CCC: A **First Principles** Self **Governance** System (v1.0)

Imagine with **Misol** (https://x.com/misolcom)

Satoshi Nakamoto set the bar so high with **Bitcoin** in 2008 that we now have the term **First Principles** of **Crypto**. Unsurprisingly, the vast majority of blockchain projects since then have fallen short of his standard. **CCC** asserts that the wisdom of first principles token holders from **XEN**, e**VMPX**, and **XONE** can be harnessed through blockchain self **governance** systems.

Centralization of decision making has plagued governance systems throughout history. Instead of electing individuals, imagine using psychology to extract wisdom from the crowd. The **marshmallow experiment** is a study of **delayed gratification**. Students are offered one marshmallow and are told that if they wait to eat it, they will receive a second marshmallow later. The experiment has shown that children who choose to wait for the second marshmallow have **greater success** in life as evidenced by SAT scores, educational attainment, body mass index (BMI), and other measures.

On the **Ethereum** network, **XEN** pioneered the idea of distributing tokens based on a merger of the first principle of **free minting** (pay only gas) with the offer of additional tokens if you **delay receiving tokens**. Over a year has passed since its inception and currently we have a holder base in XEN that has endured arguably more **challenging conditions** than the original **marshmallow experiment**.

Unfortunately, it is difficult to create community driven use cases for **XEN** due to its high inflation and the fracturing of the community with the e**VMPX** and **XONE** tokens. CCC sets out to solve this by becoming a **1 billion fixed supply token,** being **100% fairly** allocated as a **free mint** (pay only ETH gas fees) to holders of **XEN**, **eVMPX** and **XONE**. A four-week hold phase records your holdings taking in account market price against the market cap valued in ETH.

1. Hold XEN, eVMPX or XONE: From **January 28, 5 pm US Central** to **Feb 25, 5:00 pm US Central**, your average token holdings (**XEN** stake included) will be recorded to calculate distribution. Only external wallets will be able to mint (see #2 below).

2. Mint CCC: From **March 3, 2024, 5 pm US Central** to **March 24, 2024, 5 pm US Central (Daylight Time),** you can mint CCC but it will not be transferable. Only external wallets can mint CCC.

3. Multiply CCC: From **March 24, 2024, 5 pm US Central** to **April 7, 2024, 5 pm US Central**, you multiply your CCC by claiming your share of unminted tokens.

4. Use CCC: The tokens become approvable and transferrable as of **April 7, 2024, 5 pm US Central**.

It's important to note that everything will be purely based on all 17 first principles of crypto as referenced on firstprinciplesofcrypto.com. Landru and Misol are guided by these principles and once the launch of CCC is done at **April 7, 2024, 5 pm US Central**, they might continue as community members with special consideration for whatever the CCC community decides through on-chain polls. CCC is an educational project with Landru and Misol taking under consideration that they might create educational resources for the community.

Dive with **Landru** (https://x.com/LandruCCC)

The first principles of crypto provide guidelines to its practitioners that avoid the pitfalls of most typical crypto projects. Many centralized exchanges have been hacked so the first principles promote self-custody. If the rules of a game are being played before the rules are known to everyone, this can result in front-running/alpha-arbitrage, which is why the first principles promote transparency. The list goes on.

Some projects follow first principles during an initial stage or with an initial idea, but the implementation or later promotions by the founder stray from the first principles.

CCC has a simple goal: to enable initial fair holders of the fair launch tokens XEN, eVMPX, and XONE, to vote and collaborate in a fashion immune from changes to the XEN ecosystem that are beginning to lean away from the first principles. The process for token distribution and use is outlined below.

The CCC token tracks timestamped holdings on-chain, so that it can be used for on-chain voting. A contract will be deployed on Ethereum which allows anyone to make a poll which can be voted on by CCC holders. Voting for the poll will be for a fixed period of time and will use a timestamp for retrieving vote weight (as of that timestamp). The timestamp for a poll will be one that is in the past relative to the time the poll is created on the blockchain (preventing participants from being able to acquire tokens that count for vote weight in a poll that has already started). After the voting period is over the final results can be queried from the blockchain to determine which selection won the poll.

A vote-tracking-disablement switch will be included in the CCC contract. This allows the vote tracking for a particular address to be disabled or enabled after it is disabled (the msg.sender must be the address that is being switched). The default condition for vote tracking will be enabled for minters but disabled for non-minters since many contracts that are not capable of being concerned with vote tracking (and hence would want it disabled if they are to interface with CCC) are also not capable of disabling it (since they don't know the function to call to disable it). The owner of an address may disable the vote tracking function for that address, which can reduce gas costs for transfer operations involving that address. They may also enable it at any time. If the vote weight is queried for an address that had vote

weight disabled for some time period, and the query is for a timestamp within that disabled period, the vote weight will be zero.

