Keypad

an auditing file system

Agenda

* Motivation and goals
* Internal mechanisms
* Evaluation results
* Conclusions

Motivation and gloals

Some numbers:

* One in ten laptops is lost or stolen within a year of purchase
* 600,000 laptops are lost annually in U.S. airports alone
* Dry cleaners in the U.K. found over 4,000 USB sticks in pockets in 2009
* The most common Web password is “123456”

Motivation and gloals

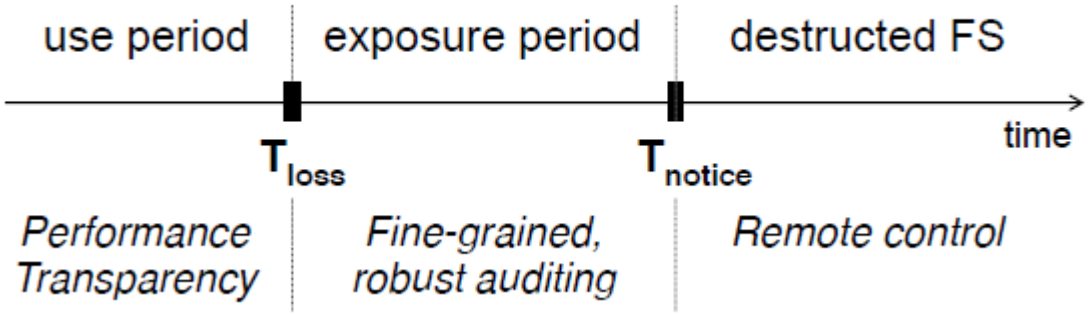
Keypad in auditing file system for theft-prone devices, such as laptops and USB sticks.

Keypad provides two important properties.

* First, Keypad supports fine-grained file auditing: a user can obtain explicit evidence that no files have been accessed after a device's loss.
* Second, a user can disable future file access after a device's loss, even in the absence of device network connectivity.

Keypad achieves these properties by weaving together encryption and remote key storage. By encrypting files locally but storing encryption keys remotely, Keypad requires the involvement of an audit server with every protected file access. By alerting the audit server to refuse to return a particular file's key, the user can prevent new accesses after theft.

Motivation and gloals



Timeline of theft/loss.

This timeline shows the two critical events during the lifetime of a device: the device loss and the user noticing that the device has been lost.

Motivation and gloals

Goals

Robust auditing semantics:

Keypad must provide robust semantics by preventing unrecorded file accesses. To achieve this, the remote auditing server must observe data and metadata operations performed on the client.

Motivation and gloals

Performance: File access latency and throughput should be acceptable for Keypad-protected data. We mainly target office productivity and mobile. We also assume multiple network environments: at the office (LANs), at home (broadband), and on the road (3G or 4G). We seek minimal overhead at work or home, but will tolerate some increased latency in challenging mobile environments in exchange for Keypad's properties.

Motivation and gloals

Fine granularity:

Keypad should produce detailed access logs of read and write accesses to individual Keypad-protected files. Administrators can control the granularity and coverage of these logs; e.g., configuring Keypad to produce audit logs for an entire file system or only for specific files identified as sensitive.

Motivation and gloals

User transparency:

We assume that users are not technically sophisticated; therefore, Keypad's operation should be largely transparent to them and its auditing security should be independent of users' technical competence.

Motivation and gloals

Remote access control:

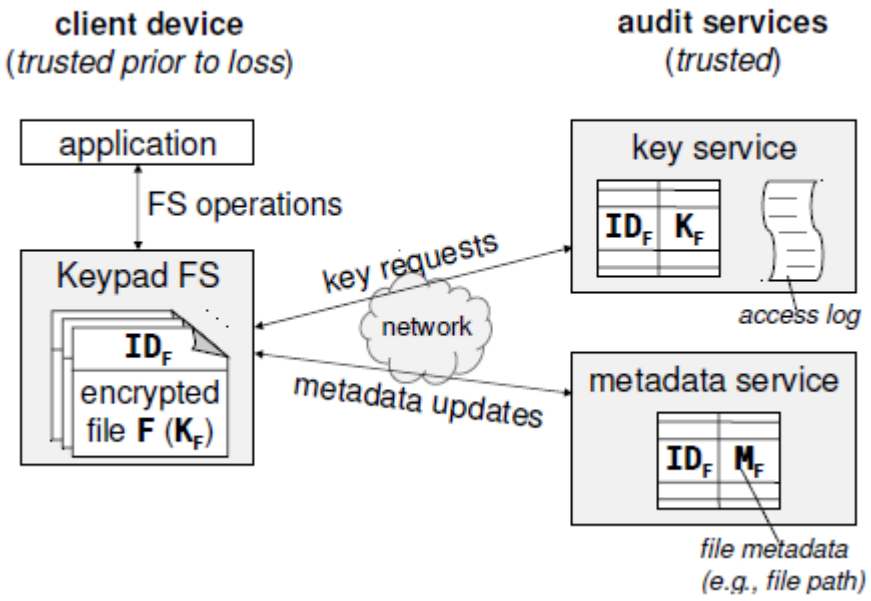
The victim should be able to disable access to protected files after device loss, even if the device has no network or computational capabilities. If an adversary has not yet accessed a protected file, then disabling access prevents any access to the file in the future. If an adversary has already accessed the file, we provide no guarantees about repeat accesses.

