nudge

Artificial Intelligence for your Organization, Business and Personal Life Interconnected in the Blockchain by a Universal Global Marketplace, Language-Aware Database and Application Development Platform

July 11, 2018 (version 0.5.0 - draft 1.01)

San Jose, California
whitepaper@nudge.io

Preface

The internet needs more intelligence. As the world continues moving to an economy where everything is done online, it becomes harder and harder to keep track of what bills to pay, what appointments are happening when, and what I should be aware of. This is only compounded by the fact that the information and digital events happening in our lives are in disparate, ad hoc data stores spread out across the internet without a common language describing what each datapoint means. Because of this, making all of this data work together is impossible. The end result is a Frankenstein of data transformations and one-off integrations to get certain features, but we can never get an overview of what we need to know and when. X service may work with Y service but Y service may not work with Z service. So in the end we are still checking a bunch of sites to get what we need to know. This is a problem in our personal lives and our business lives. We need a better way. We need a common language. We need a technology that can make sense of all of this on the fly.

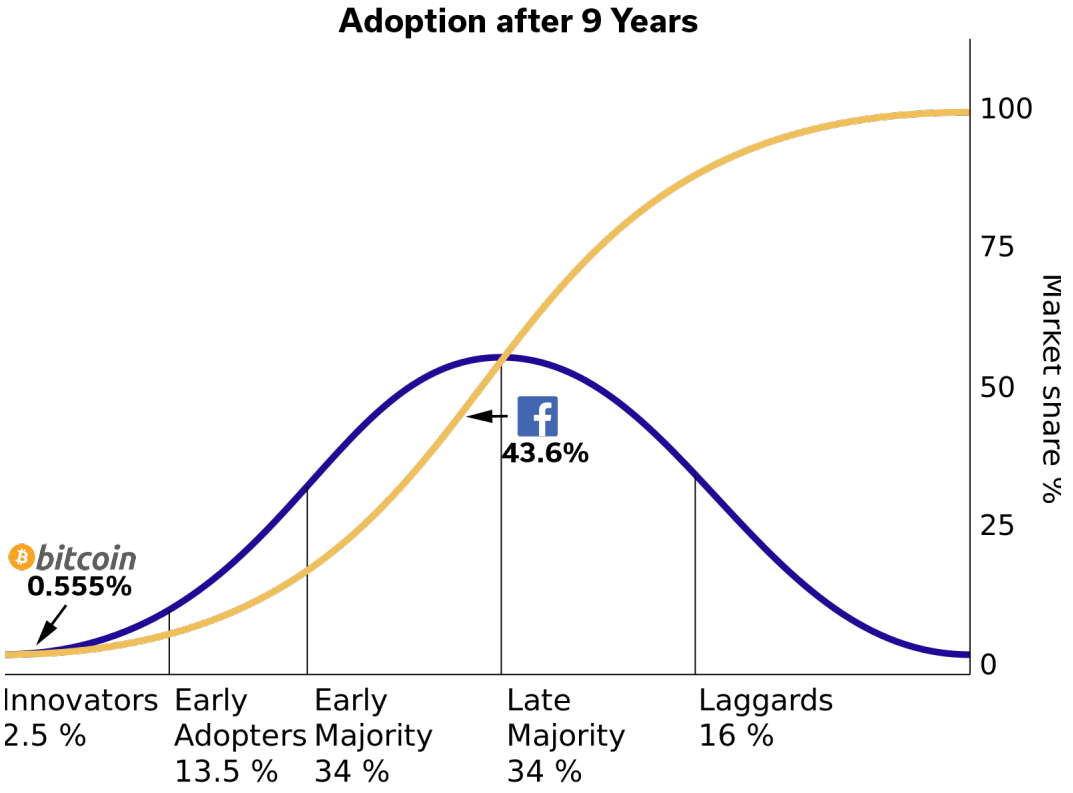
In many ways, cryptocurrencies offer a fresh start to modernize the internet. Companies will be spending the time and money to integrate, and if the technology is easy and cheap enough, and this is an opportunity for companies to offer more to their customers. Its also an opportunity for the creators of the technology to make the internet of cryptocurrencies better than the internet of fiat. Even more so, for cryptocurrencies to catch on with the masses and stay integrated with companies, this has to be true.

In the more than nine years since Bitcoin’s initial release the big companies have either not integrated or flip-flopped on integration. Lately, theres been an explosion of media coverage and more people, companies, investment firms, and even governments taking a hard look at it. In that time and even with all the current coverage, there hasn’t actually been as much adoption as one would expect - especially in comparison to the hypergrowth seen in the startup world.

Consumer Adoption of Bitcoin has Been Slow

Consider how Bitcoin's growth compares to Facebook's growth over its first nine years. As of June 2018 there are 22.2 million active Bitcoin accounts and 4 billion total internet users worldwide. Even assuming every Bitcoin account is a unique user, less than 1% (0.555%) of internet users have an active Bitcoin account.

Facebooks numbers in comparison after 9 years are staggering. In October 2013, Facebook had 1.19 billion active users - 43.6% of the internets 2.73 billion users at that time.



BLUE: **SEGMENTED ADOPTION RATE**
YELLOW: **CUMULATIVE MARKET SHARE**

Why is growth so slow?

While Facebook complements the existing internet infrastructure, Bitcoin and other cryptocurrencies are a compelling beginning to an *alternative* infrastructure. This alternative infrastructure is still missing a lot of essential services though. Because of this, theres no “must do” user experience for the average internet user. Theres nothing driving them from first hearing about cryptocurrencies, through the initial learning curve, and into becoming an active user of the ecosystem.

The most apparent missing need is finding ways to spend or earn it - a marketplace. This is what we’re building.

Users want this. Google “where to spend cryptocurrency” and you’ll see many results trying to help users along. Several currencies without any other option, have even created dedicated subreddits to deal with this problem. For example Dogecoin has the subreddit /r/dogemarket where users buy and sell items as well as DOGE. As you can imagine on a subreddit, everything is done in the comments - sales, bargaining, buyer/seller ratings, everything. Its very impressive what has managed to happen there. However, scammers continue to be a problem, searching for items is dependent on the underlying reddit search (which isn’t really that good), and because its only a comment system theres a lot of issues that just can’t be fixed. This and the other options out there are just not a compelling alternative to the existing internet infrastructure.

There needs to be a better solution. We have the opportunity to not only build this missing infrastructure, but also by taking advantage of the unique and significant technological advances we’ve made at Nudge, make the missing infrastructure incredible.

What is Nudge?

Nudge is the foundation of an Intelligence Enabled Internet
At its core, Nudge is a database that self-learns, in real time, about the content stored within it. It's intelligence comes in part, from *new algorithms unique to Nudge*, that enable it to find insights within its datasets without intervention from the user. Alongside, a continuously learning, natural language model enables data stored within Nudge to be implicitly human queryable, searchable, and return significantly more value to the user than a traditional database. Nudge is also what DARPA would describe as a third wave machine learning technology (https://www.youtube.com/watch?v=-O01G3tSYpU).

Last year we were accepted into Startup School by YCombinator. For more information and a short presentation, from last year, (in an earlier stage - June 2017), visit our Startup School page at https://www.startupschool.org/presentations/740 .

We are building an Intelligent Marketplace Powered by Cryptocurrency
On top of this foundation layer we are building a blockchain enabled marketplace. The Nudge Utility Token (NDG) will be used settle transactions in the marketplace as well as pay for storing objects in the distribute database layer.

The marketplace will enable technical and non-technical users to offer products and services (for example: SaaS, IoT, home dry cleaning pickup/delivery, appliance repair, plumbing, photography, news, events, classified listings, etc) in exchange for cryptocurrency.

This will also be an ideal place for companies or projects who have completed ICOs to fulfill bounties and hire consultants.

The marketplace vision includes providing an open protocol for standardizing product and services exchange for cryptocurrencies on the internet. This protocol will account for other currencies to be exchanged on-the-fly to NDG in order to fulfill transactions.

Under the hood, the NDG utility token will provide value as a unifying currency layer for all services and applications connected to Nudge. This means Internet of Things, services, and providers can seamlessly, autonomously or non-autonomously, interact and complete transactions without differing currencies being an obstacle.

Anything build within the database layer will be able to be monetized via the marketplace.

This is the way forward
Nudge will provide a significantly better way for data on the internet to be delivered, used, and combined. We are building a platform, where products and services that aren't even aware of each other, will be able to collaborate within a unified workflow, and provide more value to the user than they would be able to do so independently.

Every object within the database layer is intelligent, language aware, and able to support code execution via WebAssembly. In addition we are able to granularly grant and revoke read and/or write access to users and groups.

Permissioning data privacy levels in the distributed/blockchain database space has been a particularly tedious problem. Many businesses have been wary of storing customer information or proprietary information in a distributed or blockchain setting as these include variables and maybe methods that should not be accessible by just anyone.

Within this paper, we provide a concrete solution to this problem.
Nodes storing protected data will not be able to see the contents of protected data. At the same time, those supposed to have access, will, and those who have access revoked will see their privilege disappear immediately. Access to objects, variables, and methods from different users and groups can be granted and revoked seamlessly.

