Cross-Input Signature Aggregation for Bitcoin

Research by: Fabian Jahr

To learn more visit cisaresearch.org

Human Rights Foundation

Table of Contents

Executive Summary

- CISA lets multiple Schnorr signatures from different inputs be combined into a single signature, significantly cutting transaction size and saving fees.

- By making multi-input transactions cheaper, CISA incentivizes and normalizes usage of collaborative privacy tools like CoinJoin and PayJoin instead of normal transactions, strengthening user anonymity while also improving network efficiency.

- Businesses can enjoy significant savings particularly for consolidation transactions which should counteract UTXO set growth and speed up adoption by exchanges and ecommerce.

- There is no singular CISA concept, there are different aggregation modes (full and half aggregation) and scopes (transaction-wide and block-wide) and their trade-offs need to be weighed for an upcoming proposal.

- CISA requires a soft fork and further cryptographic research is needed for a proposal that maximizes the benefits for the network.

01

Cross-Input Signature Aggregation (CISA) is a proposed enhancement to the Bitcoin protocol that would allow the aggregation of multiple Schnorr signatures from different inputs within a single transaction, or even across multiple transactions, into a single, smaller signature. This change would reduce the size of Bitcoin transactions, leading to potential cost savings in fees as well as efficiency gains in bandwidth and storage.

This report was written by Fabian Jahr as part of HRF's cisa fellowship. Learn more

Introduction to Cross-Input Signature Aggregation

Beyond that, CISA also could offer privacy improvements by incentivizing certain collaborative transaction types which typically contain a larger number of inputs, like CoinJoins and PayJoins, thus aligning with Bitcoin's broader goal of improving fungibility and financial privacy. This does not have to come at the expense of user experience, at worst users should have a similar experience as CoinJoins and PayJoins provide today which still includes a lot of potential for user experience improvements. CISA builds upon the foundation laid by the activation of Schnorr signatures in Bitcoin as part of the Taproot soft fork (BIP 340). Unlike ECDSA, the signature algorithm that Satoshi chose for Bitcoin at its inception, Schnorr signatures allow multiple signers to combine their individual signatures into one while still being verifiable, thanks to their additive and linear properties.

Since the Taproot soft fork, Bitcoin users can already perform a different form of aggregation within a single multisig policy, a concept referred to as key aggregation, using protocols such as MuSig2. However, these methods are limited to specific scenarios where the keyholders are sharing a single script or address. CISA, by contrast, allows separate inputs owned by different participants with different keys to be signed by a single signature.
If activated, CISA could incentivize a shift in Bitcoin users' behavior by making collaborative transactions like CoinJoins or PayJoins cheaper, thereby normalizing and expanding privacy-enhancing practices and therefore strengthening network privacy overall. For human rights advocates and other users with heightened privacy needs, cheaper privacy features and a larger anonymity set would be substantial benefits. Meanwhile, businesses that handle large volumes of transactions, like exchanges, could enjoy lower on-chain fees, especially for operations like UTXO consolidation. As CISA would see wider adoption businesses, as well as wallets implementations in general, should follow the trend and primarily transact with collaborative transactions. Enterprises entering into this realm for cost-saving reasons would give additional protection against legal persecution of users that utilize such transactions for privacy reasons.

However, implementing CISA would require a consensus change, a soft fork, so the Bitcoin community and developers would need to agree on adopting new validation rules for transactions and blocks with aggregated signatures.

This report provides an overview of the background of Schnorr signature aggregation and related concepts, a preview of the potential technical details of CISA including variants like half-aggregation vs. full-aggregation, the feasibility of adopting it in Bitcoin's consensus rules, and provides an outlook on potential next development steps. It also explores potential applications, from business perspectives to user experiences, and discusses how CISA might interact with other Bitcoin protocol proposals.

02

This section explores the historical context of Schnorr signatures and signature aggregation in Bitcoin and looks at a selection of other cryptocurrency projects that utilize signature aggregation as part of their protocol.

Background and Context

2.1 Prior to Bitcoin

Claus-Peter Schnorr, a German mathematician and cryptographer, introduced the Schnorr signature scheme in a landmark CRYPTO ’89 paper [schnorr-wiki]. In that work titled “Efficient Identification and Signatures for Smart Cards”, Schnorr first presented an interactive identification protocol that could prove knowledge of a secret without revealing it, and then showed how to transform it into a non-interactive digital signature using the Fiat–Shamir heuristic [schnorr-alinush]. The resulting signature scheme relies on the hardness of the discrete logarithm problem, making it one of the first signature designs whose security is rooted purely in discrete log assumptions rather than factoring or other problems [schnorr-wiki].

This was a significant departure from earlier digital signatures like RSA or ElGamal, offering a simpler and more efficient approach. Schnorr’s scheme produces short signatures and involves relatively straightforward arithmetic, in contrast to RSA’s large exponentiations or ElGamal’s two-part output [schnorr-cse]. These features were motivated by practical needs: the scheme was designed to be efficient enough for devices like smart cards and to improve upon existing signatures by reducing computational and storage overhead [schnorr-wiki].

Upon its introduction, the Schnorr signature was met with considerable academic interest. Cryptographers appreciated its elegance – it was “theoretically and intuitively the right thing to do, as opposed to the hideous (EC)DSA scheme” as one expert later remarked [schnorr-cse]. Early research established confidence in Schnorr’s security: Pointcheval and Stern [ps96], for example, proved in 1996 that Schnorr signatures are provably secure in the random oracle model, assuming the discrete logarithm problem is hard [schnorr-cse]. This gave Schnorr a solid scientific footing at a time when formal security proofs for signature schemes were still a novelty. Despite this cryptographic approval, adoption of Schnorr signatures in the real world lagged for many years. The primary barrier was patent liability: Claus Schnorr patented his signature algorithm in 1990, soon after its publication [schnorr-wiki]. The patent (U.S. Patent 4,995,082 [usp]) meant that developers and standards bodies hesitated to incorporate Schnorr’s scheme due to legal and licensing concerns. In fact, the U.S. National Institute of Standards and Technology (NIST) opted not to use Schnorr’s algorithm in its Digital Signature Standard, even though Schnorr’s design was known to be simple and strong. Instead, NIST introduced the DSA algorithm in 1991, which closely resembles Schnorr’s scheme but was purposefully engineered to avoid infringing Schnorr’s patent [hackmd]. Schnorr himself long contended that DSA still fell under his patent, though this claim was disputed [bitmex]. This situation created a paradox in the 1990s and early 2000s: Schnorr signatures were viewed as cryptographically sound, even superior in some ways, yet they remained absent from standards and mainstream software.

NIST's DSA and its elliptic curve variant ECDSA had enjoyed over a decade of deployment, whereas Schnorr signatures were largely absent from standards due to the patent. OpenSSL, a widely used cryptography library that underpins much of the internet's secure communication by handling the encryption and decryption behind applications, did not offer a plug-and-play Schnorr implementation, but it did support ECDSA, which further tilted the scales toward ECDSA for any new software project.

When Satoshi Nakamoto launched Bitcoin in 2009, Schnorr's patent had just expired in 2008 [bitmex], but the ecosystem of tools and libraries like OpenSSL hadn't caught up yet. Using ECDSA was the natural choice, since it was proven, free, and readily available even if Schnorr might have been the technically better option. As Bitcoin developer Pieter Wuille notes, by the time of Bitcoin's creation "it was already too late – it was already much more appealing to use a well-known standardized scheme over designing your own cryptography" [bse-ecdsa]. It wasn't until much later that interest in Schnorr signatures resurged the Bitcoin technical community, freed from legal barriers and motivated by new potential applications.

2.2 Schnorr Signature Aggregation discussions in the context of Bitcoin

Once the Schnorr patent expired and Bitcoin matured, developers realized that introducing Schnorr signatures could help with privacy, efficiency, and new constructs such as CISA. It would allow going beyond what was practically doable with ECDSA. But migrating from ECDSA to Schnorr required a soft fork and heavy review, culminating in the Taproot upgrade which activated in November 2021.

Taproot (BIP 340/341/342) brought Schnorr signatures into Bitcoin in a backward-compatible manner [ops-taproot]. Although the initial Taproot proposal did not include CISA, it lays the groundwork for such an enhancement in the future and the idea of CISA had its inception long prior to the activation of Taproot. In fact, most aspects of CISA described in this paper were already discussed on and off throughout the build-up to the Taproot soft fork.

Bitcoin Core developers discussed Schnorr signature aggregation in a May 2016 CoreDev meeting in Zurich, outlining several levels of aggregation. The "cross-transaction" signature aggregation idea was introduced as aggregating all input signatures of a transaction into a single signature. They further theorized about extending this to block-wide aggregation (potentially using BLS signatures) and even one-way aggregation of inputs/outputs for privacy, noting potential censorship-resistance benefits if transactions share a combined signature. [coredev16]

At a mid-2016 Bitcoin developers & miners meeting, Dan Boneh, a well respected Cryptography professor at Stanford, advocated moving to Schnorr signatures and then enabling transaction-level signature aggregation. He estimated this cross-input aggregation could save ~30% in block space if widely used. Boneh suggested this could be introduced via a soft fork as a stepping stone toward full block-level signature aggregation. [boneh16]
In October 2016, Pieter Wuille presented Schnorr signatures at Scaling Bitcoin in Milan and described a potential CISA feature for Bitcoin. He noted it would reduce block size by roughly 20%, a modest gain, but one that would crucially align incentives for CoinJoins by letting participants share one signature, thereby splitting the fee burden. Wuille explained that, with SegWit activated, implementing one signature per transaction would be straightforward and would also speed up verification by batch-validating signatures in one go. [milan16]

In May 2017, Tadge Dryja proposed on the bitcoin-dev mailing list a method for non-interactive Schnorr aggregation intended to be used on a per-block basis. His idea was that miners could sum the s-values of all Schnorr signatures in a block and commit only the aggregate s in the coinbase, effectively shrinking each individual signature to 32 bytes. This would save space and also make block verification faster, since only one final scalar multiplication is needed for the whole block. He acknowledged downsides in complicating mempool caching and fail-fast case of validation, but solicited feedback on the approach. [tadge]

By early 2018, attention turned to the security of CISA. At BPASE 2018, Pieter Wuille reviewed major hurdles preventing immediate deployment. He described the rogue-key attack in naive multi-signature schemes and gave an example (discovered by Russell O'Connor) where an attacker could duplicate another signer's public key and message across two inputs to fraudulently authorize spending the victim's coin. The known fix (each signer proving ownership of their key via extra signatures or commitments) would require adding those proofs on-chain, negating most efficiency gains. These unresolved issues meant that, at the time, cross-input aggregation could not safely be included in the Schnorr/Taproot upgrade. [bpase]

In Feb 2018, Greg Maxwell proposed Graftroot, an enhancement to Taproot, and noted it could leverage cross-input signature aggregation. A naive Graftroot spend would normally need an additional 64-byte signature, but Maxwell pointed out that the "non-interactive Schnorr aggregation trick" (now known as half-agg) can merge the s-values of all regular and Graftroot signatures in a transaction into one aggregate signature. This way, multiple Graftroot surrogate scripts could be used with only ~32 bytes overhead each. He cited this as a way to bind all signatures to the transaction and keep Graftroot's cost on par with Taproot. [graftroot]
Also in 2018, developers recognized that CISA would complicate future script upgrades. Anthony Towns wrote about the difficulty of combining aggregated signatures with new opcodes in a soft fork. He illustrated a scenario involving a hypothetical covenant opcode: if a second signature became conditionally required, an old node would mis-validate the aggregated signature set, breaking soft-fork assumptions. The suggested remedy was to segregate signatures into different "buckets" for separate conditions, but this added significant complexity.

