



TECHNICAL WHITEPAPER

ICGold

A fair-launch cycle-burn mining protocol with ICP
revenue sharing for stakers on the Internet Computer

VERSION 1.0 · OCTOBER 2026

TOKEN: ICG

ABSTRACT

ICGold is an application-layer mining protocol on the Internet Computer (IC). It issues a new token, ICG, on a fixed Bitcoin-style schedule: 50 ICG per block, a target interval of ten minutes, and a halving every 210,000 blocks. Issuance is bounded below 21 million ICG. There is no premine and no allocation to the team or investors. The 50 ICG genesis reward is recorded but never minted.

Participants mine by irreversibly burning *cycles*, the resource the Internet Computer charges for computation. They obtain the cycles by depositing ICP. After two ledger fees, every deposit is split by code: 40% is converted into cycles for mining, 42% is retained as ICP and distributed to ICG stakers, 15% funds an ICG buyback, and 3% is a protocol fee that pays first for the automatic cycle maintenance of the protocol's own canisters. The buyback first builds liquidity for ICG that the protocol owns for good, then buys ICG and burns it, in small steps protected against price manipulation. At each block, one participant is selected with probability proportional to the cycles they burned since the previous block. Randomness comes from the IC's threshold random beacon, and the selected participant receives newly minted ICG.

The design separates the mined asset (ICG) from protocol revenue (ICP). Stakers lock ICG for 7 to 365 days and receive a share of real deposits, weighted by the length of their commitment, rather than inflationary emissions. The protocol funds its own infrastructure and runs without off-chain operators. This paper specifies the monetary policy, the block-selection rule, deposit accounting, staking mathematics, the buyback, transfer-safety guarantees and the pause semantics. It also states which administrative powers exist at launch, how governance is planned to move to a DAO, and where the protocol is headed.

< 21M MAXIMUM ICG SUPPLY (≈ 20,999,950 MINTABLE)	50 → halving ICG PER BLOCK; HALVES EVERY 210,000 BLOCKS (≈ 4 YRS)	40 / 42 / 15 / 3 % OF EACH ICP DEPOSIT: MINING / STAKERS / BUYBACK / PROTOCOL FEE	0% PREMINE, TEAM OR INVESTOR ALLOCATION
--	--	--	--

DOCUMENT STATUS

This paper describes the **reference implementation** (the ICGold minter, miner and buyback canisters) as of October 2026. Every parameter stated as implemented is taken from that code. Section 17 describes future directions and is marked **PLANNED**. It is not part of the current protocol.

IMPORTANT NOTICE

This document is a technical description of software whose source code is planned to be published as open source (Section 17). It is not an offer or solicitation to buy or sell any asset, and it is not investment, legal, or tax advice. ICG carries no claim on any issuer, commodity, or revenue stream beyond what the protocol code executes. Staking rewards depend entirely on future deposit activity and may be zero. Mining can cost more than the value of the ICG it yields. Statements about planned features and governance describe intentions, not commitments. Smart-contract, governance, and market risks are described in Sections 14–16. The word "Gold" refers to the fixed-supply "digital gold" monetary model and does not imply any physical backing.

CONTENTS

1	Introduction	4
2	Why ICGold	5
3	Design Principles and Non-Goals	6
4	System Overview	7
5	The ICG Token	9
6	Monetary Policy	9
7	Mining: Cycle-Burn Block Selection	11
8	ICP Deposits and Revenue Allocation	14
9	ICG Staking and ICP Rewards	16
10	Protocol Treasury and Canister Maintenance	18
11	ICG Buyback: Protocol-Owned Liquidity and Burn	19
12	Protocol Time and Emergency Pause	21
13	Transfer Safety and Reconciliation	22
14	Governance and Controllers	23
15	Economic Analysis and Risks	24
16	Security Considerations	25
17	Future Directions	26
18	Dependencies	27
19	Conclusion	28
	References	28
A	Parameter Registry	29
B	Public Interface Summary	31
C	Glossary	31

1 Introduction

Bitcoin showed that a digital asset can be scarce without an issuer. Two ingredients make that possible. The first is an issuance schedule fixed in the protocol, which no administrator can change. The second is a costly, unforgeable expenditure that decides who receives each new unit [1]. Proof-of-work makes that expenditure physical: energy is spent on hashing, and the network verifies the result cheaply.

The Internet Computer has no hashing race to join. Its consensus is provided by subnets of nodes using threshold cryptography. It does have a native, metered, and non-refundable resource: *cycles*. Canisters, the IC's smart contracts, pay cycles for computation, storage, and messaging. Cycles are created only by converting ICP through the NNS Cycles Minting Canister (CMC), at a rate pegged to the IMF Special Drawing Right (XDR): one trillion cycles per XDR [3][4]. A canister can also destroy cycles on purpose through the system API `cycles_burn128`. Burned cycles go to no one, and the call returns the exact amount destroyed [7].

ICGold uses cycle burning as its costly expenditure. Mining by burning cycles was first put forward as a model on the Internet Computer by the BOB project [11]; ICGold builds on that idea and turns it into a complete economic system. Participants deposit ICP. 40% of each deposit is converted into cycles, and dedicated miner canisters burn them. At roughly ten-minute intervals the protocol awards one block of newly issued ICG to one participant, chosen with probability proportional to the cycles burned in that interval. Issuance follows Bitcoin's arithmetic exactly: 50 units per block, halving every 210,000 blocks.

ICGold departs from a pure burn protocol in one important respect. The rest of each deposit is not burned. 42 percentage points are retained as ICP and distributed to users who stake ICG, and 15 points buy ICG back from the market. Mining demand therefore becomes revenue for holders, paid in a different asset from the one being mined, and steady demand for ICG itself. Stakers do not receive newly created ICG; they receive a share of the ICP that miners actually paid.

ICGold consists of a central minter canister, per-user miner canisters and a pooled virtual miner. Around that architecture it defines its economics and safety model:

- a new fair-launch token with a Bitcoin-style schedule and an unminted genesis reward;
- a coded split of every ICP deposit, with ICP revenue shared with stakers;
- lock-based staking in which longer commitments earn a larger share;
- a protocol fee that keeps the protocol's own canisters funded, with no off-chain operator;
- a rule-based buyback that builds protocol-owned liquidity and then buys and burns ICG;
- durable, idempotent handling of every ledger transfer, with on-chain reconciliation of uncertain outcomes;
- a protocol clock that stops during an emergency pause, so time-based entitlements are not lost.

ICGold is not a consensus protocol. Ordering, replication, and finality are inherited from the Internet Computer. The guarantees described here are application-level. They hold only as long as the deployed canister code matches the published implementation, a condition discussed in Section 14.

2 Why ICGold

ICGold combines mechanisms that usually live in separate protocols: fair-launch mining with a fixed schedule, real-yield staking, a rule-based buyback with liquidity the protocol owns, and infrastructure that pays for itself. This section explains what each contributes and what the combination offers.

2.1 Useful burn instead of wasted work

Proof-of-work makes a coin costly by spending energy outside the network; the work secures the chain but produces nothing else. ICGold's cost is burned cycles. Cycles exist only because ICP was converted through the NNS Cycles Minting Canister, which removes that ICP from circulation. Every ICP a miner spends therefore contributes to the same deflationary mechanism that pays for all computation on the Internet Computer, and the cost of mining is verified exactly by the system API rather than estimated. The concept of cycle-burn mining on the Internet Computer was introduced by the BOB project [11]. ICGold adopts it and adds what BOB did not have: revenue for holders, a buyback, self-funding infrastructure and staged governance.

2.2 Productive stake

In most proof-of-stake systems, staking rewards are newly issued tokens: holders are paid in dilution. ICGold stakers are paid in a different asset, ICP, taken from real deposits. Their share, 42% of every deposit, is weighted by commitment: a stake locked for a year earns twice as much per ICG as one that can leave soon. Stake therefore does useful work for the protocol: it absorbs the ICG that mining issues and holds it for long periods, and it is paid from actual activity rather than inflation.

2.3 Store-of-value properties

- **Fixed, predictable supply.** At most 20,999,949.9769 ICG can ever exist, issued on Bitcoin's schedule with a halving about every four years. No endpoint can change it.
- **Fair distribution.** No premine, presale, team or investor allocation. Every ICG was won through mining.
- **Supply sinks.** The buyback burns ICG once liquidity is deep; emergency exits burn half of the exiting stake; ICG sent to the minting account is destroyed.
- **Permanent liquidity.** The buyback's liquidity position belongs to the protocol and cannot be withdrawn, so market depth cannot be pulled.
- **Long-term holding.** Locks of up to 365 days, with a countdown as long as the lock, keep a share of supply out of circulation.

- **Full transparency.** Supply, stakes, reserves and every flow can be verified on-chain at any time.

These are structural properties of the protocol. They do not guarantee any market value.

2.4 Self-sustaining by design

ICGold needs no operator to keep it running. Timers inside the canisters produce blocks, burn pool fuel, move the buyback share and run the buyback. The 3% protocol fee funds the cycles of the minter, the ICG ledger and the buyback canister automatically (Section 10), and the web application is served from an asset canister on the Internet Computer. If every contributor stepped away, the protocol would continue to mine, pay stakers, buy back ICG and fund its own infrastructure for as long as deposits arrive.