This will make distributed services and applications possible, that previously had to be built using centralization. Smart contracts will now be able to be database enabled. *Furthermore, data stored in this way will be GDPR compliant.*

Founding Research
The core development of Nudge has been the result of a significant research effort to make machine learning easy to use, data simultaneously human and machine understandable, and the learned result compelling even with small data sets (small data learning). Small data learning can be intuitive, in the sense, that a human child doesn't need to see billions of examples of what a cat is in order to learn what a cat is. Small data learning is key to bootstrapping value into your application quickly, as not everyone has access to google-level amounts of data for their application, and in many cases google-level amounts of data might not even exist. These are benefits that applications built on Nudge will inherit.

Prior to this development, existing mainstream algorithms (neural networks, etc), had been complex and brittle, and required an expert-level understanding in order to adequately tune and debug (not to mention black magic trial and error of learning functions to fit data). They also required implementers to have a

comprehensive understanding of the application's domain-specific language and how significant each data feature within an application is - something that is typically only understood by existing employees or someone very familiar with the application's industry. Because of this, maintenance costs associated with retaining/hiring/training such an expert have been high and many companies have put machine learning projects on hold or have had problems delivering compelling value due to difficult to determine ROI.

Nudge simplifies this - not just by hiding the gritty details. New learning algorithms, cycle detection algorithms, and others have been created, from the ground up, to not have these problems in the first place. So companies, organizations, and individuals can have confidence pushing their products, services, and applications forward with machine learning and seamlessly monetize them with cryptocurrencies.

Standard Disclaimer

This Whitepaper has been issued by nudge.io (the “Company”) and should be read in conjunction with the Company’s terms and conditions (the “Terms”).

The purpose of this Whitepaper is to provide prospective purchasers with the information on the Company’s project to allow the prospective purchasers to make their own decision as to whether or not they wish to proceed to purchase an NDG token. This Whitepaper does not constitute an offer or invitation, sale or purchase of shares, securities or any of the assets of the Company.

As of the date of this Whitepaper, the information contained herein is accurate to the best of the management team’s knowledge and there are no other facts of omission, which would make any misleading statements in this Whitepaper. No representation, warranty, assurance or undertaking is made as to its continued accuracy after such date. The information contained in this Whitepaper may be subject to modification, supplementation and amendment at any time moving forward and would be documented accordingly. In addition new information might be added in the future.

This Whitepaper describes the Company’s business objectives and the issue by the Company of NDG tokens. It has not been reviewed, verified, approved or authorized by any regulatory or supervisory authority.

This Whitepaper does not constitute an offer or solicitation to anyone in any jurisdiction in which such offer or solicitation is not lawful or in which the person making such offer or solicitation is not qualified to do so.

Prospective purchasers of NDG tokens should inform themselves as to the legal requirements and consequences of purchasing, holding and disposing of NDG tokens and any applicable exchange control regulations and taxes in the countries of their respective citizenship, residence and/or domicile.

Prospective purchasers of NDG tokens are wholly responsible for ensuring that all aspects of this Whitepaper and the Terms are acceptable to them. The purchase of NDG tokens may involve special risks that could lead to a loss of a substantial portion or a loss of the entire purchase amount. The purchase of NDG tokens is considered speculative in nature and it involves a high degree of risk. The Company does not represent, warrant, undertake or assure that the NDG tokens are defect free or will meet any specific requirements of a prospective purchaser. You should only purchase NDG tokens if you can afford a complete loss. Unless you fully understand and accept the nature of and the potential risks inherent in the purchase of NDG tokens, you should not purchase NDG tokens.

The purchase of NDG tokens is only possible after the prospective purchaser has read, understood and accepted the Terms. Each prospective purchaser will be required to

acknowledge that they made an independent decision to purchase the NDG tokens and that they are not relying, in any manner whatsoever, on the Company, the management team or any other person or entity (other than such purchaser's own advisers). Prospective purchasers are urged to consult their own legal, tax or other advisor before purchasing NDG tokens.

The Company and the management team do not provide any advice or recommendations with respect to the NDG tokens, endorse such tokens, nor do they accept any responsibility or liability for any use of this Whitepaper by any person which is in breach of any local regulatory requirements with regard to the distribution of this Whitepaper or any applicable rules pertaining to the offer of NDG tokens.

To the maximum extent permitted by the applicable laws, regulations and rules, the Company, its founders, team members and any third parties involved in the Company's project shall not be liable for any indirect, special, incidental, consequential or other losses of any kind. Furthermore, in tort, contract or otherwise (including but not limited to loss of revenue, income or profits and loss of use or data), arising out of or in connection with any acceptance of or reliance on this Whitepaper.

All statements regarding the Company's financial position, business strategies, plans and prospects of the industry which the Company is in, are forward looking statements. Neither the Company, its founders, team members or any third parties involved in the Company's project nor any other persons represent, warrant, undertake that the actual future results, performance or achievements of the Company will be as discussed in these forward looking statements.

This Whitepaper includes market and industry information and forecasts, which the Company obtained from internal surveys, reports and studies where appropriate, as well as market research, publicly available information and industry publications. Such surveys, reports, studies, market research, publicly available information and publications state that the information that they contain has come from sources deemed reliable; there is no assurance as to the accuracy or completeness of such included information.

The Company does not make or purport to make any disclaims, any representation, warranty or undertaking in any form whatsoever to any entity or person. Including any representation, warranty or undertaking about the truth, accuracy, and completeness of any of the information set out in this Whitepaper.

Statements made in this Whitepaper are based on the law and practice currently in force in California, which is a state within the United States of America and are subject to changes in accordance with said laws.

Table of Content

Preface 2
Consumer Adoption of Bitcoin has Been Slow 3
What is Nudge? 5
Standard Disclaimer 8
The Nudge Intelligent Marketplace 11
Version 0.5 - Initial Milestone Release 11
Language Aware Data Structures 11
Two Factor Purchasing (2FP) 12
Placing an Order with Invoice Signatures 13
Fulfilling an Order 15
Predictive Escrow 15
Contextual Trust Ratings 15
Credibility Scoring - Accounting for Ratings Manipulation 17
The User Interface 18
The Nudge Intelligent Database 20
Tick-Tock Bidirectional Chain Hashing 20
Objects 22
Ensuring Trust in a Trust-less Environment 25
Hardened Threshold Consensus 27
Key Management 29
Putting It All Together 32
Separation of Concerns 35
Node Incentives 36
Business Model 38
More than a Just a Business Model 40
Competitive Landscape 41
The Nudge Utility Token (NDG) 42
Token Supply Distribution 43
Budget Allocation 44
About Nudge 46

The Nudge Intelligent Marketplace

The vision of the Nudge Intelligent Marketplace is a fully decentralized marketplace and protocol. Through the use of portable, language aware data structures and a standardized deployment model we will make it easy for buyers and sellers to extend the marketplace to their needs and offerings.

We intend to release the marketplace components in stages in order to fill the major gaps in the cryptocurrency industry as quickly as possible. This means that on the way to full distribution, the architecture will be a hybrid of centralized and decentralized components. Components will continue to be decentralized as new versions are released and adoption continues.

Transactions within the marketplace (and the database) will be priced in NDG. However, we will also be prioritizing fiat currency integration. This will have the effect of being exchanged into NDG for order settlement.

Version 0.5 - Initial Milestone Release

The first milestone release version will seek to be the best available marketplace for cryptocurrencies. We are aiming for a visually appealing, intuitive website to enable buyers and sellers to reliably and securely exchange goods and services (digital or real) for cryptocurrency, an iOS and Android app to enable what we are calling 'two-factor purchasing', and a foundation to set the stage for future releases. Its important to note that even though this release will have many compelling features, this is still a stepping stone to the longer term release vision.

Language Aware Data Structures

We want people to use the marketplace not just because its cryptocurrency enabled, but also because its the best way to get things done and find what they are looking for.

Data within the marketplace will be object-oriented, language aware, and implicitly searchable and intelligent.

By 'language aware', we mean that language is tied directly into the objects, variables, and methods we create to build our products/services and use in the marketplace to find and interact with our products/services.

Objects are implicitly considered nouns and have real semantics and context enabled on the by the ML(machine learning) and NLP(Natural Language Processing) systems. So an object of type "Order" will not be just a datatype with a simple label. The label will carry deep meaning that this object is an "Order" object defined as "a request to be made, supplied, or served".

Objects can contain language aware variables.

They can also contain language-aware verb methods. Verb methods are actions tied into the natural language system that can be performed by the user. Intuitive examples of verbs that might be assigned to a method are “cancel”, “search”, “call”, “send”, “get”, “order”. Verb methods can return nothing or an object.

By building objects this way, vendors can create a pretty sophisticated experience within the marketplace.

In a hypothetical use case where a vendor offers a food delivery service, signing up (or purchasing a membership if there is a fee to sign up), would result in the purchaser receiving an “membership” object.

This “membership” object might contain a “place order” verb method that requests food for delivery and returns a resulting “order” object. The “order” object in this case could implement a user interface for the delivery we have just requested. It could contain verb methods like “cancel” to cancel the order, and “track” to find out where the users order is.

Tying the data and language together like this gives us many advantages. It makes the data values and the intent of the data self-documenting. It opens up the possibility of a common graphical interface, a common natural language interface, and a common search capability across many different vendors and systems who are not even aware of each other.

