# Obscure Whitepaper

Whitepaper Version 1.0
ObscureIM Version 2.0.0 (Mainnet)

*Written by PausCrypto, Lead Developer, ObscureIM*
*Edited by KodakYellow, Chief Marketing, ObscureIM*

## Table of Contents

## Executive Summary

Cryptocurrency is a peer-to-peer (P2P) protocol where thousands of computer nodes simultaneously agree on the balance of a particular address.

With the inherently democratic nature of decentralised systems, cryptocurrency offers protection to participants by providing a safety in numbers hypothesis. This limits the ability of malicious actors to threaten or attack participants in the system, due to the number of participants involved, including the global and extreme robust nature of their relationships.

Once a piece of software has entered the public domain, it becomes impractical and virtually impossible to shut down completely. This enables security to the future of the system and knowledge that if the public deems it valuable, it will persist.

## Introduction

Our goal at Obscure is to create a fair marketplace for everyone. This means that an individual should be able to sell anything that they desire as long as it is within both moral and legal boundaries. For example, many marketplaces ban the sale of health products such as whey protein, because it is classified as a 'nutritional supplement'.

We at Obscure aim to create a marketplace that is both robust and enjoyable to use, and censorship-resistant while protecting the privacy of users.

The proposal to use XSC will implement compulsory privacy and enhanced block speeds with an attractive emission schedule. The focus of this whitepaper is on XSC, the first settlement layer for the decentralized marketplace.

## Problem

Currently there are several viable cryptocurrencies that we can integrate into our marketplace. However, many of these are not private. For example, it is possible to develop a decentralized market with payments backed by an ERC-20 token. However, each transaction is trackable by anyone, which makes it unsuitable for a fully private marketplace.

On the other hand, privacy coins are unsuitable for utilization to implement smart contracts for a decentralized marketplace. The networks often clog easily due to the privacy features set within, such as Ring CT and compulsory mixins. These limitations and problems can be solved with our implementation of a Cryptonote coin, XSC.

## The Coin

XSC is based on the Turtlecoin code-base. However, several improvements have been made to propel XSC as a formula for a universal payments' solution, even outside of a decentralized marketplace.

Our key goal is to apply XSC in everyday use, on top of being used as the main currency in a decentralized marketplace. This means that several considerations were made when the XSC codebase was being developed. We determined that XSC had to be easy to use for everyone, and the network propagation rates has to be high enough while maintaining a layer of privacy.

### | Comparison & Specifications |

As seen in Figure 1 below, XSC is superior compared to other coins in terms of speed and usability.

| COIN | Bitcoin | Ethereum | Neo | Monero | **Obscure** |
|---|---|---|---|---|---|
| Block times: | ~ 10 minutes | ~ 17 seconds | ~15 seconds | ~ 2 minutes | **~ 15 seconds** |
| Privacy: | NONE | NONE | NONE | YES | **YES** |
| Miner/Staker Rewards | Variable | Variable | Variable | Variable | **3.5 XSC** |
| Mobile Wallet on release | NO | NO | NO | NO | **YES** |
| Smart contract platform | NO | YES | YES | NO | **YES** |
| Government Assistance | NO | NO | NO | NO | **YES** |

### | Block Time |

A block time of 15 seconds has been selected. This will ensure that it is easy to use XSC as a payment solution, which achieves seamless peer-to-peer transactions. A 15-second block time is the upper limit of block times in current Cryptonote technologies. It is fast enough without massive orphaning occurring in the blockchain.

### | Mobile Wallet |

A mobile wallet for Android will be released before 1 April 2019 during the Obscure Launch. An Android wallet is absolutely necessary to ensure quick adoption. Peer to peer payments are almost exclusively done via mobile. The Android wallet will be key in ensuring that XSC remains competitive in the cryptocurrency space.

## | Miner/Staker rewards |

Mining staking rewards are set at 3.54 XSC per block. This is standard with other micro-cap coins. It is hard to determine a balance between inflation value, and we have decided to use an emission speed value of 24 with a block reward starting at approximate 3.54 XSC.

## | Tokenomics & Emission Speed |

### *Total Supply*

A total supply of 65,000,000 XSC has been set. Our final goal is to allow each XSC to be pegged to US$1, and for XSC to be used as a universal payment solution for everyday use, on top of the main settlement layer for our decentralized marketplace. As such, it makes sense to peg each XSC to US$1. This allows straightforward spending and much more *layman readable*.

To drive mainstream use, cryptocurrency has to be easy-to-use, and easily accessible. Mining and the operation of full nodes will be consigned to cryptocurrency enthusiasts and this will raise the difficulty for average users to penetrate into.

Alternatively, for users who lack the skill set or equipment to mine, there will be a faucet available on launch, where up to a total of 500,000 XSC will be up for grabs. However, note that not all 500,00 XSC will be added into the faucet hot wallet. Instead, it will be replenished with XSC when the balance has been depleted.

### *Emission Speed*

As with most Cryptonote coins, a non-linear emission curve is utilized. Our emission speed of 24 puts out emission below that of Bitcoin. This makes each XSC significantly rarer to get, with mining rewards starting at 3 XSC per block during launch and reducing to almost 0.5 XSC per block in 2 years. This will force the coin into scarcity, giving it inherent value. It will accelerate our goal to achieve the peg of US$1.

The emission speed curve is plotted in the Figure 1 below.

### *Faucet*

Justification for the faucet value of 500,000 will be given in this section. Our mining rewards starts at a competitive value of 3 XSC per block. With almost 5,500 blocks mined per day due to our 15-second block times, each faucet claim will dispense 0.3 XSC.

A balance has to maintained between faucet rewards and mining rewards. Hence, the faucet reward is regulated to be lower than the mining reward, to maintain the profitability of mining on the Obscure Network.

The faucet will have no limits to the number of withdrawals. It is based on a first come first served basis. In addition, to increase a high adoption rate, you can refer a friend and claim a cut of their XSC thereafter.

## Premine

A 10% premine (worth a total of 6,500,000 XSC) will be initiated. This premine will be **publicly** viewable to ensure 100% accountability by the developers. A website will be developed to easily check the balance of the premine wallet.

| **Project Goals** | **Total Coins Available** | **Total Committed Supply** |
|---|---|---|
| Community Bounties | 325,000 | 0.5% |
| Faucet | 325,000 | 0.5% |
| Developer Fees | 2,600,000 | 4% |
| Presale | 3,250,000 | 5% |
| Total Premine: | | 10% |

Funds from the presale pool which are not sold will be directed to Community Bounties & the faucet.

## Presale

There will be a small presale to raise 10 BTC for exchange fees and to fund external PR firms. Following that, there will be no ICO. Coins will be distributed equally via a faucet and miner rewards.

## Governance

### | Full Nodes |

Full nodes are servers running on a P2P network that allow peers to use them to receive updates about the network events. These nodes utilize significant amounts of traffic and other resources that incur a substantial cost. As a result, a steady decrease in the amount of these nodes has been observed for some time on the Bitcoin network[1], and hence block propagation times have been upwards of 40 seconds[2]. Many solutions have been proposed, such as a new reward scheme by Microsoft Research and[3] the Bitnodes incentive program[4].

These nodes are extremely crucial to the health of the network. They provide clients with the ability to synchronize and facilitate quick propagation of messages throughout the network. We propose adding a secondary network, known as the Obscure Masternode Network. These nodes will have high availability and provide a required level of service to the network, in order to take part in the Masternode Reward Program.