Motivation and gloals

Non-goals:

* No ensurance of the device's physical or software integrity after theft/loss.
* Theft/loss that is detectable by a user.
* Keypad ensures auditability and remote control solely at the file system interface level and below.
* Kaypad do not seek to improve the confidentiality of protected files over traditional encryption.
* Keypad do not guarantee that users can always access Keypad-protected files in the absence of network connectivity.

Internal mechanisms



Keypad Architecture.

Each file is encrypted with its own random symmetric key. Keys are stored remotely on a key service. To enable forensics, a (separate) metadata service stores file metadata.

Internal mechanisms

Keypad provides users with strong auditing semantics at audit time (i.e., post T_{loss}). We formulate an ideal invariant describing these semantics as follows:

For any file **F** with identifier $\mathbf{ID_F}$ that was accessed after $\mathbf{T_{loss}}$ the following properties hold:

(1) the key service shows an $\mathbf{ID_F}$ log entry after $\mathbf{T_{loss}}$, and

(2) the metadata service shows all metadata updates that occurred on $\mathbf{ID_F}$ before $\mathbf{T_{loss}}$.

Internal mechanisms

In theory, Keypad could achieve semantics arbitrarily close to this ideal invariant. If Keypad downloaded a file's key every time a block in the file is accessed and erased the key from memory immediately after using it, then it would obtain the first part of the invariant. Similarly, if Keypad waited for every metadata update to be acknowledged by the metadata service before completing that operation on the local disk, then it would obtain the second part.

Internal mechanisms
Encryption Key Caching and Prefetching

Many of Keypad's critical-path operations are remote key-fetching requests, e.g., issued whenever an application performs a file read or write. The number of such key requests can be minimized using standard OS mechanisms, such as caching and prefetching. For instance, instead of erasing a key immediately after use, Keypad can cache it locally. Similarly, on access to a file F, Keypad can prefetch keys for other related files, such as those in the same directory. Key caching and prefetching remove key retrieval from the critical path of many file accesses, dramatically improving performance.

Internal mechanisms
Encryption Key Caching and Prefetching

Consequences:

* Keys that are cached at time T_{loss} are susceptible to compromise: if an adversary can extract them from memory he can permanently remember those keys and bypass audit records for those files.
* For caching, we impose short lifetimes (T_{exp}) on keys and securely erase them at expiration. This bounds key accumulation in memory; the shorter the T_{exp}, the fewer keys will be exposed after T_{loss}. Experimentally, we find that key expirations as short as 100 seconds reap most of the performance benefit of caching, while exposing relatively few keys in memory at a given time.
* Prefetching affects what users can deduce from the audit log of a lost device.
* For prefetching, we designed a simple scheme to prefetch keys only when a file-scanning workload is detected (e.g., recursive file search or file hierarchy copying). This benefits file-system-heavy workloads where prefetching is the most useful, while maintaining high auditing precision for light workloads (e.g., interacting with a document).

Internal mechanisms
Identity-Based Encryption for Metadata Updates

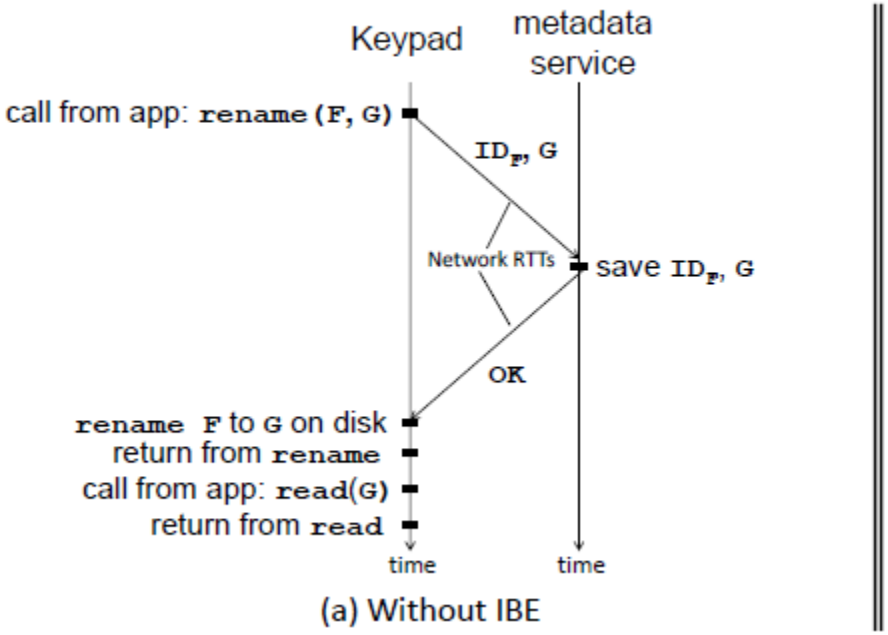
To respond to this challenge, Keypad leverages identity-based encryption (IBE) in a way that both eliminates the network from the critical path of metadata updates and retains its strong auditing semantics. IBE allows a client to perform public-key encryption using any key string it chooses as the public key. A server called a private key generator (PKG) is required to generate the decryption key for the arbitrary public key. Most importantly for our use, the PKG need not know the public key string in advance, but the public key string must be provided to the PKG to learn the decryption key.

