



Ownly

The NDN Workspace

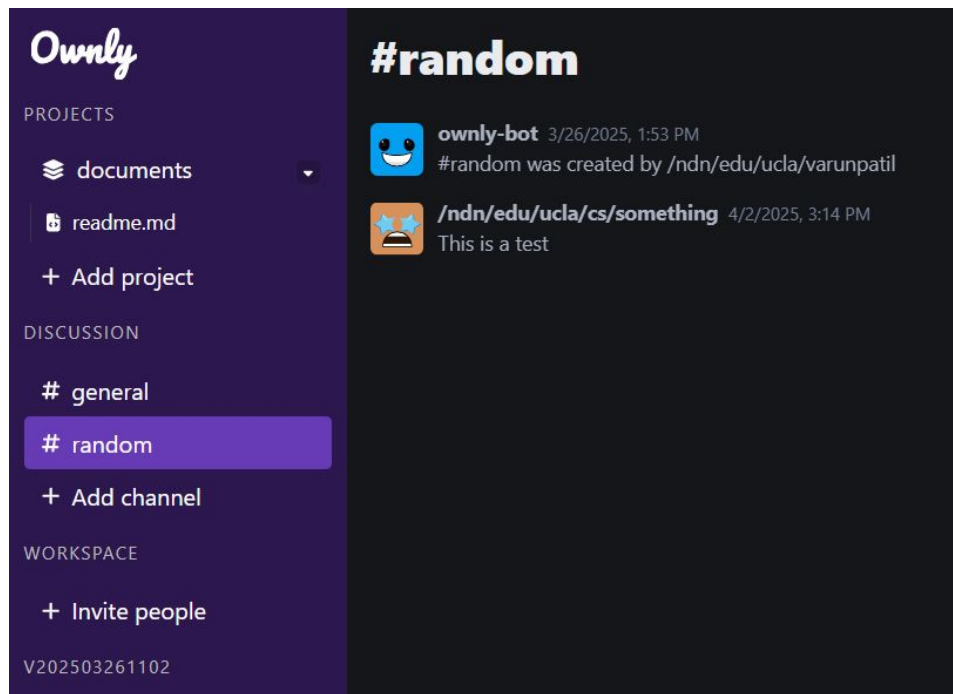


What is Ownly?

- Fully decentralized collaborative workspace
- Built over the Named Data Networking stack
 - Secured with name-based security primitives
 - Communication based on NDN primitives
- End-to-end encryption
- Focus on end-user usability

<https://ownly.work>

What can it do?



```
1 # My New Workspace
2
3 Welcome to your new workspace! 🎉
4
5 You can collaborate on files in projects in real time.
6 Try creating a new document or importing an existing one.
7 Invite people to your workspace to start collaborating!
8 /ndn/edu/ucla/cs/something
9 Live collaboration|...
```

V202503261102

How is Ownly different?

- Decentralized
 - The workspace owner has full control
 - No third party has any control power
 - Runs on permissively licensed open-source infrastructure
- Local-first
 - No central control or storage server required
 - Truly peer-to-peer
- Secure
 - Fine-grained name-based control with trust schema
 - Data is end-to-end encrypted

