Chit-based Remote Storage

Citation: Paluska, J.M., and S. Ward. "Chit-based remote storage." Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on. 2010. 517-521. ©2010 IEEE

As Published: http://dx.doi.org/10.1109/PERCOMW.2010.5470593

Publisher: Institute of Electrical and Electronic Engineers

Persistent URL: http://hdl.handle.net/1721.1/61730

Version: Final published version: final published article, as it appeared in a journal, conference proceedings, or other formally published context

Chit-based Remote Storage

Justin Mazzola Paluska and Steve Ward
{jmp,ward}@mit.edu
MIT CSAIL, Cambridge, MA, U.S.A.

***Abstract*—We propose a model for reliable remote storage founded on contract law. Consumers submit their bits to storage providers in exchange for a *chit*. A chit is a cryptographically secure, verifiable contract between a consumer and the storage provider. A user can use the chit to request the bits from the provider. In return, the provider must be able to supply the chit's bits to the user according to the terms of the chit. If the storage provider cannot supply the bits, then the provider must pay a penalty. Chits make explicit the penalty for losing data, allowing users to value their data and providers to quantify the financial risk they have undertaken by storing a set of chits. We envision chit-based storage being used as a reliable mechanism for users in pervasive environments.**

I. The Storage Problem

Alice is an ordinary computer user in a computationally pervasive environment. She has several computer in her home and office, as well as a smartphone. Each computer has its own storage. Alice is computer literate and regularly uses several large applications, but doesn't know how to maintain her systems. She never backs up her disks and is not sure how she can even back up her smartphone. Unfortunately, once in a while, a disk crashes or she drops her phone and she loses data, time, and money. Enterprise-level storage service providers who know how to reliably store bits would like to solve Alice's storage problem, hopefully making a profit in the process. The question is, then, what is the appropriate interface between a storage consumer in a computation-rich environment and a storage service provider?

Consumers want providers that are reliable and accountable to their failures while providing exactly the storage products that they need. They also want the ability to switch among providers without much hassle. No such commoditized service exists today. While dozens of online storage providers exist—from small companies like Mozy Backup, Xdrive, and rsync.net [1], [2], [3] to large ones like Amazon.com, Apple, and Microsoft [4], [5], [6]—none of these companies guarantee their ability to reliably store bits. Instead, their terms of service explicitly disavow any liability for data loss. Consumers are left to trust their data based on the reputation of their provider.

Storage service providers, on the other hand, need a way to differentiate their services from other providers and a way to offer different services to different customers. If they take on financial liability for storing customer's data, the providers also need a way of quantifying the value of storing a particular set of bits.

The rest of this paper explores *chits*, one interface model for remote storage that meets the requirements of both consumers and providers. A chit is a cryptographically secure and verifiable contract (in the legal sense) between the consumer and provider for storage of certain bits. We believe chits may form the technical basis for a competitive market for reliable remote storage: our goal with this paper is to explore the potential ways chits may be used to create a robust storage market, not to outline any particular chit implementation.

II. Bits and Chits

In its most basic form, our model involves a user sending a provider (1) an aggregate of bits over an open protocol and (2) some form of payment, in return for a cryptographically secure chit. The chit records the contractual relationship between the user and the provider including any data needed to verify the successful execution of the contract. The contract may be free-form text, URIs, or machine-readable code. The chit is verifiably signed by both the user and the provider. Hence, it cannot be modified by the user or the provider without the consent of both, nor can it be forged by a third party.

The chit encompasses the provider's promise to provide the user, on demand, with the original aggregate of bits used to form the chit. If the provider cannot provide the original bits, the chit prescribes a penalty to be assessed against the provider. The chit's contract may include constraints on the provider's obligations, such as a time limit on how long the provider must keep the chit's data or a response time under which the provider must provide the data. Each chit may have different constraints embedded in it, allowing the provider to, in general, provide a range of reliability and speed options. The consumer, on the other hand, may trust the provider with his bits because the contract (and its associated penalties) are backed with the force of contract law.

We assume that contracts do not require the provider to keep bits secret, but merely that they be reliably stored. Although it may be possible to augment our model with secrecy provisions, we assume here that any such requirements may be met by the user's encryption of data before it is sent to the provider.

978-1-4244-5328-3/10/$26.00 ©2010 IEEE

A. In Chits We Trust

Chits relieve users of the burden of reliably storing large collections of data—such as an entire disk backup—for the smaller burden of reliably storing a few chits. When viewed this way, chits are merely a form of super-compression from hundreds of gigabytes to hundreds of bytes. The user must still take care of storing the chits since the chits are as valuable as the original data they represent. Fortunately, chits are much easier to handle than the original data collection: chits may be easily copied to multiple devices or even copied to small flash sticks and stored in a fire-proof safe or bank safety deposit box.