We modified Keypad to use IBE as follows. First, we add a level of indirection for file encryption keys. A file F's content is encrypted using a locally-generated random data key (denoted $K_F{}^D$) stored in the file's header. The data key is itself encrypted under the remote key, which in turn is stored on the key server.

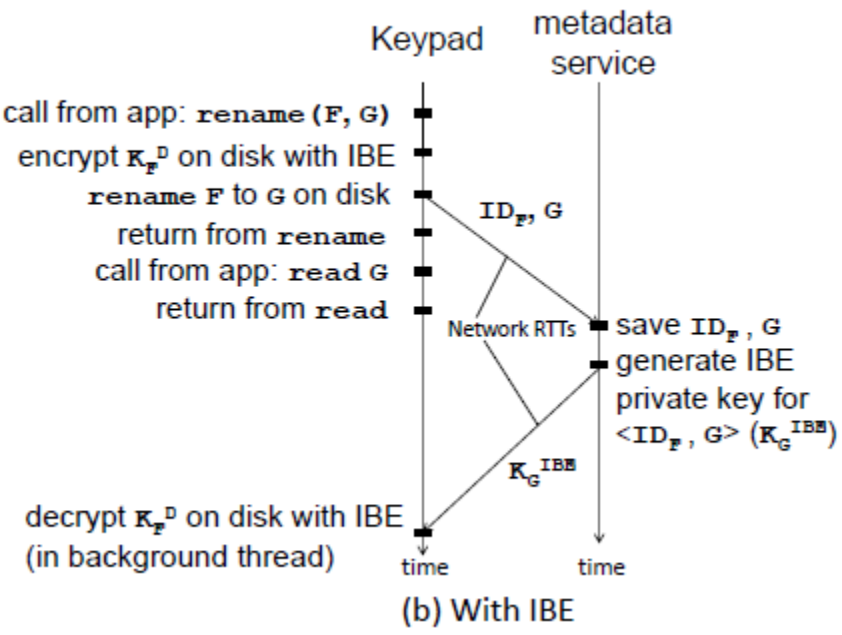
Second, Keypad's metadata service acts as a PKG. When an application invokes a metadata operation (such as rename) for a file F, Keypad "locks" its encrypted data key $K_F{}^D$ in the on-disk file header by encrypting it with IBE, using the new file's pathname as the public key string. While the metadata request is in flight, reads and writes can proceed as long as a copy of the file's cleartext data $K_F{}^D$ is cached in memory.

After the cached key times out, the file is essentially "locked" on disk by the IBE encryption, preventing subsequent file accesses until the metadata service confirms its success. On confirmation, the metadata service returns the IBE private key, allowing Keypad to "unlock" the file.

Internal mechanisms
Identity-Based Encryption for Metadata Updates



(a) Without IBE



(b) With IBE

Internal mechanisms
Using Paired Devices for Disconnected Access

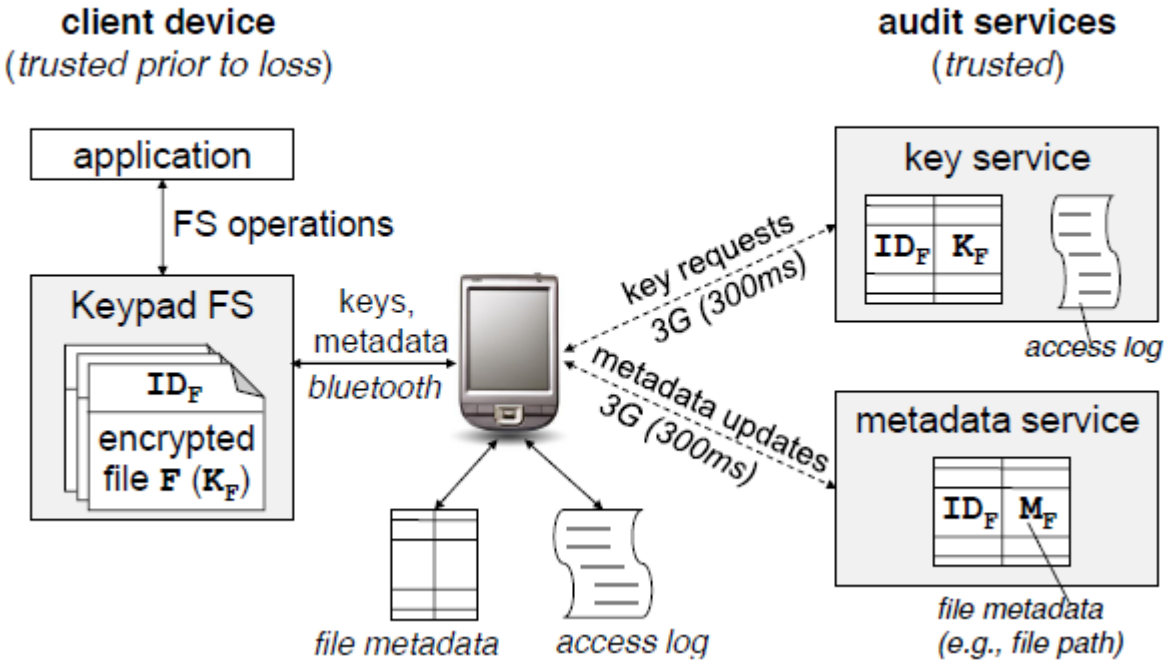
The phone is configured to hoard any recently used keys, cache them until connectivity is restored, log any accesses and metadata updates to the local disk, and upload the logs when connectivity returns. If only the laptop is lost, the phone is used along with the audit service logs to provide a full audit trail.