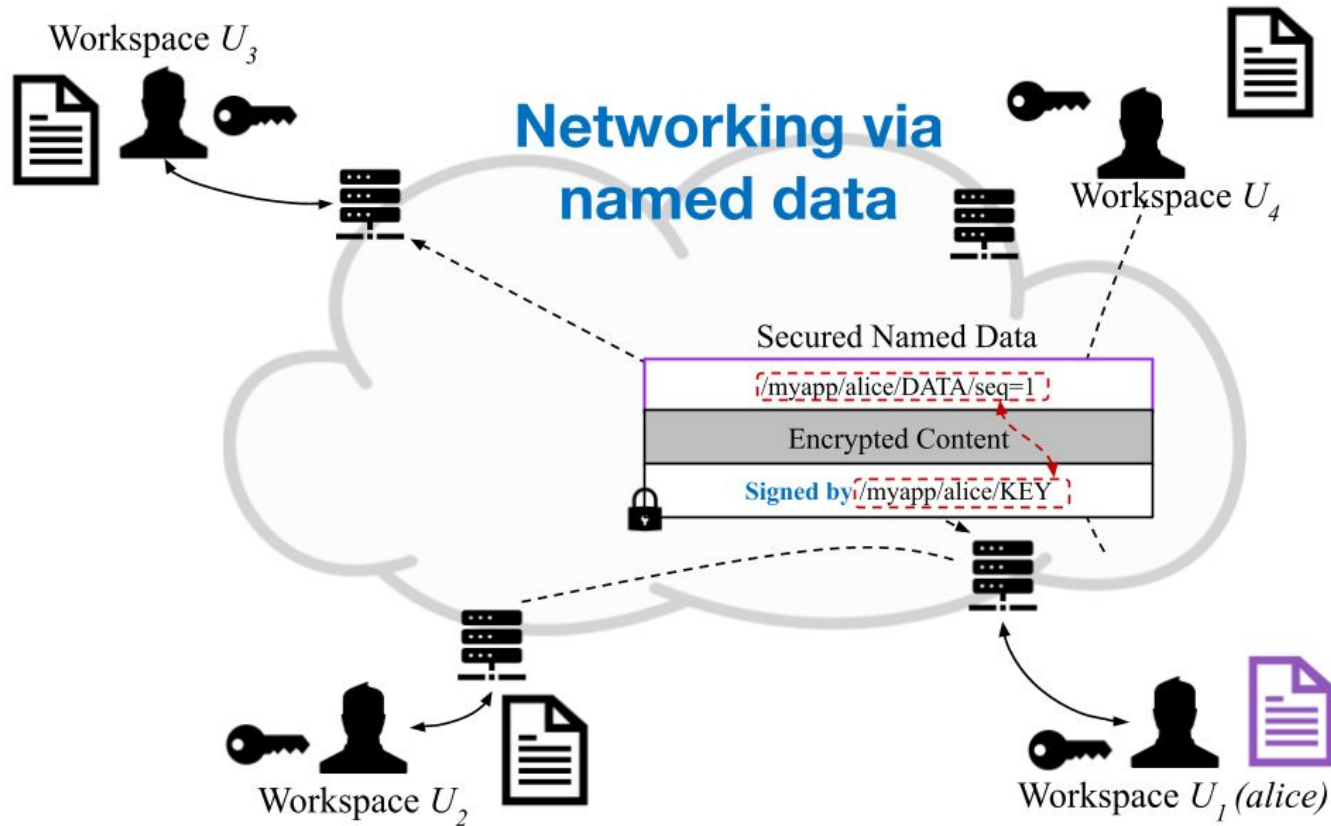
All of the above enabled by Named Data Networking

Design and Implementation Philosophy

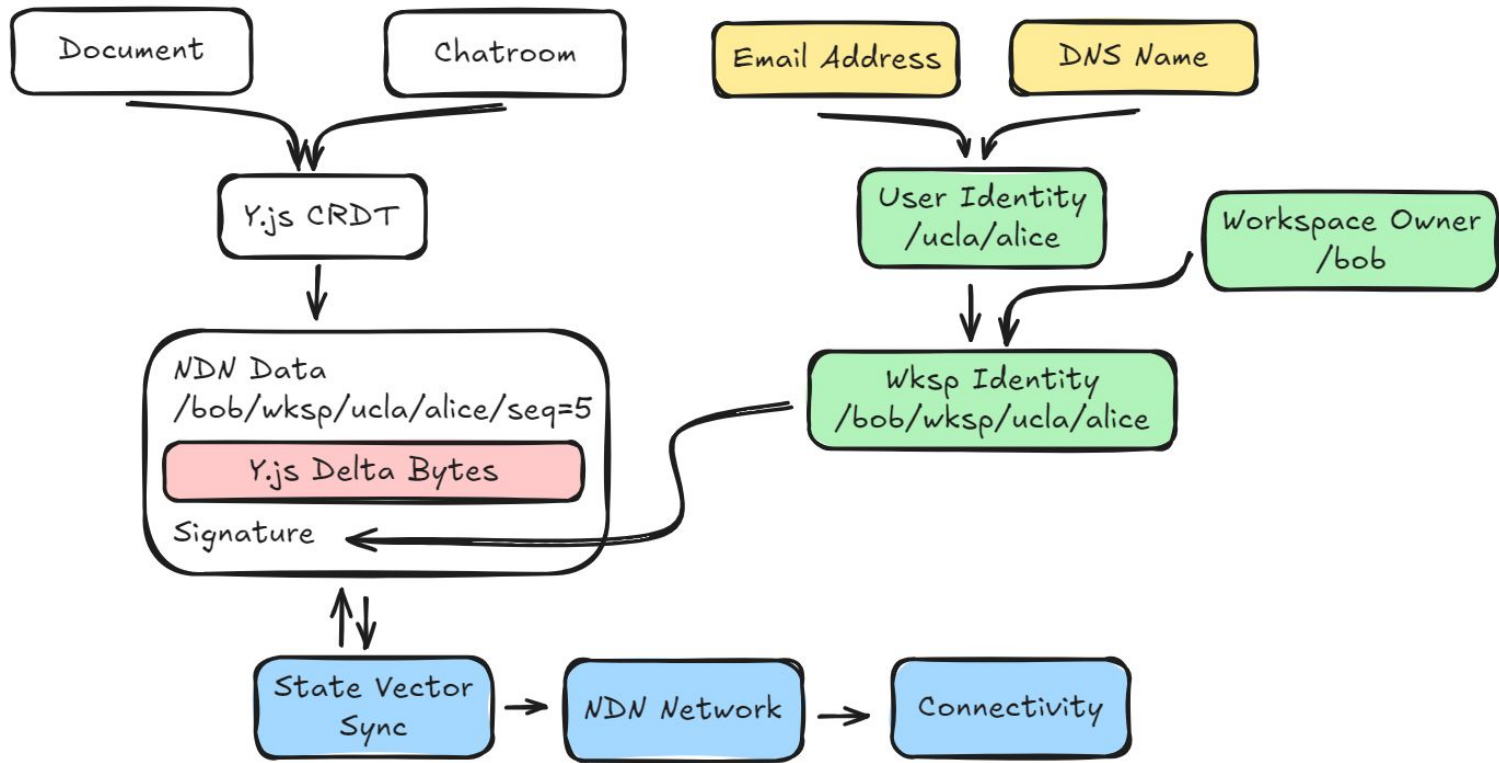
- Build a real decentralized application
 - Data-centric – applications handle data rather than channels
 - Secure data directly – eliminate gatekeepers
 - Peer-to-peer – no dependency on central servers
- Implement a generalized set of libraries usable in other applications
 - Implemented as a part of the NDNd standard library (Golang)
 - Precursor implementation in NDNts (TypeScript)
- Focus on usability and user experience
 - Minimize human costs

