A Bitcoin-Charged Crypto Economy with Trustless Vaults

Babylon Team

6th August 2025

Abstract

Bitcoin is by far the largest crypto asset by market capitalization, yet the vast majority of it remains idle. Today, less than 1% of all bitcoins are bridged to smart contract platforms, where most decentralized finance (DeFi) activity takes place. A key barrier to greater participation is that existing Bitcoin bridges are either centralized or rely on significant trust assumptions. In fact, due to the lack of covenants in the Bitcoin scripting language, there are no known ways to build trustless Bitcoin bridges.

We bypass this barrier by introducing a different primitive: **trustless Bitcoin vaults**. Bitcoin holders deposit their BTC into these on-chain vaults in a self-custodial fashion. The bitcoins in each vault are tied to a specific smart contract protocol on an external chain. These vaults are programmable, and withdrawals are permitted only when a zero-knowledge proof of a specific smart contract state is verified on the Bitcoin chain. Together with an appropriate Bitcoin scripting design of the vault, this eliminates the need for mutual trust among parties.

We demonstrate how this construction enables the use of *native* Bitcoin—without wrapping or bridging—as collateral in many DeFi applications, including lending, stablecoin issuance, and perpetual DEXs. Leveraging recent advances in BitVM3, we present experimental results showing that these vaults can be operated with minimal on-chain overhead—an essential requirement for scaling trustless Bitcoin-backed DeFi. Bitcoin vaults built on top of the Babylon Bitcoin staking protocol further improve capital efficiency by allowing staked BTC to act as collateral in DeFi protocols. We conclude by discussing how trustless Bitcoin vaults significantly strengthen the existing Babylon ecosystem and the utility of BABY as the token for its decentralized governance.

Table of Contents

1 Problem Statement

Bitcoin is the largest crypto asset, with a market capitalization exceeding that of all other cryptocurrencies combined. It recently became the fifth-largest asset across all asset classes. Yet more than 99% of this capital remains idle, disconnected from the rapidly growing smart contract and decentralized finance (DeFi) ecosystems.

This underutilization is particularly striking given that most DeFi protocols, such as lending, stablecoin issuance, and perpetual DEXs, require collateral, and BTC is arguably the ideal form of crypto collateral: most BTC holders prefer to retain their coins long term, and using BTC as collateral allows them to leverage their assets without liquidating their position. Moreover, participation of BTC in DeFi, as opposed to centralized finance (CeFi), is strongly aligned with Bitcoin's ethos of decentralization and trust minimization.

There are two key technical barriers to broader BTC participation in DeFi:

1. The Bitcoin chain supports only a primitive scripting language, which severely limits its ability to natively express general-purpose smart contracts.

2. Smart contracts are primarily deployed on other blockchains, but existing bridges that move BTC across chains are either centralized or involve strong trust assumptions, which has hindered adoption. Indeed, the two major bridged BTC assets, WBTC and cbBTC, together account for less than 1% of Bitcoin's total market capitalization today[1].

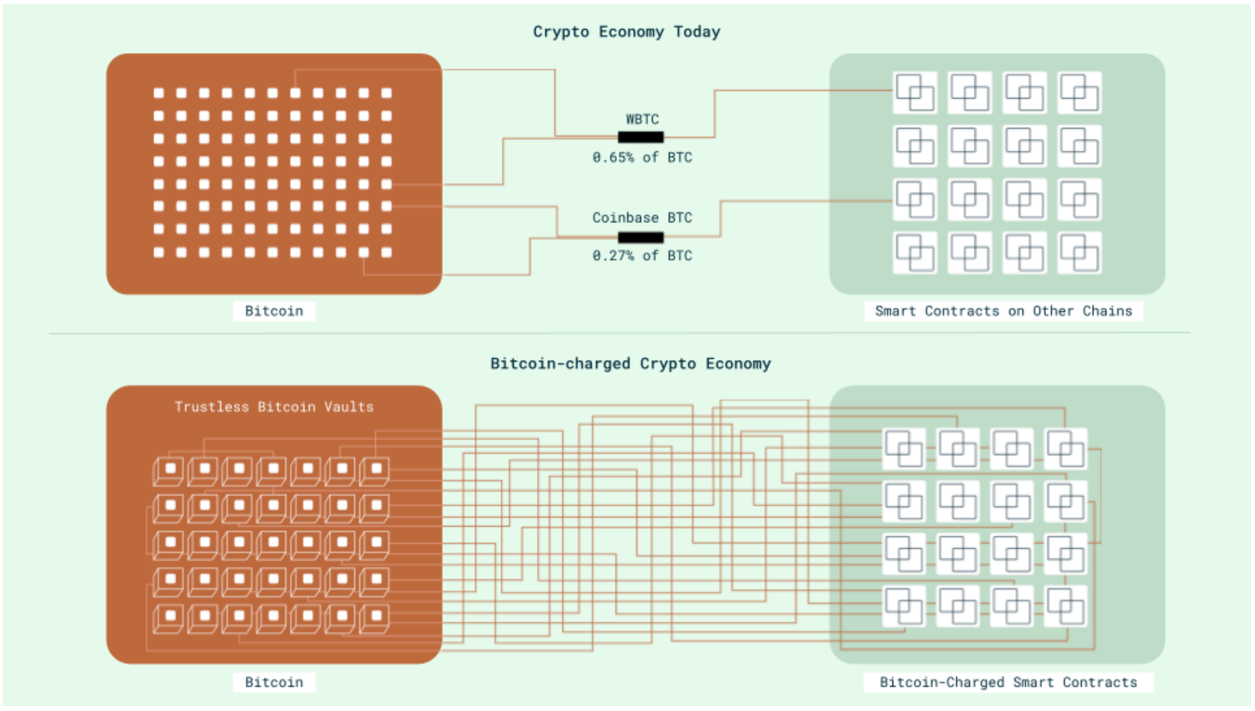
Much recent work has focused on new Bitcoin bridge designs with reduced trust assumptions, most notably through constructions based on the BitVM paradigm[2]. However, these designs[3] [4] [5] still face significant trust barriers due to the lack of covenant functionality in Bitcoin Script. As a result, these bridges need to be run by multiple third parties—a signer committee, a set of operators, and a set of challengers—on whose honesty and liveness the bridged BTC ultimately depends.

Bridges are a means, not the goal. The goal is for BTC holders to trustlessly participate in DeFi protocols on other chains. In this whitepaper, we show that even without a covenant soft-fork on Bitcoin, this goal can be achieved. Our solution is based on a new primitive: *trustless Bitcoin vaults*.

We will first motivate this primitive using the Babylon Bitcoin staking protocol. Then, we will explain it in the context of a simple peer-to-peer lending use case. Next, we will show how this

[1] https://coinmarketcap.com/currencies/wrapped-bitcoin/ Accessed: 29-Jul-2025
https://coinmarketcap.com/currencies/coinbase-wrapped-btc/ Accessed: 29-Jul-2025
[2] Robin Linus. (2023). *BitVM: Compute Anything on Bitcoin.* https://bitvm.org/bitvm.pdf
[3] Linus et al. (2025). *Bridging Bitcoin to Second Layers via BITVM2.* https://eprint.iacr.org/2025/1158
[4] Bal et al. (2025). *Clementine: A Collateral-Efficient, Trust-Minimized, and Scalable Bitcoin Bridge.* https://eprint.iacr.org/2025/776
[5] Lerner et al. (2024). *BitVMX: A CPU for Universal Computation on Bitcoin.* https://arxiv.org/abs/2405.06842

primitive has been integrated with existing, more complex lending protocols, as well as stable issuance and Perp DEX use cases.