Here the phone is configured to perform aggressive directory-level key prefetching and caching. On a key miss, the laptop contacts the phone via bluetooth and the phone returns the key, if available; otherwise the phone fetches the missed key and other related keys from the key service and returns the key to Keypad.

If the phone is stolen along with the laptop, then the audit service will list more files as exposed than if the laptop were stolen alone.

In addition to supporting (increasingly rare) disconnected cases, the paired-device architecture has another advantage: it can improve performance over slow mobile networks without sacrificing auditing. Because the laptop–phone link is relatively efficient, the paired phone can improve laptop performance by acting as a cache for it.

Internal mechanisms
Using Paired Devices for Disconnected Access



Internal mechanisms Partial Coverage

Not all files necessarily require audit log entries. For example, as a trivial optimization we could exclude non-sensitive files such as binaries, libraries, and configuration files from Keypad's audited protection domain. In this scenario protected files are encrypted locally and their keys and metadata are stored remotely; unprotected files are (optionally) encrypted locally, but their encryption keys are derived from the user's login credentials.

The benefits of this optimization are obvious: Keypad's performance and availability costs are only incurred for protected files. There is also a risk: if a sensitive file is accidentally placed in an untracked file or directory, the audit logs will not reveal accesses to that sensitive data. One reasonable protection policy is to track accesses to any file in crucial directories, such as the user's home and temporary directory (e.g., /home and /tmp on Linux).

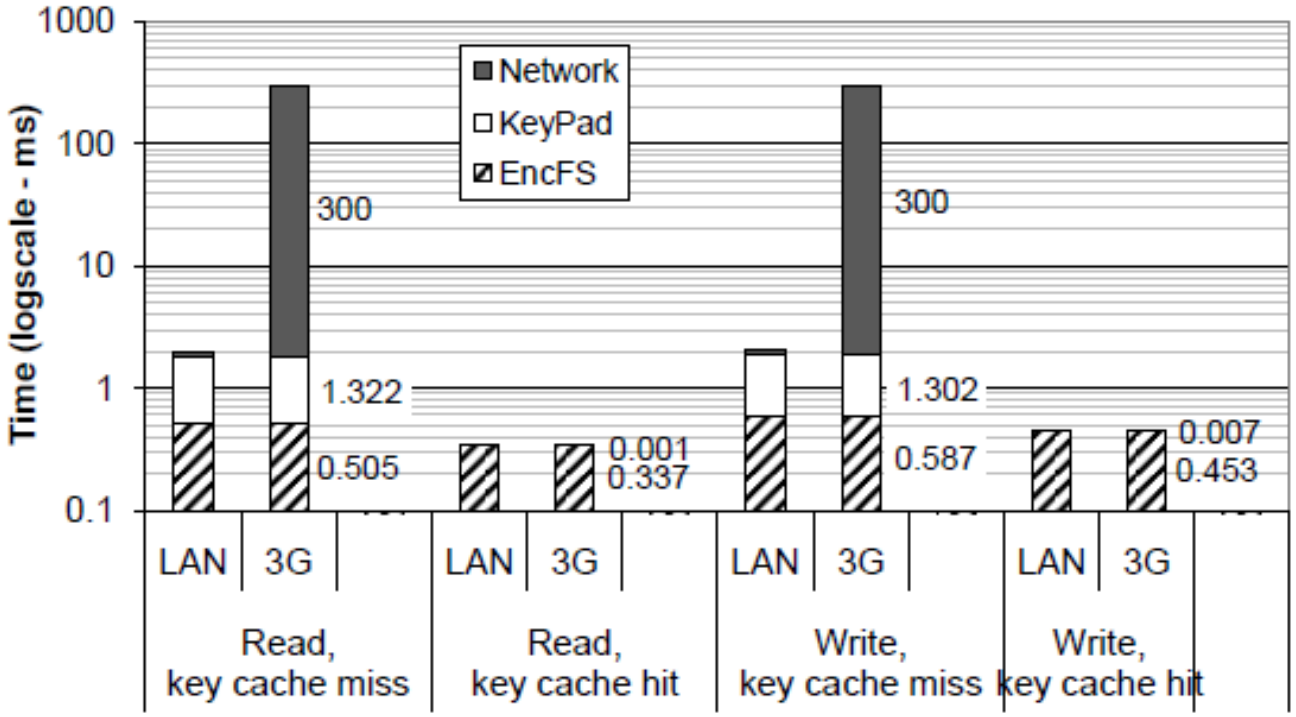
Evaluation results

For our experiments, we used an eight-core 2GHz x86 machine running Linux 2.6.31 as our client. Our key service and name service daemons ran on 8 core 2.6GHz servers with 24GB of RAM, connected via gigabit Ethernet. We used Linux's traffic control utility to emulate different network latencies. We did not emulate different bandwidth constraints, however, Keypad's bandwidth requirements are very low. During a 12-day period in which one of our authors used Keypad continuously, average Keypad bandwidth was under 5 kb/s, with occasional spikes up to 45 kb/s.