When combined with the recursive ML/NLP features present in Nudge Database, language aware objects make SIRI-like functionality a possibility across the marketplace, applications, reporting, and even organizations.

In the “Objects” section of this whitepaper you will see more concrete examples of how objects are defined.

Two Factor Purchasing (2FP)

Those familiar with and using two-factor authentication should immediately find the value in two-factor purchasing. Nudge will use an iOS/Android app to implement two-factor purchasing. Two-Factor Purchasing works as follows.

Every user of Nudge Marketplace has a *multi-signature* NDG cryptocurrency account. This is the account a users payments will be made from when buying in the marketplace. Our hope is to have account creation done entirely from within the iOS/Android app, so there is no account exposure external to user owned devices.

When the account is created, the user is given the master key. This is something that will not be retained on the device. The user should write this down and save it in a safe place. This is essentially the admin key to the account and the marketplace will never see this key so long as it is created using the Nudge App.

In addition to the master key two additional signers are given access to the account. A 2FP key will be created with a signing weight of 2 and the known marketplace public key will be added with a signing weight of 1. It will take a signature threshold weight of 3 in order to send funds from an account.

What this translates to is that in order to send funds from the account you will need a signature from both the 2FP key and the marketplace key as the sum of the weights of these would add up to 3 (2FP key weight of 2 + Marketplace key weight of 1). Alternatively you could just use the master key by itself, but for the purpose of interacting with the marketplace with 2FP this key will not be used.

Now, whenever a financial transaction in the marketplace requires approval, a popup notification will be received on the users device. They can then touch through to the app, review the fund request and approve it. The app will sign the transaction for the user and the marketplace will then sign for the marketplace.

The advantages of using this method is that it requires a simultaneous compromise of both the marketplace and the user's device to be compromised at the same time in order for funds to be lost. This drastically reduces the probability of an account being hacked and furthermore minimizes the loss to compromised user devices.

Additionally in the event of a compromise, damage can be easily mitigated similar to issuing a new credit card number. Approved keys can be updated on the users account via the master key.

Placing an Order with Invoice Signatures

Every account on the marketplace requires at least two keysets, a user's marketplace key and a user's payment key. The user's marketplace key is used to sign transactions on the marketplace, some of which may not require payment. While the user's payment key is solely used for authorizing a payment.

These are kept separate as the lifecycles of each can be very different. As well, in the future, we would like users to be able to select which payment account they would like to use (for example a business account, personal account, etc).

When a user decides they are ready to purchase and submits their shopping cart, an itemized invoice is created containing a unique invoice id, the user's marketplace public key, the user's payment public key, the products they want, the agreed prices, and the vendors they are coming from and the user's payment public key. The entire

invoice is the signed with the user's marketplace private key to assert that they have indeed requested the items. The invoice is then submitted to the marketplace.

The marketplace will receive the invoice and validate it's signature matches the invoice's marketplace public key. The account balance for the payment public key is checked non-authoritatively to estimate there are enough funds to cover the transaction (Note, this account balance check is really just a courtesy to the user at this point in the transaction as no funds have exchanged and there is no guarantee to the marketplace that funds will be there for the closing of the transaction yet.)

The marketplace will also notify vendors of an "intent to purchase". The marketplace will not send the entire invoice to vendors to review but only the details relevant to them. This allows vendors to ensure that they are able to fulfill the quantities at the indicated pricing for the items they are responsible for in the invoice. If they are able, they will sign their sections and retain the "intent to purchase" in a priority list for those items - effectively temporarily reserving those items. Note that this is a soft reserve as the transaction has not actually closed yet. The signed sections are returned back to the marketplace.

Assuming everything checks out at this point, the marketplace will integrate the newly signed sections and respond back to the user with an invoice-escrow contract pre-signed by the marketplace's private key. If things don't check out, an error message will be returned instead.

The invoice-escrow contract contacts two sections. One is a copy of the original invoice containing items to be purchased. The other contains a contract that needs to be fulfilled in cryptocurrency in order for the transaction to continue. Its this invoice escrow that will trigger a notification to the user's two-factors purchasing (2FP) app. The app will validate the signature and format of the invoice. Its important that the format match what the app is expecting to see. There should be no surprise operations on the escrow section and the escrow section should match predefined templates. If valid, the user to review what they are purchasing, and approve/deny the transaction.

Upon approving the transaction, the app will sign the escrow section with the user's 2FP private key, and sign the entire invoice-escrow contract with the user's marketplace key. This will then be submitted back to the marketplace which will sign the escrow section with their private payment key, sign the entire invoice-escrow contract and then execute it. The marketplace will now notify the vendors of the new "client escrow processing" state for the invoice and the payment transaction id waiting to complete.

Upon completion of the escrow payment transaction, the marketplace will notify vendors of the new "client escrow complete" state. Vendors need to confirm this receipt by signing this notification and returning it. This notifies vendors they should ship/activate at this point and keeps a record that they have received the notification.

Fulfilling an Order

Once an order is fulfilled, the vendor will create a fulfillment object or objects and attach it to the invoice. Every object/variable within Nudge has an address. In this case attaching a fulfillment to an invoice would be appending a fulfillment owned by the vendor to a list in the invoice object owned by the user. In this way the vendor can manage updates to the fulfillment and the user is able to access them with permissioning though the invoice.

The fulfillment objects serve as a hook into the vendors product or service. Functionality the service provides should be enabled as verb methods within this object.

Permissions and access control are covered in more detail further in this document. However within the marketplace, permissions are automatically provisioned for this fulfillment sent to the user.

Predictive Escrow

New vendors will have funds held in escrow for 30 days (the default return period). As a vendor proves to be trustful, the amount of time in escrow will slowly decrease, eventually becoming an immediate transaction.

By statistically tracking valid issues and complaints, we can confidently determine how much financial intervention is required for a particular vendor, monthly (automated or otherwise).

This number, along with other features will be used to determine a minimum amount of ‘predictive escrow’ needed on average to ensure happy customers.

Vendors that want to see their purchase escrow times reduced will be required to keep funds equivalent to the calculated predictive escrow. We are also considering requiring an ‘initial predictive escrow’ amount for new accounts to be activated.

Contextual Trust Ratings

A major limitation with existing marketplace options is the lack of available buyer/seller trust information. This usually leads to a very uneasy first transaction amongst buyer/seller pairs as both sides try to work out if the other is a scammer. While much of this uneasiness can be mitigated with the use of escrow, Contextual Trust Ratings will inform the entire lifecycle of a transaction from pre-sale to post-sale (including escrow) and ensure honest participants are both recognized and rewarded. Vendors will want to pay particular attention to their own ratings as these will influence search rankings with the marketplace.

Contextual Trust Ratings indicate the risk of performing a transaction with a target identity from the source identities perspective.

So for example, a buyer will want to know the risk of purchasing from a seller. The seller will also want to know the risk of selling to a particular buyer. The risk level from each perspective can be drastically different. If the seller is established with a high general trust rating and the buyer is a brand new user, intuitively the risk for the buyer would be low. However, for the seller, the risk would be much higher as the buyer being new has no trust history.

These ratings will be derived from a feedback loop that continually combines prior rating results with new results. The weight of each contributing feature will be learned and then the output counterbalanced to ensure those attempting to manipulate ratings will be washed out.

These ratings will eventually be used to inform more than just buy/sell transactions. However, initially we will be bootstrapping them to this use case.

Buyer/Seller Transaction Context
In order to generate a score result, calculations must be derived for the buyer, seller, and the transaction:

- Buyer Trust Segment - Generated from past transaction history and accounts for any complications arising from past transactions. For example an unusually amount of purchases that result in a return where the returned product was damaged.

- Seller Trust Segment - Generated from past transaction history and includes a multitude of features including user satisfaction consistency, product delivery speed, question response speed.

- Transaction Trust - Generated from the result of the transaction (success, fail, why)

All components of each segment will have learned significance weights multiplied in to normalized the feature for meaning. In addition components that have a human element will have the addition of a Credibility Score multiplied in (explained below).

Product Review Context
For users this will appear as your typical 0 to 5 star rating as anything else will lead to user confusion when this is effective and is clearly the marketplace standard at this point. Leaving reviews will not have an effect on a users trust rating. While reviews will be recorded how a user leaves them, the credibility of a review will also be calculated in order to account for ratings manipulation (explained below). Reviews with low credibility will not be used in search rankings and will influence how they are presented as well.

Credibility Scoring - Accounting for Ratings Manipulation

There's always temptation within a marketplace to manipulate ratings, as higher ratings can lead to higher sales. We want our ratings to be as honest as possible. In order to accomplish this every feature that has a human component that can be manipulated will be normalized by a credibility score.

Credibility is a combination of a User's Trust Ratings and their level of authority on a given topic normalized to 1. A user's authority boils down to how well rounded a user is and the depth of interaction they have on a given topic. We believe a user is the aggregation of the 3-5 distinct interests they demonstrate the most at a given time (short term credibility), and over the course of time (long term credibility). What Nudge identifies as an interest is entirely learned, and is essentially determined by hotspots generated across all user purchase patterns.

