Don't Trust Your Roommate
or
Access Control and Replication Protocols in "Home" Environments

Vassilios Lekakis, Yunus Basagalar, and Pete Keleher
$\{$*lex, yunusb, keleher*$\}$*@cs.umd.edu*

Abstract

A "home" sharing environment consists of the data sharing relationships between family members, friends, and acquaintances. We argue that this environment, far from being simple, has sharing and trust relationships as complex as any general-purpose network.

Such environments need strong access control and privacy guarantees. We show that avoiding information leakage requires both to be integrated directly into (rather than layered on top of) replication protocols, and propose a system structure that meets these guarantees.

1 Introduction

The research community has recently produced a plethora of work describing replication mechanisms and policies for sharing personal data in "home" environments [7, 8, 13, 16, 17, 18, 19, 22, 23, 25]. Table 1 summarizes the main emphasis of these projects, which notably does not include security. *Personal data* could include digitized versions of music, images, videos, as well as financial records, contact lists, etc. Much of this work describes a number of ways to express replication policies semantically, and these policies are used to replicate personal data across multiple devices. These approaches have the potential to dramatically change how personal data is managed, but we argue that the potential could be even greater but for limiting assumptions in many of these projects.

The first limiting assumption is that personal data consists primarily of read-only multimedia files. Computer science researchers are often motivated by use cases that are more applicable to themselves than the average, relatively non-technical adult. Given the recent explosive growth in personal computing devices, however, the amount and diversity of data and devices used by researchers might be a good predictor for the non-technical adult a few years, or even months, from now. If so, personal data includes not only read-only multimedia files, but also mutable files, including work-related documents, personal projects, financial records, and many other types of data. Personal devices could include a range of desktop machines and mobile devices, including laptops, tablets, and smart phones. We contend that replication policies appropriate for read-only multimedia are not sufficient to handle this diversity of data types and devices.

The second, and more important, limiting assumption is that the "home" environment is inherently single-user. Most current work considers only replication across an individual's data, and assumes that each device should be able to replicate any personal data. This is clearly false. As a simple example, consider that an individual's audio and video files might not only be replicated across his/her own devices, but also across the devices of family members, roommates, a dorm floor, friends, and possibly mere acquaintances, and least when they are within physical proximity. Furthermore, this same data might be shared across collaborators on a hobby project, despite not sharing a local area network. Not all of these users should have access to all data, and not all devices have the same need or functionality to secure data.

So...What is so Special About "Home Sharing"? The phrase "home sharing" clearly implies a number of network and sharing characteristics, but relying on these implied characteristics would result in unintentional security breaches.

There are certainly implied trust relationships, as one individual might implicitly trust a spouse with all data. However, all multimedia might not be appropriate to share with all family members, all course materials would not be appropriate with all roommates, and financial records would be tightly held. Instead, different types of data would be expected to be shared differently.

Physical proximity is clearly implied. However, many devices are by definition mobile, and will not be local all of the time. Further, sharing is not necessarily limited to the immediate family, and could easily span cities or continents.

Finally, the home sharing environment carries with it expectations of easy administration. Perhaps more than any of the other implications, this one holds true.

System	*Focus*	*System*	*Focus*
Anzere [22]	device transparency through logical predicates	HomeViews [8]	mixing of SQL and capabilities to create user-specific **read-only** data views
Bayou [26]	eventual consistency, log-based consistency, conflict resolution through merge procedures	Perspective [23]	facilitates easy data sharing between multiple users through logical data predicates
BlueFS [16]	server-based distributed file systems for mobile devices	PodBase [18]	data durability and availability, replication guarantees
Cimbiosys [19]	semantic data management through content filters; focus on efficiency (eventual filter consistency, eventual knowledge singularity)	PRACTI [1]	framework that offers combinations of partial replication, topology independence, and arbitrary consistency
EnsemBlue [17]	integrates consumer electronics in data management systems for "home"	UIA [7]	personal names and optimistic name resolution
EYO [25]	device transparency by full metadata replication in every device	ZZFS [13]	uses a low-power network interface to inform new data placement policies

Table 1: Existing systems along with their primary focus. Our purpose is to demonstrate the lack of focus on security.

University and corporate networks are administered by technical staff. Sharing between users and devices on a home network must often be administered by non-technical users [12]. Previous work used semantic descriptions of data to create high-level replication policies and communication schedules. For example, Perspective's *views* [23], Cimbiosys's *filters* [19], and Anzere's *predicates* [22] allow placement of data aggregations to be specified in terms of predicates across data tags. We contend that access control should similarly be amenable to specification at a high level.