This discussion underscored that doing cross-input aggregation and certain script changes together could cause issues, contributing to the decision to defer CISA until these issues were worked out. [aj] Shortly before Taproot's activation, developers clarified that Taproot would not include CISA. An August 2020 Reddit post on r/Bitcoin addressed this misconception, explaining that while Schnorr enables the possibility, CISA was deliberately left out of the Taproot proposal. The post also reiterated its benefit, how multiple inputs could share one signature, greatly reducing the size and fee cost of large CoinJoins. But it also emphasized that CISA introduces significant engineering challenges for future upgrades, and complicates how to extend Bitcoin in the future, so it was deferred. [reddit]

2.3 Schnorr Aggregation Case Study: MimbleWimble Protocol

While Bitcoin's adoption of Schnorr signatures is relatively recent, some other cryptocurrencies have been using Schnorr in a way that enables aggregated signatures from the start. One particularly notable protocol is MimbleWimble, which underlies the cryptocurrencies grin, beam, and the MWEB (MimbleWimble Extension Block) feature in Litecoin. MimbleWimble is often described as a privacy-centric protocol because it eliminates addresses, merges transactions, and employs confidential transactions and an important aspect of its design is that it uses Schnorr signatures in a combined manner to achieve a notion of privacy and block space efficiency.

In MimbleWimble, all inputs and outputs of a transaction collectively produce a single aggregated signature utilizing a MuSig scheme that proves ownership of the inputs without revealing amounts or addresses. Transaction data can be pruned because the protocol merges intermediate outputs, leaving behind only the final state, plus a set of combined signatures that validate the entire chain of ownership. This design leverages Schnorr's linear property to sum up partial signatures across all inputs. [mw1][mw2][mw3]

Let's take a closer look at how this works in grin, beam, and the MWEB on Litecoin:

01 grin: Grin was the first major implementation of the MimbleWimble protocol [grin]. Grin uses a transaction building process where the sender and receiver collaborate to combine partial signatures. Since each participant selects random blinding factors for their inputs and outputs, the transaction includes a final Schnorr signature that proves the sum of private keys (blinding factors) equals zero modulo the curve order. There is only one aggregated signature per transaction, and multiple transactions in a block can be merged (cut-through) if their outputs match. grin addresses complexities by requiring interactive transaction building and by carefully verifying that the sum of input commitments minus the sum of output commitments is zero, validated by the Schnorr aggregated signature. This interactive nature can be cumbersome for users in practice, but it is fundamental to the protocol's privacy and efficiency.

02 beam: Beam [beam] is another MimbleWimble-based cryptocurrency that similarly aggregates signatures within each transaction. Like grin, beam transactions rely on a multi-stage flow to produce a single Schnorr signature that covers all inputs. Beam differs from grin mainly in user experience and some additional layers of confidentiality and other features, but both share the same principle of aggregating partial signatures into one. Beam's team introduced their "Bulletproofs+" for range proofs and integrated them with the aggregated Schnorr signatures, aiming to reduce transaction verification overhead.

03 MWEB: Litecoin introduced a MimbleWimble Extension Block (MWEB) in 2022 [mweb]. The extension block coexists with Litecoin's main chain, allowing users to opt in to MimbleWimble-based transactions. Like grin and beam, it uses aggregated Schnorr signatures to prove ownership of inputs in a confidential manner. Again, each MWEB transaction that consolidates multiple inputs uses a single aggregated Schnorr signature.

Beyond grin, beam, and MWEB, there are occasional smaller projects that employ Schnorr-based aggregation or multi-signature schemes. However, the three main MimbleWimble implementations listed above remain the most prominent examples to our knowledge. They show that robust Schnorr-based aggregation can be part of the default transaction model if the entire blockchain architecture is designed around interactive signing. This offers valuable insights for Bitcoin: if one can impose a transaction flow that merges signatures early, space savings and improved privacy can become the norm. That said, the interactivity in MimbleWimble can be a barrier to adoption as well. Also, MimbleWimble's approach to combining all inputs in a single aggregated signature remains a specialized design. Bitcoin's approach to cross-input aggregation will have to differ in order to remain backward-compatible and less interactive. But wallet user experiences from these ecosystems can provide a glimpse of what a future Bitcoin wallet experience could look like, should CISA be activated. We will revisit the learnings from the MimbleWimble case study in a later section.

2.4 BLS Signature Aggregation Case Study: Chia

In addition to the cryptocurrencies that leverage Schnorr signatures for aggregation, some other high-profile projects use alternative schemes for signature aggregation or multi-signature constructs. A notable example of this is Chia. Chia employs different cryptographic assumptions and design tradeoffs than Bitcoin's secp256k1 + Schnorr approach, but may still offer key lessons on how aggregated signatures may be deployed at the protocol level and what implications this has for miners, users etc.

Chia [chia] differs from Bitcoin in that it relies on the BLS (Boneh-Lynn-Shacham) signature scheme, which uses pairings on curves like BLS12-381. BLS is known for near-seamless, non-interactive aggregation: any number of BLS signatures, each potentially over a different message, can be combined into one short signature. This is because verifying an aggregated BLS signature is done via pairing checks that confirm each public key actually signed its respective message. Chia's blockchain was built from the ground up with BLS, which allows a block producer to merge all transaction signatures of a block into one.

For the Chia protocol, the aggregated signature is 96 bytes regardless of how many separate signatures went in. The complexities revolve around ensuring that each public key is legitimate. Additionally, verifying a BLS aggregated signature can be more computationally expensive per signature than verifying a single Schnorr signature, but combining many signatures can still be more efficient overall. For Bitcoin developers, Chia demonstrates a fully realized example of cross-input, even cross-transaction, signature aggregation in a cryptocurrency production environment. The big difference is that BLS requires pairings, a different elliptic curve, and acceptance of distinct cryptographic assumptions. On the other hand, BLS is non-interactive. Bitcoin's path focuses on secp256k1-based Schnorr for continuity and conservatism.

Chia's success with BLS shows that data efficiency gains can be large if the entire chain design embraces aggregation. However, for Bitcoin, substituting BLS is unlikely due to the security model, widely deployed secp256k1 hardware, and a reluctance to add pairing-based assumptions. Instead, Bitcoin's research on aggregation for Schnorr signatures attempts to replicate some of the benefits of BLS within the existing curve and ecosystem, albeit with some interactivity trade-offs. We will revisit the learnings from the Chia case study in a later section.

03

In this section we will look into the different aspects of CISA that could play a role in a potential future soft-fork for Bitcoin.

CISA Deep Dive

3.1 Pre-face: Distinguishing Signature Aggregation from Key Aggregation in Bitcoin

The distinction between key aggregation and signature aggregation has been a constant source of confusion for people interested in building a better understanding of CISA. For this reason this section is presented first before diving deeper into the technical details, to pre-empt this from happening to the readers of this paper.

Key aggregation refers to combining multiple public keys into a single aggregated public key at address creation time, so that the group of signers can produce one joint Schnorr signature later. In a MuSig scheme, multiple signers interact to jointly sign a single message, resulting in one Schnorr signature that verifies against the aggregated public key. Verifiers don't need to know the individual keys even exist. This means the individual keys never appear on-chain improving privacy and efficiency for that input and the associated participants. The Taproot upgrade enabled such key aggregation within one input through the addition of Schnorr signatures, but it does not enable CISA. The consensus rules under Taproot remain oblivious to how a signature was produced; they only see a normal Schnorr signature and public key for each input since key aggregation is done pre-signing. [bse-keyagg] [blkstr-keyagg].

To illustrate this let's walk through how Alice and Bob might use key aggregation with MuSig to open a Lightning Network channel. In the setup phase, Alice and Bob each have their own public and private keys. Let's call Alice's keys (A_pub, A_priv) and Bob's keys (B_pub, B_priv). Using MuSig, they can combine their individual public keys into a single aggregated public key. This is done by performing some cryptographic operations on their public keys, resulting in an aggregated key we'll call AB_pub. In order to do this both their lightning nodes need to be online and interact with each other even though there is nothing put on-chain yet. When it comes to opening the channel, instead of creating a multisig address that requires both of their individual public keys (A_pub and B_pub), they use the aggregated key AB_pub. This makes it look like there's only one party involved when viewed from the outside.

To fund the channel, Alice and Bob will jointly create a transaction that sends funds to an address corresponding to AB_pub. Both need to agree on the transaction details since any spending from this address will require cooperation. When Alice and

Bob want to close the channel, they generate signatures using their respective private keys (A_priv and B_priv). These signatures are combined into a single signature using MuSig's signing process. The result is a single signature that corresponds to the aggregated key AB_pub.

Signature aggregation, on the other hand, occurs at signing time or post-signing. CISA specifically means taking multiple signatures, each corresponding to different inputs with different public keys and messages, and combining them into one. Verification of an aggregated signature would involve a new algorithm that takes a list of public keys and messages and a single signature, and validates that each key indeed signed its respective message. This requires additional consensus support, for example, a new opcode or validation rule that can observe multiple inputs together. Currently the Bitcoin protocol verifies signatures one input at a time. [bse-keyagg]

We will contrast the key aggregation example of Alice and Bob with a signature aggregation example now. Let's say that half-agg is available on the transaction level and that Alice and Bob are both participating in a CoinJoin transaction together with many other people. In this scenario, nothing during the creation of the CoinJoin transaction changes. It is also an interactive process but the signature aggregation does not need to be part of it at all. Each participant adds the signatures for their inputs individually as usual. Only when the CoinJoin transaction is complete the coordinator of the CoinJoin will aggregate the signatures of all participants into one without any involvement of Alice and Bob and then send the finalized transaction to the network. This is an example of post-signing aggregation. It is also notable that CISA itself does not directly improve privacy in terms of obfuscating participants since all the distinct public keys and inputs are still visible and observers can tell that multiple inputs were used. In fact, an aggregated signature even could potentially reveal that those inputs were likely signed together. By contrast, MuSig-style key aggregation hides the fact that there even were multiple signers for an input [blkstr-keyagg]. This is also the reason why key aggregation was a focus in Taproot while signature aggregation was not included, the benefits to privacy were more immediate with a clearer path to impact users directly.