2.5 What is distinctive

To our knowledge, no other protocol on the Internet Computer combines all of the following: cycle-burn mining with a fixed monetary schedule; staking paid in ICP from real deposits and weighted by commitment; automatic funding of its own canisters; a buyback that first builds liquidity the protocol keeps for good and then buys and burns, with protection against manipulation (Section 11); ICRC-21 consent messages, so wallets show in plain words what every call does; and a published plan to hand control to long-term stakers (Section 17).

3 Design Principles and Non-Goals

3.1 Principles

1. **Fair launch.** No premine, presale, team allocation, or investor allocation. Every ICG in circulation was won through mining. The genesis reward is recorded for accounting transparency but is never minted.
2. **Policy in code.** The block reward, the halving interval, the deposit split and the buyback rules are compile-time constants. No administrative endpoint can change them. Changing them would require a code upgrade (Section 14).
3. **Real, verifiable cost.** Mining weight equals the cycles actually destroyed, as reported by the IC system API. Weight cannot be borrowed, recycled, or reused.
4. **Revenue in a distinct asset.** Staking rewards are paid in ICP from actual deposits, not from additional ICG issuance.
5. **Explicit accounting.** Mining funds, staker liabilities, treasury funds, the buyback reserve and unassigned deposits are tracked separately. None can be spent as another.
6. **Safe external calls.** Every ledger transfer carries a durable deduplication identity. An interrupted or ambiguous call can be retried without paying twice.
7. **Disclosed trust.** Every privileged capability that exists at launch is named, together with its limits.

3.2 Non-goals

- ICG is not a stablecoin and targets no price. The buyback follows fixed rules: it buys in small steps and pauses when the price moves far from its average (Section 11). It never defends a price level.
- ICG is not backed by or redeemable for gold, ICP, or any other asset.
- Staking rewards are not a promised yield. They are a weighted share of whatever deposits occur.
- ICGold does not replace or modify Internet Computer consensus, subnet security, or NNS governance.

4 System Overview

The protocol consists of one central canister, the *ICGold minter*, which interacts with standard Internet Computer infrastructure and with a set of miner canisters that it creates. Figure 1 shows the components and the main value flows.

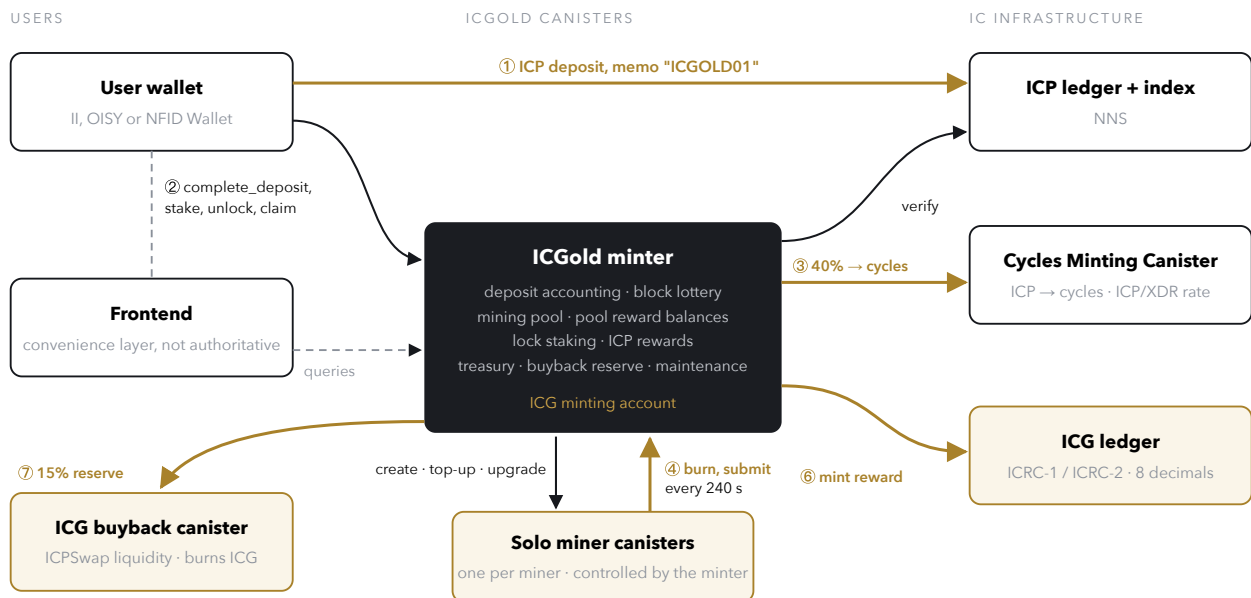


Figure 1. Components and principal flows. ① The user sends ICP to the minter’s deposit account. ② The user asks the minter to process it; the minter verifies the ICP ledger block and keeps the staker share and the protocol fee in its own accounts. ③ The minter converts the mining share to cycles through the CMC. ④ Miner canisters burn cycles and report the amount. ⑤ The minter selects a block winner (internal, not drawn). ⑥ The minter mints the block reward on the ICG ledger, or credits pool members who later claim it. ⑦ The buyback canister draws the 15% buyback reserve and spends it on ICPSwap (Section 11). Gold arrows carry value; dashed arrows carry none.

COMPONENT	ROLE
ICGold minter	Validates and splits deposits, runs the block lottery, operates the mining pool and its reward balances, pays block rewards, keeps staking accounts, holds the treasury and the buyback reserve, performs maintenance top-ups, and enforces the pause. It is the ICG ledger’s minting account: every ICG in existence was minted by it as a block or pool reward.

COMPONENT	ROLE
Miner canisters	Created by the minter from a miner WASM embedded in the minter. The minter is their controller. A miner's owner can adjust only its burn rate per round and can request an upgrade to the minter's embedded code. Miner ownership is immutable.
ICG ledger	The official DFINITY ICRC ledger WASM, installed with symbol <code>ICG</code> , name <code>ICGoId</code> , 8 decimals, zero initial balances, ICRC-2 enabled, and the minter as minting account.
ICP ledger and index	NNS canisters. Deposits are ordinary ICP transfers, and the minter verifies each one through the ICP index before accepting it. The index is also used to prove that an uncertain transfer never executed (Section 13).
Cycles Minting Canister	Converts ICP to cycles at the NNS-maintained ICP/XDR rate. It serves both the mining share and maintenance top-ups.
ICG buyback canister	Draws the buyback reserve through an allowance the minter grants and spends it on the ICG/ICP pool of ICPSwap: first building a liquidity position it keeps for good, then buying ICG and burning it (Section 11). It has no function that sends ICP, ICG or its liquidity to anyone.
Frontend	A static web application served from a certified asset canister. Users sign with Internet Identity, OISY or NFID Wallet (Google or email). It holds no authority. All state it displays can be read directly from the canisters.

4.1 Protocol accounts

ACCOUNT	ASSET	PURPOSE
Minter, default subaccount	ICP	Deposit address. The 42% staker share stays here as the reward treasury.
Minter, subaccount 0x02...02	ICP	Protocol treasury. Receives the 15% buyback share and the 3% protocol fee in one transfer. The buyback share is moved out once a day; the fee pays canister maintenance first. Above a floor (default 1 ICP), treasury funds can be sent only to the published development account.
Minter, subaccount 0x03...03	ICP	Buyback reserve. Receives the 15% buyback share. Only the ICG buyback canister can draw from it, through an allowance, and only to buy ICG.
Minter, subaccount 0x01...01	ICG	Staking custody. Holds all staked ICG and ICG waiting to be withdrawn.
Minter, default subaccount	ICG	The ICRC minting account. Transfers out of it mint ICG; transfers into it burn ICG.
ICG buyback canister	ICP / ICG / liquidity	ICP drawn for the next step, its liquidity position in the ICPSwap pool, and ICG waiting to be burned.
Published development principal <code>df4xj-...-2ae</code>	ICP / ICG	Receives voluntary donations and treasury transfers above the floor. It has no protocol powers.

5 The ICG Token

ICG is a standard fungible token on an ICRC ledger [5][6]. The ledger is the single source of truth for ICG ownership. Internal records exist only for staked positions, unclaimed pool rewards and pending payouts, and they reconcile against the ledger's custody subaccount and minting history.

PROPERTY	VALUE
Symbol / name	ICG / ICGold
Decimals	8 (1 ICG = 10^8 base units, written "e8s")
Standards	ICRC-1 transfers and metadata; ICRC-2 approve / transfer_from (required for staking); ledger transaction log with archives
Initial balances	Zero. There are no genesis accounts.
Minting account	The ICGold minter. Minting is possible only through block and pool rewards (Sections 6 and 7).
Transfer fee	A ledger parameter set at installation (the reference deployment uses 0.01 ICG). All protocol code and the frontend read it from the ledger at runtime instead of assuming it.
Maximum supply	Bounded by the emission schedule, not by a ledger cap: at most 20,999,949.9769 ICG can ever be minted.

USER CAUTION

Under ICRC semantics, a transfer *to* the minting account burns the tokens. ICG sent to the minter's default account is permanently destroyed. Users stake only through `stake_icg`, never by sending ICG directly to the minter.

6 Monetary Policy

Blocks are numbered from height $h = 0$. The reward for the block at height h , in e8s, is

$$R(h) = \lfloor 5 \times 10^9 / 2^{\lfloor h / 210\,000 \rfloor} \rfloor \quad (1)$$

so 50 ICG for heights 0–209,999, 25 ICG for the next 210,000 blocks, and so on. After 33 halvings the integer reward reaches zero, and issuance ends at height 6,930,000. The reward depends only on block height. No endpoint can change it, and no quantity observed on-chain alters it.

