Project Naptcha

NFT Gating as a Service

Basic Description

Naptcha is a B2B Plutus app that makes logins easier and more secure for any website, as well as provide the ability for web apps to make queries to validate NFT ownership, making a variety of actions conditional on NFT ownership other than a simple page load.

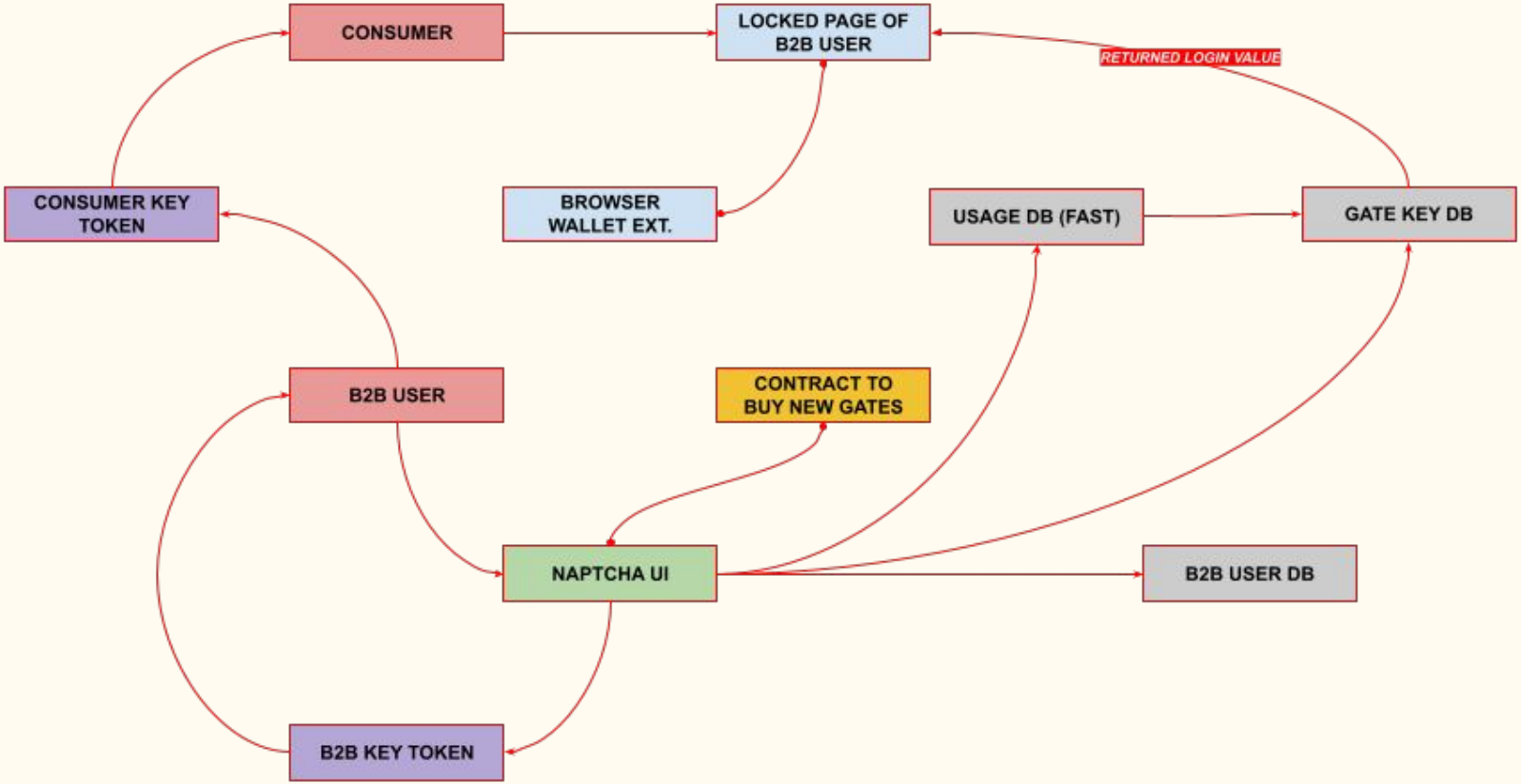
Naptcha login pages require no captchas, no username/password combinations, and have fewer classic attack vectors. They require ownership of specific NFTs to unlock the viewing of private pages, as well as enable other types of function gateways that are conditional on NFT ownership.