3.2 Full Aggregation vs. Half Aggregation

Within the broader concept of Schnorr signature aggregation, there are two available approaches to aggregating signatures that are part of Bitcoin transactions: Full aggregation requiring interactivity and Half aggregation which is non-interactive. Both aim to compress multiple Schnorr signatures into a smaller bundle, but they differ significantly in coordination requirements and resulting signature size.

In Full aggregation (full-agg), all signers collaboratively produce one signature that they jointly signed together. This process should be analogous to an extended MuSig: multiple parties exchange nonces and combine their inputs to output a single 64-byte Schnorr signature no matter how many individual signatures were aggregated. This maximum compression results in the best possible space savings too. However, it requires an interactive protocol where all participants communicate during signing. Each signer must share cryptographic commitments and then combine partial signatures. This also means the participants must perform multiple rounds of communication to arrive at a valid aggregate signature. If all the signatures belong to one entity, this interactivity is trivial. The wallet can do it internally and most classes of issues in a signatures scheme don't apply to this scenario. But if the signatures come from multiple independent participants (e.g. in a CoinJoin or PayJoin), full-agg introduces significant complexity. Every participant must be online simultaneously and cooperate or at least have a way to communicate with the rest of the participants repeatedly; if anyone fails to follow the protocol or drops out, the combined signature cannot be completed. This is a key security and reliability concern for these protocols in general: a malicious or flaky participant could disrupt the signing by refusing to reveal their part or by providing invalid partial data, leaving the transaction unable to finalize. Robust multi-party signing protocols like MuSig2 or schemes like ROAST are needed to mitigate these risks by handling signers that abort or misbehave, but they add even more complexity. Such a signature scheme and protocol is yet to be developed for full-agg and this is a key piece that is still missing until full-agg can be considered as part of a soft-fork proposal.

Half aggregation (half-agg) on the other hand allows signatures to be combined after they are individually produced, without signers needing to interact. Each participant signs their message independently, creating a regular Schnorr signature for their input. Then an aggregator (which could be anyone, a wallet, a node, or a miner) takes the set of signatures and compresses them into a single signature whose size is roughly half the total size of the originals. To be specific, an N-signature half-aggregate is about 32*N + 32 bytes rather than 64*N [halfagg-bip]. This is achieved by concatenating the r values in addition to keeping one aggregate s value. The key property is that this process is a pure function of the signatures, pubkeys, and messages, no secret data or interaction is required beyond the original signatures themselves. Because of this, half aggregation can even be done by miners or intermediary nodes after the fact. The big advantage is no coordination needed from signers: participants just sign as normal, and aggregation is an optional optimization which can be applied later. This makes half-agg much simpler to deploy in multi-party scenarios. For instance, CoinJoin users wouldn't need to run an interactive signing session beyond what they already do to assemble a transaction; a coordinator or miner could aggregate their signatures afterward.

Additionally it should be noted that it is possible to use a combination of full-agg and half-agg to limit how many parties need to interact directly. For instance, a subset of inputs, perhaps controlled by one entity or a small group, could be fully aggregated among themselves, yielding one signature per group. Those resulting signatures could then be half-aggregated into a single signature. This hybrid means only the small groups go through interactive signing, and the final aggregation across those groups doesn't require interaction. So, one could reserve full-agg for scenarios where interactivity is manageable and use half-agg to merge outputs from different parties or groups afterward. Such a design can contain the interactivity to manageable domains and still realize much of the savings.

3.3 Transaction-wide Aggregation vs. Block-wide Aggregation

Another dimension to consider in CISAs design is the scope of the aggregation: should signatures be aggregated only within each transaction or across an entire block or both? These options have different security trade-offs and performance implications.

Transaction-wide aggregation involves aggregating all input signatures per transaction or potentially even a subset of the input signatures. If a transaction has, say, 5 inputs each requiring one signature, CISA would allow combining those into one signature that covers all 5 authorization checks. This is the more straightforward use of CISA and is often cited as the most realistic use case for deployment. That's because each transaction remains self-contained: it carries an aggregated signature that can be verified independently of other transactions. The benefits include reduced transaction size and thus lower fees for the spender. The cut is much more dramatic for something like a CoinJoin, for example aggregating 100 input signatures into one, making collaborative spends much more efficient. Security-wise, transaction-level aggregation is more limited in scope, any failure in aggregation affects only that transaction. If the aggregated signature is invalid or if the signers failed to produce one, only that transaction would be unrelayable or unmineable, and it doesn't jeopardize anything beyond that. This localized failure mode is easier to handle: wallets could fall back to including regular signatures if aggregation fails for some reason.

Transaction-wide aggregation appears to offer less maximum savings compared to block-wide. There is still at least one signature per transaction. If you have many small transactions, each will carry its own aggregated signature. The space saving is most pronounced when individual transactions have many inputs; it doesn't help much if most transactions have only 1 or 2 inputs. Still, even a modest reduction per transaction can add up. Also, transaction-level CISA can be implemented in a relatively contained way: for instance, a new witness version could indicate that a transaction uses an aggregated signature for all inputs, and include that single signature in a part of the transaction with each input committing to it. Verifiers would then check that one signature against all input pubkeys. This would change validation rules but keeps the scope within a transaction. Furthermore, full-agg can only be seriously considered for tx-wide and not block-wide, more on that later.

Block-wide aggregation on the other hand is a more extreme idea entailing the aggregation of all signatures across all inputs in all qualified transactions in a block into a single signature. In theory, this could mean only one signature in the entire block covering hundreds or thousands of inputs from multiple transactions [bse-keyagg]. The potential space savings appear to be higher here: close to all signature bytes could be eliminated except for one signature per block.

The security trade-offs and complexities of block-wide aggregation are greater as well, though. Firstly, block-wide CISA requires non-interactive aggregation since you obviously can't have every user of every transaction in a block coordinate, they don't even know which block their transaction will be in before the block has been mined. It also puts the burden on miners to perform the aggregation. Miners would collect transactions each with their own signatures initially, and then combine all those signatures into one before finalizing the block [bse-keyagg]. The incentives for miners are clear, by aggregating the signatures they free up space in the witness part of a block which might allow them to add more or different transactions into a block, which can earn them more fees.

This also means the block's validity would hinge on a single aggregated signature. If that signature failed to verify, the entire block is invalid. In practice, if a miner makes an aggregation mistake or includes a transaction with an invalid signature, the aggregate won't match and the block will be rejected. This isn't fundamentally different from today, one bad signature in a block makes it invalid today as well. But with a single signature, there is less granularity and you lose the ability to pinpoint which input was wrong without additional data.

3.4 Off-chain protocols/ P2P

It is worth exploring whether signature aggregation can be achieved outside of consensus changes, as part of off-chain protocols or clever use of the P2P network. Even if CISA as a soft fork takes time, there may be ways to aggregate signatures in such specific contexts to gain some benefits in the interim. This sort of out-of-band optimization is compressing data for transit and for the sake of simplicity we assume here that putting the signatures on chain is not possible. It turns out that half-aggregation is much better suited for this because it's non-interactive and anyone can do it on a given set of signatures.

There is research suggesting this use-case for Lightning Network gossip: instead of sending four separate signatures in a channel announcement, nodes could send one half-aggregated signature for the batch of announcements [halfagg-bip], cutting down gossip traffic. Since those signatures never hit the blockchain, the network can agree to use aggregated Schnorr signatures for efficiency without needing a Bitcoin consensus change. This improves bandwidth and is an example of a successful off-chain aggregation: Lightning nodes would need to upgrade to understand the new message format, but no miner or on-chain rule is involved.

In addition to the Lightning Network, a new layer 2 proposal has emerged very recently which natively utilizes half-agg: Shielded CSV is a proposal for a privacy-preserving, scalable protocol that sits on top of an existing blockchain and allows users to exchange and verify coins without putting all transaction details on-chain. Instead of broadcasting every transaction to all network nodes, only small nullifier data is posted to the blockchain, preventing double spends. Meanwhile, actual transaction proofs are shared off-chain on a peer-to-peer basis, which conserves block space and boosts privacy. Ordinarily, each account in Shielded CSV would have to place a separate Schnorr signature on the blockchain to nullify (invalidate) its old state and confirm a new transaction. If there are many inputs or participants, storing each signature fully on-chain becomes quite large and would undermine the compactness that Shielded CSV aims for.

Half-aggregation addresses this by merging many Schnorr signatures into a single compressed signature, which is only marginally bigger than a standard single signature. The final on-chain data, called an aggregate nullifier, contains:

- A set of Schnorr public keys each representing a user's account state being nullified.
- A single half-aggregated signature that collectively proves each included user really signed off.
- A short commitment identifying who posted this aggregated signature, so they can be paid a small fee in Shielded CSV.

Because half-aggregation shrinks multiple signatures into roughly half of the original signature's space, the chain data overhead is more manageable even when many users are updating their states at once. Conceptually, these signatures are Schnorr signatures with an added sign-to-contract trick, so each user commits to their specific new transaction data in the signature. A coordinator or publisher then gathers these partial signatures, runs a half-aggregate procedure to merge them into a single signature, and posts the result. This design ensures each user's state is recognized and valid, but the blockchain only sees a short aggregated signature. That final compressed form is what allows Shielded CSV to maintain its core promise of strong privacy and minimal on-chain data usage.
Shielded CSV is still a very new concept and so far there seems to be no implementation of the concept but it is a very interesting example of what future uses of signature aggregation may be expected from layer 2 concepts even without a soft fork.

04

Benefits

4.1 Space and Fee Savings

Fundamentally there appear to be two ways to look at the potential impact that the introduction of CISA could have in terms of space savings, which, on the Bitcoin blockchain, translate into fee savings. The first way is to use the past as a predictor of the future: What could we save if the way users behave stays consistent with the past? The second way would be to assume that the way users behave will not be consistent and that instead user behavior will change, maybe even dramatically, in particular triggered by the introduction of CISA.

You will be seeing differences in the savings in fees (sats) and space (bytes). This is due to the discount that SegWit introduced. SegWit (Segregated Witness) moved the data required to check transaction validity (the “witness”) out of the transactions they are part of and into a different (segregated) part of the block. Each input has a witness and it usually includes some script but, most importantly for us, also the necessary signatures. While SegWit itself primarily fixed transaction malleability (different witnesses can be valid for the same transaction) the witness discount sought to incentivize adoption of SegWit and make it cheaper to spend outputs, counteracting UTXO set bloat. The fee discount is 75% on witness data and since we are aggregating signatures, all the space that we are saving is in the witness section, leading to more mild fee savings in comparison.

