TAMIAS: a distributed storage built on privacy and identity

Jean Lorchat, Cristel Pelsser, Randy Bush, Keiichi Shima, Helene Schlesinger
Internet Initiative Japan, 1-105 Kanda Jimbocho, Tokyo 101-0051, Japan
e-mail: jean@iijlab.net, cristel@iijlab.net, randy@iijlab.net, keiichi@iijlab.net, helene@iijlab.net

Leif Johansson
SUNET, Tulegatan 11 2tr, SE-113 53 Stockholm, Sweden
e-mail: leif@sunet.se

Paper type
Research paper.

Abstract
In this paper we present Tamias, a new distributed storage system. Tamias has identity and privacy at its core and builds upon it to bring fine-grained sharing features, delegation and revocation. It is can be used upon any low-level distributed storage that has full encryption outside the client. An identity is defined by a public-key that is circulated by the user among other users to introduce himself. In such a situation, introduction is an important step, and out-of-band is always going to be the safest bet. However, we also defined several optional in-band introduction mechanisms. Users can publish information about themselves, solicit other users with a self-introduction, and recommend users they trust to a third party. Finally, using public-key cryptography mechanisms, they can establish secure communication channels that allow to share objects safely within the Tamias storage system. Such a storage is a key piece of technology required by anyone who is privacy conscious, wants to make private online backups, or who is generally worried about Cloud-like online systems taking away their personal data.

Keywords
privacy, user identity, user introduction, distributed storage, document sharing

1. Introduction
While consumer electronic goods have become widespread and have started producing tremendous amounts of digital data, most users rely on unsafe data conservation policies. These policies range from the simple, expensive storage of local media (either magnetic, optical or electrical) to online backup tools. So-called *free online solutions* are becoming quite popular. Yet, storage is not free. The free solutions basically trade hosting for the right to do monetize the users' identity and data. Additionally, as these backup tools are marketed with buzz-words such as “Cloud”, people falsely believe that nothing can happen to their important documents, archives, and records. Currently, most of the available solutions for online storage rely on a centralized design operated for profit by a single party. In this situation, as long as the architecture is not open and well-inspected, any security promise is wishful thinking.

For this reason we believe that it is time for people to reclaim their private data and store them in their very own computers or as encrypted blocks using open standards so that they feel safe dealing with online hosting. For this reason we have designed and developed Tamias, an open source distributed storage system that is built on respect for the users' privacy by allowing fine-grained sharing. And even though it is distributed, it still offers fairly strong guarantees that the users' content is safe from loss and spying.

We wanted to borrow a well secured storage layer upon which to conduct our research into identity, association, trust, sharing, etc. Among the few open architectures that allow secure online distributed storage, we chose an existing low-level store (Tahoe-LAFS project, 2007) to model the underlying storage layer. However, the Tamias architecture is not bound to Tahoe-LAFS and can accommodate any other equivalent low-level storage. Crucially, Tahoe-LAFS provides a distributed dynamic peer-to-peer network, with capability-based access control (Sandhu & Samarati, 1994) and full encryption of the data outside the client device.

These features offer a sound architecture upon which to build the Tamias system. However, the points that make the system unique also raise a few questions: how to manage capability information for many objects, how to

share efficiently with fine-grained control. These questions emphasize the need for an identity. Using identity, we can express trust relationships, sharing authorization, storage objects with restricted access, and messaging. But we must be able to introduce identity in a secure way.

1.1 Related work

In the past decade, people gradually became aware of the privacy issues associated to our own online existence. As they register to more and more web services, they leave traces of personal information all around. Afterwards, there is no mechanism to reclaim such information. This led to the creation of many tentative solutions for personal information protection (Mydex, 2008; personal.com, 2011; Privowny, 2011), sometimes including other kind of data (Project Danube, 2010; Pidder, 2007) such as messages and files. Establishing the identity of a corresponding party became an important topic in other areas of the Internet architecture. This is the case for web servers, where the chosen solution is based on centralized Certification Authorities (CA). This has regularly proven to be a poor solution (Stevens et al., 2009; McCullagh, 2011; Mikko 2011).

In the field of interdomain routing, the Resource Public Key Infrastructure (RPKI) is also being deployed (Lepinski & Kent, 2012). It is using relative authorities that serve as trust anchor for child certificates. The five regional internet registries act as such trust anchors, but a trust anchor could be located anywhere.

