Workspace: A Data-Oriented, Decentralized Collaboration Web App

Tianyuan Yu
UCLA
tianyuan@cs.ucla.edu

Xinyu Ma
UCLA
xinyu.ma@cs.ucla.edu

Varun Patil
UCLA
varunpatil@cs.ucla.edu

Yekta Kocaoğullar
UCLA
ykocaogullar@g.ucla.edu

Lixia Zhang
UCLA
lixia@cs.ucla.edu

ABSTRACT

Multiparty remote collaborations are one of the popular applications on today's Internet. However, the existing collaborative apps are, by and large, hosted on cloud servers owned by a small number of providers. In these apps, end users can only collaborate by connecting to cloud servers. This paper explores an alternative path to building collaborative applications by making use of named, secured Web Objects (SWO). Inspired by the original data-oriented vision of web, we developed a decentralized collaborative application dubbed *Workspace*. Workspace users establish trust relations among each other and secure their data productions. This approach enables users to exchange SWO by data names directly, and to collaborate through intermittent connectivity.

CCS CONCEPTS

• **Information systems → Web applications**; **Internet communications tools**; • **Networks → Network design principles**.

KEYWORDS

Decentralized Web, Local-First Software, Information-Centric Networking

ACM Reference Format:
Tianyuan Yu, Xinyu Ma, Varun Patil, Yekta Kocaoğullar, and Lixia Zhang. 2024. Workspace: A Data-Oriented, Decentralized Collaboration Web App. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The success of clouds has centralized web-based applications onto cloud servers, which are largely offered by a small number of cloud providers. Although end users generate a large percentage of web contents, they do not have direct control over their data, nor can they collaborate directly without connecting to the cloud servers.

Figure 1 depicts a meeting agenda creation scenario with Alice and other meeting organizers sitting in the same room. Alice sets up a shared online document "agenda.xml" and invites her co-organizer Bob to join collaborative editing. Accepting the invitation, Bob has to first connect to the cloud server and login to his cloud account, before he can make changes. In this example, all users are in the same room, their devices, phones, tablets, or laptops, can directly reach each other via WiFi or bluetooth connectivity and authenticate each other by out-of-band trust relation establishment (e.g. scanning a QR code). Yet there is no existing web app implementation that support them to directly exchange document in a cloud independent way.

Conference'17, July 2017, Washington, DC, USA
2024. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

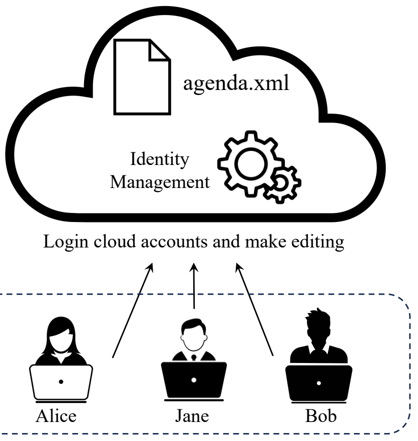


Figure 1: Alice and Bob collaborate on a shared document hosted by clouds

In a companion paper, we defined Secure Web Objects (SWO) [8] as semantically named and cryptographically Secured Web Objects. We believe that SWO opens a new opportunity to revisit the collaborative web application design. SWO enables web developers to write decentralized applications running on user devices to produce, exchange, and consume SWO directly.

An ideal collaboration environment should be a shared cyberspace where people gathered by social relations benefit from joint efforts. The shared cyberspaces naturally form *workspaces* of different groups among people. In each workspace, collaborators work on the same set of documents and can invite others to join the efforts. Everyone can see the content updates made by others. More importantly, people should be able to work without having to connect to centralized servers. Whenever connectivity to others is/becomes available, SWO that carry changes to shared documents shall be exchanged among the collaborators.

To realized this decentralized application model, We sketch a system design that requires the following functions:

- **Semantic user identity:** Identifying users (*i.e.,* collaborators) by application-wide semantic identifiers, which can be authenticated without connecting to cloud.
- **Document security:** Documents are only visible among mutually authenticated users.
- **Update consistency:** Consistently updating the documents for workspace users based on ongoing changes made by others, so that eventually all users share the same view of the workspace.
- **Rendezvous:** enabling users to exchange SWO by names, instead of n-by-n connections, without going through the cloud.
- **Asynchronous communications:** Users are able to asynchronously receive document updates, so that users are not required to be reachable at the time when new SWO are produced.