2 Trustless Bitcoin Vaults

In August 2024, the Babylon team launched the Bitcoin staking protocol[6], enabling BTC holders to *trustlessly* stake their coins to secure Proof-of-Stake (PoS) networks—including both L1s and L2s. As of this writing, 44,000 BTC (worth $5.2 billion USD) have been staked through the protocol.

In Babylon's design, each staker creates a vault (a UTXO) that locks BTC using pre-signed Bitcoin transactions. These transactions are constructed such that the BTC can only be spent in two cases: either (1) the staker initiates an unstaking operation, or (2) a cryptographic proof is provided that the staker—or a delegated validator—double-signed blocks at the same height on the PoS chain. Thus, the protocol requires no trust assumptions: the BTC is never bridged to another chain, and each staker retains full control of her assets within her own vault.

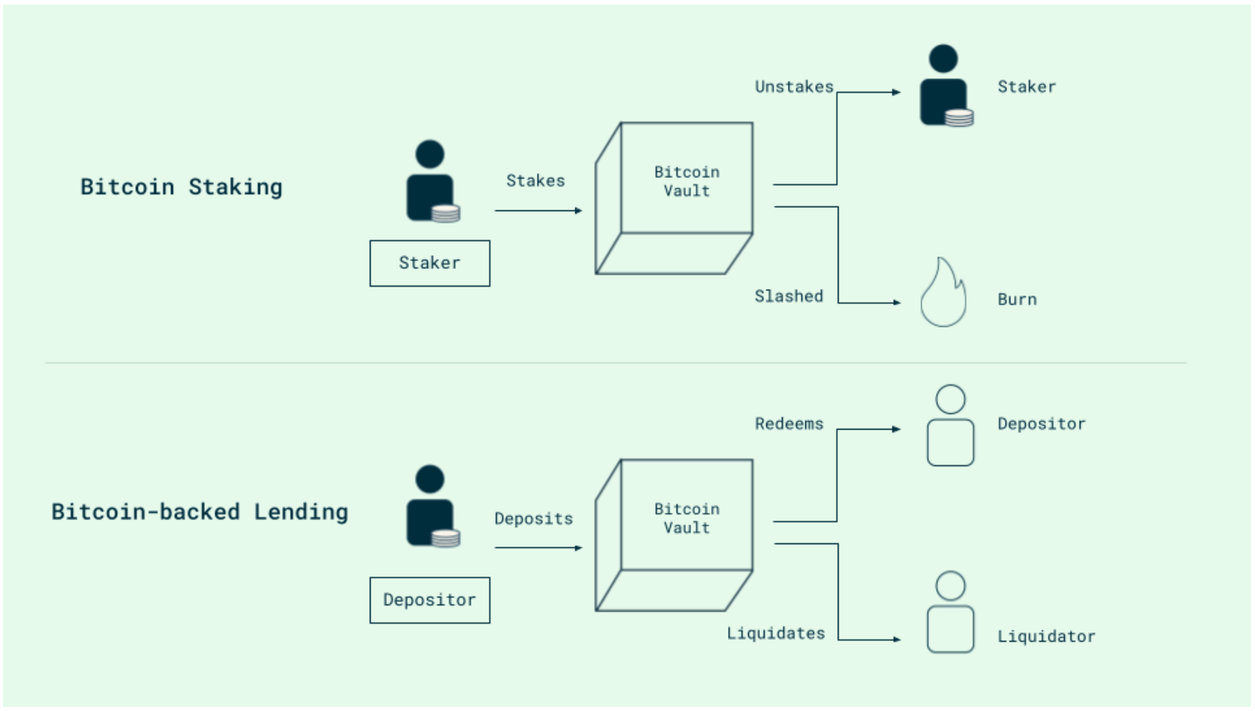
We propose generalizing the concept of a **trustless Bitcoin vault** to a foundational primitive for DeFi protocols. Any BTC holder who wishes to participate in a DeFi application creates a vault by pre-signing a set of Bitcoin transactions (possibly with other participants). The vault's

[6] (2023). *Bitcoin Staking: Unlocking 21M Bitcoins to Secure the Proof-of-Stake Economy*. https://docs.babylonlabs.io/papers/btc_staking_litepaper(EN).pdf

spending conditions depend on cryptographic proofs attesting to the state of the external smart contract.

Example: BTC-Backed Peer-to-Peer Lending

As another example, consider a simple lending scenario. Bob holds 1 BTC and wishes to borrow $50,000 in a stablecoin from Larry via a lending protocol on Ethereum. The deal is: if BTC's price drops below $50,000 during the loan period, Larry can liquidate the BTC collateral. Otherwise, if Bob repays the loan on time, he regains access to his BTC.



Traditionally, this requires trust:

- Bob might hand over BTC to Larry for safekeeping—but then must trust Larry to return it after repayment.
- Alternatively, Bob could keep the BTC and promise that Larry can liquidate under certain conditions—but then Larry must trust Bob.
- Another approach is for Bob to bridge his BTC to Ethereum and receive wrapped BTC as collateral—but this introduces trust in the wrapped token issuer.

Trustless vaults eliminate all such trust assumptions. Bob and Larry jointly pre-sign a set of Bitcoin transactions defining conditional spending rights:

- Bob can redeem the BTC if he repays the loan and the BTC price stays above $50K.
- Larry can liquidate the BTC if its price falls below $50K.

These conditions depend on the Ethereum smart contract state. To withdraw BTC, either party must provide a cryptographic proof:

- Bob provides a proof that he repaid the loan, and the price condition is met.
- Larry provides a proof that the BTC price fell below the threshold.

3 Proof Verification via BitVM3

The central technical challenge in enabling such trustless vaults lies in how Bitcoin can verify off-chain cryptographic proofs. Existing mechanisms like **hash time-locked contracts (HTLCs)**[7] and **extractable one-time signatures (EOTS)**[8] [9] provide proof via **secret revelation,** but there are limitations:

- **HTLCs** reveal a secret only if a specific event occurs off-chain associated with a counterparty's action. In atomic swaps, for example, the counterparty reveals a secret *if and only if* he claims the funds on a different chain. However, HTLCs suffer from an *abort problem*—one party can choose not to reveal the secret, disrupting the protocol. (This leads to the well-known *free-option* problem of atomic swap.)

- **EOTS**, as used in the Babylon Bitcoin staking protocol, allows BTC to be slashed upon detection of double-signing. The secret key is extracted and used to burn the staker's BTC. However, this mechanism is tailored to a specific event (double-signing) and cannot generalize to arbitrary DeFi conditions.

BitVM3[10] overcomes these limitations by generalizing secret revelation to arbitrary off-chain state proofs and by eliminating liveness dependence on the counterparty. It uses **zero-knowledge proofs (ZKPs)** and **garbled circuits**[11].

