Maelstrom

We are entropy

Bitcoin Grant Program

Annual Report

June 2026

Table Of Contents

Executive Summary

The Maelstrom grant program started in October 2024 and in the program's history, five developers have been financially supported. The program is aiming to provide its grantees with relatively stable and long term funding. The developers are working on open source technology to improve Bitcoin, with respect to scalability, robustness and privacy. The program currently supports four developers: Rkrux, Stratospher, Benalleng and Macgyver. Two of the grantees work on Bitcoin Core (Rkrux and Stratospher), while the other two (Benalleng and Macgyver) work on improving Bitcoin's privacy.

This brief report provides a basic overview of the technical work the grantees have been doing over the past 12 months and a summary of their future plans. A brief executive summary of the work of each of the four developers is provided below:

- **Rkrux** has been focusing on Bitcoin Core review work and in 2025 made 1,155 comments to the software repository, making him one of the top reviewers. He has also focused on the wallet and maintenance work by fixing wallet bugs, improving documentation and removing outdated code.
- **Stratospher** has been working on Bitcoin Core focusing primarily on hardening consensus-critical validation code and the P2P network by improving code correctness, consistency and privacy. She also handled edge-case bugs in validation logic and the wallet, which are unlikely to trigger in practice but are worth cleaning up to future-proof the code against wrong assumptions. Additionally, she worked on cryptographic code for DLEQ proofs in libsecp256k1, in relation to Silent Payments.
- **Benalleng** works full time on Payjoin, an interactive transaction system which allows senders and receivers to each contribute inputs to a Bitcoin transaction, breaking surveillance heuristics and improving scaling. This increasingly popular transaction process has been adopted by wallets such as Bull Bitcoin and Cake Wallet, thanks in part to contributions from Ben.
- **Macgyver** is working on another privacy technology, Silent Payments. This can help improve privacy by reducing the need to re-use addresses. Over the last 12 months, the protocol has matured significantly with wallet adoption, scanning performance, specification work and hardware signer interoperability. Achieving a merge in Bitcoin Core remains a key milestone for the protocol and Macgyver is focused on removing blockers in this area.

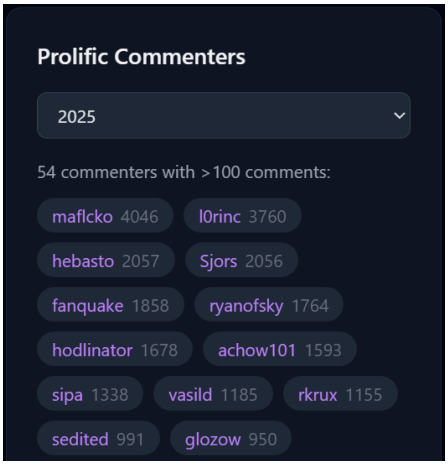
Jonathan Bier
Grant Program Administrator

Rkrux

- **Grantee since:** Oct 2024
- **Github:** https://github.com/rkrux
- **Grant Focus:** Bitcoin Core
- **Specialist Area(s):** Review work, MuSig2, Descriptors, Wallet, PSBT

Rkrux works across the Bitcoin Core software repository, mostly focusing on review work. Review is incredibly important and often underappreciated work. Due to the potentially severe consequences of bugs, every change to Bitcoin Core must be carefully reviewed by multiple qualified developers before it can be merged. Review is the project's primary defense against bugs, vulnerabilities and unintended behavior reaching the software. Software that thousands of nodes, which often process significant amounts of value, rely on. An overlooked flaw in consensus code could cause a chain split or cause a node to crash, or a bug in the wallet could cause the loss of funds. Review work is therefore one of the major bottlenecks.