However, we can remove the burden of reliably storing chits from the user if the chit provider includes a service where the user can list, verify, and access, on demand, the chits that the user has stored with the provider. If the user loses all of her copies of the chits, she need only ask the provider for all of the chits that it has for her.

Why should the user trust her provider? The answer is the penalty the provider must pay for lying to his customers. The provider must, as a part of his contractual obligations, catalog and report the chits for each of his customers. Customers can regularly check their collection of chits. If the provider fails to catalog a chit, the customer who has the missing chit has a verifiable complaint that can damage the provider's reputation or invoke monetary damages to be paid to the customer. Such a contract violation can be detected with high probability, since customers will rarely lose all of their chits.

B. Economics and Risk

As chits make explicit the penalty for losing the data they represent, they imply a quantitative value for that data. The continuum of values may be used to distinguish between data which is merely inconvenient to reproduce from that which is essential to a company's continued existance. Conventional storage models tend to abstract out this risk, promising perfectly reliable storage—a fiction that fits neither the consumer's needs nor the producer's capabilities.

Chits and related interfaces that attach economic value to data suggest a "data insurance" model for storage. One might imagine a multi-tier data insurance industry, in which consumer-facing companies buy storage wholesale and sell it retail, making decisions about the degree of redundancy used to store the data that are informed by its explicit value. The provider is free to make risk-cost tradeoffs on how reliably to store data on a per-chit basis. For example, the provider may choose to replicate the some chits across multiple data centers to reduce risk, albeit at increased cost. The provider may also take out insurance to cover its liablities, or much like the re-insurance industry backs normal insurance companies, store data on an upstream wholesale chit provider.

This risk-optimized model contrasts strongly with existing risk-oblivious storage services. By quantifying the value of the information stored in terms of enforceable compensation for its loss, both the supplier and buyer of storage can improve the cost effectiveness of their operation. Moreover, we argue that this model is more transparent than those of online backup services which obscure their value proposition by claiming trustworthiness while disclaiming liability for the data they store.

C. Modeling Risk Independence

A storage service might try to hedge risk by storing high-value chits redundantly using several competing chit-based storage vendors, assuming the probability of irrecoverable loss of the chit is the product of the loss risk probabilities of each vendor.

However, each vendor might subcontract the storage to the same wholesaler, who stores each copy of the original user's data on the same physical disk. The failure risk of the single disk remains constant while the total compensation received by the storage service is the sum rather than the product of the penalties—exposing that service to uncompensated risk. Such a situation is more than just academic: the failure of a StorageNetworks was hastened by the bankruptcy of its wholesale storage provider Exodus [7].

To address such situations, we must explore ways to represent the statistical independence of risks associated with sets of chits. We note, in passing, a potentially grim analogy to the problem of risk evaluation in pooled financial instruments such as mortgage-backed securities, blamed by many [8] for the current meltdown of the global economy.

III. Mechanism

In order to use chits, a user installs a client program that speaks an open chit protocol to servers run by providers. The client programs are responsible for encryption and transmission of bits to the provider as well as managing the resulting chits.

Client programs may offer a variety of services backed by chits. For example, one client may act as traditional automatic backup client, a "trickle-charge" archive client [9], or a virtual write-once read-many drive [10]. Other clients store may use a chit-based provider as a primary store and use the local disk only as a cache for files to hide network latency. The user may choose each backup service based on the costs of the provider's chits.

The chit protocol is shared by all chit providers so as to prevent lock-in at any point in the data storage supply chain. The exact contents of an individual chit may vary from provider to provider, but should be readable and verifiable by any open tool the user may choose to use.

A. Foundation Chit Format

While the exact format of a chit may vary from provider to provider, all chits share some common characteristics. First, all chits must be verifiable by customers, preferably using tools chosen by the customer, implying some standardization for chits. Second, since chits rely on cryptography, and in particular, secure hashing, providers must have a way of upgrading chits as cryptographic hashes are broken, while still letting the user verify the new chits.

B. Auditing and Consumer Confidence

Chits do not protect consumers from unscrupulous or fly-by-night chit providers that issue chits knowing that they cannot properly maintain data or pay the penalties for lost data. Consumer protection in such cases can be provided by a variety of mechanisms, from auditing from outside groups [11] or sufficient disclosure of insurance policies covering the provider's liabilities, much like consumers are protected from financial entities.