### | Masternode Reward Program |

Much of the reason for the decrease of full nodes on the Bitcoin network is the lack of incentive to run one. Over time, the cost of running a full node increases as the network gets used more, creating more bandwidth and costing the operator more money. As the cost increases, operators consolidate their services to sustain cheaper costs to run, or run a light client which does not help the network at all.

Masternodes are full nodes, just like in the Bitcoin network, except they must provide a level of service to the network and have a bond of collateral to participate. The collateral is never forfeited and it is safe while the masternode is operating. This permits masternode operators to provide a service to the network, earn payment for their services, and reduce the volatility of the currency.

To run a masternode, the operator must demonstrate control over 70,000 XSC. When active, masternodes provide services to clients on the network, and in return receive regular payment from the block reward. However, masternodes will play a pivotal role in ensuring the implementation of ZeroCT can be correctly integrated into the blockchain.

Due to the fact that the masternode rewards program is a fixed percentage and the masternode network nodes are fluctuating, expected masternode rewards will vary according to the current total count of active masternodes. A masternode payment formula will be decided closer to the release, which includes masternode implementation.

## Privacy Features

Cryptonote cryptocurrencies were built with privacy in mind. However, it is possible that these privacy features are disabled at any point.

Obscure requires and defaults **every** transaction to a a mixin value of 3 indefinitely, with an exception made for the first 100,000 blocks. This is necessary so that early transactions from the genesis wallet can be done.

Since we have pledged to expose the balance of the genesis wallet, this is no longer a privacy issue.

After 100,000 blocks, all transactions will be obfuscated. This should happen in approximately 17 days.

## | What are mixin values? |

In the days before Cryptonote, there were coin mixing operations for Bitcoin users to hide the origin of their coins. The mixer would be a web app that shuffled bits and pieces of Bitcoin back and forth enough where it became slightly more difficult to figure out who sent the coins. Aside from being the perfect recipe for exit scams, these days with companies like Chainalysis and the non-fungible nature of Bitcoin, it does very little to enhance privacy.

Currently, the way mixing is done with Obscure – the network essentially files the serial numbers off the coins first before the mixing is done. This shuffling happens internally, for just about every transaction, which prevents this type of analysis. This mixing is based on a concept called *Borromean Ring Signatures*, which was first written by Blockstream, a Bitcoin company.

Right now, for every transaction, a user can set a mixin number. Someone who uses a mixin of 0 sends their transaction without any benefit of privacy. This can sometimes be a from the misguided assumption that the transaction will cost less, or go faster, but this is not true.[5]

## | Simplified ZeroCT Implementation |

ZeroCT is an implementation suggested by the Alex Vasquez, of the NavCoin Core team. Our flavor of ZeroCT involves having masternodes acting as both accumulators and validators to remove network load of the main chain. [6]

### *Zero-Knowledge proofs*

ZeroCT is inspired by the Zero-Knowledge protocols, otherwise known as Zk-SNARKS, and it was implemented in coins such as Zcash and Zcoin.

In its essence, the Zerocoin protocol is a series of RSA accumulators which exist in parallel to a regular blockchain protocol. Each accumulator represents a specific denomination of coins, usually rounded into powers of 10 (e.g. 1, 10, 100, 1000, etc.). Users can burn their coins to one of these accumulators, which effectively destroys them. The spender receives a receipt for the coins which were burned and can use the receipt to generate new coins at a later date. These receipts can then be given to another user to regenerate and mint new coins.

However, there are several problems to this method:

1) It is expensive to send specific amounts of coins. If a transaction of 2112 zero coins is made, the transaction has to be split into receipts which exist RSA accumulators.
2) Each transaction takes 0.5 seconds, which is too slow.
3) Every time a receipt is generated, the wallet.dat file changes and has to be backed up.

Many cryptocurrency communities have improved the original Zcoin implementation of Zero Knowledge proofs. The PIVX team developed zPIVX , where the receipts can be derived from the wallet’s master key, and hence constant backups of the wallet.dat file becomes unnecessary.

In lieu of these weakness, ZeroCT aims to take the best of different privacy protocols and implement them in a single package.

More concise and technical information regarding ZeroCT can be found in the research paper published by Alex Vasquez of the Navcoin Team.

This whitepaper will continue to briefly explain the technicalities of ZeroCT. We will be looking at the key takeaways of how it addresses the issues that were identified with existing private transaction protocols, and what it means from a practical perspective.

### *Concealing Spender Identity*

ZeroCT uses Zerocoin based RSA accumulators to conceal the origin of transactions. Each time coins are spent, they are effectively newly-minted coins with no associated history and complete fungibility.

### *Concealing Receiver Identity*

ZeroCT’s Anonymous Identities are generated in a similar way to Monero’s one time addresses. They are derived from a Stealth Address which doesn’t leak any metadata about the users balance or transaction history, and is not susceptible to blockchain analysis.

### *Concealing Transaction Amount*

ZeroCT uses Confidential Transactions & Bulletproofs to conceal the amount of coins spent in a transaction.

### *Increasing the Anonymity Set*

The use of a Zerocoin accumulator means the anonymity set for the spenders identity is equal to the number of previous private transactions recorded to the accumulator. ZeroCT allows for and incentivizes private staking, which means this anonymity set will steadily grow, making meta analysis exponentially more difficult over time.

### *Preventing Metadata Leakage*

ZeroCT coins are able to be directly minted to a receivers-derived Anonymous Identity, and can be transferred directly from one Anonymous Identity to another. There is no pre-mixing required and once in the accumulator, coins do not need to leave the accumulator to be spent. This reduces the possibility of a third party inferring a linkage between transactions as they enter and exit the accumulator.

### *Performance*

The use of Confidential Transactions removes the need for denomination-based accumulators, which reduces the proof size necessary when spending natural amounts. The proof size to spend 999 ZeroCT transactions is equal to the size of 4 regular proofs, rather than the 27 (9x100, 9x10, and 9x1) that would be required by the original Zerocoin protocol.

### *Usability*

All ZeroCT minted coins are recoverable by the master private key, meaning you do not have to backup your wallet after every ZeroCT mint.

The ZeroCT protocol has provision for a “view key” which allows for your private transaction history to be audited by a third party (e.g. your accountant) without compromising your spending keys.

Minting ZeroCT coins directly to a third party reduces complexity for the sender by removing any pre-mixing step or wait time.

### *Key Security*

The large primes used for ZeroCT’s initial parameters are recommended to be taken from the 1991 RSA Factoring Challenge for which the keys are universally accepted to have been correctly destroyed and unable to be calculated.

### *What does this mean?*

ZeroCT is capable of anonymizing the sender, receiver and amount in a single transaction without leaking any important meta data.

### *Implementation in Obscure*

We shall refer to ZeroCT transactions as zTransactions, and coins minted using a receiver-receipt as zObscure.

Each wallet will be capable of creating a RSA Accumulator. For zObscure to be sent, the receiver (Alex) has to provide his Anonymous Identity which exposes his RSA accumulator to the sender (Bob). However, it is impossible for Bob to derive past and future RSA accumulators from the Private Identity. Only a particular RSA Accumulator can be derived from an Anonymous Identity.