By determining where a user fits within these hotspots, we can determine their general authenticity (whether this is a legitimate user) and their Topical Authenticity (the depth of knowledge a user has on a particular topic).

General Authenticity is calculated by determining the rank of a user within their top 5 hotspots both short and long term, combining the two sets, and truncating again to the five top elements. Overlapping short and long term rankings are adding together so that users that show authority over the short and long term have their resulting score shifted upwards. Users with less than 5 hotspot matches will have the missing match fields set to 0. The final vector of 5 numbers is summed and then divided by 10. This yields a general authenticity score. We can see that this number demonstrates is we are dealing with a bot, short term throwaway user, or a primary user account. Bots will have a lower ranking within their top five hotspots. Many bots will not even have 5 significant interests. This will yield them generally low scores. Short term throwaway users will miss out on long term scoring in addition to not having 5 ranking interests. Real primary users however will see their credibility continue to go up as they interact with the system. With new ranking interests generating more credibility.

Topical Authenticity is calculated by multiplying a users general authenticity by the general ranking of the user for the specific topic:

*((topic short ranking + topic long ranking) /2) * general authenticity.*

A Users Credibility Score for a feature is:

*(Users Trust Score * User's Topic Authenticity for a particular feature)*

Sales Tax

It will be up to vendors to handle sales tax as required by the laws of their country/city/etc. An automatic sales tax calculator is something we are considering integrating in a future release.

The User Interface

The long term goal of the marketplace is an intelligent, open, common user interface and dashboard built from the products and services a user has purchased or has access to. The interface will not just enable users to buy products and services with cryptocurrency. Users will be able to get an overview of what they need to do, what they want to do, and what is relevant to them at the current moment. We will integrate a users services and products in a way that they complement each other and work together regardless of the vendor.

Nudge's Intelligent Machine Learning is one of the fundamental technologies that make this possible. This enables us to find insights within a user's work and product flow and make those insights apparent to the user.

The first steps in building this is releasing a common web interface, search, and natural language interface. We will follow this up with a common interface for mobile which will be an update to the application running two factor purchasing. We will also release APIs that allow others to interface into the system as well. These APIs will give users access to the search and natural language interface as well as access to the verb methods tied to the objects they own.

The initial user interface release will make use of some centralized infrastructure. Future releases will have much of this functionality distributed through the use of Execution Nodes.

Execution Nodes are distributed server nodes that are given the authority to execute code and access data on behalf of a provisioned database user. Vendors and application developers will be able to permission these nodes to have access to databases and execute object verb methods. These nodes will also have the capability of enabling *smart contracts* backed by the full capability of an object-relational database instead of the more simple smart contracts available today.

Execution Nodes enable a fault tolerant distributed execution environment. Many applications on the internet would benefit from transitioning to this type of deployment model. The code run with these nodes is compiled to WebAssembly.

Users who want more control over their execution nodes will be able to assign their code execution to run only on nodes within their environment or datacenter. This will enable existing services or services not suitable to WebAssembly to still be first class

citizens within the Nudge Platform. In this case some code would run within the execution node and other code would be called into locally via service calls.

The Vendor Interface

Many platform users will want to get started quickly selling their services and products. They will not want to spend time writing or even learning to write code. This is perfectly fine and will be supported via the vendor interface.

The vendor interface will enable users to define objects, products, pricing, and fulfillments entirely from within the GUI. This will manage class definitions and object creation within their database automatically. And if they decide they need to dig into the code and do something more advanced in the future they will still be able to.

If you want to become a vendor, the only requirement is you have an active database to store fulfillment and product objects within. We will also provide a simple way to create a database for non-technical users as well.

Vendors within the system will bumped up the search listings for their products base upon their trust rating. One easy way to increase this trust rating will be to get verified. Periodically we will also allow high trust users to allow issue verifications as well. We will explain more about the verification process as we get closer to launch.

The Nudge Intelligent Database

All applications built on Nudge exist within the blockchain database. This includes the marketplace. While we've covered some features of the Marketplace Layer, they need a foundational technology to sit on. This section covers that in more detail.

Database Blockchaining

All objects within Nudge are stored within the concept of a database. All databases within Nudge are attached to a parent database. There is no limit to the depth at which you can append child databases to parent databases. Child databases inherit default permissions and cryptographic settings from their parents. In many ways its appropriate to imagine this structure like the directory structure of a file system with the "root folder" being the root database created in the genesis block of Nudge. All databases will exist as a child somewhere under the root database and in many cases use another database under the database creators control as their parent.

For example within the Nudge Marketplace, every user has a home database under which there is an child database in which a user's entitlements/purchases/subscriptions/etc are stored. This allows for a very adaptable storage environment for adding data organization as needed. In another example, consider an 'Internet of Things' use case where every device exists within a user's common devices database. Every device would be able to implement its own self contained child database for its needs within the parent devices database.

Another helpful way of thinking about child databases is as if each database is its own micro service. In this way we can keep an organized separation of concerns that allow us to isolate datasets to the minimal amount of users needing to know about that data.

Architecting databases in this hierarchical way allows us to distribute data within a database without major concern for what is happening in other databases. It also allows us to implement a chain of accountability and tamper-evident logging without needing to store a complete copy of all existing databases at every node. This is done in a way we are calling "Tick-Tock Bidirectional Chain Hashing".

Tick-Tock Bidirectional Chain Hashing

Blocks in most blockchain implementations are concatenated in a list. Within Nudge, block growth occurs much more like a tree. Every database has its own chain. When blocks are consolidated and finalized within each database, they notify their parent and child databases of their hash. This hash is retained within these external databases and eventually integrated into their own hash during their finalization.

This procedure “Ticks and Tocks” back and forth between database relatives, keeping a check-in of where each database is. The end result is a tamper-evident log that asserts nothing in the block has been illicitly modified.

Nudge databases are typically journaled and retain an audit trail of the change transactions occurring on objects and the corresponding user who initiated the changes.

Non-Journaled databases that behave more like your traditional database with only the current object state being able to be recalled are also being considered. This would work by have the database creator enable a garbage collection procedure that would delete data not relevant to the current state. This would need to ensure that the audit-ability of the database is still retained. So any automated garbage collection would need a consensus across database nodes. This is not something being prioritized for the early release version.

Objects

All objects within Nudge have a globally unique address. They are capable of containing data and other objects. They have cryptographic permissions restricting who can see the objects, modify the objects, or execute calls on the objects. Its this conceptual layer that the marketplace will be built on top of.

Defining an object
Those familiar with python will see a lot of inspiration between defining an object with Nudge and defining an object with python. As Nudge is object-oriented, defining a class is the first step in defining an object.

For example if you are defining a simple User class within your database, you might define it as follows:

```
class User ‘user.n.01’ (object):
    text ‘first_name.n.01’ first_name
    text ‘last_name.n.01’ last_name
```

Note the quotes labels after User and text. These reference a continually learning language model and dictionary in Nudge that translates semantics to meaning. In this case, we are telling Nudge that class User is of a semantically defined type ‘user.n.01’ and that a User class has a first_name and last_name text variable that also have semantically defined definitions of ‘first_name.n.01’ and ‘last_name.n.01’. Note that unlike python we do not have to use an “__init__” method to setup instance variables. All variables within Nudge are instance variables, although there is a way to create a static-like behavior by referencing an object instance directly into the class.

For example if I had a previously defined class SharedCalendar and I wanted every user to use the same shared calendar I might modify the User class to look like:

```
class User ‘user.n.01’ (object):
    text ‘first_name.n.01’ first_name
    text ‘last_name.n.01’ last_name
    SharedCalendar * shared_calendar = @“$abc123”
```

There are several things worth pointing out in this example. The line defined the SharedCalendar variable has an asterisk(*) for its semantic definition. The use of this wildcard tells nudge to use what ever semantic definition is supplied by the SharedCalandar class and not to add any additional definitions (Its not a pointer for those C and C++ inclined.).

The other thing to notice is the @“$abc123”. The @ sign has a special role to translate what is after it to a target object. In this case, the ‘$’ is a shortcut to translate to a target object by object id. Values assigned in this way are not stored in a per instance

fashion. So changing this value will not do O(n) updates where n is the number of User instances already created. This update will just be an O(1) journaled change to the class definition.

Methods
Methods can also be defined in a similar format as python, but with the addition of semantic awareness.

```
class User ‘user.n.01’ (object):
    text ‘first_name.n.01’ first_name
    text ‘last_name.n.01’ last_name
    SharedCalendar * shared_calendar = @“$abc123”

    def getNews ‘get.v.01 news.n.01’ (self):
        return wasm.exec(“get_news”, self)
```

WebAssembly can also be applied if needed for more advanced functionality.

Semantic Inheritance
There are several ways to define semantic inheritance. These are applicable regardless of whether you are applying it to a class, variable or method.

*	no additional semantics. Use what is inherited from parent element. Example: text * var_name
!	no semantics at all. This element will not be semantically queryable. Example: text ! var_name
!’key.n.01’	ignore inherited semantics and override with the quoted value Example: text !’key.n.01’ var_name
‘key.n.01’	The default behavior. Inherit parent semantics and merge this one. Example: text ‘key.n.01’ var_name

Arrays
Arrays are supported and are configured by just appending square brackets to the end of a variable.

