MEV-COMMIT WHITEPAPER

Primev Team

Version 1.0
November 25, 2024

Abstract

Decentralized blockchains have revolutionized digital trust and coordination, enabling applications that are transparent, censorship-resistant, and secure. Despite their benefits, these systems face limitations due to transaction delays, uncertain inclusion order, and the risk of reorganizations (reorgs), which can disrupt transaction execution and user experience. These challenges are particularly significant in real-time or price-sensitive applications, where delayed or uncertain transaction finality can lead to inefficiencies and financial risks. To address these issues, we introduce *insured commitments*, a protocol framework where providers guarantee certain conditions, compensating users if these conditions are not met. A special case thereof are preconfirmations, where providers ensure transaction inclusion—possibly with execution guarantees—before they are recorded on-chain. We present the mev-commit protocol that implements insured commitments.

We further demonstrate how our implementation of insured commitments can mitigate reorg risks and enable new applications, such as instant cross-chain bridges, ultimately providing a more robust infrastructure for decentralized finance and other blockchain-based ecosystems.

1 Introduction

Permissionless and decentralized blockchains have transformed our understanding of trust and coordination in the digital landscape. They enable a new generation of decentralized applications that are inherently censorship-resistant, transparent, and secure. Users interact with these applications by submitting transactions to the blockchain network. These transactions are processed by blockchain nodes and eventually included in a block, which is appended to the blockchain. However, the user typically gets no guarantee that a transaction will be included in the chain until it is included in a block. To achieve consensus in a highly decentralized setting, many blockchain systems require a substantial delay between blocks. This interval is to allow sufficient time for the network to propagate the proposed block, helping prevent frequent forks of the chain. In addition to the delay, blockchains are also susceptible to reorganizations (reorgs), where a different branch of the chain is adopted. Reorgs can occur due to network delays, network partitions, or malicious behavior by block proposers. Reorgs present risks such as double-spending attacks, where a user can spend the same funds on two branches of the chain. To mitigate these risks, many blockchains have some finality rule that ensures that finalized transactions can never be removed from the chain via a reorg. Ethereum, which is one of the most popular blockchains and processes around 1 million transactions daily [1], has a time between blocks of at least 12 seconds, and it typically reaches finality in roughly 12.8 minutes (every 64 slots).

This delay in reaching block inclusion, let alone finality, can hinder applications that require fast, responsive transaction confirmations, limiting the technology's adoption in real-time or price-sensitive settings. Moreover, state changes during this delay period can lead to execution difficulties, creating poor user experiences and even potential fund losses. These inefficiencies

become further compounded when malicious actors try to exploit them, and in cross-chain transactions, where different blockchains operate asynchronously without shared access to each other's state. Together, these factors underscore the need for a solution that addresses blockchain limitations, providing a more efficient and responsive transaction experience.

Toward this, we introduce the concept of *insured commitments*: a flexible protocol enabling a set of bidders to offer a payment to providers fulfilling specified conditions. If a provider issues an insured commitment but fails to fulfill it, they have to compensate the bidder, creating accountability and trust. A specific type of insured commitment is a *preconfirmation*, where a provider guarantees the inclusion or execution of a transaction even before it is included in a block. This framework offers a pathway to enhance transaction reliability and efficiency, addressing core limitations in current blockchain systems and fostering broader adoption across decentralized applications.

1.1 Insured Commitments

We here give some high-level overview of insured commitments; more details are given in Section 2.

Insured commitments definition. An insured commitment protocol allows bidders to send bids to providers. A bid contains a bid amount, a compensation amount, an activation condition, and a fulfillment condition. The fulfillment condition is what the bidder wants to be fulfilled. The activation condition allows to specify a condition such that commitments are only relevant if this condition is met. A provider can then issue a commitment to a bid, which is a guarantee that if the activation condition is met, the provider will either fulfill the fulfillment condition or pay the compensation amount to the bidder. Bid and compensation amounts can be functions of the time the commitment is issued, e.g., enabling the bidder to pay less if commitments are issued later.

Preconfirmations. Preconfirmations are a special case of insured commitments where the fulfillment condition is the inclusion of a transaction in a future block plus potentially additional requirements such as executing the transaction without reversion. The commitment providers for preconfirmations can be different entities, such as block proposers, block builders, or sequencers for rollups. The activation condition in this case specifies that a commitment is only relevant if the block is built by the provider. Moreover, the time dependence of the bid amount can be used to incentivize providers to issue commitments fast. This allows the involved actors to issue and obtain preconfirmations in real-time. In particular, a bid can specify that the commitment must be issued before the block is supposed to be built, i.e., to ensure that the preconfirmation in fact comes before the confirmation.

Privacy. If the information in bids and commitments were public, it could allow competing bidders and providers to observe and adjust their strategies based on the visible bids. This visibility potentially harms the expected utility of bidders, e.g., as their competitors could frontrun them or outbid them by adjusting their bids accordingly. Moreover, the premature revelation of a provider's commitment to a bid could lead to information leakage about the total value of the block being constructed. This not only affects the privacy of the providers but also skews the competitive landscape, enabling providers to underbid based on the leaked value information, and thus also potentially reducing the value the block and harming the revenue of the validators. To prevent these issues, we consider three important privacy properties of insured commitments: bid privacy (preventing information leakage about bid contents to anybody except the desired providers), commitment privacy (keeping commitment contents confidential until

settlement), and recipient anonymity (hiding to which providers bidders send their bids). More details about these properties are provided in Section 2.3.

1.2 Actors in Insured Commitment Protocols on Ethereum

Insured commitments are a general concept with a wide range of applications and are applicable to many blockchain systems and even beyond blockchains. In this whitepaper, we mostly focus on the Ethereum ecosystem. We here give a brief overview of the involved actors.

Ethereum's PBS. To address proposer centralization and prevent transaction censorship in proof-of-stake Ethereum, the concept of proposer-builder separation (PBS) [2] was introduced. It aims to separate the block proposer role into two entities: (1) Block builders, sophisticated actors that construct blocks to maximize profits, and (2) proposers, network-selected entities responsible for proposing blocks. Instead of constructing blocks, proposers can now choose blocks from builders, who offer fees to have their blocks proposed. These fees, which redistribute mev profits, allow even less sophisticated proposers to earn competitive profits, helping to counter proposer centralization driven by mev. The only current implementation of PBS is Flashbots' mev-boost [3], that operates on Ethereum's mainnet and is currently responsible for building around 90% of every new Ethereum block.

Who are bidders and providers? A bidder can be any user interested in having a commitment for a bid, such as searchers or solvers. Providers are entities that can credibly issue commitments. Block proposers have the highest credibility since they are ultimately responsible for the proposed blocks. Proposers can also appoint another entity to issue commitments on their behalf, e.g., through an auction.

Block builders can also issue commitments for the blocks they are building. Since they are sophisticated actors specialized in block building, builders are best suited for economically deciding which transactions to include and how to price their commitments. However, a commitment from a single block builder is less credible than one from a proposer since a builder can only promise to include transactions in the blocks they build, but blocks from other builders can make it to the L1 chain. The activation condition for commitments from block builders can be that the block at height h in L1 must is built by that provider. Such commitments are thus *conditional* in the sense that the only become active if that builders builds the block. They are still valuable for the bidder since they allow them to gauge the probability of activation. This probability increases when commitments from multiple block builders are sourced, approaching 100% if commitments are obtained from all block builders. These two models have been compared by Matt and Akdeniz in an Ethereum Research article [4].

Furthermore, one can consider solvers acting as commitments providers. They could source commitments in a separate insured commitment protocol from multiple block builders, gauge the remaining risk, and then issue *unconditional* commitments to bidders. The solvers would thus absorb the remaining risk and price it into the commitments accordingly. This allows to construct an unconditional insured commitment protocol from a conditional one and allows less sophisticated bidders to benefits from insured commitments without worrying about underlying risks. This discussed in more detail in Section 4.

1.3 Applications of Insured Commitments

Insured commitments are a powerful tool and widely applicable, especially in the context of blockchains. We here discuss some implications and possible applications.

Reorg risk mitigation. Perhaps surprisingly, we demonstrate that insured commitments can serve as a tool to mitigate the risk of a transaction "disappearing" from the blockchain

or not being executed as intended after a reorganization. This holds particularly for the case where insured commitments come in the form of preconfirmations provided by block builders. The reason behind this is that when a block builder issues an insured commitment (i.e., a preconfirmation) for including a transaction bundle in a block at height h, this block builder is responsible to do so on any branch of the blockchain. In particular, if preconfirmations are obtained from all block builders, the bidder has the guarantee that whichever of them builds the block at height h, either the transaction bundle is included in that block or the bidder is compensated, regardless of which branch "survives" a possible reorganization. A more detailed analysis of this approach is presented in Section 3.

Instant bridges. Preconfirmations can be used to transform a cross-chain bridge into an "instant" bridge. In a typical bridge, the user sends assets to a smart contract on the source chain, and the bridge operator sends the equivalent amount of assets to the user on the target chain. However, the user has to wait for the transaction to be included in a block on the source chain to have only minimal reorg risk (or even be finalized) before the bridge operator can send the assets on the target chain. By using preconfirmations, the bridge operator can send a bid to a set of providers, offering to pay a certain amount if the transaction is included in a block. The providers can then issue preconfirmations to these bids, ensuring that the transaction is either included in a block or the bridge operator is compensated. This allows the bridge operator to send the assets on the target chain immediately after receiving the preconfirmation, creating an instant bridge experience for users. We discuss this applications in more detail in Section 7.1.

Preconfirmations for searchers. Another application of preconfirmations is to let searchers reduce the risk of not having their transactions included (or executed) fast enough in a block. For example, consider an arbitrageur that wants to execute a time-sensitive trade on a decentralized exchange (DEX). The arbitrageur can send a bid to a set of providers, offering to pay a certain amount if the trade is executed. The providers can then issue preconfirmations to these bids, ensuring that the trade is either executed or the arbitrageur is compensated. Furthermore, searchers can optimize the price for the transaction by issuing multiple bids with increasing bid amounts until one is preconfirmed, instead of setting a high priority fee upfront.