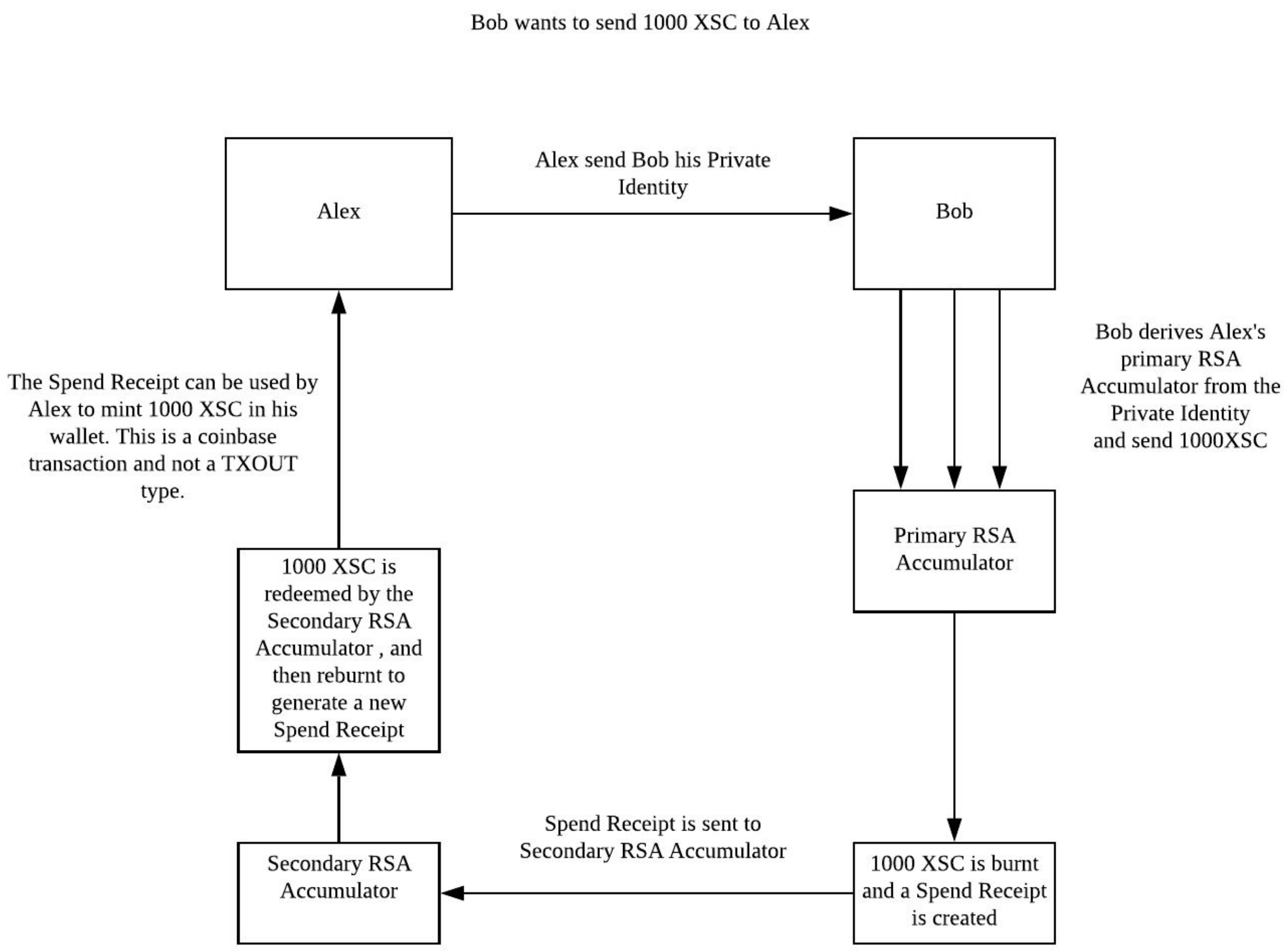
Bob will send his XSC to Alex’s Private Identity, which sends the XSC to Alex’s RSA Accumulator. This transaction will be obfuscated with other transaction outputs, making it impossible for an outsider to know that funds have been transferred from Bob’s wallet to Alex’s RSA Accumulator.  This is made possible because XSC uses compulsory mixin for all transactions with a consistent value across the whole blockchain.

Every coin that is sent to an RSA Accumulator is considered *lost/burnt*. However, the RSA accumulator will create a **Spend Receipt,** which will then be sent and redeemed to a secondary RSA Accumulator also owned by Alex. This secondary RSA Accumulator is hidden and is impossible to be derived from Alex's Anonymous Identity. This secondary RSA Accumulator then generates a Spend Receipt which can be used in the future to mint new Obscure on the public blockchain.

Now that Alex the receiver has XSC in his secondary RSA Accumulator, he has two choices:

1. He can keep his coins in the RSA Accumulator indefinitely.
2. He can *cash out* his coins from RSA Accumulator. He will be able to redeem and mint new coins within his regular XSC wallet, thus increasing his balance by the amount sent by Bob.

Refer to the flow chart for process visualization:


Bob wants to send 1000 XSC to Alex
Alex
Alex send Bob his Private Identity
Bob
Bob derives Alex's primary RSA Accumulator from the Private Identity and send 1000XSC
Primary RSA Accumulator
1000 XSC is burnt and a Spend Receipt is created
Spend Receipt is sent to Secondary RSA Accumulator
Secondary RSA Accumulator
1000 XSC is redeemed by the Secondary RSA Accumulator , and then reburnt to generate a new Spend Receipt
The Spend Receipt can be used by Alex to mint 1000 XSC in his wallet. This is a coinbase transaction and not a TXOUT type.

Each zTransaction involves a coin being burnt effectively, burnt again, and finally redeemed by the receiver.

Each Spend Receipt can be redeemed by anyone, as long as the hash for the Spend Receipt is known. However, each Spend Receipt will generate a unique Coinbase Transaction Hash. Hence, if a Spend Receipt is redeemed, and the Coinbase Transaction Hash created has never been created previously, this means that the Spend Receipt is valid.

A network of Masternodes (MNs) will be used to track every Spend Receipt for validity and destruction. The MNs will not have a record of Spend Receipt hashes, but rather a chain of all Coinbase transactions. MNs which attempt to attack the Coinbase transaction chain will lose their collateral of 20,000 XSC.

## | Technical  ZeroCT implementation |

This section lifts information from the ZeroCT white paper, but we have made significant changes where necessary for a successful implementation in a cryptonote coin.

### *Minting*

Minting is defined as  the creation of private tokens (zObscure) as defined by the ZeroCoin Protocol[7]. Similar to bitcoin[3], the amounts of input must equal the amount of outputs minus any fees.

“To mint a zerocoin c of denomination d, Alice runs Mint (params) → (c,skc) and stores skc securely. She then embeds c in the output of a Bitcoin transaction hat spends d + fees classical bitcoins. Once a mint transaction has been accepted into the blockchain, c is included in the global accumulator A, and the currency cannot be accessed except through a Zerocoin spend”

Mathematically minting a coin means calculating a Pedersen Commitment[8] which value will be later accumulated in the accumulator of the corresponding denomination. A Pedersen Commitment is a one-way function where you can commit to a value v under a blinding factor s without revealing the value v until a later time:

$$c = g^v h^s$$

Given *c*, it is impossibly hard to determine a value for *v* and *s* for 0 < i < n. This is known as a Discrete Logarithm Problem. thus, a Pedersen Commitment is robust in hiding the value that it commits too, allowing it to be resistant to brute-force attacks.