Here is how it works in the lending context:

[7] Spilman, J. (2013). *BIP 199: Hashed Time-Locked Contract Transactions*. Bitcoin Improvement Proposal. https://github.com/bitcoin/bips/blob/master/bip-0199.mediawiki
[8] Poettering and Stebila. (2014). *Double-authentication-preventing signatures*. In ESORICS. https://eprint.iacr.org/2013/333
[9] Ruffing, Kate, and Schröder. (2015). *Liar, liar, coins on fire!:Penalizing equivocation by loss of bitcoins*. In CCS. https://dl.acm.org/doi/10.1145/2810103.2813686
[10] Rubin J. (2025). *Delbrag*. https://rubin.io/public/pdfs/delbrag.pdf
[11] https://en.wikipedia.org/wiki/Garbled_circuit

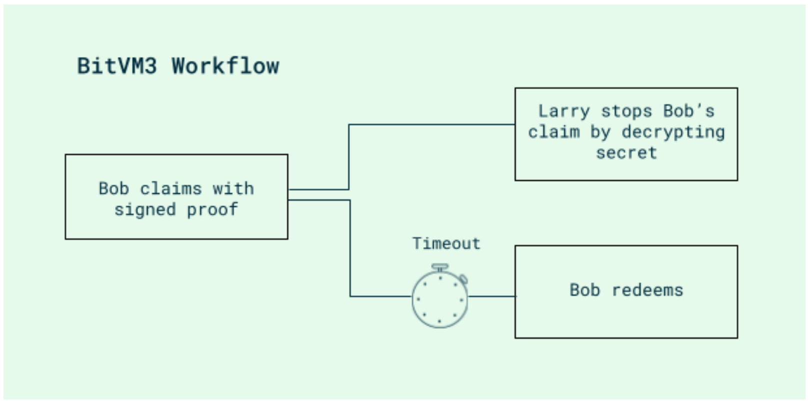
- Bob and Larry each generate and commit to a secret and a garbled circuit. Bob's circuit is designed such that, if Bob later attempts to withdraw Bitcoin without repaying his loan, Larry can use the garbled circuit along with Bob's signature on a claimed ZK proof to extract Bob's secret. Crucially, this extraction is only possible if the signed ZK proof is *invalid*—i.e., if Bob is cheating. Larry's circuit operates under the same principle.

- Bob verifies the garbled circuit committed by Larry, specifically, that the circuit enables extracting Larry's secret if Larry submits a signed, invalid ZK proof. Similarly, Larry verifies Bob's circuit.

- After verification, both Bob and Larry pre-sign transactions spending the locked Bitcoin, and store each other's garbled circuit for use in potential future challenges.

- When Bob wants to withdraw his BTC, he posts a claim transaction on Bitcoin.
    - In the happy path—when Bob has genuinely repaid—the claim goes unchallenged by Larry, and Bob withdraws the Bitcoin.
    - If Larry suspects fraud, he posts a challenge transaction on Bitcoin.
    - Bob must then respond with his signed ZK proof of repayment and price condition.
        - If the proof is *valid*, Bob successfully withdraws after a timelock.
        - If the proof is *invalid*, Larry extracts Bob's secret from Bob's garbled circuit (off-chain) and posts Bob's secret on-chain, preventing the withdrawal.

- Larry follows a similar flow when he wants to liquidate.


BitVM3 Workflow
Bob claims with signed proof
Larry stops Bob's claim by decrypting secret
Timeout
Bob redeems

This design ensures that **either party can unilaterally withdraw funds only by providing a valid ZK proof**, removing the need for mutual trust or third-party custodians. In essence, BitVM3 makes Bitcoin vaults programmable agents capable of enforcing complex logic tied to external smart contract states—without compromising Bitcoin's base layer security or requiring native smart contracts.

Cost Estimates

On-chain bond: In the typical (unchallenged) case, no ZK proof is posted on Bitcoin, and the cost of deposit and withdrawal is minimal (only 3 Bitcoin transactions[12] which totally cost $2.66 in our Bitcoin mainnet experiment). If a challenge occurs, the withdrawing party must post a Lamport signature of the ZK proof and its public inputs, totaling approximately 100KB. In our Bitcoin mainnet experiment, this transaction cost $93 in fees[13]. This fee is almost never paid in practice because a properly operating system should never have challenges. However, this amount should be locked in the vault by each party as a bond, in addition to the deposit used to pay for challenges, in case they occur. This amount will be refunded to each party after the vault is closed.

Note that this fee was more than $15K in an earlier experiment[14] with the BitVM2 technology, a precursor technology to BitVM3, where the ZK proof is directly verified on-chain rather than by leaking a secret off-chain using garbled circuits. **This amounts to a reduction by a factor of 170**, meaning that the bond can now be much smaller.

Off-chain cost: The primary off-chain cost involves generating and storing the garbled circuits used for potential challenges. In our benchmarks, generating 1 garbled circuit takes 20 minutes on a single core and can be further parallelized. Each party stores the counterparty's garbled circuit, whose size is 43 GB. This storage can be handled by each party with a local hard drive or cloud services for $1 per month[15].

Amortizing off-chain costs: Large borrowers may prefer to generate and store garbled circuits themselves to retain full trustlessness. However, smaller borrowers are more likely to delegate this task to professional operators. Importantly, these operators *cannot steal BTC* since the borrower still co-signs all transactions spending the Bitcoin.

In this model, each party (i.e., operators and Larry) generates a garbled circuit per counterparty, enabling them to challenge him. If, for example, Larry challenges an operator, the garbled circuit the operator generated for Larry is consumed. If the operator loses the challenge, they are disqualified from future vault withdrawals; if Larry loses, he forfeits the right to challenge future withdrawals. Because a failed challenge removes a party from the system, each party needs to generate circuits only *once per counterparty*, rather than for each deposit. This amortizes the off-chain cost.

[12] 3 transactions: Deposit, Claim, Withdraw optimistically
[13] https://mempool.space/tx/c77528a02b9ee33653d45174a32fd50cdc2d0c1b2cf2c8b6d666e0e6ed0b3c59
[14] https://x.com/dntse/status/1930272316124287393
[15] https://cloud.google.com/storage/pricing Accessed: 30-Jul-2025

4 Bitcoin Vault Security: A Comparison

Desired Actions	Lending using a discreet log contract (e.g. Lendasat[16])	Lending using generic BitVM bridge[17] [18] [19] [20]	**Lending using a trustless vault**
Bob and Larry create lending contract	Trustless	Trusts n-of-n signer committee and m-of-m operators	**Trustless**
Bob (borrower) can withdraw collateral	Trusts Larry	1-of-n signer committee 1-of-m operators 1 challenger	**Trustless**
Larry (lender) can liquidate collateral	Trustless	1-of-n signer committee 1-of-m operators 1 challenger	**Trustless**

Various prior approaches have sought to bring smart contract functionality to Bitcoin, but they typically introduce additional trust assumptions that **trustless Bitcoin vaults** avoid. To illustrate the differences, the table above compares the trust assumptions of several such methods, all within the context of the simple lending application between Bob (the borrower) and Larry (the lender). We assume all protocols rely on a trusted price oracle—an unavoidable requirement for implementing collateralized lending.