Rkrux is an experienced software developer. He started contributing to Bitcoin in 2022 and to Bitcoin Core in 2024. Since then, Rkrux has built up his knowledge on the codebase and is now a valuable resource to the project. Rkrux started his work focusing on the wallet and has since broadened out to more areas of the codebase. In 2025, Rkrux reviewed more than 200 pull requests (PRs) and made 1,155 review comments, making Rkrux the 11th most prolific commenter, according to a dashboard built by Niklas Gögge. In the first five months of 2026, Rkrux made more than 400 PR comments. Rkrux has made 43 PRs to Bitcoin Core, with his first substantive PR occurring in early 2025. Much of the work is improving codebase architecture and maintainability, along with fixing wallet bugs, removing expired or outdated features, improving RPC documentation and functional tests, and raising follow-ups for the issues raised in PR reviews.



Rkrux reviewed every PR in the MuSig2 project that adds support for parsing MuSig descriptors, parsing MuSig keys in Miniscript expressions, importing such descriptors in the wallet, generating corresponding addresses to receive bitcoin, building PSBTs containing MuSig inputs & outputs and creating transactions spending from inputs that involve MuSig aggregate keys. This makes multisig transactions look like single-sig transactions on the blockchain, significantly improving user privacy and reducing fees. Rkrux was also involved in deprecating legacy wallets in favour of modern descriptor based wallets, that makes the Bitcoin Core wallet interoperable with other wallets in the ecosystem, while enabling easier backups of more sophisticated setups.

Stratospher

- **Grantee since:** Nov 2025
- **Github:** https://github.com/stratospher
- **X:** @spherostrat
- **Grant Focus:** Bitcoin Core
- **Specialist Area(s):** Validation, P2P network & Cryptography

Overview

Over the last 6 months, Stratospher has been contributing across the Bitcoin Core codebase, primarily in validation and the P2P network - both through her own PRs and by reviewing others' work in these security-critical subsystems. She has fixed edge-case bugs in consensus-critical validation code and in the wallet, though they are unlikely to trigger in practice, cleaning them up is valuable for robustness and future proofing a critical part of the code base. She has been working on improving privacy in the P2P network. She has also worked on cryptographic code for DLEQ proofs in libsecp256k1, in relation to Silent Payments. She has summarised her work in more detail in the sections below.

Validation

The bulk of my focus this period was on validation, a security-critical subsystem in Bitcoin Core; bugs here can cause nodes to disagree on the state of the network.

I improved the logic in FindMostWorkChain (#34884) based on what the codebase guarantees now, and in the process discovered and fixed an undefined behavior bug in the same function (#35070). The bug is unlikely to trigger in practice but important to clean up - incorrect assumptions in consensus-critical code can cause future logic built on top of it to break in ways that are much harder to trace back. While working in this area I also helped investigate an assertion failure in the same area of code (#35050). I also got #32950 - removal of the BLOCK_FAILED_CHILD flag across the finish line in this period after addressing review comments which simplifies how Bitcoin tracks invalid blocks and fixes another rare undefined behavior bug.

On the review side, I covered invalid block handling, block validation tests, code cleanup (#34254, #35000, #35001), and reducing the interaction between validation and assumeutxo (#33477, #34786), among others.

P2P

P2P is the other area I spent time on. Changes in this area tend to have privacy implications and are where subtle, hard-to-spot bugs can happen.

A good example: #34146 changed the ordering of the initial address relay, which made m_getaddr_sent track an incorrect state - it could be set when we'd only seen a self-announcement and not an actual GETADDR response. The practical effect is minor (we could relay a small GETADDR response unnecessarily), but a variable that doesn't mean what it says is bad for codebase maintainability and can mislead future contributors. I reviewed the fix and while working through it, found the address relay tests weren't catching any of this and were just silently passing - so I opened a fix in #34750.

On the review side, I covered network-dependent timers for inbound scheduling to reduce fingerprinting risk (#33464), block downloads after assumeutxo validation (#33604), and address relay token abuse (#34774), among others. I also continued addressing review comments on #30951, my earlier PR to disallow v1 connections on IPv4/IPv6 peers. I also participated in the ASmap collaborative launches.

Wallet

Explored wallet code for the first time and picked up an issue where AmountWithFeeExceedsBalance - an error that tells users their transaction amount including fees exceeds their balance was never actually reachable (#34299).