1.4 The mev-commit Protocol

The mev-commit protocol[1] is a decentralized insured commitment protocol with a peer-to-peer network and no central authority controlling the network. The protocol requires a fast blockchain, called mev-commit chain, to support the inclusion of commitments in near real-time so they can be publicly settled later. The protocol is also designed to be private, with bidders and providers able to send and receive bids and commitments without revealing their contents to other parties.

Overview. The protocol is run between a set of bidders and a set of providers, where each bidder can send bids to a subset of the providers. While the protocol is general enough to support a wide range of providers, the default is that block builders act as commitment providers. The fulfillment conditions in the bids can vary from simple transaction inclusion to complex execution strategies. The default for mev-commit is to bid for execution of a transaction, i.e., the transaction gets included in the block and does not revert in case of a smart contract call. Providers can accept these bids by committing to them, where a cryptographic commitment is stored on the mev-commit chain. The protocol is described in more detail in Section 5.

[1]The name alludes to the fact that mev-commit is a commitment protocol that can be used for transactions related to maximal extractable value (mev) and works within the current PBS pipeline.

Bid decay to incentivize timely commitments. To incentivize providers to issue timely commitments early, the bid amount in mev-commit decreases linearly over time until the commitment is issued. Bidders can specify an expiry time in their bids, after which the bid amount is zero. Timestamps on the mev-commit chain are used to determine the commitment times. This mechanism solves what has been referred to as the "preconf fair exchange problem" [5, 6].

Privacy of bids and commitments. As discussed above, maintaining privacy of bids and commitments is crucial. The mev-commit protocol leverages a cryptographic solution to ensure end-to-end privacy throughout the entire protocol execution. The privacy of the bids and commitments is achieved through the use of two novel cryptographic primitives: anonymous receiver-updatable broadcast encryption (ARUBE) and dual-phase commitments (DPCOM). These primitives are described in detail in Section 6.

1.5 Organization of the Whitepaper

In Section 2 we formally introduce the notion of insured commitments and define its required properties; we also show that preconfirmations are a special case of insured commitments. In Section 3 we discuss how insured commitments can be used to mitigate the risk of transactions disappearing from the blockchain after a reorganization. Section 4 shows how to construct unconditional insured commitments from conditional ones by using solvers. In Section 5 we introduce the mev-commit protocol as a decentralized insured commitment protocol. In Section 6 we discuss the privacy properties of the mev-commit protocol and introduce the cryptographic primitives ARUBE and DPCOM. In Section 7 we discuss possible applications of the mev-commit protocol, focusing on preconfirmations. Finally, in Section 8 we conclude the document by discussing future work directions.

2 Insured Commitments

2.1 Overview

We here introduce the general concept of *insured commitments*. These are protocols in which a so-called *bidder* can send bids to a set of *commitment providers*, offering to pay a certain amount (the *bid amount*) to a provider fulfilling some condition specified in the bid (the *fulfillment condition*). This condition can be thought of as the bidder's objective or intent and the bid as an incentive for the providers to fulfill this objective. Providers can then issue commitments to these bids. The credibility of their commitments comes in the form of an economic guarantee: if the fulfillment condition is not satisfied, the provider has to pay a *compensation amount* to the bidder. Thus, the commitment serves as a form of *insurance* for the bidder that guarantees that either the bidder's objective is fulfilled, or the bidder receives the agreed compensation. More formal definitions are provided in Section 2.2. To increase the expressiveness and allow for a wide range of applications, the definitions there include several generalizations, such as bid and compensation amounts that depend on the time the commitment is issued and allowing the bidder to specify an *activation condition* that needs to be satisfied for the commitment to be considered relevant.

An important special case of insured commitments are *preconfirmations*. The bidder's objective for these in the simplest form is the inclusion of a transaction in a future block. More complex conditions such as execution guarantees can be considered, and we discuss preconfirmations in more detail in Section 2.4.

2.2 Definition of Insured Commitments

We here give a definition of *insured commitments*. The definitions we provide here are more technical than the rest of the whitepaper to provide a precise basis for later discussions. It is not necessary to fully understand all the details here to understand the rest of the whitepaper. Insured commitments involve a set of *bidders* $\mathcal{B}$ and a set of *commitment providers* $\mathcal{P}$. Each bidder $B \in \mathcal{B}$ can send bids to a subset of the providers in $\mathcal{P}$, where a bid is defined as follows.

Definition 1 (Bids)**.** Let A be a set of amounts and T be a set of points in time. A *bid* from $B \in \mathcal{B}$ consists of a tuple $\mathsf{bid} = (B, \beta, \gamma, \phi, \psi)$, where

- the *bid amount* $\beta \colon T \to A$ is a function mapping the time at which a commitment is given to the amount the bidder is willing to pay for a commitment at that time,
- the *compensation amount* $\gamma \colon T \to A$ is a function mapping the time at which a commitment is given to the amount the bidder wants to receive if the commitment is not fulfilled,
- the *activation condition* ϕ is a condition that needs to be met for a commitment to the bid to be relevant, and
- the *fulfillment condition* ψ is the condition the bidder wants to be fulfilled.

The set of amounts A can contain any kind of amounts, e.g., ETH, tokens, or other assets, whereas the time T could be real time, or some chain-specific time such as slot numbers. The commitment-time dependence of the bid and compensation amounts allows to incentivize providers to commit to bids early by letting the bid amount decay over time. The activation condition restricts the commitments to be effective only under certain conditions. This allows commitment providers to issue commitments also in situations in which they don't have full control over the fulfillment condition. E.g., if the commitment provider is a block builder, the activation condition could be that a certain block is built by that provider. More concrete examples are given in Section 2.4.

Any commitment provider receiving a bid from a bidder can then commit to the bid $\mathsf{bid} = (B, \beta, \gamma, \phi, \psi)$. An insured commitment is linked to the bid, the committing provider, the time at which the commitment was issued, and may contain auxiliary information. It is defined as follows:

Definition 2 (Insured commitments)**.** Let bid be a bid. An *insured commitment* for bid is a tuple $\mathsf{icom} = (\mathsf{bid}, P, t, \mathsf{aux})$, where P is the provider committing to the bid, t is the time at which the commitment was issued, and aux is some arbitrary (possibly empty) auxiliary information.

When a provider $P \in \mathcal{P}$ issues such a commitment at time $t \in T$, the provider is responsible for fulfilling the fulfillment condition ψ of the bid if the activation condition ϕ is met. An *insured commitment protocol* is a protocol that facilitates the interaction between bidders and providers.

Definition 3 (Insured commitment protocols)**.** Let $\mathcal{B}$ and $\mathcal{P}$ be sets of bidders and providers, respectively, let $\mathcal{T}_\beta$ and $\mathcal{T}_\gamma$ be sets of functions from some set T to some set A, let Φ and ψ be sets of conditions, let Aux be an arbitrary set, and let $\Delta \in \mathbb{R}$. An *insured commitment protocol* for these sets is a protocol that allows bidders $B \in \mathcal{B}$ to send bids $\mathsf{bid} = (B, \beta, \gamma, \phi, \psi) \in \{B\} \times \mathcal{T}_\beta \times \mathcal{T}_\gamma \times \Phi \times \Psi$ to providers in $\mathcal{P}$. For such a bid, it then allows providers $P \in \mathcal{P}$ to issue commitments $\mathsf{icom} = (\mathsf{bid}, P, t, \mathsf{aux})$ to this bid, where $t \in T$ and $\mathsf{aux} \in \mathsf{Aux}$, such that the following is enforced:

- The time t is within time Δ of the time at which the commitment is issued.[2]

[2]This technically requires T to be a metric space with some distance function.

- If the conditions ϕ and ψ are both satisfied, the provider receives the amount $\beta(t)$ from the bidder.
- If the condition ϕ is satisfied, but the condition ψ is not satisfied, the bidder receives the amount $\gamma(t)$ from the provider.
- If there a multiple commitments from the same provider for bids with the same fulfillment condition from the same bidder, only one of them gets settled.

We note that protocols supporting larger sets $\mathcal{T}_\beta$, $\mathcal{T}_\gamma$, Φ, and Ψ and hence more complex bids are more powerful. Furthermore, it is preferable to have protocols enforcing smaller Δ and ones that guarantee settlement quickly after both conditions ϕ and ψ are can be decided. Only settling one commitment for each fulfillment condition allows, e.g., bidders to issue multiple bids for the same fulfillment condition with increasing bid amounts, without risking to pay multiple times for the same fulfillment.

Remark 1. Note that each commitment should only be settled once and this must happen at a time at which the validity of both conditions ϕ and ψ are determined. This means in particular that if these conditions refer to the state of a blockchain, the commitment should only be settled after the state is finalized or at least after the probability of a potential reorganization is negligible. We discuss reorganizations more in Section 3.

2.3 Privacy of Insured Commitment Protocols

To prevent malicious actors from exploiting the information contained in bids and commitments, it is crucial to maintain the privacy of the bidders and providers throughout the protocol. We divide the privacy properties into three parts and (informally) define them separately below: bid privacy, commitment privacy, and recipient anonymity.

Definition 4 (Bid privacy)**.** An insured commitment protocol provides *bid privacy* if it allows bidders to specify a set of recipient providers for each bid and ensures that no party other than the bidder and the recipient providers can learn anything about the contents of the bid until a commitment to that bid gets settled.

Definition 5 (Commitment privacy)**.** An insured commitment protocol provides *commitment privacy* if no party other than the bidder and the provider issuing the commitment can learn anything about the contents of the commitment until it is settled.

Definition 6 (Recipient anonymity)**.** An insured commitment protocol provides *recipient anonymity* if for every provider $P \in \mathcal{P}$, nobody except for P can learn whether a given bid has been sent to P or not until a commitment from P to that bid is settled.

Bid privacy protects the privacy of the bidders and is crucial to prevent attackers from, e.g., frontrunning the bidders' transactions or adjusting their own bids based on other bidders' bids. Commitment privacy protects the privacy of the commitment providers and prevents competing providers from adjusting their strategies based on the commitments of other providers. Recipient anonymity prevents other bidders and providers from adjusting their strategies based on the recipients of a bid, e.g., by prioritizing transactions that have not been sent to a competing block builder.