On the other hand, decentralized solutions exist in other places. They are based on the concept of web of trust as defined in the PGP (Zimmerman, 1994) electronic messaging solution. With these solutions, the main problem becomes finding a way to get a safe public-key introduction. In the area of Voice-over-IP, this is tackled by the ZRTP key agreement protocol (Zimmerman et al., 2011). It relies on displaying an authentication string on both ends of the communication and having the parties spell it to each other. A man in the middle cannot impersonate both parties, if they know each other a priori.

In this paper, we will first introduce the Tamias architecture and the dominant role that identity systems have to play. We will then describe how we managed to build identity services on top of our storage system, and how they enable identity creation (or import), self-introduction, and secure messaging.

2. The Tamias architecture

Current capability-based storage solutions seem to offer no strong sense of identity built into the system. Usually, owning an object is actually equivalent to knowing its capability. This capability can then be passed on, from user to user, effectively spreading unbeknownst to the original creator, and out of the creator's control.

In order to be able to build a fine-grained sharing mechanism that can prevent this uncontrolled spreading, we associate an identity with every user. This identity allows us to establish ownership and to control capability dissemination, and thus ultimately the scope of sharing.

As we try to keep the Tamias architecture as distributed as possible, we wish to avoid single points of failure. For this reason, the identity scheme is based on public-key cryptography, allowing people to introduce themselves using a public key, and to build a network of acquaintances through exchange of public keys.they enable identity creation (or import), self-introduction, and secure messaging.

2.1 Identity in Tamias

As we have stated, identity is the core value of the Tamias architecture. In the case of Tamias, it takes the form of two public-key cryptography key-pairs. The first key-pair, the **root key-pair**, is used to locate the user's content and is kept private to the user. The second key-pair represents the user's identity as such and is used to perform all public-key cryptography computations for communications with others. It is called the **identity key-pair**. The private key from this key-pair remains the user's secret, and is used for signing and decryption. On the other hand, the public key of this pair is shared with other users who use it for encryption and signature validation. To represent their identity to other users, the user circulates the public-key of their identity key-pair.

Upon the user's first connection to Tamias, the client asks the user to define their identity key-pair. This can be done by generating a key-pair (current defaults are RSA algorithm with a key length of 2048 bits) or importing an existing key-pair. The root key-pair is then generated and used to create the user's Tamias root folder. The private key of the root key-pair may also be imported. This allows the user to configure other devices using their same identity, obtaining a consistent view of their storage space spread across multiple devices.

2.2 Fine-Grained Sharing

As explained above, knowing the capability for an object in the storage is enough to gain access. To make privacy with fine-grained sharing possible, we must enforce that only those who received the capability legitimately can make use of it.

Since it is impossible to prevent data from flowing, leaking, or being stolen, we choose to protect the capabilities. Without its capability, data cannot be retrieved, let alone decrypted. Capability protection is done by creating authorization objects. These objects contain information about the sender's identity, the intended recipient's identity and a public-key encrypted version of the actual capabilities. This authorization is then signed by the sender. Subsequently, we extend the storage servers so that they keep a copy of the public-key of the owner with the stored objects. Now that the public-key of the sender is available on the client side (within the authorization) and on the server side (next to the stored object), we avoid need of a public-key distribution infrastructure.

Per usual public-key cryptographic properties, this signed authorization is impossible to hijack without knowledge of either the destination's private key or the owner's private key. The reason is that, whenever a server is asked for a block, it will challenge the requester to prove that his identity matches the target identity recorded in the signed authorization. Of course, the server also compares the key used for signing the authorization with the key used to create the object in first place. There is no need to encrypt the authorization with the destination's public key since the authorization is protected from modifications by the signature of the sender.

Finally, the owner may choose to grant delegation privileges, where the recipient may share the object with others. This permission must be signalled explicitly to the signed authorization, i.e. it is not granted by default. The intermediate recipient may then create a new signed authorization. This new authorization encapsulates the previous one while targeting a new recipient.

2.3 Tamias root

The Tamias root folder is the place where all the personal data and control information of the user are stored. The control information includes a list of public keys describing acquaintances, and a list of granted authorizations. Since this control information is used by the Tamias client to perform all operations, the Tamias root folder should only be accessible by the user, that is, the owner of the control data. It should never be shared, even read-only. This rule must be applied to all control files within the Tamias root folder. We will now detail the different key elements in the Tamias root folder.