For instance for an object array:
object * object_arr[]

Permissions
Users of nudge can restrict who is authorized to make changes to data within individual objects and what methods on the objects are available to them. This is configured on the class definition by defining variables that will have the ability to make changes or access methods that are outside that objects typical purview.

```
class User 'user.n.01' (object):
    | variable.allow @admin_object shared_calendar
    | method.allow @admin_object getNews

    text 'first_name.n.01' first_name
    text 'last_name.n.01' last_name
    SharedCalendar * shared_calendar = @"$abc123"

    object * admin_object = @"$xyz789"

    def getNews 'get.v.01 news.n.01' (self):
        return wasm.exec("get_news", self)
```

The changes in bold illustrate that whatever object is assigned to admin_object has permission to make changes to the shared_calendar variable as well as call the getNews method.

Any object that is used as an identity object for permissioning must have an assigned public/private key. This is used to by the object to assert its identity when making requests/transactions.

Additionally, objects that are explicitly granted permission, do not have to exist in the same database.

Note that there are no quotes in @admin_object as we are making a direct variable determination and not doing a resolution.

Any object transacting under its own identity, is by default, able to access all member methods and variables on its own object. This permissioning is not recursive. So assigning the shared_calendar variable to the User object, does not give the user object any implied permissions to the object referenced via the shared_calendar.

How permissions are implemented depends on the type of access you are enabling for the variable/method. In some cases data will be available as plaintext and access control will be centers around authorizing users to append or modify the respective datatype. In other cases AES256 will be used to protect the data/methods with method bodies and data values being encrypted on upload.

Objects as Tokens
Objects will also carry the ability to be traded like assets/tokens. Asset traded objects will have a higher degree of replication and consensus applied to them.

Ensuring Trust in a Trust-less Environment

Enacting a trade with cryptocurrencies carries it own unique set of risks when compared with credit cards or physical cash. Because of the finality of a crypto transaction, and the potential for anonymous trading partners, hackers and scammers have significantly more incentive to look for exploits.

This concern is also needed when storing data in a trust-less environment. In an age where data can carry significant value, care must be taken in how we store information in a distributed environment and especially how we access that data.

Design decisions must be true building blocks toward the longer term vision and also provide real security on the way there.

Run As Identity/Object
Code execution within Nudge, can occur in the cloud and on user's client device. It is up to the developer to decide where executing their code best fits their application. In many cases this will involve a combination of both. A developer may even want to designate their own trusted servers to be the *Execution Nodes* for their cloud code. They may want to make requests to non-Nudge systems (For example a vendor fulfillment system).

This sort of configurability opens up a lot of security concerns. What code should nodes be allowed to execute? How to ensure methods are only run both those authorized to do so? How to ensure data is visible to only those authorized to see it? To address this we are taking page from unix best practices. All requests for data and code must also include an identity of who they are running as so that only the data and methods they are authorized to interact with are available to them. This is relevant for both client and cloud executions. Although in some cases use cases where there is no identifiable user, an 'anonymous' user can be configured to stand in with defined permissions for that use case.

By segmenting access and execution in this way, it creates a solid foundation for security, auditing, and debugging.

Execution Identities can be any object. There are no special objects in Nudge. The one requirement in this case is the object must have an assigned public/private key pair to assert operations and self identity. So you can create a robot object to identify which methods and data your raspberry pi power robot has access to in your cloud. In the

same vein, you might create a “Parking Meter” object to identify which methods and data it has access to in the city cloud.

All data and methods that are protected are encrypted by default, so they will not even be readable to non-authorized users (How this works is covered in the next section). In many cases, if we can heuristically determine some data or method is not relevant to a particular identity, they will not even be made aware that data/method exists.

Securing Data and Methods

In an ad hoc distributed database system, with the costs of data loss or exposure potentially being very high, we operate under the assumption all nodes that store user data are adversarial. If a data value is protected, it’s going to be encrypted. There are too many ways things could go terribly wrong by storing high value data in plaintext on a potentially anonymous server.

First, there are no scalable ways to discern what is happening behind the scenes at an anonymous node. They might be looking to collect information on a specific database and attempt to become a replication node or they might offer to sell the data they have to a 3rd party.

Even if an operator has the best intentions, other things can still go wrong. Someone could physically break into the building where their node is and steal the computers hard drive. Poorly administered nodes could be compromised remotely and have their data stolen. Some nodes might even be on an insecure wifi network - again potential for data theft. Even attaching an infected USB drive to the computer could be a problem.

In a world where data is becoming incredibly valuable, all of these scenarios (amongst others) need to be consider carefully.

Nudge uses a multi-pronged approach to handle these and other scenarios. Some techniques are established industry best practices. Some are newer developments in cryptography, and some are innovations at Nudge. The integration of all these practices is what we are calling “Hardened Threshold Consensus”.

Hardened Threshold Consensus

Hardened Threshold Consensus is a series of checks and balances designed to ensure that access to distributed data or methods is restricted to only identities who have been granted access. Under this scheme, protected and private data will be able to be stored in blockchain. However, database nodes storing the data will never, even for a moment, be able to peek at the values of protected data. The performance of this is quite quick and doesn't require an elaborate byzantine consensus before values can be revealed.

Storage and retrieval of protected data or methods involve the following roles:

- Database Administrator
  This is the identity is authorized and responsible for creating the database, setting up permissions. The database administrator is entitled to read/write to the database. The database administrator can authorize other identities to read/write to the database.

- Database Client
  This is an identity who is authorized to read and/or write data to the database.

- Database NodeGroup
  This is a group of servers (Database Nodes) that hold the stored data for the database we are interacting with.

- Access Control NodeGroups
  This is several groups of servers that authenticate access requests, log access if desired, and returns signature shares (see Threshold Signatures below).

In order to achieve this we make use of the following cryptographic technologies:

- **AES256**
  The 256bit key variant of AES encryption is the current industry standard for storing high value data. *It is quantum-resistant* and can be hardware accelerated.

- **BLS Threshold Signatures (BLS, Shoup et al.)**
  Threshold cryptography enables you to split sensitive data into (n) multiple shares and distribute them among a set of participants. The threshold (t) is the minimum number shares you need to use the shares in a way that is to *equivalent* to having the original sensitive data. In a (t,n)-threshold signature scheme, n parties jointly set up a public key (t*he group public key*) and each party retains an individual secret (*the secret key share*). After this initial setup, t out of the n parties are required and sufficient for creating a signature (*the group signature*) that validates against the group public key. The mathematical techniques used in these operations are designed such that the unshared private key never appears during computation.

Setup Overview

Helpful Terms:

- *Ciphertext* - In cryptography, ciphertext or cyphertext is the result of encryption performed on plaintext using an algorithm, called a cipher.

- *Plaintext* - In cryptography, plaintext or cleartext is unencrypted information, as opposed to information encrypted for storage or transmission.

Every database created within Nudge has two security domains, a Public domain and a Private domain. Every domain constitutes a layer of encryption used to establish a wall between different security concerns. So every protected element within nudge is encrypted twice (with AES256) - once for the private domain, and then that data is encrypted once again for the public domain. These are the root of checks and balances to ensure only those intended to access data are able to.

The Public Domain contains the full database blockchain.This is an intermix of encrypted and possibly non-encrypted data and it is stored on the Database Nodes. It is assessable for anyone authorized to view that specific database blockchain in entirety. This might mean anyone can see it. Applications that are intended to be publicly auditable would be configured in this way. In other cases, access to the public blockchain might be restricted to only those specifically authorized (for example, a shipping company might authorize partner companies in the supply chain). Protected data in the Public Domain has two layers of AES256 encryption. We call this encrypted data *public ciphertext.* The public ciphertext is included in the hashed value when calculating chains (although the hash of the internal private ciphertext is also accounted for).

Database nodes only store their protected data as public ciphertext (2-layer AES256). If a database node containing were compromised, at best, the hacker would be able to gain access to this twice and encrypted data and some single encrypted private domain data depending on how sophisticated the attacker was and database configuration.

Database Nodes are never given access to the details necessary to decrypt the internal private ciphertext.

The Private Domain contains a more granular view of the database. What data you can view in this domain, is determined by what has been authorized to your identity. Data is decrypted from the public domain to the private domain ‘on the fly’ by the Database Node the Database Client is requesting from. This decrypted data is still not plaintext data yet. It is *private ciphertext* data and has one layer of AES256 encryption.

Decrypting to Plaintext

The final decrypt operation to translate the private ciphertext to the fully decrypted *plaintext data is done on the Database Client device.* This assures that plaintext data only passes through the consuming system.

Key Management

Every AES256 decrypt/encrypt operation relies on a 256 bit secret key. Access to this keys is restricted by a consensus mechanism that only allows the requesting identity to determine what the key is upon a minimum threshold of approvals. In addition this consensus mechanism only enables users who have been previously authorized by the database administrator to actually be able to recover the key. This works in the following way.

The Signature Encrypted Envelope

Every individual piece of encrypted data within Nudge is contained within what we are calling a “Signature Encrypted Envelope” (large blocks of data may be broken into multiple envelopes). This data structure contains three values relevant to decryption/ encryption:

- a signature id - identifies a particular threshold signature scheme (distributed public/private key pair)

  The key pairs we use are special in that the signatures they output are deterministic and unique. This means that whenever we sign the same input data it will generated the exact same signature output and that output will be unique when compared to the output of alternative input data.