Genesis. Block 0 records the full 50 ICG reward, but it is assigned to the anonymous principal and no ledger transfer is ever made. As with Bitcoin's genesis coinbase, those 50 ICG can never enter circulation. The participant selected for block 0 receives nothing.

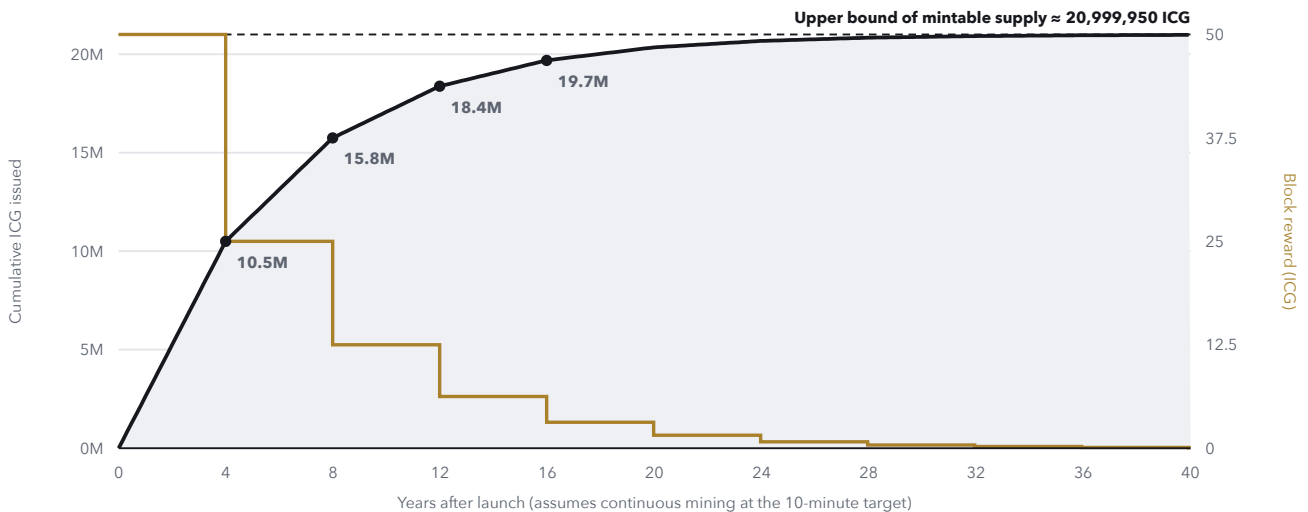


Figure 2. Cumulative ICG issuance (black, left axis) and block reward (gold, right axis). The calendar axis assumes blocks are produced continuously at the 600-second target. Blocks are not produced while the protocol is paused or when no cycles are burned (Section 7.3), so real calendar dates can only be later than shown.

EPOCH	HEIGHTS	REWARD (ICG)	EPOCH ISSUANCE	CUMULATIVE	SHARE OF TOTAL	≈ YEARS
0	0 – 209,999	50	10,500,000*	10,500,000	50.00%	0 – 4.0
1	210,000 – 419,999	25	5,250,000	15,750,000	75.00%	4.0 – 8.0
2	420,000 – 629,999	12.5	2,625,000	18,375,000	87.50%	8.0 – 12.0
3	630,000 – 839,999	6.25	1,312,500	19,687,500	93.75%	12.0 – 16.0
4	840,000 – 1,049,999	3.125	656,250	20,343,750	96.88%	16.0 – 20.0
5	1,050,000 – 1,259,999	1.5625	328,125	20,671,875	98.44%	20.0 – 24.0
...	...	...	...	...	...	...
32	6,720,000 – 6,929,999	0.00000001	0.0021	20,999,999.9769	100%	≈ 131.8
Scheduled total, less the unminted 50 ICG genesis reward			20,999,949.9769			

* Includes the 50 ICG genesis reward, which is recorded but never minted. One epoch is $210,000 \times 600 \text{ s} \approx 3.99$ years.

6.1 Issuance rate

At the 600-second target, the protocol produces about 144 blocks per day. During epoch 0 that is about 7,200 ICG per day, or about 2.63 million ICG per year. The rate halves with each epoch. All issuance goes to miners. No other issuance path exists in the code: no staking emissions, treasury mint, or discretionary mint.

6.2 Issuance does not depend on burn volume

Bitcoin adjusts difficulty to hold block time constant as hash power changes. ICGold needs no difficulty adjustment, because the block interval is set by the protocol's own timer rather than discov-

ered through search (Section 7.3). More cycles burned does not mean more ICG issued; additional burning only raises competition for the same reward.

6.3 Supply accounting

For reporting, ICG supply should be broken down into categories that can each be observed:

$$\text{Supply}_{\text{ledger}} = \text{Liquid} + \text{Staked}; \quad \text{Minted} = \text{Supply}_{\text{ledger}} + \text{Burned}; \quad \text{Won} = \text{Minted} + \text{Owed} \quad (2)$$

Staked ICG, whether locked, counting down or waiting to be withdrawn, sits in the custody subaccount; totals are reported by `get_staking_state` and each position by `get_stake_position`. *Owed* ICG has been won but not yet minted: pool rewards credited to members but not yet claimed (`get_pool_reward_balance`) and block payouts still pending (`pending_blocks` in `get_statistics`). The genesis 50 ICG is never part of any of these.

7 Mining: Cycle-Burn Block Selection

7.1 Burn rounds

Each solo miner canister runs a timer with a 240-second round. In each round it first checks that the protocol is running. It then burns up to its configured b_{max} cycles per round (default 10,000,000,001) using `cycles_burn128`, which returns the amount actually destroyed. If that amount is at least $10^{10} + 1$ cycles, the miner reports it to the minter through `submit_burned_cycles`. The minter accepts reports only from miner canisters it created itself. It rejects reports below 10^9 cycles. While the protocol is paused no new burn begins, because each miner checks the protocol state before it burns; a burn already under way when the pause begins is still accepted and held, without producing a block, until the protocol resumes, so no participant loses cycles that were destroyed in good faith. Accepted amounts accumulate per miner until the next block, and are kept in stable memory so that a code upgrade does not reset them.

The miner's owner cannot install different code: the minter is the canister's only controller, and upgrades install only the WASM embedded in the minter. The reported amount is therefore the return value of the system burn call in the reference miner code, not a claim made by the user.

7.2 Block selection

Let c_j be the cycles burned and accepted for participant j since the last block. The mining pool counts as one participant (Section 7.7). Let $C = \sum c_j$. When the block timer fires, the minter:

1. burns the pool's accrued fuel (Section 7.7) and adds it to the pool's c_j ;
2. if $C = 0$, produces no block and checks again 60 seconds later;
3. obtains 32 bytes of randomness ρ from the management canister's `raw_rand`;
4. seeds a ChaCha20 generator with ρ and shuffles the participants with it;

5. sums the burns in 128-bit arithmetic, draws u uniformly from $[0, C)$ with the same generator, and selects the first participant whose running cumulative burn exceeds u ;
6. records the block with reward $R(h)$, resets every c_j to zero, and schedules the payout and the next block.

$$Pr[j \text{ wins block } h] = c_j / C \quad (3)$$

The draw uses rejection sampling, so it has no modulo bias, and the 128-bit sum cannot overflow however much is burned: every participant's chance is exactly its share of the cycles burned. If the pool is drawn at the moment its last members have expired, the pool's burn is discarded and the draw is repeated among the remaining participants.

7.3 Block interval

After each block, the delay until the next selection is drawn from the same randomness:

$$\Delta = \text{clamp}(600 + 10 \cdot z, 570, 630) \text{ seconds}, \quad z \sim \mathcal{N}(0, 1) \quad (4)$$

The mean interval is therefore 600 seconds, with a standard deviation of 10 seconds truncated at $\pm 3\sigma$. Because the interval is set by the timer, block production is regular and independent of burn volume. Issuance stops only when the protocol is paused or when no participant burns.

7.4 Randomness

`raw_rand` returns bytes derived from the subnet's random tape, which is produced by threshold BLS signatures of the subnet's nodes [8]. The bytes cannot be predicted by any canister or user before the call completes. A participant therefore cannot tell in advance whether a given burn will win. Manipulating the draw requires corrupting a threshold of the nodes in the minter's subnet.

7.5 Expected reward and cost

From (3), a participant's expected ICG per block is $R(h) \cdot c_j / C$, and their expected ICG per cycle burned is

$$E[\text{ICG per cycle}] = R(h) / C \quad (5)$$

This is the same for every participant, regardless of size. Selection is linear in burn, so there are no economies of scale comparable to specialised proof-of-work hardware. The cycles C burned per block are set by the market, not by the protocol. In a competitive equilibrium, participants burn until the expected value of $R(h)$ roughly equals the cost of C cycles plus the ICP they route to stakers and to the protocol fee.

Cycles are bought with ICP. A deposit of D ICP yields a mining share of about $0.4(D - 2\phi)$ ICP, where ϕ is the ICP ledger fee. At an ICP/XDR rate of X , that share converts to

$$\text{cycles} \approx 0.4 (D - 2\phi) \cdot X \cdot 10^{12} \quad (6)$$

The mining share does not affect who wins: every participant converts the same share of their deposit, and selection depends only on each participant's share of all cycles burned.