Extrapolation Savings Model

The extrapolation model assumes there will be no fundamental changes to how the Bitcoin network is used after the introduction of CISA. This means we can use the average structure and size of transactions in the past and calculate how much users would save by using CISA. Jonas Nick has written a Python script that formed the base for this calculation [jonas-savings] but the content has been slightly modified and updated [jonas-script].

The script uses an average number for inputs and outputs of transactions. We are using an average number from last year (block range 833,000 to 886,000) calculated by a different script. Our script found that the number of average inputs for the last year were 2.12 and the average number of outputs were 2.64. Using these numbers such average transactions would allow an average saving of 6.9% of fees and 19.3% of space using half-agg. Using full-agg the fee savings would increase to 7.3% with space savings of 20.5%.

These numbers may not look all that impressive and particularly the lack of a meaningful difference between the half-agg and full-agg savings may disappoint some readers. However, keep in mind that these are historical average numbers which are quite small and the optimistic case for CISA adoption does include an expected change in network behavior, which the next model will take into account.

Disruptive Savings Model

The disruptive model assumes that the introduction of CISA will fundamentally change how most users interact with bitcoin and that this will also change what type of transactions we will see on-chain. In particular the model assumes we will see a lot of larger CoinJoin and PayJoin transactions because these will become a lot more economically feasible after the introduction of CISA. At the time of writing this paper, there are very little CoinJoin transactions observed which is primarily caused by the recent crack-down on the most popular CoinJoin implementations [rip-wasabi][rip-samourai], thus deducting future use of CoinJoins from the numbers seen in blocks today makes little sense. Instead we will look at what CoinJoin transactions looked like at the times they were popular and what might happen optimistically if they become even more popular than they were, triggered by the activation of CISA on Bitcoin. In particular we will look at the WabiSabi style of CoinJoins in detail, since these favored larger sets of inputs and outputs which means that they would profit the most from the usage of CISA.

Let’s first look at an individual participant in a CoinJoin that is similar to the historical average of WabiSabi style CoinJoins. On average, such CoinJoin transactions had an average of 76 inputs and 121 outputs [cjadopt]. We are further assuming that the 76 inputs correspond to 76 participants and that the feerate we need to pay is 10 sat/vbyte. Under these assumptions the users participating in the above lined out CoinJoin transaction would together need to pay 95,835 sats in transaction fees. The transaction would have the size (13,347 bytes). If this transaction instead uses CISA half-agg for its signatures, the fees to cover would instead be 89,653 sats (and 2,432 in bytes), a saving of 6,078 sats or 6.35% in fees (18.2% in bytes). The same transaction would cost 83,573 sats in fees when using CISA full-agg instead, resulting in savings of 12,157 sats (and 4,863 in bytes) or 12.7% in fees (36.4% in bytes). These are significant numbers considering the relatively mild effect of CISA on the network overall. A more bullish scenario would be that CoinJoin usage becomes more frequently used than was the case previously. WabiSabi style CoinJoins targeted a size of 100 inputs, a natural limit given that this is an interactive protocol. For this second case we will assume 150 outputs in the transaction as well as, again, 100 different participants with one input each and 10 sat/vbyte feerate. Currently such a transaction would have to pay 122105 sats in fees. Utilizing half-agg CISA the total fees for this transaction would be 114,002 sats, a saving of 7,999 sats or 6.6%. The same transaction would cost 106,002 sats in fees with full-agg CISA, relating savings of 15,998 sats or 13.1%.

As mentioned above, these large kinds of CoinJoins were used by WabiSabi-style CoinJoins. The similarly popular CoinJoin implementation Whirlpool by Samourai used a series of much smaller transactions, consisting of 5 inputs and 5 outputs. For comparison, those kinds of transactions would see 7.6% in fee savings and 20.2% in space savings under half-agg. Under full-agg it would be 15.2% in fees and 40.3% in space. These numbers may be surprisingly high compared to the much larger CoinJoins outlined above. The reason for this is that Whirlpool operates with an equal number of inputs and outputs while in the WabiSabi-style case we assume a larger number of outputs than inputs. Of course it seems possible that CoinJoins could become even larger given the incentives that CISA would enable, though, as mentioned above there is a certain limit given the interactivity constraints. It seems more likely that there will be multiple CoinJoins of the sizes mentioned above filling up a block in the most optimistic scenario for CISA adoption.

Consolidation Transaction

For many businesses dealing with lots of UTXOs, like ecommerce businesses or exchanges, CISA is highly attractive because it would allow for much cheaper consolidation transactions. Such transactions don't suffer from the same limitations as CoinJoins since interactivity is avoided plainly by the fact that a single entity controls all the UTXOs that are being aggregated. In theory a block could be fully taken up by a single aggregation transaction with thousands of inputs. However, in practice it seems that businesses keep consolidation transactions moderately large to participate in the fee market at a competitive feerate and gradually get their transactions in a series of upcoming blocks. One case of such a series of consolidation transactions by the crypto exchange OKX attracted a lot of attention because of a bug that led them to bid against themselves and thus may still be top of mind for some readers [okx]. The consolidation transactions in this series each used 150 inputs, so we will take this as the example to illustrate savings, assuming again a feerate of 10 sat/vbyte.

Such a transaction (150 inputs, 1 output) would cost 86,785 sats today. Using half-agg the fees would be 74,865 sats instead, a saving of 13.7%. Using full-agg the total fees would go down further to 62,945 sats, leading to a saving of 27.4%.

Another type of transaction that could profit from saving and that sometimes get named in conversations about CISA are batched payouts. However, batched payouts only profit from savings if they have multiple inputs which equals a consolidation transaction. This is why we are not showing numbers on such a case, it seems more likely that in practice a well managed exchange would consolidate UTXOs as much as possible in a low fee environment and then batch payouts would have very few, if not just one, input which would mean very low or no savings for these types of transactions.

Block-wide half-agg

To approximate how much savings we could get from introducing block-wide half-agg into bitcoin without assuming any further effect, we will look at the average number of signatures per block over the last year.

Throughout the past year from the time of this writing, the average number of signatures per block were 5,941 which took up an average of ~380 kbytes. Should all of these signatures in the future be aggregatable Schnorr signatures, this would allow total space savings within such an average block of ~190 kbytes. The saved space could be filled with more transactions, allowing for slightly higher fee revenues per block and slightly higher throughput of the Bitcoin blockchain in general.

4.2 Privacy

By merging multiple input signatures into one, CISA provides numerous advantages for Bitcoin's privacy-oriented use cases. It stands to significantly incentivize the usage of CoinJoin and PayJoin, particularly by making these privacy-enhancing techniques more economically attractive for users.
In the context of CoinJoin, where multiple participants combine their inputs to obscure the mapping between origin and destination of funds, the economic incentives are clear: Without CISA, each participant in a CoinJoin must include an individual signature for each input, resulting in a larger transaction size that consumes more block space and incurs higher fees. With tx-wide CISA, all the signatures can be aggregated into one, substantially decreasing the transaction's overall size. This reduction directly translates into lower transaction fees which benefits all participants of the transaction.

This cost-saving mechanism is particularly compelling for users of CoinJoin, who might otherwise be discouraged by the higher fees associated with using CoinJoins compared to normal transactions. By mitigating the fee burden, CISA lowers the barrier to entry for users considering CoinJoin, thereby potentially increasing its adoption. As more individuals participate in CoinJoin transactions due to these reduced costs, the anonymity set expands. A larger anonymity set enhances the privacy of all users involved, as it becomes increasingly difficult for external observers to trace specific transaction paths.

Similarly, PayJoin, which usually involves two parties collaborating to create a transaction that spends one input from each, also benefits from CISA. In a typical PayJoin transaction without CISA, each party would need to add their unique signature, inflating the transaction size and, consequently, the fees. With CISA, same as in the CoinJoin case, these signatures can be combined into a single one, decreasing the total transaction weight and the associated costs.

Beyond direct cost savings, CISA indirectly promotes privacy on the Bitcoin base layer by encouraging the normalization of CoinJoin and PayJoin usage broadly. Looking at the above savings calculations it becomes clear that using CoinJoin do not only become cheaper than CoinJoins without CISA, they also become significantly cheaper than sending a normal transaction by each individual user. As these methods would become both cheaper and more accessible, they are likely to see increased adoption not only among privacy-conscious users but also among those who are purely financially motivated. This broader adoption contributes to a network effect where privacy-enhancing practices become standard, making surveillance efforts less effective across the entire network. The more widespread the use of such techniques, the more challenging it becomes for adversaries to draw accurate conclusions from blockchain analysis. More usage and adoption could lead to better tooling and UX and so on, eventually creating a flywheel effect.

Moreover, the potential for CISA to weaken the common-input ownership heuristic, a method often used in blockchain analysis to infer the identities of transaction participants, further enhances privacy. By encouraging multi-user transactions through the economic incentives provided by CISA, it becomes increasingly difficult for analysts to distinguish between transactions involving multiple participants and those involving a single entity managing multiple UTXOs.

4.3 Computational Efficiency

Although CISA is often discussed in terms of fee or space optimization, it also offers notable gains in computational efficiency by trimming down the volume of discrete signature checks nodes must perform. Under normal circumstances, verifying an n-input transaction usually requires n separate Schnorr or ECDSA checks [core-blog]. Each check involves performing elliptic curve operations, which, despite optimizations, remain a significant portion of node validation time. CISA aggregates these n signatures into one before they are included in a block, meaning the node only needs to perform one signature verification when it sees the transaction for the first time. This consolidation can translate into tangible speedups when blocks contain many multi-input transactions, as the total signature checks across the entire block decrease substantially. It has to be noted that offloading the combination of multiple signatures onto an aggregation algorithm is only applicable to full-agg. In half-agg the aggregated signature is essentially decompressed into the original number of signatures which then have to be checked individually. Thus, there is no computational efficiency gain from half-agg usage.

Additionally, usage of Schnorr signatures alone already allows for batch validation, i.e. letting a node verify multiple signatures in parallel via shared elliptic curve operations [elem]. A node can queue up all the aggregated signatures from a block, perform a handful of group operations, and finalize the verification of each signature set in unison [halfagg-paper]. Work on this is already being done and could be used in Bitcoin without a soft-fork [batch-pr]. That does not mean that introducing full-agg CISA would not be completely without additional computational efficiency gains though. With full-agg the aggregation happens at signing time while with pure batch verification the aggregation of the signatures has to happen at validation time, which means there would be a small but additional efficiency gain from the usage of full-agg, even when batch validation has been rolled out.