Pedersen commitments are also homomorphic, where the values of the products of any two commitments is the sum of commitments.

The original Zerocoin protocols[7] uses an RNG generator to the value of S. The future spender of the minted coin is required to prove knowledge of both values S and r constraining the spending action to the original minter. Our contribution allows an actor to commit in zero knowledge to secret values only known to an external party, even if those are publicly disclosed later.[9]

### *Constructing a transaction*

First, lets define how a private identity is generated. Let B be the public part of an elliptical curve key, where *B = bG* , for **$j_1$ <- $Z^*_q$ , $k_1$ <- $Z^*_q$ , $j_2$ <- $Z^*_q$ , $j_2$ <- $Z^*_q$.**

The triplets **($B,Z_1,Z_2$)** is known as the anonymous identity. The tuple **($b,j_1,j_2,z_1,z_2$)** is considered the private view key $PK_{view}$ , and this allows the wallet to identify which outputs contains spendable private coins. The $PK_{view}$ can be publicly shared to determine an addresses private transactions without compromising exclusive spending rights. The tuples **$(b,j_1,j_2,k_1,k_2)$** is known as the $PK_{spend}$ and is used to construct proof of for zObscure ownership, allowing it to be spent.

Thus the process of sending zObscure from Alex(receiver) from Bob(spender) , where Alex will supply Bob with his private identity $I_{alex}$ .

1) Extract B, $Z_1$, $Z_2$ from $I_{alex}$.

2) Generates a new EC key, A = aG , calculates a Diffie-Helman secret $\chi$ using Alex's EC public key B.

$$x = H_s(aB)$$

3) Uses H as a Pseudorandom Number Generator to compute $\sigma$ and $\Theta$ taking the shared secret $\chi$ as the initial seed.

$$\sigma = H(\chi)\ (mod\ q)$$

$$\Theta = H(\sigma)\ (mod\ 2^u - 1)$$

4) Lets $c = z_1^{\chi} z_2$ (mod p) and $\epsilon \leftarrow C(w, c, \sigma)$ .

5) Verifies c and $\epsilon$ are prime numbers and within the allowed range required in the accumulator proof[10] . If the test fails, she repeats the process going back to the second step. If it passes, she continues with the next step.

6) Includes a zero knowledge range proof that the value committed in $\epsilon$ is a positive number and lies in the range [0, 2u).

$$NIZKPoK\{(v,\ \sigma)\ :\ \epsilon\ =\ gwhcg\sigma 1\ \wedge\ 0 \leq v \leq 2u - 1\}$$

Methods like Bulletproofs allow provers to bundle many range proofs in one of compressed size, making it possible to compute one proof per transaction instead of using the more expensive model of one-proof-peroutput.

7) Let $W = w \oplus \epsilon$ the amount obfuscated with

8) Reveals (A, c, W, $\epsilon$) in the output of a transaction.

$$c = z_1{}^x z_2 = (g^{j1})^x (h^{k1})^x g^{j2} h^{k2} = g^{(j1x+j2)} h^{(k1x+k2)}$$

We consider that the above equation is satisfied. We can claim that the value c from the equation above is equivalent to the Pedersen commitment with one secret and one randomness value.

Bob will know the knows z1, z2 and $\chi$ but she does not have knowledge of j1, j2, k1 or k2 because of the properties of the Pedersen Commitment and under the assumption of the hardness of the Discrete Log Problem, thus she would be committing without retaining the ability of later opening the commitment by using the serial number S = j1 $\chi$ + j2 or the randomness r = k1 $\chi$ + k2 in the construction of the proofs that are necessary to spend the coins.

This scheme retains the perfectly hidden property from the Pedersen Commitment construction as j1, j2, k1 and k2 are uniformly drawn from Z∗q while $\chi$ is calculated mod q, being the distribution of the resulting j1 $\chi$ +j2 and k1 $\chi$ +k2 equally uniform.

An actor observing the chain and acting as a validator would accumulate c and $\epsilon$ in different accumulators A and V respectively. Masternodes will be acting as the actor. A more in depth explanation of how masternodes act as a validator is explained in the next section.

The private key a will be stored by Bob and used to prove the minting of specific coins without revealing Bob's whole transaction history or identity.

Due to the use of only one anonymous identity to receive coins, this scheme does not facilitate the use of short-lived addresses to identify individual payments, which is a common use case for merchants in other cryptocurrencies like Bitcoin. To solve this we propose the calculation of an extra parameter o = H( $\Theta$ ) used to obfuscate a Payment ID/Message M as in M′ = M⊕ o, being
the maximum admitted length for |M| the bit length of the output from the chosen hash function H. M′ can be attached to an extra metadata parameter of a transaction, as an additional byte array in the output's scriptPubKey or as an OP_RETURN OP_PAYID script in a 0-value output from the transaction.

If Bob wants to anonymously spend private coins to fund the transaction, she will need to construct and attach as inputs a set of spend proofs for each of the outputs she wants to spend.

Tim Ruffing, Sri Aravinda Thyagarajan, Viktoria Ronge and Dominique Schrder has published a paper[11] where a denial of spending attack can be executed in the original Zerocoin protocol by simply reusing it's serial number S to create a new ZeroCoin mint. If this Serial number is marked as spent before the honest coin is spent, the honest coin will be marked as spent.

There is a need to ensure that the Serial Number is not reused to mint unlimited zObscure minted. To do so, a network of Masternodes will be used to do mathematical computations to ensure that a serial number has not been reused.[11]

An additional layer of computation is done by Bob to prove that the spent transaction is not already spent

1) When computing a coin spend proof for a transaction's input, we consider S a private key and provide the serial number's public key S instead as in:

$$S' = g^{s}\ (mod\, p)$$

2) Using a Schnorr identification protocol[12] transformed using the Fiat-Shamir heuristic, Bob will include an additional zero knowledge proof[13] :

$$ZKSoK[m]\{(S)\ :\ S' = g^{S}$$

This scheme removes an attacker's ability to reuse a serial number to mint a new coin and later proceed with a Denial-Of-Spending attack, as even if he could mint a new coin with the serial number public key S, he'd be unable to spend it without knowledge of the serial number private key S.

Further modification of the Spend algorithm is required to accommodate a new transaction's value commitment *W*.

$$W = g^{m}{g_1}^{r} = g^{m}g_{1}^{(k1x+k2)}\ (mod\, p)$$

The description of the original algorithm in [4, Appendix B] defines $\pi$ as a signature of knowledge "composed of two proofs that (1) a committed value c is accumulated and (2) that c is a commitment to S". A prover using our implementation will need to extend (1) with an extra proof of the accumulation of $\epsilon$ in V using the accumulation witness w′ , and substitute (2) with a new proof to prove in zero knowledge that he knows the secrets of both c and $\epsilon$, that $\epsilon$ commits to c as an exponent of h and that W and $\epsilon$ commit to the same transaction amount w:

$$\pi = ZKSoK[m] = \{(c, w, S, r, v, \sigma)\} :$$

$$AccVerify((N, u), A, c, w) = 1 \square AccVerify((N, u), V, \varepsilon, w') = 1 \square$$

$$S' = g^{S} \square \; c = S'h^{r} \square \varepsilon = g^{w} h^{c} g_1{}^{\sigma} \square = g^{w} g_1{}^{r}$$

As suggested by the NavCoin team, an additional layer of mathematical computation will be added to further provide cryptographic proof of the above statement