To summarize, interactions between users and devices on a home sharing network are not qualitatively different from users and devices on a more general-purpose network. Sharing and trust relationships are often asymmetric (the fact that Alice shares personal information with Bob does not imply reciprocation) and can take a variety of forms. Therefore, the need for access control and security protocols is prominent in these environments as well.

The rest of this paper develops the thesis that access control and information leakage are important issues in home environments. We argue that access control and strong cryptographic mechanisms must be integrated directly into (rather than layered on top of) the replication protocol to avoid information leakage. Such a system could provide the flexible sharing and strong privacy guarantees needed by users.

2 But...What About the Cloud?

Use of cloud services has become so dominant, both in reality and in mindshare, that it is almost obligatory for papers advocating other approaches to demonstrate why the cloud does not solve all of our problems.

There are several parts to this answer. First, if by a "cloud" we are referring to a large cluster of reliable data or application servers in a data center, we are inherently discussing a non-local solution. This type of storage works extremely well for multimedia streaming and archival storage, but the latency gap between local and non-local communication will never close, making clouds less appropriate for collaborative activities where consistency is an issue (though cloud latencies might suffice for interactive activities, such as editing documents in Google Docs). This is even more true if we consider the many cases where device connectivity is still either poor (in a relative sense), or even nonexistent.

Second, cloud storage is ostensibly highly available but observed outages are frequent [20, 24, 5, 21]. Failure recovery can be challenging in these systems [9], leading to serious incidents of data loss.

Finally, clouds are usually owned and administered by other entities. Not only does this entail obvious privacy challenges, but privacy can even be at odds with cloud provider business models, which often treat user behavior and access characteristics as their product.

However, whether or not a cloud is used is actually irrelevant to the mechanisms and policies discussed in this paper. We feel they are more easily and profitably implemented in local systems, but cloud interfaces and policies are malleable and will probably eventually accommodate any necessary changes.

3 An Approach, and Problems Therein

The majority of work in this area concentrates on data placement, but without access control or authorization restrictions. Any data in "the system" is allowed to migrate anywhere. The assumption is that security issues (access control, authentication, confidentiality) are orthogonal to issues of data replication, and may be handled through existing mechanisms at higher levels. Once users are authenticated onto participating devices, they essentially have access to all system data.

One exception is the extension of Cimbiosys to enforce policies disseminated as ordinary objects containing SecPal access rules [28]. This approach may be problematic, as Cimbiosys does not enforce eventual consistency or multi-item coherence. More importantly, while SecPal statements are signed, data is unencrypted (information leakage), the system assumes a single trusted authority, and the underlying system requires communication through a tree-like hierarchy in order to meet stated efficiency goals.

3.1 Role-Based Security

The previous sections make clear the conflicting requirements for a security architecture in a home sharing system: 1) it should be flexible and functional, and 2) it should be easy to manage for a non-technical user.

However, providing access control in home environments is a hard problem because data sharing characteristics are affected by many issues, including human relationships, content, and system design. Users want access control; however, their lack of technical knowledge often leads to misalignment between actual computer security, and what users think is computer security [12]. Another difficulty arises because ownership in common environments is not always well defined [10].

We believe that an access control scheme for home environments should give the ability to users to organize their collaborators into dynamic groups, much like online social networks organize users' contacts. Users should be able to manage these definitions easily. Moreover, the candidate scheme should be flexible and support fine-grained access rights for all of its users and their objects. Several recent user studies of access control in home environments [4, 10, 11, 12] advocate similar approaches. However, the majority of this work assumes simpler sharing patterns and constraints than we explore here.

We advocate augmenting object metadata to include application-specific semantic information as label-value pairs, which can then be used to inform both the replication protocol and the security architecture.

The overriding goal of our system is to preserve users' privacy. At a high level a role is similar to a Google+ circle, or a Facebook list. At a lower level, a role would be defined as a set of cryptographic secrets, together with boolean *role predicates* over object labels. Any device of a specific role can view and modify those objects that meet the role's predicate.

For example, Alice, Bob, and Charlie might be students who have decided to keep notes collaboratively for a class they both take. Alice acts as "role master", creating role `Alice.Notetakers` with role predicate and access rights `''class==cmsc818 and type==notes''`. She then assigns this role to the devices of all three students. After each class, Alice's careful notes are propagated to Bob's and Charlie's devices, where they make their own additions. The final versions eventually propagate back to Alice.

3.2 Access Control Issues

Role predicates can be used to completely describe data that may be seen or modified through a given role. Any data access outside the realm of these predicates we define as information leakage.