While working on this, I also spotted and fixed a separate bug in coin selection - a switch from set<COutput> to set<shared_ptr<COutput>> had quietly broken the uniqueness guarantee, allowing two distinct pointers to the same output to coexist in the set. This could theoretically cause the same coin to be selected twice during coin selection, though it is not exploitable in practice.

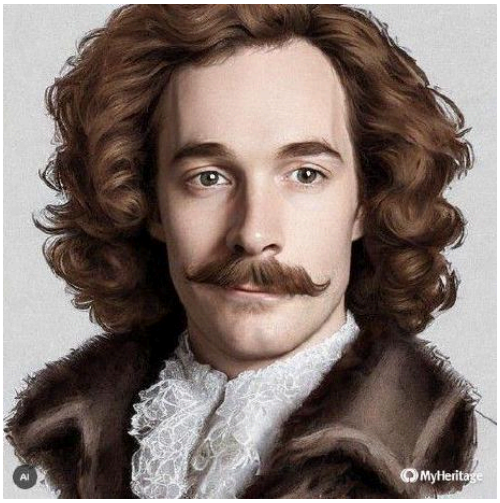
Cryptography (libsecp256k1)

DLEQ proofs let a party prove a relationship between two public keys without revealing the underlying secret. DLEQ proofs have several applications, one of which is verifying correct address generation in silent payments. I originally started this work in #1651, which had a broader scope also covering silent payments. Macgyver (who is also supported by Maelstrom) split out the DLEQ proof component into a focused PR #1802, on which we collaborated.

Community

Gave sessions on a variety of topics in my local developer community. Also spoke on two panels (one on open source and another on privacy) at Africa Bitcoin Conference.

Benalleng



- **Grantee since:** June 2025
- **Github:** https://github.com/benalleng
- **Grant Focus:** Payjoin
- **Specialist Area(s):** Quality Assurance

Overview

Ben's contributions to the Payjoin project have spanned test coverage expansion and novel test additions, workspace-wide refactors reducing external dependencies, dependency cleanups and alignment. Ben has also helped to overhaul the build infrastructure and hardened the primary Payjoin state machine and API refactors.

Payjoin Protocol

Improving Bitcoin's privacy characteristics is extremely important to Maelstrom. Two of our four grantees are wholly focused on privacy technologies, Ben Allen on Payjoin and Macgyver on Silent Payments. Before Payjoin existed, one could assume all inputs to a Bitcoin transaction were owned by the sender. Payjoin breaks that assumption, by creating transactions which include inputs provided by both the sender and the receiver. In order to achieve this, the sender and receiver must coordinate in a separate communications channel and therefore Payjoin adds significant complexity compared to the normal transaction process. However, Payjoin has a significant advantage, in that it materially degrades the ability of one analysing Payjoin transactions to determine the flow of funds. Payjoin also has an additional advantage in that the batching can result in more efficient coin control, resulting in savings on transaction fees.

The privacy advantages of Payjoin are much stronger that it would initially seem. Often, even when not using Payjoin, Bitcoin transactions have multiple inputs. Up until recently, chain surveillance analysts have been able to assume that all these inputs are provided by the same entity. If Payjoin adoption is sufficient, Payjoin breaks this surveillance heuristic and therefore Payjoin can improve the privacy of even users who have not adopted it. Bitcoin is about game theory and asymmetry. Achieving supermajority adoption of a privacy technology like Payjoin is likely to be extremely challenging and is perhaps not a realistic objective in the short to medium term. However, minority adoption of Payjoin can have a material positive impact on the systemwide privacy properties of the network. A significant impact on privacy can therefore realistically be achieved in the short to medium term. This is why Maelstrom considers Payjoin such an important technology and why Maelstrom is keen to support Ben in his endeavors in this area.

Over the past year Payjoin has hit substantial milestones continuing to improve as a protocol and demonstrate that it is a viable stepwise improvement for privacy in Bitcoin. Payjoin has demonstrated the viability of the API, integrating into both Bull Bitcoin and Cake Wallet, while also currently working on integrations in 5+ additional wallets, where it is expected support for payjoin will be added soon.

