

# Fast and Furious Withdrawals from Optimistic Rollups

Mahsa Moosavi ✉ 🏠 📧

Concordia University, Montreal, QC, Canada

OffchainLabs, United States

Mehdi Salehi ✉

OffchainLabs, United States

Daniel Goldman ✉

OffchainLabs, United States

Jeremy Clark ✉ 🏠 📧

Concordia University, Montreal, QC, Canada

---

## Abstract

Optimistic rollups are in wide use today as an opt-in scalability layer for blockchains like Ethereum. In such systems, Ethereum is referred to as L1 (Layer 1) and the rollup provides an environment called L2, which reduces fees and latency but cannot instantly and trustlessly interact with L1. One practical issue for optimistic rollups is that trustless transfers of tokens and ETH, as well as general messaging, from L2 to L1 is not finalized on L1 until the passing of a dispute period (aka withdrawal window) which is currently 7 days in the two leading optimistic rollups: *Arbitrum* and *Optimism*. In this paper, we explore methods for sidestepping the dispute period when withdrawing ETH from L2 (called an exit), even in the case when it is not possible to directly validate L2. We fork the most-used rollup, *Arbitrum Nitro*, to enable exits to be traded on L1 before they are finalized. We also study the combination of tradeable exits and prediction markets to enable insurance for withdrawals that do not finalize. As a result, anyone (including contracts) on L1 can safely accept withdrawn tokens while the dispute period is open despite having no knowledge of what is happening on L2. Our scheme also allows users to opt-into a fast withdrawal at any time. All fees are set by open market operations.

**2012 ACM Subject Classification** Security and privacy; Security and privacy → Cryptography

**Keywords and phrases** Ethereum, layer 2, rollups, bridges, prediction markets

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2023.22

**Supplementary Material** *Software (Source Code)*: <https://github.com/MadibaGroup/nitro/tree/fast-withdrawals>

**Funding** *Jeremy Clark*: acknowledges support for this research project from (i) the National Sciences and Engineering Research Council (NSERC), Raymond Chabot Grant Thornton, and Catalaxy Industrial Research Chair in Blockchain Technologies (IRCPJ/545498-2018), (ii) the Autorité des Marchés Financiers, and (iii) a NSERC Discovery Grant (RGPIN/04019-2021).

**Acknowledgements** This paper includes useful comments from the reviewers, discussions with Edward W. Felten and Rachel Bousfield, and feedback from presentations at Devcon 6 and a16z crypto research.


## 1 Introductory Remarks

Ethereum-compatible blockchain environments, called Layer 2s (or L2s) [5], have demonstrated an ability to reduce transaction fees by 99–99.9% while preserving the strong guarantees of integrity and availability in the underlying Layer 1 (or L1) blockchain. The subject of this paper concerns one subcategory of L2 technology called an optimistic rollup. The website *L2*

 © Mahsa Moosavi, Mehdi Salehi, Daniel Goldman, and Jeremy Clark; licensed under Creative Commons License CC-BY 4.0

5th Conference on Advances in Financial Technologies (AFT 2023).

Editors: Joseph Bonneau and S. Matthew Weinberg; Article No. 22; pp. 22:1–22:18

 Leibniz International Proceedings in Informatics  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

44 *Beat* attempts to capitalize all tokens of known value across the top 25 L2 projects. It finds  
 45 that the top two L2s are both optimistic rollups, *Arbitrum* and *Optimism*, which respectively  
 46 account for 50% and 30% of all L2 value—\$4B USD at the time of writing.<sup>1</sup>

47 We will describe the working details of optimistic rollups later in this paper but here are  
 48 the main takeaways: currently, rollups are faster and cheaper than Ethereum itself. However,  
 49 each L2 is essentially an isolated environment that cannot instantly and trustlessly interact  
 50 with accounts and contracts that are running on either L1 or other L2s. An optimistic  
 51 rollup project will typically provide a smart contract, called a validating bridge [9], that can  
 52 trustlessly move ETH (and other tokens and even arbitrary messages) between L1 and its  
 53 own L2. It implements a transfer by locking the ETH in an L1 contract and minting the  
 54 equivalent ETH on L2 and assigning it to the user’s L2 address. More precisely, L2 ETH is a  
 55 transferrable claim for L1 ETH from the L1 bridge at the request of the current owner of  
 56 the L2 claim. Later when the user requests a withdrawal, the ETH will be destroyed on L2  
 57 and released by the bridge back onto L1 according to whom its new owner is on L2 at the  
 58 time of the request. This requires the rollup to convince the L1 bridge contract of whom the  
 59 current owner of withdrawn ETH is on L2. We provide details later but this process takes  
 60 time: the bridge has to wait for a period of time called the dispute window. The current  
 61 default is 7 days in *Arbitrum* and *Optimism*, however the filing of new disputes can extend  
 62 the window. The bottom line is that users have to wait at least 7 days to draw down ETH  
 63 from an optimistic rollup.

## 64 Contributions.

65 In this paper, we compare several methods—atomic swaps and tradeable exits—for working  
 66 around this limitation. While we argue workarounds cannot be done generally (*e.g.*, for  
 67 NFTs, function outputs, or arbitrary messages), some circumstances allow it: namely, when  
 68 the withdrawn token is liquid, fungible, and available on L1 and the withdrawer is willing to  
 69 pay a fee to speed up the withdrawal. While these techniques work easily between human  
 70 participants that have off-chain knowledge, such as the valid state of the L2, it is harder to  
 71 make them compatible with L1 smart contracts that have no ability to validate the state  
 72 of L2. We propose a solution using tradeable exits and prediction markets to enable an  
 73 L1 smart contract to safely accept withdrawn tokens before the dispute period is over. We  
 74 fork the current version, *Nitro*, of the most used optimistic rollup, *Arbitrum*, maintained as  
 75 open source software<sup>2</sup> by *Offchain Labs*. *Arbitrum* is a commercial product with academic  
 76 origins [8]. We implement our solution and provide measurements. We also provide an  
 77 analysis of how to price exits and prediction market shares.

## 78 **2** Background

79 While we describe optimistic rollups as generally as possible, some details and terms are  
 80 specific to *Arbitrum*.

### 81 2.1 Inbox

82 Rollups have emerged as a workable approach to reduce fees and latency for Ethereum-based  
 83 decentralized applications. In a rollup, transactions to be executed on L2 are recorded in

<sup>1</sup> L2 Beat: <https://l2beat.com/scaling/tvl/>, accessed Oct. 2022.

<sup>2</sup> GitHub: Nitro <https://github.com/OffchainLabs/nitro>

84 an L1 smart contract called the inbox. Depending on the system, users might submit to  
85 the inbox directly, or they might submit to an offchain service, called a sequencer, that will  
86 batch together transactions from many users and pay the L1 fees for posting them into the  
87 inbox. Transactions recorded in the inbox (as `calldata`) are not executed on Ethereum,  
88 instead, they are executed in a separate environment off the Ethereum chain, called L2. This  
89 external environment is designed to reduce fees, increase throughput, and decrease latency.

## 90 2.2 Outbox

91 Occasionally (*e.g.*, every 30–60 minutes), validators on L2 will produce a checkpoint of the  
92 state of all contracts and accounts in the complete L2 according to the latest transactions  
93 and will place this asserted state (called an RBlock) in a contract on L1 called the outbox.  
94 Note that anyone with a view of L1 can validate that the sequence of transactions recorded  
95 in the inbox produces the asserted RBlock in the outbox. This includes Ethereum itself, but  
96 asking it to validate this be equivalent to running the transactions on Ethereum. The key  
97 breakthrough is that the assertion will be posted with *evidence* that the RBlock is correct so  
98 Ethereum does not have to check completely.