CCC cannot be minted to any address except an external wallet address. This will be enforced by the minting contract. Holdings for non-external addresses will be included in the average holdings data, and its Merkle root, for simplicity. But the minting contract will prevent them from being minted since it will check whether the origin of the transaction is equal to the msg.sender.

For tokens and stakes held by contracts, where the real owner wants to receive CCC in proportion to those tokens or staked-XEN held by a contract, the user may move those tokens and XEN stakes to be held by an external wallet address before January 28, 5PM Central, and their calculated amount of CCC mint will be unaffected (since the averaging period starts at that time). One key reason for minting to require an external wallet address is that there is a period where the tokens have been minted but cannot be transferred, so all participants gain the ability to transfer at the same time. Without the provision for external-wallets-only, control of accounts which hold minted CCC could be traded before the CCC itself can be traded, which would otherwise give participating non-external addresses (i.e. participating contract addresses) an advantage.

Staked XEN counts as held XEN. The average staked amount of a user is calculated similar to how the average held amount is calculated (see below). Note that the average amount of staked XEN for a user is totaled and rounded down to 18-zeros format before being added to the amount of average XEN holdings. In some cases this rounding reduces average total XEN amount (average staked XEN plus average XEN holdings) by a billionth of a billionth of a XEN token. The averaging calculation of course divides by the number of blocks (see below, B_e minus B_s) and this rounding down can affect the total average supply calculation (the sum of all average holding amounts across all addresses, or in the case of XEN, the sum of all average holding amounts and all average staked amounts). The effect is very small but it does affect round numbers (e.g. 500,000,000 fairly-owned XONE) to non-round numbers (e.g. 499,999,999.9999…)

The following procedures and rules are intended to describe the CCC mint allocation so precisely that they are bit-accurate so that anyone could code up the implementation, upload the data uploaded to IPFS and the contract and data can be precisely validated by the community.

1. An average price determination period (APDP, determines the average price of XEN, eVMPX, and XONE, in ETH) and average holding determination period (AHDP, determines the average amount that was held for each token, for each address), is intended to be 28 days, or 30*24*60*60 seconds, and will have the same target start time Ts, starting block B_s, ending block B_e, ending target time, Te, actual start time Tas and actual end time Tae.

a. Ts is determined at time of this whitepaper publication, 1706482800 (January 28, 5:00 pm US Central).
b. B_s is the first block ("block" and "block ID" are used interchangeably in this document) with a timestamp greater than or equal to Ts.
c. Tas is the actual start timestamp, which is the timestamp of B_s.
d. Te is Ts + 28*24*60*60. B_e is the last block with a timestamp less than or equal to Te.
e. Tae is the actual end timestamp, which is the timestamp of B_e
f. In the case that any XONE is transferred from/to the "XONE owner address", or XOA (i.e. the address holding 500m XONE as of this writing, which is 0xC73Fc08C931Efe3FCE850C09278472e8a81c2e05), If the APDP and AHDP have not started yet, then Xone average holding amount is instead calculated as a snapshot that is instantaneous at the time of the transaction right before the transfer transaction ("transfer-containing-owner-address transaction", or TCOAT) containing the XOA. Similarly the market cap of XONE will be the snapshot price, in the most recent swap of the XONE-ETH 1% Uniswap v3 liquidity pool, prior to the TCOAT. If the APDP and AHDP have started and a block containing TCOAT, call it B_TCOAT, is greater than or equal to 1 + B_s, then the average holding amount and average price for XONE are computed as if B_e is B_TCOAT except that all transactions at or after TCOAT are treated as not present in B_TCOAT (i.e. calculations do not include transactions after TCOAT, nor do they include TCOAT). In fact the ignoring of certain transactions within B_TCOAT is unnecessary to state since, as may be seen in subsequent procedures and calculations, such transactions would have zero duration, since they are in the B_e being used for XONE, and so don't affect average price or average holding amount. The calculations of APDP and AHDP are not affected by the TCOAT.
g. In post processing of the average holding amount for each address, in the case of tracking average holding amounts, the 0 address is always deleted, of course (it ends up with negative balance and is not a real account). The XOA is also deleted.