Consider the equation of *y* below:

$$y = \square^{c} \beta^{v} = \square^{(g^{s} h^{r})} \beta^{v} \; and \; Y = \square^{\varepsilon} \beta^{\S} = \square^{(g^{w} h^{\sigma} g^{c}{}_{1})} \beta^{\S}$$

which has been taken from the AccVerify Algorithm, and we take *a* and *b* as generators for a group whose order equals the modulus of the Pedersen commitment. This gives us the equation below:

$$y\prime = a^{(g^{S} h^{r})} b^{v\prime} \text{ and } \mathcal{Y}\prime = a^{(g^{w} h^{\sigma} g_1^{c})} b^{\varsigma\prime}$$

Bob will have to prove that y and y’ open to the same values as Y and Y’. In addition, based on the double discrete log proof[7], Bob will need to prove that he is capable of opening y’ , Y’ and W to fulfill additional conditions :

1) Bob will compute for each 1 < i < l:

$$p_{i}, T_{i}, \alpha_{i}, \hat{y} \square Z_{q}$$

$$\zeta_{i}, \psi_{1} \omega_{i} \square Z_{n}$$

$$t_{i} = a^{(S'h^{p_i})} b^{\omega_i}$$

$$v_{i} = a^{(a^{\tau_i} h_{i}^{y} g_{i}^{\alpha})} b^{w_i}$$

$$\mu_{i} = g^{\tau_i} g_1{}^{p_i}$$

$$k_{I} = \alpha^{Y_I} b^{\psi_i}$$

$$\omega = H(m \parallel y \parallel y' \parallel \alpha \parallel b \parallel g \parallel h \parallel g_1 \parallel W \parallel S' \parallel t_1 \parallel \ldots k_l$$

For every bit w[i] , we let:

$$\varepsilon_i = p_i$$

$$l_i = \tau_i$$

$$\delta_i = \alpha_i$$

$$\psi_I = \gamma_I$$

$$\Omega_i = \omega_I$$

$$n_i = \sigma_I$$

If and only if w[i] is equivalent to 1:

$$\varepsilon_i = p_i\Gamma$$

$$l_i = \tau_i - \omega$$

$$\delta_i = \alpha_i - \sigma$$

$$\psi_i = \zeta_i - v'h^{(p_I - r)}$$

$$v_i = y_i - c$$

$$\Omega = \omega_i - \zeta' g^{(\tau_i - \omega)} h^{(y_I - c)} g_i^{(\alpha_i - \sigma)}$$

$$n_i = \psi_i - v'$$

The proof is sent to Masternode which acts as a verifier:

$$(\omega, \xi_1, \ldots, \xi_l, \iota_1, \ldots, \iota_l, \delta_1, \ldots, \delta_l, \psi_1, \ldots, \psi_l, \nu_1, \ldots, \nu_l, \Omega_1, \ldots, \Omega_l, \eta_1, \ldots, \eta_l)$$

For every w[i], bob will check that :

$$t\prime_i = a^{(\mathbb{S}h^{\xi_i})} b^{\psi_i}$$

$$\upsilon\prime_i = a^{(g^{\iota_i} h^{\nu_i} g_1^{\delta_i})} b^{\Omega_i}$$

$$\mu\prime_i = g^{\iota_i} g_1^{\xi_i}$$

$$\kappa\prime_i = a^{\nu_i} b^{\eta_i}$$

otherwise

$$t\prime_i = y\prime^{(h^{\xi_i})} b^{\psi_i}$$

$$\upsilon\prime_i = \mathcal{Y}\prime^{(g^{\iota_i} h^{\nu_i} g_1^{\delta_i})} b^{\Omega_i}$$

$$\mu\prime_i = \mathcal{W} g^{\iota_i} g_1^{\xi_i}$$

$$\kappa\prime_i = y\prime a^{\nu_i} b^{\eta_i}$$

Bob can now calculate:

$$\omega\prime = \mathrm{H}(m||y||y\prime||a||b||g||h||g_1||\mathcal{W}||\mathbb{S}||t\prime_1||\dots||t\prime_l$$

$$||\upsilon\prime_1||\dots||\upsilon\prime_l||\mu\prime||\dots||\mu\prime||\kappa\prime_1||\dots||\kappa\prime_l)$$

The transaction is only valid if Bob can prove to a Masternode that w == w'.

A full security proof is included in Appendix A of the Zero Protocol whitepaper.

The reader might have noticed that implementing additional proofs has created a significant overhead compared to the original Zero protocol used by Zerocoin. Computations will be done by masternodes to offset this massive overhead.

Assuming that these computations are done on full nodes, Considering A the size of an accumulation proof, E the size of a discrete logarithm equality proof and C the size of a challenge used in the double logarithm proof, a transaction's input communication cost of the original protocol can be approximately denoted as

$$W = \mathbb{A} + \mathbb{E} + 2l\mathbb{C}$$

If we were to implement the additional computational steps as required by ZeroCT the equation transforms to:

$$W = 2\mathbb{A} + 2\mathbb{E} + 7l\mathbb{C}$$

As we can see, there is a multiplicative increase in cryptographic costs W, when using ZeroCT compared to the original Zero Protocol.

### *Transaction Signatures*

We substitute the public amounts from transactions with secret values hidden in the coin and spend proof commitments. The amounts being publicly verifiable is a key part of how traditional blockchains work to confirm all value transfers occur inside of a constrained money supply limit and that no user is able to spend more coins than those he proved ownership of.

For a transaction T with m inputs and n outputs we will also require the transaction fee (following strict network policies) to appear explicit as the last output at index n with transparent amount f. This output can be denoted with a special unspendable script like OP_RETURN OP_FEE.

$$\mathcal{N} = \frac{\prod_{i=0}^{m} \mathcal{W}_i}{g^f \prod_{i=0}^{n-1} \epsilon_i h^{-c_i}} = \frac{g^{(\mathfrak{w}'_0 + \cdots + \mathfrak{w}'_m)} g_1^{(r_0 + \cdots + r_m)}}{g^{(f + \mathfrak{w}_0 + \cdots + \mathfrak{w}_{n-1})} g_1^{(\sigma_0 + \cdots + \sigma_{n-1})}} \pmod{p}$$

Only if the committed values of Wi and the committed amounts in i + ϵf match using the equation below :

$$\sum_{i=0}^{m} \mathfrak{w}\prime_i - f - \sum_{i=0}^{n-1} \mathfrak{w}_i \overset{?}{=} 0$$

By using the formula below as a private key

$$\sum_{i=0}^{m} r_i - \sum_{i=0}^{n-1} \sigma_i \pmod{q}$$

## *Validating Transactions*

Assuming without the use of masternodes , Alex will scan the blockchain for all incoming transactions. When he detects a private transaction :

1. The transaction will be rejected if:
    - The range proof is insufficient
    - Fee is not explicitly stated
    - C and ϵ are not prime numbers or in the requested range
    - Transaction values is not signed by N

2. Extract b, $z_1$ and $z_2$ from his own $PK_{view}$.
3. Calculate a Diffie-Helman secret $\chi'$ using his own EC private key b and Alice's EC public key A.
4. Derive $\sigma'$ and $\Theta'$ from $\chi'$ .
5. Decode the transaction amount into w' .

$$\mathfrak{w}\prime = \mathfrak{W}\prime \oplus \varrho\prime$$

6. Reconstruct ϵ' and c':

$$c\prime = z_1^{\chi\prime} z_2 \pmod{p}$$
$$\epsilon\prime = g^{\mathfrak{w}\prime} h^{c\prime} g_1^{\sigma\prime} \pmod{p}$$

7. The transaction is valid if ϵ' and c' is equal to the value of ϵ and c supplied by bob.

## Obscure.sol

Obscure.sol will be the first cryptonote & ethereum cross chain oracle.

