- ❑ Probably used by all mail services.



# Solution II

---

- Perform deduplication at the server.
- Files are always uploaded
  - Users do not notice whether dedup occurs 😊
  - But, high communication costs ☹️ (at Amazon S3, cost of uploading is  $\approx$ cost of two months of storage).
- Probably used by all mail services.
- Variant: upload all small files, perform client-side dedup only on large files.
  - After we notified Mozy about our findings, they started using this solution!

# More Solutions

---

- Solution III – randomized approach
  - Server sets a random threshold  $T_x$  per file. Only if  $T_x$  copies of file are uploaded, dedup occurs.
  - Details omitted.
  
- Solution IV
  - Give users an interface which enables them to define which files are to be deduped.

# Hash values and Proofs of Ownership



# Deduplication and hash values

---

- A different (and more direct) attack

# Deduplication and hash values

---

- A different (and more direct) attack
- During upload
  - Client computes and sends server hash of file
  - If this is the first time server receives this hash value, it tells the client to upload the file
  - Otherwise (dedup), it skips the upload and registers the client as another owner of the file
  - Client is then allowed to download the file...

# Implications

---

- A short hash value serves as a proof of file ownership
  - This hash value is not really meant to be kept secret
  - The hash value is computable from the file using an algorithm shared by all clients

# Attack I – Abusing known hash values

---

- Suppose that the dedup procedure uses a common hash function (e.g., SHA256)
  - Bob is a researcher who writes daily lab reports, and publishes their hash as a time-stamp.
  - He also uses an online backup service.

# Attack I – Abusing known hash values

---

- Suppose that the dedup procedure uses a common hash function (e.g., SHA256)
  - Bob is a researcher who writes daily lab reports, and publishes their hash as a time-stamp.
  - He also uses an online backup service.
  
  - Alice signs to the same backup service.
  - She attempts to upload a file. When asked for its hash value, she sends a hash published by Bob.
  - The service forgoes the upload.
  - Alice can now download Bob's lab report.



# Attack II – Efficient file leakage

---

## □ Malicious software

- A malicious software running on Bob's machine wishes to *stealthily* leak **all** his files to Alice.
- Instead of sending huge files to Alice, can send her the short hash values of the files.
- Alice can then attempt to upload files, present the hash values she received, and obtain access to Bob's files.

# Attack II – Efficient file leakage

---

## ❑ Malicious software

- A malicious software running on Bob's machine wishes to *stealthily* leak **all** his files to Alice.
- Instead of sending huge files to Alice, can send her the short hash values of the files.
- Alice can then attempt to upload files, present the hash values she received, and obtain access to Bob's files.
- The malicious software can even store all Bob's hash values in a single file, and send its hash value to Alice.
  - ❑ A 20-32 bytes message can leak all of Bob's files!

# Attack III – Content distribution network (CDN)

---

## □ Content distribution

- Alice wishes to send a large file to all her friends, but she has a limited uplink.
- Instead of sending the file to each of her friends, she can upload the file once and send its hash value to her friends.
- Each friend can now present the hash value to the server and obtain access to the file.

# Attack III – Content distribution network (CDN)

---

## □ Content distribution

- Alice wishes to send a large file to all her friends, but she has a limited uplink.
- Instead of sending the file to each of her friends, she can upload the file once and send its hash value to her friends.
- Each friend can now present the hash value to the server and obtain access to the file.
- Server essentially serves as a Content Distribution Network (CDN). This might break its cost structure, if it planned on serving only few restore ops.

# “Solutions” to the hash attacks

---

- The source of the problem is that a single hash value represents the file.

# “Solutions” to the hash attacks

---

- The source of the problem is that a single hash value represents the file.
- “Solution”: Use a non-standard hash algorithm (e.g.  $\text{SHA}(\text{“service name”} \mid \text{file})$  )
  - All users must still know the hash algorithm.  
Therefore Attacks II and III are not prevented ☹

# “Solutions” to the hash attacks

---

- The source of the problem is that a single hash value represents the file.
- “Solution”: For every client, server picks a random nonce , and asks client to compute  $\text{SHA}(\text{nonce} \mid \text{file})$ 
  - Server, too, must retrieve file from (multi-petabyte) secondary storage, and compute hash ☹

# Constraints that must be satisfied by a solution

---

- ❑ Must be bandwidth efficient
- ❑ Server cannot retrieve files from secondary storage
  - Must store only a few bytes per file
- ❑ Client might need to process huge files
  - File cannot be stored in main memory
- ❑ Attacker might have partial knowledge of file (e.g., 95% of file)
- ❑ Accomplices might send information to the attacker



# Proofs of Ownership (PoWs)

---

- ❑ Server preprocesses file.
- ❑ Server then stores some short information per file.
- ❑ Client proves ownership of the file
  - Client has access to the file (but not to any preprocessed version of it, as prover has in PoR).
  - Server has only access to short information.
  - Unlike PoRs, do not require extraction.
- ❑ **Security definition:** if min-entropy of file  $>$  security parameter, then proof fails whp.