- a payload key - very simply a cryptographically random string of bytes

- a payload - the ciphertext.

In order to recover the secret key for the payload, we need to generate a signature from the private key identified by the signature id. How we generate that signature is covered in the next section, but just understand that *the secret key for the AES256 payload is the signature of the payload key truncated to 32 bytes (256 bits)*.

Generating the Signature
Every security domain is assigned an authorized decrypter. This is the identity who is explicitly authorized to know what the secret key is for a particular signature id. For the public domain, this is any database node retaining information for a particular database. For the private domain, this is the identity of any Database Client assigned read or write access.

Whenever a security domain is created, a BLS (t,n)-threshold signature scheme is also created for that domain.

For those unfamiliar with BLS threshold signatures. BLS threshold signatures (Shoup et al.) are a cryptographic algorithm that enable a *group public key* to be created alongside n secret key shares. In a (t,n)-threshold signature scheme, each party retains an individual secret (*the secret key share*). After this initial setup, t out of the n parties are required and sufficient for creating a signature (*the group signature*) that validates against the group public key. The mathematical techniques used in these operations are designed such that the unshared private key never appears during computation.

What this means is we can distribute key shares to different parties and have them generate partial signatures, that when combined is equivalent to what we would generate from the private key of a public/private key pair directly.

The threshold (t) and the number of key shares (n) for these schemes that we use by default is 4. Three of these four keys are given to three separate Access Control NodeGroups (Access Control Nodes authenticate access requests, log access if desired, and return signature shares).

The remaining key is given to the authorized decrypter's identity.

So in order to generate a valid signature, we would send the data to be signed to the an Access Control Node in each NodeGroup (we could also group multiple signatures in one request). Each node will validate access to that particular threshold signature for the requesting identity. If everything checks out they will respond back with a secret key share. The authorized decrypter will then calculate their own key share, combine it with the requested key shares and now have the signature.

As it takes four shares to generate a proper signature, and the authorized decrypter doesn't share theirs, only authorized decrypters are actually able to generate signatures. If we want to revoke access quickly, we can do so by revoking the permissions for a particular decrypter identity. This in turn will stop Access Control Nodes from returning key shares for that particular decrypter identity.

An additional benefit of using signatures as private keys, especially in a consensus environment is that they can be validated via the group public key. This ensures that there is honesty within the key generation and that subsequent authorized users will be able to recover and decrypt encrypted data.

Access Levels
Note that in regards to data access, there are three different levels (public, private, and protected). Access level protection can be identified at a class, or variable scope. The words ‘public’, ‘protected’, and ‘private’ are reserved words and should not be used for variable naming, etc.

public:
- This data has no encryption applied to it at all. This data will be visible by anyone with access to the database blockchain.

private:
- This data is only viewable to authorized users on the database client device. The threshold consensus to view this data relies on Access Control Nodes, Database Nodes, *and a secret key share for the database client.*

protected:
- This data is only viewable to authorized users. The threshold consensus to view this data relies *ONLY* on Access Control Nodes and Database Nodes. In these scheme a secret key share is NOT issued to the database client

  We will release an early version of this by allowing the database administrator to issue a (t,n)-threshold signature scheme where t is user-definable and n >=t.

  However, what we view as a proper implementation of this will require a Random Beacon Implementation (Dfinity, et al.) in order to remove coordination of which nodes to be assigned to a particular database from centralization.

Object Identities
Not all objects will have an identity. An identity is only required for an object if we want that object to be able to initiate read or write transactions within a database. Objects have an identity if they have a public/private keypair assigned to them. This will in turn also enable a keychain for the object.

An object’s KeyChain is a list of KeyLink objects. A KeyLink object contain a secret key share as used for the private access control level. This secret key share is encrypted with the object identity’s public key. So this key share is only able to be decrypted by the object identity’s private key.

The lifecycle of a KeyLink object is not controlled by Object it is assigned to. It is primarily owned by the identity that assigned it to the user. In this way, secret key shares can be removed by the assigner. It should be understand that this removal is informational rather than authoritative as there is nothing stopping a sophisticated user from keeping a copy of this secret key share. The authoritative access revocation occurs with the Access Control Nodes no longer returning their signed shares and the Database Nodes no longer returning decryptable data relevant to that KeyLink.

Putting It All Together

Example:
A Database Administrator wants to create a database where a producer identity can upload objects and a consumer identity can read them.

Note, the following also includes 'behind the scenes' steps and not every step is something an administrator will have to take action on themselves. Important keys should be saved to an encrypted local file for the administrator to move to an offline back up.

1. **Create Empty Database**
   An empty database tree is created. This database has the default root database as its parent. Underneath this database, two child databases are created - one for _data (this is for user defined data) and one for _ext_data (this is for framework data). This creation is signed by the database administers private key and upon creation, he/she is assigned admin status to the database.

   An Access Control NodeGroup list of multiaddr and public keys along with a Database NodeGroup list multiaddr and public keys will be assigned and returned as part of the response.

2. Create Threshold Signatures and Ids
   A threshold signature scheme is created for both the Public and Private domains. This includes four key shares for each scheme. Three keys from each scheme are submitted to the Access Control NodeGroups. This will return a response of two signature ids.

   The remaining key share, group public key, and signature_id for the Public Domain is signed and sent to the Database NodeGroups KeyChain for association with the new database. A test signature coordinated between the NodeGroups and Access Control NodeGroups will be run and validated against the group public key prior to commitment.

3. Create Class Definitions
   class Message 'message.n.01' (object):
   text 'value.n.01' value

   class User 'user.n.01' (object):
   |variable.grant.append class.User messages
   text 'first_name.n.01' first_name
   Message * messages[]

   Class definitions are signed by the database admin and submitted.

4. Create User Identity Objects
   The database administrator signs and submits a request to create two User objects as defined in the class definitions. Each have a public key assigned to them. Their private keys are retained so we can assert their identities. The administrator also adds the Private Domain keyshare to the keychain for both users.

5. Distribute User Identities
   Note, that in this example we have the database administrator setup the user identities. However, this could be done perfectly fine in a decentralized way with the database administrator never knowing the private key of the producer or consumer. For simplification in this example we do it this way.

   For reference though, the database administrator could have just simply assigned known public key to the user identity or even used an existing object identity from a separate database.

6. **Producer sends a Message**
   The producer will create the entire message object on their device. As this is going to be a private object, the ‘value' variable will be encapsulated into a Signature Encrypted Envelope.

   The producer will generate a random byte string to be used as the Payload Key. This will be signed to be use as the AES256 encryption key.

   The producer will then look in their keychain for a keylink relevant to the database they are operating in. They will choose a keylink, decrypt it using their private key and use the containing secret key share to generate a signature share. They will also in parallel request signature shares for the payload key from the access control servers. The signature shares will be combined to generate the complete signature. This signature will be validated against the group public key to ensure validity.

   The signature is then truncated to 32bytes and the variable is encrypted to the payload ciphertext. We now are able to contract the Signature Encrypted Envelope and attach it to the Message Object.

   This object is then signed by the producer to assert their identity and it is submitted to a Database Node for append to the Consumer User Object messages variable.

7. **Database Node Processes Message**
   The database will validate that the producer has the appropriate rights for the operation they are attempting. If so, the Database Node will apply the Public Domain encryption layer.

   This plays out much like the Producers Private Layer Encryption. The database

node will generate a random byte string to be used as the Payload Key.

The database node looks at their keychain for a secret key share relevant to the database they are working on. They will make a request to the Access Control Nodes for signature shares, combine it with their local signature share to get the complete signature. This signature is validated against the group public key and the Signature Encrypted Envelope is created.

This Message Object is now updated to reflect the new Public Domain Version. This message is distributed for NodeGroup replication.

8. NodeGroup Replication
   NodeGroup Replication distributes the object to other nodes within the NodeGroup. Nodes will validate the object decrypts, track hash values for their copy of the database blockchain and sign confirmations to the other nodes in the NodeGroup.

9. **Consumer Requests Message**
   All variables, objects, methods, etc keep a change_count. This is a hierarchal versioning that can be used to request an update to a locally stored object.

   In this case, the consumer, depending on its local object state would request from the Database Node changes since the change_count it has. This would cause the Database Node to first validate if the signed request is authorized for the data it is requesting. If so, then it will determine which data should be returned and translate any of that data from the Public Domain to the Private Domain. This is done by decrypting the outer Signature Encrypted Envelope on each variable.

   For each variable (much of this will be batched, including request to the Access Control Nodes), the Database Node will look up the signature id it uses and translate that to a secret key share (as stored in its keychain or locally in memory). A local signature share will be generated, combined with the Access Control signature shares and this will be used to decrypt the payload ciphertext. As the public payload ciphertext simple contains the private signature encrypted envelope, this is updated to the appropriate object for the users request and sent to the user as a signed response to their request.

10. **Consumer Reads Message**
    At this point the consumer has a Private Domain copy of their User object. They simply need to look up the secret key share in their keychain, make the appropriate calls to the Access Control Nodes and they will be able to obtain the complete plaintext copy of the object.