Data Centric Apps with NDN

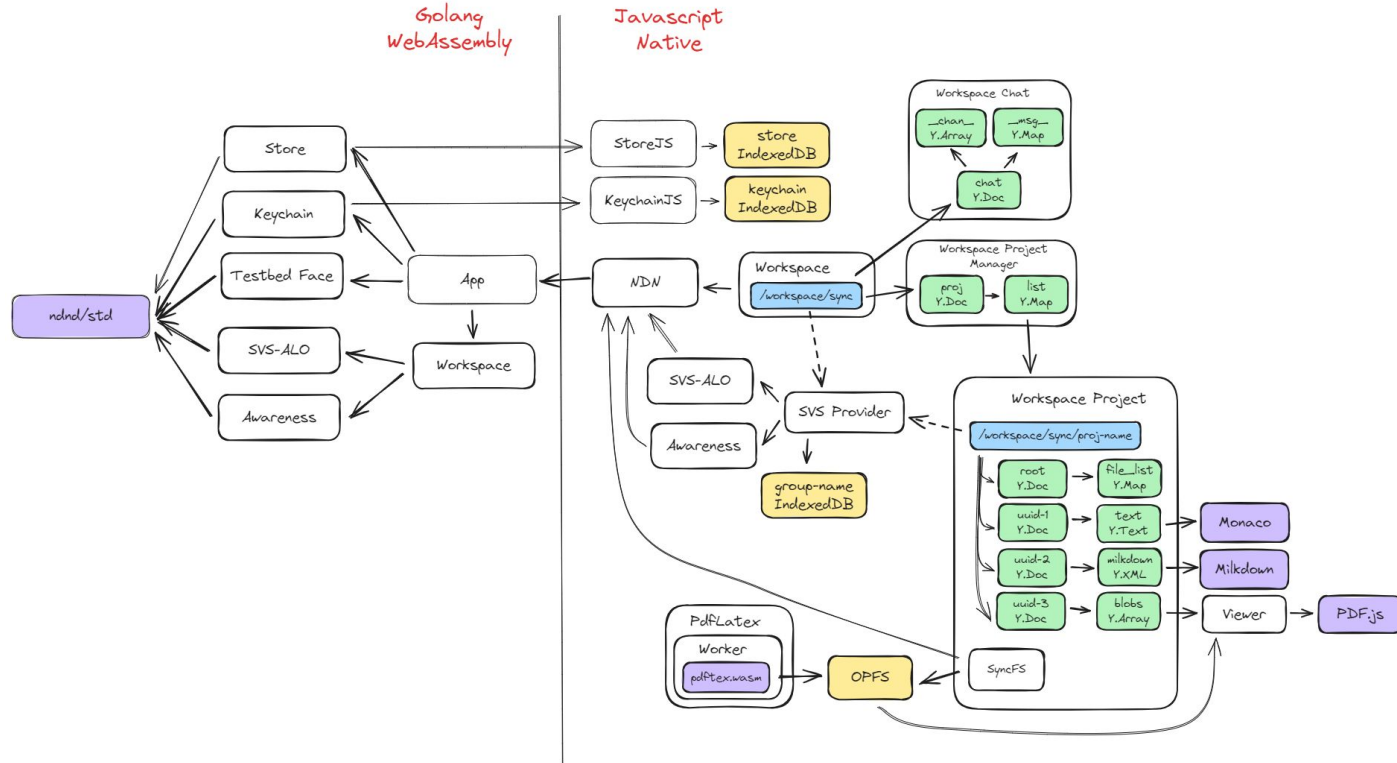
- **Producer** publishes **Data**: named and secure each piece
 - Just make one's data available
 - Data \neq Data packet
- Consumer fetches desired data by **name**, validates them
- Synchronize the dataset among all parties by **Sync**
- Resilient communication: make use of any available connectivity
 - Ethernet, WiFi, Bluetooth
 - TCP/UDP tunnels
 - HTTP, WebSocket
 - Avian Carriers



