BLOCKSTACK BERLIN

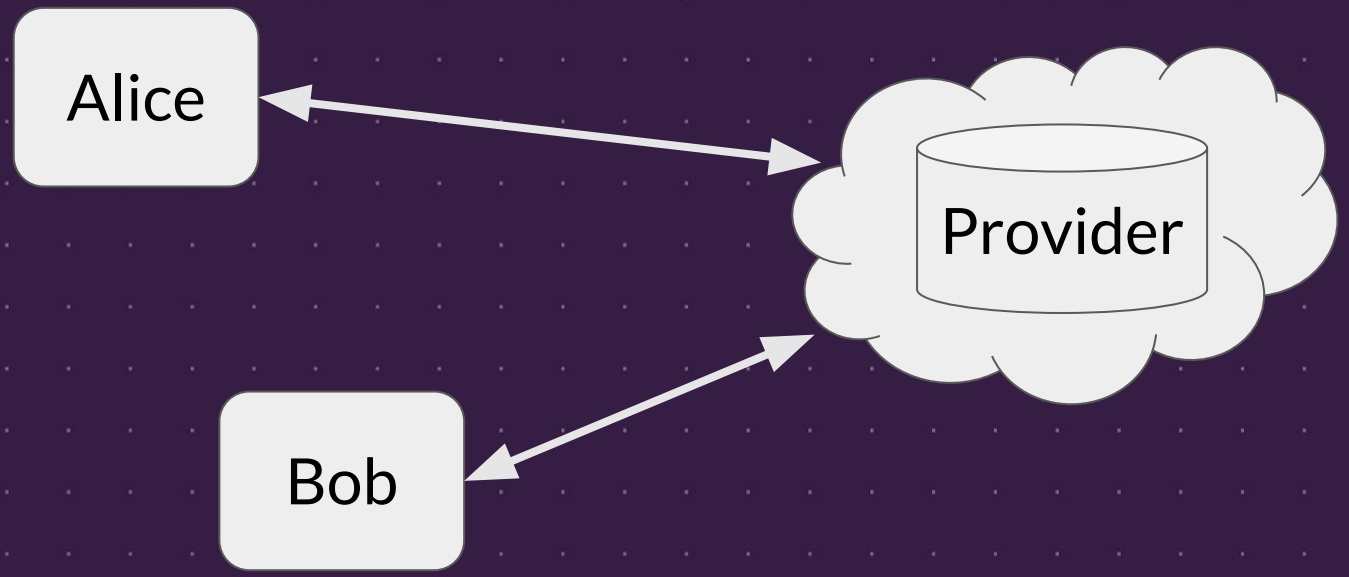
A SIGNATURE FUND EVENT

Gaia Storage: User-owned Data

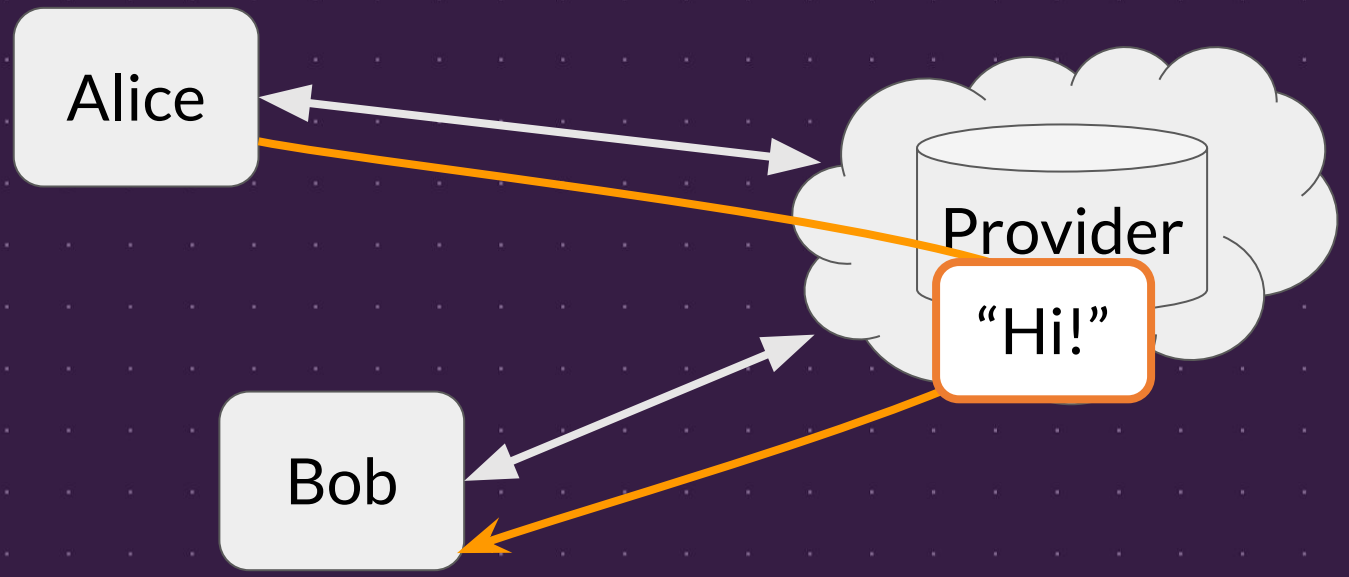
Aaron Blankstein

BLOCKSTACK

Provider-Oriented Storage



Provider-Oriented Storage



Provider-Oriented Storage

This is centralization!

Provider writes data on behalf of Alice and Bob
Provider governs how it is shared

Centralized Information Routing

How does Bob find Alice's message?

In a centralized system, the provider is *always* the source of information: Bob's client just asks the provider for the message

Problems of Centralized Storage

- Reads/writes are not strictly associated with the user
- Users cannot choose different storage providers
- Users cannot control who sees their data