Throughout the evaluation, we emulate the following RTTs for various networks: 0.1ms RTT for a LAN, 2ms RTT for a wireless LAN (WLAN), 25ms RTT for broadband, 125ms RTT for a DSL network, and 300ms RTT for a 3G cellular network. To illustrate network latency effects on Keypad performance, we often use examples from extreme network conditions, such as fast LANs and slow 3G networks, even though popular mobile connections today rely on WLAN and 4G.

To understand where the time goes for Keypad operations, we microbenchmarked file content (read and write) and metadata (create, rename, and mkdir) operations. Our measurements included client, server, and network latencies, as well as latency contributions for EncFS and Keypad.

Evaluation results
Encryption key caching

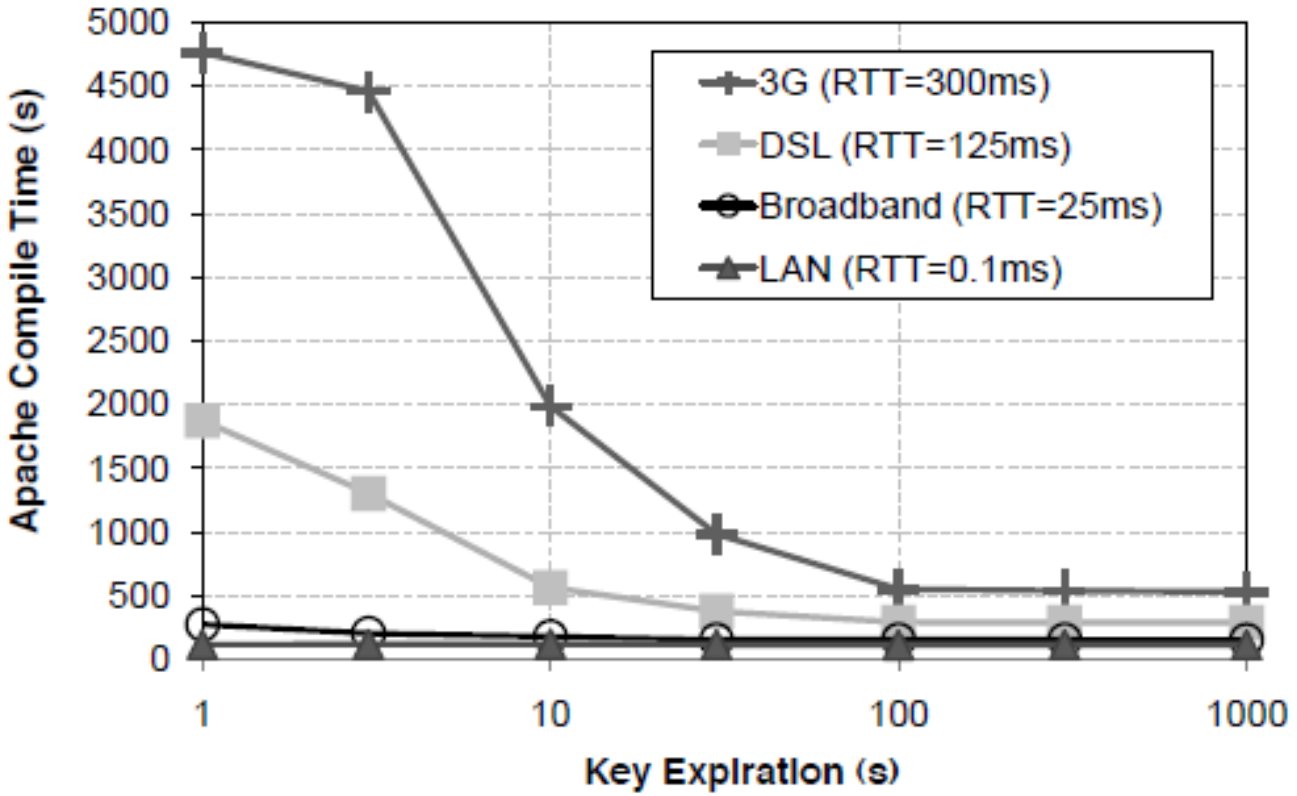


(a) FS Content Operations: `read`, `write`.

Evaluation results
Encryption key caching

This figure shows the latency of file read and write operations for two cases: key-cache misses, which must fetch the key from the the server, and key-cache hits, which use a locally cached key. For each case we show data for two extreme networks: a fast 0.1ms-RTT LAN and a slow 300ms-RTT 3G network. The results show that misses are expensive on both networks, but for different reasons. On a LAN, the network is insignificant, but Keypad adds to the base EncFS time due to the XML-RPC marshalling overhead. On 3G, network latency dominates. When the key-cache hits, both the network and marshalling costs are eliminated; a file read with a cached key is only 0.01ms slower than the base EncFS read time of 0.337ms. This shows the importance of key caching to avoid misses, which we accomplish by carefully choosing our expiration and prefetching policies.