Discreet Log Contracts

Discreet Log Contracts (DLCs) have been employed in Bitcoin-based lending protocols such as **Lendasat** and **Lava**[21]. In a typical DLC-based lending setup, Bob locks his BTC into a UTXO that can be spent under two mutually exclusive conditions:

1. **Liquidation**: If the price oracle releases a signature certifying that the BTC price has dropped below $50,000, Larry (the lender) can use this signature to claim the BTC collateral.

[16] https://lendasat.com/
[17] Linus et al. (2025). *Bridging Bitcoin to Second Layers via BITVM2.* https://eprint.iacr.org/2025/1158
[18] Bal et al. (2025). *Clementine: A Collateral-Efficient, Trust-Minimized, and Scalable Bitcoin Bridge.* https://eprint.iacr.org/2025/776
[19] Lerner et al. (2024). *BitVMX: A CPU for Universal Computation on Bitcoin.* https://arxiv.org/abs/2405.06842
[20] https://docs.bitlayer.org/docs/Learn/Bitlayer%20Rollup/bridge
[21] https://www.lava.xyz/

2. **Repayment**: If Bob obtains a secret—issued to him only upon repayment of the loan—he can use it to withdraw the BTC.

This construction inherits the same limitation as HTLCs: it grants **Larry a free option** at the point when Bob attempts to repay. By refusing to release the secret required for withdrawal, Larry can unilaterally prevent Bob from reclaiming his collateral—even if Bob intends to repay the loan. Lendasat allows Bob to repay the loan unilaterally, but this requires repaying in Bitcoin, which is undesirable. This asymmetry means **Bob must trust Larry** to act honestly when the repayment condition is satisfied.

In contrast, **Larry does not need to trust Bob** for the liquidation path. He only requires a valid signature from the price oracle to execute liquidation, making that part of the protocol effectively trustless for the lender.

BitVM Bridge

BitVM enables smart contract state verification on Bitcoin and can be used to build a *general-purpose bridge* instead of a *trustless vault*. The key difference is:

- A **general-purpose bridge** issues *fungible wrapped BTC* on a smart contract chain, which *any party* can redeem.
- A **trustless vault** is tied to a *specific application and known parties* (e.g., Bob and Larry), who are the only ones able to redeem the BTC.

To support open redemption in bridges, multiple third parties are required—bridge operators, signer committee, and challengers—each introducing trust assumptions.

Operators: Bitcoin UTXOs must specify a fixed set of parties who can redeem them. In the BitVM bridge, these parties are the *operators*. For Bob to withdraw BTC, he must rely on at least one operator to send him BTC and later redeem it from the bridge. Moreover, Bob requires *all* operators to participate while depositing BTC. In contrast, trustless vaults eliminate operators entirely—Bob and Larry are predefined and control redemption directly.

Signer committee: A signer committee is used at *peg-in* to pre-sign BTC transactions, ensuring that an operator who fronts capital to Bob during peg-out can later claim the BTC. However, if the committee colludes, they can sign an alternate transaction to steal the BTC. As such, the BitVM bridge is a multi-sig bridge with an advantage that the signer committee is not needed to approve peg-out, and hence only 1 of n committee members needs to be honest during peg-out. Trustless vaults avoid signer committees altogether: every transaction spending the locked BTC requires signatures from *both* Bob and Larry, so the protocol is trustless from their point of view.

Challengers: In BitVM, *challengers* reveal secrets to prevent fraudulent withdrawals—e.g., stopping Larry from liquidating BTC when price conditions aren't met. Some BitVM bridge designs require *permissioned challengers*, forcing Bob to trust them to act when needed.

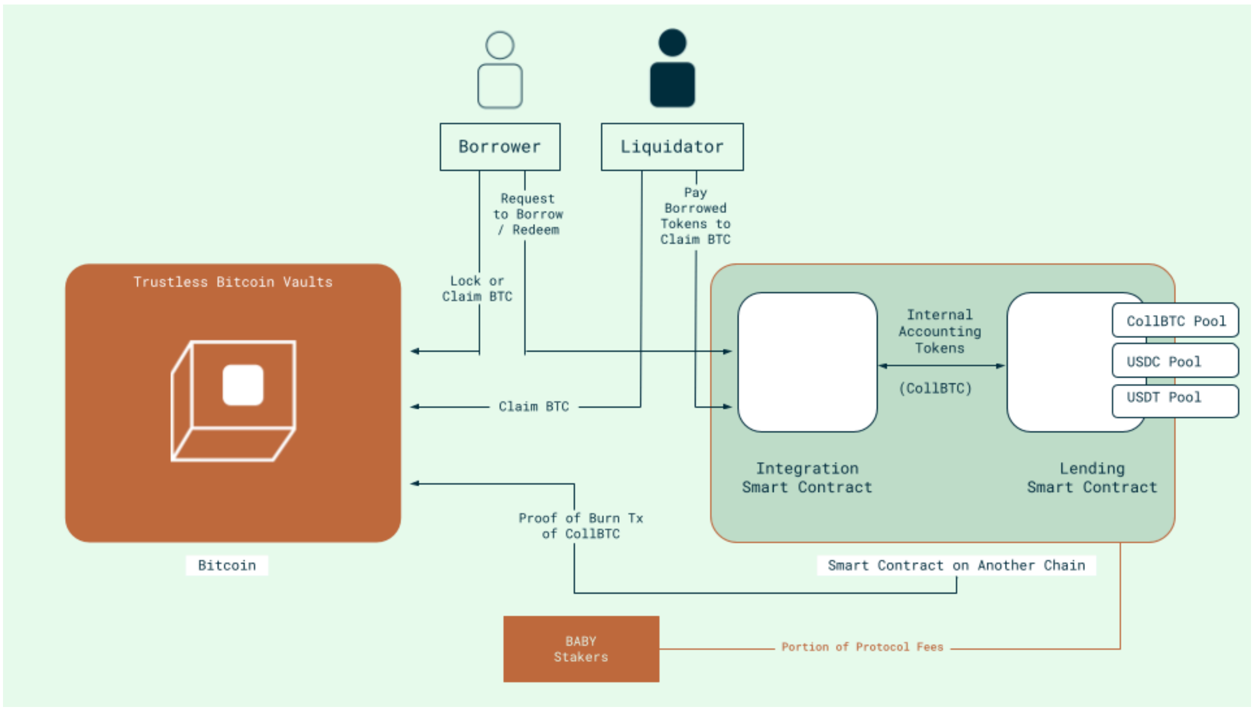
Trustless vaults allow Bob himself to act as a challenger, removing the need to trust external parties.

5 BTC-Backed Lending

We have used the simple peer-to-peer lending scenario as a running example to illustrate the basic mechanics and security guarantees of trustless Bitcoin vaults. In this section, we will do a deeper dive into this important application, focusing on integration with existing lending protocols. In the next two sections, we will discuss other applications of trustless Bitcoin vaults.