## 99 2.3 Optimistic vs. zk-rollups

100 In practice, two main types of evidence are used. In zk-rollups,<sup>3</sup> a succinct computational  
101 argument that the assertion is correct is posted and can be checked by Ethereum for far  
102 less cost than running all of the transactions. However the proof is expensive to produce.  
103 In optimistic rollups, the assertions are backed by a large amount of cryptocurrency acting  
104 as a fidelity bond. The correctness of an RBlock can be challenged by anyone on Ethereum  
105 and Ethereum itself can decide between two (or more) RBlocks for far less cost than running  
106 all of the transactions (by having the challengers isolate the exact point in the execution  
107 trace where the RBlocks differ). It will then reallocate the fidelity bonds to whoever made  
108 the correct RBlock. If an RBlock is undisputed for a window of time (*e.g.*, 7 days), it is  
109 considered final.

## 110 2.4 Bridge

111 A final piece of the L2 infrastructure is a bridge, which can move ETH, tokens, NFTs, and  
112 even arbitrary messages, between L1 and L2. Our fast withdrawals is limited to ETH and  
113 fungible tokens. If Alice has ETH on Ethereum, she can submit her ETH to a bridge smart  
114 contract on Ethereum which will lock the ETH inside of it, while generating the same amount  
115 of ETH in Alice's account inside the L2 environment. The bridge does not need to be trusted  
116 because every bridge operation is already fully determined by the contents of the inbox. Say  
117 that Alice transfers this ETH to Bob's address on L2. Bob is now entitled to draw down the  
118 ETH from L2 to L1 by submitting a withdrawal request using the same process as any other  
119 L2 transaction—*i.e.*, placing the transaction in the inbox on L1, having it executed on L2,  
120 and seeing it finalized in an RBlock on L1. Optimistically, the RBlock is undisputed for 7  
121 days and is finalized. Bob can now ask the bridge on L1 to release the ETH to his address  
122 by demonstrating his withdrawal (called an exit) is included in the finalized RBlock (*e.g.*,  
123 with a Merkle-proof).

---

<sup>3</sup> zk stands for zero-knowledge, a slight misnomer: succinct arguments of knowledge that only need to be complete and sound, not zero-knowledge, are used [10].

## 124 2.5 Related Work

125 Arbitrum is first described at *USENIX Security* [8]. Gudgeon *et al.* provide a systemization  
 126 of knowledge (SoK) of Layer 2 technology (that largely predates rollups) [5]. McCorry  
 127 *et al.* provide an SoK that covers rollups and validating bridges [9], while Thibault *et al.*  
 128 provide a survey specifically about rollups [13]. Some papers implement research solutions  
 129 on Arbitrum for improved performance: decentralized order books [11] and secure multiparty  
 130 computation [2]. The idea of tradeable exits predates our work but is hard to pinpoint a  
 131 source (our contribution is implementation and adding hedges). Further academic work on  
 132 optimistic rollups and bridges is nascent—we anticipate it will become an important research  
 133 area.

134 Other related topics are atomic swaps and prediction markets. Too many papers propose  
 135 atomic swap protocols to list here but see Zamyatin *et al.* for an SoK of the area (and a  
 136 new theoretical result) [14]. Decentralized prediction markets proposals predate Ethereum  
 137 and include Clark *et al.* [1] and Truthcoin [12]. Early Ethereum projects *Augur* and *Gnosis*  
 138 began as prediction markets.

## 139 3 Proposed Solution

140 For simplicity, we will describe a fast exit system for withdrawing ETH from L2, however it  
 141 works for any L1 native fungible token (*e.g.*, ERC20) that is available for exchange on L1. We  
 142 discuss challenges of fast exits for non-liquid/non-fungible tokens in Section 6.4. Consider an  
 143 amount of 100 ETH. When this amount is in the user’s account on L1, we use the notation  
 144  $100 \text{ ETH}_{L1}$ . When it is in the bridge on L1 and in the user’s account on L2, we denote it  $100$   
 145  $\text{ETH}_{L2}$ . When the ETH has been withdrawn on L2 and the withdrawal has been asserted  
 146 in the L1 outbox, but the dispute window is still open, we refer to it as  $100 \text{ ETH}_{XX}$ . Other  
 147 transitionary states are possible but not needed for our purposes.

### 148 3.1 Design Landscape

149 In Table 1, we compare our solution to alternatives in industry and the blockchain (academic  
 150 and grey) literature that could be used for fast withdrawals.

#### 151 3.1.1 Properties

152 We are interested in solutions that do not require a trusted third party. If trust is acceptable,  
 153 a centralized exchange that has custody of its users funds is a fast and user-friendly solution.  
 154 We consider anything faster than the 7-day dispute period as “fast” but take measurements  
 155 of solutions that can settle within a fully confirmed “L1 transaction” (*e.g.*, minutes) and  
 156 within a unconfirmed L2 RBlock (*e.g.*, hours). This assumes that all counterparties perform  
 157 instantly upon request. Settlement is from the perspective of the withdrawer, Alice, only  
 158 and does not necessarily mean other counterparties will complete within the same timeframe.  
 159 For example, in many solutions, Alice will have her withdrawn ETH quickly at the expense  
 160 of a counterparty waiting out the dispute period.

161 Some solutions require one party to act, followed by an action of the counterparty in a  
 162 follow-up transaction. This creates the risk that the counterparty aborts the protocol before  
 163 taking their action. Since it is unknown if the counterparty will act or not, these protocols  
 164 establish a window of time for the counterparty to act and if the window passes without action,  
 165 the initial party has to begin the protocol again with a new counterparty. The protocols  
 166 ensure that funds are never at risk of being lost, stolen, or locked up forever, however the

Type	Example	No trusted third party	Within an L1 transaction	Within an L2 rollup	No grieving	No free option	Opt-in anytime	Crosschain or L2-to-L2	L1 gasUsed	L2 gasUsed	Other Fees
Normal Exit (baseline)	Arbitrum	•		•	•				200K	80K	—
Centralized	Binance		•	•	•	•	•		400K	21K	Operator
HTLC Swaps	Celer	•	◦	•			•		625K	92K	
Conditional Transfers	StarkEx	•	•	•				⊥	⊥		Operator
Bridge Tokens	Hop	◦	•	•		•	•		1.8M	300K	Operator
Tradeable Exits	This Work	•	~	•	•	•	•		200K	80K	Discount
Hedged Tradeable Exits	This Work	•	~	•	•	•	•		265K	80K	FAIL <sub>PM</sub>

■ **Table 1** Comparing alternatives for fast withdrawals from optimistic rollups for liquid and fungible tokens where • satisfies the property fully, ◦ partially satisfies the property, and no dot means the property is not satisfied. ⊥ was not measured. For our work, ~ means we propose how to fully achieve the property but do not by default (see caveats in Section 6.1).

167 protocols admit two smaller issues. The first issue is that a malicious counterparty could  
 168 accept to participate with no intention of completing the protocol just to “grief” the party  
 169 taking the action—wasting their time and possibly gas fees for setting up and tearing down  
 170 the conditions of the trade. The second issue is that a strategic counterparty can accept to  
 171 participate and then selectively choose to complete or abort, as well as timing exactly when  
 172 they choose to complete (within the window), based on price movements or other market  
 173 information. This is called (somewhat cryptically) a “free option;” finance people might  
 174 recognize it as akin to being given an American call option for free.

175 A solution is “opt-in anytime” if the user can withdraw normally and then (say upon  
 176 realizing for the first time that there is a 7 day dispute window) decide to speed up their  
 177 transaction. While it is not a design goal of our paper, many of these solutions are generic  
 178 cross-chain transactions (including L2-to-L2 swaps). A drawback of our solution is that it is  
 179 narrowly scoped to L2-to-L1 withdrawals on rollups. Therefore our solution is not intended  
 180 as a complete replacement of atomic swaps or the other solutions in Table 1. It is designed  
 181 to be best-in-class only for slow rollup withdraws.