2. During the average price determination period (APDP), a time-weighted price is calculated, in ETH, for each of XEN, eVMPX, and XONE, under the following rules. The resulting values of APDP will be loaded into the CCC contract. The average price of the token (XEN, eVMPX, or XONE) is determined by multiplying the price of each token multiplied by the number of blocks that it was at that price, divided by the total blocks of the APDP. Note that only the closing price of a block is used (which provides some protection from flash loans).
    a. When processing swap events, if the filter for events (used to fetch events from the an Ethereum Provider) is such that the earliest block for the filter is B_s, but no relevant swap events exist in B_s, then the price is not known at B_s which is necessary for the calculations. In that case, to determine the starting price of B_s, some search for the most recent swap that occurred before B_s must be

done. One algorithm for this is to find a block about 24 hours before B_s, and use a filter for events from that block up to (and including) B_s-1. The last event returned with that filter is the starting price for block B_s. If no events are returned with that filter, use a filter for events that chooses a starting block 48 hours before B_s, and so-on until a swap event is found. To date very few if any periods for any of the 3 tokens xen/vmpx/xone have gone 24 hours without a swap. If an error is returned due to too many swap events being returned from the chosen filter, increase the start of the filter by e.g. a quarter its distance to B_s, or so, until no such error occurs (unlikely as the limit is typically 10,000 events and the tokens generally do not ever have 10,000 swaps in 24 hours). This all is quite intuitive for manually finding such a transaction but we specify the above details to highlight some of the programming requirements necessary to handle arbitrary levels of swap activity.

b. Note that "block durations", rather "seconds durations" are used for the inner loops of this calculation as well as the average holding calculation below, for simplicity. Fetching timestamps for each block with standard web3 APIs could be extremely bandwidth intensive since it may require fetching the data for an entire block just to get the timestamp data.
c. The algorithm for determining the average price of a token continues as follows. To calculate the average price of a given token, initialize by setting the most recent price MRP to the initial price IP which is the closing price as of B_s, which is either the price of the last swap emitted from B_s, or the most recent price in a previous block as calculated in step 2a. Set the most recent swap block MRSB to B_s.
d. Initialize sum prices SP to (B_e – B_s)*MRP. This is the duration of the analysis period, in blocks (B_e - B_s) multiplied by most recent price (MRP).
    i. For each swap event that gives a price of the token, processing events in order from blocks B_s + 1 to B_e -1, inclusive, where the block of the current event being processed is B_c, the price of the token (XEN/ eVMPX/XONE, priced in ETH) as priced in the in the current event is P_c ("price current"),
        1. The price difference PD is P_c – MRP.
        2. Set Sum += (B_e-B_c)*P_c
        3. Set MRP equal to P_c
    ii. Finally, the average price is SP / (B_e – B_s), which can be done with integer division (truncation) since the prices are in 18-decimals format.
    iii. Note that prices in swaps are encoded as square roots with 96-bit decimals that must be handled properly to generate prices like IP and P_c in 18 decimal format. The price is calculated from sqrtPriceX96 as first squaring the value, multiplying by 10**18, and then dividing by 2**192, with integer (truncating) division. Python integers or an alternative large-integer library should be used for this calculation as well as most other calculations described in this document. This document

generally assumes values are represented in 18 decimal format using integers compatible with large numbers. The document does not go into detail but it is assumed to be understood when to pre-multiply by e.g. 10**18 (before a division) or post-divide by 10**18 (after a multiplication) which is a required part of calculating using such an integer representation.

e. The average price avg_xen, avg_vmpx, and avg_xone, for each of XEN, eVMPX, and XONE respectively, are loaded into the CCC contract after APDP but before the first mint (see below).