7.6 Solo miners

- **Creation.** A miner is created from a regular deposit of at least the current miner price. The price follows the ICP/XDR rate: it is the smallest deposit, in hundredths of an ICP, whose mining share mints 2.5×10^{12} cycles plus a 2% margin, and never less than 1 ICP. With $\phi = 0.0001$ ICP the floor of 1 ICP applies from $X \approx 6.4$ XDR per ICP upward; at $X = 2.55$ the price is 2.51 ICP. In USD terms the price stays near the cost of about 6.4 XDR whenever the floor does not apply. A larger deposit is allowed. The minter transfers the entire cycles output of the deposit's mining share into the new canister (the IC deducts its standard canister-creation fee). Creation checks the cycles actually minted: a deposit that falls short after a rate change is not lost and remains available for the pool or a top-up. The 1 ICP floor limits how many miner canisters can be created cheaply.
- **Fuel.** A miner can be topped up with the cycles of any completed, unassigned deposit made by its owner (minimum deposit 0.1 ICP).
- **Burn rate.** At the default rate, a miner burns about 3.6×10^{12} cycles per day (360 rounds of 10^{10}). The owner may raise or lower $b_{\max}$ to trade burn speed against duration. The chance of winning per cycle does not change.
- **Upgrade and ownership.** The owner may ask the minter to upgrade the miner to the embedded WASM. Ownership, and therefore the reward address, cannot be transferred.

7.7 Mining pool

The pool is a virtual participant whose identity is the minter itself. This is enforced at installation, and the pool cannot submit burns through the public endpoint. Users join the pool with a deposit:

- **Entry and duration.** The minimum is 0.1 ICP. Membership time is proportional to the deposit: 1 ICP buys 24 hours, so 0.1 ICP buys 2 hours 24 minutes. A new deposit extends the current membership from its existing expiry.
- **Fuel.** The cycles actually minted from each pool deposit are recorded as that member's fuel. At every block evaluation, each member's fuel is burned in proportion to the membership time that has elapsed, so the pool can never burn more than its members' deposits produced. Cycles from completed deposits not yet assigned to any use are held in a separate reserve, and the pool cannot consume them. A floor of 5×10^{12} cycles is always kept for the minter's own operation. Fuel that expires before it could be burned is recorded and reported.
- **Rewards.** When the pool wins, the set of active members is fixed at that moment. The reward is split equally among them, the indivisible remainder is distributed one e8 at a time, and each share is credited to the member's pool reward balance at once. No ledger call is made per member, so the cost of a pool block does not grow with the size of the pool. Members collect their accumulated ICG at any time with `claim_pool_rewards`, which mints the whole balance in one transfer with no fee.

Why an equal split is proportional. A member's burn rate equals their fuel divided by their duration. Duration is proportional to deposit size, so every member burns at about the same rate, $0.4 \cdot X \cdot 10^{12}$ cycles per day. This rate differs only through the ICP/XDR rate in effect at their deposit. An equal split per active member is therefore approximately proportional to each member's contribution. Splitting a deposit across several identities brings no advantage, because each identity buys proportionally less time.

7.8 Reward payout

Block rewards and pool claims are minted by transfers from the minting account, with fee zero. Every transfer is given a fixed deduplication identity when it is first sent, and that identity is stored before the call. A failed or ambiguous transfer is retried with the same identity and can never mint twice. Pending payouts and pool balances are stored in stable memory and survive upgrades (Section 13).

8 ICP Deposits and Revenue Allocation

8.1 Deposit procedure

1. The user transfers $D \geq 0.1$ ICP from their principal's default ICP account to the minter's deposit account with an ICRC-1 `icrc1_transfer` carrying the 8-byte memo "ICGOLD01". The equivalent numeric memo `0x4943474F4C443031` of the ICP ledger's legacy `transfer` is accepted as well. The user pays the inbound ledger fee separately. Deposits must be made through the ICGold application: transfers from exchanges, or wallet transfers without this memo, cannot be credited.
2. The user calls `complete_deposit(block_index, use)`, where *use* is the pool, a new miner (at least the current miner price), or a top-up of one of the user's own miners. The minter fetches that block from the ICP index, or from the ICP ledger itself while the index has not yet synced it, and checks the sender, recipient, memo, and amount. Each ICP block index can be consumed exactly once.
3. The minter splits the deposit (Section 8.2), moves the buyback share and the protocol fee to the treasury, sends the mining share to the CMC, and notifies the CMC to mint cycles. It then credits the staker share to the reward index (Section 9).
4. The minter applies the deposit to the named use in the same call. The call is idempotent: repeating it with the same block and use returns the same receipt and never counts the deposit twice. The two-step form, `process_deposit` followed by `join_pool`, `spawn_miner` or `top_up_miner`, remains available and leads to the same state.

Wallet approvals. With a signer wallet such as OISY or NFID Wallet, a deposit takes two approvals: the ICP transfer, explained by the ICP ledger, and `complete_deposit`, explained by the minter. The minter implements the ICRC-21 consent-message standard for every user method, so the wallet shows in plain words what each call does, including amounts, lock lengths and destinations, and lists its standards through ICRC-10. The application records each deposit before sending it; an interrupted deposit is found on the ledger after a reload and completed without sending ICP again.

The deposit record moves through the states `Accepted` → `DevelopmentTransferred` → `MiningTransferred` → `Completed` and is saved after every step, so an interrupted call resumes where it stopped. The state `DevelopmentTransferred` marks the completed treasury transfer.

8.2 The split

Let ϕ be the current ICP ledger fee, read from the ledger when the deposit is accepted. Two outbound fees are reserved first: one for the CMC transfer and one for the treasury transfer. With $d = D - 2\phi$ (in e8s):

$$m = \lfloor 0.40 \cdot d \rfloor, \quad b = \lfloor 0.15 \cdot d \rfloor, \quad v = \lfloor 0.03 \cdot d \rfloor, \quad s = d - m - b - v \quad (7)$$

Here m is the mining share, b the buyback share, v the protocol fee and s the staker share. Rounding remainders always go to the staker share. A deposit too small to give every share a positive amount is rejected. The buyback share and the protocol fee are sent to the treasury subaccount in one transfer; b is booked as owed to the buyback reserve and moved there in a daily batch, and v is treasury.

1.00000 ICP deposit ($\phi = 0.0001$ ICP)

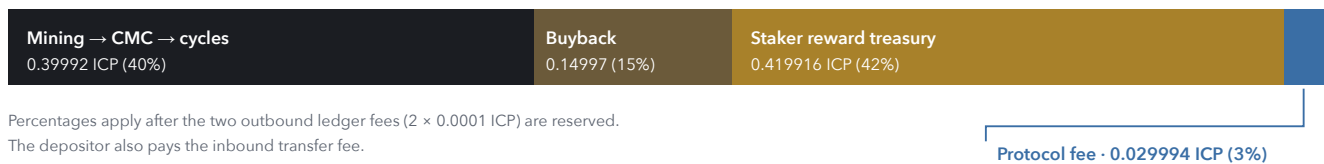


Figure 3. Allocation of a 1 ICP deposit, drawn to scale.

DESTINATION	FORMULA	1 ICP DEPOSIT	10 ICP DEPOSIT
Mining (CMC → cycles)	m	0.39992000	3.99992000
Staker reward treasury	s	0.41991600	4.19991600
ICG buyback reserve	b	0.14997000	1.49997000
Protocol fee (treasury)	v	0.02999400	0.29999400
Outbound ICP ledger fees	2ϕ	0.00020000	0.00020000
Total	D	1.00000000	10.00000000

8.3 Fee changes during processing

The ledger fee is queried, never hard-coded. If the fee rises after the treasury transfer but before the CMC transfer, the difference is taken first from that deposit's staker share and then from its mining share. If it falls, the difference is added to the staker share. The buyback share and the protocol fee never change after acceptance. An unresolved deposit's split is frozen and is never recomputed (Section 13).

8.4 Refunds

ICP that reaches the deposit account with the wrong memo, from a non-default subaccount, or below the 0.1 ICP minimum can be returned with `refund_rejected_deposit`. The refund is the gross amount minus one ledger fee and goes back to the exact account it came from, including its subaccount. The caller must prove that source account. Refunds are replay-protected, and a block can be either refunded or processed, never both.

9 ICG Staking and ICP Rewards

Staking lets ICG holders receive the 42% ICP share of every deposit, weighted by how long they commit their ICG. It does not create ICG and does not change the monetary schedule.