182 Finally we estimate the costs involved for the seller of ETH<sub>L2</sub>. For some protocols, the  
 183 gas cost of the buyer might differ from the seller depending if its actions are symmetric or  
 184 not—we comment on this but did not find it interesting enough to put in the table. The  
 185 more interesting aspect is that many alternatives do require a third party to be involved  
 186 (we generically call them “operators”) and they must be compensated for their actions. In  
 187 some alternatives, the operators might be not be inherently necessary (e.g., an HTLC swap)  
 188 but are used in practice (e.g., Celer) to ease friction (e.g., users finding other users to swap  
 189 with): in this case, we are charitable and do not mark the fee. So the fees are for things  
 190 fundamental to how the alternative works. We expand more within the discussion of each  
 191 alternative below.

192 **3.1.2 Alternatives**193 **Centralized**

194 Consider Alice who has 100  $\text{ETH}_{L2}$  and wants 100  $\text{ETH}_{L1}$  for it. A centralized exchange (*e.g.*,  
 195 *Coinbase, Binance*) can open a market for  $\text{ETH}_{L2}/\text{ETH}_{L1}$ . Alternatively, a bridge might rely  
 196 on an established set of trustees to relay L2 actions to L1. This is called proof of authority;  
 197 it is distributed but not decentralized (*i.e.*, not an *open* set of participants). The gas costs  
 198 consists of Alice transferring her  $\text{ETH}_{L2}$  onto the exchange (withdraw to L1 is paid for by the  
 199 exchange). An exchange will not be profitable if it offers this for free, therefore it captures a  
 200 operator fee for the service.

201 **Hash Time Locked Contracts (HTLCs)**

202 Assume Bob has 100  $\text{ETH}_{L1}$  and is willing to swap with Alice. An atomic swap binds together  
 203 (i) an L2 transaction moving 100  $\text{ETH}_{L2}$  from Alice to Bob and (ii) an L1 transaction moving  
 204 100  $\text{ETH}_{L1}$  from Bob to Alice. Either both execute or both fail. HTLC is a blockchain-friendly  
 205 atomic swap protocol. Its main drawback is that it also has a time window where Alice  
 206 (assuming she is the first mover in the protocol) must wait on Bob, who might abort causing  
 207 Alice's  $\text{ETH}_{L2}$  to be locked up while waiting (called the griefing problem), or might watch  
 208 price movements before deciding to act (called free option problem). Bob needs to monitor  
 209 both chains so he cannot be an autonomous smart contract. HTLCs can work generically  
 210 between any two chains capable of hash- and time-locking transaction outputs; this includes  
 211 between two L2s.

212 The transaction (containing a hashlock and timeout) is slightly more complicated than  
 213 a standard ETH transfer, requiring smart contract logic on both layers. The measurement  
 214 based on Celer is not a pure HTLC and uses operators as well for liquidity and staking, but  
 215 we omit these fees from the table because theoretically Alice and Bob could find each other  
 216 and perform a pure HTLC with no added infrastructure.

217 **Conditional Transfers**

218 The intuition behind a conditional transfer (CT) is that L1-to-L2 messaging (or bridging)  
 219 is fast even if L2-to-L1 messaging is slow. CT exploits this to build an HTLC-esque swap  
 220 specifically for withdrawing from rollups (while HTLCs are designed generically for cross-  
 221 chain swaps). Alice begins by registering her intent to trade 100  $\text{ETH}_{L2}$  for 100  $\text{ETH}_{L1}$  in  
 222 a special registry contract on L1, and she locks (*e.g.*, for an hour) 100  $\text{ETH}_{L2}$  in escrow  
 223 on L2. If Bob agrees to the swap, Alice provides him (off-chain) with a signed transaction  
 224 (called the conditional transfer) that transfers the escrowed 100  $\text{ETH}_{L2}$  to Bob, conditioned  
 225 on Alice having receiving 100  $\text{ETH}_{L1}$  in the registry contract on L1. After Bob transfers the  
 226  $\text{ETH}_{L1}$  on L1, this fact can be bridged to the L2 escrow contract (with customization of the  
 227 rollup's inbox) quickly (recall that L1-to-L2 messaging is fast). The L2 escrow contract will  
 228 flag that the L1 transaction has paid by Bob, and Bob can broadcast his signed (by Alice)  
 229 L2 transaction to recover 100  $\text{ETH}_{L2}$  from escrow (if Bob broadcasts it before the flag is set,  
 230 it simply reverts).

231 In terms of existing implementations, we could not adequately isolate the conditional  
 232 transfer component from the rest of the bridge to measure gas costs (denoted in the table  
 233 using a  $\perp$  symbol) however it should be slight more expensive than an HTLC as the logic of  
 234 the transaction is more complex.

235 Also note that Bob must be a validator on L2 to confirm that the state of the escrow and  
236 conditional transfer on L2 will result in him being paid—this is where the speedup really  
237 comes from, if he waits for L1 to finalize this, then the transfer happens after the dispute  
238 period and it is no different than a normal exit. Consequently, Bob cannot be an autonomous  
239 L1 smart contract unable to validate L2 state until it is finalized on L1 (which is the design  
240 goal of our alternative: hedged tradeable exits).

## 241 Bridge Token

242 A bridge token is not a novel technical innovation but it is a practical market design for  
243 supplying bridges with liquidity. Bridges between L1 and L2 can technically be implemented  
244 by anyone. It is natural for the inbox/outbox provider to provide a bridge but it is not  
245 strictly necessary.

246 Assume a third party creates a contract on L1 that accepts  $\text{ETH}_{L1}$  and releases a  
247 transferable claim for  $\text{ETH}_{L1}$ ; it creates the same contract on L2. Assume enough of these  
248 claims come into circulation that a liquid market for them emerges on both layers. To move  
249  $\text{ETH}_{L2}$  to  $\text{ETH}_{L1}$ , Alice starts by trading her  $\text{ETH}_{L2}$  for a claim to the same amount on  
250 L2. She then asks the L2 contract to transfer the claim which it does by burning them and  
251 firing an event. An authorized party, called a bonder, notices the event on L2, goes to the L1  
252 contract and mints the same number of claims on L1 for  $\text{ETH}_{L1}$  and transfers them to Alice's  
253 address. Technically the L1 contract is insolvent as more claims exist than actual  $\text{ETH}_{L1}$  in  
254 the contract, but the L2 contract is oversolvent by the same amount. The contracts can be  
255 rebalanced (1) through movements in the opposite direction; (2) through a bulk withdrawal  
256 after the normal 7-day dispute period; or (3) by incentivize bonders to purposefully rebalance  
257 the contracts by burning claims on L1 and minting on L2. To prevent the bonder from  
258 maliciously minting tokens on L1 that were not burned on L2, it must post a fidelity bond of  
259 equal or greater value. (Alternatively, the bonder can be a trusted party which makes it the  
260 same in analysis as a centralized exchange). After the 7-day dispute period, the L1 contract  
261 can verify the bonder's actions are consistent with the burns on L2 and release its fidelity  
262 bond.

263 Note that when you collapse this functionality, it is equivalent to the bonder buying  
264  $\text{ETH}_{XX}$  from Alice for  $\text{ETH}_{L1}$  and receiving their  $\text{ETH}_{L1}$  back 7 days later. The extra  
265 infrastructure is necessary because today native bridges do not support transferring  $\text{ETH}_{XX}$ .  
266 As in atomic swaps, the bonder can fail to act (griefing) which is worst in this case if Alice  
267 cannot 'unburn' her tokens, but there is no free option because Bob is a relay and not  
268 a recipient of the tokens. The gas fee measurement is based on Hop and standard token  
269 transfers on L1 and L2. The main cost of bridge tokens is paying the bonder (called an  
270 operator in the table) who are providing a for-profit service.

