Querying Data on Decentralized Networks

Matteo Campanelli

Protocol Labs

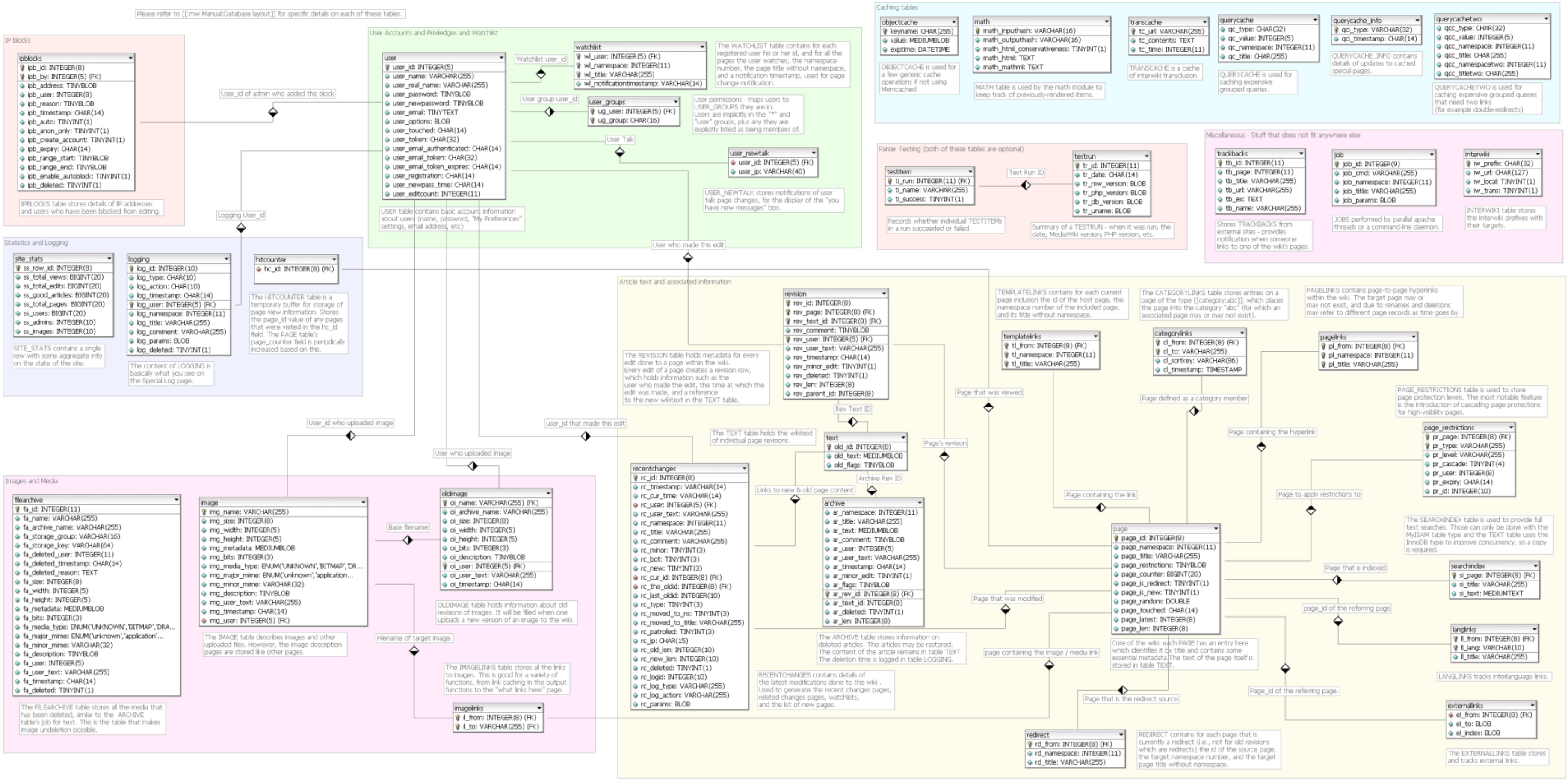
Meta: A link to these slides

- At: binarywhales.com/EC22.pdf
  - They contain additional pointers, notes and references