4.4 Bandwidth

The usage of CISA in the P2P layer of the Lightning Network layed out in the previous chapter does allow space savings, however these are only relevant for bandwidth alone and do not lead to on-chain space or fee savings.

05

Despite the previously mentioned exciting benefits, the idea of CISA has struggled to amount significant attention and development support. The reason for this most likely lies in the challenges that CISA poses. These are what we will be looking at in detail in this section.

Challenges

5.1 Interactivity

A key challenge with full-agg cross-input signature aggregation is that it requires an interactive signing protocol among all parties. Unlike normal transactions where each input's signature can be produced more independently, a full-agg signature means all signers must collaborate to produce one joint signature. In practice, all signers need to be online and exchange data during the signing process. This adds significant complexity in the form of multiple rounds of communication, coordination protocols, and the potential for new failure scenarios if any participant drops out or misbehaves. Full-agg works seamlessly when one person controls all inputs, e.g. a single user spending their own UTXOs, since that user can generate the aggregated signature solo. However, for multi-party transactions like CoinJoins or PayJoins, coordinating every participant in real time is much more difficult. This interactive requirement is a major hurdle for implementation and user experience. To avoid any doubt, half-agg does not face this issue, it allows aggregation without signers interacting.

The primary strategy to deal with the interactivity is to develop a robust signature scheme similar to the role that MuSig2 and FROST play for key aggregation. Not having such a scheme in place before considering a soft-fork for full-agg CISA must be considered a deal breaker. Without such a scheme the heuristic that all the inputs that are aggregated are owned by the same entity becomes much stronger. This problem is laid out in great detail in the privacy section below. At the same time people may try to develop a scheme on their own and publish it without enough peer review. This could put user funds at significant risk.

Interactivity can be a stumbling block for any protocol [inter]. In an earlier section we laid out how the MimbleWimble protocol is built with an interactivity requirement from the ground up. To date, even the most popular implementations have struggled to see significant adoption and some early supporters of the protocol have partially blamed the interactivity requirement for this. Thankfully, the Bitcoin protocol will never rely on only CISA transactions, so users will always have the option to opt out of this issue.

5.2 Privacy

This report has mentioned before that CISA by itself is not a magic bullet for privacy. In fact, CISA provides no direct on-chain anonymity improvement, it only seeks to compress signatures for efficiency.

Any privacy gains would be indirect, for example, cheaper CoinJoins thanks to lower fees, which is outside the protocol change itself. Focusing on the potential downsides, we find that CISA could complicate privacy in several ways as well.

By reducing the number of signatures visible in a transaction, CISA might initially confuse certain heuristics used by blockchain analysts. However, analysts will quickly adapt by developing new heuristics for aggregated signatures. In fact, in particular the common-input heuristic might become even stronger in the beginning of full-agg adoption: because full-agg requires coordination, a transaction that successfully used an aggregated signature is likely to have been crafted by a single entity controlling all inputs. Chain surveillance companies already assume all inputs are one owner unless there are obvious CoinJoin patterns; with CISA, an aggregated signature could reinforce that assumption, as it suggests the inputs were signed in one go by cooperating keys. To prevent this it is paramount to have a robust signature scheme ready which allows collaborative transactions, such as PayJoin and CoinJoin, to adopt CISA quickly after deployment. Only when this is the case multi-user full-agg can spread quickly which can then prevent chain analysis from flagging full-agg multi-input transactions as definitely single-user.

Aggregated signatures might also introduce subtle new fingerprints that surveillance can exploit. Analysts could develop heuristics to detect that a transaction used CISA and label them accordingly. If CISA usage is rare at first, those transactions will stick out and if only few wallets support CISA those transactions can be attributed to a specific wallet software more easily.

CISA's interaction with other privacy techniques needs careful thought. Many advanced privacy protocols use scriptless scripts or adaptor signatures – for example, atomic swaps and coin swaps rely on extracting a secret from a signature. While this can also be a privacy issue, it is discussed separately in the next section.

5.3 Adaptor Signature Incompatibility

Cross-input signature aggregation is incompatible with adaptor signatures which are used for protocols like PTLCs in Lightning, atomic swaps via scriptless scripts, and Discreet Log Contracts [halfagg-bip]. In these protocols, an on-chain signature reveals a hidden secret when compared to an adaptor signature. If all inputs or an entire block share one aggregated signature, there is no way for that single signature to reveal multiple independent secrets to different parties [bse-blockwide]. In short, the individual, tweaked signatures never appear on-chain to allow secret extraction, so schemes relying on adaptor signatures break under CISA.

As with any privacy preserving feature it is currently hard to tell what the adoption of adaptor signatures is. To evaluate how hurtful the introduction of CISA would be to existing use-cases leveraging adaptor signatures, further research is needed. However, adoption of PTLCs in the Lightning Network seems certain in the mid to long term. This alone may not be a dramatic issue since non-collaborative closes are easy to identify in any case. However this is still not an ideal outlook.

5.4 Blockchain Reorg Challenges

Block-wide signature aggregation introduces challenges during chain re-orgs. If a block is reorganized, the aggregate signature that covered all transactions in the orphaned block is no longer valid with very high likelihood. Any difference in the transactions included in the block would invalidate it. Even if mostly the same transactions appear in a new block, the aggregated signature for the new block will be different. Thus, nodes must verify the new block's aggregate signature from scratch. There is no concept of carrying over a previously verified signature for an individual transaction or a previous aggregate that included the transaction, because only the combined signature existed on-chain [thoughts]. In contrast, with ordinary transactions today each input's signature can be verified independently. Bitcoin Core caches verification results, so a transaction that moves from an orphan block to a new block doesn't need to be re-verified in full if the node recognizes it. With a single block-level signature, such per-input caching is harder; "the whole half-aggregate signature S2 must be verified, including the contribution of X despite having it verified already […] This is in contrast to ordinary signatures, which do not have to be re-verified in a reorg." [thoughts]. In short, block-wide aggregation undercuts some benefits of cached or partial validation, potentially making block verification after reorgs less efficient.

Another potential issue is what happens to dropped transactions in a reorg. Suppose a transaction X was included in Block A with an aggregated signature, and then Block A gets orphaned and X is not included in the winning Block B. Normally, nodes would simply return X to the mempool or relay it again, since they still have the fully signed transaction. All signatures were in Block A's data after all. But with block-wide aggregation, X's individual signature never appeared on-chain – it was absorbed into Block A's combined signature. Once Block A is orphaned, nodes cannot extract X's original signature from it. The aggregate signature can't be split apart to recover individual s values. If the node did not previously have X or didn't cache its witness, X effectively disappears from that node's view because it's now an invalid transaction. Even if the node had verified X as part of the block, it only saw the aggregate, not X's own s. Thus, X can't be automatically put back into the mempool or rebroadcast by nodes on the new chain [thoughts]. This poses a security and reliability risk: users might assume X is confirmed and go offline, only for a reorg to drop it. Without special handling, the funds from X's inputs would be spendable again, but the transaction itself might be lost until the user or some form of watchtower re-transmits it. In essence, block-wide aggregation breaks the normal guarantee that a valid transaction, once seen, can be mined again if a reorg occurs.

Developers have proposed mitigations to address the above challenges. One idea is for nodes to store the original signature data for each transaction even after it's included in a block [thoughts]. Instead of just keeping the block's aggregate signature, a node would retain each transaction's individual signature, perhaps until the block is buried deep enough that reorgs are unlikely. In the event of a reorg, this allows two things: (1) if the transaction still appears in the new block, the node can subtract the cached signature from the new block's aggregate and avoid re-verifying that part, and (2) if the transaction is dropped, the node still has a fully-signed copy of X which it can return to the mempool or relay. This cache strategy essentially sacrifices some memory/storage by keeping signatures that aren't in the chain in order to preserve the same resilience Bitcoin has today. Another mitigation is to require wallets/miners to handle reorgs at the wallet layer, meaning the transaction's sender or some third-party would rebroadcast X if it falls out of the block. However, relying on users to manually rebroadcast is a very weak solution. It has potential privacy downsides and it assumes users are online and monitoring the chain [thoughts].

An earlier section of this paper described how the Chia Network uses a block-wide signature aggregation with BLS signatures. This might have come as an exciting case study for how Chia deals with these issues, however, after researching the topic it appears that Chia has not addressed these issues on a protocol level so far. Instead it's on the client to re-submit their transaction if they want to ensure their transaction gets mined. It should still be interesting to watch further development since Chia develops much faster than Bitcoin and might still present solutions that Bitcoin can learn from in the future.

5.5 Slow Adoption

History has shown that new Bitcoin features at the scale of CISA see gradual uptake. We can expect CISA to follow a similar trajectory as previous soft forks like SegWit and Taproot. With SegWit in 2017, even though it offered fee savings and other benefits, it did not immediately dominate the network. It took on the order of years for adoption to reach a majority of transactions. In fact, about two years after SegWit's activation, only roughly 50% of transactions were using SegWit, and it took four years to hit around 80% adoption. This slow roll-out was largely due to wallets and services taking time to upgrade and users being free to opt-in at their own pace [chainalysis-taproot].

Taproot's adoption, after activating in late 2021, has likewise been gradual; even with broad support among node operators, actual usage in transactions grew slowly as wallets added support. CISA's adoption will likely be incremental as well. Support for a new SegWit v2 output has to be implemented in wallets, hardware devices, exchanges, block explorers, etc. This requires extensive developer time and testing, which will not happen overnight. Many wallets might not prioritize CISA until they see clear demand for it, especially since the immediate benefit might seem minor to the average user not participating in complex transactions yet, a chicken and egg problem if you will. Some software might never be updated if it's no longer well maintained, meaning certain users would be left using old formats unless they switch wallets. All these factors mean that even after the consensus change activates, it could be a long time before CISA is ubiquitous and reaches its full potential.

Let's compare CISA to SegWit and Taproot. SegWit had the advantage of fixing a specific pain point, transaction malleability and block capacity, and it still took years to become dominant. Taproot's benefits, privacy in multisig and script flexibility, were less immediately relevant to most users, so its adoption has been slower and very gradual. CISA's benefits are primarily scalability/fee and secondarily enabling cheaper CoinJoins. This means CISA might follow a path somewhere in between, potentially faster than Taproot if fee pressure increases, but possibly slower if users are apathetic. One optimistic scenario is that exchanges adopt CISA outputs quickly because of the potential savings in consolidations, and generate a lot of CISA transactions, demonstrating the feature and boosting visibility. A pessimistic scenario is that only a small set of power-users use it, and most everyday wallets ignore it, resulting in a long tail of low uptake. Either way, the transition period will have to be managed. Coordination bodies like Bitcoin Optech [optech] may again help by tracking CISA adoption metrics and nudging services to implement it, much as they did with Taproot adoption.