Overview

Overcollateralized lending is a popular use case on different blockchains. Today, the largest lending protocol, Aave, has the largest TVL (~57b USD) across all DeFi applications. However, the only form of bitcoin that is widely accepted in lending protocols is WBTC and Coinbase wrapped BTC, which have very limited adoption relative to the supply of bitcoin, as well as compared to ETH (e.g., WBTC and Coinbase wrapped BTC combined TVL is less than ⅓ of the TVL of various ETH tokens on Aave). Enabling *native bitcoins* as collateral will unlock a huge market opportunity with the rest of bitcoin supply with on-chain lending protocols.



**As the protocol's activity grows, it will consistently generate revenues. Part of those revenues will be distributed to BABY Stakers. See Section 10 for more details on protocol fee distribution models.*

Architecture and Workflow

The BTC-backed lending system works by linking trustless vaults on Bitcoin with existing lending protocols deployed on smart contract chains via verifiable integration logic. The integration is expected to be seamless as long as the set of liquidators can be specified programmatically. The high-level process is outlined below:

Collateralization and Loan Origination:

1. A borrower deposits BTC into his own Bitcoin vault.

2. The borrower sends metadata about the vault (e.g., deposit TXID, vault ID, target lending contract address, desired loan amount/token) to a smart contract on a lending-enabled chain.

3. The smart contract verifies the BTC vault via a Bitcoin light client, then mints an internal accounting ERC20 token (collBTC) representing the locked BTC.

4. The collBTC is deposited directly into a lending protocol, where it backs a loan in the desired stablecoin or token.

5. The borrowed funds are transferred to the borrower’s wallet on the smart contract chain.

Redemption & Loan Repayment:

1. The borrower repays the borrowed tokens to the integration smart contract.

2. The contract closes the loan position and burns the associated collBTC tokens.

3. A ZK proof of the burn is generated off-chain and submitted by the borrower to the Bitcoin vault. The proof is verified by BitVM3.

4. After the timeout, the BTC is unlocked and returned to the original depositor.

Liquidation Flow:

1. If the BTC price falls below the liquidation threshold for the vault, the borrower’s position becomes eligible for liquidation.

2. Whitelisted liquidators monitor price feeds and vault state.

3. A liquidator repays the borrower’s debt to the integration smart contract. The contract receives collBTC and burns it on-chain.

4. A burn-proof is generated and submitted to the Bitcoin vault created by the liquidator.

5. After the timeout, the liquidator can redeem the BTC as compensation for closing the position.

This cross-chain lending workflow ensures trustless collateral custody, while maintaining compatibility with existing lending protocols on Ethereum, rollups, or other smart contract chains.

Benefits & Advantages

Native Bitcoin as Collateral: Unlocks the full value of BTC for on-chain credit markets without the need for wrapped tokens or bridges.

Trustless, Cross-Chain Lending: Combines Bitcoin’s base-layer security with the capital efficiency of DeFi on smart contract chains.

Composable Collateral Layer: BTC vaults can be reused across multiple applications—lending, stablecoins, and derivatives—without requiring separate integrations.

Minimized Counterparty Risk: Removes reliance on centralized lenders or custodians, providing a censorship-resistant and transparent alternative.

Open Participation: Enables liquidators, borrowers, and developers to plug into the protocol with minimal onboarding or custom infrastructure.

5.1 Vault Design and its Security

The simple peer-to-peer lending example in Section 2 involved only two parties—a lender and a borrower. Real-world lending protocols involve a borrower, multiple liquidators, and multiple lenders who provide capital to the lending contract (a lending pool).
The table below generalizes the trust assumptions table from Section 4 to this more general context for a proper design of the trustless vault.

Desired Actions	Lending using a discreet log contract (e.g., Side[22])	Lending using a generic BitVM bridge	**Lending using a trustless vault**
Borrower can deposit collateral	Trusts k-of-n committee	Trusts n-of-n signer committee and m-of-m operators	Trusts k-of-n liquidators j-of-m large lenders
Borrower can withdraw collateral	Trusts k-of-n committee	Trusts 1-of-n signer committee 1-of-m operators 1 challenger	**Trustless**
Liquidator can liquidate collateral	Trusts k-of-n committee	1-of-n signer committee 1-of-m operators 1 challenger	**Trustless**
Large lender can withdraw from lending contract	Trusts k-of-n committee	1-of-n signer committee 1 challenger	**Trustless**
Small lender can withdraw from lending contract	Trusts k-of-n committee		Trusts (n-k+1)-of-n liquidators or (m-j+1)- of-m large lenders

Trustless vaults

Both the borrower and the permissioned liquidators should co-sign Bitcoin transactions to create a vault so that they can withdraw BTC trustlessly when the correct conditions are met—just like in the two-party vault setup. To prevent censorship of new deposits, vault creation requires a threshold **k of n** liquidators to co-sign transactions, rather than all. The borrower and the liquidators should also act as challengers of malicious withdrawals.

A whitelisted set of large lenders—those who have deposited significant capital into the lending contract—is also allowed to co-sign the vault and challenge malicious withdrawals. This enables them to ensure the Bitcoin collateral is secure so that they can later withdraw capital from the lending contract. If BTC were withdrawn without repayment or liquidation, the lending contract would be unable to repay lenders. Other small lenders get the same guarantees as long as sufficiently many liquidators or sufficiently many large lenders are honest.

[22]https://docs.side.one/resources/whitepaper/whitepaper/native-btc-collateralized-lending-system/lending-workflow-overview

Discreet Log Contracts (Side Protocol)

Side Protocol enables lending from pools using *Discreet Log Contracts (DLCs)*. Since not all lenders are known in advance, a designated *committee* represents them and signs the DLC as the borrower's counterparty. As a result, the borrower trusts the committee to both initiate the deposit and reveal a secret during withdrawal.

During liquidation, upon receiving the price oracle's signature on a Bitcoin price below the liquidation threshold, the committee takes custody of the Bitcoin. The committee is responsible for selling the Bitcoin to liquidators that pay the borrower's debt to the lending contract. Thus the liquidators trust the committee. Also, the lenders trust the committee to not take the Bitcoin for themselves upon liquidation.

BitVM Bridge

The same trust assumptions hold for the borrower and the liquidator as in the two-party example. Lenders never withdraw Bitcoin from the bridge, but trust that the native BTC will not be stolen. Therefore, they trust the signer committee and the challengers who ensure that the Bitcoin is not stolen. Unlike the trustless vault and DLC, liquidators and large lenders cannot be included in the bridge's signer committee because this would introduce trust on the lenders for other users of the bridge.

6 BTC-Backed Stablecoin

Stablecoins are foundational to the crypto economy, providing a stable unit of account and a reliable medium of exchange. However, most existing stablecoins are either fiat-backed and custodial (e.g., USDC, USDT) or overcollateralized using assets like ETH (e.g., DAI). Despite being the most widely held crypto asset, Bitcoin has not been meaningfully integrated into decentralized stablecoin systems due to the absence of native, trust-minimized infrastructure.

Trustless Bitcoin vaults introduce a new paradigm for stablecoin issuance: one in which **native BTC** can serve as fully verifiable collateral for the minting of decentralized USD-pegged tokens. This removes the need for custodians, bridges, or synthetic representations, allowing Bitcoin holders to unlock liquidity while preserving custody and sovereignty.