Separation of Concerns

Database Nodes are responsible for determining if you have access to a particular Public or Private Layer object. They do not have any capability by themselves to read any private or protected data.

Access Control Nodes are responsible for determining if you have access to a particular (t,n)-threshold scheme. They do not have any capability by themselves of accessing any Private Layer data or access restricted Public Layer data.

Object Identities are the only ones capable of reading private access data. They must have a secret key share in their keychain for the specific signature scheme the data was encrypted with in order to do so. Additionally, Database Nodes are responsible for restricting Object Identities to only data they are authorized to. This means that they should not even have the ciphertext to decrypt in the first place.

Encryption Flexibility
Note that in this use case, the only identities capable of reading the message were the producer, consumer, and the database administrator. In some scenarios, a database administrator might want to allow data to be hidden even from them. This is also possible.

As Private Domain Encryption is a coordination between issuing parties, setting up an encryption scheme that JUST the producer and consumer are able to decrypt is a simple as adding a new threshold scheme to the Access Control Nodes and updating target user keychains.

Access Control Flexibility
There are multiple paths to revoke a users access. To remove a specific user access entirely, we can remove a keylink from their keychain and then their identity from the Access Control Nodes. We can also change a user's access rights to specific data. This will cause the Database NodeGroups to stop serving that data to that users identity.

Additionally, in some cases, advanced users might want to use multiple different Private Domain (t,n)-threshold schemes within a database or a common scheme across multiple databases.

Data Control Flexibility
Being able to link data within multiple granular permissioned databases allows us to share, track, and update data in ways we never used to be able to. For example a user would be able to share access to their address to enable a shipment, and then revoke access when they no longer want that data shared. As well, if a user moves, they are not updating their address in multiple places. They would update their address object and all other consumers of their data would have the latest information.

Node Incentives

Ensuring nodes are profitable is critical to the stability of the network. The incentives we use to get new node operators and retain existing ones is something that we will continue to refine even after the initial release. As we observe the network and get feedback from users, we will make changes where needed.

An ideal system would enable price discovery and incentivize new providers to join if capacity is low.

The main metrics that we want to optimize for are:
- Availability - Is the data is there when we want it and not corrupted.
- Latency - How quickly does request for data start to return data (milliseconds)
- Throughput - How fast do uploads and downloads stream (bits/second)

Transaction Receipts
Transaction Receipts are used by nodes to assert the work they have done for the network. These are issued by the client when they request data or operations. Nodes will also receive signed transaction receipts from clients asserting performance statistics. As part of node discovery a summary of these statistics will be shared to help clients make data sourcing decisions.

Tier Ratings
Transaction receipts are also used by Nudge to make tier rating decisions for each node. A tier rating is a group of nodes that have similar performance characteristics. These nodes do not have to be servicing the same data, but have similar performances in regards to the factors we are optimizing for. Several times a day Nudge will generate Tier Ratings. Generation will be done using a nearest-neighbor chain algorithm. This tier rating is used as the basis for the payout a node will receive during that generation. The drop off in tier payouts will be based on the distance between clusters.

NodeGroup Replication
We will attempt to generate a probabilistic minimum of 3 copies of database data available at all times. At times this will be more than 3 copies available depending on the tier rating of the nodes available. Replication decisions will initially be be coordinated by a central server.

The decision to assign a node to a NodeGroup will be base on how well it complements the group. Performance metrics like latency, throughput, data integrity, software version, how long it took to do a version update, as well as time of day

variance statistics will be considered. The goal will be to ensure consistent availability, security, latency and performance throughout the day.

Part of ensuring that will include using past performance to predict future performance. So nodes that demonstrate bad metrics will be predicted to produce bad metrics until they demonstrate they can be relied upon.

Nodes that continue to produce bad metrics will see themselves get deprioritized when clients request peer information from a particular NodeGroup. This means they will end up with less transaction receipts and thus less payout from the payout pool.

Payout Rate
Payouts will come from the Payout Pool and be a prorated sum of the node contribution to each NodeGroup they are assigned to. All payouts will be in NDG.

Node contributions are measured as:

*(storage_allocated * transaction_receipt_weight * predicted_tier_rating * actual_tier_rating)*

Each variable will be normalized relative to the NodeGroup and every calculation will involve the predicted_tier_rating being updated.

The Payout Pool

The payout pool will be funded from two different fees:
- A Marketplace fee - This is a small fee that will be charged for every sale.
- A Database fee - This is a fee charged for database upkeep.

Payout Distribution
As the payout pool is a common fund from two different income sources, it will be used to fund multiple portions of the infrastructure and support infrastructure. Tentatively, we expect the breakdown to be as follows:

- 50% Node Payouts
- 20% Growth and Marketing
- 30% Platform Development

Business Model

We want the Nudge Marketplace and Database to be accessible for as many people as possible. Fundamental to the long term success of the project will be a stable business model. While Nudge is raising funds through the use of a token sale to fund development of the Marketplace and distributed protocols we are relying on these funds as the cornerstone of continued platform development.

We will arrange our future revenue streams as follows:
- Marketplace Transaction Fees - A portion of the transaction fee in the marketplace will be set aside for platform development.
- Database Fees - A portion of distributed database upkeep fees paid by vendors and other database users will be set aside for platform development.
- Development of our own DApps on top of the Nudge Protocol
- Machine Learning Services to Companies.

Fiat Proportionality
As the value of NDG will float, we are also looking at different ways to keep costs fiat proportional within the marketplace. We do not want to end up in a scenario where selling something in the marketplace costs the equivalent of $60 in transaction fees (happened to bitcoin as its value increased).

Costs need to be predictable in order for businesses to see the viability of integrating Nudge.

So as the value of NDG changes so will the amount of NDG nodes receive for executing transactions and storage.

Fiat Onboarding
In addition to being able to pay for on boarding fees with NDG, either at launch or shortly after, we will enable alternative methods of paying for onboarding. This would be credit cards or even purchase orders for larger agreements.

Payments made with fiat onboarding will be exchanged for NDG before addition to the payment pool as all node operations will transact and operate solely with NDG.

We consider fiat onboarding important as many businesses are limited by organizational structure to either using corporate credit cards or purchase order agreements. Additionally, there are others who will want to use the marketplace to sell products/services for cryptocurrency or use the database for a project/application but find getting past the initial learning curve of obtaining and transacting with cryptocurrencies an obstacle. We want to make getting people into the system as easy

as possible and guide them along the way. We find fiat on boarding will be the easiest way to do so.

Service Tiers

Free Tier
This will be the onboarding tier for new users of the Marketplace. Users who sign up will be walked through some cryptocurrency basics if they desire. Users who have existing NDG will also be able to link them to their your account at here as well.

Users who validate their accounts via an online identity or id will be eligible for some free NDG added to their account. How many NDG and how we will be validating identities is something we will disclose prior to launch. The amount of onboarding credits issued to each user will decrease as more users sign up.

The goal of onboarding credits is to incentivize user sign up and to lower the resistance of making a first purchase. The amount of on boarding credits issued, will be decided as we get closer to launch.

Paid Tier
Users who are paying for database services will be a part of the paid tier. This includes vendors as well. Vendors sending fulfillments to users or offering products/services will store these objects within the distributed database environment. The number of objects they need to store will influence the size of the database they need.

While costs won't be finalized until shortly before launch, when pricing, we are also taking into consideration the financial limitations of geographical regions like Africa that have low annual incomes. We want the lowest tier of access to be fiat proportional to be approximately $1.00 a month when paid yearly. What this translates to in terms of storage and services has yet to be determined. Higher priced tiers will intuitively offer larger storage and service options. We anticipate mid-level tier offerings will be the most common option for vendors as they get situated into the platform and use it more.

More than a Just a Business Model

Nudge aims to be a community driven project. We will be allocating 10% of the initial NDG supply to support that mission in a “Community Fund”. This allocation will be available to support projects that the Nudge Community feel strongly about, as well as to continue to grow the community through Free Tier Incentives for users and nodes, and feature development.

We believe DogeCoin has lead the way in showing what is possible in this context. With Nudge, we would like to have a dedicated fund to ensure projects are consistently fundable.

In order to ensure this fund remains topped up, two years after launch we will implement a variable inflation of up to 1% (inflation would never exceed 1%). What formula we use to determine the percentage will be finalized within a community discussion. Data points will likely include market cap and volume growth over the previous 12 month period.

Decentralized Governance

Over time, a decentralized governance may be introduced to further strengthen the Nudge community. In an effort to cast an ever growing net, various stakeholders would come together to co-create an ecosystem around the protocol itself. In this light, the decentralized governance model may be introduced in the future to drive protocol-level development updates, conventions and update integration.

Exchanges

Nudge will strive to provide as many diverse opportunities for users to purchase or sell tokens. We will actively engage in partnerships with both centralized and decentralized exchanges. When looking for partner exchanges we will apply the criteria of legal fit as well as acceptable commercial conditions.

Competitive Landscape

The immense capabilities of the blockchain lie in front of us and have barely even been tapped into. As of the moment of writing this whitepaper, we are unable to identify a competitor focusing on developing an open, language aware, distributed database and marketplace protocol and architecture for the internet. While there are existing companies in the marketplace space and database space, existing solutions are closed and not entirely comparable products.