Ownly Architecture



Ownly Architecture - Implementation



Ownly Application - CRDT

```
1 \documentclass[12pt]{article}
2 \usepackage{lingmacros}
3 \usepackage{tree-dvips}
4 \begin{document}
5
6 \section*{Section 1}
7
8
9
10 \end{document}
11
```

The Document



Yjs CRDT

@xinyu/DATA/1

Insert: "\docu..."
Author: Xinyu

@tianyuan/DATA/1

Insert: "\sect..."
Author: Tianyuan

@xinyu/DATA/2

Insert: "\end{..."
Author: Xinyu

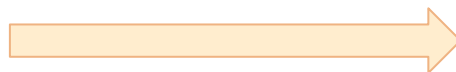
- **Sequentialize** changes to solve conflicts
- Mature algorithms and implementations
- Eventual consistency

Changes

Example Workflow: Capture the Change

```
1 \documentclass[12pt]{article}
2 \usepackage{lingmacros}
3 \usepackage{tree-dvips}
4 \begin{document}
5
6 \section*{Section 1}
7
8
9
10 \end{document}
11
```

CRDT gets notified and
extracts a new **change**

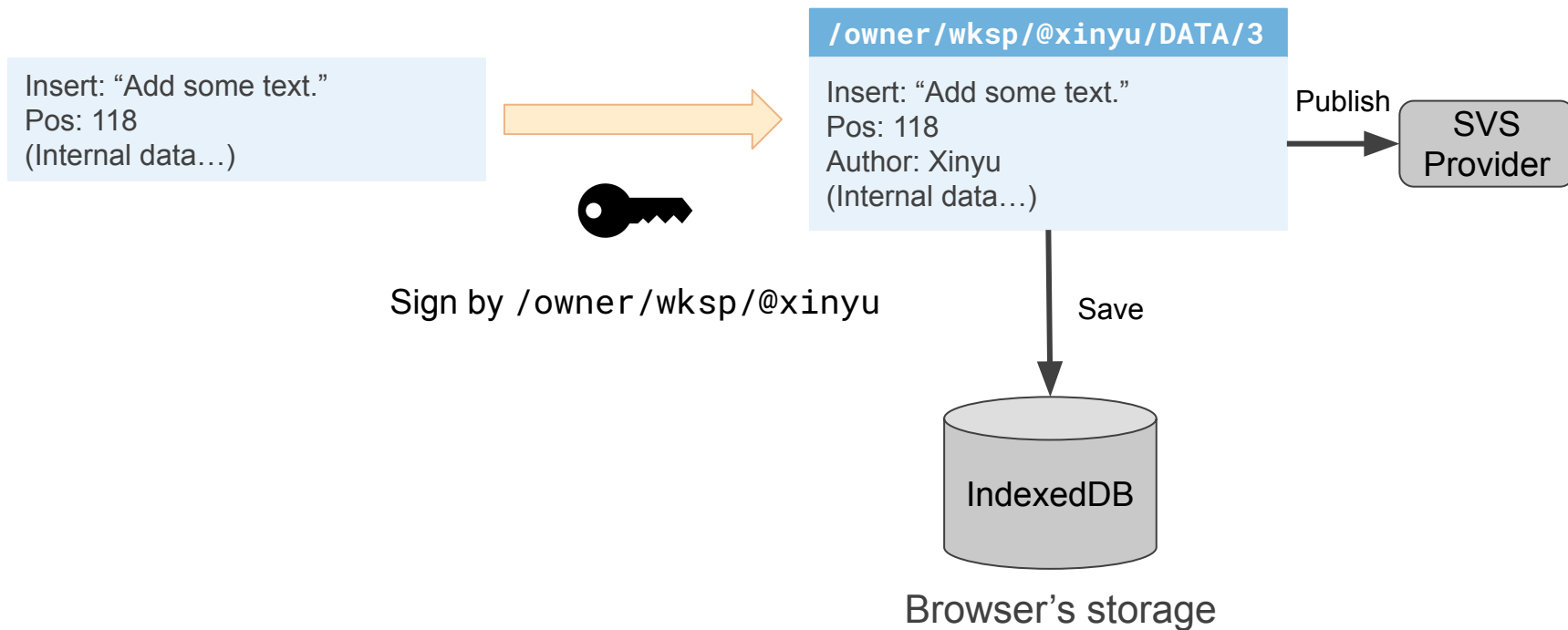


Yjs CRDT

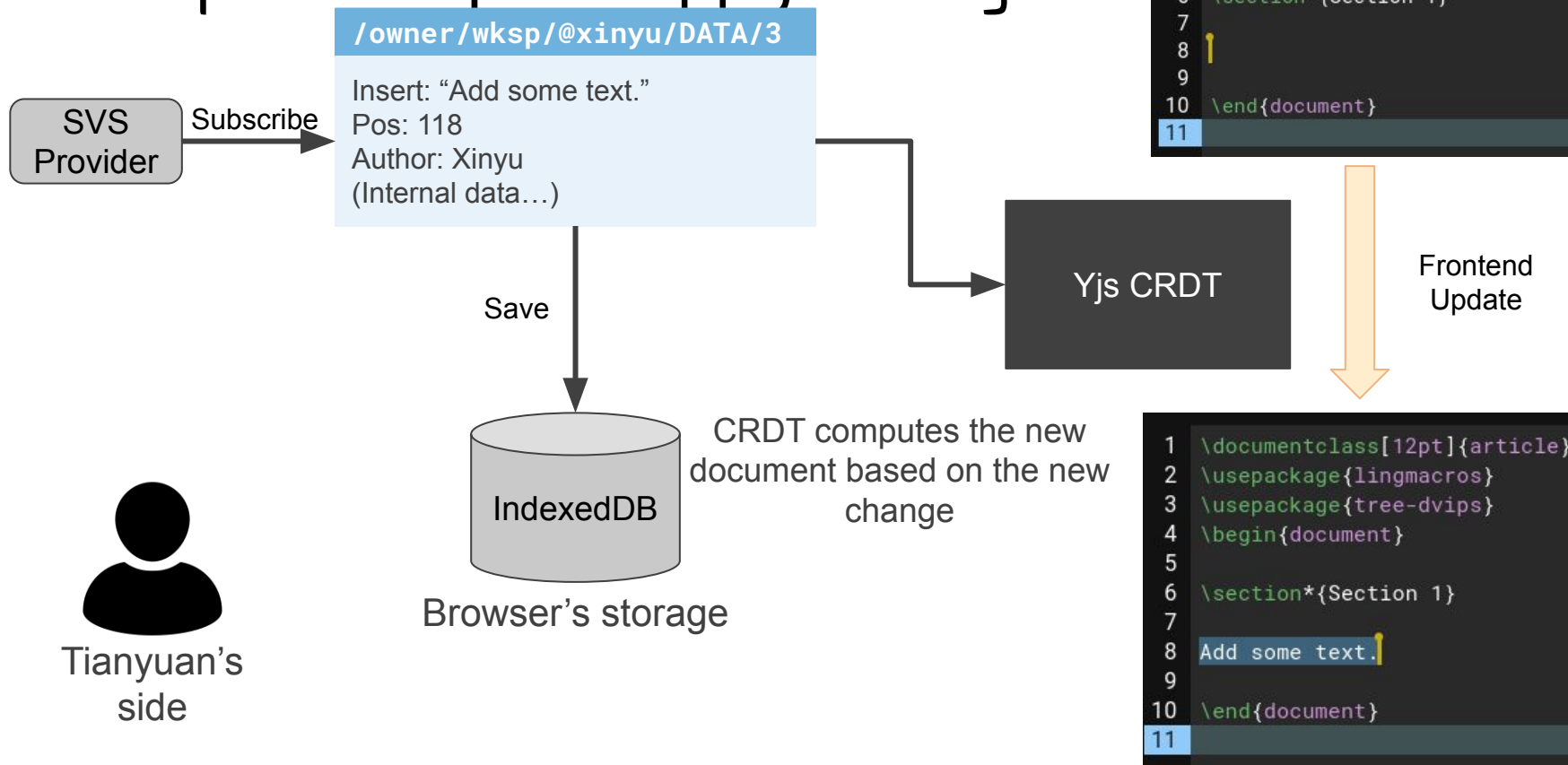
```
1 \documentclass[12pt]{article}
2 \usepackage{lingmacros}
3 \usepackage{tree-dvips}
4 \begin{document}
5
6 \section*{Section 1}
7
8 Add some text.
9
10 \end{document}
11
```

Insert: "Add some text."
Pos: 118
(Internal data...)

Example Workflow: Wrap into Data Record



Example Workflow: Apply Change



Browser as a platform

- Ownly is implemented in TypeScript and Go
 - Runs on all platforms with WebAssembly
- Browsers support most functionality needed to run NDN Apps
 - Local-first functionality, e.g. persistent databases and filesystem
 - Continuously improving WASM support
- Multiple connectivity options
 - WebSocket (current implementation)
 - QUIC Datagrams (NDNts)
 - WebRTC (?)

Browser as a platform

The screenshot displays the Ownly web interface, which functions as a browser-based platform for editing and compiling LaTeX documents. The interface is divided into three main sections: a sidebar on the left, a central code editor, and a right-hand preview pane.

Sidebar (Left): The sidebar is dark purple and contains the 'Ownly' logo at the top. Below it, there are three sections: 'PROJECTS' with a list of 'documents' containing 'main.tex' and 'readme.md'; 'DISCUSSION' with a list of '# general' and '# random'; and 'WORKSPACE' with an 'Invite people' button. At the bottom of the sidebar, there is a 'Share your Identity' button and a user profile icon for 'suns.cs.ucla.edu'.