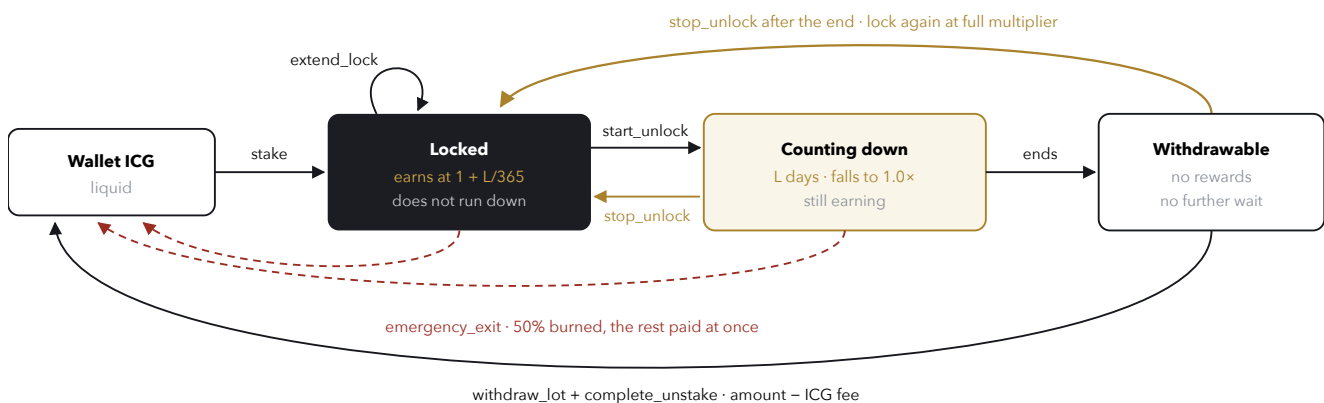


Figure 4. Staking lifecycle. Every stake is a separate position (*lot*) with its own lock length L of 7–365 days. `claim_staking_rewards` pays the ICP earned while Locked and Counting down at any time, without touching the stake.

9.1 Positions and multiplier

Every stake is a separate position (*lot*) with a lock of L days, $7 \leq L \leq 365$, chosen when it is created. A locked lot earns with the multiplier

$$M = 1 + r / 365 \quad (r = \text{remaining days of lock or countdown}), \quad 1 < M \leq 2 \quad (8)$$

for as long as it stays locked: the lock does not run down by itself. A 7-day lot earns at 1.019×, a 90-day lot at 1.247×, a 365-day lot at 2×. Up to 100 lots may be open per principal. A stake must exceed twice the ICG ledger fee, so that it can always be withdrawn, including after an emergency exit.

LOCK	MULTIPLIER WHILE LOCKED	WEIGHT OF 1,000 ICG
7 days (minimum)	1.019×	1,019
30 days	1.082×	1,082
90 days	1.247×	1,247
180 days	1.493×	1,493
365 days (maximum)	2.000×	2,000

9.2 Unlocking

Unlocking starts a countdown equal to the lock length. During the countdown the multiplier falls linearly with the remaining time, recomputed every hour, from $1 + L/365$ towards $1.0\times$. When the countdown ends, the lot stops earning and can be withdrawn at once, with no further wait. At any time, including after the countdown has ended, the owner can stop the countdown and lock the lot again at its full multiplier. A locked lot can be extended to a longer lock (up to 365 days); the higher multiplier applies from that moment, and a later unlock counts down the new length.

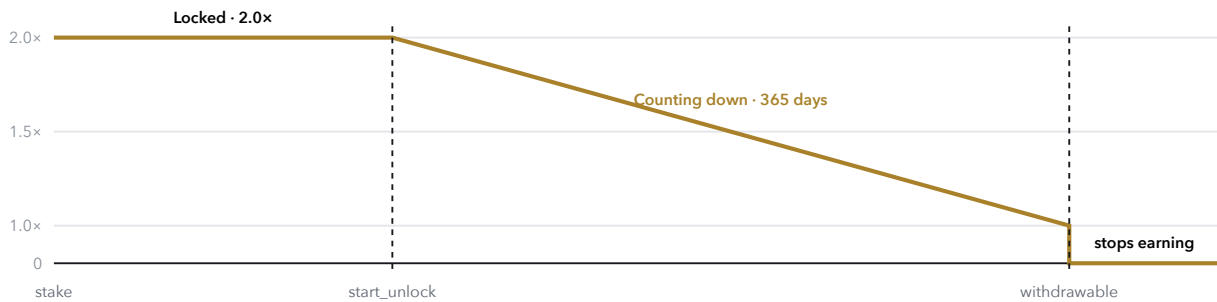


Figure 5. Multiplier of a 365-day lot over time. It stays at $2.0\times$ until the owner starts unlocking, falls linearly to $1.0\times$ over the 365-day countdown, then stops earning. Not to time scale.

9.3 Emergency exit

While a lot is locked or still counting down, it can be exited early: 50% of it (the larger half when the amount is odd) is burned permanently and the rest is withdrawable at once. Once the countdown has ended, emergency exit is refused, because the lot can be withdrawn in full.

9.4 Reward accounting

Weights are measured in e8s-hours: a lot of a e8s with r hours of lock or remaining countdown weighs $a \cdot (8,760 + r)$. The total weight is kept as $W = \text{locked} + \text{bias} - \text{slope} \cdot h$, where h is the current hour, and countdowns that end in the same hour are grouped, so every operation costs the same regardless of the number of stakers. When a staker share s is credited, the reward index grows:

$$I \leftarrow I + \lfloor s \cdot 10^{18} / W \rfloor, \quad J \leftarrow J + h \cdot \Delta I \quad (9)$$

A locked lot earns $a \cdot (8,760 + 24L) \cdot \Delta I$. A counting-down lot that ends at hour e earns $a \cdot ((8,760 + e) \cdot \Delta I - \Delta J)$, which is exactly its hourly weight times the index increments. When a countdown ends, its group records a snapshot of I and J , so the lot earns nothing afterwards. Sub-e8 rounding dust stays in the treasury, so the treasury can never owe more than it holds.

9.5 Operations

OPERATION	BEHAVIOUR
Stake	The wallet approves the minter (ICRC-2), then calls <code>stake_icg(amount, lock_days)</code> . The minter pulls the ICG into the custody subaccount. The wallet needs <code>amount + 2 × ICG fee</code> .

OPERATION	BEHAVIOUR
Unlock	<code>start_unlock(lot)</code> starts the countdown; <code>stop_unlock(lot)</code> locks the lot again; <code>extend_lock(lot, days)</code> lengthens a locked lot.
Withdraw	<code>withdraw_lot(lot)</code> after the countdown, or <code>emergency_exit(lot)</code> before it; then <code>complete_unstake(id)</code> pays the amount minus the ICG ledger fee. Up to 100 withdrawals may be open per principal.
Claim	<code>claim_staking_rewards</code> pays all accrued ICP minus the ICP ledger fee. The stake is not affected.

9.6 Rewards received while nobody is staking

A staker share that arrives while nothing earns is not lost and is not given to the first staker. It is added to an undistributed reserve U . Once stake exists, the reserve at that moment (U_0) is released linearly over 90 days of active protocol time and shared by weight among whoever is staked as the schedule advances:

$$released(\tau) = U_0 \cdot \min(\tau, 90 \text{ days}) / 90 \text{ days} \quad (10)$$

Here τ counts only time during which the protocol is running and some weight exists. Depositing ICP does not speed up the release. The reserve is never moved to any team or treasury account.

9.7 Reward arithmetic (illustrative, not a forecast)

Over a period in which D ICP is deposited in n deposits and the average total weight is $\bar{W}$ (in ICG-equivalents), each unit of weight accrues about

$$r \approx 0.42 \cdot (D - 2\phi \cdot n) / \bar{W} \quad (11)$$

For example, if 1,000 ICP is deposited in a month and the average weight is 1,500,000, each unit of weight accrues about 0.00028 ICP that month: a 1,000 ICG lot locked for 365 days (weight 2,000) accrues about 0.56 ICP, a 7-day lot (weight 1,019) about 0.29 ICP. If deposits stop, rewards stop.

10 Protocol Treasury and Canister Maintenance

A protocol that holds user funds must not stop because a canister runs out of cycles. ICGold pays for its own existence: no operator has to top up its canisters. Every deposit sends its 15% buyback share and its 3% protocol fee to the treasury subaccount. Once a day, when at least ten ICP ledger fees' worth is owed, the buyback share is moved to the buyback reserve in one transfer; its fee is paid by the treasury. The protocol fee is treasury, and it keeps the protocol's own canisters funded first:

PARAMETER	DEFAULT
Check interval	every 6 hours (configurable, minimum 60 seconds)
Top-up trigger	the target's own cycles below 5×10^{12}

PARAMETER	DEFAULT
Top-up target	10×10^{12} cycles
Rolling daily cap	20×10^{12} cycles
Minimum partial top-up	0.5×10^{12} cycles
Eligible targets	the minter (always), the ICG ledger and the ICG buyback canister, and nothing else

For the minter, only its own cycles are measured: cycles that belong to unassigned deposits and unburned pool fuel are excluded. When a target falls below the trigger, the minter reads the live ICP/XDR rate and pays for the shortfall, limited by the daily cap, from the treasury share only, never from the buyback share or from funds reserved for an unconfirmed transfer. If the treasury cannot pay all of it, it pays what it can, provided that buys at least 0.5×10^{12} cycles. The top-up is recorded before any external call, and an interruption resumes the same transfer. Measured use is about $0.005\text{--}0.1 \times 10^{12}$ cycles per day, so a top-up lasts weeks.

Treasury ICP above a floor (default 1 ICP) can be sent to the published development account only: never elsewhere, never from the buyback share, and never while another treasury transfer is unconfirmed. All treasury transfers are serialized. Maintenance and the daily buyback move keep running during a global pause, so a long emergency cannot starve the protocol itself.

Reading a canister's cycle balance requires controller access. The minter is therefore installed as an additional controller of the ICG ledger and of the buyback canister (Section 14).

11 ICG Buyback: Protocol-Owned Liquidity and Burn

15% of every deposit funds an ICG buyback. The buyback runs in a separate canister and spends the reserve on the ICG/ICP pool of ICPSwap, a decentralized exchange on the Internet Computer whose pools are governed by its SNS [12]. Its purpose is twofold: to give ICG deep, permanent liquidity, and then to turn a steady share of mining demand into purchases and burns of ICG.