## 271 3.2 Tradeable Exits

272 Alice wants to withdraw 100  $\text{ETH}_{L2}$ . Unlike the other solutions, Bob takes the risk that  
273 the exit never finalized and therefore will offer less than 100  $\text{ETH}_{L1}$  (say 99.95  $\text{ETH}_{L1}$ ) for  
274 it (this is denoted "discount" in Table 1). Assume Bob has 99.95  $\text{ETH}_{L1}$  that will not use  
275 until after the dispute window. Bob also runs an L2 validator so he is assured that if Alice  
276 withdraws, it is valid and will eventually finalize. With a tradeable exit, the outbox allows  
277 Alice to change the recipient of her withdraw from herself to Bob. Thus Alice swaps her  
278 pending exit of 100  $\text{ETH}_{L1}$  (which we call 100  $\text{ETH}_{XX}$ ) for Bob's 99.95  $\text{ETH}_{L1}$  on L1 (note  
279 we discuss the actual difference in price in Section 5). Since  $\text{ETH}_{L1}$  and  $\text{ETH}_{XX}$  are both on

280 L1, Alice can place an ask price for her  $\text{ETH}_{\text{XX}}$  and the first trader willing to swap can do  
 281 so atomically, with no ability to grieve or capitalize on a free option. After 7 days, Bob can  
 282 ask the bridge to transfer the  $\text{ETH}_{\text{L1}}$  to his address, and the bridge checks the outbox to  
 283 validate that Bob’s address is the current owner of the exit.

284 In our forked bridge, Alice can transfer any of her exits that are in an RBlock (*i.e.*, an  
 285 asserted L2 state update registered in the outbox). Technically, Bob can check the validity  
 286 of the withdrawal as soon as it is in the inbox, and not wait 30-60 minutes for an RBlock.  
 287 However for implementation reasons, it is easier to track an exit based on its place (*i.e.*,  
 288 Merkle path) in an RBlock, rather than its place in the inbox. When we say a withdrawal is  
 289 ‘fast,’ we mean 30-60 minutes (*i.e.*, one L2 rollup).

290 Tradeable exits can be approximated by a third party L1 contract that does not modify  
 291 the rollup. In this scenario, a L1 contract would act like a proxy for the exit. Alice would  
 292 specify that she is exiting 100  $\text{ETH}_{\text{L2}}$  to the proxy contract address (instead of to her address)  
 293 and set the proxy contract to forward it to her address (if/when it comes through after 7  
 294 days). Before the dispute window closes, she can sign a transaction instructing the proxy  
 295 contract to forward the exit to Bob instead of to her (while giving Bob signing authority  
 296 over it). In this way, the exit becomes tradeable. After 7 days, the current owner can ask  
 297 the proxy to fetch the actual transfer from the bridge and forward it to them. If the exit  
 298 fails, the bridge will refuse the exit.

299 Given this option, why modify the bridge/outbox of the rollup? This paper is not  
 300 intended as a strong endorsement of either approach—the reader can decide between the two  
 301 approaches. Our intention with this research is to discuss, design, implement, and measure  
 302 the actual functionality of what is needed. This will be largely the same whether it is placed  
 303 inside or outside the bridge/outbox. The main advantage of modifying the bridge/outbox  
 304 is that is backward compatible with existing web3 bridge interfaces and with current user  
 305 behaviour—if web3 interfaces or users do a slow withdraw, our solution can “bail them out”  
 306 after the fact. Placing the functionality inside the bridge/outbox is more challenging in  
 307 some regards (*e.g.*, existing code is complex to understand) but also easier in other regards  
 308 (*e.g.*, our code has direct access to state variables). An outside contract might require minor  
 309 changes to the bridge anyways, such as creating public interfaces to state variables or other  
 310 data (*e.g.*, as one example, we later discuss how a prediction market must be able to query  
 311 the outbox to know if an RBlock is pending, finalized, or failed, which is not a current  
 312 feature). By contrast, the main advantage of an outside contract is modularity and reducing  
 313 complexity (and thus risk) within the bridge.

### 314 3.3 Hedged Tradeable Exits

315 One remaining issue with tradeable exits is how specialized Bob is: he must have liquidity in  
 316  $\text{ETH}_{\text{L1}}$  (or worst, every token being withdrawn from L2), be online and active, know how to  
 317 price derivatives, and be a L2 validator. While we can expect blockchain participants with  
 318 each specialization, it is a lot to assume of a single entity. The goal of this subsection is to  
 319 split Bob into two distinct participants: Carol and David. Our goal is to allow Carol who  
 320 does not (or functionally cannot) know anything about L2’s current state to safely accept a  
 321 tradeable exit as if it were equivalent to finalized  $\text{ETH}_{\text{L1}}$  (or L1 tokens). Carol could be a  
 322 L1 contract that accepts the withdrawn tokens for a service or enables exchange. In order  
 323 to make Carol agnostic of L2, we need David to be aware of L2: David is a L2 validator  
 324 who understands the risks of an RBlock failing and is willing to bet against it happening.  
 325 Therefore David needs to also have some liquidity to bet with however it could be  $\text{ETH}_{\text{L1}}$  or  
 326 a stablecoin, while Alice and Carol can interact with all sorts of tokens that David need not



327 heard of or even ones David would not want to hold himself.

328 Recall that Alice wants  $\text{ETH}_{L1}$  quickly in order to do something on L1 with it; Carol  
 329 can be that destination contract. The primary risk for Carol accepting  $\text{ETH}_{XX}$  as if it were  
 330  $\text{ETH}_{L1}$  is that the RBlock containing the  $\text{ETH}_{XX}$  withdrawal fails and the exit is worthless.  
 331 If Alice can obtain insurance for the  $\text{ETH}_{XX}$  that can be verified via L1, then Carol's risk  
 332 is hedged and she could accept  $\text{ETH}_{XX}$ . The insurance could take different forms but we  
 333 propose using a prediction market.

### 334 Prediction markets

335 A decentralized prediction market is an autonomous (*e.g.*, vending machine-esque) third  
 336 party contract. Since we are insuring L1  $\text{ETH}_{XX}$ , we need to run the market on L1 (despite  
 337 the fact that it would be cheaper and faster on L2). Consider a simple market structure  
 338 based on [1]. A user can request that a new market is created for a given RBlock. The market  
 339 checks the outbox for the RBlock and its current status (which must be pending). Once  
 340 opened, any user can submit 1  $\text{ETH}_{L1}$  (for example, the actual amount would be smaller  
 341 but harder to read) and receive two 'shares': one that is a bet that the RBlock will finalize,  
 342 called  $\text{FINAL}_{PM}$ , and one that is a bet that the RBlock will fail, called  $\text{FAIL}_{PM}$ . These shares  
 343 can be traded on any platform. At any time while the prediction market is open, any user  
 344 can redeem 1  $\text{FINAL}_{PM}$  and 1  $\text{FAIL}_{PM}$  for 1  $\text{ETH}_{L1}$ . Once the dispute period is over, any  
 345 user can request that the market close. The market checks the rollup's outbox for the status  
 346 of the RBlock—since both contracts are on L1, this can be done directly without oracles or  
 347 governance. If the RBlock finalizes, it offers 1  $\text{ETH}_{L1}$  for any 1  $\text{FINAL}_{PM}$  (and conversely if  
 348 it fails). The market always has enough  $\text{ETH}_{L1}$  to fully settle all outstanding shares.

349 It is argued in the prediction market literature [1] that (i) the price of one share matches  
 350 the probability (according to the collective wisdom of the market) that its winning condition  
 351 will occur, and (ii) the price of 1  $\text{FINAL}_{PM}$  and 1  $\text{FAIL}_{PM}$  will sum up to 1  $\text{ETH}_{L1}$ . For  
 352 example, if  $\text{FAIL}_{PM}$  trades for 0.001  $\text{ETH}_{L1}$ , then (i) the market believes the RBlock will fail  
 353 with probability of 0.1% and (ii)  $\text{FINAL}_{PM}$  will trade for 0.999  $\text{ETH}_{L1}$ . These arguments  
 354 do not assume market friction: if the gas cost for redeeming shares is  $D$  (for delivery cost),  
 355 both share prices will incorporate  $D$  (see Section 5). Lastly, prediction markets are flexible  
 356 and traders can enter and exit positions at any time—profiting when they correctly identify  
 357 over- or under-valued forecasts. This is in contrast to an insurance-esque arrangement where  
 358 the insurer is committed to hold their position until completion of the arrangement.