Evaluation results
Encryption key caching



Evaluation results
Encryption key caching

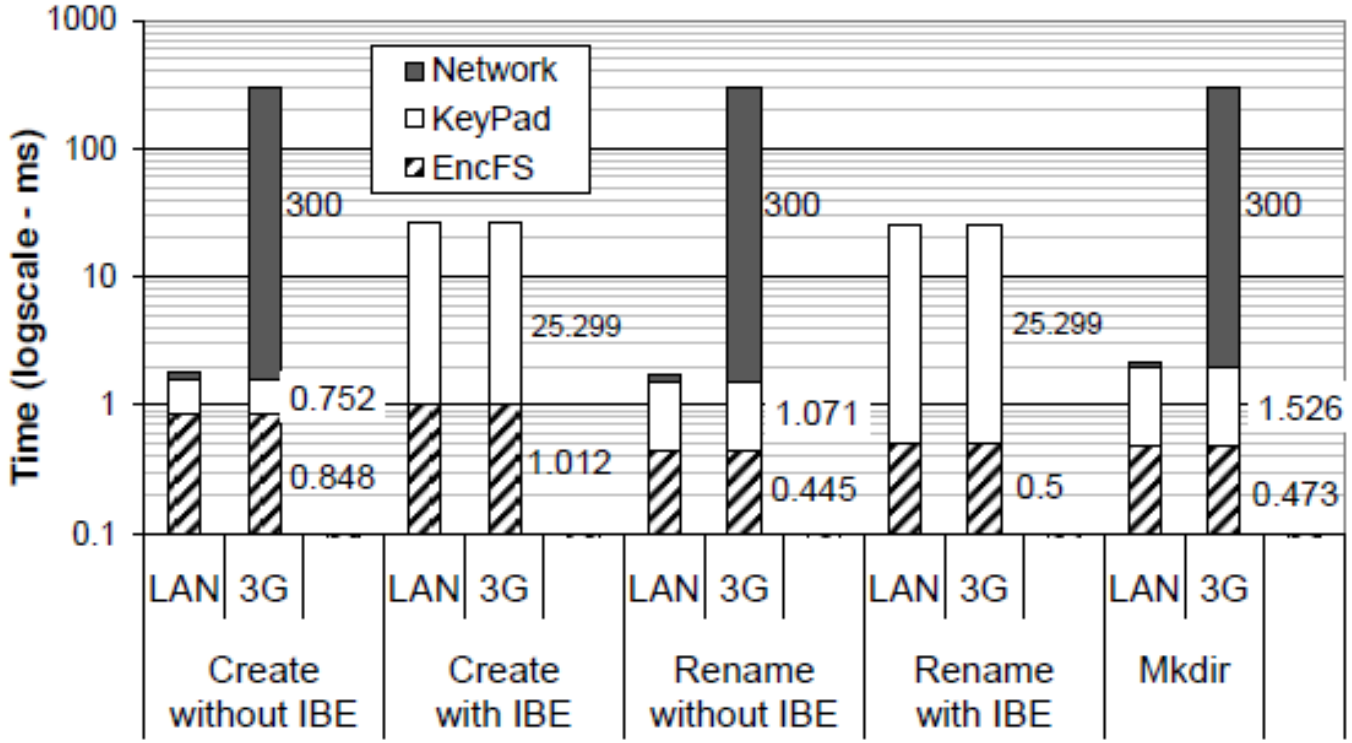
Key caching is crucial to performance. Even a cache with one-second expiration time has significant impact: 18% improvement on a LAN and 4.9x on 3G, relative to no caching at all. This figure shows additional improvements for Apache compilation time as expirations are lengthened beyond one second. No optimizations other than caching are enabled here. Our results suggest that short expiration times are sufficient to extract nearly all the benefits. For LAN, Broadband, or DSL latencies, an expiration of 10s or so is optimal. Over 3G, a 100s key expiration time achieves all the benefit and provides 8.6x improvement over 1s (from 79.4 minutes down to 9.2 minutes). In comparison to EncFS, Keypad's performance degradation for 100s expiration times is already small over a LAN (5.3% overhead over EncFS), while for the other network types, further optimizations are required for performance.

Note that a 100s timeout is extremely small. To benefit from cached keys, a thief needs to steal the device within 100 seconds of the user's last access. Even in such cases, the user will know which files were exposed. We therefore believe that we can achieve both good performance and accurate auditing with these parameters.

Evaluation results
Directory-key prefetching.

Key caching alone avoids many key service requests: of the 75,744 reads and writes in the Apache compilation, only 486 involve the server when using a 100s expiration time. Directory-key prefetching avoids additional server requests. Prefetching a directory key on the first, third, or tenth miss in a directory results in 101, 249, and 424 key-cache misses, which translates into 63.3%, 24.1%, and 2.4% improvements, respectively, over not using directory-key prefetching over 3G.