Code Editor (Center): The central area is a dark-themed code editor showing LaTeX source code. The code is as follows:

```
4
5 \usepackage{amsmath} % \usepackage is a command that allows you to add
  functionality to your LaTeX code
6
7 \title{Simple Sample} % Sets article title
8 \author{My Name} % Sets authors name
9 \date{\today} % Sets date for date compiled
10
11 % The preamble ends with the command \begin{document}
12 \begin{document} % All begin commands must be paired with an end command
  somewhere
13   \maketitle % creates title using information in preamble (title,
    author, date)
14
15   \section{Hello World!} % creates a section
16
17   \textbf{Hello World!} Today I am learning \LaTeX. %notice how the
    command will end at the first non-alphabet character such as the .
    after \LaTeX
18   \LaTeX{} is a great program for writing math. I can write in line
    math such as  $a^2+b^2=c^2$  %% tells LaTeX to compile as math
19   . I can also give equations their own space:
20   \begin{equation} % Creates an equation environment and is compiled as
    math
21     \gamma^2+\theta^2=\omega^2
22   \end{equation}
23   If I do not leave any blank lines \LaTeX{} will continue this text
    without making it into a new paragraph. Notice how there was no
    indentation in the text after equation (1).
24   Also notice how even though I hit enter after that sentence and here
    $\downarrow$
25   \LaTeX{} formats the sentence without any break. Also look
    how it doesn't matter how many spaces I
    put between my words.
26
27   For a new paragraph I can leave a blank space in my code.
28
29 \end{document} % This is the end of the document
```

Preview Pane (Right): The right-hand pane shows the rendered output of the LaTeX code. It has a 'Compile' button and a 'Download' button at the top. The rendered document has the title 'Simple Sample', the author 'My Name', and the date 'April 2, 2025'. It features a section titled '1 Hello World!' followed by a paragraph of text that includes the rendered equation $\gamma^2 + \theta^2 = \omega^2$ labeled (1). The text explains how LaTeX handles line breaks and indentation, demonstrating the effect of the \downarrow command and the $\LaTeX{}{}{}{}$ command.

Exchanging App Data Securely: Basic Ingredients

- Naming
 - Each user needs a unique name – DNS and derived names
 - Each piece of data is uniquely named and immutable
- Security
 - Each piece of data should be directly secured
- Sync
 - Consumers should be notified of data production
- Storage
 - Data should always be available for consumption

Naming Users and Data

- Using DNS and DNS derived names:
 - DNS name delegation
 - With an assigned DNS name N, one can assign any other names under N
 - “named-data.net”
- Name each user
 - “gmail.com/@alice”
- Name each application instance
 - “named-data.net/ndncomm-2025”
- Name each piece of data
 - “named-data.net/ndncomm-2025/gmail.com/@alice/DATA/seq=8”
- Important: well-designed **naming conventions**

Securing Named Data

- Trust Anchor
- Certification of identity
 - For the user's identity, e.g. "gmail.com/@alice"
 - In the application, e.g. "named-data.net/ndncomm-2025/gmail.com/@alice"
- Security Policies – who can say what
 - Semantic naming enables systematic definition of security policies

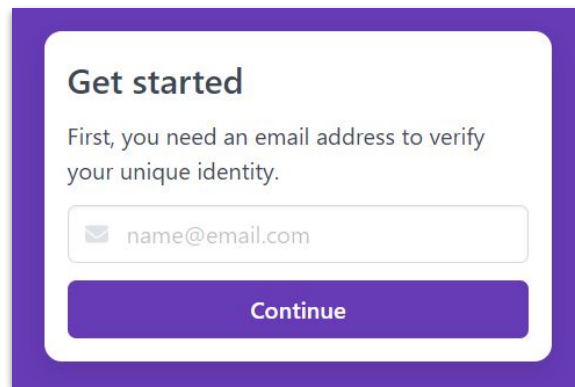
Users need to security bootstrapped

Security Bootstrapping in NDN

- An NDN based system: made of named entities with trust relations among them
 - Network delivery uses the same namespace by apps/users at high layers
- A new NDN entity **N** entering the system: the bootstrapping step provides **N** a set of security parameters (certificate, trust anchor, security policies)
 - Producer: which key to use to sign data
 - Consumer: is received data signed by a valid key
- After bootstrapping: **N** can sign data produced, verify received data

Identity Bootstrapping in Ownly

- Install Trust Anchor out-of-band
 - Bundled with the application code
 - Can be user-configurable (planned)
- Reuse external identity for users
 - Email address
 - DNS namespace ownership (planned)
 - Requirement: unique and verifiable
- Verify external identity and issue identity certificate
 - NDN CERT protocol, CA running on global NDN testbed
 - Email Challenge
 - Certificate proves user identity to other Ownly users
 - NDN CERT CA only verifies identity, does not control application



Get started

First, you need an email address to verify your unique identity.