### 359 Hedging exits.

360 Given a prediction market, Alice can hedge 100  $\text{ETH}_{XX}$  by obtaining 100  $\text{FAIL}_{PM}$  as insurance.  
 361 Any autonomous L1 contract (Carol) should be willing to accept a portfolio of 100  $\text{ETH}_{XX}$   
 362 and 100  $\text{FAIL}_{PM}$  as a guaranteed delivery of 100  $\text{ETH}_{L1}$  after the dispute period, even if  
 363 Carol cannot validate the state of L2.

364 Perhaps surprisingly, this result collapses when withdrawing  $\text{ETH}_{L2}$ —consider Path 1  
 365 through the protocol. Alice withdraws 100  $\text{ETH}_{L2}$  from L2 and obtains 100  $\text{ETH}_{XX}$ . Bob  
 366 creates 100  $\text{FAIL}_{PM}$  and 100  $\text{FINAL}_{PM}$  for a cost of 100  $\text{ETH}_{L1}$ . Alice buys 100  $\text{FAIL}_{PM}$  from  
 367 Bob for a small fee. Alice gives Carol 100  $\text{ETH}_{XX}$  and 100  $\text{FAIL}_{PM}$  and is credited as if she  
 368 deposited 100  $\text{ETH}_{L1}$ . In seven days, Bob gets 100  $\text{ETH}_{L1}$  for his 100  $\text{FINAL}_{PM}$  and Carol  
 369 gets 100  $\text{ETH}_{L1}$  for her 100  $\text{ETH}_{XX}$ . If the RBlock fails, Bob has 0  $\text{ETH}_{L1}$  and Carol has 100  
 370  $\text{ETH}_{L1}$  from the 100  $\text{FAIL}_{PM}$ . In both cases, Alice has a balance of 100  $\text{ETH}_{L1}$  with Carol.

371 In path 2, Alice withdraws 100  $\text{ETH}_{L2}$  from L2 and obtains 100  $\text{ETH}_{XX}$ . Alice sells 100

372 ETH<sub>XX</sub> to Bob for 100 ETH<sub>L1</sub>. Alice gives Carol 100 ETH<sub>L1</sub> and is credited with a balance  
 373 of 100 ETH<sub>L1</sub>. In 7 days, Bob gets 100 ETH<sub>L1</sub> for his 100 ETH<sub>XX</sub> and Carol has 100 ETH<sub>L1</sub>.  
 374 If the RBlock fails, Bob has 0 ETH<sub>L1</sub>, Carol has 100 ETH<sub>L1</sub>, and Alice has a balance of 100  
 375 ETH<sub>L1</sub> with Carol.

376 Modulo differing gas costs and market transaction fees, paths 1 and 2 are equivalent.  
 377 Path 2 does not use a prediction market at all, it only uses basic tradeable exits. Given this,  
 378 do prediction markets add nothing to tradeable exits? We argue prediction markets still  
 379 have value for a few reasons. (1) Speculators will also participate in the prediction market  
 380 which gives Alice a chance for a fast exit even without Bob (an L2 validator). (2) If Alice  
 381 withdraws a token other than ETH, the prediction market should still be set up to payout in  
 382 ETH (otherwise you end up with 50 separate prediction markets for the 50 different kinds  
 383 of tokens in any given RBlock). In this case, Alice can obtain FAIL<sub>PM</sub> when Bob has no  
 384 liquidity or interest in the token she is withdrawing (however Carol needs to incorporate an  
 385 exchange rate risk when accepting an exit in one token and the insurance in ETH). (3) The  
 386 PM can also help with NFTs and other non-liquid tokens (see Section 6.4).

387 Three of the most common types of traders are utility traders, speculators, and dealers [6].  
 388 With a prediction market, Alice is a utility trader and Bob is a dealer. However, there might  
 389 exist speculators who want to participate in the market because they have forecasts about  
 390 rollup technology, a given RBlock, the potential for software errors in the rollup or in the  
 391 validator software, *etc.* Executives of rollup companies could receive bonuses in FINAL<sub>PM</sub>.  
 392 Quick validators might profit from noticing an invalid RBlock with FAIL<sub>PM</sub> or they might  
 393 be betting on an implementation bug or weeklong censorship of the network. Speculators  
 394 add liquidity to the prediction market which reduces transactional fees for Alice. However,  
 395 speculation also brings externalities to the rollup system where the side-bets on an RBlock  
 396 could exceed the staking requirements for posting an RBlock, breaking the crypto-economic  
 397 arguments for the rollup. In reality, these externalities can never be prevented in any  
 398 decentralized incentive-based system [3].

## 399 4 Implementation and Performance Measurements

400 We run *Arbitrum Nitro* test-net locally and use Hardhat [4] for our experiments. We obtain  
 401 our performance metrics using TypeScript scripts.

### 402 4.1 Tradeable Exits

#### 403 Trading the exit directly through the bridge/outbox.

404 We fork the *Arbitrum Nitro* outbox to add native support for tradeable exits. The modified  
 405 outbox is open source, written in 294 lines (SLOC) of Solidity, and a bytecode of 6,212 bytes  
 406 (increased by 1,197 bytes). The solidity code and Hardhat scripts are available in a GitHub  
 407 repository.<sup>4</sup> Our modifications include:

- 408 • Adding the `transferSpender()` function which allows the exit owner to transfer the exit  
 409 to any L1 address even though the dispute period is not passed.
- 410 • Adding the `isTransferred()` mapping which stores key-value pairs efficiently. The key  
 411 of the mapping is the exit number and the value is a boolean.

---

<sup>4</sup> GitHub:Nitro, Fast-Withdrawals: <https://github.com/MadibaGroup/nitro/tree/fast-withdrawals>

- 412 • Adding the `transferredToAddress` mapping which stores key-value pairs efficiently. The  
413 key of the mapping is the exit number and the value is the current owner of the exit.
- 414 • Modifying the `executeTransactionImpl()` function. Once the dispute period is passed  
415 and the withdrawal transaction is confirmed, anyone can call the `executeTransaction()`  
416 function from the outbox (which internally calls the `executeTransactionImpl()`) and  
417 release the funds to the account that was specified by the user 7 days earlier in the L2  
418 withdrawal request. With our modifications, this function is now enabled to release the  
419 requested funds to the current owner of the exit.

420 To execute the `transferSpender()` function; Alice (who has initiated a withdrawal for  
421 100 ETH<sub>L2</sub>) has to provide variables related to her exit (*e.g.*, exit number), which she can  
422 query using the Arbitrum SDK<sup>5</sup>, as well as the L1 address she wants to transfer her exit  
423 to. The `transferSpender()` function then checks (1) if the exit is already spent, (2) it is  
424 already transferred, and (3) the exit is actually a leaf in any unconfirmed RBlock. If the  
425 exit has been transferred, the `msg.sender` is cross-checked against the current owner of the  
426 exit (recall exit owners are tracked in the `transferredToAddress` mapping added to the  
427 outbox). Once these tests are successfully passed, the `transferSpender()` function updates  
428 the exit owner by changing the address in the `transferredToAddress` mapping. This costs  
429 85,945 units of L1 gas. Note that the first transfer always costs more as the user has to  
430 pay for initializing the `transferredToAddress` mapping. `transferSpender()` costs 48,810  
431 and 48,798 units of L1 gas for the second and third transfer respectively. The `gasUsed` for  
432 executing the new `executeTransactionImpl()` function is 91,418 units of L1 gas.