Information leakage is extremely problematic in these environments because of the combination of two desirable properties of replication protocols. First, *partial replication* means that not every replica (or device) needs to have a replica of every piece of data. This seems an obvious requirement. After all, an MP3 player does not need to have copies of movies or financial records.

Replication protocols for home environments also generally support *topology independence*, meaning that any device can exchange replication data or metadata with any other device. Metadata does not have to be explicitly routed anywhere, it eventually gets copied everywhere through pairwise exchanges. Partner choice in the pairwise exchanges could be random, or chosen to prefer high-bandwidth or highly-available connections. Topology independence is also important in allowing the home environment to be dynamic. No device need know the complete system membership, and changes in system membership do not require global consensus.

However, partial replication combined with topology independence implies that a device might be given metadata for data that it has no right to access. In terms of roles, a device might be given role-inappropriate data during pairwise exchange.

This situation is at the heart of our contention that access control is currently implemented at the wrong level for these systems. For example, Bob and Charlie are collaborating on a two-person project in `cmsc818`. Bob creates role `Bob.project`, and assigns this role to both his and Charlie's devices. Assume Bob works on the project at a coffee shop whose WiFi is down, at a table with Alice. His project updates can be communicated through local network connections to Alice's device, where they will later be transmitted to Charlie's device at the apartment Alice and Charlie share.

No clear-text information about the project should leak to Alice. However, access control implemented above the level of the replication protocol has no way to control, or even detect, this information leakage. Access control implemented in concert with the replication protocol allows role-inappropriate information to be safely encrypted.

What's In a Role?

The simple view of role-based access control is that members should only have access to data viewable through their roles. Users outside a role are considered eavesdroppers, and the system should not leak them information. Whether or not this is possible is a function of the specific role implementation, and how the data is stored.

Part of the difficulty arises because we cannot assume that users, especially non-technical users, will come up with roles that are completely orthogonal, i.e., do not share any data. Furthermore, overlapping roles might be unavoidable. For example, it may be infeasible to determine a priori whether a role that allows access to financial records overlaps with a role concerned with vacation planning. We therefore contend that any flexible design must support overlapping roles, and not make strong assumptions about the relationships between user predicates.

A Replica With More Than One Hat

Just as a single device can be used for more than one purpose, a single replica might wear more than a single hat (role). However, a replica with multiple roles introduces ambiguities in what types of sharing is allowable, and what types of sharing are really information leakage.

For example, let us now assume that Charlie decides that the project documents should include one of the lecture notes: `may10.notes`. There are now two distinct roles, *Notetakers*, or *N*, and *Project*, or *P*. We refer to *N*'s role predicate as N_{pred} . For the purposes of this example, $N_{pred} \cap P_{pred} = \{\texttt{may10.notes}\}$, which we name O_{leak}. We assume the existence of an update mechanism, such as an anti-entropy based [26] protocol that permits replicas of the same role to exchange update/invalidation information.

If Charlie creates an update, x_i, to `may10.notes` while working on the project, the semantics of applying x_i at one of Bob's machines, *d*, are not immediately clear. Applying this update to *d*'s local copy of the object is correct from the perspective of role *P*, but `may10.notes` is also covered by *N*. If these modifications eventually propagate to Alice's devices, which have the *N* role though not *P*, information leakage has occurred.

This could occur as follows. One common way to encapsulate object updates is to represent them as whole-object overwrites. If so, a subsequent modification of one byte of `may10.notes` by *d* in role *N* would result in a new role *N* update that would explicitly include data of x_i, i.e. data from an update from a different role.

Alternatively, updates are often communicated as *deltas*, i.e. insertions of bytes at a specific offset from the beginning of the object. If so, imagine that x_i was an insertion, meaning that the object size changes. A new append to *x*, even in role *N*, would be at a different object offset than if x_i had not been applied. The new update, when applied at another device of role *N*, might even be an insertion at an illegal offset.

Most replication systems support *eventual consistency*, a very weak form of consistency that nonetheless requires all replicas of an object to eventually agree on the object's final state. The natural extension of this property to a system with roles is to define *eventual role consistency*, which would require replicas playing a given role to eventually agree on the final state of all objects in that role's predicate. Stated another way, all objects in a role's predicate must be *role-consistent* with respect to that role. In the example above, *d*'s copy of `may10.notes` is role-consistent with respect to *P*, but not with respect to *N*.