Continue

Security Policies – Trust Schema

- Ownly uses LightVerSec to define trust policy
- Static schema bundled with compiled application

```
// Only owner can sign all user certificates
// The delegation will happen using a separate CrossSchema
#user_cert: #owner/wksp/#user/#KEY <= #owner_cert
#owner_cert: #owner/wksp/#owner/#KEY <= #owner_id_cert
#owner_id_cert: #owner/#KEY <= #testbed_site_cert | #testbed_root_cert

// Testbed trust model
#testbed_site_cert: /"ndn"/_/_/#KEY <= #testbed_root_cert
#testbed_root_cert: /"ndn"/#KEY

// Project sync group
#proj: #owner/wksp/proj
#proj_data: #proj/#user/_/_ <= #user_cert
#proj_blob: #proj/#user/_/"32=blob"/_ <= #user_cert
```

Dynamic Security Policies – CrossSchema

- Security policies may need to change at runtime
 - E.g. inviting users to a workspace
 - Need to allow users to publish data under the owner's namespace
- Break up schema into smaller pieces
 - These are generated at runtime
 - The schema itself is a signed NDN data
- Producer attaches required schema to NDN Data
 - NDN Data describes how it can be verified
 - KeyLocator + CrossSchema

Invitations in Ownly

- Invitations are policies
 - Allow a user identity to read and publish data in the workspace
- Multiple possible designs
 - Allow users to self-certify their workspace identity
 - Owner produces policy rule to allow this for a particular identity
 - CrossSchema is attached to workspace identity certificate
 - Owner directly certifies each user
 - Requires a key directory and an additional exchange
 - Each workspace can have an independent trust anchor
 - Eliminates identity verifier from trust chain

Invite people to NDN Call Agenda

An invitation will be generated for each email address. Note that Ownly does not automatically send any emails – you must ask the recipients to join the workspace using the email address listed below using the invite link.

<https://ownly.work/join/ndn-edu-ucla-varunpatil-ndn--call/?label=NDN+Call+Agenda>

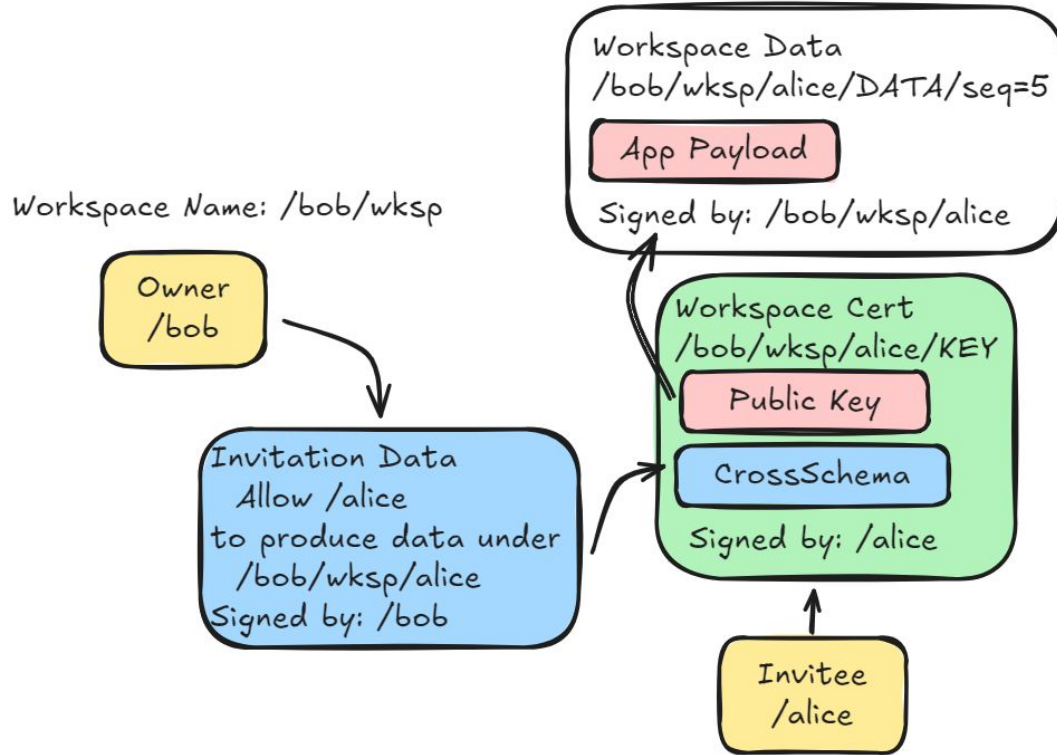
Enter upto 100 email addresses or NDN names below, one on each line

name@example.com
/ndn/user-name

Cancel

Invite

Invitations in Ownly



Confidentiality

- Group key encryption
- Two shared secrets
 - PSK - shared with invitation link (static)
 - DSK - managed in the group (dynamic)
- Key exchange during bootstrapping
 - Request DSK once you can produce data
 - Any group member can respond (after ECDH exchange)
- No central controller - E2EE