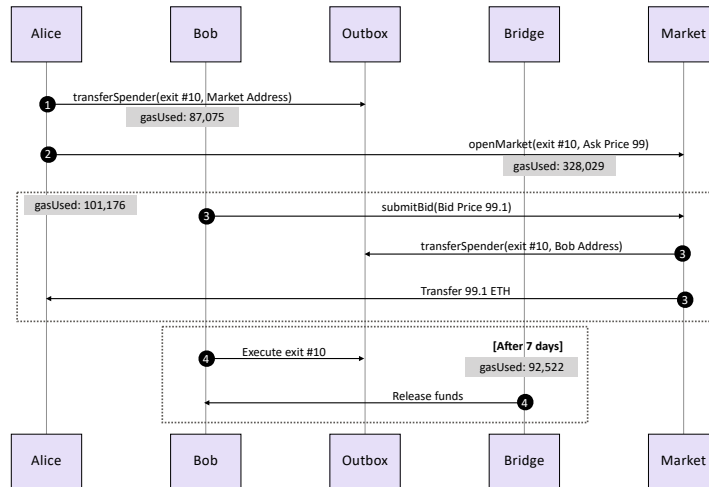
### 433 Trading the exit through an L1 market.

434 We also implement and deploy an L1 market that allows users to trade their exits on L1 even  
435 though the dispute window is not passed (see Section 6.3 for why *Uniswap* is not appropriate).  
436 In addition, we add a new function to the *Arbitrum Nitro* outbox, the `checkExitOwner()`,  
437 which returns the current owner of the exit. Figure 1 illustrates an overview of participant  
438 interactions and related gas costs. To start trading, Alice needs to lock her exit up in the  
439 market by calling the `transferSpender()` function from the outbox. Next, she can open a  
440 market on this exit by calling the `openMarket()` from the market contract and providing the  
441 ask price. The market checks if Alice has locked her exit (by calling the `checkExitOwner()`  
442 from the outbox) and only in that case a listing is created on this exit. The market would be  
443 open until a trade occurs or Alice calls the `closeMarket()` on her exit. Bob, who is willing  
444 to buy Alice’s exit, calls the payable `submitBid()` function from the market contract. If the  
445 `msg.value` is equal or greater than Alice’s ask price, the trade occurs; (1) the market calls  
446 the `transferSpender()` from the outbox providing Bob’s address. Note that market can  
447 only do that since it is the current owner of the exit being traded, and (2) the `msg.value` is  
448 transferred to Alice.

449 The market and modified outbox are open source and written in 125 and 294 lines (SLOC)  
450 of Solidity respectively. The solidity code for these contracts in addition to the Hardhat  
451 scripts are available in a GitHub repository.<sup>6</sup> Once deployed, the bytecode of the market  
452 and outbox is 5,772 and 6,264 bytes respectively.

<sup>5</sup> A typescript library for client-side interactions with Arbitrum.

<sup>6</sup> GitHub:Nitro, Fast-Withdrawals: <https://github.com/MadibaGroup/nitro/tree/fast-withdrawals>



■ **Figure 1** Overview of trading the exit through an L1 market.

## 4.2 Prediction Market

As described in Section 3.3, a prediction market can be used to hedge the exit. We do not implement this as one can use an existing decentralized prediction market (e.g., *Augur* or *Gnosis*). However, we further modify *Arbitrum Nitro* to make it friendly to a prediction market that wants to learn the status of an RBlock (pending, confirmed). More specifically, we modify the *Arbitrum Nitro* outbox and RollupCore smart contracts, modifications include:

- Adding the `assertionAtState` mapping to the outbox which stores key-value pairs efficiently. The key of the mapping is the exit number and the value is the user-defined data type `state` that restricts the variable to have only one of the `pending` and `confirmed` predefined values.
- Adding the `markAsPending` function to the outbox which accepts an RBlock and marks it as pending in the `assertionAtState` mapping.
- Adding the `markAsConfirmed` function to the outbox which accepts an RBlock and marks it as confirmed in the `assertionAtState` mapping.
- Modifying the `createNewNode()` function in the RollupCore contract. To propose an RBlock, the validator acts through the RollupCore contract by calling a `createNewNode()` function. We modify this function to call the `markAsPending()` from the outbox which marks the RBlock as pending.
- Modifying the `confirmNode()` function in the RollupCore contract. Once an RBlock is confirmed, the validator acts through the RollupCore contract via `confirmNode` to move the now confirmed RBlock to the outbox. We modify this function to call the `markAsConfirmed()` from the outbox which marks the RBlock as confirmed.

The modified outbox and RollupCore are open source and written in 297 and 560 lines (SLOC) of Solidity respectively. The solidity code for these contracts in addition to the

477 Hardhat scripts are available in a GitHub repository.<sup>7</sup> Once deployed, the bytecode of the  
478 outbox and RollupCore is 6,434 and 3,099 bytes respectively.

## 479 **5 Pricing**

### 480 **Pricing ETH<sub>XX</sub>.**

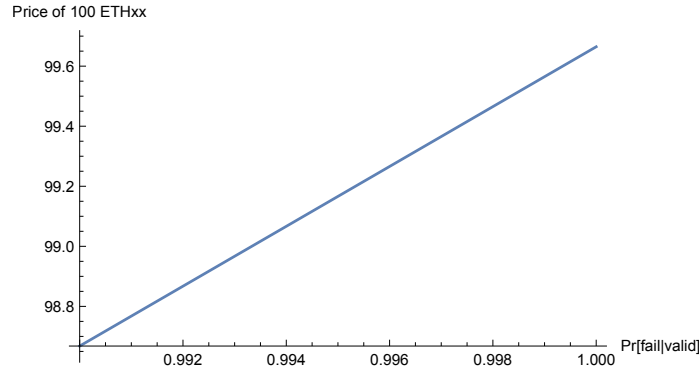
481 Consider how much you would pay for 100 ETH<sub>XX</sub> (finalized in 7 days = 168 hours) in  
482 ETH<sub>L1</sub> today. Since ETH<sub>XX</sub> is less flexible than ETH<sub>L1</sub>, it is likely that you do not prefer it  
483 to ETH<sub>L1</sub>, so our intuition is that it should be priced less (e.g., 100 ETH<sub>XX</sub> = 99 ETH<sub>L1</sub>).  
484 However, our solution works for any pricing and we can even contrive corner cases where  
485 ETH<sub>XX</sub> might be worth more than ETH<sub>L1</sub> by understanding the factors underlying the price.

486 In traditional finance [7], forward contracts (and futures, which are standardized, exchange  
487 traded forwards) are very similar to ETH<sub>XX</sub> in that they price today the delivery of an asset  
488 or commodity at some future date. One key difference is that with a forward contract, the  
489 price is decided today but the actual money is exchanged for the asset at delivery time.  
490 When ETH<sub>XX</sub> is sold for ETH<sub>L1</sub>, both price determination and the exchange happen today,  
491 while the delivery of ETH<sub>L1</sub> for ETH<sub>XX</sub> happens in the future. The consequence is that we  
492 can adapt pricing equations for forwards/futures, however, the signs (positive/negative) of  
493 certain terms need to be inverted.