06 Feasibility of CISA options

This section makes an attempt at laying out the feasibility of the combinations of different CISA options, arriving at an indication of which combination could be viable for a soft fork proposal in the future.

6.1 With Consensus Change

First of all, a quick recap of the options we have:

- Half-Agg: A non-interactive scheme where anyone, even a third party, can combine multiple Schnorr signatures into a single aggregate signature about half the size of the originals. Pros: It requires no coordination between signers at signing time – participants can sign as usual and an aggregator later compresses the signatures. This simplicity is also reflected in a comparatively low implementation complexity. Cons: The aggregate signature is larger than a single signature, so savings are modest relative to full aggregation.
- Full-agg: An interactive protocol where multiple signers cooperate to produce one signature equal in size to a single Schnorr signature (64 bytes) no matter how many inputs/keys are involved. Pros: It provides the maximum space savings – e.g., 10 inputs could be satisfied by one 64-byte signature instead of ten, greatly improving scalability. Fewer signatures also mean fewer verification operations, potentially improving validation speed for full nodes. Cons: Full aggregation requires complex coordination. Signers must exchange data in multiple rounds to create the joint signature, and they must securely manage nonces/state between rounds. This adds protocol complexity and opportunities for error. In multi-party contexts this would usually mean that all participants would need to be online and interact during signing, which is a practical hurdle.

It can be concluded that half-agg is much more likely to be introduced to Bitcoin than full-agg at this point. However, it should be noted that the concerns with full-agg are mostly in the nature of research and implementation work, and are believed to be solvable problems.

- Transaction-wide aggregation: This approach aggregates all input signatures per transaction. Instead of each input carrying its own Schnorr signature, one aggregate signature in the transaction's witness would cover all of that transaction's inputs. This could be achieved via full-agg or via half-agg. The benefit is a size and fee reduction for multi-input transactions, which is especially significant for cases like consolidation transactions or CoinJoins with many inputs. Reducing the number of signatures from, say, 10 to 1 can noticeably cut the transaction's size. Transaction-wide CISA alone could make CoinJoins and multi-input spends cheaper per participant than separate transactions.

- Block-wide aggregation: A more aggressive approach would allow combining signatures across different transactions, theoretically resulting in one signature for the entire block in the ideal case where all signatures are elligible and merged. This yields maximum space savings and minimum total signatures to verify. But this comes with various potential issues, particularly reorg efficiency etc.

Looking at the above trade-offs the first choice seems clear: tx-wide half-agg is the most likely combination due to it having few open questions left and limited complexity. On the other end the combination of block-wide full-agg can simply be considered impossible at the current state of research into this topic. All participants in the transactions would need to collaborate in order to make this happen while at the same time it's not even clear which transactions the miners want to include in the next block. Hence this would need a complete remodelling of how transaction propagation, mempool and mining work, which is simply not going to happen.

Maybe the most interesting question of this section is whether block-wide half-agg is more attractive than tx-wide full-agg or not. There is no indication of community consensus on this topic but the impression of the author is that tx-wide full-agg complexity on the protocol level appears to be a more manageable problem as well as there appears to be a lower risk of unforeseen side-effects. In addition people interested in CISA typically express the desire to get the full savings potential if a soft-fork should be seriously considered. The fact that more research is needed to make full-agg feasible for a soft-fork would probably trigger the community response that the soft fork would need to wait for that research in that case.

Last but not least the interplay between the different options should also be considered. As described in the savings benefits section, the CISA feature is bound to the optimism of changing how people interact on-chain all-together, meaning that we would see a lot more large collaborative transactions. Assuming that this will become a reality and we would have tx-wide full-agg as an option, we might have a much lower number of signature data in blocks under these circumstances. This would then mean that an additional block-wide half-agg would not have nearly as big of an impact as it might have promised in a pre-CISA environment.

6.2 Without Consensus Change

Even without changing Bitcoin's base-layer rules, some benefits of signature aggregation can be achieved in off-chain protocols and cooperative arrangements. While CISA isn't possible on-chain today, creative techniques in layer 2 and P2P settings can mimic its effects or improve efficiency in related ways:

The Lightning Network's messaging layer can also leverage signature aggregation. LN nodes announce channels to the network with a channel announcement message, which contains four signatures. Each node signs the message under both its node key and Bitcoin key, for authenticity. These announcements aren't on-chain, but they are bandwidth-heavy since they should propagate through the whole network. Developers have noted that since these signatures are produced only by the channel partners, they can use aggregation techniques to reduce them [thoughts]. For example, the two node signatures can be merged into one, and potentially the Bitcoin-key signatures could be aggregated as well. In fact, because creating a channel announcement already requires coordination between the two nodes, they could fully aggregate the required signatures into a single signature in an interactive protocol [thoughts]. By doing so, the channel announcement would be cutting its size significantly.

This is an example of using CISA in a P2P context to save bandwidth and improve efficiency without any blockchain rule change. It's worth noting that while this is technologically possible, the Lightning developers have a long list of priorities (like routing reliability, liquidity management, security etc.), so optimizing gossip with signature aggregation may not be considered very urgent. Still, it's a clear win made possible by Schnorr: off-chain messages can be aggregated just like transactions could. We may see future LN protocol updates adopting these ideas to reduce traffic.

07

A post CISA Bitcoin network would provide lots of exciting opportunities for all of its users. This section lays out the most promising of these ideas.

Applications

7.1 CISA adoption from a business perspective

CISA promises to reduce transaction fees for Bitcoin businesses by cutting down on signature data. From this perspective, different industry players stand to benefit in distinct ways which align with their primary incentive: to generate a profit.

Exchanges

High-volume exchanges process thousands of deposits and withdrawals daily, often resulting in many UTXOs that need consolidation eventually in order to prevent incurring high cost from spending them in a high fee environment. CISA would make UTXO consolidation far more economical. Today, an exchange might defer merging low value UTXOs because the fee to spend each UTXO separately outweighs its value. With cross-input aggregation, even UTXOs worth only a few hundred sats could be aggregated cost-effectively [trezor]. An exchange could combine, say, 100 small inputs into one output and pay for only one signature instead of 100, significantly shrinking the consolidation transaction's size. An example for the savings calculation has been laid out above in the benefits section. This not only saves fees for these businesses but also yields a cleaner UTXO set with fewer outputs benefiting the whole network. In effect, CISA slightly raises the feasible threshold for dust consolidation, lowering the minimum economically spendable UTXO size. Exchanges that adopted SegWit and batching earlier have already seen fee benefits from this [segwit-batching], and CISA would have a similar effect. The result is improved operational efficiency and lower long-term costs. It's conceivable that exchanges would encourage customers to use compatible wallets for deposits/withdrawals once CISA is available, to maximize these savings, much like many exchanges rolled out SegWit bech32 addresses to reduce withdrawal fees for themselves and their users.

Ecommerce/Payment processors

For ecommerce businesses as well as payment processors that handle payments for ecommerce businesses the situation is very similar to the one of exchanges. Ecommerce companies receive a new UTXO for every single purchase that is paid for with an on-chain bitcoin transaction. Frequent consolidations are necessary. But in addition, it may be interesting for more technically savvy ecommerce businesses to explore adoption of PayJoin as soon as CISA is deployed.

Wallet Providers

Wallet software, especially custodial or enterprise wallets managing many UTXOs, could leverage CISA to optimize transactions. For instance, a wallet app that needs to spend from multiple inputs would consume less block space with CISA. Even individual users would indirectly save money when spending coins from multiple past receipts, a common scenario when a payment exceeds the value of any single UTXO they own [trezor]. By adopting CISA, wallet providers can offer customers lower fees, which is a competitive advantage. Some wallets already offer CoinJoin features; with CISA, they could consolidate UTXOs or combine user payments with much smaller weight overhead, making such features more attractive to use regularly. This should increase the adoption of CoinJoin and PayJoin further, particularly in wallets that push the user experience for these features to be more accessible by the masses, which will help privacy of the whole network, as laid out previously in the privacy benefits section.

Miners

Note that bitcoin mining is an existing business model that would be heavily influenced by the introduction of CISA. However, the actions and incentives of miners also heavily influence the network as a whole, which is why there is a separate section specifically on this topic below which talks about miner incentives and their network level effects.

Ecash Mints

Ecash mints are essentially custodial wallets with much better privacy guarantees. This means that the same benefits apply to them as outlined in previous sections, e.g. exchanges and custodial wallets.

Potential New Business Models

Beyond improving current operations, CISA opens the door to entirely new business models and services in the Bitcoin ecosystem. By enabling multiple independent parties to collaborate within a single transaction with one signature using full-agg, CISA creates opportunities for businesses that facilitate such cooperation. Here we outline some potential new models and how they might work:

One obvious new service would be a CISA-powered fee-saving coordinator. This service would allow users who want to make a Bitcoin transaction to join an interactive pool instead of sending their own transaction immediately. The coordinator collects multiple users' pending transactions and assembles them into one larger transaction that uses CISA. Because only one signature is used for all inputs, the total fee is much lower than if each user sent separate transactions. Each participant could then pay a fraction of the fee they would have paid alone. Payment of the fractional fees could be settled over the Lightning Network. Such a model is an extension of the concept of batching, but generalized across users who don't necessarily trust each other. Essentially, the service would operate similarly to today's CoinJoin implementations but marketed for fee savings rather than primarily for privacy. The value proposition is clear: users get cheaper transactions, and the coordinator earns a small fee. This could appeal to retail wallet users, merchants scheduling payouts, or even exchanges partnering to combine withdrawals. A business offering this would focus on providing a smooth interface, perhaps wallets could have an "economy send" option that uses such a pool. If structured non-custodially, the service might avoid heavy regulation, acting more like a transaction messenger than a funds custodian. In effect, these fee pools or batching brokers could become a new niche in the Bitcoin economy, especially during high-fee periods when demand for savings is highest.

CISA could also supercharge existing CoinJoin-based business models although the landscape looks pretty bleak at the moment. It seems more likely that it will inspire new market participants to appear. Today, CoinJoin rounds typically impose a fee overhead – participants pay for the additional inputs/outputs needed to mix coins, which is often acceptable for privacy but still a deterrent for some. With cross-input aggregation, a CoinJoin with many inputs would be much more fee-efficient, potentially even more cost-effective than a normal standalone spend [trezor]. One analysis suggests CoinJoin transactions could become slightly more fee-efficient than an ordinary spend with CISA [trezor]. This flips the script: rather than privacy costing extra, users might improve privacy and save money at the same time. A new business model could be CoinJoin-as-a-Service with minimal or zero fee penalty, where a provider runs massive, regular CISA-enabled CoinJoin pools that anyone can join to get both anonymity and low fees. As privacy becomes essentially subsidized by fee savings, the anonymity set of such pools could grow dramatically, increasing the overall effectiveness of CoinJoin.