IV. Related Work

Hasan et al. [7] identify identify several business models storage providers may follow, as well as provide a case study of two storage providers, the defunct StorageNetworks and IBM. They note that trust in a provider's reliability is critical to the success of a provider, attributing the failure of StorageNetworks to a lack of such trust. We view chits as a vehicle for trustable storage contracts to a much larger community of ordinary users.

Kher et al. [12] and the CATS system [13] attempt to establish trust between storage service providers and clients by building secure and automated accounting systems. Both works provide ways for both clients and servers to openly verify data access or resource usage. They do not address data integrity. A Chit-based system may use many of their mechanisms to make chits into verifiable receipts.

Amazon.com's Simple Storage Service (S3) and Elastic Block Store (EBS) [4] is a commercial storage service specializing in large-scale data storage. Like the other storage providers, Amazon does not guarantee its data storage. Amazon does give users hedge storage failures by putting storage in different "availablility domains". S3/EBS might be useful as a upstream storage provider for a small chit provider.

Several large-scale peer-to-peer storage systems like OceanStore [14], [15], Plutus [16], FARSITE [17], and PAST [18] have been developed by the research community. These systems aim to be storage utilities in an infrastructure roughly analogous to the national power grid. Larger chit providers may run their own private storage utility. In OceanStore, users contract with a *responsible party* that stores data on their behalf in the utility system. Chits can formalize the relationship between users and the storage utility. Lillibridge et al. and Cox et al. have both proposed cooperative storage services [19], [20] where volunteers agree to store each other's data. Such systems could be useful in forming chit collaboratives among smaller storage providers. In particular, Samsara [21], which builds on top of Pastiche, introduces the concept of storage "claims" which allow users to barter storage, much as chits allow users to purchase storage.

Credential-based systems like Fileteller [22], on the surface seem similar to a chit system. Fileteller relies on a micropayment system as an access control mechanism: you must pay to get a token and present your token to access storage. KeyNote [23] (on which Fileteller is based) allows service providers to make risk assessments based on the payment history of clients. Chits address the inverse problem: once a client has paid for a service, how does he ensure that his data persists without needed to constantly check it.

The Creative Commons [24] (along with its offshoots like Science Commons) provide ready-made licenses for content that encourage sharing. All of the licenses that Creative Commons provide also contain RDF that provides a machine-readable bridge between the legalese of the license and the content licensed. Chits may benefit from the lessons learned when developing the machine-readable aspects of the Creative Commons licenses.

On the client side, the rsync algorithm [25] is useful in online backup because the algorithm efficiently finds deltas between two sets of not necessarily local sets of files. rdiff-backup [26] uses the rsync algorithm to provide versioned backups. duplicity [27] extends rdiff-backup with encryption so that consumers need not trust their data store to keep data secret. Elements of both tools may improve chit-based client backup programs.

Ateniese et al. [28] present two models for provable data possession (PDP). In their models, storage customers compute a small, constant-sized set of metadata before sending a large data set to a storage provider. The customers can then use their metadata to make requests of the provider to check and ensure that the provider still has the data. Algebraic signatures [29] provide an alternative scheme that allows random checks with smaller data bounds. PDP schemes provide a theoretical foundation for designing efficient audit protocols in a chit system.

Finally, in order for providers to use chits profitably, they must be able to accurately assess the value of their data and the risks that may cause them to lose their data. Penalties prescribed by a chit are a proxy for the value of data; risks to the data may come from anywhere in the hardware/software stack. At the lowest-level, aggregate disk hardware failure rates [30], [31] are well known and, as such, can be accounted for in system design. While we have less of a handle on file system failure rates, work by Sivanthanu et al. [32] provides a practical theoretical framework for reasoning about the correctness of interactions between a file system

and its underlying storage nodes. Disk-scrubbing techniques [33], [34] provide ways of decreasing the likelihood of data loss in traditional systems by finding and correcting errors early, before they propagate into all archives.

Other researchers focus on cluster reliability as a whole rather than the reliability of a single node. The Google File System [35] and Amazon's Dynamo [36] both provide reliable storage systems on commodity hardware, optimizing for bandwidth and latency, respectively. HP's FAB architecture [37] uses commodity storage "bricks" to provide fault tolerance storage. Ursa Minor [38] adds online configuration of fault tolerance parameters. Each of these systems provide a myriad of ways for a chit provider to configure its private storage network.