This talk at the high level:

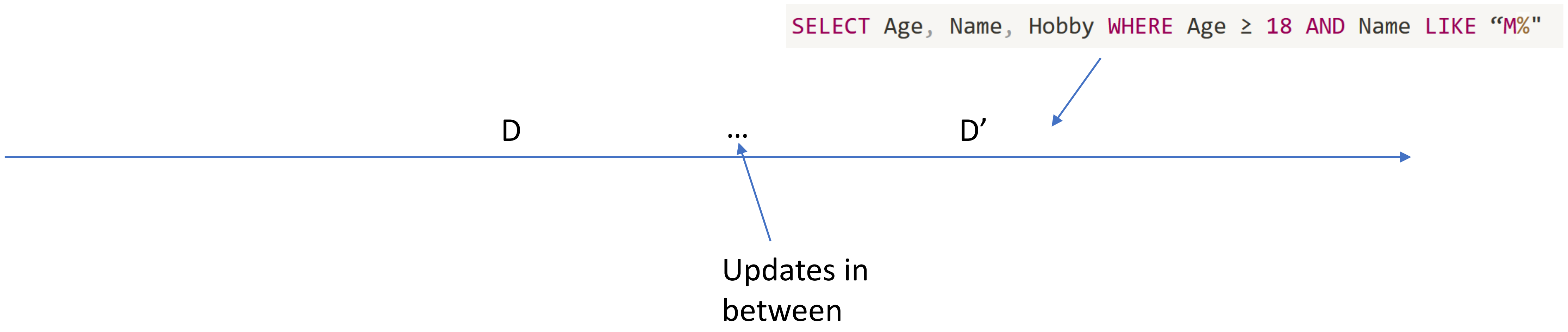
- Here is an application
- Unclear if what is out there is the best (or good) solution for it

Data are at the core of applications



A schema of Wikipedia DB. Source: Wikipedia

Data in applications traditionally



Data and queries in this talk

- Queries on data that are:
  - On chain*
  - Verifiable

*on chain ~ on decentralized networks

On chain data

[x] = “small digest to x”

Change 1:
- Chain storage is expensive
- no actual data on chain; just digest

On chain data

```
SELECT Age, Name, Hobby WHERE Age ≥ 18 AND Name LIKE “M%"
```

[D]

[D’]

Result +
Prf(query_result)

Change 2:

- We cannot trust who delivers the query response
- Add **Publicly Verifiable** proof of correct query execution
    - Verify([D’], Query, Result, Proof) -> accept/reject

[x] = “small digest to x”

Why?

- Smart contracts can verifiably query DBs
  - E.g. “Update internal state with (verified) result of query Q”
  - E.g. “Release coins to the result of query*
    SELECT ARGMAX(effectiveness_score) FROM organizations WHERE type=charity
- In general: achieve complex logic & state with succinctness and verifiability

*Not real SQL syntax, but you get the gist.

Question of this talk: *How to build this?*

- There are cryptographic primitives that are available
- But it is unclear* if any of them is right for the job

* At least, it is unclear to yours truly

Let’s start:
what’s the first candidate we could think of?

SNARKs

```
SELECT Age, Name, Hobby WHERE Age ≥ 18 AND Name LIKE "M%"
```

Result + SNARK_proof

[D]

…

[D’]

Some SNARK proof of updates in between

[x] = “small digest to x”

SNARKs as candidate for the job

Some features of SNARKs
+ expressive (any query)
+ good time(V) and |prf|
- **Too general?**
- Prover has high cost

The (potential) issue with “too general”

- Conversions of computations into circuits is generally expensive
- Conversions of computations into circuits/RAM/etc is generally messy
- **There are many lost opportunities for optimizations**

Going around "too general"
Option #1: engineering SNARKs appropriately

- **Research Question A:**
  What are ways we can optimize general SNARKs for a database-like setting?