# Solution – first attempt

---



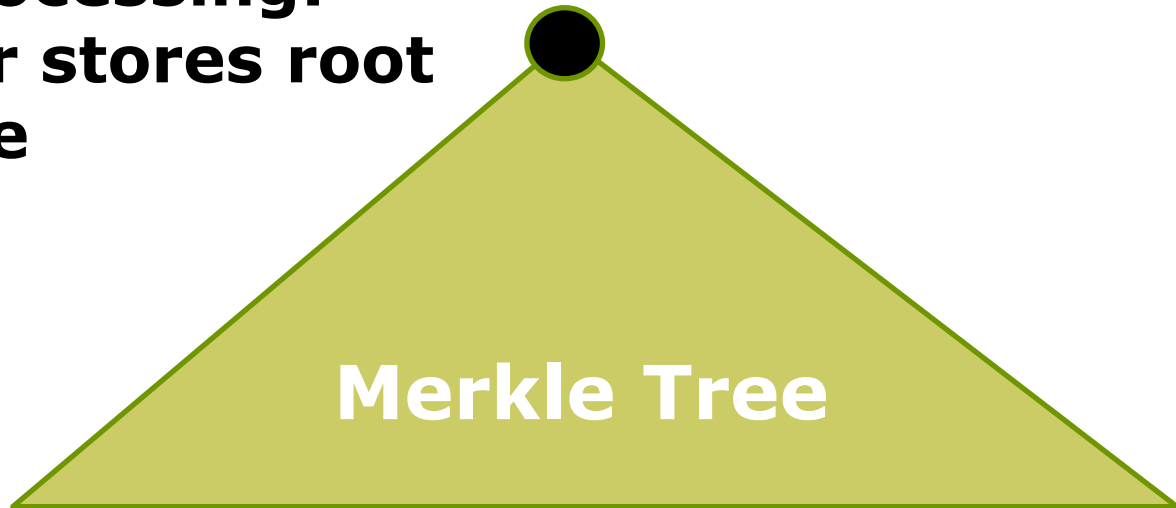
**File**



# Solution – first attempt

---

**Preprocessing:  
server stores root  
of tree**



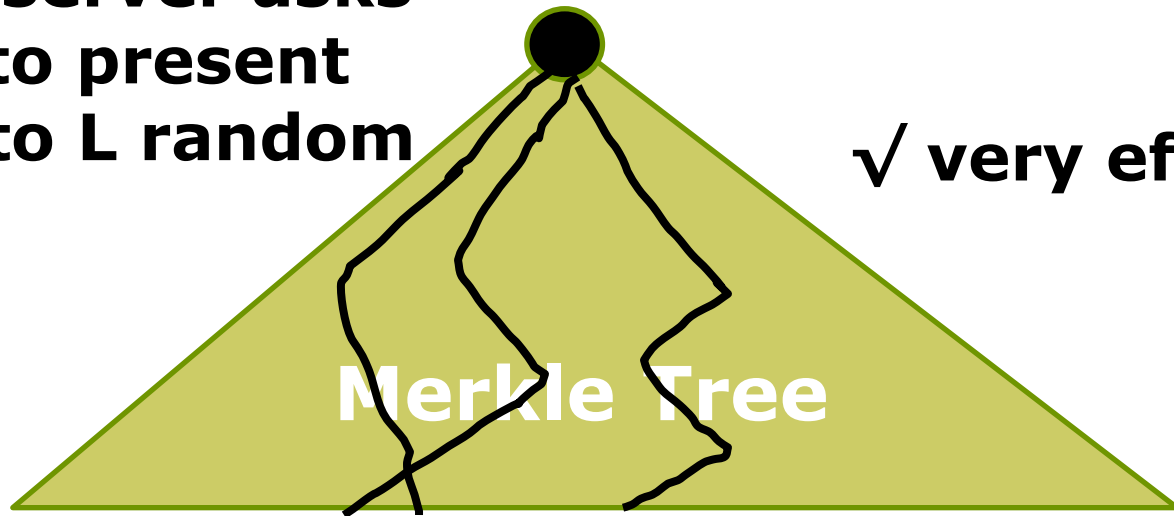
**File**



# Solution – first attempt

---

**Proof: server asks client to present paths to  $L$  random leaves**



✓ very efficient

**File**



**A client which knows only a  $p$  fraction of the file, succeeds with prob  $< p^L$ .**

# Problem and solution

---

- A client which knows a large fraction of the blocks (say, 95%), can pass the test with reasonable probability ( $0.95^{10}=0.6$ ).
- Solution:

Apply solution  
to new tree

**Merkle Tree**

**File  
Erasure  
code**



# Ensuring low answer probability of cheating client

---

- Apply an erasure code to the file, and then construct a Merkle tree over the encoding
  - Erasure code property: knowledge of, say, 50% of the encoding suffices to recover original file.
  - Therefore an attacker who does not know all the file, does not know  $> 50\%$  of the encoding.
  - Fails in each Merkle tree query w.p. 50%.
  - Cheating probability is now  $2^{-L}$

# Efficient encoding

---

- ❑ Computing an erasure code by the client requires
  - Random access, i.e. storing the file in main memory
  - Or, running many passes over the file
- ❑ But the file might be much larger than client's memory... ☹️

# Protocols with small space

---

## □ Relax the requirements

- Only  $L$  bytes are needed for the computation (say,  $L=64\text{MB}$ )
- (Therefore leaking  $L$  bytes to the attacker by an accomplice, enables it to cheat.)



# Protocols with small space

---

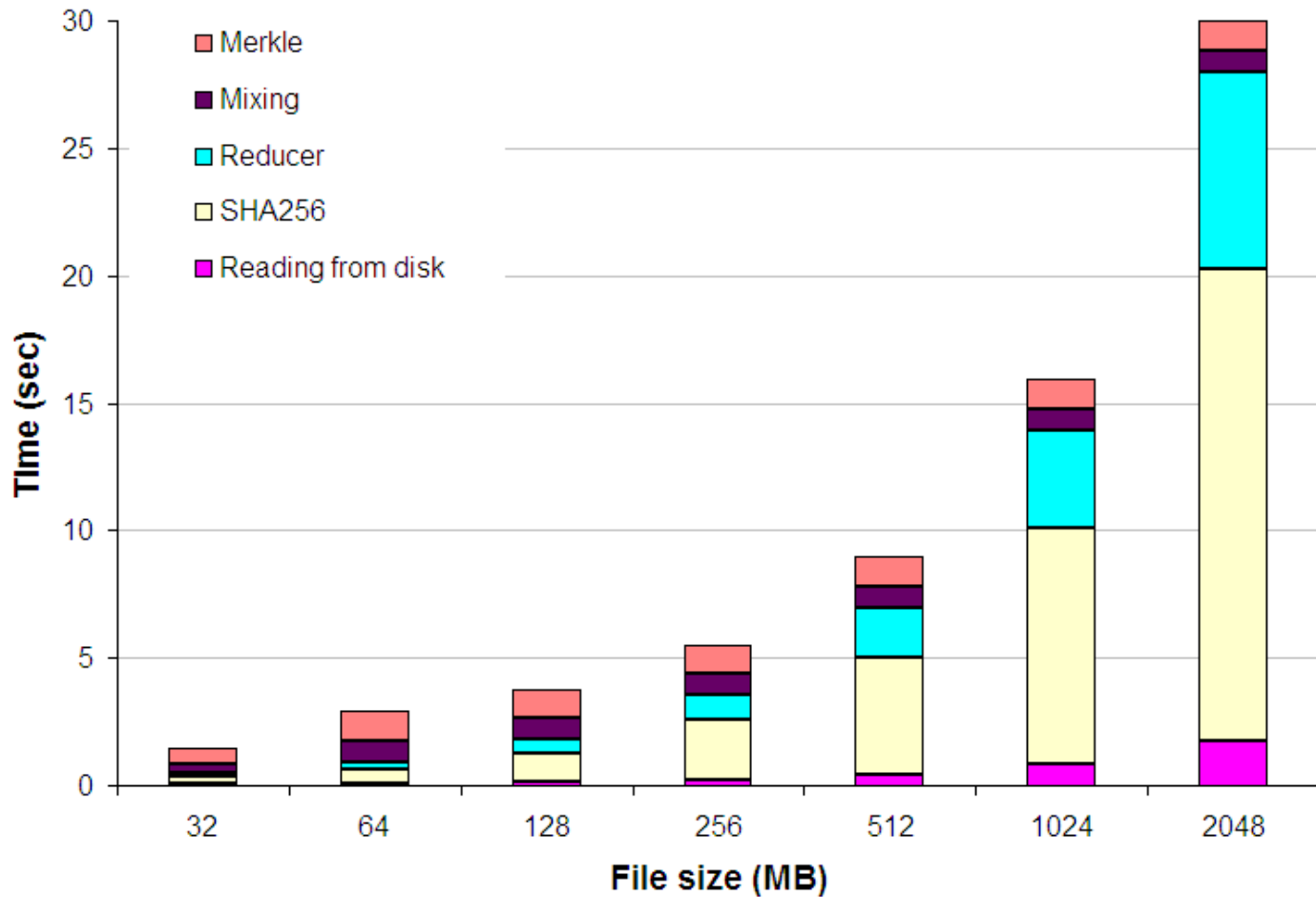
- First hash file to a buffer of L bytes. Then construct Merkle-tree over the buffer.