Amazon for instance continues to provide a great marketplace for fiat - regardless of the fact they have largely neglected cryptocurrencies. We are not building another Amazon.

What we are building a significantly better way for object-oriented data on the internet be delivered, used, and combined. We are building a platform where services that aren't even aware of each other can collaborate and provide more value that the services would be able to do independently. Our marketplace will eventually provide common hooks for any currency to hook into.

Under the hood, the NDG utility token will provide value as a unifying currency layer for all services and applications connected to Nudge. This means Internet of Things, services, and providers can seamlessly interact, autonomously or non-autonomously, without differing currencies being an obstacle.

Our unique machine learning/AI approach will also be an anchor in ensuring value for new users and applications and continued value for existing ones.

The Nudge Utility Token (NDG)

Token name:	NDG (Stellar)
Token Supply:	10,000,000,000 NDG
Community Fund Growth	variable, never exceeding 1% (activated 2 years after launch)

The Nudge Utility Token (NDG) will be issued on the Stellar Network.
Many factors influenced our decision in choosing Stellar over other networks.

Why Stellar vs Ethereum ERC20?
Performance metrics and its ability to do on network exchange to other tokens or fiat were the deciding factors as these features will provide a clean path for NDG Tokens to make seamless border crossings between Stellar's network and Nudge's payment network.

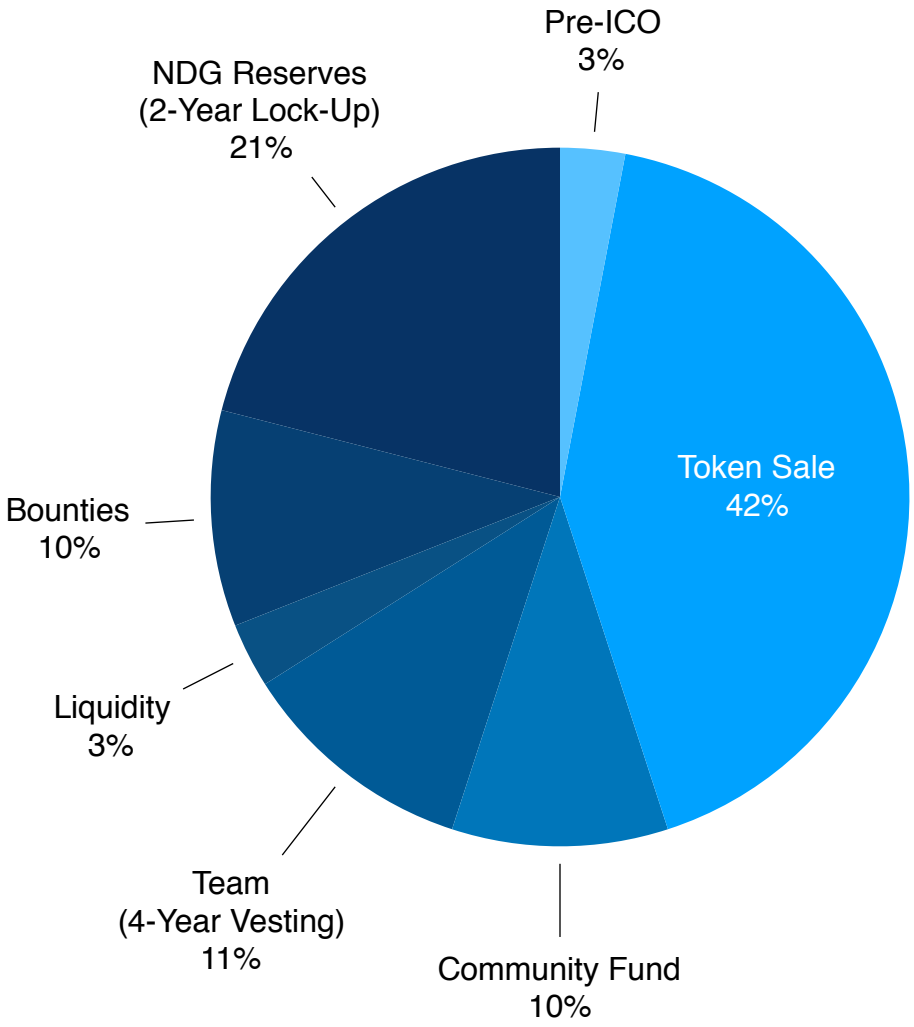
PROTOCOL COMPARISON				
	Bitcoin	Ethereum	Stellar	Ripple
Average Transaction Confirmation Time	1 hour	15 minutes	3 to 5 seconds	3 to 5 seconds
Average Transaction Fees	$0.61 per transaction	$0.02 per transaction	$0.01 for 300,000 transactions	$0.01 for 3 transactions
Transactions Per Second	3 transactions per second	7 transactions per second	3000 transactions per second	3000 transactions per second
Consensus Mechanism	Proof of Work	Proof of Work	Stellar Consensus Protocol (SCP)	Ripple Consensus Algorithm
Validator Control	Decentalized	Decentalized	Decentalized	Centrailized
Governance	Non-profit	Non-profit	Non-profit	For profit

MARCH 2018

Stronghold

Token Supply Distribution

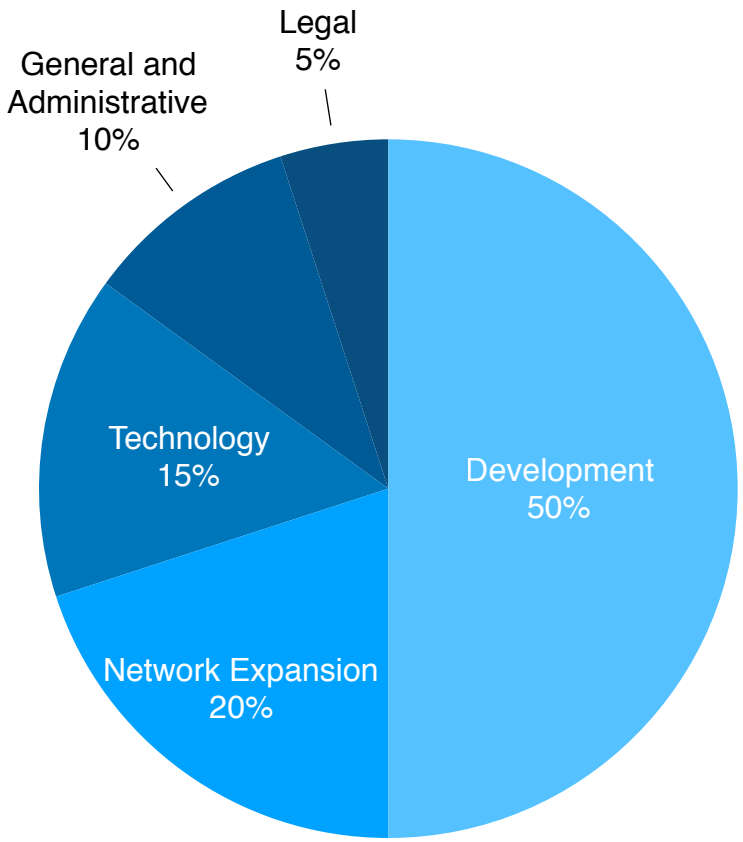
- Team: founders, advisors, current and future employees
- Token Sale: private and public sales



Budget Allocation

Funds raised in token sales will be used solely for the development and benefit of the Nudge Platform.

50%	-	Development (engineering, R&D, team expansion)
20%	-	Network Expansion (global awareness, community building, user aquisition)
15%	-	Technology (building infrastructure, maintenance)
10%	-	General and Administrative
5%	-	Legal



Token Distribution Event : Pre-ICO

Pre-ICO Purchasers will receive a 26.3% bonus:

Token Amount	**300,000,000 NDG (3% of the Token Supply)**
Pre-ICO Price	**1 XLM = 24 NDG (0.04167 XLM)**
Public Sale Price	1 XLM = 19 NDG (0.05263 XLM)

Pre-ICO Purchases of 50k XLM or more will receive a 31.5% bonus:

Token Amount	**300,000,000 NDG (3% of the Token Supply)**
Pre-ICO Price	**1 XLM = 25 NDG (0.04 XLM)**
Public Sale Price	1 XLM = 19 NDG (0.05263 XLM)

Participation in the token sale event will require buyers to:

A. Have Stellar (XLM) cryptocurrency
B. Have a Stellar (XLM) wallet address
C. Do a KYC required for the purchasing of tokens
D. Be a citizen of countries that are eligible for their citizens to participate in the token sale (more details in the Token Sale Terms and Conditions).

About Nudge

Nudge was founded with the vision of simplifying the technical knowledge needed to apply machine learning to applications, organizations, and businesses to about what it takes to learn HTML. We've crossed this threshold and for many applications are able to simplify it even further.

Nudge is lead by its founder, Matt Gagnon. Matt has been involved in the field of AI and Machine Learning since 2009, and has been involved with blockchain projects since 2014.

Prior to that, Matt lead development on a significant amount of the technology powering Dow Jones, The Wall Street Journal, and Barrons Online. Some of these projects include the marketplace, billing, entitlements, fulfillment, paywall, identity, and demographics systems powering wsj.com and barrons.com.