This paper describes a collaborative text editing application we have built, named *Workspace*, as an demonstrative example of how the SWO concept enables decentralized collaborative application. The rest of the paper is organized as follows: Section 2 explains the how we build Workspace on top of SWO with mutually authenticated, semantic user identifiers Section 3 describes Workspace implementation. We discuss *Workspace* featurs and its need for in-network storage in Section 4, and conclude the work in Section 5.

2 FROM SECURE WEB OBJECTS TO DECENTRALIZED APP

In this section, we describe Workspace design by first giving explaining how Workspace users are semantically identified and mutually authenticated, then we demonstrate how Workspace organizes document data by SWO.

2.1 Security Bootstrapping

Today's Internet is built on TCP/IP's network model, where a network is made of interconnected nodes identified by IP addresses; there is no security relations among the nodes at the IP layer. When E-commerce raised the demand for secure Internet communications in early 90's, the solution was built *on top of TCP/IP* by authenticating servers by their DNS names and then encrypting data sent over TCP connections.

Named Data Networking (NDN) takes on a fundamentally different networking model: a network is made of semantically named entities with various trust relations among each other [9], where these named entities can be devices, servers/services, app instances, or anything that produce and/or consume named data. The names of these entities are decoupled from their specific attachment points to the network in general, they can explore any available connectivities to communicate. To enable this new networking model, each entity needs to obtain its name(s) and security credentials, and to establish trust relations with other entities it interacts with. Workspace is implemented over NDN, thus all users in workspace need to go through a bootstrapping process to obtain the above information to enable secure exchanges of SWO.

Workspace uses hierarchically structured and semantically meaningful names as application and user identifiers. Each workspace instance can be identified by a URI-like name, "`/prefix/meeting-i`", where *prefix* uniquely identifies the context of a workspace instance within its scope, and *meeting-i* identifies a specific workspace instance in the scope[1] To simplify the description, in the following example we use prefix *meetings* as the identifier for a workspace instance, in place of "`/prefix/meeting-i`".

Assuming Alice, who has an email address *alice@example.com*, is joining the workspace *meetings* to edit a shared document "agenda.xml". As shown in Figure 2, during the bootstrap step, Alice is assigned a Workspace identity "`/meetings/alice@example.com`", which concatenates her email with the Workspace context, and creates a self-signed certificate for her identity. Anyone who trusts this self-signed certificate can verify data generated by Alice.

2.2 Mutual Authentication

Workspace users can mutually authenticate each other in two procedures: *origin authentication* verifies the identity of the owner of a certificate, and *endorsement* propagates the result of origin authentication.

To verify the identity of the a owner, one can make use of existing unique identifiers in the Internet. If the existing identity provider offers some resources modifiable by the owner of a identifier (*e.g.,* Tweet, Gist, DNS Records), the verifier (say Bob) can verify the ownership using some method similar to Let's Encrypt ACME [1] or Keybase [2]. For example, using authentication by email, Bob can send a random nonce to Alice via email, ask her to sign it and send back the signature. In this way, Bob both verifies Alice's identifier (her email address) and obtains her public key.

Having authenticated Alice, Bob signs Alice a endorsement certificate, thereby she can present Bob's endorsement when others try authenticating her. Eventually, every user in same workspace will have certificates signed by everyone else, and a fully meshed web of trust is formed in workspace.

Workspace only uses origin identities for initial user security bootstrapping. After receiving an endorsement certificate, a user is bootstrapped into the web of trust and no need to communicate with today's authentication servers, and this web of trust runs completely independent from today's clouds. When there does not exist any trustworthy identity provider, one can use out-of-band channel to transmit the self-signed certificate directly to bootstrap a new user as a fallback.