4 Another Approach, With Solutions

This section describe potential solutions to the issues raised in the previous section. We can address the multiple-hats information leakage problem by stating that there is, in fact, no problem. An *object-centric* definition of consistency might allow any updates to object *x* to be seen at any device whose role predicate(s) include *x*. This approach is amenable to simple reasoning and implementation, but clearly does not rise to the level of non-interference [14], which states that low-security results are unaffected by high-security data. In our context, we would restate this to say that data from one role should not affect the data accesses of another role.

A more general solution is to directly support the *role-centric* definition of consistency we implicitly assumed in the previous section. No data from an update of one role should be accessible by devices that do not play that role. We can meet this goal by maintaining per-role versions of a single object in any replica that plays multiple roles. Received updates from a given role are only applied to the version of the object specific to that role.

Local reads and writes are more problematic. A local write could be applied to all local versions of an object, but a local read must either apply to a single version, or return multiple versions to the application. While the multiple versions approach has been often used to accommodate conflicting updates [6, 23], these semantics seem inappropriate when distinct versions have different security characteristics.

Instead, we will require application reads to at least implicitly specify a context. This could be an additional read parameter, session definitions that aggregate multiple accesses into a single context, hints provided by a provenance-tracking subsystem [15], or even heuristics informed by locality and history. Writes could be similarly categorized.

This is somewhat analogous to the notion of abstract files in quFiles [27]. There, the version of a file read is dependent on the device characteristics. Here, the version is dependent on access characteristics implied by the device's role(s).

How Do We Route That?

One-way pairwise information exchange is usually optimized by the source sending only data not seen at the destination. This data is often summarized through the use of version vectors or vector timestamps. However, preventing role-centric information leakage requires that if metadata for role *P* is given to a device that is not of that role, all data must be encrypted or randomized. This even includes object ID's of a metadata packet. If a device *x* learns that device *y* just created a new modification for an object in role *P*, it has potentially learned something of value.

We prevent this by requiring all data, except a randomized GUID, be encrypted on non-role devices. The source of unidirectional pairwise exchange learns about data already held by the destination through an *accumulator*, which we define here as a general data structure that can be queried for specific items, but not enumerated. The source then sends only those updates to the destination that do not match the destination's accumulator. The source learns nothing of the destination, other than it had not seen the updates in question. Neither the source nor the destination need even learn the role(s) of the other.

Accumulators could be represented as bloom filters [3] or cryptographic accumulators [2]. Cryptographic accumulators are very space-efficient, but might be computationally expensive. Bloom filters might be a good compromise, though their space requirements scale with the number of keys for a given probability of false positives. Hence, some means of pruning the keys in bloom filters would be needed.

Flexibility

Consider the protocol pitfalls avoided with the approaches outlined above. Naively applying encryption without integration into the replication protocol would have broken the protocols. Implementing access control above the replication layer would have allowed information to leak among roles. Allowing communication only between like-role devices would have broken topology independence. The sum total of our approaches results in an efficient and flexible system.

5 Conclusions

A "home" sharing environment is as complex as any general-purpose network. Replication and security protocols must therefore handle this complexity while meeting goals of completeness (eventual consistency) and security.

This paper has outlined several problems inherent to such protocols, and sketched a solution based on integrating the security architecture with the replication protocol, using accumulators to communicate encrypted meta-information, and using version forking to prevent information leakage between roles.

Acknowledgments

We thank our shepherd, Steve Schlosser, for his advice and guidance in producing the final version of this paper.

References

[1] N. M. Belaramani, M. Dahlin, L. Gao, A. Nayate, A. Venkataramani, P. Yalagandula, and J. Zheng. PRACTI replication. In *NSDI*. USENIX, 2006.

[2] Benaloh and de Mare. One-way accumulators: A decentralized alternative to digital signatures. In *EUROCRYPT: Advances in Cryptology: Proceedings of EUROCRYPT*, 1993.

[3] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the Association for Computing Machinery*, 13(7):422–426, 1970.

[4] A. B. Brush, B. Lee, R. Mahajan, S. Agarwal, S. Saroiu, and C. Dixon. Home automation in the wild: Challenges and opportunities. In *CHI '11*. ACM, 2011.

[5] CNET. Widespread google outages rattle users. `http://news.cnet.com/widespread-google-outages-rattle-users`.

[6] 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 *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, SOSP '07, pages 205–220, New York, NY, USA, 2007. ACM.

[7] B. Ford, J. Strauss, C. Lesniewski-Laas, S. Rhea, F. Kaashoek, and R. Morris. Persistent personal names for globally connected mobile devices. In *OSDI*, Seattle, Washington, Nov. 2006.