For context, one of the largest current CoinJoin pools, Samourai's Whirlpool, held about 4,350 BTC in its liquidity pool at a given time [trezor]; with lower costs, we could imagine even larger pools and more frequent mixes. Businesses that already offer mixing services might reduce their fees or even find new revenue by taking a share of the fee savings instead of charging users directly. On the other side, exchanges and wallet providers could integrate privacy pools without burdening users with extra costs. This could normalize privacy-preserving transactions. Interactive transaction pools in this context serve both privacy and fee efficiency; businesses facilitating them, either for profit or as a value-add feature, would likely emerge. We might also see collaborations – e.g. an exchange partnering with a CoinJoin provider so that many customers' withdrawals get combined in one large, private, low-fee transaction. This particular idea would need careful compliance navigation, but technically it could provide users with one-click "privacy withdrawals" that actually cost the exchange less in fees than sending outputs individually. Overall, CISA could make privacy-centric business models more financially viable and attractive to a broader user base.

Note that, while it seems possible that new business models similar to those lined out emerge, it seems most likely that the above will be added as features by existing wallet providers or might be used as differentiators by newly emerging wallet providers.

7.2 CISA adoption from a user perspective

To realize the benefits of full-agg CISA, users must coordinate their transactions with others in real time. Full signature aggregation is interactive and requires participants to exchange data during the signing process. This coordination can be technically challenging to implement in wallets and may inconvenience users if not automated seamlessly. Ensuring a smooth user experience for finding partners, handling communication, and dealing with failures (e.g., a participant dropping out) is non-trivial and will be critical for mainstream adoption.

When multiple independent users cooperate in a single transaction, new trust considerations emerge. If using a central batching service or coordinator, users must trust that service not to leak sensitive data linking inputs to outputs or to abort the process maliciously. Even in decentralized coordination, each participant must trust that others will sign as agreed; one dishonest or offline participant can stall the entire transaction. Although no funds can be stolen unilaterally, the pooling model could introduce denial-of-service risks where malicious actors repeatedly disrupt CoinJoin attempts. Users and businesses may also worry about inadvertently transacting alongside someone with "tainted" coins, potentially raising compliance concerns. Robust protocols will be needed to mitigate these risks and build trust in aggregated transactions.

CISA's effectiveness grows with wide adoption. In the early phases, only some wallets and UTXOs will support the new output type [bse-output]. This means users might find few peers to aggregate with, limiting opportunities to save. If only a minority uses CISA, the cost savings per user may be more modest and not outweigh the coordination effort which may include longer wait times until enough participants have appeared. Thus, if adoption is sparse, the incentive might not radically change behavior, and transactions that do use CISA could still be in the minority, making them stand out. Only when a critical mass of users adopt the new method will network effects kick in to realize the full potential, lower overall fees, more liquidity in CoinJoin-like batching, and improved fungibility. Until then, users face a bootstrapping problem where the benefits remain partly unrealized.

Widespread CISA adoption could strengthen the social and legal standing of privacy-enhancing techniques like CoinJoin. By making multi-user transactions economically attractive, CISA provides a financial justification for practices that were previously seen as purely privacy-motivated. Users and companies can plausibly say they aggregated transactions to save on fees, not just to obfuscate payments. In practice, CISA would make CoinJoin rounds cheaper than ordinary transactions [trezor], fulfilling earlier predictions that even modest fee savings could lead to more users opting for CoinJoins [ops-cisa]. This fee incentive could increase CoinJoin usage dramatically, expanding the overall anonymity set and normalizing the behavior.

If batched, aggregated transactions become common for cost-saving reasons, it blurs the line between regular transfers and privacy-enhanced transfers. This normalization protects users: choosing to CoinJoin will be seen as a standard economic decision, reducing the stigma or suspicion associated with it. Industry analyses suggest that a ubiquitous use of CoinJoins could make chain surveillance in its current form nearly impossible [trezor]. In a scenario where many or most transactions are aggregated, attempts to ban or discourage CoinJoin usage would face practical and political resistance, since the practice would be indistinguishable from ordinary fee-saving measures. CISA may bolster Bitcoin's fungibility by embedding privacy into an efficiency improvement, thereby safeguarding the user's choice to enhance privacy under the pretext of prudent cost management. Finally, this should hopefully lead to improved protection against persecution of CoinJoin users by law enforcement which is currently heavily biased to suspect illegal activity from these users.

7.3 Miner incentives and Network level effects

In Bitcoin, transaction inputs consume previously existing outputs, and create new outputs that become part of the UTXO set. If a transaction has more inputs than outputs, the UTXO set shrinks, if outputs exceed inputs, the UTXO set grows. CISA changes the economics of using multiple inputs in a single transaction by greatly reducing the signature overhead per input. This can influence how readily users consolidate UTXOs or participate in multi-input transactions, thereby affecting the balance of output destruction vs creation.

Under the current fee regime, there is a gap between the cost of creating outputs and the cost of spending them. It's historically been cheap to create many small outputs, but expensive to spend them. This has led to dust UTXOs that sit unspent because the fee to spend them would exceed their value. By reducing the signature cost for spending outputs, CISA further lowers the barrier to include low-value UTXOs in transactions. Even tiny outputs can be aggregated with others at minimal overhead cost. This means users are more likely to clean up dust by spending it batched with other inputs when fees allow, rather than leaving it indefinitely. Improved ability to prune dust contributes to a leaner UTXO set over time. CISA thus tilts incentives toward output destruction consuming existing UTXOs by making spends with multiple inputs more fee-efficient, helping control UTXO set bloat. Users face less penalty for consolidating or using many inputs, so the network could see slower growth in total UTXOs or even net reductions during consolidation phases compared to a scenario without CISA. However, it should be noted here again that SegWit dampens the savings effect because it has already put rules in place that have the same goal. SegWit introduced the concept of witness data being discounted in weight/fee calculations. In today's protocol, each witness byte counts as only 0.25 bytes toward the block size limit equaling a 75% discount [bse-segwit-cheap]. This discount was designed to reduce the cost of spending outputs relative to creating them, aiming to mitigate the dust problem by making input-heavy transactions cheaper. Unless there are further changes that change the SegWit discount, CISA will have to work within this framework and has several implications for fee structures and incentives. Since aggregated signatures reside in the witness section, the fee savings are real but not as drastic as the actual space savings [ops-cisa]. From the miners' point of view, the SegWit discount and CISA change how weight is distributed, but miners still want to fill blocks up to the same 4M weight unit limit and earn as much fees as possible doing it. A CISA transaction has lower weight than the equivalent non-aggregated transaction, so miners can include a bit more transaction data in each block for the same weight budget as long as the 1MB limit without witness data is not reached. If user demand for block space stays constant, this effectively increases throughput slightly, which could put mild downward pressure on fee rates in the long run since supply of block space is marginally higher. However, any space savings might be offset or even exceeded by new usage (Jevons paradox). In either case, the impact on miner fee revenue is likely small.

Additionally, since space usage is reduced particularly in the witness section of a block, there could be additional incentive for miners to include large ordinal transactions in a block. For example, if a large number of consolidation transactions fill up a block the limit of total bytes of non-witness data (1MB) would be hit first. Compared to the 1MB in non-witness data, the witness section of this block would be rather small. It could now be possible that the miner offers a relatively cheap fee for an ordinal transaction that is close to 3MB in size, since ordinals utilize witness data to store data on the blockchain for a variety of use-cases. This effect would likely be controversial as ordinals themselves are controversial and the SegWit discount has largely been blamed for enabling their development and popularity.

An additional interesting side effect of block-wide signature aggregation using half-agg would be that users have some additional complexity to work with to estimate the fees they might need to pay. The size of their transaction at the time they send it to propagate through the network may not be the same size as the transaction will have when included in a block. This means that fee conscious users may try to further undershoot a fee estimate they would receive based on today's logic. But primarily this means the fee estimation algorithms for wallets that make CISA enabled transactions would probably need to change to account for this development in order to continue to be as accurate as possible.

08

In this final section we take a look at the current status of specification and implementation of CISA and give an outlook on the necessary steps to get to a full proposal. Aside from the steps that have already been taken and published all of this is subject to change if unforeseen issues are encountered.

Implementation status and outlook

8.1 Consensus Rule Changes

This first section discusses the consensus changes that would need to be detailed in the BIPs and implemented in the code, in order not to repeat this in each of these sections below.

Deploying cross-input signature aggregation isn't as simple as many of the new op code proposals that are discussed in public today – it necessitates introducing a new SegWit version via a soft fork. Bitcoin's current rules do not allow one signature to cover multiple inputs even if Schnorr signatures are in use; each input is verified independently against its own signature. CISA would change that by allowing a single signature to satisfy the validation of all inputs or possibly a subset of inputs in a transaction. To do this safely, developers will likely propose adding a new SegWit version that defines different validation logic for the witness. Essentially, this new version would be almost a copy of the Taproot rules except that it permits to have only one signature for the supported inputs which would need to be accompanied by some script changes. By introducing a fresh SegWit version, older nodes and software that don't understand CISA will simply see an unknown witness version – they won't attempt to validate those signatures and will treat those outputs as anyone-can-spend or unrecognized, which is how SegWit ensures forwards-compatibility. This approach isolates the new behavior to a new output type, avoiding any ambiguity or retroactive change to existing transaction types.

As mentioned above, the new SegWit version would mostly be a copy of Taproot. However, some very important changes to Bitcoin Script would be necessary. Firstly, there would need to be new signature checking op codes which would correspond to the existing signature checking op codes but would allow for signature aggregation, think of OP_CHECKAGGSIG and OP_CHECKAGGSIGADD. Secondly, there is an incompatibility issue between these aggregating signature checking op codes and the OP_SUCCESS upgrade mechanism that was added together with Taproot. In short: since the signature checking op code defers the check of signatures and and OP_SUCCESS op code may be redefined or not, there are scenarios where consensus between nodes that have upgraded and know about the new rules for OP_SUCCESS and nodes that have not upgraded, can fail if a script contains both. Since OP_SUCCESS always makes a script valid, its inclusion in a script may hide a potentially invalid aggregated signature aggregation that a node that has not upgraded will not be able to see.

Several ideas to resolve this have been described by developer AJ Towns already in the mailing list thread where he made the public aware of this issue [cisa-success] but no consensus has formed around the question which solution is the best and should be used in a potential proposal.