Nova: Recursive Zero-Knowledge Arguments from Folding Schemes

Abhiram Kothapalli[*†] Srinath Setty[*] Ioanna Tzialla[‡]

[*]Microsoft Research [†]Carnegie Mellon University [‡]New York University

vSQL: Verifying Arbitrary SQL Queries over Dynamic Outsourced Databases

Yupeng Zhang[*], Daniel Genkin[†,*], Jonathan Katz[*], Dimitrios Papadopoulos[‡,*] and Charalampos Papamanthou[*]
[*]University of Maryland [†]University of Pennsylvania [‡]Hong Kong University of Science and Technology
Email: {zhangyp,cpap}@umd.edu, danielg3@cis.upenn.edu, jkatz@cs.umd.edu, dipapado@cse.ust.hk

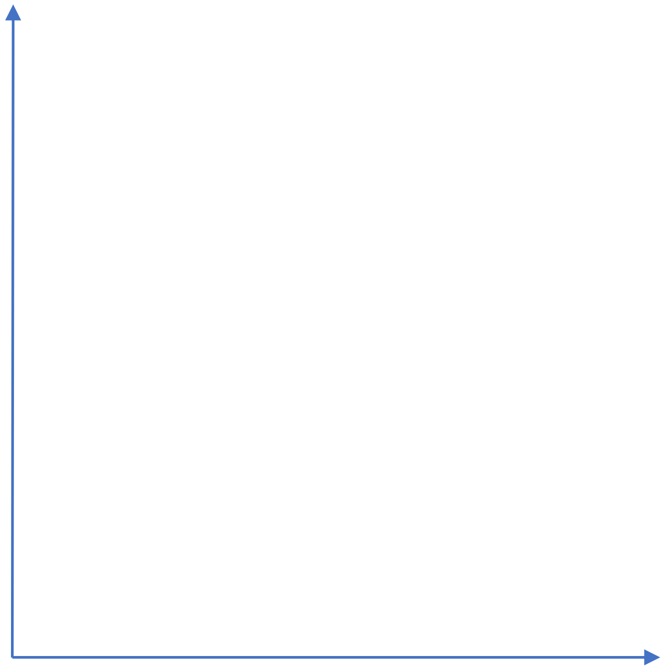
LegoSNARK: Modular Design and Composition of Succinct Zero-Knowledge Proofs

Matteo Campanelli[1], Dario Fiore[1], and Anaïs Querol[1,2]

Going around "too general"

Option #2: let's think "non-general and non-SNARKs"

- **Warm-up question.**



Succinct Functional Commitments (sFC)

Authenticated Data Structures

- Very active area earlier in the millennium (~ 2002-2013)
- **Goal**: ~ making cryptographic version of data structures
  - Think: interval trees, dictionaries, etc.
- **The question for us:** What can they bring to the table here?
- **Challenges** (?):
  - ”Different” security definition (e.g. rely on trusted updates/generation)
  - Most of them are very specific to certain tasks

Streaming Authenticated Data Structures

Charalampos Papamanthou[1], Elaine Shi[2], Roberto Tamassia[3], and Ke Yi[4]

Efficient Authenticated Data Structures for Graph Connectivity and Geometric Search Problems*

Michael T. Goodrich † Roberto Tamassia ‡ Nikos Triandopoulos ‡,§

- **Research Question B:**

  How can we leverage existing constructions/techniques in Auth Data Structures (and sFC) for our goal?

Subquestions:

Are there limitations in their security models for our setting and how to go around them?

When are they really more efficient?

- **(Meta) Research Question C*:**
  What is the minimal set of “non general” queries that would be worth having in applications?

*** Support question for question B**

That's all! Questions?