2.4 Preconfirmations as Insured Commitments

A preconfirmation in the context of Ethereum and blockchains has been defined by Akdeniz as "A credible heads-up before a confirmation happens" [7]. A confirmation in the blockchain setting could mean different things, such as a transaction being included in a block, or the block

being finalized. We consider a preconfirmation to come before the transaction is included in a block. We here show how preconfirmations can be seen as a special case of insured commitments. The credibility of a preconfirmation in this case comes from the cryptoeconomic guarantee that the provider needs to pay a compensation to the bidder if the preconfirmation is not fulfilled.

The fulfillment condition ψ for a preconfirmation in its simplest would correspond to a transaction being included in a future block, specified by its block height. More complex condition would be the inclusion of a transaction bundle, inclusion in the top $X\%$ of the block, inclusion with execution guarantees such as non-reversion of a smart contract call, or even conditions spanning multiple blocks.

Preconfirmations from proposers and block builders. One can consider different commitment providers for preconfirmations. In particular, block proposers can issue preconfirmations for the blocks they propose and block builders can issue ones for block they build. The activation condition ϕ can here be used to ensure that the commitment is only relevant if the block is indeed proposed or built by the provider. More concretely, in the case of block builders as providers, if the fulfillment condition is about inclusion in a block at height h, the activation condition would be that the block at height h was built by the block builder that issued the commitment.

Ensure timely commitments and fuel builder competition. To incentivize providers to issue timely commitments (and in particular ensure that the preconfirmation is indeed a heads-up, and does not come after the confirmation), the bidder can specify a decay of the bid amount by letting β be a decreasing function such that providers receive less for later commitments. More concretely, if the fulfillment condition is about inclusion in a block at some height that is supposed to be proposed at time t, the bidder would set the bid amount β such that $\beta(t') = 0$ for all $t' \geq t$.

Furthermore, in case of block builders as providers, committing early and thus receiving a high bid amount allows the corresponding block builder to bid more in the mev-boost auction, potentially giving them a significant advantage over other builders.

More detailed execution guarantees from providers. The bidder can specify some execution requirements in the fulfillment condition, but in many cases, the bidder is not able to precisely specify the transaction outcome without restricting the bid too much. E.g., for swapping some number of token X to token Y, the bidder can specify that they want to receive at least some amount of token Y in the bid (e.g., as part of the transaction via a defined slippage). The bidder, however, will usually not be able to determine the precise amount of token Y they will receive since this also depends on transactions of other users included in the same block. The bidder can, however, require the commitment provider to inform the bidder about the precise amount as part of the preconfirmation: To achieve this, the fulfillment condition would include that the auxiliary information aux of the commitment contains the precise amount of token Y the bidder will receive and that this is indeed the correct amount.

2.5 Probability of Insured Commitments Being Activated

As discussed in Section 2.2, an insured commitment for a bid $\mathsf{bid} = (B, \beta, \gamma, \phi, \psi)$ between a bidder B and a provider P becomes "active" once its activation condition ϕ is met. When the commitment is activated, then there is a guarantee that either one of the two outcomes will happen: (1) the fulfillment condition ψ of the commitment will be met and P will receive the bid amount β, or (2) the bidder B will receive the compensation amount γ. It is thus typically in the bidder's interest to have a commitment that gets activated. For a simple preconfirmation commitment from a block builder, the activation condition is only met when the provider P that

issued the commitment builds the block that is included in L1 at the specified height. Thus, the probability of the commitment being activated equals the probability that P builds the block at the specified height. A bidder can increase the probability of having its fulfillment condition met by simply getting insured commitments from as many block builders as possible. As an example, the top 3 builders on Ethereum recently had a market share of over 97% of all block built via mev-boost, which contribute to over 90% of all Ethereum blocks.[3] Thus, if a bidder manages to source insured commitments from the top 3 block builders, then the probability of having *at least* one of its commitments activated is over 87%.

Increasing the probability via proposer lookahead and validator opt-in. The 87% in the above example are based on the assumption that whether the next proposers are using mev-boost is determined randomly. While proposers are indeed selected randomly, all proposers of the current epoch (spanning 32 slots in which blocks can be proposed) are predetermined and publicly known at the beginning of the epoch. If a proposer is using mev-boost, the probability of that proposer self-building the block is less than 3% [8]. Hence, if a sophisticated bidder knows that the next proposers are using mev-boost, the probability that at least one of the commitments from the top 3 block builders being activated increases to 94%. The probability can be increased further by letting validators "opt-in" to the insured commitment protocol by pledging to only propose blocks from block builders that are providers in that protocol. They can be incentivized to do so by an additional protocol. This way, the probability of having at least commitment activated can be increased to almost 100% if a commitment is obtained from all providers.

3 Reorganization Risk Mitigation

In most blockchains (such as Ethereum), reorganizations (or reorgs) can occur when a different fork of the chain gets adopted. Such forks can happen due to network delays, when a proposer needs to propose a new block before learning about the previous block, or during a network partition where different parts of the network lose communication with each other. Forks can also occur when a proposer behaves maliciously by proposing multiple blocks for the same slot. To resolve such forks, blockchains include a fork choice rule that determines which branch to adopt, ensuring that alternative branches eventually disappear, along with any blocks and transactions within them. Reorgs present risks such as double-spending attacks, where a user can spend the same funds on two branches of the chain. Therefore, many blockchains have some finality rule that ensures that finalized transaction can never be removed from the chain via a reorg. On Ethereum, however, finality is only reached after at least 12.8 minutes. In this section, we examine how insured commitments can help users to mitigate the risks associated with transactions disappearing after a reorg without waiting for finality.

3.1 Reorg Risk

As discussed above, forks can occur mostly in two scenarios. We now look a concrete example to illustrate these and the associated risk. See Figure 1 for a graphical depiction of the example. Assume the proposers of slots N, $N+1$, and $N+2$ all propose blocks at consecutive heights h, $h+1$, and $h+2$, respectively. Now, the proposer of slot $N+3$ receives the previous block too late, and is forced to again extend the chain from the block at height $h+1$. This creates a fork that divides the chain into two branches. Next, we assume the proposer of slot $N+4$ is malicious and proposes two different blocks, both extending the block from the previous proposer. The chain now has three branches, labelled branch 1, 2, and 3. For this example, we assume that

[3] As of November 2024, source https://explorer.rated.network/

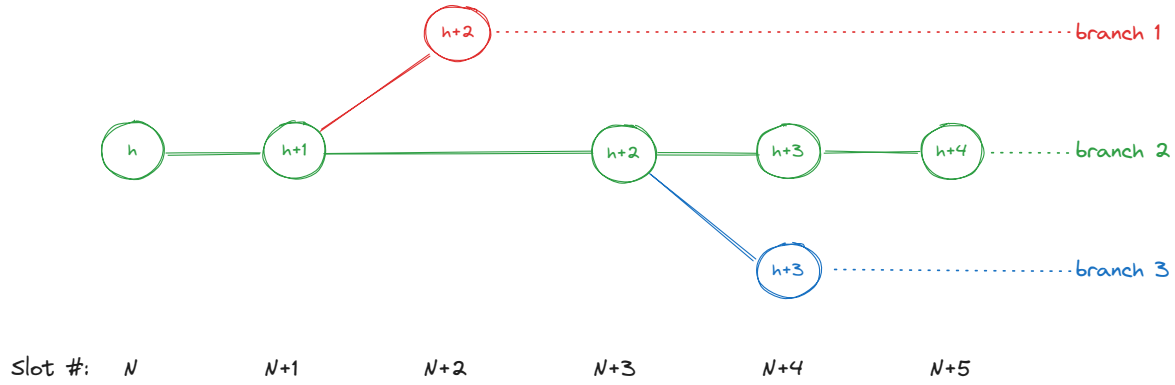

Figure 1: This graph illustrates two ways how Ethereum can fork, first by the proposer of slot $N+3$ receiving the previous block too late, and second by the proposer of slot $N+4$ maliciously proposing two different blocks. This creates three branches, where we assume the second one gets finalized and the others get abandoned. Each block is labeled with its block height.

future block proposers continue extending branch 2, while branches 1 and 3 are abandoned. Therefore, branch 2 ultimately gets finalized.

Now consider a user waits for a transaction to be "confirmed", i.e., included in a block, and assume it gets included in the red block at height $h+2$ in Figure 1. At this point, the user might take actions based on the assumption that this transaction gets executed. However, there is a risk that this transaction does not get included in branch 2, e.g., because a conflicting transaction is included there first. In this case, the user might lose funds.

3.2 Risk Mitigation with Insured Commitments

Now assume the user submitted a bid $\mathsf{bid} = (B, \beta, \gamma, \psi, \phi)$, where the fulfillment condition ψ specifies that the relevant transaction is included in a block at height $h+2$ and executes without reversion. And assume the user received an insured commitment $\mathsf{icom} = (\mathsf{bid}, P, t, \mathsf{aux})$ from a block builder P, where ϕ specifies that the block at height $h+2$ is built by P. Once the user receives this commitment (which is already *before* the transaction "confirmation"), the user is ensured that if ϕ is satisfied, then either the transaction gets included and does not revert, or the user receives the compensation amount γ. Note that this holds regardless of potential future forks and reorgs: As discussed in Remark 1, icom only gets settled when the remaining reorg risk is negligible. So in this example, the activation condition ϕ refers to the block at height $h+2$ in the finalized branch 2.

Therefore, the reorg risk for the user is now substituted by the risk that the activation condition Φ does not get fulfilled. As discussed in Section 2.5, the activation probability can be increased to almost 100% (and thus risk reduced to almost 0) by sourcing insured commitments from multiple providers and having opted-in validators.

Note reorgs can impact the effect of validator opt-in: If the proposer of slot $N+2$ is opted in, but the one of slot $N+3$ is not, then the activation probability on branch 1 is higher than on branch 2. However, if both validators are opted in, the activation probabilities on both branches are the same, assuming block builders build blocks independently of the branch that gets adopted (which might not be true in extreme scenarios such as network partitions). In particular, if insured commitments are sourced from all providers, the activation probability is almost 100% on both branches (even in extreme scenarios). This means that if the next n proposers are all opted in and reorgs spanning more than n slots are considered negligible,[4] the

[4]E.g., one can choose $n = 10$ or less based on the fact that the longest reorg in years was affecting 7 blocks [9], and that was before the transition to proof-of-stake, which should make reorgs even less likely.