For a user to own their data, these problems must be solved

Problems of Centralized Storage

- Reads/writes are not strictly associated with the user
- Users cannot choose different storage providers
- Users cannot control who sees their data

Solution:
Crypto. Signatures

Problems of Centralized Storage

- Reads/writes are not strictly associated with the user
- Users cannot choose different storage providers
- Users cannot control who sees their dat

Solution:
Crypto. Signatures

Solution:
Lookup Specification

Problems of Centralized Storage

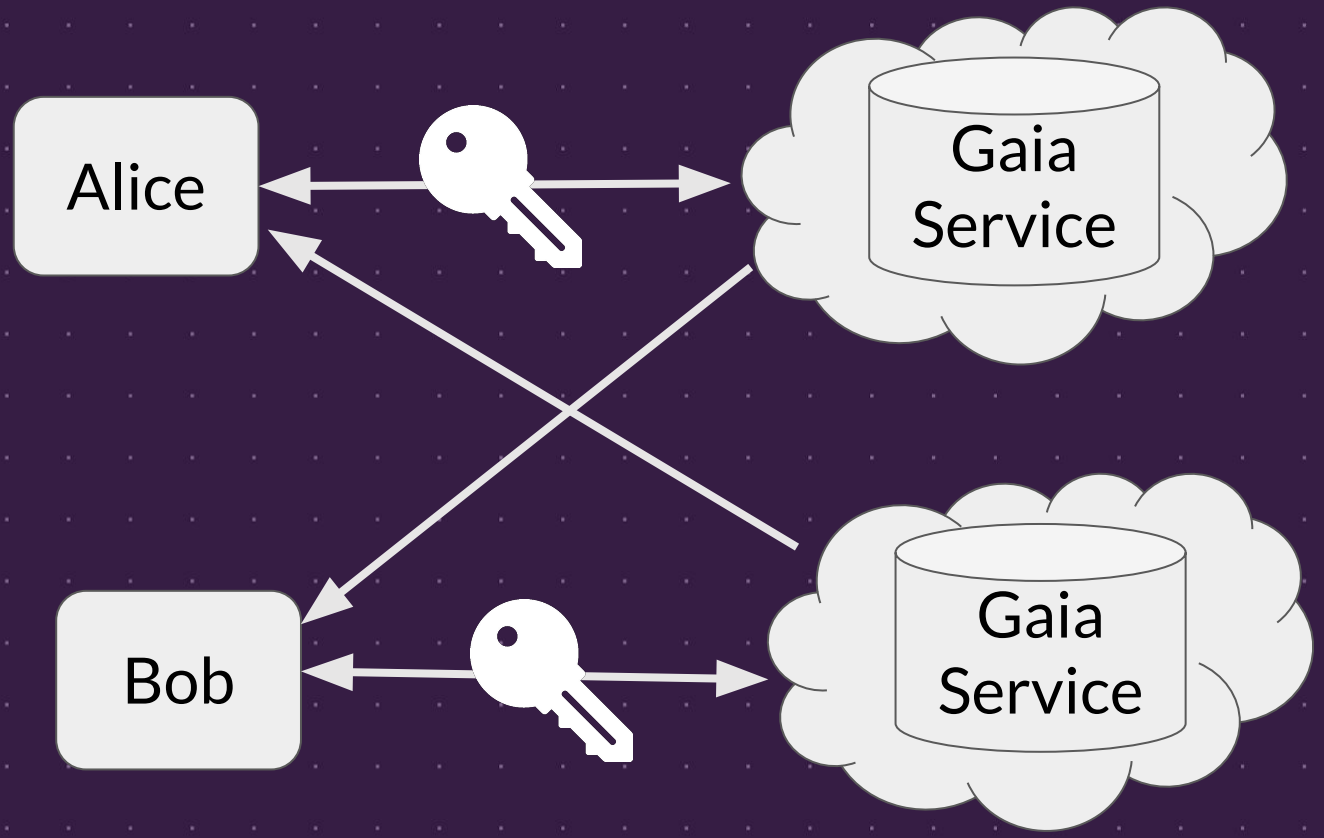
- Reads/writes are not strictly associated with the user
- Users cannot choose different storage providers
- Users cannot control who sees their dat