The Payjoin team has stabilized the API in preparation for a full 1.0 release that downstream wallets can rely on. In addition to the primary source code the team has released bindings for cross language compatibility with the Payjoin dev kit in 4 languages, Python, Javascript, Dart, and CSharp; with the expectation that this will cover a majority of developers interested in adding Payjoin into their software, whether on desktop, mobile, or even web based applications.

Ben also had a chance to participate in the Bitcoin Rust Summit in October 2025, where along with many other developers within the space they were able to share what they believe to be fundamental standards and best practices for not only the rust tooling, but also AI workflows and interoperability goals for the Bitcoin ecosystem. In the section below, Ben has written more details about his contributions in the last year.

Ben's Contributions

Fuzzing and mutation testing

I added Payjoin fuzzing infrastructure #1310, starting with a fuzzer for the BIP-21 interface used by Payjoin and later adding a fuzzer to complement our internal URL added to help reduce the code bloat and meet some goals of stakeholders hoping to integrate our software in the future. I also introduced mutation coverage #573 and expanded it to cover most of the primary state machine within the sdk as well as introducing diff based mutations to catch regressions where possible and encouraged others in the bitcoin ecosystem to follow suit as well.

Build, CI, and Nix infrastructure

I helped to expand the nixification of the Payjoin repo to ensure our developer environment was reproducible and consistent even across different machines, architectures and operating systems. Following discussion from the rust-bitcoin summit I bumped our MSRV to 1.85.0 to be inline with the rest of the bitcoin rust ecosystem and brought the entire workspace's dependencies in line with each other to optimize for the project's transition to a better manageable monorepo format. I made a large effort to expand and optimize our CI pipeline to ensure code added into the sdk was of the highest quality and easily reviewable by our developers ensuring a steady pace forward.

FFI Bindings and Integrations

I introduced the initial testing infrastructure for the BIP77 compatible tests in python #85 off of which the rest of the binding QA infrastructure has been based. As well I assisted in the initial scaffolding for the Dart bindings which have since grown substantially and the tooling is now used by several other bitcoin projects.

Looking Forward

In the coming year I hope to continue to harden Payjoin in the wild through stress tests and building QA systems in downstream implementations of the Payjoin sdk to ensure interoperability between wallets across the ecosystem and catch regressions when they happen. I also want to continue to expand the testing coverage and infrastructure within the Payjoin dev kit to achieve more complete coverage and expand the novel testing approaches like mutants and fuzzing. To improve the reliability of the sdk I want to Integrate Payjoin into and help streamline continuous fuzzers within the bitcoin ecosystem to get more reliable feedback from those systems to be leveraged by developers. Lastly I plan to polish and maintain a dev environment that is easy to use and quick to iterate on to help encourage future contributors to assist in moving Payjoin forward.

Macgyver

- **Grantee since:** June 2025
- **Github:** https://github.com/macgyver13
- **Grant Focus:** Silent Payments
- **Specialist Area(s):** Project management and development tracking

Silent Payments

In March of 2022 Ruben Somsen proposed "Silent Payments," a new approach to reusable payment codes, that removes the need for a notification transaction by entirely leveraging information that was already in the transaction, to signal to the recipient when funds are intended for them. Using Silent Payments, when requesting a payment, a recipient can create a payment code and supply it to the sender, who then has the ability to use this code to create multiple addresses, that only the recipient can spend from, while the recipient is offline. This capability reduces the incentive for address re-use, because a sender is now able to make multiple payments to a recipient, while the recipient is offline, critically without re-using addresses. Address re-use is a major factor in degrading user privacy.

Adoption Milestones

The Silent Payments ecosystem has matured considerably over the last 12 months, moving from early conceptual projects towards production-ready implementations. Adoption of Silent Payments typically occurs in two phases, first sending support, followed by the more complex receiving phase. Receiving introduces new challenges for wallet developers. They must design UI/UX for non-interactive payment codes and choose a scanning strategy with trade-offs between performance and privacy.