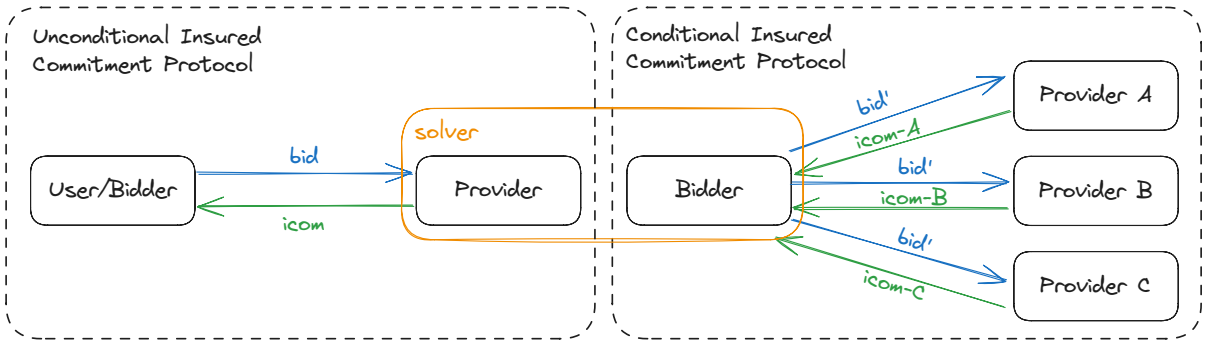

Figure 2: Overview of the construction of an unconditional insured commitment protocol from a conditional one. The solver receives a bid from a user and acts as a commitment provider in the unconditional protocol, issuing a commitment to the user, and at the same time as a bidder in the conditional protocol, sourcing conditional commitments from multiple providers.

risk can be almost completely mitigated. Note that all of this is known to the user at the time the commitments are received, i.e., even *before* the transaction is "confirmed".

4 From Conditional to Unconditional Insured Commitments

Insured commitments are subject to an activation condition ϕ. In case of block builder issued commitments, this mean that commitments only get activated with a certain probability, as discussed in Section 2.5. Sophisticated actors can mitigate this by sourcing commitments from multiple providers, but this is not feasible for all use-cases. In this section, we discuss how one can construct an *unconditional* insured commitment protocol from one with activation conditions as in the block builder case. Technically speaking, an unconditional insured commitment protocol is one with $\Phi = \{\mathsf{true}\}$, i.e., the activation condition is always satisfied for all commitments. Bidders can thus directly treat a single commitment as an insurance in the sense that either the fulfillment condition is met, or the bidder is compensated. As discussed in Section 3, this insurance holds even in case of reorganizations. Therefore, if such a protocol is used for preconfirmations, the credibility of the preconfirmation is even higher than an actual confirmation in the sense of inclusion in a block, and essentially provides instant finality if the bidder is equally happy with the compensation amount as with getting the fulfillment condition met.

4.1 Unconditional Insured Commitment Construction

Assume a protocol for conditional insured commitments is given, e.g., one with block builders as providers. We now show how to construct an unconditional insured commitment protocol from this. The idea is simple: The commitment provider in the unconditional protocol is a sophisticated actor, such as a solver, that uses the conditional commitment protocol to source multiple commitments for the same fulfillment condition and compensation amount (from block builders). The solver then (or already optimistically before receiving these commitments) issues an unconditional commitment to the user. This mechanism has been described as "sourcing leaders" for preconfirmations by Matt and Akdeniz in an Ethereum Research article [4]. See Figure 2 for an illustration of the construction.

The user in this case gets an unconditional commitment from the solver, which means that the solver needs to pay the user the compensation amount if the fulfillment condition is not met, for whatever reason. The solver thus takes on the risk of the activation conditions for the conditional commitments not being met. Hence, the solver will try to minimize the risk of this happening and carefully assess the probability as discussed in Section 2.5. Any remaining risk then needs to be priced into the bid amount, e.g., if the bidder asks for a 1 ETH compensation

amount and the solver has a 10% probability of not having any of the conditional commitments activated, then the solver needs to charge at least 0.1 ETH (plus some profit margin) for the unconditional commitment.

5 The mev-commit Protocol

In this section we describe our implementation of an insured commitment protocol, called *mev-commit* protocol. Currently, mev-commit focuses on a special type of insured commitments, i.e., preconfirmations, where the fulfillment condition ψ is the inclusion or execution of a transaction bundle in a future block of Ethereum. The default providers on mev-commit are Ethereum block builders and the activation condition ϕ is that the block to which the fulfillment condition refers is built by the provider committing to the bid. The bid amount β is the amount the bidder is willing to pay for the inclusion or execution of the transaction bundle (with a linear decay with respect to the time the bid is emitted). The credibility of active commitments comes from the economic guarantee that the provider has to pay the compensation amount to the bidder if the fulfillment condition is not met. The protocol is described in more detail next.

5.1 Protocol Overview

The mev-commit protocol proceeds in rounds and is executed between a set of bidders $\mathcal{B}$ and a set of providers $\mathcal{P}$. In each round N, bidders can send bids of the form $\mathsf{bid} = (B, \beta, \gamma, \psi, \phi)$ to chosen providers. Each bidder $B \in \mathcal{B}$ can define and regularly update a set of providers $\mathcal{R} \subseteq \mathcal{P}$ to which they wish to send their bids to. The bids are then received precisely by the providers in $\mathcal{R}$, and remain hidden from everyone else (more about the privacy features of mev-commit in Section 6).

One core component of mev-commit is the mev-commit blockchain, used to store the commitments. Providers commit to bids by sending a cryptographic commitment to the bidder and recording this commitment on the mev-commit chain. The cryptographic commitment binds the provider to that particular bid while also hiding any information about the bid. This commitment later needs to be "opened" in order for it to be publicly settled. This opening information contains secret values that are only shared by the bidder and the provider, which are the only entities capable of opening a commitment. An oracle monitors the L1 chain, recording details about transactions included in specific blocks on the mev-commit chain. If a provider has built the corresponding L1 block and seeks to claim rewards for a fulfilled commitment, they must open the commitment activated by that block's inclusion in L1. Should the provider fail to open a commitment—perhaps because the commitment was not fulfilled—the bidder can open it to enable settlement by the oracle. An illustration of an honest execution of the mev-commit protocol is shown in Figure 3.

Increasing the probability of commitments being activated. To ensure a high level of assurance for bidders in the mev-commit protocol, the ideal scenario is to have insured commitments from all providers for a specified fulfillment condition. However, even with full coverage, there remains the risk that an L1 block is constructed by a builder outside the mev-commit protocol, which prevents the activation of any commitments. To address this, the protocol allows Ethereum validators to opt into mev-commit, increasing the likelihood that L1 blocks are built by participating providers. Validators who join the protocol commit to proposing only L1 blocks constructed by block builders participating in mev-commit, with penalties imposed on non-compliant validators. Additionally, validator opt-in helps prevent Sybil attacks by builders who might attempt to build blocks under different identities to avoid accountability for commitments they have made. This approach can also extend to other blockchains and rollups where validators directly construct blocks, like sequencers that organize transactions

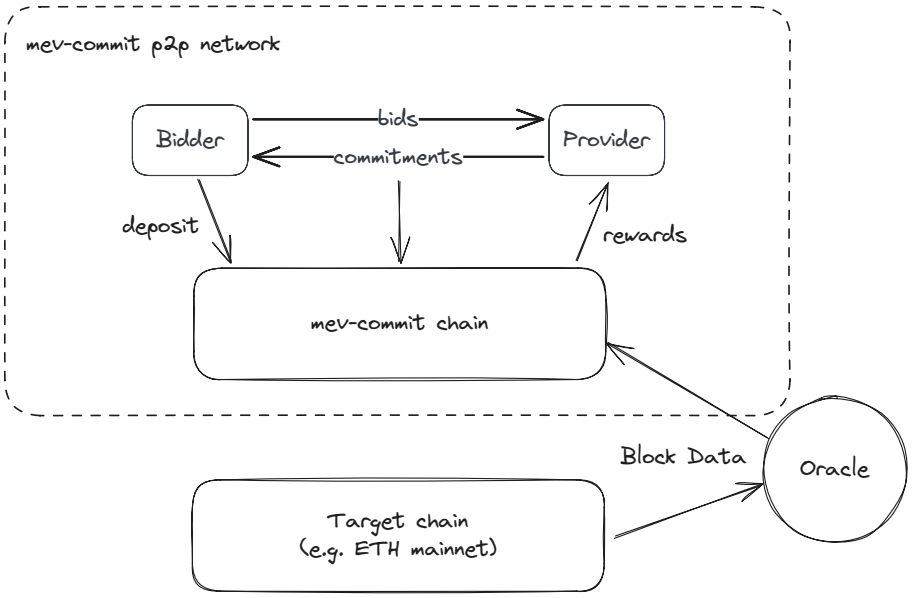

Figure 3: Overview of an honest execution of the mev-commit protocol for a single bidder and provider. Bidders and providers interact through the mev-commit p2p network, which in turn interacts with the mev-commit chain to include commitments. The oracle service monitors the L1 chain and reports the relevant information about the winning block to the mev-commit chain for settlement. Bidders need to deposit sufficient funds to cover their bids, which is assisted by an auto-deposit mechanism. Providers get the rewards after opening their commitments if they honored their commitments.

into blocks. These entities can register as providers in mev-commit without additional validator opt-in, ensuring that commitments within their domains are reliably activated.

Validators can opt into mev-commit by registering their BLS public key through mev-commits's validator registry contracts, signaling participation to other actors. Notably, validators can also restake their 32 ETH from Ethereum's beacon chain through compatible restaking protocols, including Eigenlayer and Symbiotic. This staked or restaked value supports the security of the mev-commit protocol and incentivizes validators to include blocks from mev-commit providers in the L1 chain. To enable this, validators can use mev-boost relays that support mev-commit by having a filter that ensure that only block from registered providers are relayed to the proposer. Otherwise, validators risk proposing a non-compliant block. Tokens staked directly on the mev-commit chain can be withdrawn upon request, while restaked tokens may follow different rules based on the restaking protocol. Importantly, if a validator acts maliciously within mev-commit, they may be penalized and lose a portion of their staked tokens. More details on the validator's economics are discussed in Section 5.2.2.