Although Workspace users mutually authenticate self-signed certificates, this approach differs from the concept of Web-of-Trust [3] with PGP [11] keys, which focuses on endorsing self-signed certificates without unique semantic bindings. In order to collaborate, Workspace users already have out-of-band trust relations on their unique semantic identifiers (*e.g.,* emails, social media accounts), and mutual authentications leverage these pre-existing trust relations to authenticate and endorse self-signed certificates bound to their identifiers.

2.3 Secure Web Objects As Building Blocks

Security bootstrapping enables users to produce SWO that carries document changes. The SWO namespace starts with each user's Workspace identity (discussed later in §2.1) and is followed by a

[1]A locally scoped workspace instance can use a localized prefix; a global scoped instance needs to use DNS names in its prefix to assure global uniqueness.

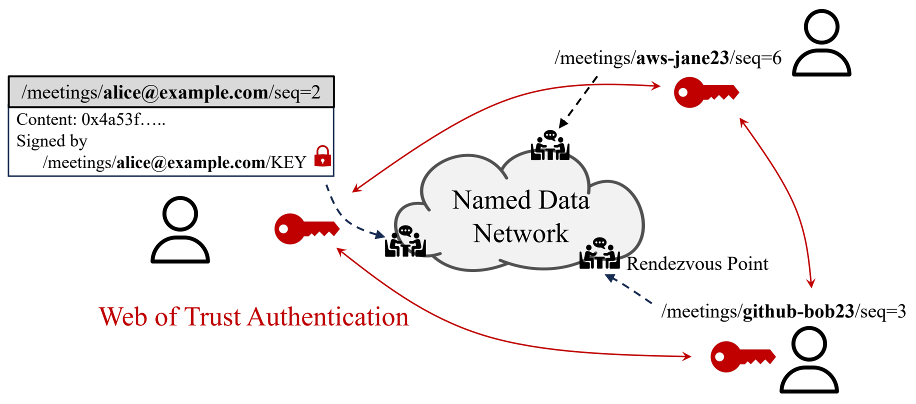

/meetings/alice@example.com/seq=2
Content: 0x4a53f.....
Signed by
/meetings/alice@example.com/KEY
/meetings/aws-jane23/seq=6
Named Data Network
Rendezvous Point
/meetings/github-bob23/seq=3
Web of Trust Authentication

Figure 2: A scenario where three users exchange SWO in the "meetings" workspace. Through peer authentications and the web of trust, Bob and Jane are able to verify Alice's SWO, which includes the latest change she made to the workspace.

sequence number, indicating the number of updates have been made by this user. Figure 2 gives an example of the Workspace naming convention, where Alice's changes are named sequentially under her *meetings* identity prefix.

The *meetings* workspace consists of multiple folders, each can contain multiple files. Workspace ensures that all users in *meetings* eventually see the same agenda by representing this file hierarchy as a collection of shared data structures, referred to as *Doc*. From Workspace's perspective, meeting organizer collaboration is driven by users making changes to the *meetings* Doc. The file creation of "agenda.xml" is a change made to the file index structure, and the text Alice added to the agenda are a series of changes on text structure. By exchanging SWO on Doc changes, all users will get everyone else's updates made to the shared document, therefore hold the same view of the workspace.

Workspace secures SWO by encrypting and signing them with keys, and keys themselves are also named and secured as SWO. When Alice's Workspace instance signs SWO with her private keys, the signature can carry a URI-like key locator point to her certificate SWO. Data content are also encrypted by Alice's key with a key locator pointing to the encryption key SWO.

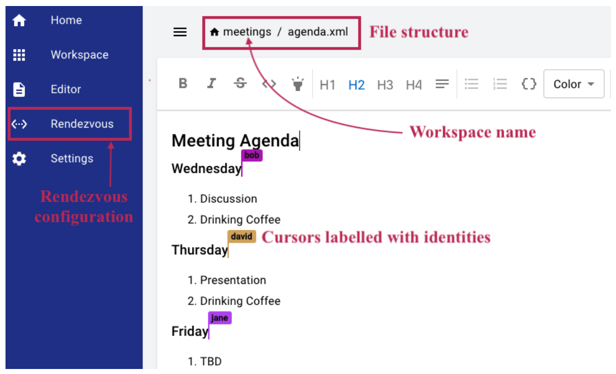
3 IMPLEMENTATION

We have implemented Workspace as web application in TypeScript [2], and Figure 3 shows the in-app screenshot of our implementation. This section explains the implementation details of Workspace, focusing on conflict resolution, SWO synchronization and local data storage.