2.3.1 Buddies' public-key list

The **buddy list** holds the identities (public keys) of a user's data-friends and allows the formation of user-centred groups.

The public keys to be found in the buddy list are added by the user themselves, based on information they receive via the introduction mechanism (see section 3.2.2), or by any external out of band means (e.g. email, instant messaging, hard-copy). When adding a public key, the user can associate a meaningful nickname to it. A public key is never added to the list without the user's consent.

Also, in order to provide easier sharing operations, peer identities may be grouped on the client side. These user-centred groups have no meaning outside the user's client. Indeed, the client is responsible for repeating each operation for each group member. This means that when the user chooses to share a file with a group, the Tamias client actually splits this request in as many per-user sharing operations as there are buddies in the group. Among other advantages, this allows the user to revoke rights granted to one member of a group.

2.3.2 Authorizations' list

To enforce strong privacy, the Tamias client keeps track of all active authorizations that the user has signed. These authorizations are kept in a specific location within the Tamias root folder. They can be browsed leisurely to be revoked at any time.

2.3.3 User content list

The **user content list** in the Tamias root folder is the only place that the user is free to directly manipulate. It is the central point for the user's personal files as handled by Tamias. All of the user's files are listed here, and are stored with the public-key of the owner. This enforces the use of signed authorizations for all access.

The users may build this part of the storage in any way they wish. The actual files are stored somewhere in the underlying storage, and the users build a structure that keeps data organized in the way they see fit. It could be a tree mimicking a file system, a big bag of files like a flat namespace, or anything that can be built with a directed graph.

3. Identity Based Services

As the previous section shows, the user identity is the key element that allows us to establish fine-grained control over sharing. In addition, it is possible to take advantage of the user's identity to build other services.

Since we are using a public-key identity scheme, having a good identity introduction is a decisive step. Of course, out-of-band introduction is always safest, or at least we can blame failure on some other process. On the other hand, in-band mechanisms are much easier to use. This is why we decided to provide an in-band mechanism, as an optional introduction scheme for Tamias users.

In the next sections, we first detail how users can create messaging boxes for each other to exchange sharing notifications and authorizations, and, then, how users can self-introduce themselves to potential buddies.

3.1 The Conversation Box

A user creates a **conversation box** to receive messages from a single source. Sharing notifications arrive in one of the user's conversation boxes. Thus, a user creates a conversation box for every other user that may share content with him. The capability granting access to this storage object is recorded in the buddy-list. The access granted to the conversation box is append-only. Both the recipient and the sender may read the content of the box, however the sharer always writes at the end of the box. Only the recipient may remove content.

Such an append-only capability must be treated with care, because it allows a file belonging to one user to grow arbitrarily. Conversation boxes should thus be limited in size, preventing additional notifications whenever full. However, in this initial trial, we are not too worried about this because the conversation box was granted by the user to one of their buddies because the user trusts the buddy.

The conversation box is made of statements as proposed by the W3C in the Resource Description Framework (RDF) for the semantic web. These statements are of the form "*subject verb object*". In our application, the *subject* is the public key of the user making the statement. The *verb* specifies the action, like "*sharing*" or "*removed*" and, the *object* contains the signed authorization for the file that is acted upon. For example, if a user wishes to signal the availability of a shared file, it will store a statement of the form "u_k *shares* c_f" where u_k is the user's public key and c_f is a signed authorization generated by user *u* for the owner of the conversation box to access file *f*.

3.2 In-band Self-Introduction

The in-band self-introduction mechanism uses a phonebook and relies on user inboxes. The phonebook is a mechanism that enables to publish one's identity, inbox and, eventually, selected personal information. A user can get in touch with other users via their inbox, once it discovered their identity and inbox.

3.2.1 The Phonebook

The phonebook is a repository of statements users make about themselves. There is one such repository per storage domain, currently a single Tahoe/LAFS cloud. The purpose of the repository is for a user to discover other users' public keys. It serves as user introduction mechanism.

As in the conversation box, statements are of the form "*subject verb object*". The *subject* is the public key of the user, representing their identity. The *verb* and *object* combination defines a property of the user which they agree to publish in the repository. Registering one's information in the phonebook is optional. Without publishing in the repository, a user needs to find an alternative way (out-of-band) to communicate their public key to other users who they want to befriend and share content with.