[8] R. Geambasu, M. Balazinska, S. D. Gribble, and H. M. Levy. Homeviews: peer-to-peer middleware for personal data sharing applications. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, SIGMOD '07, 2007.

[9] H. S. Gunawi, T. Do, P. Joshi, P. Alvaro, J. M. Hellerstein, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, K. Sen, and D. Borthakur. Fate and destini: a framework for cloud recovery testing. In *NSDI*, 2011.

[10] M. T. Hong Zhang. Mine, yours and ours: Using shared folders in personal information management. *Personal Information Management (PIM)*, 2012.

[11] T. H.-J. Kim, L. Bauer, J. Newsome, A. Perrig, and J. Walker. Challenges in access right assignment for secure home networks. In *HotSec*, Aug. 2010.

[12] M. L. Mazurek, J. P. Arsenault, J. Bresee, N. Gupta, I. Ion, C. Johns, D. Lee, Y. Liang, J. Olsen, B. Salmon, R. Shay, K. Vaniea, L. Bauer, L. F. Cranor, G. R. Ganger, and M. K. Reiter. Access control for home data sharing: Attitudes, needs and practices. In *CHI '10*, pages 645–654, 2010.

[13] M. L. Mazurek, E. Thereska, D. Gunawardena, R.Harper, and J. Scott. ZZFS: A hybrid device and cloud file system for spontaneus users. In *USENIX File and Storage Technologies*, 2012.

[14] J. Mclean. Security models and information flow. In *In Proc. IEEE Symposium on Security and Privacy*, pages 180–187. IEEE Computer Society Press, 1990.

[15] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. I. Seltzer. Provenance-aware storage systems. In *USENIX Annual*. USENIX, 2006.

[16] E. B. Nightingale and J. Flinn. Energy-efficiency and storage flexibility in the blue file system. In *OSDI*, pages 363–378, 2004.

[17] D. Peek and J. Flinn. Ensemblue: Integrating distributed storage and consumer electronics. In *OSDI*, pages 219–232, 2006.

[18] A. Post, J. Navarro, P. Kuznetsov, and P. Druschel. Autonomous storage management for personal devices with podbase. In *USENIX*. USENIX, 2011.

[19] V. Ramasubramanian, T. L. Rodeheffer, D. B. Terry, M. Walraed-Sullivan, T. Wobber, C. C. Marshall, and A. Vahdat. Cimbiosys: A platform for content-based partial replication. In J. Rexford and E. G. Sirer, editors, *NSDI*, pages 261–276. USENIX Association, 2009.

[20] T. Register. Microsoft: Office 365 outages 'will' happen. `http://www.theregister.co.uk/2011/06/30/microsoft_cloud_uptime/`.

[21] T. Register. Microsoft's azure cloud down and out for 8 hours. `http://www.theregister.co.uk/2012/02/29/windows_azure_outage/`.

[22] O. Riva, Q. Yin, D. Juric, E. Ucan, and T. Roscoe. Policy expressivity in the anzere personal cloud. In *Proceedings of the 2nd ACM Symposium on Cloud Computing*, SOCC '11, pages 14:1–14:14, New York, NY, USA, 2011. ACM.

[23] B. Salmon, S. W. Schlosser, L. F. Cranor, and G. R. Ganger. Perspective: semantic data management for the home. In *Proccedings of the 7th conference on File and storage technologies*, pages 167–182, Berkeley, CA, USA, 2009. USENIX Association.

[24] Slashdot. Major outage at the amazon web services. `http://slashdot.org/story/11/04/21/1515238/major-outage-at-the-amazon-web-services`.

[25] J. Strauss, J. M. Paluska, C. Lesniewski-Laas, B. Ford, R. Morris, and F. Kaashoek. Eyo: device-transparent personal storage. In *Proceedings of the 2011 USENIX conference on USENIX annual technical conference*, USENIXATC'11, pages 35–35, Berkeley, CA, USA, 2011. USENIX Association.

[26] D. B. Terry, M. M. Theimer, K. Petersen, A. J. Demers, M. J. Spreitzer, and C. H. Hauser. Managing update conflicts in bayou, a weakly connected replicated storage system. *SIGOPS Oper. Syst. Rev.*, 1995.

[27] K. Veeraraghavan, J. Flinn, E. B. Nightingale, and B. Noble. qufiles: the right file at the right time. In *FAST*, 2010.

[28] T. Wobber, T. L. Rodeheffer, and D. B. Terry. Policy-based access control for weakly consistent replication. In *Proceedings of the 5th European conference on Computer systems*, EuroSys '10, pages 293–306, New York, NY, USA, 2010. ACM.