The mev-commit chain. The mev-commit chain is an EVM-compatible blockchain designed to provide fast transaction processing and low transaction fees. Currently, it is built out as an Ethereum sidechain with a single validator node. This setup allows for faster block production. The block time of the mev-commit chain is set to 200 milliseconds, meaning 60 mev-commit chain blocks are produced for every Ethereum mainnet block. The block time will likely change as enhancements are added and chain state grows.

At this point, Primev entities run all validating infrastructure for the mev-commit chain. Nevertheless, correct and honest operation can be permissionlessly audited by spinning up a full node and connecting to the mev-commit chain as a peer. Furthermore, the strong privacy guarantees of the mev-commit protocol ensure that commitments cannot be censored based on their contents. Over time, it is planned to have entities outside of Primev to join the network

and progressively decentralize the system. Furthermore, the mev-commit chain will continue to evolve. Primev conducts ongoing research into consensus algorithms that can be employed to decentralize the mev-commit chain without compromising on speed.

Bridging. The mev-commit chain utilizes advanced bridging mechanisms to ensure a secure and efficient transfer of assets between the Ethereum mainnet (L1) and the mev-commit chain. Primev envisions mev-commit to eventually integrate a multitude of bridges to other chains, using multiple tech stacks. In the meantime, Primev has implemented its own lock/mint bridging protocol. Native ETH on the mev-commit chain maintains a 1:1 peg with ETH on L1. The only way to mint ETH on the mev-commit chain is to lock equivalent ETH in a bridge contract on L1. ETH can be burned on the mev-commit chain's bridge contract to release equivalent ETH from the L1 bridge contract.

L1 oracle service and builder attribution. The oracle is an off-chain process that interacts with the oracle contract on the mev-commit chain. It monitors and extracts the winning block builder and corresponding transaction data from each L1 block, and submits this data to the oracle contract residing on the mev-commit chain. The oracle service is currently operated by Primev, but there is an ongoing effort to decentralize the oracle role, c.f. Section 8.2.

To prevent reorgs from interfering with settlements as discussed in Remark 1, the oracle is currently running 10 block behind the head of the Ethereum chain. Given that the longest reorg in years was affecting 7 blocks [9] and that this was before the transition to proof-of-stake, which should make deep reorgs even less likely, waiting for 10 blocks is a good tradeoff between safety and settlement speed.

Currently, the oracle can recognize the builder that built a L1 block by querying the relays for the BLS public key of the builder. The key is then compared to the list of public keys of registered providers in the mev-commit protocol. This mechanism currently relies on the relay providing accurate information, but there are plans to improve it in the future.

5.2 Economics of mev-commit

The providers and validators that opt-in to the mev-commit protocol are required to stake a certain amount of tokens to participate in the mev-commit protocol. We stress that the protocol is designed to reward providers for fulfilling their commitments and to penalize them for breaking them. Additionally, validators that are chosen as proposers in a slot are penalized if they propose blocks from builders that are not registered as mev-commit providers.

5.2.1 Provider's Incentives and Penalties

There are three possible outcomes after a commitment is made by a provider and an Ethereum (L1) block is proposed: (1) The bid amount is rewarded to the provider for fulfilling the commitment, (2) the provider does not fulfill the commitment and pays the bidder the compensation amount, (3) nothing happens as the commitment is neither opened by the provider nor the bidder. We discuss each case in more detail below.

1. Upon successfully fulfilling and opening a commitment, a provider will be rewarded the bid amount. This reward is paid by the bidder (through the mev-commit oracle) to the provider's account after the corresponding slot is settled by the oracle.

2. Upon not fulfilling a commitment, the bidder can open the commitment and the provider will pay the bidder the compensation amount. This payment is done by the mev-commit oracle on behalf of the provider. Currently, the compensation amount in every bid is set by default to be equal to the bid amount they committed to.

3. The protocol does not take action on commitments that do not get opened by either the provider or the bidder. This can happen in particular if the provider did not build the corresponding L1 block.

More details and other up to date mev-commit economic policies can be found in the mev-commit protocol documentation [10].

5.2.2 Validators Incentives and Penalties

A validator that opts into mev-commit makes an implicit commitment to propose a block from a mev-commit registered block builder in L1 whenever they become the proposer of that slot. An important observation here is that proposers opting-in increases the credibility of the commitments and consequently their value. Due to the increased values, the providers have additional value to bid in the mev-boost auction, thus driving up the total revenue a proposer will get. Next, we discuss the possible outcomes for a validator that is chosen to be the proposer for a slot.

1. The proposer includes a block in L1 by a builder that is not participating in the mev-commit protocol. This can be due to a malicious behavior by the proposer or due to the proposer using a relay that is not compatible with mev-commit (and thus delivering blocks not built by mev-commit providers). In this case, the proposer will be penalized by having a portion of its stake slashed and transferred to the mev-commit treasury account.[5]

2. A proposer can completely miss its slot and not include any block in L1. This can happen either due to network issues by the proposer, due to the relay not delivering the block's contents on time, or due to chain reorgs. In any case, mev-commit does not penalize the proposer for missing a block as the proposer does not profit from omitting a block and it is potentially out of their control.

3. The proposer includes a block that was built by itself. This case should only happen occasionally (around 3% of the time [8]) when the value of the block the proposer receives from the relay is too low. This should happen even less often when providers issued commitments for the block as this increases the value of the block. Proposers are not slashed in this case. The rationale is that slashing a proposer that proposes a self built block could lead to low adoption by proposers. Looking ahead, in the future work section (Section 8.2), we briefly describe a way to improve this mechanism even further.

See the mev-commit protocol documentation [10] for more details.

5.2.3 Transaction Fees

The transactions fees in mev-commit are paid using mev-commit's wrapped ETH token. The base fees accumulate in the mev-commit protocol treasury and the priority fees are paid to the mev-commit block proposer. The base fee is used to cover the costs of running the mev-commit chain, while the priority fee is used to incentivize the block proposer.

5.3 Bid Decay Mechanism

Bidders are generally willing to offer higher rewards to providers who issue commitments more promptly. This is where the bid decay mechanism comes into play: it computes the bid amount

[5] In case of restaking with Eigenlayer, it is only possible to freeze the stake instead of slashing, where proposers with frozen stake are considered not opted in, and need to pay a fee to unfreeze their stake. This is a current limitation of Eigenlayer that is expected to be resolved.

as a function of the time elapsed since the bid was placed. In alignment with the definition of insured commitments, the bid decay mechanism is how mev-commit implements the β function. The decay mechanism must achieve the following properties:

- *Predictability of decay*: The providers can *accurately compute* how much their decay will be for a commitment.
- *Public verifiability of decay*: The decay mechanism offers public verifiability, as the decay amount can be calculated using the public timestamps from the bid and the commitment.
- *L1 synchrony independent*: The decay mechanism must maintain independence from the L1 block time, ensuring that the decay mechanism functions correctly regardless of whether the L1 block confirmation occurs more quickly or experiences delays.

5.3.1 Mechanism Description

When submitting a bid, a bidder attaches two timestamps to the bid: (1) a timestamp indicating the exact time of dispatch of the bid, and (2) an expiry timestamp, marking the moment when the decay for this bid reaches 100%—essentially, the point in time beyond which the bidder is unwilling to pay anything for a commitment. When a provider decides to commit to the bid, they must issue a commitment and also assign a "commitment timestamp" to it. This commitment is then sent to the mev-commit chain for inclusion by the mev-commit validators. The mev-commit chain will include all commitments, however the commitment will only be considered valid if the commitment timestamp supplied by the provider and the block timestamp where the commitment was included are "close enough". The meaning of close enough here is determined by a parameter Δ that is set by the mev-commit network to account for the latency in message delivery among nodes in the network. More precisely, for a commitment to be valid, it is required that $t_b - t_c \leq \Delta$, where t_c denotes the commitment timestamp and t_b denotes the timestamp of the block where the commitment was included.

Decay function. The decay function adopted in mev-commit is a linear decay, i.e., the bid's value β decreases linearly from the moment it is issued until its deadline. For example, if the decay proportion is 1/2, the bid's value decays by 50%. The linear function is simple to compute and seems to be an effective choice. A future research direction is to determine how the selection of different decay functions can influence the providers' strategy in determining their commitments, as well as the bidders' strategy in setting their bid amounts.

Predictability of the decay. An important feature of the decay mechanism design is that a provider can compute the exact decay of its commitment before issuing it. This predictability of decay allows for providers to determine the accurate value of their commitment without the risk of issuing a commitment that will decay beyond expectations. This can only be achieved because the provider sets the commitment timestamp locally before sending it to the mev-commit chain for inclusion. A commitment will be deemed *valid* or *invalid* depending on the difference between the commitment timestamp set by the provider and the block timestamp of the block, where the commitment was included. Once the commitment is included in the chain, all the timestamps related to the commitment are final. Thus, it can be publicly determined if the commitment is valid, and if so, what the value of the decay of the commitment is. The decay function is then applied to actually compute the value of the commitment appropriately.

Setting the parameter Δ. The parameter Δ should be set such that whenever a provider sets the commitment timestamp correctly to the current time, the commitment will be deemed valid. An assumption that we make is that for any pair of nodes A and B in the mev-commit

network, if A sends a message to B at time t, then the message will be received by B by time $t + \varepsilon$ (where ε is an upper bound of message delivery latency in the network). For simplicity, we also assume that a slot is never missed in the mev-commit chain. As an example, if we have that $\varepsilon < s$, where s denotes the slot duration in the mev-commit chain, then any transaction sent at the very beginning of slot n will be received by the mev-commit chain validator during slot n and thus be included in the block produced at slot $n + 1$.[6] Since validity of the commitment is determined based on the timestamp of the block the commitment was included in, one needs to account for an additional delay of up to s time after the mev-commit validator receives the commitment. Therefore, Δ needs to be set such that $\Delta \geq s + \varepsilon$. The parameter ε should be computed by gathering empirical data through measuring the actual delay for message delivery among nodes on different geolocations.