Architecture & Workflow

The BTC-backed stablecoin system leverages the trustless vault primitive described earlier, combined with cross-chain light clients and verification protocols:

1. **Collateralization**: A user deposits BTC into a trustless vault he created on the Bitcoin chain. This deposit is recorded and verifiable via light clients deployed on a smart contract chain.

2. **Stablecoin Minting**: Upon verification, a smart contract mints stablecoins (e.g., USDB) at a defined collateralization ratio. These tokens can then be used freely across DeFi protocols.

3. **Utility and Circulation**: The stablecoins can be used for trading, payments, lending, or as liquidity in other protocols. Peg stability can be supported by redemption mechanisms, arbitrage incentives, and optional Peg Stability Modules (PSMs).

4. **Redemption**: To redeem BTC, the stablecoins must be burned on the smart contract chain. A ZK proof of this burn is submitted to the vault, initiating a claim. After the challenge window, the BTC is released back to the user.

5. **Liquidation**: If BTC value falls below a safety threshold, whitelisted liquidators can burn stablecoins, submit proofs, and claim the BTC collateral — ensuring solvency and trustless enforcement of risk parameters.

Benefits & Advantages

- **Native BTC Collateral**: Eliminates reliance on wrapped assets or bridges.

- **Trustless, Auditable, and Decentralized**: Fully verifiable without requiring intermediaries.

- **Cross-Chain Liquidity**: Stablecoins can circulate across any integrated chain.

- **Composability**: Stablecoins can plug into lending protocols, DEXs, or yield strategies.

- **Censorship Resistance**: Vaults operate entirely within Bitcoin’s security model.

7 BTC-Backed Perps DEX

Perpetual futures (perps) dominate crypto trading volume, yet most on-chain derivatives platforms are built around ETH or stablecoin collateral. Despite its scale and significance, Bitcoin is largely excluded from on-chain perps due to technical and custodial limitations. Trustless Bitcoin vaults solve this by enabling decentralized, BTC-collateralized perps trading without bridges, wrapped tokens, or custodians.

This allows Bitcoin holders to margin long or short positions across chains while retaining full control over their BTC — unlocking new capital efficiency and expanding the role of BTC in decentralized finance.

System Design & Workflow

The integration of BTC vaults into perps infrastructure involves the following process:

1. **BTC Collateralization**: A trader deposits BTC into a trustless vault on the Bitcoin network. The vault metadata is synced to a perps-compatible chain via light client infrastructure.

2. **Margin Representation**: A corresponding internal accounting token (collBTC) is minted on the smart contract chain, representing the vault deposit. This margin token is used as collateral for perps trading.

3. **Trading Execution**: Traders use the collBTC token to open long or short positions on the perps DEX. The protocol's margin engine monitors PnL and risk thresholds.

4. **Settlement & Redemption**: Upon position closure, traders can burn the margin tokens and submit a proof to the Bitcoin vault. Once validated, the BTC is unlocked and returned.

5. **Liquidation**: If a margin position falls below maintenance thresholds, liquidators can purchase and burn collBTC, generate a ZK proof, and claim the BTC from the vault after a timeout.

Benefits & Advantages

- **Enables Native BTC as Margin**: Traders can use BTC directly without converting or wrapping.

- **Minimizes Counterparty Risk**: No custodians or synthetic assets required.

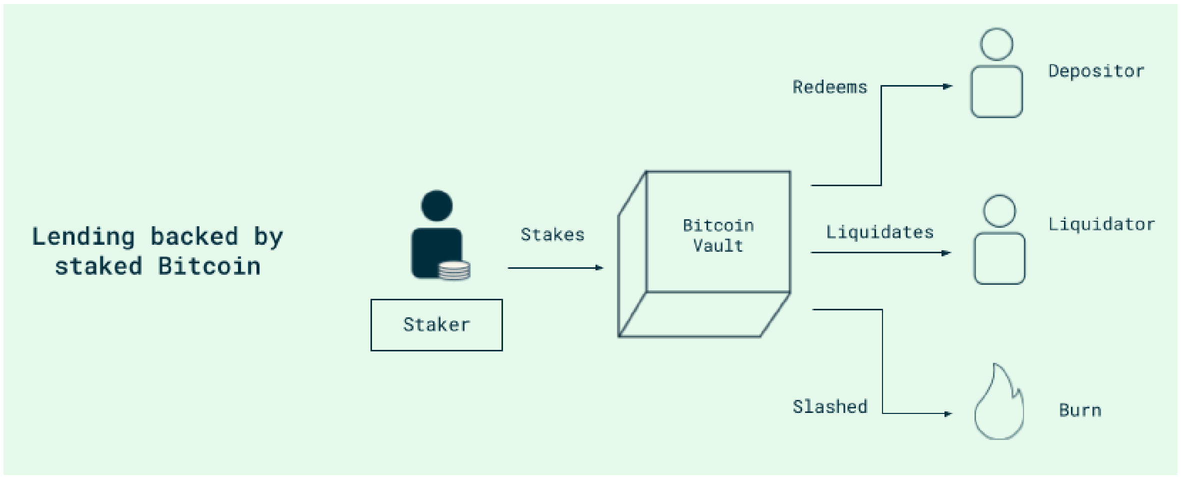
- **Composable with DeFi**: Margin tokens can be reused across positions.

- **Cross-Chain Deployment**: Supports any chain with integration and light client support.

- **Capital Efficiency for BTC Holders**: Unlocks new yield and leverage opportunities without giving up custody.