494 We review the factors [7] that determine the price of a forward contract ( $F_0$ ) and translate  
495 what they mean for ETH<sub>XX</sub>:

- 496 • *Spot price of ETH<sub>L1</sub> ( $S_0$ ):* the price today of what will be delivered in the future. ETH<sub>XX</sub>  
497 is the future delivery of ETH<sub>L1</sub>, which is by definition worth 100 ETH<sub>L1</sub> today.
- 498 • *Settlement time ( $\Delta t$ ):* the time until the exit can be traded for ETH<sub>L1</sub>. In *Arbitrum*,  
499 the time depends on whether disputes happen. We simplify by assuming  $\Delta t$  is always 7  
500 days (168 hours) from the assertion time. A known fact about forwards is that  $F_0$  and  
501  $S_0$  converge as  $\Delta t$  approaches 0.
- 502 • *Storage cost ( $U$ ):* most relevant for commodities, receiving delivery of a commodity at a  
503 future date relieves the buyer of paying to store it in the short-term. Securing ETH<sub>XX</sub>  
504 and securing ETH<sub>L1</sub> is identical in normal circumstances, so not having to take possession  
505 of ETH<sub>L1</sub> for  $\Delta t$  time does not reduce costs for a ETH<sub>XX</sub> holder.
- 506 • *Delivery cost ( $D$ ):* the cost of delivery of the asset, which in our case will encompass gas  
507 costs. Exchanging ETH<sub>L1</sub> for ETH<sub>XX</sub> requires a transaction fee and also creates a future  
508 transaction fee to process the exit (comparable in cost to purchasing a token from an  
509 automated market maker). An ETH<sub>L1</sub> seller should be compensated for these costs in  
510 the price of ETH<sub>XX</sub>.
- 511 • *Exchange rate risk:* a relevant factor when the asset being delivered is different than the  
512 asset paying for the forward. In our case, we are determining the price in ETH<sub>L1</sub> for  
513 future delivery of ETH<sub>L1</sub>, thus, there is no exchange risk at this level of the transaction.  
514 However, the price of gas (in the term  $D$ ) is subject to ETH/gas exchange rates. For  
515 simplicity, we assume this is built into  $D$ .
- 516 • *Interest / Yield ( $-r + y$ ):* both ETH<sub>L1</sub> and ETH<sub>XX</sub> have the potential to earn interest  
517 or yield (compounding over  $\Delta t$ ), while for other tokens, there might be an opportunity  
518 to earn new tokens simply by holding the token. Let  $r$  be the (risk-free) interest (yield)

<sup>7</sup> GitHub:Nitro, Fast-Withdrawals: <https://github.com/MadibaGroup/nitro/tree/fast-withdrawals>



■ **Figure 2** Price of 100 ETH<sub>XX</sub> (in ETH) as the probability an RBlock actually finalizes (given the validator checks it with software validation) varies from 99% to 100%, which is denoted by  $R$ . Note that 99% is an extraordinarily low probability for this event (considering an RBlock has never failed at the time of writing). The take-away is that the price is not very sensitive to how precisely we estimate  $R$ .

519 rate for ETH<sub>L1</sub> that cannot be earned by ETH<sub>XX</sub>, while  $y$  is the opposite: yield earned  
 520 from ETH<sub>XX</sub> and not ETH<sub>L1</sub>. Initially  $y > 1$  and  $r = 0$ , however, with ETH<sub>XX</sub> becoming  
 521 mainstream, it is possible  $r = y$  (especially hedged ETH<sub>XX</sub>).

522 • *Settlement risk ( $R$ )*: the probability that ETH<sub>L1</sub> will fail to be delivered for ETH<sub>XX</sub>  
 523 discounts the price of ETH<sub>XX</sub>. We will deal with this separately.

524 Put together, the price of ETH<sub>XX</sub> ( $F_0$ ) is:

$$525 \quad F_0 = (S_0 + U - D) \cdot e^{(-r+y) \cdot \Delta t} \cdot R$$

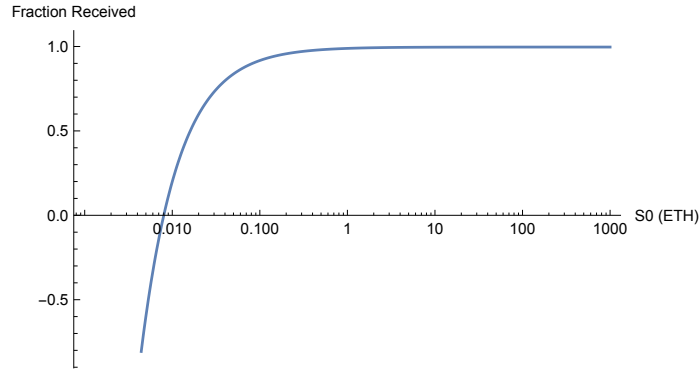
526 This value,  $F_0$ , is an expected value—the product of the value and the probability that  
 527 the RBlock fails to finalize. However, the trader is informed because they have run verification  
 528 software and checked that the RBlock validates.

$$529 \quad R = (1 - \Pr[\text{rblock fails to finalize} | \text{rblock passes software verification}])$$

### 530 Working Example.

531 We start with  $R$ . For an RBlock to be up for consideration, it must be submitted to the  
 532 outbox as a potential solution and for it to fail, a dispute must be filed with an alternative  
 533 RBlock that the L1 outbox deems to be correct. In our case, the buyer of ETH<sub>XX</sub> actually  
 534 runs a L2 validator and thus performs software validation on the RBlock, and will not accept  
 535 it if the software does not validate it. For an RBlock to fail given the software validation,  
 536 it software must have an error that causes a discrepancy between it and the L1 outbox.  
 537 Furthermore, at least one other validator would need to have different, correct software, and  
 538 this validator would need to be paying attention to this specific RBlock and independently  
 539 check it. This should be a rare event and assume  $R = (1 - 10^{-15})$  for this example. Figure 2  
 540 shows a range of  $R$  values.

541 Next, consider the resulting price of  $F_0$ . Alice starts with 100 ETH<sub>XX</sub> and Bob purchases  
 542 it from her. Bob can hold ETH<sub>XX</sub> with no cost ( $U = 0$ ). Alice pays the transaction fee  
 543 for the deposit, however the cost for the contract for exiting ETH<sub>XX</sub> into ETH<sub>L1</sub> after the



**Figure 3** This chart shows the percentage of ETH recovered ( $F_0/S_0$ ) as the amount withdrawn ( $S_0$ ) increases (log scale), demonstrating it is only economical for withdrawing larger amounts of  $ETH_{L2}$ . At low values, the gas costs of a withdrawal dominate. At very low values, the gas costs exceed the price of  $ETH_{XX}$  causing the curve to go negative.

544 dispute period is expected to be  $D = 0.008$  ETH ( $D$ ). Assume a safe-ish annual percent yield  
 545 (APY) on ETH deposits is 0.2%. Assume  $ETH_{XX}$  expires in 6 days (0.0164 years).  $ETH_{XX}$   
 546 earns no yield ( $y = 0$ ). Plugging this into the equation,  $F_0 = 99.665$  ETH.

547 As a second example, consider a smaller amount like 0.05  $ETH_{XX}$  (less than \$100 USD at  
 548 time of writing). Now the gas costs are more dominating.  $F_0 = 0.04186$   $ETH_{L1}$  which is only  
 549 83.7%. This demonstrates that fast exits are expensive for withdrawals of amounts in the  
 550 hundreds of dollars. Figure 3 shows a range of withdraw amounts.

551 Lastly, could  $ETH_{XX}$  ever be worth more than  $ETH_{L1}$ ? The equation says yes: with  
 552 a sufficiently high  $U$  or  $y$ . A contrived example would be some time-deferral reason (*e.g.*,  
 553 tax avoidance) to prefer receiving  $ETH_{L1}$  in 7 days instead of today. However, in order to  
 554 purchase  $ETH_{XX}$  at a premium to  $ETH_{L1}$ , it would have to be cheaper to trade for it than to  
 555 simply manufacture it. Someone holding  $ETH_{L1}$  and wanting  $ETH_{XX}$  could simply move it  
 556 to L2 and then immediately withdraw it to create  $ETH_{XX}$ . The gas cost of this path will be  
 557 one upper bound on how much  $ETH_{XX}$  could exceed  $ETH_{L1}$  in value.