Decentralized Transport with NDN Sync

- How to notify users of a new change?
- State Vector Sync protocol
 - Producer increments a sequence number on new data
 - Reliably synchronize sequence numbers (multicast a “Sync Interest”)

Interest 879	
Name 119	/ndn/multicast/ndn/edu/ucla/varunpatil/irltest2/z50-Fj7QTsd1yH0gW5DL2/32=svs/54= =%03/params-sha256=64bf4fe742eca6f927440dbf475ca15c1392d4d9c1cee72e900a1edd5bd8 8705
Nonce 4 af4fc273	InterestLifetime 2 1000
ApplicationParameters 744	
Data 740	
Name 110	/ndn/edu/ucla/varunpatil/irltest2/z50-Fj7QTsd1yH0gW5DL2/ndn/edu/ucla/varunpat il/56=g%E5%BD%CD/32=svs/54=%00%062p%A8%93%2B%18
MetaInfo 3	ContentType 1 Blob
Content 450	
StateVector_V3 446	
StateVectorEntry 54	Name 28 /ndn/edu/ucla/g/stheera

StateVectorEntry 52	Name 27 /ndn/edu/ucla/cs/lixia	
SvSeqNoEntry_V3 10	SvBootTime_V3 4 1741900789	SvSeqNo_V3 2 2921
SvSeqNoEntry_V3 9	SvBootTime_V3 4 1743832152	SvSeqNo_V3 1 73
StateVectorEntry 44	Name 30 /ndn/edu/ucla/cs/tianyuan	
SvSeqNoEntry_V3 10	SvBootTime_V3 4 1741900774	SvSeqNo_V3 2 2387
StateVectorEntry 44	Name 30 /ndn/edu/ucla/cs/xinyu.ma	
SvSeqNoEntry_V3 10	SvBootTime_V3 4 1742247681	SvSeqNo_V3 2 921
StateVectorEntry 35	Name 22 /ndn/edu/ucla/jzhi	
SvSeqNoEntry_V3 9	SvBootTime_V3 4 1742339002	SvSeqNo_V3 1 213

SVS-PS

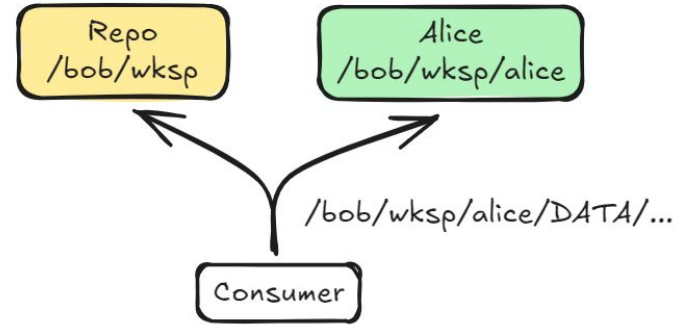
- High Level API to synchronize data over SVS
 - `SvsPS(<group>, <name>)`
 - `publish(<name>, <blob>)`
 - `subscribe(<producer-prefix>, callback(<producer>, <blob>))`
- Fetches data from all subscribed producers automatically
 - Per-member ordering
 - Eventual consistency
- Handles security, segmentation and snapshots internally
 - Both signing and validation are internal (just supply policy and trust anchor)
 - Multiple snapshot strategies - reduce bootstrapping time
- Alternative - SVS PubSub (SVS-PS)
 - Allows application to selectively fetch data objects

Sync Group Granularity

- Trade-off
 - More groups = more overhead
 - Less groups = more unused information
- One Sync group for each workspace
 - Exchange workspace metadata
 - Membership management
 - Chat Module
- One Sync group for each “project”
 - Directory structure
 - CRDT deltas

Transparent In-Network Storage - Sync Repo

- Repo runs as a network-provided service
 - Run by a provider or ISP for a cost
- Application asks repo to join Sync group
 - Announces app data prefix
- Repo fetches all data on the group
 - Needs to verify data before storage
 - Does not need to decrypt – untrusted storage
 - Makes data available even producer is gone
- Sync functions as usual, but now with availability
 - “Transparent” – application does not interact with storage
 - Repo tracks latest Sync Data



Ownly - Deployment

- NDN Connectivity using NDN Testbed
 - <https://named-data.github.io/testbed/>
 - Does not have control power
- Testbed NDNCERT infrastructure for identity verification
 - Alternative – each workspace can run its own NDNCERT CA
 - Tested with original NDN Workspace implementation
- Static application hosted on Netlify

Future / Ongoing Work

- Completing Encryption Implementation
 - Consider key rotation
- Verifying data after certificates expire
 - Ideas from NDN DeLorean
- Snapshot efficiency and rollback
 - Make bootstrapping faster
 - Reduce duplication of data