### | Problem |

Many cryptonote currencies act only as mere currencies and nothing more. Because the cryptonote platform does not include any smart contract virtual machine (VM) , it is hard for a cryptonote coin to be integrated into dApps.

Smart contracts will revolutionized many industries because it will totally eliminate the need for traditional legal agreements and automated digital agreements. In these traditional contracts, the performance and speed is determined by manual actions by any of the contracting parties and a layer of trust is needed to ensure that contracts are respected and executed when the conditions are met.

Unfortunately, the native ecosystem of a blockchain means it is impossible for data from external sources to enter this blockchain. This is because there is no way to communicate between blockchains and external networking systems.

Oracles serve to solve this problem by being an intermediary layer between 2 different blockchains. Currently existing oracles are centralized servers.

Obscure.sol is an oracle service that is decentralized, and supported by our Masternode network. In this paper, we will briefly describe the on-chain components that each Masternode will be implemented with to allow the Masternode to receive information from the Ethereum blockchain.

Obscure.sol does not only link Ethereum code with the Obscure chain, but all cryptonote currencies will be compatible (eg turtlecoin,monero,bytecoin).

### | Architectural Overview |

This section assumes that the reader has knowledge regarding Solidity and Web3.js.

Obscure.sol is a solution that mimics Web3.js, and is capable of interacting with the Ethereum Virtual machine (EVM) . Each Obscure.sol ''module'' is a MasterNode. Hence, a masternode is capable of making calls to the EVM which include all call,view and payable functions.

To execute a call, XSC will be used as gas. This spent XSC will then be added on top of the block reward for stakers. Our goal is to make calls cheap, without costing the user too much. We will be discussing the fees for making Obscure.sol calls in the following sections.

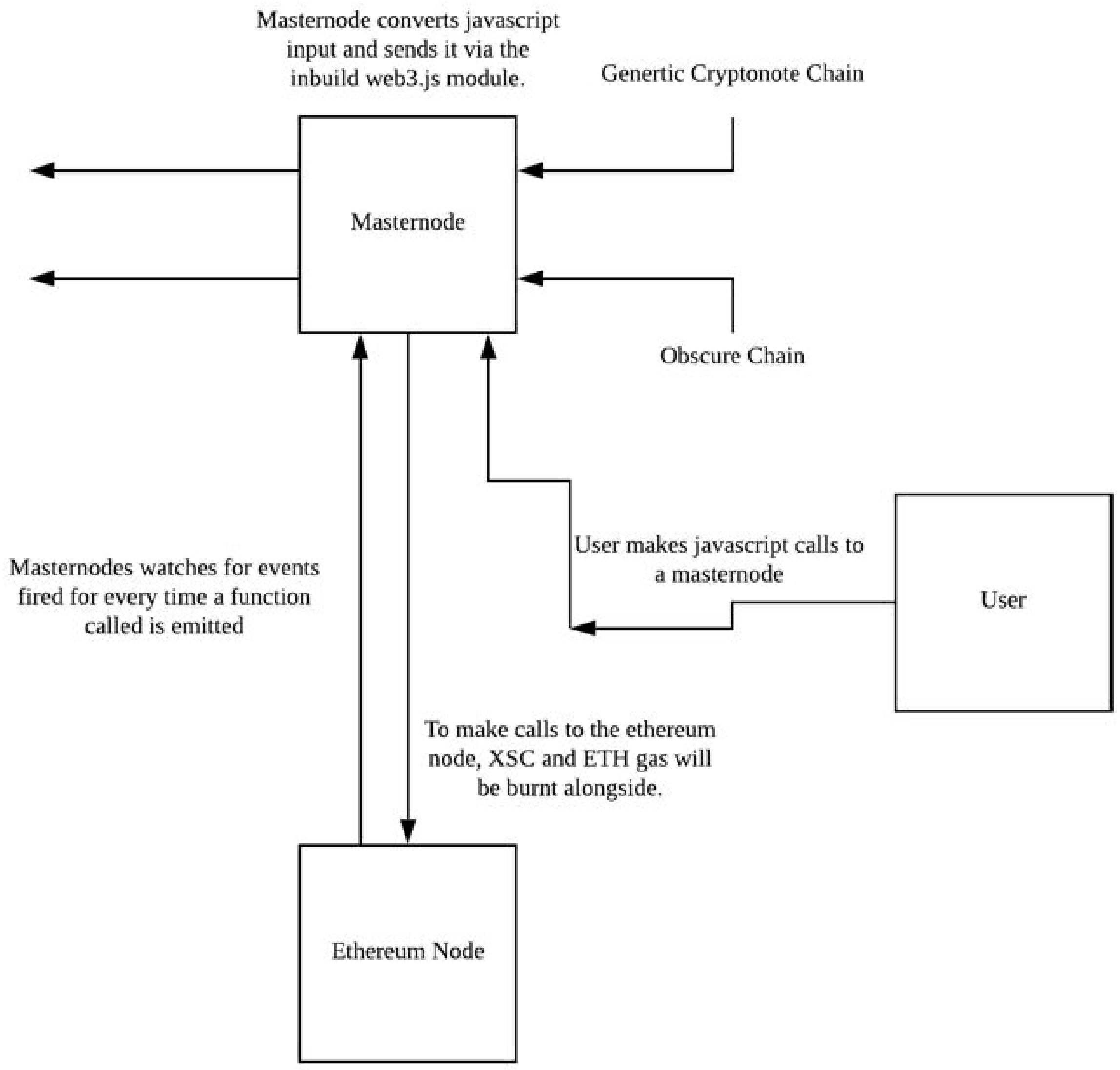
## | Implementation Flowchart |

First a masternode is running either:

1) Single daemon ( Connecting Ethereum to Obscure)
2) Double daemon (Connecting Ethereum to generic Cryptonote, with Obscure intermediary)

On top of that a modified version of Web3.js will be able integrated into each masternode. This installation is necessary.

A flowchart is shown below:


Masternode converts javascript input and sends it via the inbuild web3.js module.
Genertic Cryptonote Chain
Masternode
Obscure Chain
User makes javascript calls to a masternode
User
Masternodes watches for events fired for every time a function called is emitted
To make calls to the ethereum node, XSC and ETH gas will be burnt alongside.
Ethereum Node

As shown, a masternode requires two addresses namely an ETH address and XSC address minimally.

### | Integration with secondary coins |

There are many cryptonote coins forked from the same codebase. We can exploit this fact to allow the Obscure masternodes to actually be cross chain compatible among cryptonote coins.

Each Masternode will comprises of 3 parts:

1) The XSC Daemon
2) Generic Cryptonote Daemon
3) Link Daemon
4) Obscure.sol (Web3.js Module)

We propose this solution to search for the correct daemon integration, with Monero being the coin to be integrated.