3. An average holding determination period (AHDP), is the same period as the APDP.
    a. The average token amount of an address ATA(addr) is calculated as the sum of the token balance of the address at the end of each block in the AHDP (every block whose ID is greater than or equal to B_s and less than B_e) divided by the number of blocks counted in APDP (B_e – B_s).
    b. An algorithm to efficiently compute ATA(addr) starts by computing an initial table "Ti(addr)" of all addresses and their holdings as of the end of block B_s. Next, the final table T_f is initialized such that for every address the balance "T_f(addr)" is set to Ti(addr)*(B_e – B_s). Then all transfer events are processed in order, where current block B_c of the current event increases from as low as B_s + 1 inclusive, to as much as B_e – 1, inclusive. For every transfer, with "from" address A_f, and "to" address "A_t", with "amount" amt:
        i. T_f[A_f] -= amt*(B_e-B_c)
        ii. T_f[A_t] += amt*(B_e-B_c)
    c. After processing all B_c, the next step is:
        i. For every entry in T_f, divide by B_e-B_s with ordinary integer division (truncation rounding). Since token amounts are in 18-zeros for all the tokens, XEN, eVMPX, and XONE, such rounding is reasonable and congruent with solidity/blockchain math.
    d. For XONE, see previous discussion regarding special cases regarding XOA and TCOAT.
4. Chunking of the snapshot data:
    a. Name each of the final T_f tables T_fxen, T_fvmpx, and T_fxone for each of the tokens.
        i. The process above is also performed for Staked and Withdrawn events of the XEN contract, to calculate average staked XEN for each address. The final average result for an address is added to that address's entry in T_fxen. Note for implementers that one simplification that can be understood is that a Staked Event always has one or zero Withdrawn events that follow it. This is because the XEN staking contract only allows one stake per address, and a withdrawal always removes all of the stake. This simple implementation of XEN staking removes the need to sort the two types of events together to figure out which Withdrawn event corresponds to which Staked event.

b. For each address that has any positive amount in T_fxen, T_vmpx, or T_fxone (not the 0 address which should be negative, and excluding XOA as previously mentioned), a 512-bit entry (64 bytes) is constructed with 160-bits of address, followed by 96-bits covering the XONE amount, 128-bits covering the xen amount, and 128-bits covering the eVMPX amount, and this entry is appended to the master holding list MHL. The list is sorted by address (in lowercase) in increasing address order and the list. A chunking method may be employed since the list is anticipated to be at least 13 Megabytes. In any case the full file without chunking will be uploaded for simpler verification. Smaller chunks may be uploaded to allow minters to download a smaller piece of data to create the Merkle proof which they must present to the CCC contract to mint.
    i. The Merkle Tree, whose hash will be uploaded to the CCC contract prior to the minting period start, uses the following value as the hash of an empty leaf:
        1. empty_leaf_hash = 0x0123456789abcdeffedcba98765432100011223344556677889 9aabbccddeeff
        2. empty_hash[0] = empty_leaf_hash
    ii. Empty entries at higher levels empty_hash[i+1] in the tree are calculated as keccak256(abi.encodePacked(empty_hash[I], empty_hash[I])
    iii. Otherwise the hash of nodes in the tree are keccak256(left, right). The hash of a non-empty leaf is the keccak256 hash of the 512-bit MHL entry value previously described (using encodePacked).
c. The data is uploaded, e.g. to IPFS, and a link to the data and possibly other relevant data to navigate smaller chunks of the data (such as first address represented in a chunk, and a link to the smaller chunk, as well as the index in of the chunk in the larger set of data, and the power-of-two size of the chunk) may also be included if optimization is done to reduce minting download burden. It can be noted that it has been tested that javascript in a browser does not currently strictly need an optimization and currently the entire file can be processed in browser javascript to generate the proofs required to mint.

5. The sum token amounts stored in the MHL, sum_xen, sum_vmpx, and sum_xone for each token XEN/eVMPX/XONE respectively, is totaled and stored in the CCC contract.
6. During the mint period, people with wallets with token amounts that have been captured in the MHL and can be proven by the Merkle Root hash loaded into CCC contract, can mint a number of CCC roughly equivalent to mint1_CCC, which is derived in the following manner (subject to rounding errors inherent in computers) to target a max supply of 1B tokens (if every outstanding token were minted).
    a. Total average ETH value before scaling TAEVBS = (avg_xen * sum_xen + avg_vmpx*sum_vmpx + avg_xone*sum_xone)
        i. Each is stored in 18-zeros format and fixed point math is done as normal in solidity.

b. During first mint, a user with a given address, has its XONE, XEN, and eVMPX amounts (amt_xone, amt_xen, and amt_vmpx respectively) loaded from the relevant chunk (e.g. by using links stored in the contract, and loading from ipfs). The amount of CCC minted in the first mint CCC1[addr] is:
    i. 1,000,000,000 * (amt_xone*avg_xone + amt_xen*avg_xen + amt_vmpx*avg_vmpx) / TAEVBS
    ii. The total amount of CCC minted in the first mint TCCC1 is tracked
    iii. Note the attempted target supply of approximately 1 billion if every address in every chunk performs a first mint. Since some of the addresses in the MHL contract will be contract addresses, which can't mint, it is expected for the total supply reached during the first mint to be less than one billion. Hence, there is a second period for multiplying mint amounts which require users to do a second transaction.
c. The mint transaction receives the Merkle proof and the token amounts. The address is the msg.sender which must also be the tx.origin (i.e. external wallet address, as previously discussed).