11.1 Funding

The buyback share reaches the buyback reserve through the treasury, as described in Section 10. The minter grants the buyback canister an ICRC-2 allowance over the reserve, renewed every twelve hours to the reserve's balance and valid for two days. Nothing moves until the buyback canister draws what its next step needs; no other principal, the administrator included, can draw from the reserve.

11.2 Three phases

ICP IN THE POOL	EACH STEP
below 100	Nothing runs. A pool this shallow is too easy to move.

ICP IN THE POOL	EACH STEP
Phase 1 · 100 to 1,000	Builds liquidity: half of the step buys ICG, and both halves are added to the protocol's full-range position.
Phase 2 · 1,000 to 10,000	Builds liquidity or buys and burns ICG, with even odds decided by the subnet's randomness.
Phase 3 · from 10,000	Buys ICG and burns it.

The liquidity position belongs to the buyback canister for good: no function can withdraw it, and the trading fees it earns stay in the pool. Burned ICG is sent to the ICG ledger's minting account, which destroys it.

11.3 A step

1. Read the pool's price and the ICP it holds; record a price sample at most every ten minutes.
2. Burn any ICG the canister holds from an earlier step.
3. Decide the phase and check the price against its average (Section 11.4).
4. Size the swap: at most 0.5% of the ICP in the pool, at least 0.1 ICP, and no more than the reserve allows.
5. Draw the ICP from the reserve, deposit it into the pool and swap it for ICG, refusing any fill more than 1% below the pool's quote, and skipping the swap if that quote buys less than 94% of what the same ICP buys at the six-hour average price.
6. Add both halves to the liquidity position, or withdraw the ICG to be burned.

Each step decides from balances it reads first: the canister's wallet, its unused balance inside the pool and the reserve allowance. ICPSwap settles some transfers in the background; a step that stops halfway is finished by the next, and ICP never waits outside the pool or the reserve.

11.4 Protection against manipulation

- **Small steps.** A swap of 0.5% of the pool's ICP moves the price by about half a percent. A shallow pool slows the buyback instead of paying more.
- **Slippage guard.** Every swap sets a minimum output 1% below the quote; a worse fill is refused and the ICP stays in the pool for the next step.
- **Average-price floor.** The quote is read a few seconds after the price check, and the reserve draw before it is public. So that a price pushed up in between cannot make the buyback overpay, the quote must also buy at least 94% of what the same ICP buys at the six-hour average price; otherwise the step is skipped and the ICP waits in the pool. A liquidity step whose tokens would be added at a price pushed more than 3% below the average keeps them in the pool for the next step.
- **Price band.** Steps wait for two hours of ten-minute price samples and are skipped while the price is more than 3% above its six-hour average, or, for liquidity steps, more than 3% below it. The

buyback therefore never buys into a spike and never adds liquidity at a pushed-down price. Its own buying pauses itself before it can push the price far.

- **Unpredictable timing.** Steps run at random ten-minute ticks, about once an hour, so their timing cannot be traded against. The Internet Computer has no public mempool from which pending swaps could be front-run.
- **Verified pool.** The pool is set once and must be the one ICPSwap's factory registers for ICG/ICP at the 0.3% fee tier; its tokens and fee are checked as well.

11.5 Activation and capacity

The buyback starts once participants have seeded an ICG/ICP pool with at least 100 ICP of liquidity, since there is no premine to seed it. The administrator connects the pool, runs the first steps by hand and then switches on the hourly steps; the limits above can change only by a code upgrade. At one step an hour, the buyback can spend about 12% of the pool's ICP per day in the burn phase, and about twice that while building liquidity, since only half of a liquidity step is swapped. More than the reserve holds is never spent; what the limits leave waits in the reserve.

DAY	ICP IN THE POOL	OWNED BY THE PROTOCOL	WHAT HAPPENS
0	250	0%	Participants seed the pool; 30 ICP a day reaches the reserve.
10	≈ 550	≈ 55%	Every step deepens the pool.
25	≈ 1,000	≈ 75%	Phase 2 begins.
later	≥ 10,000	—	Phase 3: every step buys and burns.

Illustration, not a forecast: 200 ICP of deposits a day and a constant ICG price. Rising prices reach each threshold sooner.

12 Protocol Time and Emergency Pause

The minter is always installed in the Paused state and only runs after the administrator calls `start_protocol`. It keeps a protocol clock:

$$t_{\text{protocol}} = t_{\text{wall}} - (\text{total time spent paused}) \quad (12)$$

Every time-based entitlement uses this clock: pool membership expiry, pool fuel burning, lock countdowns, the 90-day reserve release, and block timing. A pause therefore freezes these clocks, and each entitlement resumes with the same remaining duration.

While the protocol is paused, the following are blocked: new cycle burns, block production, deposits and their assignment, miner creation and upgrade, pool joins and pool claims, staking, all lot actions, reward claims, and withdrawals. Automatic maintenance and the daily buyback move keep

running. Ordinary ICG and ICP wallet transfers keep working, because they are ledger operations the minter does not control. Read-only queries remain available. The one exception concerns timing: a miner's report for a burn that was already in flight when the pause began is booked and waits for the resume; no block is selected from it while the protocol is paused.

DELIBERATE TRADE-OFF

The pause is total by design: it also stops claims and withdrawals. This allows a complete stop in an emergency, such as a suspected accounting fault. The cost is that users cannot exit while the protocol is paused. This is an administrative risk and is listed in Section 14.

13 Transfer Safety and Reconciliation

Inter-canister calls on the IC are asynchronous, and a reply can be lost. A naive retry of a lost transfer can pay twice. ICGold therefore gives every outbound transfer a durable identity, stored in stable memory before the call is made:

identity = (from, to, amount, fee, memo, created_at_time) (13)

Both the ICP ledger and ICRC ledgers reject an exact duplicate within their 24-hour deduplication window and return the original block index instead. A retry with the same identity is therefore safe: it either executes once or reports the earlier execution. Responses are classified as follows:

CLASS	EXAMPLES	ACTION
Success / duplicate	block index; Duplicate	Mark the step complete.
Proven not executed	ICRC <code>InsufficientFunds</code> , <code>InsufficientAllowance</code>	Clear the attempt. The ledger checks deduplication before these errors, so the transfer cannot have run.
Definitive rejection	<code>BadFee</code> , <code>TooOld</code> , <code>CreatedInFuture</code> , ICP errors	Keep the exact identity and mark the transfer uncertain. The response alone is not accepted as proof that an earlier attempt failed.
Unknown / transient	transport or reject errors	Retry later with the identical identity.

13.1 Reconciliation

An uncertain transfer is settled with `resolve_uncertain_transfer` , which first takes the same lock as the normal retry of that operation, so a resolution and a retry can never both apply the result.

- **Executed(block).** The minter reads that block from the ICP index or the ICG ledger, including archives, and finalizes only if every identity field matches.
- **NotExecuted.** Accepted only after the original timestamp is older than the 24-hour window plus 60 seconds, when the transfer can no longer execute. The minter then proves the absence itself. For ICP it confirms that the ICP index has caught up with the ledger and scans the sending account's history back to the transfer's time. For ICG it scans the ledger's blocks from that time win-

dow. It refuses if a matching transfer exists. Only then does the next attempt use a fresh time-stamp and the current fee.

Reconciliation covers stakes, reward and pool claims, unstakes and emergency burns, both deposit transfers, refunds, maintenance, buyback moves, treasury transfers and block payouts. It rejects operations that are not marked uncertain, so settled state cannot be rewritten.

13.2 Upgrade procedure

Every minter upgrade follows this sequence, automated by the upgrade script: wait for a round with no burn; pause; wait until `get_upgrade_status` shows no user call, timer task, stake transfer or block payout in flight, and otherwise resume on the old code and stop; stop the canister, so that no callback is lost during code replacement; take a snapshot on mainnet; install; start; resume. All accounting state, including the current round's burned cycles, lives in stable memory and survives upgrades.

14 Governance and Controllers

Governance is a security boundary: a correctly specified rule means little if the code that enforces it can be replaced. ICGold's governance is planned to move in stages to a DAO in which long-term ICG stakers decide on protocol changes. Control will be handed over step by step, starting with the least critical powers, as each governance component is implemented, audited and published. The DAO is planned to be formed about twelve months after launch, after the source code has been published as open source at about six months (Section 17). The form and voting rules of each stage have not yet been fixed; each will be published and reviewed before adoption.

Until the corresponding stage is reached, the following powers are held by the protocol administrator (the deploying principal):

ROLE	CAPABILITIES	LIMITS IN CURRENT CODE
Protocol administrator	Start, pause and resume the protocol. Tune or disable maintenance within validated bounds, and trigger it. Set the treasury floor and send treasury ICP above it to the published development account. Resolve uncertain transfers. Name the buyback canister and its ICPSwap pool, once each; run buyback steps by hand and switch the hourly steps on or off. As a canister controller, upgrade the minter, ICG ledger and buyback code.	No endpoint can mint ICG outside block and pool rewards, change the reward schedule or the split, move staker funds, spend the buyback reserve other than through the buyback's rules, withdraw the protocol-owned liquidity, send treasury funds anywhere but the development account, or rewrite settled transfers. The buyback pool must be the one ICPSwap's factory registers. Both reconciliation outcomes are verified on-chain. A code upgrade could change any of these.
Minter canister	Minting account of ICG. Additional controller of the ICG ledger, used for cycle inspection and top-up. Sole controller of all miner canisters.	Acts only as its code prescribes.