5.3.2 Analysis of the Mechanism

The robustness of the mechanism comes from the fact that the participating parties have conflicting incentives. In particular, we note that there is no incentive for the bidder to manipulate the choice of the bid's timestamps; if the bidder includes a timestamp in the future the bid decays less (so the bidder pays more). If the bidder includes a timestamp that is too old, then the provider can choose to not commit as the decay for the bid is too high (then the bidder gets no commitment). Moreover, we note that both manipulations would have the exact same effect as the bidder using an accurate timestamp and changing the price of its bid prior to the submission of the bid. Similarly, the commitment timestamp set by the provider must not be too old, as otherwise, it won't be valid when included in the mev-commit chain. However, the provider can attempt to set the commitment timestamp slightly in the past if they have a low latency to the mev-commit chain. This approach may offer a slight advantage to the provider while still ensuring the commitment is valid, which is at most ε. In many network-based systems, including our decay mechanism, faster connections often provide better opportunities, such as quicker access to information or response times. This situation is similar to high-frequency trading in financial markets, where traders with faster connections can outperform others. In our system, any advantage due to a faster connection is bounded by ε.

5.4 Bidder Deposit and Double-spending Prevention

Bidders need to deposit and lock tokens to ensure that they can cover all their bids. The used mechanism in particular needs to prevent double-spending. Our setting poses several challenges: First, bid amounts are private and only known to bidder and the corresponding provider until the commitment is opened. Secondly, bidding happens at a high frequency, so making an on-chain transaction for every bid is undesirable.

Potential issues and example. To better understand the potential problems, we consider an example scenario involving two providers A and B and a bidder who has deposited 10 ETH in total. The bidder can make bids for different L1 block heights and the two providers can commit to a subset of those bids. Clearly, all individual bids must be at most 10 ETH, but this is not sufficient to prevent double spending if providers commit to multiple bids. On the other hand, it is clearly sufficient if all bids together sum up to at most 10 ETH. But this is not necessary as the following example illustrates.

For the block at height h, provider A can commit to two bids with amounts 5 ETH and 3 ETH, respectively, and provider B can commit to a 7 ETH bid. Even though the total committed value equals 15 ETH, it is guaranteed that the bidder can always cover all bids because only one of the two providers can build block number h. Since the maximum total commitments over

[6] For more information on how slots behave in Ethereum and in the mev-commit chain (that is a variation of Ethereum's geth client) we refer the reader to [11].

both providers for block h corresponds to 8 ETH, both providers can commit to a bid worth 2 ETH for block number $h+1$, and the bidder is still able to cover all possible outcomes.

Now assume provider B commits to a 1 ETH bid for block $h+2$. Even though both providers only committed to bids totaling 10 ETH, it is possible that the bidder cannot cover the total amount: If provider A builds block h and provider B blocks $h+1$ and $h+2$, the bidder would need to pay 11 ETH in total. Note that since bid amounts in the commitments are private, there is no way for provider B to know how much provider A has committed to. The example is summarized in Table 1.

Table 1: Example with two providers and a bidder with 10 ETH total deposit.

	Block h	Block $h+1$	Block $h+2$	Sum
Provider A commitments	5 ETH, 3 ETH	2 ETH		10 ETH
Provider B commitments	7 ETH	2 ETH	1 ETH	10 ETH
Maximum	8 ETH	2 ETH	1 ETH	11 ETH

Preventing double-spending. The examples above illustrate that a provider accepting bids for block h needs to know the amount deposited by the bidder and an upper bound on the committed amounts from all providers for that bidder for previous not yet settled blocks. One possible way to achieve this would be to only accepts bids for block h once all previous blocks have been settled and thus the available amount of the bidder is determined. This is, however, not a viable solution because settlement has some inherent delay and bidders should be able to place bids for block $h+1$ even before block h is built on L1.

The issue is circumvented in mev-commit by having a dedicated deposit for each round, i.e., L1 block number. E.g., a bidder can have deposited 1 ETH for the next round. Even though bid amounts are private, they are known to the providers accepting the bids and thus each provider can ensure that they do not commit to more than 1 ETH for this particular block. Commitments by other providers are not known, but since each block can be built only by one provider and deposits are locked per round, double spending can be prevented effectively. To ease the process for bidders, they can deposit a larger amount that is automatically split over a window spanning multiple rounds. Bidders can further enable automatic re-deposits, which automatically deposits remaining funds from previous windows. Withdrawals are only possible after settlement of the corresponding window.

5.5 implementation Benchmarks

The mev-commit protocol has been implemented by the Primev team and stress-tested internally. As mentioned in Section 5.1, the mev-commit chain was operated by a single validator node with a block time of 200 milliseconds. For the stress test, 5 bidder nodes and 3 provider nodes were used, with the bidders constantly submitting bids for transaction inclusion, and the providers being configured to issue commitments for around 80% of the bids. Over a period of five days, the average number of bids sent per second was 126.8, and the average number of commitments issued per second was 101.4. This means that every bidder sent around 25 bids per second, and each provider issued around 42 commitments per second. Since there are 5 mev-commit blocks per second, this corresponds to 25 commitments per mev-commit block, and since there is an Ethereum block every 12 seconds, there can be up to 1500 commitments for each L1 block.

6 Privacy of mev-commit

In this section, we introduce two novel cryptographic primitives that, when combined, enable mev-commit to achieve the privacy properties essential for the insured commitments described in Section 2. These primitives are anonymous receiver-updatable broadcast encryption (ARUBE) and dual-phase commitments (DPCOM). ARUBE facilitates the private dissemination of bids from bidders to selected providers, while DPCOM allows for secure commitment to bids, followed by the ability to open these commitments later. Each primitive is detailed separately in the following sections. The privacy of mev-commit and these primitives have previously been described in a Mirror article by Magri and Matt [12].

6.1 Anonymous Receiver-Updatable Broadcast Encryption (ARUBE)

We now introduce a cryptographic primitive that allows bidders in the main protocol to privately send bids to a subset of the providers without revealing what this subset is. This functionality is provided by *anonymous broadcast encryption* [13, 14]. A feature of broadcast encryption is that a new set of recipients can be specified for each message that is encrypted. In our use-case, the set of recipients for a given bidder typically remains stable since bidders mostly send their bids to the set of providers they work with and only occasionally add or remove a provider from that set. To be able to leverage this for more efficient schemes, we refine the notion of broadcast encryption by adding an algorithm for updating the set of recipients, while the encryption algorithm encrypts a message for a previously determined recipient set. We call this notion *anonymous receiver-updatable broadcast encryption (ARUBE)*. See Figure 4 for an illustration of ARUBE and its security.

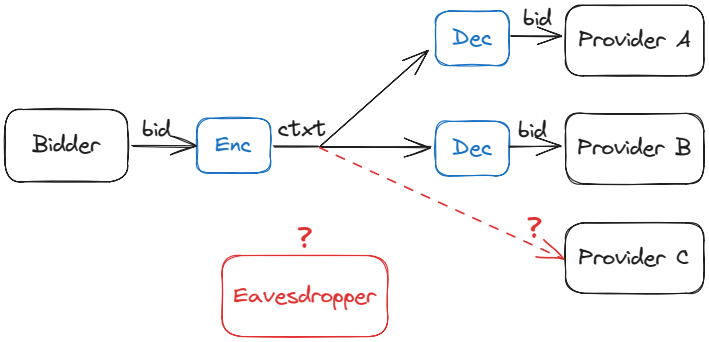

Figure 4: Illustration of ARUBE in operation, where a bidder securely broadcasts an encrypted bid such that only providers A and B can decrypt. Provider C and any eavesdropper, including other bidders, are unable to learn anything about the bid.

6.1.1 Security Properties of ARUBE

ARUBE provides two security properties: *anonymity* and *confidentiality*. Anonymity ensures that for every provider P, nobody except P learns whether P is in the set of recipients $\mathcal{R}$. Confidentiality ensures that eavesdroppers and providers not in $\mathcal{R}$ cannot learn anything about the bid. Note that the way ARUBE is used in mev-commit implies that after a provider opens a commitment, everybody learns that this provider was in the recipient set of the corresponding bidder. Anonymity can thus only be guaranteed in the long run for bidders who regularly update their recipient sets.

We note that while security guarantees of ARUBE resemble the bid privacy and recipient anonymity properties required for insured commitments (cf. Definitions 4 and 6), they alone are not sufficient to fully satisfy them. Bid privacy demands that no information about the bid's contents is revealed to other parties until the commitment is settled. ARUBE alone cannot

guarantee this level of privacy, as the commitment itself could potentially leak information. The same holds for recipient anonymity. However, when ARUBE is combined with DPCOM, both properties are fully achieved.

6.1.2 Instantiation of ARUBE

We here describe a generic instantiation of ARUBE using an anonymous public-key encryption scheme [15] and a symmetric encryption scheme. We use elliptic curve ElGamal encryption as the anonymous public-key encryption scheme and the Advanced Encryption Standard (AES) in Galois/Counter Mode (GCM) as the symmetric encryption scheme.

Setup. When providers join the network, they generate a public/private key pair $(\mathsf{pk}, \mathsf{sk})$ for the anonymous public-key encryption scheme. The public key pk also serves as the public key of the ARUBE scheme and is shared with all bidders. The providers keep their secret key sk private.

Setting and updating recipients. Bidders can define the set of providers $\mathcal{R}$ they wish to send their bids to. To do so, they generate an AES secret key k and encrypt it under the public keys of the providers in $\mathcal{R}$. The obtained ciphertexts are then concatenated (in random order) and sent to the providers. To prevent replay attacks and to allow the providers to learn which of the ciphertexts is intended for them, the bidder also sends an encryption of the current time under key k. The providers then try to decrypt all public-key ciphertexts to obtain secret keys k' and decrypt the timestamp with k'. If a valid and recent timestamp is obtained, the provider stores the corresponding key k'.

Sending bids. To send a bid to the providers in $\mathcal{R}$, the bidder encrypts the bid under the AES secret key k and sends the ciphertext to all providers. The providers in $\mathcal{R}$ can decrypt the ciphertext using the previously obtained secret key k.