Naptcha uses NFTs as a sort of authority token without explicit identity, both to use the business customer's system but also to use the Naptcha system itself, through the use of authority tokens that trigger "root functions" in the smart contract.

While Naptcha is a standalone service, it also contains prerequisite functionalities for a larger project, Dropnir.

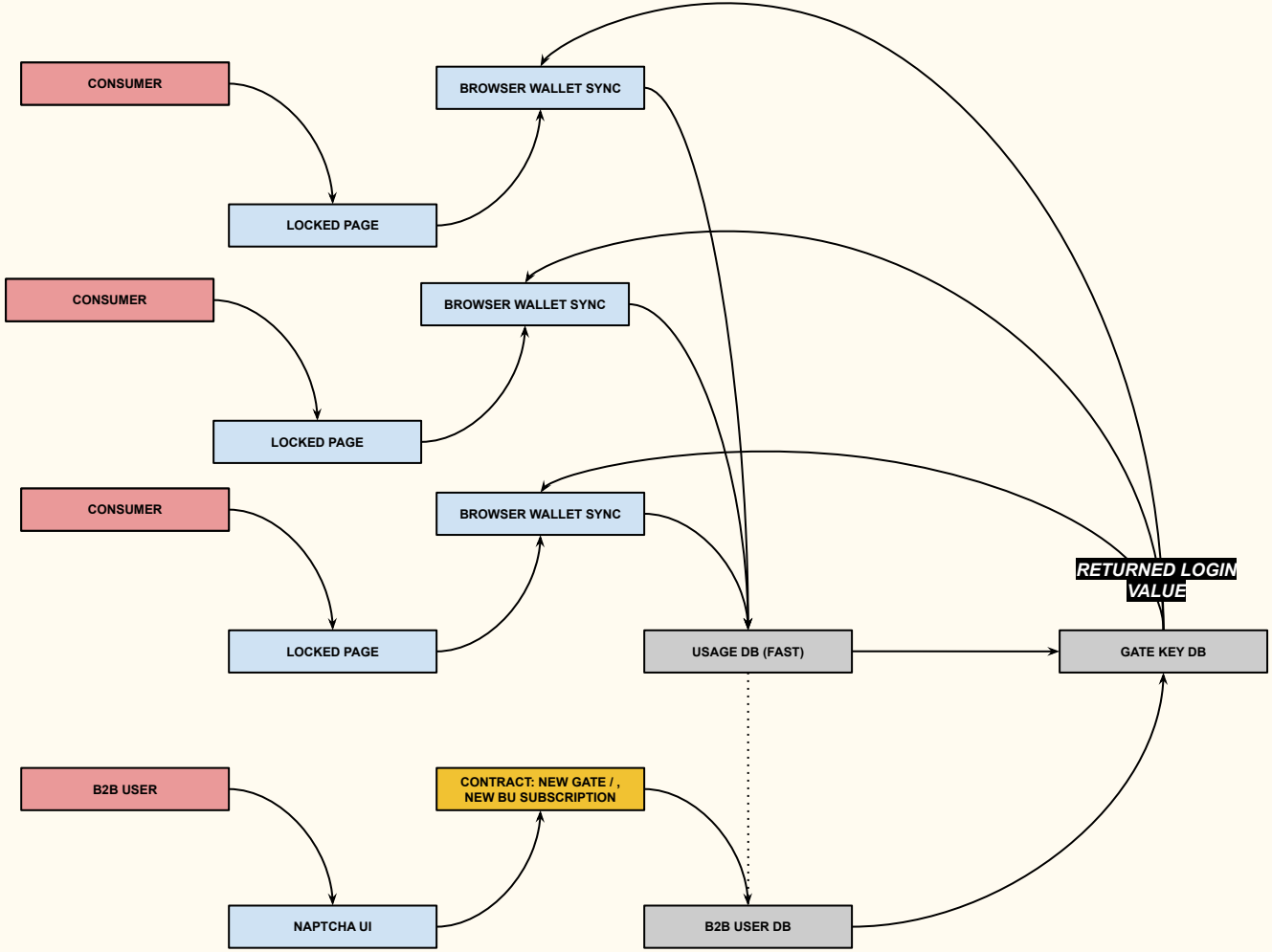
Basic Process

1. Business user (BU) needs to block a page or domain, like a shopping site, subscription site, or anything else, contingent on ownership of an NFT in the place of a username/password hash.
2. BU registers an account on Naptcha, buying a “substriber token”
3. BU buys a custom “gate key” in the Naptcha UI, specifying which NFTs unlock what web page or domain regex, and what string or dataset to return.
4. Naptcha provides a unique Javascript snippet to be pasted on the user’s site.
5. Customers (of the business user) visit the page, triggering the script.
6. The gatekey script leverages a browser wallet extension to authenticate and iterate through all tokens on the customer wallet, off-chain and without fees.
7. Gatekey script sends a database request to the Naptcha Gatekey DB, again off-chain and fee-less, but the DB tracks and caps query usage according to the paid subscription plan of the business user.
8. Gatekey script uses the DB’s return value (such as a profileID) to perform post-login steps as normal (like fetching order history from another DB)


CONSUMER
LOCKED PAGE OF B2B USER
RETURNED LOGIN VALUE
CONSUMER KEY TOKEN
BROWSER WALLET EXT.
USAGE DB (FAST)
GATE KEY DB
B2B USER
CONTRACT TO BUY NEW GATES
NAPTCHA UI
B2B USER DB
B2B KEY TOKEN

Same model, with three BU Consumers and one BU.

The model extrapolates in the same way for multiple BUs and their clients.



Index of Processes

1. Business User
    a. Creating an Account (purchasing BU token)
    b. Buying A Gate Key
    c. Minting An NFT (inlinable to 1.b.i.)
    d. Edit a Gate Key (including deletion)
    e. Update Business Subscription
    f. Browse gatekeys, browse key script snippets
2. BU Consumer
    a. Proceed through gatekey (script runs off-chain DB check)
3. BU App (API Access)
    a. Perform token gating interstitially, within an app created by the BU. Strings or datasets returned by Naptcha queries can do more than log you in; for example, they can help power a game.

Procedural Overview

1. Business User (BU) creates an account on our site (purchase a BU login token)
2. BU clicks menu item “New Gate” to make a new NFT gate expression.