- **Research Question A:**
  What are ways we can optimize general SNARKs for a database-like setting?
- **Research Question B:**
  How can we leverage existing constructions/techniques in Auth Data Structures (and sFC) for our goal?
- **Research Question C:**
  What is the minimal set of "non general" queries that would be worth having in applications?

These slides (with additional pointers) available at:
binarywhales.com/EC22.pdf

For questions/comments:
matteo@protocol.ai

Some pointers on Authenticated Data Structures

- IntegriDB: this system is the closest thing to something that could be used in practice. Its problem seems to be the requirement (as in other ADS) of a secret key for certifying the updates. We do not expect to have a secret key in our setting
- The two works it is based on
  - https://eprint.iacr.org/2010/455.pdf
  - https://eprint.iacr.org/2013/724.pdf
- Streaming Authenticated Data Structures (2013): http://elaineshi.com/docs/streaming.pdf

On existing solutions

Auth Data Structures	Succinct Functional Commitments	Snarky approaches*
+ vast literature	+ "nice(r)" assumptions (than SNARKs)	+ expressive (any query)
+ simple constructions	+ (some) simple constructions	+ extractability
? Succinct updates?	+ succinctness / non det	+ good time(V) and |prf|
? Security model for non-det ?	? When is eval binding sufficient for application(vs, say, extractability)s ?	**- Prover is high cost**
- Limited to specific queries	? Efficient in practice ?	**- Too generic? Losing opportunities for optimizations?**
	? Expressive queries ?	- Less "nice" Assumptions (comparatively)

*includes vSQL and recursive approaches

Why care about better proving time

- Counterarguments to “Just delegate to somebody with more powerful hardware”
  - Privacy
  - Democratizing as much as possible roles in decentralized networks (o.w. this is introducing yet another plutocracy)
  - Even powerful hardware is going to hit a point of max capacity at some point. Let’s lower that point with better proving time
  - The ”Why not?”-response: lower proving time is a better dimension for proof systems

Security of (streaming) ADS

Definition 3 (Security). *Let $\mathcal{A}$ be an SADS scheme consisting of the set of algorithms* $\{\mathsf{genkey}, \mathsf{initialize}, \mathsf{updateVerifier}, \mathsf{updateProver}, \mathsf{query}, \mathsf{verify}\}$, *$k$ be the security parameter, D_0 be the empty data structure and* $\mathsf{pk} \leftarrow \mathsf{genkey}(1^k)$. *Let also* Adv *be a PPT adversary and let d_0 be the state output by* $\mathsf{initialize}(D_0, \mathsf{pk})$.

- *(Update) For $i = 0, \ldots, h-1 = \mathsf{poly}(k)$,* Adv *picks the update* upd_i *to data structure D_i. Let* $d_{i+1} \leftarrow \mathsf{updateVerifier}(\mathsf{upd}_i, d_i, \mathsf{pk})$ *be the new state corresponding to the updated data structure D_{i+1}.*
- *(Forge)* Adv *outputs a query q, an answer α and a proof Π.*

We say that the SADS scheme $\mathcal{A}$ is secure *if for all $k \in \mathbb{N}$, for all* pk *output by algorithm* genkey, *and for any PPT adversary* Adv *it holds that*

$$\Pr\left[\begin{array}{r}\{q, \Pi, \alpha\} \leftarrow \mathsf{Adv}(1^k, \mathsf{pk});\ 1 \leftarrow \mathsf{verify}(q, \alpha, \Pi, d_h, \mathsf{pk}); \\ 0 \leftarrow \mathsf{check}(q, \alpha, D_h).\end{array}\right] \leq \mathsf{neg}(k)\,. \qquad (2.1)$$

Streaming Authenticated Data Structures

Charalampos Papamanthou[1], Elaine Shi[2], Roberto Tamassia[3], and Ke Yi[4]

Piperine

Replicated state machines without replicated execution

Jonathan Lee Kirill Nikitin* Srinath Setty
Microsoft Research **EPFL*