CISA would most naturally apply to Taproot key-path spends, but not necessarily to script-path spends. There does not seem to be agreement yet whether supporting script-path aggregation is a feature worth supporting and there is a alternative idea named Generalized Taproot that may be an elegant solution that resolves both the need for CISA support in script-path spends as well as resolves the needs for solution to deal with the OP_SUCCESS issues described in the previous paragraph [thoughts]. Generalized Taproot, however, would add an additional layer of complexity to a CISA proposal [groot-cisa].

8.2 BIPs

Given the complexity of CISA there will be a series of BIPs required to describe the behavior of a potential soft-fork, similar to how Taproot consisted of 3 BIPs (340-342). The split proposed here is just a rough outline of what can be expected and the final result may look differently after the proposal has gone through wider community feedback.

The BIP that describes the half aggregation scheme is in draft mode and has only a few outstanding issues that need to be resolved in order to add it to the BIPs repository. It makes a lot of sense to have this BIP stand alone since half-agg can also be relevant for off-chain use-cases, as laid out in this paper previously. [halfagg-bip]

Analogous to the half-agg BIP there will need to be a full-agg BIP which will primarily have to describe the signature scheme, similar to the MuSig2 BIP. Laying the groundwork of research for this BIP and developing a signature scheme that is satisfactory for the community appears to be the biggest blocker for a CISA proposal that utilizes full-agg. A section further below talks about this in greater detail.

Finally there would need to be at least one further BIP (or possibly a series of BIPs) that specify the details of the new transaction version, SegWit version 2. The new transaction version would differ in the way the Witness program is structured, permitting empty signatures to be possible in the context of an aggregated signature that will be checked later. This would be accompanied by script changes that propose new signature check op codes for these aggregated signature checks and it would include a way to resolve the incompatibility issues with OP_SUCCESS op codes.

There would probably also be a need for new PSBT BIPs that amends their behavior to be able to deal with transactions with aggregated signatures or to-be aggregated signatures.

There are currently no resources allocated to working on a block-wide proposal, so it is unclear how the BIPs would be structured in such a proposal.

8.3 Code

For half-agg there are several implementations available, most notably the feature has been implemented in secp256k1-zkp [halfagg-zkp]. Based on the code of that implementation, a pull request to secp256k1 is open as well [halfagg-secp]. Additionally there exist implementations in hacspec (pending a rewrite to hax) and Python.

Since further code has not been written, there can only be speculation on where the complexities will lie when this is tackled. Bitcoin Core would need new code that allows validating an aggregate signature against multiple (pubkey, message) pairs. This could be done, for instance, by a special opcode that accumulates a running aggregate over inputs and then checks the final signature in the last input or in the transaction-level data. One good indicator that surfaces at least part of the complexity exists with the work on batch validation in Bitcoin Core [batch-core]. The aggregated signature needs to be maintained as a state that persists across checks and is only verified after all signatures have been collected in the aggregated signature. This requires some changes to the check queueing logic that are not fully done at the time of this writing.

In the case of tx-wide CISA, there would be considerable changes to wallet software while block-wide half-agg would primarily add engineering effort for node software and miners.

8.4 Cryptographic Research

As previously noted, the forming of a full-agg signature requires an interactive process. Interactivity adds tremendous amounts of complexity to any protocol. For example, all signers need to be online at signing time. Existing protocols that want to integrate full-agg signature aggregation will need to implement and handle this complexity in the future, including newly introduced failure scenarios, privacy implications etc.

To this date no scheme specifically for full-agg has been developed. But such a scheme will use similar ideas as MuSig and Bellare-Neven although.

The following properties are desirable when it comes to a full-agg scheme according to cryptographer Jonas Nick:

- Provably secure
- Allow duplicate public keys
- Does not require proofs-of-possession
- Works with schemes like Taproot Tweaking and MuSig
- Two rounds like MuSig
- Batch verifiable

To be clear, there is currently no ongoing research that tackles these goals specifically. In comparison, half-agg is already much further developed. The only indication that has been given by researchers is that Bellare-Neven [bn] might be a good starting point.

8.5 Deployment

CISA is explicitly envisioned as a soft fork upgrade [thoughts]. By using a new SegWit output version, it ensures backward compatibility so older nodes won't validate those spends but also won't reject them. This is already discussed in more detail in the consensus rule changes above. Additionally, it is noteworthy that a new output type does not mean that the address format has to change as well. It seems likely that addresses supporting CISA will be using Bech32m just like Taproot.

Soft fork upgrades in Bitcoin typically require miner signaling and broad agreement in the ecosystem. A discussion of activation mechanisms is outside of the scope of this paper.

8.6 Interactions and combinations with other Softfork proposals

There are currently no known blockers that would prevent CISA from being deployed together with any other soft fork proposal that is currently in serious consideration. A CSFS implementation would opt out of aggregation or there would be an alternative op code necessary that takes advantage of aggregation. Such a proposal does not seem to exist yet.

A combined deployment with another proposal seems somewhat unlikely, since CISA alone packs quite a bit of complexity. But it would be possible to pair it with any of the more simple proposals like a single new op code (controversial as they may seem today) or the Great Consensus Cleanup for example.

A notable exception are changes that target Quantum Resistance, though these proposals are still in very early stages of discussion and are thus also considered out of scope of this paper.

References

[cjadopt] https://arxiv.org/pdf/2109.10229

[schnorr-alinush] https://alinush.github.io/schnorr-signatures

[schnorr-wiki] https://en.wikipedia.org/wiki/Schnorr_signature

[schnorr-cse] https://crypto.stackexchange.com/questions/48528/security-of-schnorr-signature-versus-dsa-and-dlp

[bitmex] https://blog.bitmex.com/the-schnorr-signature-taproot-soft-fork-proposal/

[bse-ecdsa] https://bitcoin.stackexchange.com/questions/73049/why-was-ecdsa-chosen-over-schnorr-signatures-in-the-inital-design

[coredev16] https://bitcoincore.org/logs/2016-05-zurich-meeting-notes.html

[boneh16] https://diyhpl.us/wiki/transcripts/2016-july-bitcoin-developers-miners-meeting/dan-boneh/

[milan16] https://diyhpl.us/wiki/transcripts/scalingbitcoin/milan/schnorr-signatures/

[tadge] https://gnusha.org/pi/bitcoindev/CAKEeUhh3Rj3Dh8ab5FFR6dGKc2O-jm5Z0uyWtAtrPrh=7dvj-GA@mail.gmail.com/

[bpase] https://diyhpl.us/wiki/transcripts/blockchain-protocol-analysis-security-engineering/2018/schnorr-signatures-for-bitcoin-challenges-opportunities/

[graftroot] https://gnusha.org/pi/bitcoindev/CAAS2fgSnfd++94+40vnSRx-Qfi9fk8N6+2-DbjVpssHxFvYveFQ@mail.gmail.com/

[aj] https://gnusha.org/pi/bitcoin-dev/20180321040618.GA4494@erisian.com.au/

[reddit] https://www.reddit.com/r/Bitcoin/comments/ibcnsv/taproot_coinjoins_and_crossinput_signature/

[hackmd] https://hackmd.io/@PIncentivus/r1xkfbGO3

[mweb] litecoin.gitbook.io/mweb-docs

[grin] grin.mw

[beam] beam.mw

[chia] chia.net

[bse-keyagg] https://bitcoin.stackexchange.com/questions/106163/what-is-the-difference-between-key-aggregation-and-signature-aggregation

[blkstr-keyagg] https://blog.blockstream.com/en-musig-key-aggregation-schnorr-signatures/

[halfagg-bip] https://github.com/BlockstreamResearch/cross-input-aggregation/blob/master/half-aggregation.mediawiki

[jonas-savings] https://github.com/BlockstreamResearch/cross-input-aggregation/blob/master/savings.org

[jonas-script] https://github.com/fjahr/hrf-cisa-scripts/blob/main/jonas-savings.py

[okx] https://x.com/mononautical/status/1799077584795582551

[core-blog] bitcoincore.org/en/2017/03/23/schnorr-signature-aggregation

[elem] elementsproject.org/features/schnorr-signatures

[halfagg-paper] https://eprint.iacr.org/2021/350.pdf

[bse-blockwide] https://bitcoin.stackexchange.com/questions/107196/why-does-blockwide-signature-aggregation-prevent-adaptor-signatures

[thoughts] https://github.com/BlockstreamResearch/cross-input-aggregation

[chainalysis-taproot] https://www.chainalysis.com/blog/bitcoin-taproot-upgrade/

[trezor] https://blog.trezor.io/taproot-v2-how-will-the-latest-bitcoin-upgrade-evolve-in-the-future-e8559d0c5886

[segwit-batching] https://bitcoinops.org/en/veriphi-segwit-batching/

[ops-cisa] https://bitcoinops.org/en/topics/cross-input-signature-aggregation/

[bse-output] https://bitcoin.stackexchange.com/questions/106240/will-cross-input-signature-aggregation-need-a-new-output-type

[bse-segwit-cheap] https://bitcoin.stackexchange.com/questions/108333/why-are-segwit-transactions-cheaper-than-legacy-transactions

[halfagg-zkp] https://github.com/BlockstreamResearch/secp256k1-zkp/pull/261

[halfagg-secp] https://github.com/bitcoin-core/secp256k1/pull/1566

[batch-core] https://github.com/bitcoin/bitcoin/pull/29491

[bn] https://cseweb.ucsd.edu/~mihir/papers/multisignatures.pdf

[ops-taproot] bitcoinops.org/en/topics/taproot

[ps96] https://www.di.ens.fr/david.pointcheval/Documents/Papers/1996_eurocrypt.pdf

[usp] https://patents.google.com/patent/US4995082A/en

[rip-wasabi] https://thedefiant.io/news/regulation/wasabi-wallet-to-eliminate-coinjoin-amid-u-s-regulatory-fears

[rip-samourai] https://www.justice.gov/usao-sdny/pr/founders-and-ceo-cryptocurrency-mixing-service-arrested-and-charged-money-laundering

[batch-pr] https://github.com/bitcoin/bitcoin/pull/29491

[inter] https://www.youtube.com/watch?v=uI15RKnyX_E&t=1s

[optech] https://bitcoinops.org/

[mw1] https://docs.beam.mw/Mimblewimble.pdf

[mw2] https://eprint.iacr.org/2020/1064.pdf

[mw3] https://eprint.iacr.org/2020/061.pdf

[groot-cisa] https://gnusha.org/pi/bitcoindev/CAPg+sBhAE_WtZvbe7dEmWPMCsxwK4GmD-Kx2721X45Ua9KvhfEg@mail.gmail.com/

[cisa-success] https://gnusha.org/pi/bitcoin-dev/20180321040618.GA4494@erisian.com.au/