Evaluation results
Identity-Based Encryption

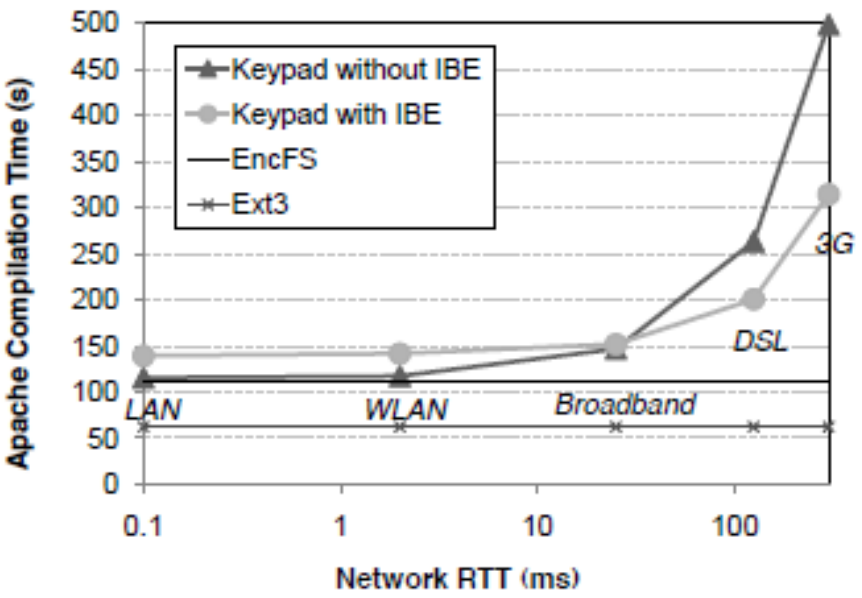


(b) FS Metadata Operations: `create`, `rename`, `mkdir`.

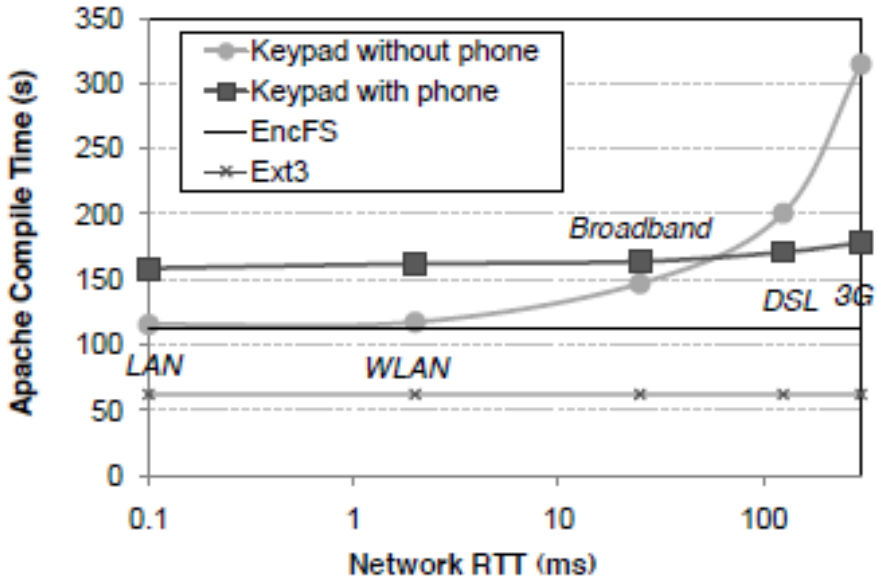
Evaluation results
Identity-Based Encryption

This figure shows the latency of file metadata update operations. For create and rename, we show latency with and without IBE; mkdir is shown only without IBE, since it does not benefit from this optimization in our prototype. Without IBE, metadata update latency is driven primarily by network RTT: file creation takes 1.618ms on a LAN, and 302ms over 3G. With IBE, metadata update latency is independent of network delay and is dominated by the computational cost of IBE itself. The figure shows that IBE meets its goal of improving performance of metadata updates over 3G. While IBE would add overhead for a LAN, it is unnecessary and would be disabled in the LAN environment.

Evaluation results
IBE and device pairing



(a) Effect of IBE.



(b) Effect of Device Pairing.

Evaluation results
Device pairing

Our paired device design is aimed at facilitating disconnected operation, but it can also provide performance benefits for high-latency network environments. Last figure shows the effect on the Apache workload of using paired device as a caching proxy for key and metadata services. Two conclusions can be reached from the figure.

First, performance for disconnected operation over Bluetooth should be similar to or better than that of a broadband connection (the latencies are similar).

Second, pairing with another device is always beneficial for performance over cellular networks, because most operations only traverse the lower latency Bluetooth link. Obviously the paired device should not be used if fast networks are available, where Keypad is already efficient enough compared to EncFS.

