

# CS290i - Lecture 18 Secure Distrib. FileSystem

Scalable Internet Services and Systems, Spring 2001

Thorsten von Eicken  
Department of Computer Science  
University of California at Santa Barbara

## SFS

### † Fast and Secure Distributed Read-Only File System

- † K. Fu., F. Kaashoek, D. Mazieres, MIT
- † OSDI 2000, <http://www.fs.net/>

### † Motivation

- † Publish read-only (write seldom) data
- † Use untrusted servers (or CDNs)
- † Allow clients to cryptographically verify data origin and integrity
- † Push cryptographic cost onto clients

### † Alternatives

- † SSL servers: cannot use untrusted machines, expensive crypto on server
- † Signed software (PGP/GPG): not transparent, no expiration/revocation

## SFS model

### † Publication

- † Owner prepares FS image on local drive
- † Runs sfsrodb to package and sign FS image
- † Copies image to untrusted servers
- † Client uses sfsrocd to access servers
- † Client daemon performs crypto signature & integrity checks

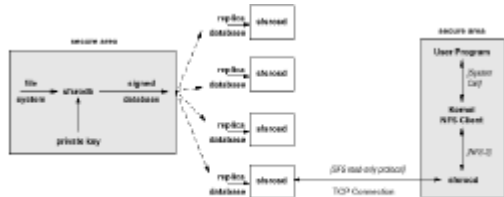


Figure 1: The SFS read-only file system. Shaded boxes show the trusted computing base.

## Features

### † Attacker may compromise any server machine

- † SFS does not prevent denial of service attacks

### † SFS verifies integrity of data

- † does not provide confidentiality

### † SFS verifies the age of the data

- † No older than the expiration period
- † No older than any previously retrieved data

### † Crypto

- † Rabin public keys
- † SHA-1 crypto hash

Operation	Cost (cycles)
Sign 68 byte block	34,400
Verify 68 byte block	68
SHA-1 256 byte (n+1)block	17
SHA-1 2,096 byte (n+1)block	106

Figure 2: Performance of basic primitives on a 650 MHz Pentium III. Signing and verification use 1,024-bit Rabin-Williams keys.

## Protocol

### † Getfsinfo RPC

- † Returns signed FSINFO struct
- † Public key is embedded in server's name
  - ‡ /sfs/new-york.lcs.mit.edu:85xq6pznt4mgfvj4mb23x6b8adak55ue
- † Start & duration: time FS got created, duration of validity
- † Iv: initialization vector for SHA-1 hashes
- † Rootfh: hash of root directory inode
- † Fhdb: tree of all handles in FS, used to detect deleted blocks

```
struct FSINFO {
    sfs_time start;
    unsigned duration;
    opaque iv[16];
    sfs_hash rootfh;
    sfs_hash fhdb;
};
```

### † Getdata RPC

- † Argument: 20-byte SHA-1 hash
- † Returns: data block producing that hash

## File System Structure

### † I-node

- † Metadata: file type, size, modify time
- † Handles for 8 KB blocks
  - ‡ Indirect blocks for larger files

### † Directory

- † <name, handle> pairs

### † FS generator

- † Bottom-up construction
- † No “..” entry (circular)
  - ‡ Each dir contains its own full pathname

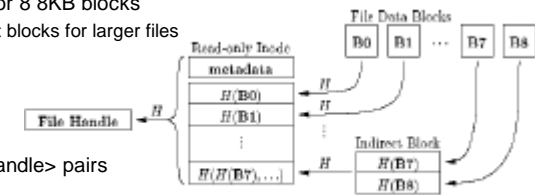


Figure 4: Format of a read-only file system inode.

## FS Updates

### † Updating a FS

- † New FS image is pushed to servers
- † Unchanged files keep the same handles
- † Modified or removed files result in “handle not found” errors
- † Must prevent “handle not found” attacks
  - ‡ E.g when client is looking for revoked certificates

### † Fhdb

- † Tree of all handles in FS
- † Nice property: can keep deleted files in FS for a few versions
- † Problem: tree changes completely even if only a small portion of the FS changes
- † Alternative: clients keep pathname of all open files and reopen files if FSINFO struct changes

## Applications

### † Certificate authority

- † /verisign/www.cs.ucsb.edu symlink to self-certifying SFS file server for CS UCSB

### † Software distribution

- † Compared to signed RPMs, provides revocation, and “complete-ness” check

# Performance

## † Set-up

- † 550Mhz P-II, FreeBSD 3.3
- † Fast Ethernet
- † 9GB SCSI disks

## † Observations

- † Good throughput
- † Difference to NFS due to user-level implementation
- † Crypto is only 0.086s

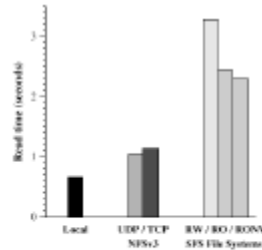


Figure 6: Time to sequentially read 1,000 1-Kbyte files. Local is FreeBSD's local FFS file system on the server. The local file system was tested with a cold cache. The network tests were applied to spanned server racks, hot cold client caches. RW, RO, and ROSV denote respectively the read-write protocol, the read-only protocol, and the read-only protocol with no verification.

Breakdown	Cost (sec)
NFS Inoplock	0.001
Computation in client	1.396
Communication with server	0.777
Total	2.25

Table 1: Breakdown of SFS read-only performance reported in Fig 6

# Performance (cont.)

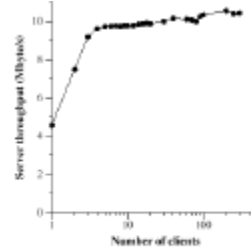


Figure 9: The aggregate throughput delivered by the read-only server for an increasing number of clients synchronously compiling the Emacs 20.6 source. The number of clients is plotted on a log scale.

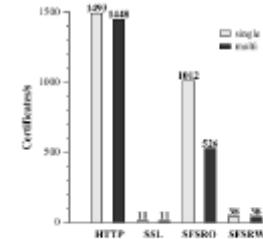


Figure 10: Maximum sustained certificate download per second. HTTP is an Internet Web server, SSL is a secure Web server, SFSRW is the secure SFS read-write file system, and SFSRO is the secure read-only file system. Light bars represent single-component lookups while dark bars represent multi-component lookups.

# Project changes

## † Benchmark changes

- † Each of the benchmarks will have requests for quotes in addition to the stock trading requests. Probably around 4x to 10x as many quote requests as trading requests.

## † Added new robot/quotes?ticker request

- † provide quote for requested company, or for all companies sorted alphabetically by ticker if no ticker is specified

## † Added parent order id robot/transactions response

- † Field was missing

## † Persistency requirement

- † Your system **must** commit each trade entered into the web site to persistent store, i.e. the database on bugatti, **before** returning the HTTP response. That is, if all three web servers failed simultaneously, all orders for which a response was received before the crash must be stored in one form or another in the database.