Rao et al. [39] quantifies reliability for a combination of cluster and RAID configurations in terms of data loss events per Petabyte of storage per year, allowing storage operators to optimize their storage. Baker et al. [40] provide a model to help providers optimize for reliability. Other systems, like Glacier [41], take the view that exact failure statistics are unreliable and try to protect against correlated failures, like Internet worms, by using massive amounts of redundancy. Lastly, long-term archival systems like LOCKSS [42] provide systems that ensure the persistence of data over decades, even when data is rarely accessed. Chit providers may be able to use such systems to quantify and reduce the risks they take.

V. Concluding Remark

Our model may be viewed as a small step toward the *commoditization* of storage as a service, by encouraging explicit valuation of the information stored. Commoditization, in turn, provides the foundation for a competitive free market for computing services, an interesting alternative to historic models in which software and hardware mechanisms, rather than services, have been the economic commodities. We feel that such alternatives are essential to the evolution of pervasive computing delivery models.

References

[1] Berkeley Data Systems, Inc., "Mozy backup," http://mozy.com.

[2] Xdrive, LLC, "Xdrive: Secure online storage," http://www.xdrive.com.

[3] Rsync.net, "rsync.net: Secure offsite backups," http://www.rsync.net.

[4] Amazon.com, "Amazon Web Services products," http://aws.amazon.com/products.

[5] Apple, Inc., ".mac," http://www.apple.com/dotmac.

[6] Microsoft, Inc., "Windows live skydrive," skydrive.live.com.

[7] R. Hasan, W. Yurcik, and S. Myagmar, "The evolution of storage service providers: techniques and challenges to outsourcing storage," in *StorageSS '05: Proceedings of the 2005 ACM workshop on Storage security and survivability*. New York, NY, USA: ACM, 2005, pp. 1–8.

[8] F. Salmon, "Recipe for disaster: The formula that killed wall street," *Wired Magazine*, vol. 17, no. 3, Feb. 2009.

[9] B.-G. Chun, F. Dabek, A. Haeberlen, E. Sit, H. Weatherspoon, M. F. Kaashoek, J. Kubiatowicz, and R. Morris, "Efficient replica maintenance for distributed storage systems," in *NSDI*. USENIX, 2006.

[10] R. Pike, D. L. Presotto, S. Dorward, B. Flandrena, K. Thompson, H. Trickey, and P. Winterbottom, "Plan 9 from Bell Labs," *Computing Systems*, vol. 8, no. 2, pp. 221–254, 1995.

[11] M. A. Shah, M. B. J. C. Mogul, and R. Swaminathan, "Auditing to keep online storage services honest," in *HOTOS'07: Proceedings of the 11th USENIX workshop on Hot topics in operating systems*, 2007, pp. 1–6.

[12] V. Kher and Y. Kim, "Building trust in storage outsourcing: Secure accounting of utility storage," in *26th IEEE International Symposium on Reliable Distributed Systems*, Oct. 2007, pp. 55–64.

[13] A. R. Yumerefendi and J. S. Chase, "Strong accountability in network storage." USENIX, 2007.

[14] J. Kubiatowicz, D. Bindel, Y. Chen, S. E. Czerwinski, P. R. Eaton, D. Geels, R. Gummadi, S. C. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Y. Zhao, "Oceanstore: An architecture for global-scale persistent storage," in *ASPLOS*, 2000, pp. 190–201.

[15] S. C. Rhea, P. R. Eaton, D. Geels, H. Weatherspoon, B. Y. Zhao, and J. Kubiatowicz, "Pond: The oceanstore prototype," in *FAST*. USENIX, 2003.

[16] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: Scalable secure file sharing on untrusted storage," in *FAST*. USENIX, 2003.

[17] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. Wattenhofer, "Farsite: Federated, available, and reliable storage for an incompletely trusted environment," in *OSDI*. USENIX, 2002.

[18] A. I. T. Rowstron and P. Druschel, "Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility," in *SOSP*, 2001, pp. 188–201.

[19] M. Lillibridge, S. Elnikety, A. Birrell, M. Burrows, and M. Isard, "A cooperative internet backup scheme," in *USENIX Annual Technical Conference, General Track*. USENIX, 2003, pp. 29–41.

[20] L. P. Cox, C. D. Murray, and B. D. Noble, "Pastiche: Making backup cheap and easy," in *OSDI*. USENIX, 2002.

[21] L. P. Cox and B. D. Noble, "Samsara: honor among thieves in peer-to-peer storage," in *SOSP*, M. L. Scott and L. L. Peterson, Eds. ACM, 2003, pp. 120–132.