    a. Gate key name (arbitrary)
    b. Gate key code (randomly generated, unique)
    c. NFT regex (one or many NFT IDs)
    d. Target URL, domain, or domain regex
3. BU pays to generate the gate page script.
    a. BU clicks (OK, Submit)
    b. A popup asks the BU to send a fee to a smart contract address to continue
    c. Our website server sends the data to a simple script which only requires payment and valid data
    d. Our website receives payment confirmation, then adds the data to our “NFT-To-Page Map” DB
    e. The new “Gate Key” is added to the BU’s dashboard, in his list of gate keys
4. BU generates his gate key script
    a. BU clicks “Embed Script” button
    b. A popup provides a script for the BU. Copy and paste right after the <body> tag on target page
5. BU Consumer requests the locked page
    a. The script is a compact page blocker that connects to a wallet integrator like Nami. It makes a list of the tokens in that wallet and passes it to the “Speed DB”. If more efficient, make the BU Consumer select one token to try as the “key”.
    b. The Speed DB controls script access by tracking and capping query usage. It first checks if the BU has any query credits left, for example, before passing the request to the Gatekey DB.
    c. The Gatekey DB returns an OK message and a string or dataset pre-programmed by the BU User when making the gatekey. The login page continues as a normal page would, if the username/password were correct. It uses, for example, a profileID of a consumer linked to an NFT, to fetch profile information and order history from its own systems as normal.

BU Website Process: Installing The Script

1. Business User (BU) pastes his script code into any page, after the </head> tag.
2. The script blocks all rendering of the original content, instead displaying a popup.
3. Each script is a little different - they each contain the unique gate key ID. You can't put the same script on a different domain than it was configured for.
4. The popup checks the customer's Yoroi wallet, or requests authentication with it.
5. The script sends an API call to our NFT-To-Page database with the gate key ID and a list of NFT IDs found inside the wallet.
6. The database checks if any of the customer's NFTs match any specific row or regular expression where the same row also matches the gate key ID. The query returns either true or false.
7. Based on the database results, the script reveals the rest of the page, or displays a "rejected" message.