8 Integration with the Babylon Bitcoin staking layer

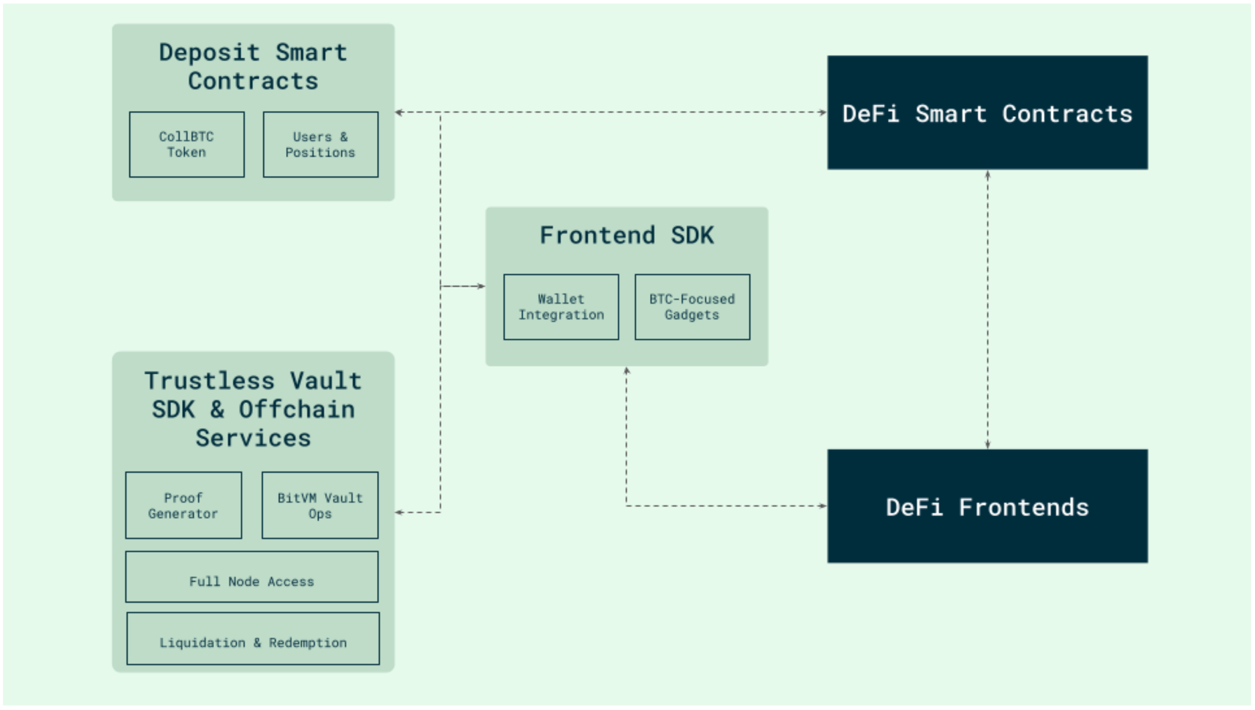
The Babylon Bitcoin staking protocol provides staking rewards to Bitcoin stakers securing Proof-of-stake networks and rollups (BSNs). By using staked BTC instead of BTC to participate in DeFi protocols, capital efficiency is increased. Liquid staking protocols achieve this capital efficiency by minting liquid staking tokens based on staked BTC; however, currently the minting is done through a trusted entity or a committee. Bitcoin vaults provide an alternative trustless way for staked BTC to participate in DeFi protocols on smart contract chains. Take the example of the lending application. Since both lending and staking can be supported by trustless vaults, a staked BTC can be collateral for BTC-backed lending by having a staker create a single vault with three spending conditions: 1) redemption/unstaking, 2) liquidation, and 3) slashing. Condition 1 is fulfilled if the BTC price is above the liquidation threshold and the staker/borrower wants to redeem/unstake. Condition 2 is satisfied if BTC price falls below a threshold. Condition 3 is satisfied if the staker or the finality provider it delegates to double-signs. With this contract, the staker/borrower can get yield on its BTC while using it as collateral to obtain liquidity.



9 Multi-Chain Deployments & Integration of Trustless Vaults

With the global decentralized finance (DeFi) projected to reach $231.19 billion by 2030[23], the DeFi community has attracted an active developer community of thousands[24]. Trustless Bitcoin vaults have the potential to enable all DeFi smart contracts with access to native BTC to significantly strengthen and enlarge their existing collateral base. To realize such potential, 1) on-chain smart contract interfaces, 2) trustless vault SDK and off-chain services, as well as 3) a frontend SDK, need to be provided to the developers and participants of DeFi applications to access and program trustless vaults for their specific use cases.

[23] Grand View Research, "Decentralized Finance Market Size | Industry Report, 2030, 2024 https://www.grandviewresearch.com/industry-analysis/decentralized-finance-market-report
[24] Electric Capital Developer Report 2024. https://www.developerreport.com/reports/devs/2024



Deposit Smart Contracts

The goal of the deposit smart contracts is to minimize the efforts required to DeFi smart contracts for the adoption of trustless vaults. A particular version of a deposit smart contract may need to be developed for each specific use case (e.g., lending, stablecoin) and each smart contract language (e.g., EVM, Rust). Such contracts deployed on multi-chains rely on an on-chain Bitcoin light client to capture events on the Bitcoin chain, regarding the actions performed on BTC vaults, and provide interfaces to DeFi smart contracts in the most transparent and compatible manner. For example, on EVM, such deposit contracts will create a fungible ERC20 token as the internal accounting tokens representing the non-fungible bitcoin UTXOs locked in the vaults, allowing ERC20 tokens smart contracts to integrate seamlessly. On the Solana Virtual Machine (SVM), such a deposit smart contract (or so-called program) will create a fungible SPL token to represent locked BTC in vaults. The deposit smart contract can exist on any chain with a capable virtual machine runtime as well as a Bitcoin light client.

For example, a deposit smart contract for a Bitcoin-charged lending protocol on EVM would need to implement the following interfaces:

- Accessing Bitcoin vault information from the on-chain Bitcoin light client (or verifying with a verification smart contract regarding information obtained from an off-chain Bitcoin light client)
- Minting/burning controller for CollBTCToken

- Calling the relevant functions of the smart contracts of the lending protocol for 1) pool creation and parameter setting (such as for allowlisting liquidators) and 2) position creation, redemption, and liquidation.

The Babylon team plans to develop such deposit smart contracts for key DeFi protocols on EVM chains initially, including Babylon Genesis, which is equipped with an on-chain Bitcoin lightclient, to ease the integration effort of developers from various ecosystems, while over time supporting third-party developments of more such smart contracts for more use cases.

Trustless Vault SDK & Off-Chain Services

Trustless vault SDK and off-chain services are required for DeFi applications that integrate trustless vaults for direct access to the native BTC. For example, liquidators of a lending protocol need to be able to take over the corresponding native BTC put up as collateral by a borrower during a liquidation event. To enable developers to achieve these, an open-source Software Development Kit (SDK) and off-chain service software needs to be developed to allow such services to be hosted by any party.

They will allow DeFi participants to perform the following actions:
- Participating in the pre-signing of transactions at vault creation
- Access to a full node for the smart contract blockchain to generate proofs of on-chain events and activities
- Executing Bitcoin transactions to claim native BTC collateral by submitting the necessary proofs

Frontend SDK for Bitcoin-Centric User Experience

As the global adoption of Bitcoin accelerates, more and more DeFi applications may not only integrate with trustless vaults at the protocol level, but also adapt their frontends to provide more seamless UI/UX for BTC holders. On the other hand, there is a rising trend for DeFi aggregation services that provide unified UI/UX to users while connecting them to various often competing DeFi applications behind the scenes[25]. Especially along the journey of onboarding more native Bitcoin users to DeFi, such aggregators can play a pivotal role.

For widespread adoption of trustless Bitcoin vaults in both scenarios, it is critical to have frontend SDKs that provide smooth developer experience to abstract away the complexities of low-level details around Bitcoin. This will significantly reduce the overhead of the inclusion of native bitcoin in DeFi applications.