[22] J. Ioannidis, S. Ioannidis, A. Keromytis, and V. Prevelakis, "Fileteller: Paying and getting paid for file storage," in *Proceedings of the Sixth International Conference on Financial Cryptography*, 2002, pp. 282–299.

[23] J. Ioannidis and A. D. Keromytis, "Experience with the keynote trust management system: Applications and future directions," in *Proceedings of the 1st International Conference on Trust Management*. Springer-Verlag, 2003, pp. 284–300.

[24] "Creative Commons," http://creativecommons.org.

[25] A. Tridgell, "Efficient algorithms for sorting and synchronization," Ph.D. dissertation, Austrailian National University, February 1999.

[26] "rdiff-backup," http://www.nongnu.org/rdiff-backup/.

[27] "duplicity," http://duplicity.nongnu.org/.

[28] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security.* New York, NY, USA: ACM, 2007, pp. 598–609.

[29] T. J. E. Schwarz and E. L. Miller, "Store, forget, and check: Using algebraic signatures to check remotely administered storage," in *ICDCS*. IEEE Computer Society, 2006, p. 12.

[30] B. Schroeder and G. A. Gibson, "Disk failures in the real world: what does an mttf of 1,000,000 hours mean to you?" USENIX, 2007.

[31] E. Pinheiro, W.-D. Weber, and L. A. Barroso, "Failure trends in a large disk drive population." USENIX, 2007.

[32] M. Sivathanu, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, and S. Jha, "A logic of file systems," in *FAST*. USENIX, 2005.

[33] V. Prabhakaran, L. N. Bairavasundaram, N. Agrawal, H. S. Gunawi, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Iron file systems," in *SOSP*, A. Herbert and K. P. Birman, Eds. ACM, 2005, pp. 206–220.

[34] T. J. E. Schwarz, Q. Xin, E. L. Miller, D. D. E. Long, A. Hospodor, and S. W. Ng, "Disk scrubbing in large archival storage systems," in *MASCOTS*, D. DeGroot, P. G. Harrison, H. A. G. Wijshoff, and Z. Segall, Eds. IEEE Computer Society, 2004, pp. 409–418.

[35] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *SOSP*, 2003, pp. 29–43.

[36] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, , and W. Vogels, "Dynamo: Amazon's highly available key-value store," in *SOSP*, 2007.

[37] S. Frølund, A. Merchant, Y. Saito, S. Spence, and A. C. Veitch, "Fab: Enterprise storage systems on a shoestring," in *HotOS*, M. B. Jones, Ed. USENIX, 2003, pp. 169–174.

[38] M. Abd-El-Malek, W. V. C. II, C. Cranor, G. R. Ganger, J. Hendricks, A. J. Klosterman, M. Mesnier, M. Prasad, B. Salmon, R. R. Sambasivan, S. Sinnamohideen, J. D. Strunk, E. Thereska, M. Wachs, and J. J. Wylie, "Ursa minor: Versatile cluster-based storage," in *FAST*. USENIX, 2005.

[39] K. K. Rao, J. L. Hafner, and R. A. Golding, "Reliability for networked storage nodes," in *DSN*. IEEE Computer Society, 2006, pp. 237–248.

[40] M. Baker, M. Shah, D. S. H. Rosenthal, M. Roussopoulos, P. Maniatis, T. Giuli, and P. Bungale, "A fresh look at the reliability of long-term digital storage," in *EuroSys '06: Proceedings of the ACM SIGOPS/EuroSys European Conference on Computer Systems 2006*. New York, NY, USA: ACM, 2006, pp. 221–234.

[41] A. Haeberlen, A. Mislove, and P. Druschel, "Glacier: Highly durable, decentralized storage despite massive correlated failures," in *NSDI*. USENIX, 2005.

[42] P. Maniatis, M. Roussopoulos, T. J. Giuli, D. S. H. Rosenthal, and M. Baker, "The lockss peer-to-peer digital preservation system," *ACM Trans. Comput. Syst.*, vol. 23, no. 1, pp. 2–50, 2005.

[43] *Proceedings of the FAST'07 Conference on File and Storage Technologies, February 13-16, 2007, San Jose, California*. USENIX, 2007.

[44] *Proceedings of the 5th Symposium on Operating System Design and Implementation*. USENIX, 2002.

[45] *Proceedings of the FAST '03 Conference on File and Storage Technologies, March 31 - April 2, 2003, Cathedral Hill Hotel, San Francisco, California, USA*. USENIX, 2003.

[46] *Proceedings of the FAST '05 Conference on File and Storage Technologies, December 13-16, 2005, San Francisco, California, USA*. USENIX, 2005.