Trusted storage

Anjo Vahldiek* *MPI-SWS*
Eslam Elnikety *MPI-SWS*
Ansley Post *Google*
Peter Druschel *MPI-SWS*
Deepak Garg *MPI-SWS*
Johannes Gehrke *Cornell*
Rodrigo Rodrigues *MPI-SWS*

We study the properties, design, implementation and performance of *trusted storage*, an architecture that ensures the integrity, confidentiality and accountability of data, by enforcing storage policies at the lowest layer of a storage system, within the hardware and firmware of disk enclosures. The guarantees provided by trusted storage depend only on the integrity and correctness of the trusted device/enclosure firmware and hardware, not on the absence of bugs and security vulnerabilities in any higher level software of a system and operator error or malice.

Trusted storage primitives enable applications to associate and enforce a policy with each data object they create, and to obtain firmware-generated, cryptographically signed certificates, which attest to a given stored data object's name and content hash, the policy in effect for the object, access history for the object, as well as certain properties of the device including its approximate location. A typical policy states the conditions under which a data object may be read, updated, or deleted, to what extent access to the object should be recorded, how often the object should be scrubbed, and the conditions under which the policy may be changed.

To implement this functionality, each trusted storage device has a globally unique identifier and a unique hardware-protected asymmetric cryptographic private key, whose corresponding public key is certified by the manufacturer. Using this private key, the device can cryptographically sign messages, generate short-term session keys and establish secure connections to other trusted storage devices, trusted servers (e.g. time, location, firmware update) and client computers by tunnelling securely through the (untrusted) operating system.

Trusted storage addresses the problems created by two complementary trends in computing. On the one hand, both the volume and the value of digitally stored data are increasing. On the other hand, the risk associated with ensuring the integrity and confidentiality of data is increasing for two main reasons. First, storage system software is increasingly more complex, which in turn increases the likelihood of software bugs and security vulnerabilities. There is mounting evidence that data corruption is not uncommon, and that software bugs are among the causes [3]. Trusted storage protects data integrity and confidentiality despite software bugs and vulnerabilities outside the storage device. Second, individuals increasingly entrust their data to third-party providers for storage and processing (e.g., Web mail, DropBox, Amazon S3). Most of these providers are reputable companies that care about their customers, follow best practices and respect applicable laws. Nevertheless, accidents happen, as several recent examples demonstrate [2]. Today, customers have to blindly trust that providers respect policies and take appropriate measures to safeguard data from threats. Trusted storage would allow customers to specify policies associated with their data, e.g., regarding data access restrictions, the physical location of disks containing their data, the number of copies that are maintained, or the scrubbing frequency. Signed certificates from trusted storage would attest correctness of policies in effect for a customer's data, independent of the service provider's assurance. Moreover, compliance with those policies would depend only on the integrity of the lowest storage layer, which could be certified by an independent authority.

Policies

Trusted storage supports a declarative policy language used to state the conditions for reading, modification of or deletion of an object, access logging, as well as the conditions under which the policy may be changed. Examples of storage policies include combinations of the following:

Identity-based access restriction Allow access to au-

*Will present the poster and WiP.

thorized principals authenticated through a secure channel to the trusted storage device. Prevents unauthorized access.

Attestation-based access restriction Allow access subject to remote attestation of an acceptable hardware/software configuration (or another trusted storage device). Ensures that data can only be obtained by devices with a hardware and software configuration trusted to respect the owner's policies.

Location-based access restriction Allow access to data only if the trusted storage device is in a specific geographic region or country. Useful to enforce legislation and policy regarding data dissemination and storage.

Expiration Allow read access only until a given date. Useful to enforce data retention policies.

Time capsule Allow read access only after a given date. Useful to ensure the confidentiality of information prior to a given release date, e.g., in the case of classified information.

Storage lease Allow modifications or deletions only after a given date. Useful to protect the integrity of objects, snapshots and backup archives with a pre-determined lifetime.

Quota-based access restriction Allow a fixed number of read accesses. Useful to enforce DRM and other licensing restrictions.

API and performance

The API of a trusted storage device is a superset of the traditional block device API. One command can be used to associate an object name (e.g., /usr/bin/csh) and a policy with an arbitrary set of extents in the storage device. Conventional read and write access to any block within these extents are then checked against the policy. Another command can be used to obtain a signed certificate about an object; the object can be specified by its name or one of its block numbers.

Our preliminary experiments indicate that using a small amount of flash memory to store policies and associated metadata, the overhead for enforcing policies can be within a few percent points. This holds for hybrid magnetic disk (conventional disks with a few GB of flash memory) as well as solid-state disks.

Guarantees

The guarantees provided by a trusted storage device depend on the correctness of its firmware, which is related to its complexity. Storage firmware, while of substantial complexity, is still far smaller, simpler and subject to a much lower rate of change than the remaining system software. Moreover, the correctness and integrity of trusted storage firmware could be certified by an independent organization.

For policies that depend on real time, a trusted storage device could connect to an existing trusted time source that issues signed real time stamps. Protocols for communication with such sources include a fresh nonce to prevent replay attacks. Since a granularity on the order of a day is likely sufficient, the resulting overhead is small. Similarly, for policies that depend on geographic location, a few strategically placed trusted servers could be used to obtain an approximate location based on RTTs. The resulting location fix is sufficiently accurate to distinguish continents, perhaps countries.

Related work

Trusted computing leverages low-level trusted hardware primitives to ensure high-level security properties of a system. Trusted storage similarly leverages trusted primitives to ensure integrity, confidentiality and accountability of stored data. Self-securing storage [5] transparently retains shadow copies of modified data for a predetermined period. Self-encrypting disks ensure data confidentiality. Trusted storage addresses integrity, confidentiality and accountability, and supports more general policies. Differentiated storage [1] provides performance related policies for different classes of requests. Active disks [4] allow the execution of general application code in storage devices for performance and flexibility. Trusted storage relies on on a declarative policy language to provide integrity, confidentiality, and accountability.

References

[1] MESNIER, M., CHEN, F., LUO, T., AND AKERS, J. B. Differentiated storage services. In *Proc. SOSP'11*.

[2] MUSIL, S. Google blames software update for lost gmail data. `http://tomkarpik.com/articles/massive-data-loss-bug-in-leopard/`, 2011.

[3] PANZER-STEINDEL, B. Data integrity. `http://indico.cern.ch/getFile.py/access?contribId=3&sessionId=0&resId=1&materialId=paper&confId=13797`, 2007.

[4] RIEDEL, E., FALOUTSOS, C., GIBSON, G. A., AND NAGLE, D. Active disks for large-scale data processing. *Computer 34* (June 2001), 68–74.

[5] STRUNK, J. D., GOODSON, G. R., SCHEINHOLTZ, M. L., SOULES, C. A. N., AND GANGER, G. R. Self-securing storage: protecting data in compromised system. In *Proc. OSDI 2000*.