Evaluation results

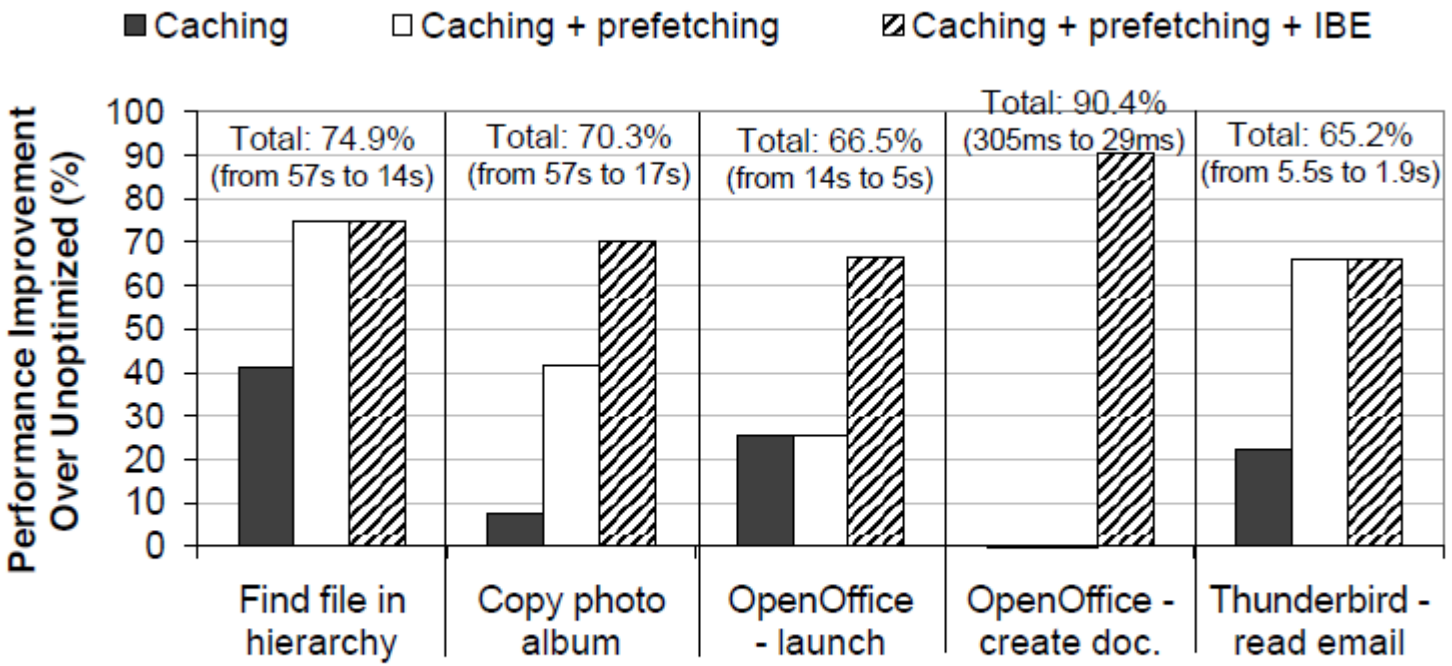


Figure 9. Impact of Optimizations on Various Applications.

Evaluation results

Application	Task	Time (seconds)					
		EncFS	Keypad				
			LAN (RTT=0.1ms)	WLAN (RTT=2ms)	Broadband (RTT=25ms)	DSL (RTT=125ms)	3G (RTT=300ms)
OpenOffice Word Processor	Launch	0.5	0.5 | 0.5	0.6 | 0.6	1.3 | 1.3	2.7 | 2.7	4.6 | 4.6
	New document	0.0	0.0 | 0.0	0.0 | 0.0	0.0 | 0.0	0.0 | 0.1	0.0 | 0.3
	Save as	1.4	1.4 | 1.4	1.4 | 1.4	1.5 | 1.5	1.6 | 1.8	2.0 | 2.3
	Open	1.7	1.7 | 1.7	1.8 | 1.8	2.0 | 2.2	2.1 | 4.0	2.1 | 7.5
	Quit	0.1	0.1 | 0.1	0.1 | 0.1	0.3 | 0.4	0.4 | 0.7	0.4 | 1.2
Firefox	Launch	3.7	3.7 | 3.7	3.8 | 3.8	4.4 | 4.4	6.0 | 6.0	8.8 | 8.8
	Save a page	0.7	0.7 | 0.7	0.7 | 0.7	0.7 | 0.8	0.9 | 1.5	1.3 | 2.8
	Load bookmark	4.5	4.5 | 4.5	4.5 | 4.5	4.5 | 4.6	4.5 | 5.0	4.5 | 5.7
	Open tab	0.2	0.2 | 0.2	0.2 | 0.2	0.2 | 0.2	0.2 | 0.4	0.2 | 0.8
	Close tab	0.0	0.0 | 0.0	0.0 | 0.0	0.0 | 0.0	0.0 | 0.1	0.0 | 0.3
Thunderbird	Launch	1.3	1.3 | 1.3	1.3 | 1.3	1.4 | 1.4	2.0 | 2.0	3.1 | 3.1
	Read email	0.3	0.4 | 0.4	0.4 | 0.4	0.5 | 0.6	1.0 | 1.5	1.9 | 2.5
	Quit	0.2	0.2 | 0.2	0.2 | 2.2	0.2 | 0.4	0.2 | 1.3	0.2 | 2.9
Evince PDF Viewer	Launch	0.1	0.1 | 0.1	0.1 | 0.1	0.1 | 0.1	0.1 | 0.1	0.1 | 0.4
	Open document	0.1	0.1 | 0.1	0.1 | 0.1	0.1 | 0.1	0.2 | 0.2	0.4 | 0.4
	Quit	0.0	0.0 | 0.0	0.0 | 0.0	0.0 | 0.0	0.0 | 0.0	0.0 | 0.0

x | y: x = time with warm key cache
y = time with cold key cache

Table 1. Typical Application Performance Over Keypad. For Keypad, we show both warm and cold key-cache times, separated by a |.

Conclusions

This paper described Keypad, an auditing file system for loss- and theft-prone devices. Unlike basic disk encryption, Keypad provides users with evidence that sensitive data either was or was not accessed following the disappearance of a device. If data was accessed, Keypad gives the user an audit log showing which directories and files were touched. It also allows users to disable file access on lost devices, even if the device has been disconnected from the network or its disk has been removed. Keypad achieves its goals through the integration of encryption, remote key management, and auditing. Our measurements and experience demonstrate that Keypad is usable and effective for common workloads on today's mobile devices and networks.

Questions?

Thank you!