Optional: The service can be subscription-based; if the BU has not paid his monthly fee via our website, access to his gate keys are shut off.

Naptcha UI

List of Functions

- Register: fill an application and purchase a business user auth token
- Login: confirm business user NFT, load user profileID from Naptcha Gatekey DB (same used by customers for custom gatekeys).
    - A second internal Naptcha DB references profileID to fetch the full set of user data
        - Basic user information
        - My Gate Keys list
- New Gatekey: define a new gatekey and pay a fee to add new row to Naptcha Gatekey DB
- Edit Gatekey: update an existing gatekey with new return values or token conditions. Small fee to edit the Gatekey DB. Option to buy query credits linked specifically to that gatekey.
- Update Subscription: redeem an expired business user auth token for another or upgrade your query cap (stored in a Customer DB). Purchase Universal Query Credits (query credits in the Customer DB, available to use by any of his gatekeys).
- Browse Gatekeys: browse your previously made gatekeys. The gatekey scripts for pasting on your desired pages can be copied from here.

Partnership Payment As Smart Contract

The main payment address can itself be a smart contract.

It holds all incoming revenue until the next epoch.

ADA staked on behalf of the script address receives rewards, which triggers payouts to both partner entities, 50/50.

Naptcha Test-Case Project

Hosk Poker

Overview of Test Case App

Normally, Naptcha won't be responsible for anything the business user requires on his website other than the login page, and perhaps additional gate key scripts for his own purposes. To get Naptcha running, it needs at least one customer, so Naptcha design its own customer in the form of a B2B user app, Hosk Poker. Hosk Poker utilizes one new smart contract:

1. In-Game Ante: each turn, send your bet; must be a Hosk Coin policyID

That SC is not related to Naptcha; although we must make it, we are playing the role of a business user with his own game app. As a business customer, the Hosk Poker app uses an on-page script provided by the Naptcha interface and pasted into page code. That script requires a validator to process True before continuing with the game.

The B2B customer makes gate keys that not only specify which NFTs will unblock the page, but will also return a dataset that the business customer requires for his app to work. In this case, it is profile and game configuration data, stored off-chain.

1. Profile data: a profileID synonymous with the consumer token assetID is returned by the GatekeyDB, and it is the business customers' responsibility to use his own internal databases to fetch in-depth profile information from that profileID. This way, game history and other personal details can be loaded into the game UI.
2. Game configuration templates: the poker game is offered in several different modes, such as No Betting or Unlimited, etc. This mode determines which game lobby you are placed in (if game invites are not explicit) and the rules of the game for both the browser UI and the smart contract that handles the game bet.

Index of Build Requirements

1. Web app game environment
2. SC: Game Entry TX (Hoskinson ownership > 1, new game fee)
3. Poker Game
    a. Rules Configuration (1 schema per game mode)
    b. Pool.Pm card image randomizer
    c. Ante-up within game (inlined Game Entry SC)
4. SC: Poker Server Endgame TX
5. Data:
    a. Hoskinson Business User Auth Token
    b. Gatekey: Game Entry
        i. Attach to game-in-waiting
            1. Return YES - connect to pending game
            2.
        ii. Create new game instance
            1. Return dataset: game configuration template for game mode chosen
    c. Gatekey: Game Ante-Up

Game Lobby Procedures

Ideas...

- New game: you must specify exactly which people you want to play with.
- Neverending game contracts: come in and out of the poker table as you please.
- Database of unstarted games is kept off-chain, automatically deleted if un-filled within a certain time.
- Discord bot: sends direct messages to players being invited to a game, with a link back to the game.
- Game page: integrates with a wallet app to identify you
    - Shows you open games you have been invited to, and their status (X players left to join)
    - Shows a list of users you have added to Favorites
    - Start New Game: pay a small fee, select users from Favorites or full list search bar

Beta Testing Environment

To guarantee that the cost of bandwidth stays low, the game will require ownership of a specific cNFT, perhaps the Hosk Donkey, because only 100 were minted.

- Small, cheap server architecture
- Gatekey: Must own a Hosk Donkey
- Ideally CDNs in North America and EU

Naptcha Test Customer: Hosk Poker