6.2 Dual-Phase Commitments (DPCOM)

DPCOM is a two-party commitment protocol that allows a message to be committed in two distinct phases. In the first phase, one party (in our case the bidder) *precommits* to a message and receives both a *precommitment* and an *opening key*. In the second phase, the other party (the provider) commits to the precommitment generated in the first phase, receiving a *decommitment information* as the output.

A key feature of a dual-phase commitment scheme is its flexibility in opening the resulting commitment using either the opening key or the decommitment information. This is especially useful in the context of the mev-commit protocol, where the bidder generates a precommitment to its bid, after which the provider commits to that precommitment. This process ensures that both parties can open the commitment: the bidder obtains the opening key when generating the precommitment, and the provider obtains the decommitment information when committing to the precommitment. Consequently, both parties can later prove the content of the committed message to third parties. Crucially, if the provider misbehaves and refuses to publicly open the commitment, the bidder can independently open it without needing assistance from the provider. See Figure 5 for an illustration of DPCOM and its security.

6.2.1 Security Properties of DPCOM

Like a traditional commitment scheme, dual-phase commitment schemes must satisfy two security properties, namely *hiding* and *binding*. Hiding means that one cannot learn anything about the bid locked into the commitment without knowing either the opening key or the decommitment

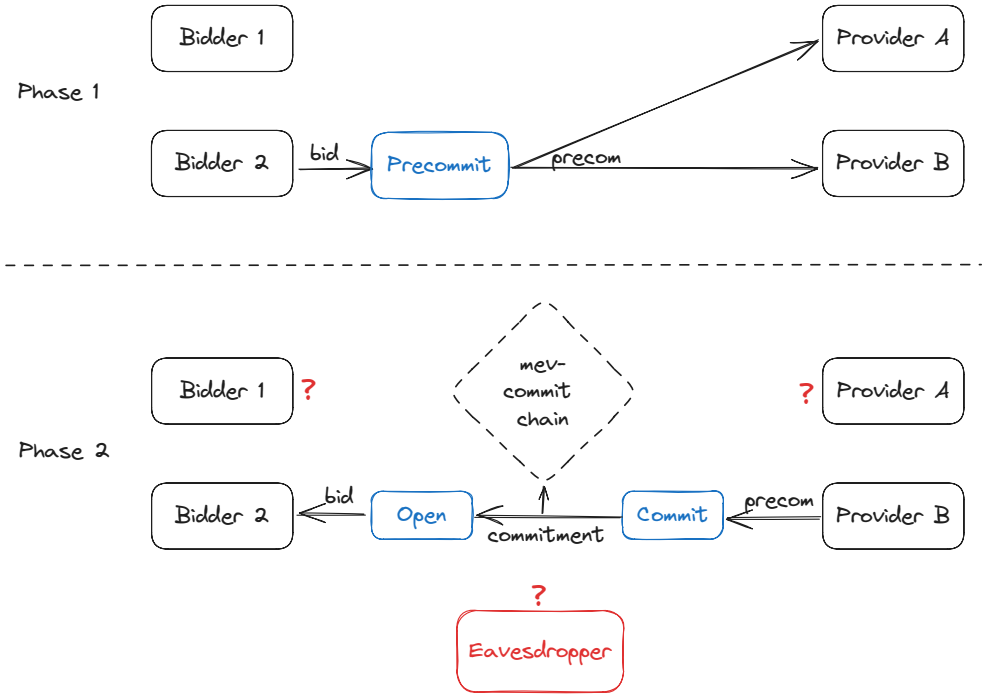

Figure 5: Illustration of the two phases of DPCOM. In phase 1, bidder 2 precommits to "bid" and sends the precommitment value to providers A and B. In phase 2, provider B commits to the precommitment sent by bidder 2 during phase 1. The commitment is then stored in the mev-commit chain and sent to bidder 2 that can open it for validation. Importantly, other providers, bidders, and eavesdroppers are unable to learn anything about the bid.

information. In particular, this means that only the provider committing to a bid and the bidder who generated it learn anything about the bid from the commitment. Binding means that a commitment to a bid cannot be opened to reveal a different bid. This holds for both the bidder and the provider opening and even if they collude to open the commitment together.

Similarly to ARUBE, the security guarantees of DPCOM alone are not sufficient to fully satisfy the commitment privacy property (cf. Definition 5) of insured commitments. The reason is that the DPCOM properties are only with respect to the commitment, and in particular do not rule out that the bid itself is leaked. However, as pointed out before, when DPCOM is combined with ARUBE, all the privacy properties of insured commitments are achieved.

6.2.2 Instantiation of DPCOM

We instantiate DPCOM generically from a non-interactive key-exchange (NIKE) scheme – a cryptographic primitive that allows parties to agree on a common secret key without any interaction – and a non-interactive cryptographic commitment scheme. We use the Diffie-Hellman key exchange protocol as the NIKE scheme and a hash-based commitment scheme in the random-oracle model.

Setup. Each provider samples a NIKE key pair $(\mathsf{NIKE.pk}, \mathsf{NIKE.sk})$ and publishes the public key.

Sending bids. To send a bid to the providers, the bidder generates a NIKE key pair $(\mathsf{NIKE.pk}', \mathsf{NIKE.sk}')$ and sends the public key $\mathsf{NIKE.pk}'$ together with the signed bid to the providers (using ARUBE). The secret key $\mathsf{NIKE.sk}'$ serves as the DPCOM opening key and the public key $\mathsf{NIKE.pk}'$ together with the signed bid correspond to the precommitment.

Committing to bids. When a provider wants to commit to a bid, they first generate a shared secret s with the bidder using the NIKE scheme by combining the provider's NIKE secret key $\mathsf{NIKE.sk}$ with the bidder's public NIKE key $\mathsf{NIKE.pk}'$. This shared secret is then used as the randomness to commit to the bid using the hash-based commitment scheme. The obtained commitment is sent to the bidder (and in our overall scheme recorded on the mev-commit chain). The public key $\mathsf{NIKE.pk}'$ corresponds to the decommitment information.

Opening commitments. Both the bidder and the provider can compute the shared secret s using their NIKE keys. They can therefore both open the commitment, but no other party can do so.

7 Applications of mev-commit

We here describe two direct applications of insured commitments that are possible with mev-commit today. Many additional application are possible, some of which we sketch in Section 8.2.

7.1 Instant Bridging

Blockchain ecosystems are inherently isolated, each operating independently with its own native tokens and protocols. This isolation presents challenges for asset transfer across blockchain networks. For instance, transferring tokens directly from Ethereum to Arbitrum is impossible without a bridging solution due to the structural differences between these networks. Blockchain bridges address this issue by enabling cross-chain interoperability, allowing assets to move between networks by issuing equivalent tokens on the target chain.

However, existing bridge solutions frequently yield a slow and cumbersome user experience. Transaction finalization often requires several seconds or even minutes, as bridges cannot release assets on the target chain until the transaction is confirmed on the origin chain. Factors such as transaction inclusion time, network congestion, and chain finality can contribute to these delays. For example, on Ethereum, transaction inclusion may take multiple seconds, and in cases of congestion, several minutes; waiting for transaction finality may add an additional 13 minutes.

In this section we explore how mev-commit can be integrated with bridge protocols to dramatically enhance the speed of cross-chain transfers, creating an instant bridging experience. We discuss the different types of bridges, the role of mev-commit in the bridging process, and the risks and mitigations associated with instant bridging. We also examine the costs of securing preconfirmations and the current limitations of mev-commit-based instant bridging solutions.

7.1.1 Existing Types of Bridges

In this section we give a quick introduction to the two primary types of bridges: wrapped asset minting bridges and liquidity pool bridges. Readers familiar with these concepts can safely skip to Section 7.1.2.

Wrapped asset minting bridges. Wrapped asset minting bridges facilitate token transfers by creating a "wrapped" version of an asset on the target blockchain that corresponds to the original asset on the origin chain. This method allows the wrapped token to be traded or used within the target network while the original asset is held securely in a smart contract on the source chain. The process typically involves the following steps:

- *Escrow on origin chain*: When a user initiates a cross-chain transfer, the original asset is locked in a smart contract on the source chain, securing the asset until the user redeems it.

- *Minting on target chain*: An equivalent wrapped token is minted on the target chain, maintaining a 1:1 value parity with the original asset. These wrapped tokens can then circulate freely within the target network.
- *Withdrawal process*: To reclaim the original asset, users burn the wrapped token on the target chain, which triggers the release of the original asset from the escrow contract on the source chain.

Liquidity pool bridge. Liquidity pool bridges enable cross-chain asset transfers by maintaining pools of native and wrapped tokens on each chain. Users deposit assets into these pools on one blockchain and receive an equivalent amount on the other chain. The process typically involves the following steps:

- *Deposit*: Users deposit assets into a liquidity pool on the source chain.
- *Token exchange*: When a user seeks to exchange tokens from the origin to the target chain, a designated entity (e.g., a solver) holding tokens on the target chain receives the tokens from the user on the origin chain and releases the equivalent amount on the target chain.

7.1.2 Enhancing Bridges with mev-commit

Both types of bridges encounter similar challenges in achieving instant transfers, primarily due to the time required for transaction confirmation or finalization on the origin chain. Integrating mev-commit with the bridging process allows "bridge-initiating" transactions on the origin chain to be preconfirmed through mev-commit. This approach minimizes the risks of the transaction not be included in a block, or even rolled back due to a reorg, enabling nearly immediate fund release on the target chain. Typically, in wrapped asset bridges, the bridger[7] is the bridge operator, while in liquidity pool bridges, it is a solver.

Our mechanism for instant bridging is very simple and it involves the following two steps:

- *Preconfirmation via mev-commit*: When a user initiates a bridge transaction on the origin chain, the bridger employs mev-commit to bid for transaction inclusion in the subsequent block. mev-commit providers (i.e., block builders) offer preconfirmations, ensuring either inclusion of the transaction in the next block or financial compensation.
- *Instant release on target chain*: Upon receiving a preconfirmation from mev-commit, the bridger can release the corresponding funds on the target chain immediately, bypassing the need for transaction inclusion confirmation on the origin chain. This enables a truly instant bridge experience for the user.