Babylon team plans to develop a frontend SDK that provides a comprehensive toolkit to enable any web or mobile application to integrate trustless Bitcoin Vaults for the end-to-end user

[25] Journal Future Internet, 2024, https://www.mdpi.com/1999-5903/16/3/76

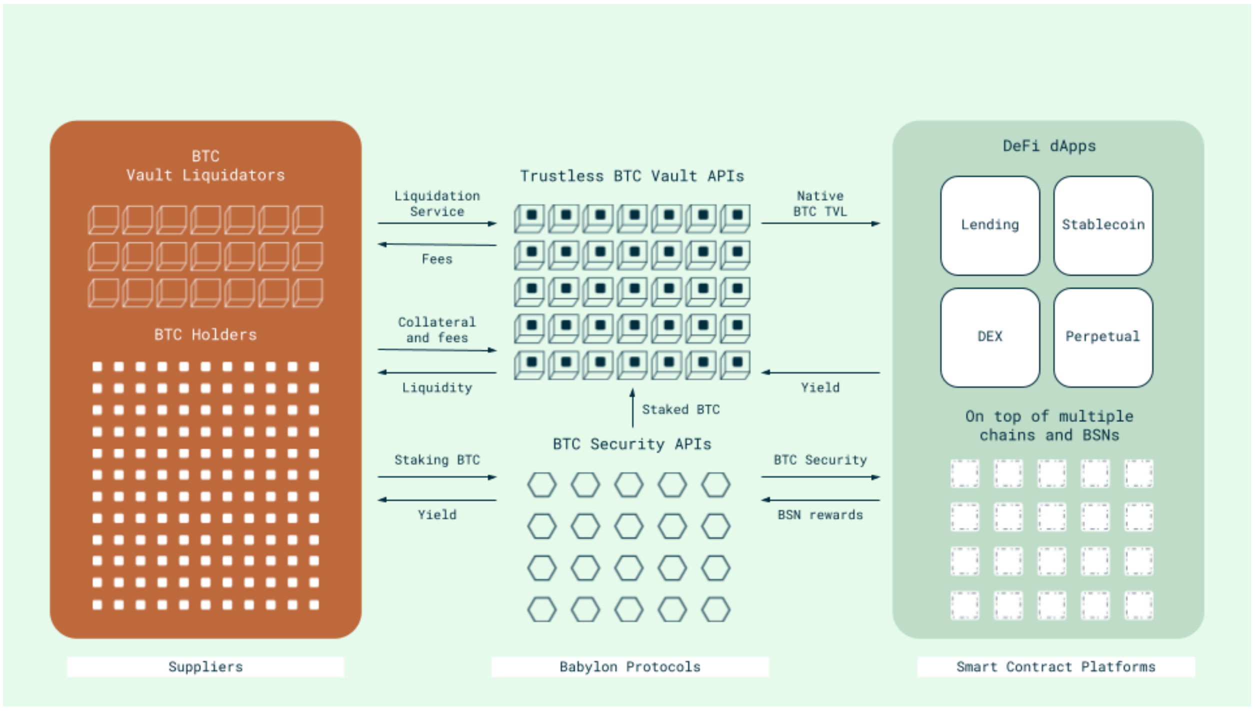
journey, including interactions with the deposit smart contracts and off-chain trustless vault and services SDK.

10 Launch Strategy and Protocol Fees Routing

Babylon is a Bitcoin-native infrastructure protocol that enables both security and liquidity to flow across blockchain ecosystems. Its native token, BABY, functions as the gas asset, governance unit, and staking token for securing the Babylon network. Babylon introduces the concept of Bitcoin-Supercharged Networks (BSNs)—external blockchains that can be secured by BTC staking through Babylon's decentralized coordination layer. As the first BSN, Babylon Genesis, a Cosmos-SDK–based Layer 1 chain, serves as the neutral coordination and control plane for BTC staking and routing.

In parallel, the Babylon team is building a suite of APIs and vault primitives to bring BTC liquidity into DeFi systems as outlined in this paper. Together, these services form the foundation of Babylon protocols' role as the base layer of a broader BTCFi ecosystem.

This diagram illustrates the Babylon-enabled BTCFi architecture, where BTC holders supply collateral or staking assets into two core modules: the Vault API for decentralized liquidity, and the Security API for securing external BSNs. Liquidators enable redemptions and liquidations, while DeFi apps on Ethereum, rollups, and BSNs consume BTC-backed assets and security. Babylon protocol acts as the trustless coordination layer that routes native BTC across ecosystems.



To bootstrap adoption of the Bitcoin Vault standard, Babylon will allocate targeted BABY rewards to incentivize early DeFi partners. These incentives will be structured to encourage:

- Integration of Bitcoin Vault APIs into existing DeFi protocols (e.g., lending, stablecoins, perpetuals)
- Integration of Bitcoin Vault APIs into DeFi protocols on Babylon Genesis, Babylon EVM, and BSNs
- Deployment on Ethereum mainnet, popular rollups, and other mainstream EVM chains
- Development of vault-compatible frontends and liquidator infrastructure.

Once the core vault and Liquidator services demonstrate stability across Ethereum and rollup ecosystems, Babylon governance will be able to:

- Open the Vault deployment SDK to additional Layer 1 chains (e.g., Solana, Sui),
- Offer "Babylon Inside" vault-as-a-service modules for native BTC integration.

For newer chains, Babylon may shift from an incentive-driven model to a fee-based deployment model. This evolution positions Babylon as a foundational BTC infrastructure layer, with recurring network usage fees from cross-chain deployments (akin to software licence fees).

Babylon's upcoming BTC liquidity software solution will enable native Bitcoin assets to connect with external smart contract platforms — including, but not limited to, Bitcoin-Supercharged Networks (BSNs), Ethereum Layer 1, and its rollups. Rather than competing with these chains, Babylon Genesis operates as the BTC control plane and base layer, coordinating and enabling cross-chain BTC flow. In return for providing this infrastructure, Babylon protocols programmatically capture and route a portion of the protocol-level usage fees.

At the core of this infrastructure, Babylon Vault APIs enable the technological backbone for decentralized BTC collateralization and trustless redemption. As BTC enters and exits the vault, context-specific fees are applied programmatically across various DeFi applications. These fees may be denominated in native BTC, aligning long-term incentives with BTCFi system growth.

Additionally, similar to another recent proposal[26], deflationary protocol features may be introduced on Babylon Genesis to ensure smooth and secure operation of this foundational layer — for example, by deploying an automated on-chain auction system where BTC-denominated fees are auctioned for BABY. The winning bidder receives the BTC, while the spent BABY is programmatically burned, eliminating the need for discretionary treasury management or human intervention.

Importantly, all of the mechanisms described above are proposals under active design and discussion. Final implementation details—including fee structures, staking rules, and integration rollout—will be subject to Babylon governance approval.

---

[26] *Programmatic Deflation of BABY.*
https://forum.babylon.foundation/t/programmatic-deflation-of-baby/676/25

11 Conclusion

On February 11, 2009, Satoshi Nakamoto introduced Bitcoin to the world:

> "I've developed a new open source P2P e-cash system called Bitcoin. It's completely decentralized, with no central server or trusted parties, because everything is based on crypto proof instead of trust."[27]

In Bitcoin, Nakamoto planted a seed of trustlessness. This seed has since inspired a whole economy of trustless smart contracts on many other blockchains. Sixteen years later, it is high time to integrate the initial seed with this smart contract economy into a trustees whole.

Acknowledgements

Our BitVM3 work benefits from collaboration with the BitVM alliance. We would like to thank particularly Hakan Karakuş from Citrea for his work on the Groth16 verifier circuit, and Robin Linus for the many discussions. We would also like to thank Bin Ren and Orfeas Stefanos Thyfronitis Litos for the many in-depth discussions on Bitcoin lending protocol designs, Weikeng Chen for discussions on the cryptography of garbled circuits, and Lukas Aumayr for discussions on BitVM.

[27] Nakamoto. (2009). *Bitcoin open source implementation of P2P currency.* P2P Foundation. https://satoshi.nakamotoinstitute.org/posts/p2pfoundation/1/