3.1 Doc Conflict Resolution

As each SWO carrying users update to the shared Doc, Workspace needs to resolve conflicts from concurrent updates. Our implementation automatically resolve conflicts by defining Doc as a collection of Conflict-free Replicated Data Types (CRDT) [5]. We represent folders as CRDT *Maps*, which maintains a mapping between files and CRDT sub-structures. Text files are mapped to CRDT *Texts*,

[2]https://ndn-workspace.web.app/



Meeting Agenda

Wednesday

1. Discussion
2. Drinking Coffee

Thursday

1. Presentation
2. Drinking Coffee

Friday

1. TBD

Figure 3: Alice joins the "meetings" workspace to collaboratively edit the agenda with her colleagues

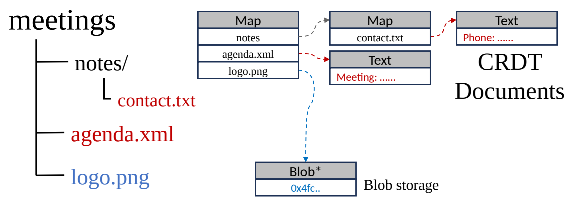


Figure 4: Workspace represents file structure as a document, which is a collection of shared data structures.

allowing real-time collaboration. Other types of files, which do not support real-time collaboration, are transformed into versioned, immutable binary string.

3.2 Synchronizing SWO at Rendezvous

Synchronizing SWO publications in "meetings" namespace requires a rendezvous that enables efficient SWO exchanging among Workspace users by names. Ideally, such rendezvous is a SWO exchange network that forwards Workspace SWO requests and response by names, and anyone who has Workspace content can join.

We have setup a testbed network [7] using a data-centric Internet architecture Named Data Networking (NDN) [10], to serve as the rendezvous that Workspace needs. This data-centric network testbed has 20 nodes across four continents, and is managed by collaborator universities in a decentralized fashion.

When started up, Alice's Workspace instance connects to the geographically closest router in the testbed, and registers her user prefix "`/meetings/alice@example.com`" to the router. NDN Sync [4] protocol ensures keeping all Workspace instances in *meetings* up-to-date regarding the current SWO production state, and instances can subsequently fetch the SWO they desire directly from each other, eliminating a central control node.

A key reason we used NDN testbed for rendezvous is that the testbed purely serves inter-connectivity purpose, and do not understand or parse the underlying SWO. Users or organizations may also choose to run their NDN networks in a decentralized fashion, and optionally connect these nodes to the existing testbed. Traditional clouds, on the other hand, must understand the semantics of the underlying data to be able to route them to the correct clients, and thus need to be centrally controlled and secured.

For situations where reaching the NDN testbed may not be possible, *e.g.,* in an airplane, we also provide the option of using PeerJS to establish direct $n \times n$ connectivity between all Workspace users in a group using WebRTC.

3.3 Local-First Collaboration

The Workspace application is supposed to work both offline and online to the rendezvous. This leads to two requirements: (a) necessary data must be stored locally, and (b) must support collaborations through asynchronous communications.

To provide offline access, we currently use browser's origin private file system (OPFS), which is a file-like storage persistent on the disk and private to the web application. All SWO received are stored in its original bytes, so the security properties (such as signatures) are preserved. For example, if Bob receives an SWO generated by Alice, he can forward this change to Jane, while its original signature by Alice is still valid.

4 DISCUSSION

Using SWO as basic building blocks, Workspace get independence from cloud by enabling user controlling their own data (*i.e.,* SWO). However, SWO storage needs resources, and current implementation of utilizing browser storage has limitation when there is only one active user at the rendezvous point. In this situation, later users that access to the rendezvous point are not aware of the first user's published SWO.