The phonebook is an append-only file (see section 3.1). The append capability is well-known. It is advertised through the Tahoe-LAFS introducer or, alternatively, the DNS could be used. This is where we made a tradeoff between usability and security, because anyone could publish forged information in the phonebook. For this reason, using the repository is optional, as cautious people will probably avoid trusting users they know only from the phonebook.

Even though the phonebook is optional, there is a concern that it might be browsed at leisure by users or programs wanting to harvest personal information however scarce. It is possible to resort to centralized solutions in this very specific context, especially when selling Tamias hosting service for example. Or it might be possible to introduce a tightly controlled distributed phonebook search agent, acting as a proxy between the request and the phonebook which would then be hidden.

3.2.2 The Inbox

The user's inbox is the mechanism to get in touch with other users within Tamias. Since user introduction is eventually up to the user's judgment, the user inbox should not be taken as more than a convenient means to provide a channel for out-of-band public key exchange mechanism. The introduction via public inbox is the result of a trade-off between risk and ease-of-use. For this reason, the user inbox is entirely optional. It is up to the user to decide whether they want to create one. If they do, they may publish the capability for this box in the phonebook for everyone to see. The capability can then be used by anyone to write statements to this user. It is an append-only file with a size limitation, and serves to gather buddy requests.

Buddy requests are statements that other users make about themselves and publish into another user's inbox. Such statements are basically a befriending request, and users can handle it in the same way as in most existing social networking services: by accepting or rejecting the request immediately, or by looking in the phonebook for the self-introduced person. Accepting the request leads to the friend's public key being added in the user's buddy list. Then, the next step in the trust building model of Tamias is to exchange conversation box information, which can also happen via the user inbox.

That is, if a user *A* wants another user *B* to share content with them, user *A* will create a conversation box for user *B* in *A*'s own storage space and publish a statement about the capability of said box into user *B*'s inbox. The advantage of this is that users choose to opt-in to other users feeds. A user cannot share files with someone who did not opt to receive them.

The write capability of the user's inbox is public, and hence requires both a quota to avoid starvation, and a per-user rate-limit to avoid flooding. However, an attacker could generate a lot of identities to overflow the inbox. As much as the inbox is useful, it is still an optional mechanism. Nevertheless, it is possible to imagine spam control additions that would work as a Tamias application to mitigate the flooding problem. Another possibility involves third-party introduction by a trusted peer (see Figure 1). In this case, a peer *Alice* trusted by *Bob* would serve as the introduction channel for the identity *Chris* that *Alice* already trusts. *Alice* would post statements about this friend of his to the conversation box that he uses to share with *Bob*. The statement thus looks like "*Alice trusts Chris*". Of course in the end, the choice of adding *Chris* to their buddy-list is entirely up to *Bob*. In addition, once a conversation box has been created, users have no reason to go back to using the public inbox, so it should only be used by people who are not yet trusted.

Figure 1: Third-party introduction through mutual friend

The hierarchy of the three statement boxes is summarized in Table 1.

Box name	Writer	Reader	Purpose
Conversation Box	One friend	Owner	Sharing discovery
Phonebook	All users	All users	User discovery
User Inbox	All users	Owner	User introduction

Table 1: Statement boxes, from global to specific

3.3 Illustrative Example

As an example, let's study what happens when Alice wants to get in touch with Bob to share a document. She will start by looking for all Bobs in the *Phonebook*. This will probably yield several answers that she will have to narrow down. To do that, she needs to use further identifying information she knows about Bob and look for it in the *Phonebook*. If she succeeds, she gets to know Bob's public key and Bob's *User Inbox*. She will then contact Bob at this inbox by writing an introductory statement. This statement includes her public-key. She then creates a *Conversation Box* to receive statements from Bob and publishes its capability to Bob's *User*

Inbox. From then on, all exchanges from Bob to Alice will use this *Conversation Box*. If Bob decides that he trusts Alice, he creates a corresponding *Conversation Box* to receive statements from Alice. He then publishes its capability to the *Conversation Box* that Alice created earlier. Alice can now write directly to Bob using this new *Conversation Box*. This is summarized in Figure 2.