Solution:
Encryption

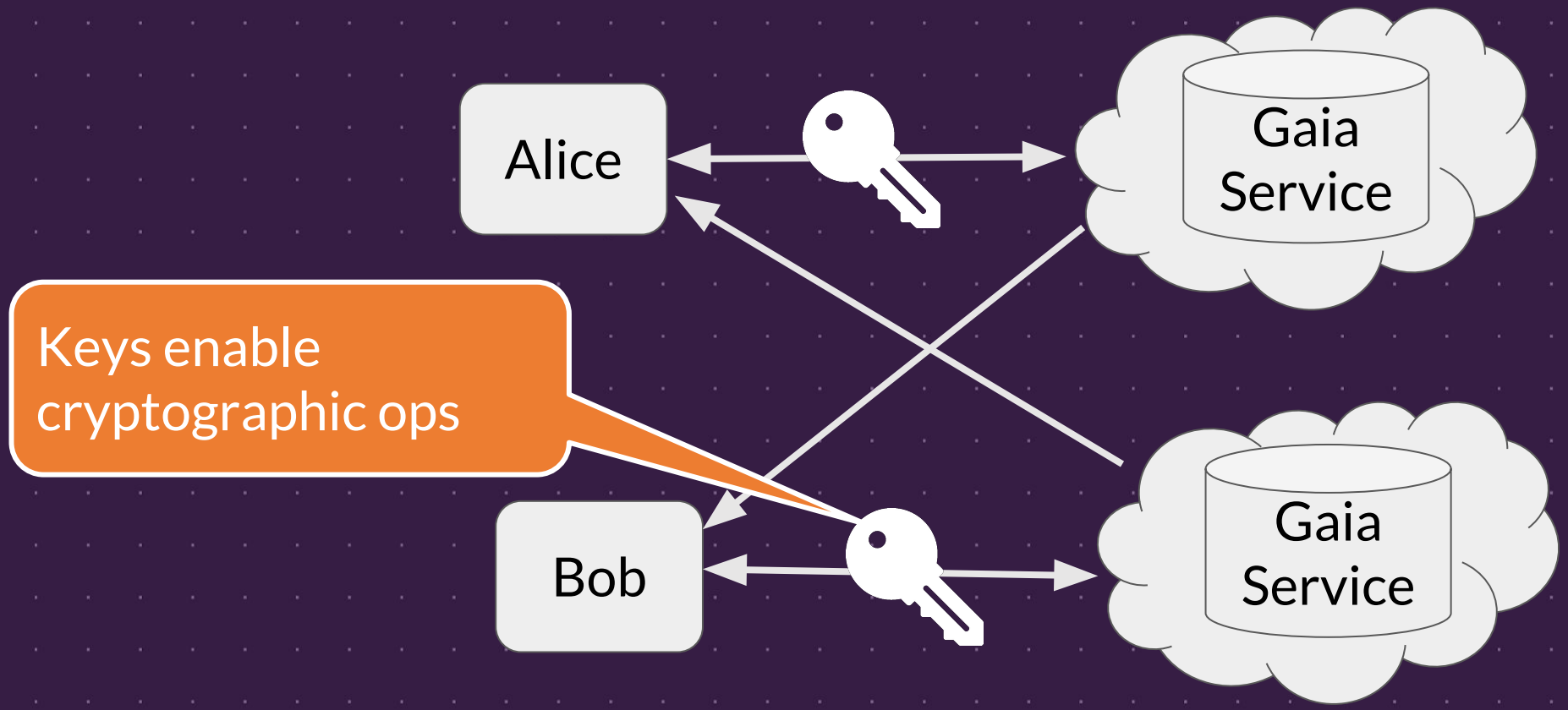
Solution:
Lookup Specification

Solution:
Crypto. Signatures

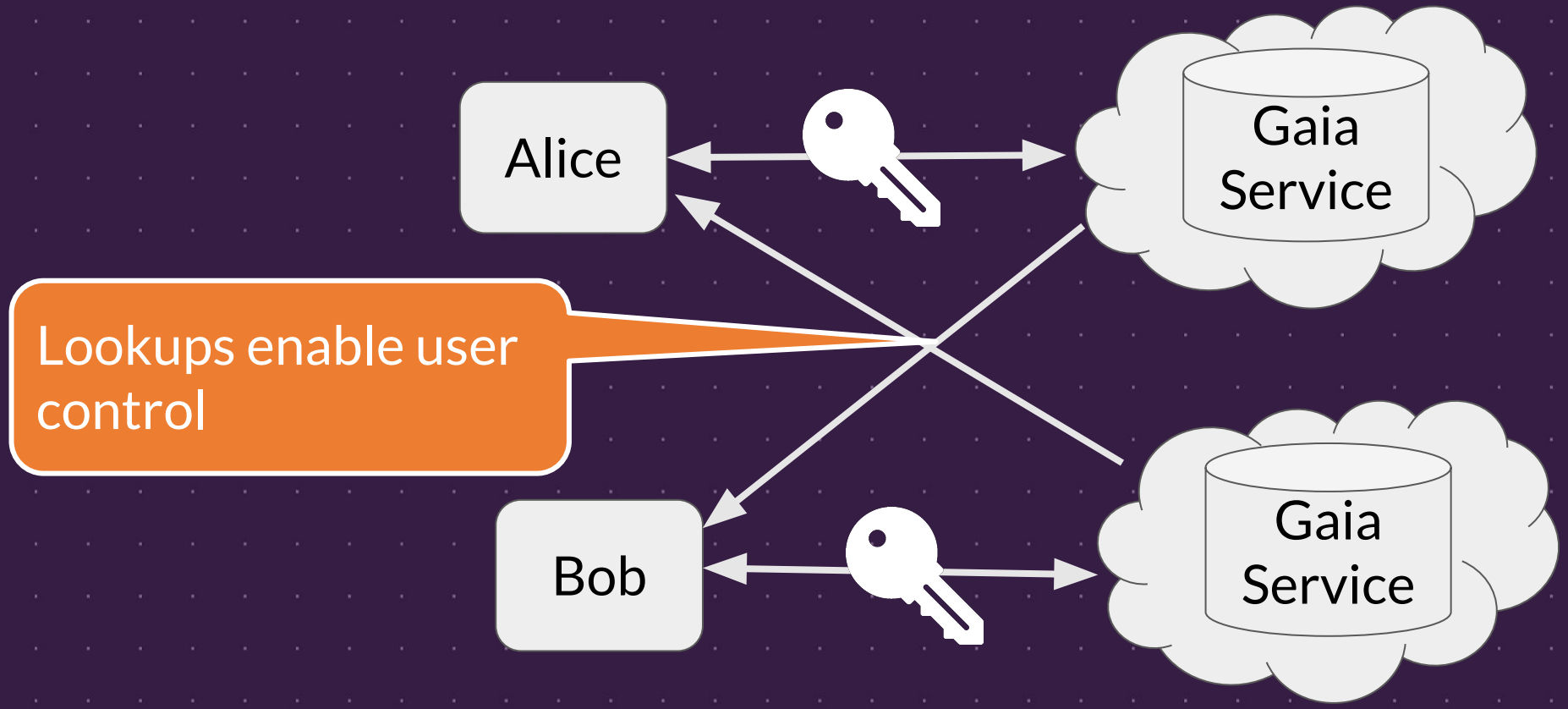
Gaia: User-Owned Storage



Gaia: User-Owned Storage



Gaia: User-Owned Storage



Looking Up User-controlled Data

1. Resolve username to data root/user profile (controlled via BNS and Atlas)
2. Discover app root (stored in data root)
3. Fetch data from the app root (Gaia service specification)

Looking Up User-controlled Data

1. Resolve username to data root/user profile (controlled via BNS and Atlas)
2. Discover app root (stored in data root)
3. Fetch data from the app root (Gaia service specification)

If Alice wants to change Gaia providers, or run her own:
She controls her data root (via name ownership in BNS), therefore, she can change the app root URL

Control of Data == Lookup Path

The method of *finding* data defines the control of data

By requiring applications to perform this multi-step lookup, we put control in the user's hands, because the lookup *starts* with a user-owned data source

The Gaia Interface

PUT /store/<public-key-hash>/<filename>

writes a file on behalf of user defined by public-key-hash

GET /<public-key-hash>/<filename>

reads file from user defined by public-key-hash

GET /hub_info/

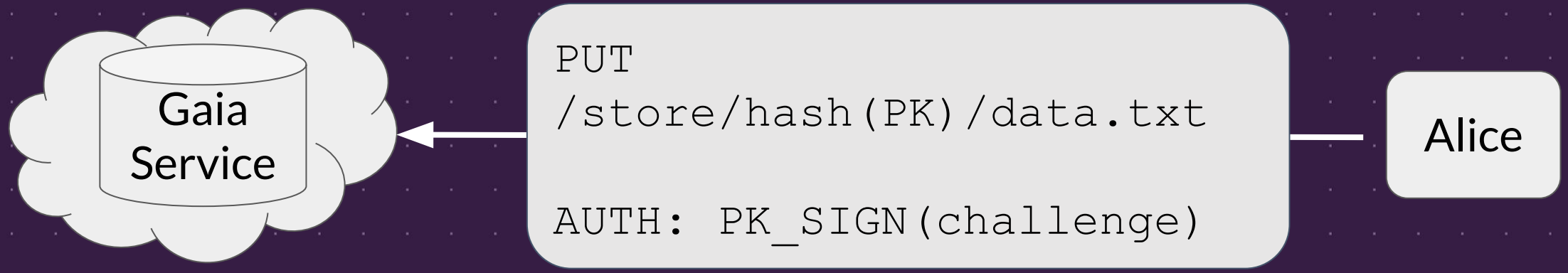
returns information about where the hub is writing

The Gaia Interface

The simplicity of this interface enables a wide range of implementations to exist:

A backend driver for Gaia needs to only be capable of writing a file to a given path

Writing to Gaia



Reading from Gaia

Reading from Gaia

Reading from Gaia

Using normal URL resolution allows for a multitude of protocols and deployment of best engineering practices (proxy caching, CDNs)

How we built a Gaia Provider

A data provider *may* operate multiple users' gaia stores and we provide a default one for Blockstack users

Gaia service provides a GET/PUT interface to a client

Write control is performed via user's ECDSA key signing (remember, each Blockstack user has an ECDSA key)

Thanks! And Now, Questions

Gaia Source Code: https://github.com/blockstack/gaia

We're looking for:

more deployments!

more backends (send PRs!)