Challenge: Must be secure even if attacker knows hash function (e.g., can ask for 50% of the L byte buffer).

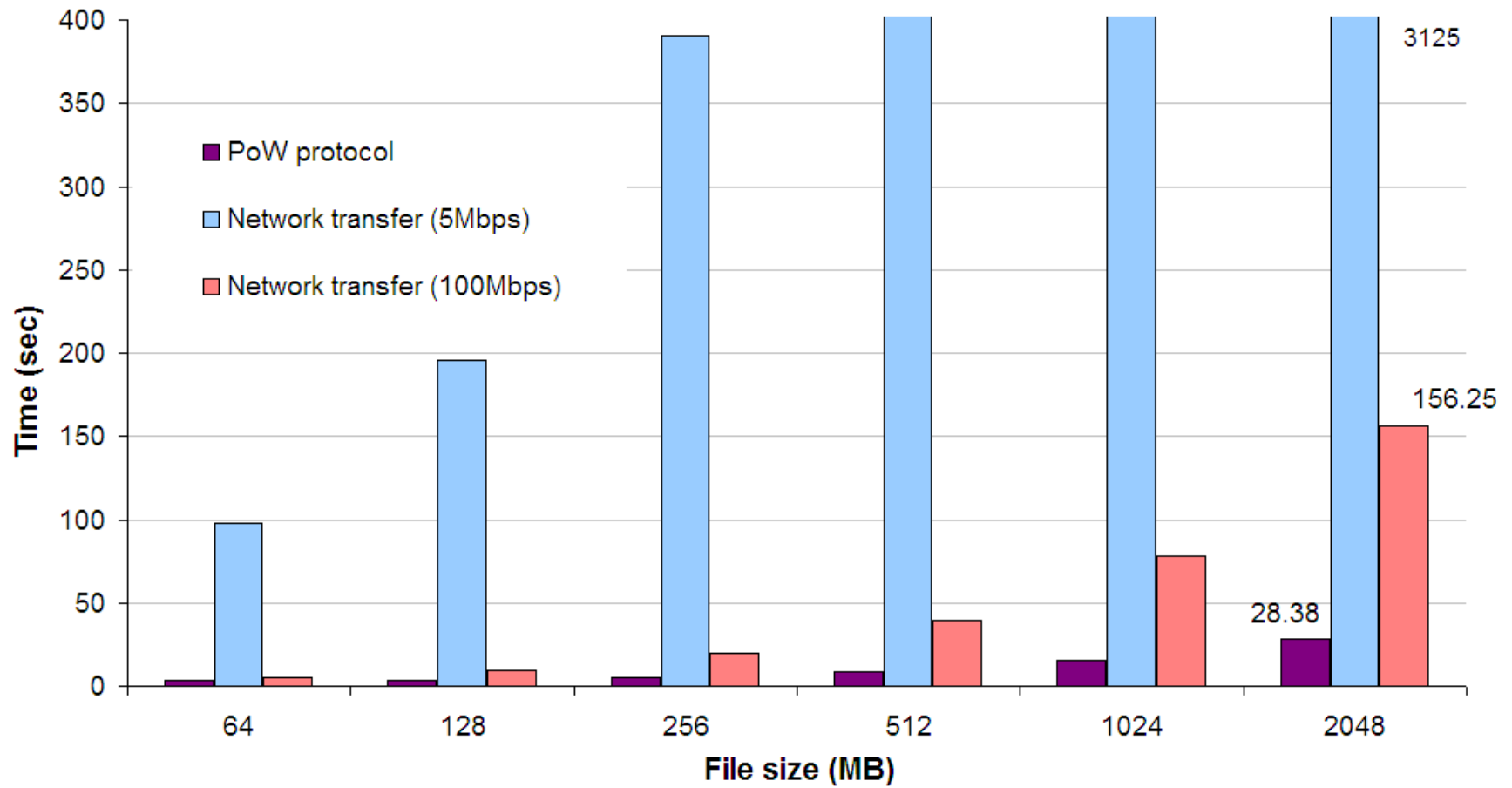
Apply solution to tree  
computed over reduced data



# Performance of the different phases of the low space PoW



# Running the PoW protocol compared to sending the file



# Conclusions

---

- ❑ Deduplication offers huge savings and yet might leak information about other users
- ❑ Most vendors are not aware of this
- ❑ The challenge: offer meaningful privacy guarantees with a limited effect on cost