4. Conclusion

We introduced Tamias, a distributed private storage system which builds identity, trust, and sharing on top of Tahoe-LAFS, providing file ownership, management of fine-grained sharing, and prevention of unauthorized access with hijacked capabilities. It then becomes possible to build applications on top of Tamias that take advantage of all the security and identity features: secure sharing of secret data, safe disclosure of personal information, and so on.

Tamias is currently under development and available from our project page at `http://tamias.iijlab.net`.

Figure 2: Example of How Alice finds out about Bob

References

Danube Project (2010). Identity and communication for political and social innovation.
URL: http://projectdanube.org

Lepinski, M. & Kent, S. (2012), RFC6480, an infrastructure to support secure internet routing, Internet Engineering Task Force.

McCullagh, D. (2011), Comodo hack may reshape browser security. CINet.
URL: http://news.cnet.com/8301-31921_3-20050255-281.html

Mikko (2011), Diginotar hacked by black.spook and iranian hackers. F-secure weblog.
URL: http://www.f-secure.com/weblog/archives/00002228.html

Mydex (2008), Mydex gives you back control over your personal data.
URL: http://mydex.org

personal.com (2011), a personal network as unique as you.
URL: http://www.personal.com

Pidder (2007), Your secure private social network.
URL: https://www.pidder.com

Privowny (2011), Own your privacy.
URL: http://www.privowny.com

Sandhu, R. & Samarati, P. (1994), Access control: principle and practice, in 'IEEE Communications magazine'.

Stevens, M., Sotirov, A., Appelbaum, J., Lenstra, A., Molnar, D., Osvik, D. A. & de Weger, B. (2009), Short chosen-prefix collisions for MD5 and the creation of a rogue CA certificate, in 'Advances in Cryptology'.

Tahoe-LAFS project (2007), The least authority file system.
URL: https://tahoe-lafs.org

Zimmerman, P. (1994), PGP user's guide.
URL: ftp://ftp.pgpi.org/pub/pgp/2.x/doc/pgpdoc1.txt

Zimmerman, P., Johnston, A. & Callas, J. (2011), Rfc6189,
ZRTP: Media path key agreement for unicast secure RTP, Internet Engineering Task Force.

Biographies

Jean Lorchat is a researcher at Internet Initiative Japan, Japan's first historical commercial ISP. His research interests include wireless networking, mobile adhoc networks, layer 2 and layer 3 mobility, and more recently secure distributed storage systems. Jean got his PhD in Computer Science from Universite de Strasbourg, France in 2005 and moved to Keio University as an assistant professor until 2008 when he joined IIJ.

Cristel Pelsser received her Master degree in computer science from the FUNDP in Belgium in 2001. She then obtained her PhD in applied sciences from the UCL, in Belgium, in 2006. From 2007 to 2009, she held a postdoctorate position at NTT Network Service Systems Laboratories in Japan. She is now a researcher at Internet Initiative Japan (IIJ). Her current research interests are in Internet routing and privacy, identity in distributed storage solutions.

Randy Bush is a Research Fellow and Network Operator at Internet Initiative Japan, Japan's first commercial ISP. He specializes in network measurement especially routing, network security, routing protocols, and IPv6 deployment. Randy has been in computing for 45 years, and has a few decades of Internet operations experience. He was the engineering founder of Verio, which is now NTT/Verio. He has been heavily involved in transferring Internet technologies to developing economies for over 20 years.

Keiichi Shima is a senior researcher at Research Laboratory of IIJ Innovation Institute (as well as Internet Initiative Japan), working on IP mobility technology, datacenter network/storage resource migration technology, wireless Internet technology. He achieved a Ph.D. degree on information science, at Nara Institute of Science and Technology in March 2009.

Helene Schlesinger was the lead developer on the Tamias prototype from September 2011 to April 2012. For this purpose, she has been using python extensively, including the Twisted framework, Foolscap remote objects, and much more.

Leif Johansson is a system specialist at NORDUNet A/S working on digital identity and federation for SUNET (the Swedish University Network). Leif is a member of the technical operations team for the Swedish Academic IDentity Federation (SWAMID) and a member of the technical reference group of the Swedish University Network (SUNET). He is also part of the operations team for the SWAMI identity federation, responsible for the technical operation of SAML federation infrastructure. Currently Leif is working on the Swedish national federation testbed.