7. The time period for completing the second transaction (Multiply CCC) is 14 days.
    a. The additional mint for an address, CCC2[addr] is the following formula:
    b. CCC2[addr] = CCC1[addr] * (1,000,000,000 – TCC1) / TCCC1
    c. Therefore total minted for an address is CCC1[addr] + CCC2[addr]
    d. This second mint gets the final supply much closer to the targeted 1 billion supply because people who minted in the first phase are more likely to mint in the second phase and if they all do perform this second mint, the final supply is very close to 1 billion with some small rounding errors.
8. After the time period for Multiply CCC completes, then CCC becomes transferable and approvable.

CCC Launch Steps 1-4 with Ethereum Timestamps:

1. Hold XEN, eVMPX or XONE: From **January 28, 5 pm US Central** to **Feb 25, 5:00 pm US Central**, your average token holdings will be recorded to calculate distribution. (First Ethereum block with timestamp greater than or equal to 1706482800 through last Ethereum block with timestamp less than or equal to 1708902000)

2. Mint CCC: From **March 3, 2024, 5 pm US Central** to **March 24, 2024, 5 pm US Central (Daylight Time),** you can mint CCC but will not be tradable. Only external wallets can mint CCC. (First Ethereum block with timestamp greater than or equal to 1709506800 through last Ethereum block with timestamp less than or equal to 1711317600; these timestamps are correctly adjusted for daylight time change)

3. Multiply CCC: From **March 24, 2024, 5 pm US Central** to **April 7, 2024, 5 pm US Central**, you multiply your CCC by claiming your share of unminted tokens. Only external wallets can Multiply CCC. (First Ethereum block with timestamp greater than

or equal to 1711317600 through last Ethereum block with timestamp less than or equal to 1712527200

4. Use CCC: The tokens become approval and transferrable as of **April 7, 2024, 5 pm US Central**. All CCC tokens only become tradable at the end. (First Ethereum block with timestamp greater than or equal to 1712527200)

Cat Church LLC & CCC Token **Disclaimer**

No further development by members of Cat Church LLC will be performed in their capacity as founders, after the token launch. The founders will be equal members of the community at that time and all members will be able to shape the use of the token as equals after the launch. There should be no expectation that any managers or founders will develop additional uses cases beyond the launch use cases after launch. No such commitment or promises are being made.

The founders are not selling CCC, it is free to mint, subject to the normal transaction fees on the Ethereum blockchain. The founders do not promote purchasing the token on the open market if the token is made available by some owners for sale. The founders do not believe the token will appreciate and it should not be considered an investment.

This whitepaper is intended for informational purposes only and does not constitute a guarantee of the future performance or value of CCC. The contents of this document are not to be construed as legal, business, investment, or tax advice. Potential token holders are advised to conduct their own due diligence and consult with professional advisors for any legal, tax, accounting, or investment advice.

The issuance of CCC is intended to be in compliance with applicable laws and regulations. However, the regulatory status of digital tokens is subject to significant regulatory uncertainty and may change. Potential token holders are advised to familiarize themselves with relevant legal and regulatory constraints in their own countries of residence.

Involvement with digital tokens involves a high degree of risk, including but not limited to market volatility, regulatory changes, and technology risks. Potential token holders should be prepared to sustain a total loss of any price or costs they may have paid for any tokens.

While every effort has been made to ensure that the information set forth in this document is accurate as of the date hereof, Cat Church LLC makes no warranties or representations as to the accuracy, reliability, or completeness of this document and disclaims any liability for any errors or omissions.

This whitepaper does not constitute an offer, solicitation, or sale of CCC in any jurisdiction where such offer, solicitation, or sale would be unlawful under the securities laws of such jurisdiction.

Residents of certain jurisdictions may not be eligible to purchase or hold CCC due to legal restrictions. It is the responsibility of potential token holders to ensure compliance with their local laws and regulations.

This document may contain forward-looking statements which are based on Cat Church LLC's current expectations and projections about future events. These forward-looking statements are subject to risks, uncertainties, and assumptions about CCC, and there are important factors that could cause actual outcomes to differ materially from those expressed or implied by such statements.

Cat Church LLC reserves the right to modify, amend, or update this document and its contents at any time and without prior notice. The most current version of this whitepaper, as made available by Cat Church LLC, will supersede all previous versions of this document.