In order to enable fully asynchronous communications, Workspace needs a SWO storage that have following properties:

- *Generality.* A generic storage is not designed for specific application. Multiple applications can share the same storage to improve cost efficiency if needed. An application like Workspace may also migrate to another storage provider easily.
- *Untrustworthiness.* A generic storage is not trusted by the application users. It cannot decrypt data generated by application users, nor can it author any changes by itself.
- *Simplicity.* The only functionality needed is to store and serve SWO. Extra functionalities such as access control is performed by the application, making a generic storage easy to implement and maintain.

Based on the features described above, we believe that a *in-network data storage* should be provided as *network services* by the network operators, instead of an application service of the cloud. As a short term solution, our current implementation used PythonRepo [6] as a prototype of such service in NDN testbed. We identify the generic storage service to be an important component of local-first software and expect to see more work on this topic coming out in future. When such a service is unavailable, a traditional cloud storage (such as Google Cloud Storage and AWS S3) can be used as a fallback.

5 CONCLUSION

Today's web applications run on cloud servers. Users connect to the cloud via secure connections, and the cloud identifies and authenticates users, serves as a rendezvous place for users to meet and a secure castle to host data and run apps.

In this paper we use the development of Workspace as a use case to show that, by using SWO as the basic building block, one can build decentralized web applications by solving three basic problems: unique user identities, decentralized security, and rendezvous. In Workspace, users obtain cloud independent identities and mutually authenticate each other through web of trust; they rendezvous over name-based NDN connectivity to exchange and synchronize SWO publications, which are secured directly by users, under Workspace's namespace. We map the Workspace file structure into a collection of shared data structure, and use CRDT as the solution to achieve eventual consistency among all users sharing the same document. Users can also work on documents without Internet connectivity, and synchronize SWO with others when rendezvous becomes available.

REFERENCES

[1] Josh Aas, Richard Barnes, Benton Case, Zakir Durumeric, Peter Eckersley, Alan Flores-López, J Alex Halderman, Jacob Hoffman-Andrews, James Kasten, Eric Rescorla, et al. 2019. Let's Encrypt: an automated certificate authority to encrypt the entire web. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security.* 2473–2487.
[2] KeyBase contributors. 2024. https://keybase.io/. Accessed: 2024-2-3.
[3] Jennifer Golbeck. 2008. Weaving a web of trust. *Science* 321, 5896 (2008), 1640–1641.
[4] Philipp Moll, Varun Patil, Nishant Sabharwal, and Lixia Zhang. 2021. A brief introduction to state vector sync. *NDN, Technical Report NDN-0073, Revision 2* (2021).
[5] Marc Shapiro, Nuno Preguiça, Carlos Baquero, and Marek Zawirski. 2011. Conflict-free replicated data types. In *Stabilization, Safety, and Security of Distributed Systems: 13th International Symposium, SSS 2011, Grenoble, France, October 10-12, 2011. Proceedings 13.* Springer, 386–400.
[6] NDN Team. 2024. https://github.com/UCLA-IRL/ndn-python-repo. Accessed: 2024-2-3.
[7] The NDN Team. 2024. NDN Testbed. Online at https://named-data.net/ndn-testbed/.
[8] Tianyuan Yu, Xinyu Ma, Varun Patil, Yekta Kocaogullar, Yulong Zhang, Jeff Burke, Dirk Kutscher, and Lixia Zhang. 2024. Secure Web Objects: A Data-Oriented Paradigm. In *Submission to ACM Web Conference 2024.*
[9] Tianyuan Yu, Philipp Moll, Zhiyi Zhang, Alexander Afanasyev, and Lixia Zhang. 2021. Enabling Plug-n-Play in Named Data Networking. In *MILCOM 2021 -*

2021 IEEE Military Communications Conference (MILCOM). IEEE Press. https://doi.org/10.1109/MILCOM52596.2021.9653033

[10] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, KC Claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Beichuan Zhang. 2014. Named data networking. *ACM SIGCOMM Computer Communication Review* 44, 3 (2014), 66–73.

[11] Philip R Zimmermann. 1995. *The official PGP user's guide.* MIT press.