1) A masternode is required to run the monero daemon and obscure daemon.
2) The masternode must have geth installed with the Obscure.sol SDK, which will be heavily based on the web3.js module
3) A masternode will then listen to calls made by a Monero user who wishes to make smart contract calls
4) The monero user sends XMR as gas to an address owned by the masternode. The Masternode's link daemon signs and acknowledges this transaction. This XMR is then switched on the fly XSC via linking to an exchange API.
5) A call is made using the Obscure.sol module. XSC will be burnt and sent to the masternode owner. 100% of all XSC will go to the masternode owner.

Of course there is a problem of knowing which masternode to send to. This is solved by the Obscure.sol maintaining a mapping of each ticker to a masternode address. To be added to this ticker, a masternode needs to be up 95% of the time. Any attempts to attack the chain will cause the collateral of 70,000 XSC to be forfeited and burnt.

## Role of Masternodes

Masternodes play a key role in ensuring that Obscure's implementation of ZeroCT is scalable and resistant to double spends, and denial of service attacks.

In addition also act as the oracles to connect to the Ethereum Virtual Machine, as well as providing 0-confs for all transactions signed with a zSignature.

The role of maternodes are five fold:

## | Generation of RSA Accumulators |

When a sender wants to send a private transaction, the amount of XSC will be sent to a random masternode which has proven to be up for at least 95% of the time. This masternode will act as an RSA accumulators. Recall that a ZeroCT transaction requires 2 RSA accumulators.

The first layer of privacy will be that the XSC sent to the masternode will be mixed as enforced by Obscure's compulsory mixin values of 3 after blockchain height of 100,000.

The intended receivers Private Identity, *I* , will also be received by the same masternode. Thus, using input parameters required by the ZeroCT protocol is fed to a masternode.

## | Masternode transaction construction |

A transaction will be constructed on behalf of the sender using the same mathematical proofs used by the original ZeroCT implementation.

Thus, a masternode will now be the new sender. Computation of proofs has been shifted away from a full node to a masternode to add a second layer of anonymity.

The masternode then derives a secondary RSA accumulator, which then receives the spend key from the original transaction and then mints new coins. Again, the same proofs will be required as in the original ZeroCT implementation.

## | Validation of Transactions |

When the receive receives an unconfirmed  ZeroCT transaction, the wallet-service does not guarantee the validity of the transactions. Instead, parameters that are required to confirm the validity of an incoming transactions are first extracted and then sent to a different masternode. This second masternode then attempts to ensure the integrity of the incoming transaction.

All of this happens when the transaction is still in the mempool. Hence, when a receiver receives a 'confirmed' private transaction it simply means that all verification is already done by the masternode.

## | Tracking of all minted transactions |

When a receiver mints new coins which they received, this minting process generates a coinbase transaction hash, similar to a block base reward. A masternode keeps track of all privacy coinbase hashes.

This is necessary as an additional layer of security to ensure that no minted coins are minted again in the future, although the ZeroCT implementation ensures that this is unlikely to happen.

## | Masternode collateral |

A masternode collateral of 70,000 XSC is suggested. This is subject to change.

## | Rewards |

Each masternode will claim 0.1% of all private transactions for both incoming and outcoming transactions. The first masternode where the sender feeds a transaction too generates a spend receipt that contains only 99.9% of the original coin count.

When the transaction is sent to the secondary Masternode, and additional 0.1% is claimed by the secondary masternode. The receiver will receive 99.8% of the original coin count.

On top of this, masternodes will be entitled to 20% of all block rewards, even if the block is not staked by a masternode. An example table is given below:

| Blockchain Participants | Miners | Full node | Masternode |
|---|---|---|---|
| Block Rewards (PoW) | 100% | 0% | 0% |
| Block Rewards (PoS) Staked by Full Node | N.A | 80% | 20% to masternode pool, split equally among all masternodes |
| Block Rewards(PoS) Staked by MasterNode | N.A | 0% | 100% |
| Private Transactions | N.A | 0% | 0.1% of total coin count |
| Obscure.sol | N.A | 0% | 100% of fees |

This can be subject to changes if the community decides to vote against it.

## XSC as a payments solution

The fundamental reason for choosing a 15-second block time was to make XSC a viable payments solution that is nearly instant and seamless. As explained in the beginning of this whitepaper, there are two main objectives for XSC:

1) A global, fast, and efficient P2P payment platform in the likes of Venmo & Cash app
2) The Main Settlement for the Obscure.IM decentralized market platform

### | P2P Electronic Cash Transfer |

Because transferring Cryptocurrency requires a degree of technical expertise that is often not present in the general masses, an easier method has to be created to facilitate this transfer of value.

xPay will be released on 1 April 2019, alongside the Obscure Mainnet launch.

#### *Use in Singapore*

xPay will be the first fully private P2P payment option that is available on mobile. Our goal is to make xPay a household name among all Singaporeans.

Singapore is considered one of the most affluent countries , with GDP per capita exceeding that of the United States, Germany and France.

Singapore's friendly blockchain regulatory market, a highly technical populace and the fact that there is fierce competition for the number one payment app makes Singapore the perfect viable first market.

#### *Payment space disruption*

xPay will be to the cashless payment space as to Bitcoin was to Cryptocurrencies. Main features of xPay that makes it extremely attractive for everyone to use:

1) No KYC needed to create an account
2) Create as many accounts as desired
3) On KYC-ed accounts, the ability to top up your XSC wallet via credit cards
4) Money transfers between KYC and non-KYC accounts
5) ZeroCT on all transactions providing impenetrable privacy
6) All payments will be processed in SGD , but transfers made in terms of XSC
7) NFC tap transfers between two mobiles making payments easy
8) No need to set up merchant accounts , all accounts are equal
9) Easy to integrate POS systems with xPay

We at Obscure are confident that these features of xPay will make it a successful product in singapore.

### | Obscure.IM Decentralized Marketplace |

Obscure.IM will be the world's first decentralized marketplace for **all** asset classes. Asset classes that will be available upon launch in 2019 will include but not limited to:

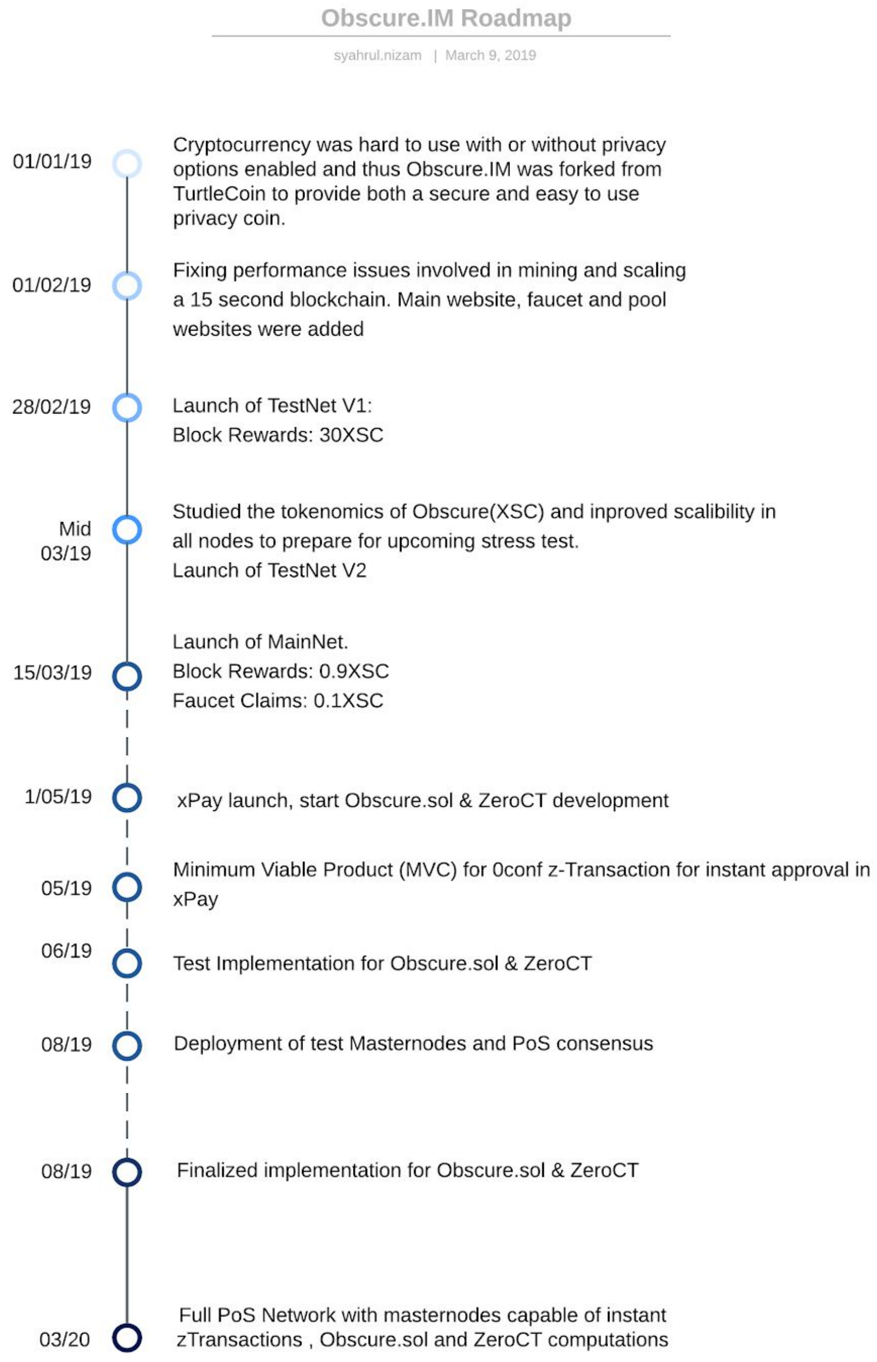
1) Property
2) Gold
3) Daily Goods & Services

Data & Asset Ownership will be encoded onto the Obscure main chain. The blockchain can be easily read without the need of an oracle. MNs will be able to expose the RPC commands specifically for Asset Ownership Transfer between addresses.

Each Asset Ownership will be considered as a special token held by an address, and movement between addresses to transfer tokens is equivalent to an XSC transfer. In addition, these Asset Ownership Tokens will hold additional JSON data payload containing information regarding the asset. This data payload can be read for free by anyone in the blockchain.

## Roadmap

Our roadmap going forward up to 2020 is displayed in the next page for viewing.



## Obscure.IM Roadmap

syahrul.nizam | March 9, 2019

| | |
|---|---|
| 01/01/19 | Cryptocurrency was hard to use with or without privacy options enabled and thus Obscure.IM was forked from TurtleCoin to provide both a secure and easy to use privacy coin. |
| 01/02/19 | Fixing performance issues involved in mining and scaling a 15 second blockchain. Main website, faucet and pool websites were added |
| 28/02/19 | Launch of TestNet V1: Block Rewards: 30XSC |
| Mid 03/19 | Studied the tokenomics of Obscure(XSC) and inproved scalibility in all nodes to prepare for upcoming stress test. Launch of TestNet V2 |
| 15/03/19 | Launch of MainNet. Block Rewards: 0.9XSC Faucet Claims: 0.1XSC |
| 1/05/19 | xPay launch, start Obscure.sol & ZeroCT development |
| 05/19 | Minimum Viable Product (MVC) for 0conf z-Transaction for instant approval in xPay |
| 06/19 | Test Implementation for Obscure.sol & ZeroCT |
| 08/19 | Deployment of test Masternodes and PoS consensus |
| 08/19 | Finalized implementation for Obscure.sol & ZeroCT |
| 03/20 | Full PoS Network with masternodes capable of instant zTransactions , Obscure.sol and ZeroCT computations |

## Acknolwedgements

Firstly, we would like to thank the developers over at Turtlecoin for making the codebase easy to develop on. Without them, the base XSC cryptocurrency and the ObscureGo wallet, along with the mining pool, would not have been possible.

Also, we would like to thank the NavCoin team, specifically Craig McGregor for providing the simplified ZeroCT in his medium post, which made it easier for the layman to understand. Special acknowledgements go to Alex Vasquez who wrote the original ZeroCT whitepaper, which Obscure has drawn inspiration to implement our own flavor of ZeroCT.

We also acknowledge the team over at Chainlink for being a pioneer in oracle technology, linking off-chain solutions to ethereum blockchain. Obscure.sol is heavily based on the concepts highlighted in the Chainlink whitepaper.

## References

1) Cawrey, D. (2015, March 19). Why Don't People Run Bitcoin Nodes Anymore? Retrieved 2019, from https://medium.com/zapchain-magazine/why-don-t-people-run-bitcoin-nodes-anymore-d4da0b45aae5
2) Decker, C., & Wottenhofer, R. (2015). Information Propagation in the Bitcoin Network. *13-th IEEE International Conference on Peer-to-Peer Computing*.
3) Babaioff, M., Dobzinski, S., Oren, S., & Zohar, A. (n.d.). On Bitcoin and Red Balloons. *Microsoft*. Retrieved from https://www.microsoft.com/en-us/research/wp-content/uploads/2012/06/bitcoin.pdf
4) https://getaddr.bitnodes.io/nodes/incentive/
5) T. (2018, June 22). Let's Talk About Mixins. Retrieved from https://medium.com/@turtlecoin/lets-talk-about-mixins-430730035297
6) McGregor, C. B. (2019, February 13). The New Privacy Protocol on the Block — ZeroCT Explained. Retrieved from https://medium.com/@craig.b.macgregor/the-new-privacy-protocol-on-the-block-zeroct-explained-b34f6885dd5
7) Distributed E-Cash from Bitcoin. http://zerocoin.org/media/pdf/ZerocoinOakland.pdf
8) Pedersen T.P. (1992) Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: Feigenbaum J. (eds) Advances in Cryptology CRYPTO 91. CRYPTO 1991. Lecture Notes in Computer Science, vol 576. Springer, Berlin, Heidelberg
9) Vazquez, A. (2019, January 17). ZeroCT: Improving Zerocoin with Confidential Transactions and more. Retrieved March 5, 2019.

10) J. Camenisch and A. Lysyanskaya, Dynamic accumulators and application to efficient revocation of anonymous credentials. in CRYPTO 02, 2002, pp. 6176.

11) Tim Ruffing, Sri Aravinda Thyagarajan, Viktoria Ronge, Dominique Schrder: Burning Zerocoins for Fun and for Profit A Cryptographic Denial-of-Spending Attack on the Zerocoin Protocol. https://www.chaac.tf.fau.de/files/2018/04/attack-cryptocur.pdf

12) Claus P. Schnorr. Efficient signature generation for smart cards. Journal of Cryptology, 4(3):239252, 1991.

13) Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. CRYPTO 1986: pp. 186-194