Wallet Adoption

Currently Blindbit-Desktop, Cake Wallet and Dana Wallet have completed both send and receive phases. Sparrow Wallet and Nunchuk have added send support in the last 6 months. Bitcoin Core has draft implementations for send and receive but they remain on hold due to a dependency on the Silent Payments module in libsecp256k1 (PR #1765, currently under review). For those interested in tracking Bitcoin Core Silent Payments development, the work is linked in PR #28536. Wallet adoption status can be tracked on silentpayments.xyz or SP Roadmap.

Scanning Performance

Scanning for Silent Payments outputs at scale required completely new tooling, since recipients must check every P2TR output on-chain to find their own. The first step was a specification proposal defining terminology

and characterizing the strategies available to wallets and indexing services. Two projects have produced scanning services that match the Tweak Server or Remote Scanner classifications. Blindbit-Oracle (Tweak Server) published a v2 release and is actively used by Blindbit-Desktop and Dana Wallet. Frigate, an Electrum-style Remote Scanner with a GPU pipeline, has significantly reduced scanning time from minutes to seconds for most use cases. Frigate also supports notifications for unconfirmed payments, matching the experience users expect from standard bitcoin wallets. Scanning backend server development can be tracked on silentpayments.xyz or SP Roadmap.

Specification Maturity

BIPs related to Silent Payments continued to mature over the past 12 months. BIP-352 received updates addressing edge cases and clarifications. BIP-375, critical for secure wallet and hardware signer coordination, received test vectors and a reference implementation for validating PSBTs. BIP-376 introduced new PSBT fields required for spending Silent Payments. BIP-392 defined the sp() output script descriptor format.

Foundational Libraries

BDK-SP and SPDK have continued to add features and lay the foundation for future wallet development. These libraries are designed to minimize the custom code required for integration, lower the barrier for wallet developers and accelerate adoption.

Macgyver's Achievements

- Formalized Silent Payments SP Roadmap and contributed to silentpayments.xyz.
- Created BIP-375 test vectors and practical reference for validating PSBTs for Silent Payments.
    - Created several proof of concepts bip375-examples to demonstrate coordinator / signer interactions via PSBT.
- Proposed PR #587 for Coldcard: first working BIP-375 hardware signer implementation of Silent Payments.
- Defined Indexing Server Specification.
- Defined SP Wallet Recommendations.
- Organized monthly meetups for Silent Payments working group, discussions from the meetings are archived here.

Macgyver's Plans for the next 12 months

- Produce a reference implementation for MuSig2 and Silent Payments.
- Propose BIP-375 workflow reference implementation.
- Assess emerging post-quantum signature schemes and their potential for interoperability with Silent Payments.
- Support wallet integrators and early adopters with technical coordination and testing.

About The Grant Program

- Grants are issued based on 12 month contracts.
- Payments are made monthly in Bitcoin.
- Grant stacking is permitted with a cap at US$400,000 per year.
- The Maelstrom grant review committee consists of the following:
    - Arthur Hayes (Maelstrom CIO)
    - Jonathan Bier (Grant Program Administrator)
- The program is wholly funded by Maelstrom, the family office of Arthur Hayes.

About Us

About Arthur Hayes

Arthur Hayes is the CIO of Maelstrom, a family office that invests across the crypto ecosystem. He is also the co-founder of BitMEX – the first crypto unicorn. Prior to entering the crypto industry, he worked as a trader in the capital markets divisions of Deutsche Bank and Citibank. Arthur holds a Bachelors of Economics from the Wharton School of Business. He has appeared on major business news networks including Bloomberg and CNBC. He is active on X (@cryptohayes) and releases a monthly newsletter (Crypto Trader Digest) read by thousands of investors globally.

About Maelstrom

Maelstrom is an investment fund focused on digital assets. It is managed by the family office of Arthur Hayes (co-founder, BitMEX). The fund's mandate is to build a portfolio of infrastructure companies that will serve as the foundation of the next wave of trustless decentralization.

LinkedIn | Naver | X | YouTube

pr@maelstrom.fund

Maelstrom