ROLE	CAPABILITIES	LIMITS IN CURRENT CODE
Miner owner	Set the burn per round; top up; request an upgrade to the embedded miner WASM.	Cannot change the miner's code or transfer ownership.
Development principal	None.	Receives only donations and treasury transfers.

15 Economic Analysis and Risks

15.1 The mining-staking loop

Mining demand creates staker revenue: every ICP deposited sends 42% to stakers and 15% to the ICG buyback. Demand for ICG as a staking asset, in turn, supports the value of the block reward that motivates mining. This loop is reflexive. It reinforces itself while mining is profitable and weakens symmetrically when it is not. Nothing in the protocol guarantees that the loop runs in either direction. Low activity means both low mining competition and low staker rewards.

15.2 Staker-miner rebate

A participant who holds a fraction σ of the total staking weight and also mines receives about $0.42 \cdot \sigma$ of their own deposit back as staking rewards. Their effective cost per unit of deposit is

$$cost_{eff} \approx 1 - 0.42 \cdot \sigma \quad (14)$$

A participant with 20% of the weight therefore mines about 8% more cheaply than one with none. This advantage is linear, bounded, and fully visible on-chain. Still, it favours large, long-locked holders who also mine, and it is a structural pressure toward concentration.

15.3 Just-in-time staking

Staker shares are credited the moment a deposit is processed. A holder who stakes just before a large deposit is processed must still lock for at least 7 days and then count down at least 7 days, or give up 50% through an emergency exit. Longer commitments receive a higher weight, so short-term stake receives the smallest share.

15.4 Locks and liquidity

A lot cannot be withdrawn without a countdown as long as its lock, except by burning half. Locked ICG is therefore illiquid for up to 365 days after an owner decides to leave. If many large positions finish their countdowns together, supply can become liquid at once; no claim is made about price effects.

15.5 ICP/XDR dependence

The cycles obtained per ICP vary with the ICP/XDR rate maintained by the NNS. When ICP falls in XDR terms, the same deposit buys fewer cycles. The pool accounts for this exactly, because it burns

only the cycles each deposit actually minted. Below about 6.4 XDR per ICP the solo miner price rises above 1 ICP so that a new miner still starts with 2.5×10^{12} cycles (Section 7.6).

15.6 Market and liquidity

The protocol defends no price. Its buyback builds liquidity that cannot be withdrawn and then buys ICG in small steps (Section 11); this supports market depth but does not set a price. Early issuance is highest (7,200 ICG per day in epoch 0). Markets may be thin, especially at launch, and the first pool must be seeded by participants, since there is no premine. The protocol never converts ICG to ICP, so staker rewards cannot be paid out of ICG sales.

15.7 Where yield comes from

A simple equilibrium model makes the flows explicit. Suppose miners deposit until the ICG they win is worth what they pay, and sell it; let X_{in} be the ICP that new ICG buyers bring per day, E the ICG issued per day, S the ICG staked and $\beta = 0.15$ the buyback share. Then deposits, price and the stakers' annual yield are about

$$D \approx X_{in} / (1 - \beta), \quad P \approx D / E, \quad APR \approx 0.42 \cdot 365 \cdot E / S \quad (15)$$

Three conclusions follow. First, the stakers' yield in percent does not depend on price: it falls as the staked supply grows and as issuance halves, from very high in the first months to single digits after the early epochs. Second, price follows the inflow of new buyers; a short burst of demand followed by a fall leaves early buyers with losses even though stakers keep earning. Third, for the same inflow the buyback raises deposits and price by a factor $1/(1 - \beta)$, about 18%, and removes a share β of new issuance from circulation. The model is a simplification; it ignores speculation and changes in the ICP price.

16 Security Considerations

THREAT	MITIGATION	RESIDUAL RISK
Deposit replay or double use	Each ICP block index is consumed once. Refund and processing are mutually exclusive and rechecked after every await. Assignment is recorded before any external call.	None known.
Forged burn reports	Only minter-created miners may report. Miner code is controlled by the minter. Minimum report sizes apply. The pool cannot report externally.	Relies on the correctness of the reference miner code.
Draw manipulation	Threshold random beacon (<code>raw_rand</code>), unknown until the call returns.	IC subnet threshold trust.
Double payment on retry	Durable transfer identities set when a transfer is first sent; uncertain states preserved; reconciliation locked against retries; NotExecuted proven on-chain.	None known.

THREAT	MITIGATION	RESIDUAL RISK
Pool payout scale	Pool wins credit balances; members claim in one mint.	None known.
Lost callback on upgrade	Mandatory pause → drain check → stop → upgrade → start → resume; the upgrade aborts if work stays in flight.	Operator error.
Cycle exhaustion	Automatic maintenance measured on the minter's own cycles; partial top-ups; 5×10^{12} cycle operating floor; unassigned deposit cycles reserved.	Treasury exhausted during a long period without deposits.
Ingress and denial-of-service abuse	An ingress filter rejects anonymous callers (except for a price read), unknown methods and malformed arguments. Untrusted input is validated before any lock is taken; serialization is per principal.	Ordinary IC rate limits.
Pool identity spoofing	At installation the pool identity must equal the minter.	None known.
Frontend compromise	Content Security Policy; pinned dependencies; all state verifiable from the canisters.	Users who do not verify what they sign.
Buyback pool substitution	The pool is set once and must be the one ICPSwap's SwapFactory registers for ICG/ICP at 0.3%; its tokens and fee are checked too.	ICPSwap's own governance (SNS).
Price manipulation around buyback steps	Swaps of at most 0.5% of the pool with a 1% minimum-output guard; random timing; two hours of price history before any step; steps skip beyond 3% from the six-hour average.	Slow manipulation within the 3% band.
Exchange failure	Only small chunks leave the reserve per step; each step decides from balances read first, so interrupted steps are finished by the next.	Loss of the protocol-owned liquidity if ICPSwap itself fails.
Administrator key compromise	Operational: hardware-backed key custody.	Decreases as control moves to the DAO.

17 Future Directions

STATUS · PLANNED, NOT IMPLEMENTED

This section describes intentions. None of it is part of the current protocol. Each part will be specified, implemented, audited and announced before it takes effect, and plans may change.

17.1 A DAO, in stages

ICG is intended to gain governance properties step by step. Control of the protocol is planned to pass gradually to a DAO in which long-term stakers decide on protocol changes, starting with the

least critical powers, as each governance component is implemented, audited and published (Section 14).

Planned milestones, counted from launch:

- **About 6 months: open source.** The source code of the minter, miner and buyback canisters is planned to be published under an open-source licence, so that anyone can review it and check that the deployed WASM matches it.
- **About 12 months: DAO.** A DAO is planned to be formed that governs the protocol, with long-term stakers voting on protocol changes. Its first stage takes over the least critical powers; the rest follow as described above.

These dates are targets, not commitments, and may move if an audit or a security concern requires it.

17.2 ICG as the centre of a DeFi protocol

The longer-term plan is for ICG to be the centre of a DeFi protocol on the Internet Computer, including the launch of a stablecoin.

17.3 An advanced CDP lending platform

A later phase plans a DeFi platform that lets users unlock liquidity from their assets without giving up passive yield:

- **Yield-bearing collateral.** Users deposit assets that already earn a yield, such as staked or liquidity tokens, and borrow (mint) the protocol's stablecoin against them.
- **Self-repaying loans.** The collateral's yield keeps increasing the value of the deposit, which reduces the risk of liquidation over time.
- **Isolated risk.** Each collateral type has its own independent market, so a fall in one asset never endangers the rest of the system.
- **One-click leverage.** An integrated tool opens a leveraged yield position in a single, simple transaction, within each market's risk limits.

This upgrade is intended to bring deep liquidity to the ecosystem, steady demand for the stablecoin and new revenue for the protocol. Borrowing, stablecoins and leverage carry their own risks, including liquidation and loss of the peg; these will be analysed in the specification of each component.

18 Dependencies

- **Administrative trust.** Until the DAO transition is complete, upgrade and pause authority are held by the protocol administrator (Section 14).
- **Total pause.** Withdrawals stop during a pause. A future governance stage could allow exits during extended pauses.

- **Buyback dependence.** The buyback needs an ICG/ICP pool on ICPSwap with at least 100 ICP of liquidity, seeded by participants; its protocol-owned liquidity carries ICPSwap's risk (Section 11).
- **Reflexive economics.** Staker revenue depends on continuing mining demand, and mining demand depends on the value of ICG (Section 15.1).
- **Concentration.** Linear selection has no economies of scale, but the staker-miner rebate (Section 15.2) and the lock multiplier favour large, committed holders who also mine.
- **Dependence on IC infrastructure.** The ICP ledger, index, and CMC rate, and subnet randomness, are external dependencies. Proving that an uncertain transfer did not execute requires the ICP index to be current.

19 Conclusion

ICGold brings Bitcoin's monetary arithmetic to the Internet Computer and replaces hashing with the IC's native, irreversible cost: burned cycles. Its issuance is fixed in code and fair from genesis. Its block selection is linear and uses verifiable randomness. Unlike a pure burn design, most of the resources committed to mining become ICP revenue for ICG stakers, weighted towards those who commit for longer, and a rule-based buyback turns part of every deposit into permanent liquidity and burned ICG. The protocol pays for its own infrastructure. The result is a real flow of value in a different asset rather than a promise of yield.