Risks, mitigations and costs. The primary risk for a bridger in any cross-chain transaction is that the bridge-initiating transaction on the origin chain may fail to be included in a block or be removed later due to a blockchain reorg. This uncertainty is why cross-chain bridges generally operate slowly—bridgers often await multi-block confirmations or even finality on the origin chain before releasing funds on the target chain. The reorg risk that mev-commit offsets was already discussed in Section 3 and it also applies here. Moreover, by setting a compensation amount proportional to the funds being bridged, the bridger can completely eliminate the risk of lost funds due to the transaction not being included in the chain.

Securing a commitment from a block builder involves costs, as block builders provide preconfirmations only if the bid is economically viable. While more complex preconfirmation bids, which restrict the block builder's flexibility, incur higher costs, inclusion preconfirmations

[7] We here refer to the entity bridging the assets (i.e., receiving funds on the origin chain and releasing funds on the target chain) as the *bridger*.

for bridging use cases are straightforward and relatively cost-effective, as they only require the block builder to reserve transaction space without imposing further constraints on block construction.

Current limitations. While this approach presents significant improvements, current limitations include:

- *Dependence on L1*: Preconfirmations are limited to Ethereum L1 transactions, constraining instant bridging to transfers initiated from Ethereum.
- *Target chain processing speed*: The overall bridging process can be delayed if the target chain processes transactions slowly. In such cases, the release transaction on the target chain will also be delayed. This is not a sever limitation in practice since most target chains have very fast block times.

7.2 Preconfirmations for Searchers

The simplest application of insured commitments are preconfirmations for searchers. There, a searcher is willing to pay extra for the fast feedback on the inclusion and execution of transactions. Examples include arbitrage between a decentralized exchange on chain and a centralized exchange, where the searcher would only execute the transaction on the centralized exchange if the on-chain transaction is executed. A preconfirmation for the on-chain transaction can thus speed up the overall process and reduce the risk of the searcher. Another use-case is the ability for a searcher to rapidly increase the bid amounts to ensure inclusion in the next block without overpaying. In Section 8.2 we discuss potential future uses of preconfirmations for searchers.

8 Conclusion and Future Work

8.1 Conclusion

In this whitepaper, we introduced the concept of insured commitments and presented an implementation of it through the mev-commit protocol—a decentralized framework enabling bidders to bid for the inclusion or execution of transactions in Ethereum and providers to commit to fulfilling the bids in the blocks they build. The protocol is designed to be fast, secure, and efficient. We proposed two novel cryptographic primitives, ARUBE and DPCOM, which enable the privacy features essential for insured commitments. ARUBE allows bidders to send encrypted bids to a subset of providers without revealing the recipient set, while DPCOM facilitates secure and private commitments, enabling bidders and providers to commit and later open commitments as needed. Additionally, we implemented a bid decay mechanism, enabling bidders to define the time-based decay of their bids, and discussed the economic incentives and penalties structured for providers and validators within the mev-commit protocol.

8.2 Future Work

There are several promising directions for future work, which each offer the potential to expand the functionality and economic value of insured commitments in decentralized ecosystems. We sketch some of these below.

Further decentralization. One area is to further decentralize the network, which includes implementing decentralized bridges as well as decentralizing the mev-commit chain consensus. Achieving this requires a careful balance between decentralization and performance, given the high bandwidth and low latency demands of the mev-commit chain.

Reduce trust required in oracle. The mev-commit protocol currently relies on a single oracle for commitment settlements. A future direction is to investigate ways to reduce the required trust assumptions. One possibility would be to employ a decentralized oracle system operated by multiple providers. Another option is to use new features such as the one proposed in RIP-7728 [16] that allow to directly read the state of the Ethereum chain from the mev-commit chain. This would allow the mev-commit chain to verify the commitments without relying on an external oracle. Finally, the oracle could be replaced by advanced cryptographic techniques such as SNARKs that actors can use to prove fulfillment or violation of commitments directly on the mev-commit chain.

Improved robustness and more analysis. We plan to investigate ways to improve the robustness of the builder attribution mechanism and reduce the required trust in the relay. We also plan to look into ways to further improve the benefits of the validator opt-in, e.g., by introducing different levels of opt-in, where the highest level also disallows self-built blocks. Additionally, we plan to improve the double-spending prevention mechanism using homomorphic encryption and zero-knowledge proofs. Furthermore, there are plans to conduct a game theoretic analysis of the overall system. This includes determining fair prices for insured commitments as well as a thorough analysis and potential improvement of the decay mechanism.

Conditional disclosure of transaction payload data. The transaction payload itself can be attached to a mev-commit bid and encrypted together with the bid, inheriting the bid's privacy guarantees. We plan to investigate ways to increase the flexibility of this mechanism such that certain aspects about the payload (e.g., the transaction signature) can be disclosed to certain parties under certain conditions, e.g., using signature-based witness encryption [17, 18]. This allows for bidders to sell or auction off different parts of the transaction payload, leading to granular order flow mechanisms.

Extending dual-phase commitments. Our dual-phase commitments allow two separate parties to open a commitment. However, there are situations in which additional providers need the ability to open a commitment, e.g., for validity checks or when multiple providers are involved in building a block. Toward achieving that, we are working on extending dual-phase commitments to allow for multiple parties to open a commitment.

Integrating other preconfirmation protocols. There are several leader-based preconfirmation protocols that focus on preconfirmations from proposers or dedicated preconfirmation gateways. We plan to investigate ways to integrate with these to obtain two benefits: First, these protocols could benefit from some mev-commit innovations, e.g., the cryptographic privacy guarantees and the unique decay mechanism to incentivize timely commitments. This could help to mitigate several limitation of existing protocols outlined by Oshitani [19]. Secondly, running both types of commitment protocols at the same time would improve the overall user experience and increase the economic efficiency.

Additional applications. Another direction is to explore additional applications of insured commitments. For example, synchronous commitments could be developed, where multiple actors commit to an operation, and execution proceeds only when all parties have made their commitments. This concept could, for instance, be leveraged to enhance the interoperability between rollups and L1 by ensuring coordinated actions across both systems. Another potential application is multi-block mev, aimed at generating new revenue streams by leveraging mev opportunities that span multiple blocks. Examples include repricing a Uniswap pool over several blocks as asset prices shift across domains, handling liquidations that persist over multiple blocks, and executing long/short positions or non-atomic flash loans. Finally, we plan to look

into state-lock preconfirmations where a searcher can get a preconfirmation for a certain state without specifying a transaction such that the searcher can later add any transaction to be executed on top of the preconfirmed state.

References

[1] Statista. Number of daily cryptocurrency transactions by type. https://www.statista.com/statistics/730838/number-of-daily-cryptocurrency-transactions-by-type/. Accessed: 2024-11-07.

[2] Vitalik Buterin. Proposer/block builder separation-friendly fee market designs. Ethereum Research, 2021. https://ethresear.ch/t/proposer-block-builder-separation-friendly-fee-market-designs/.

[3] Flashbots. Mev-boost in a nutshell. https://boost.flashbots.net/. Accessed on 2024-11-07.

[4] Christian Matt and Murat Akdeniz. Leaderless and leader-based preconfirmations. Ethereum Research, 2024. https://ethresear.ch/t/leaderless-and-leader-based-preconfirmations/.

[5] Justin Drake. Based preconfirmations. Ethereum Research, 2023. https://ethresear.ch/t/based-preconfirmations/.

[6] Yaonam. Solutions to the preconf fair exchange problem. Ethereum Research, 2024. https://ethresear.ch/t/solutions-to-the-preconf-fair-exchange-problem/.

[7] Murat Akdeniz. Preconfirmations: The fulfillment-delivery paradigm. Mirror, 2024. https://mirror.xyz/preconf.eth/sgcuSbd1jgaRXj9odSJW-_OlWIg6jcDREw1hUJnXtgI.

[8] Data Always. The silhouette of resilience. https://hackmd.io/@dataalways/resilience, 2024.

[9] Stacy Elliott. Ethereum beacon chain suffers longest blockchain 'reorg' in years. Decrpyt, https://decrypt.co/101390/ethereum-beacon-chain-blockchain-reorg. Accessed on 2024-11-08.

[10] Primev Team. mev-commit documentation, 2024. https://docs.primev.xyz/get-started/welcome-to-primev.

[11] Chris Meisl. Anatomy of a slot: The tumultuous 12 seconds during ethereum slots. Blocknative, 2023. https://www.blocknative.com/blog/anatomy-of-a-slot.

[12] Bernardo Magri and Christian Matt. Pioneering end-to-end privacy for intra-block commitments. Mirror, 2024. https://mirror.xyz/preconf.eth/iz2J0uTXHhl8DiAkG-VLLwvCp-8qcc_Z7A8_4rU0A3g.

[13] Adam Barth, Dan Boneh, and Brent Waters. Privacy in encrypted content distribution using private broadcast encryption. In Giovanni Di Crescenzo and Avi Rubin, editors, *FC 2006*, volume 4107 of *LNCS*, pages 52–64. Springer, Berlin, Heidelberg, February / March 2006.

[14] Benoît Libert, Kenneth G. Paterson, and Elizabeth A. Quaglia. Anonymous broadcast encryption: Adaptive security and efficient constructions in the standard model. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 206–224. Springer, Berlin, Heidelberg, May 2012.

[15] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 566–582. Springer, Berlin, Heidelberg, December 2001.

[16] Haichen Shen and Péter Garamvölgyi. Precompile for L1SLOAD. Rollup Improvement Proposal, 2024. https://github.com/ethereum/RIPs/blob/master/RIPS/rip-7728.md.

[17] Nico Döttling, Lucjan Hanzlik, Bernardo Magri, and Stella Wohnig. McFly: Verifiable encryption to the future made practical. In Foteini Baldimtsi and Christian Cachin, editors, *FC 2023, Part I*, volume 13950 of *LNCS*, pages 252–269. Springer, Cham, May 2023.

[18] Gennaro Avitabile, Nico Döttling, Bernardo Magri, Christos Sakkas, and Stella Wohnig. Signature-based witness encryption with compact ciphertext. Cryptology ePrint Archive, Paper 2024/1477, 2024. https://eprint.iacr.org/2024/1477.

[19] Lin Oshitani. Strawmanning based preconfirmations. Ethereum Research, 2024. https://ethresear.ch/t/strawmanning-based-preconfirmations/.