What this paper claims is operational and can be checked: the rules are encoded, the flows are reconcilable against the ledgers, and every administrative capability is disclosed, together with the plan to hand it to a DAO.

References

1. S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. bitcoin.org/bitcoin.pdf
2. DFINITY Foundation and the identity & wallet standards working group, ICRC-21 canister call consent messages and the ICRC-25/27/29/49/94 signer standards. github.com/dfinity/wg-identity-authentication
3. DFINITY Foundation, Internet Computer developer documentation: cycles, canister resource accounting and cycle costs. internetcomputer.org/docs
4. DFINITY Foundation, Cycles Minting Canister (NNS) interface and ICP/XDR conversion. [github.com/dfinity/ic\(rs/nns/cmc\)](https://github.com/dfinity/ic(rs/nns/cmc))
5. DFINITY Foundation and the ledger & tokenization working group, ICRC-1 and ICRC-2 fungible token standards. github.com/dfinity/ICRC-1
6. DFINITY Foundation, ledger suite: ICRC ledger, ICP ledger and ICP index canisters. [github.com/dfinity/ic\(rs/ledger_suite\)](https://github.com/dfinity/ic(rs/ledger_suite))
7. DFINITY Foundation, The Internet Computer Interface Specification: management canister (raw_rand, canister creation and installation) and system API (cycles_burn128). internetcomputer.org/docs
8. DFINITY Team, "The Internet Computer for Geeks," technical whitepaper, 2022. Random beacon and threshold signatures.

9. M. Egorov, "Curve DAO: vote-escrowed CRV," 2020. The scheduled slope-change technique used for countdown weights.
10. ICGold reference implementation: docs/PROTOCOL.md, docs/TRANSFER-RECONCILIATION.md, docs/AUDIT-2026-10-01.md, canisters/minter, canisters/miner, canisters/buyback.
11. BOB project: minter and miner canisters for proof-of-burn mining on the Internet Computer. github.com/bob-robert-ai/bob
12. ICPSwap, SwapFactory and SwapPool canister interfaces. github.com/ICPSwap-Labs/docs

A Parameter Registry

All values are implemented in the reference code unless marked otherwise. "Constant" means changeable only by a code upgrade.

PARAMETER	VALUE	MUTABILITY	SOURCE
Initial block reward	50 ICG (5×10^9 e8s)	Constant	INITIAL_BLOCK_REWARD_E8S
Halving interval	210,000 blocks	Constant	BLOCK_HALVING
Genesis reward	Recorded, never minted	Constant	mine_block
Mintable supply bound	20,999,949.9769 ICG	Derived	Eq. (1)
Block interval	600 s mean, clamp 570–630 s	Constant	next_block_time
Idle block retry	60 s	Constant	IDLE_RETRY
Miner round	240 s	Constant	ROUND_LENGTH_SECS
Default burn per round	10,000,000,001 cycles	Per miner owner	DEFAULT_BURNED_CYCLES_PER_ROUND
Minimum accepted burn report	10^9 cycles	Constant	submit_burned_cycles
Deposit split (mining / stakers / buyback / protocol fee)	40% / 42% / 15% / 3%	Constant	*_BPS in economics.rs
Outbound fees reserved per deposit	$2 \times$ ICP ledger fee	Constant	DEPOSIT_LEDGER_FEES
Minimum deposit	0.1 ICP	Constant	MIN_DEPOSIT_E8S
Deposit memo	"ICGOLD01" (ICRC-1 bytes) or 0x4943474F4C443031	Constant	ICG_DEPOSIT_MEMO
Solo miner price	max(1 ICP, deposit whose 40% share mints $1.02 \times 2.5 \times 10^{12}$ cycles), in 0.01 ICP	ICP/XDR rate	minimum_solo_deposit
Pool minimum / time	0.1 ICP / 24 h per ICP	Constant	MIN_POOL_DEPOSIT_E8S

PARAMETER	VALUE	MUTABILITY	SOURCE
Minter operating cycle floor	5×10^{12} cycles	Constant	PROTOCOL_CYCLE_SAFETY_RESERVE
Lock length	7–365 days	Constant	MIN_LOCK_DAYS , MAX_LOCK_DAYS
Multiplier	1 + remaining days / 365, hourly, max 2×	Constant	staking.rs
Countdown after unlock	Equal to the lock length	Constant	staking.rs
Emergency exit	50% burned, rest withdrawable at once	Constant	unlock_payout_e8s
Open lots / open withdrawals	100 / 100 per principal	Constant	staking.rs
Reward index scale	10^{18}	Constant	REWARD_INDEX_SCALE
Undistributed reserve release	Linear, 90 active days	Constant	RESERVE_RELEASE_DURATION_NANOS
Buyback move to the reserve	Daily, when ≥ 10 ledger fees are owed	Constant	SWEEP_INTERVAL_NANOS
Treasury floor	1 ICP	Administrator	DEFAULT_TREASURY_FLOOR_E8S
Maintenance trigger / target / cap	5T / 10T / 20T cycles per day	Administrator, validated	maintenance.rs
Minimum partial top-up	0.5T cycles	Constant	MIN_TOP_UP_CYCLES
Maintenance interval	6 h (≥ 60 s)	Administrator, validated	maintenance.rs
Maintenance targets	Minter + ICG ledger + buyback canister only	Constant	configure_maintenance
Buyback phases (ICP in the pool)	< 100 idle; < 1,000 liquidity; < 10,000 mixed; then burn	Constant	buyback/lib.rs
Buyback swap size	$\leq 0.5\%$ of the pool's ICP; ≥ 0.1 ICP	Constant	MAX_SWAP_BPS
Buyback slippage guard	1% below the quote	Constant	SLIPPAGE_BPS
Buyback average-price floor	$\geq 94\%$ of the output at the 6-hour average	Constant	MAX_SHORTFALL_BPS
Buyback price band	$\pm 3\%$ of the 6-hour average; 2 h of samples first	Constant	PRICE_BAND_BPS
Buyback timing	10-minute ticks, a step about hourly at random	Constant	timer_runs_step
Buyback reserve allowance	Renewed every 12 h, valid 2 days	Constant	renew_buyback_allowance

PARAMETER	VALUE	MUTABILITY	SOURCE
Uncertain-transfer negative wait	24 h + 60 s, then on-chain proof	Constant	LEDGER*_NANOS
ICG decimals / symbol	8 / ICG	Ledger install	deployment
ICG transfer fee	Ledger parameter (reference: 0.01 ICG)	Ledger	deployment

B Public Interface Summary

AREA	UPDATE CALLS	QUERIES
Deposits	complete_deposit, process_deposit, refund_rejected_deposit	get_deposit, get_refund, get_deposit_account
Mining	spawn_miner, top_up_miner, upgrade_miner, join_pool, claim_pool_rewards, submit_burned_cycles (miners only), get_solo_mining_minimum	get_statistics, get_blocks, get_latest_blocks, get_block_by_index, get_current_block_status, get_miners, get_leader_board, get_pool_statistic, hours_left_in_pool, get_pool_reward_balance
Staking	stake_icg, start_unlock, stop_unlock, extend_lock, withdraw_lot, emergency_exit, complete_unstake, claim_staking_rewards	get_staking_state, get_stake_position
Treasury	set_treasury_floor, withdraw_treasury, set_buyback_canister (administrator only)	get_protocol_fees
Buyback canister	set_pool, run_step, set_automatic (administrator only)	get_status
Wallets	icrc21_canister_call_consent_message (ICRC-21)	icrc10_supported_standards (ICRC-10)
Protocol	start_protocol, pause_protocol, configure_maintenance, run_maintenance_now, resolve_uncertain_transfer (all administrator only)	get_protocol_config, is_protocol_running, get_maintenance_status, get_pending_stake_count and get_upgrade_status (administrator only)

C Glossary

Buyback reserve	The ICP subaccount that receives the 15% buyback share; drawn only by the buyback canister, to buy ICG.
Countdown	The period after start_unlock, equal to the lock length, during which a lot's multiplier falls to 1.0×.

Cycles	The Internet Computer's unit of computation cost. 10^{12} cycles are minted per XDR of ICP converted through the CMC. Burned cycles are destroyed.
CMC	Cycles Minting Canister, the NNS canister that converts ICP into cycles.
e8s	Base units of ICP or ICG; 1 token = 10^8 e8s.
Epoch	A span of 210,000 blocks with a constant block reward.
Fuel	The cycles a pool member's deposits produced, burned in proportion to their membership time.
ICRC-1 / ICRC-2	Internet Computer fungible-token standards for transfers, and for approvals with delegated transfers.
Lot	One staking position with its own amount, lock length and state.
Multiplier	$1 + \text{remaining days} / 365$; a lot's reward weight per staked ICG.
Protocol fee	The 3% of every deposit that funds the treasury and the protocol's own cycles.
Protocol-owned liquidity	The buyback canister's full-range position in the ICG/ICP pool, which no function can withdraw.
Protocol time	Wall-clock time minus all time spent paused.
Reward index	A cumulative ICP-per-unit-of-weight counter that lets rewards be distributed in constant time.
Uncertain transfer	A ledger call whose execution could not be proven either way. It is kept with its exact identity until resolved.
XDR	The IMF Special Drawing Right, the unit to which the ICP-to-cycles rate is pegged.