## 558 Pricing $FINAL_{PM}$ and $FAIL_{PM}$ .

559 It might appear surprising at first, but one of the main results of this paper is that the price  
 560 of 100  $ETH_{XX}$  and the price 100  $FINAL_{PM}$  are essentially the same. Both are instruments  
 561 that are redeemable at the same future time for the same amount of  $ETH_{L1}$  (either 100 if  
 562 the RBlock finalizes and 0 if the RBlock fails) with the same probability of failure (that the  
 563 RBlock fails). The carrying costs of both are identical. There may be slight differences in the  
 564 gas costs of redeeming  $ETH_{L1}$  once the dispute period is over. However, the operation (at  
 565 a computational level) is largely the same process. This is actually a natural result: if 100  
 566  $FAIL_{PM}$  perfectly hedges (reduces the risk to zero) the failure of 100  $ETH_{XX}$  to finalize, then  
 567 the complement to  $FAIL_{PM}$ ,  $FINAL_{PM}$ , should be priced the same as  $ETH_{XX}$ .

## 568 **6** Discussion

### 569 6.1 Prediction Market Fidelity

570 A prediction market that covers a larger event should attract more interest and liquidity. For  
 571 example, betting on an entire RBlock will have more market interest than betting on Alice's



572 specific exit. On the other hand, if markets are exit-specific, the market can be established  
 573 immediately after Alice’s withdrawal hits the inbox instead of waiting for an RBlock (hence  
 574  $\sim$  in Table 1 to indicate it could be done within one L1 transaction). Another consideration  
 575 arises when tokens other than ETH are being withdrawn—if the payout of the market  
 576 matches the withdrawn token,  $\text{FAIL}_{\text{PM}}$  will perfectly hedge the exit. Otherwise the hedge  
 577 is in the equivalent amount of ETH which could change over 7 days. Our suggestion is to  
 578 promote the most traders in a single market and avoid fragmentation—so we suggest one  
 579 market in one payout currency (ETH) for one entire RBlock.

## 580 6.2 Withdrawal Format

581 As implemented, transferable exits can only be transferred in their entirety. If Alice wants  
 582 to withdraw  $100 \text{ ETH}_{\text{L2}}$  and give  $50 \text{ ETH}_{\text{XX}}$  to one person and  $50 \text{ ETH}_{\text{XX}}$  to another, she  
 583 cannot change this once she has initiated the withdraw (if she anticipates it, she can request  
 584 two separate withdrawals for the smaller amounts). We could implement divisible exits and  
 585 for ETH; there are no foreseen challenges since the semantics of  $\text{ETH}_{\text{L1}}$  are specified at the  
 586 protocol-level of Ethereum. However for custom tokens, the bridge would need to know  
 587 how divisible (if at all) a token is. In fact, a bridge should ensure that the L2 behavior of  
 588 the tokens is the same as L1 (or that any inconsistencies are not meaningful). Even if a  
 589 token implementation is standard, such as ERC20, this only ensures it realizes a certain  
 590 interface (function names and parameters) and does not mean the functions themselves are  
 591 implemented as expected (parasitic ERC20 contracts are sometimes used to trick automated  
 592 trading bots.<sup>8</sup> The end result is that bridges today do not allow arbitrary tokens; they  
 593 are built with allowlists of tokens that are human-reviewed and added by an authorized  
 594 developer. In this case, ensuring divisible exits are not more divisible than the underlying  
 595 token should be feasible, but we have not implemented it.

## 596 6.3 Markets

597 At the time of writing, the most common way of exchanging tokens on-chain is with an  
 598 automated market maker (AMM) (*e.g.*, *Uniswap*). If Alice withdraws  $\text{ETH}_{\text{XX}}$  and Bob is a  
 599 willing buyer with  $\text{ETH}_{\text{L1}}$ , an AMM is not the best market type for them to arrange a trade.  
 600 AMMs use liquidity providers (LPs) who provide both token types: Alice has  $\text{ETH}_{\text{XX}}$  but no  
 601  $\text{ETH}_{\text{L1}}$  that she is willing to lock up (hence why she is trying to fast exit). Bob has  $\text{ETH}_{\text{L1}}$   
 602 but to be an LP, he would also need to have  $\text{ETH}_{\text{XX}}$  from another user. However, this only  
 603 pushes the problem to how Bob got  $\text{ETH}_{\text{XX}}$  from that user. The first user to sell  $\text{ETH}_{\text{XX}}$   
 604 cannot use an AMM without locking up  $\text{ETH}_{\text{L1}}$ , which is equivalent to selling  $\text{ETH}_{\text{XX}}$  to  
 605 herself for  $\text{ETH}_{\text{L1}}$ . The second challenge of an AMM is the unlikely case that an RBlock fails  
 606 and  $\text{ETH}_{\text{XX}}$  is worthless—then the LPs have to race to withdraw their collateral before other  
 607 users extract it with worthless  $\text{ETH}_{\text{XX}}$ . It is better to use a traditional order-based market;  
 608 however, these are expensive to run on L1 [11]. One could do the matchmaking on L2 and  
 609 then have the buyer and seller execute on L1, but this reintroduces the griefing attacks we  
 610 have tried to avoid. For now, we implement a very simple one-sided market where Alice can  
 611 deposit her  $\text{ETH}_{\text{XX}}$  and an offer price, and Bob can later execute the trade against. If Alice  
 612 is unsure how to price  $\text{ETH}_{\text{XX}}$ , an auction mechanism could be used instead.

---

<sup>8</sup> “Bad Sandwich: DeFi Trader ‘Poisons’ Front-Running Miners for \$250K Profit.” *CoinDesk*, Mar 2021.



## 6.4 Low Liquidity or Non-Fungible Tokens

For tokens that have low liquidity on L1, or in the extreme case, are unique (*e.g.*, an NFT), fast exits do not seem feasible. All the fast exit methods we examined do not actually withdraw the original tokens faster; they substitute a functionally equivalent token that is already on L1. However, we can still help out with low-liquidity withdrawals. We should consider *why* the user wants a fast exit. If it is to sell the token, they can sell the exit instead of the token to any buyer that is L2-aware and willing to wait 7 days to take actual possession. To sell to an L2-agnostic buyer, the seller can insure the exit with enough  $\text{FAIL}_{\text{PM}}$  to cover the purchase price. In this case, the buyer does not get the NFT if the RBlock fails but they get their money back.

## 7 Concluding Remarks

This paper addresses a common ‘pain point’ for users of L2 optimistic rollups on Ethereum. The 7-day dispute period prevents users from withdrawing ETH, tokens, and data quickly. Tradeable exits provide users with flexibility after they request a withdrawal. If they decide 7 days is too long, they can seek to trade their exit for  $\text{ETH}_{\text{L1}}$  or they can ask a contract to accept their  $\text{ETH}_{\text{XX}}$  by bundling it with insurance against the failure of the RBlock—this way the contract does not have to be L2-aware. While some users might still prefer the features of other withdrawal methods (centralized exchanges or solution like *Hop*), it is useful to make the native rollup functionality as flexible as possible, especially for users who do not realize that a withdrawal induces a 7-day waiting period until it is too late.

## References

- 1 Jeremy Clark, Joseph Bonneau, Edward W Felten, Joshua A Kroll, Andrew Miller, and Arvind Narayanan. On decentralizing prediction markets and order books. In *Workshop on the Economics of Information Security (WEIS)*, volume 188, 2014.
- 2 Didem Demirag and Jeremy Clark. Absentia: Secure multiparty computation on ethereum. In *Workshop on Trusted Smart Contracts (WTSC)*, pages 381–396. Springer, 2021.
- 3 Bryan Ford and Rainer Böhme. Rationality is self-defeating in permissionless systems. Technical Report cs.CR 1910.08820, arXiv, 2019.
- 4 Nomic Foundation. Hardhat. <https://hardhat.org>, October 2022. (Accessed on 10/18/2022).
- 5 Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. Sok: Layer-two blockchain protocols. In *Financial Cryptography*, 2020. URL: [https://doi.org/10.1007/978-3-030-51280-4\\_12](https://doi.org/10.1007/978-3-030-51280-4_12).
- 6 Larry Harris. *Trading and exchanges: market microstructure for practitioners*. Oxford, 2003.
- 7 John Hull, Sirimon Treepongkaruna, David Colwell, Richard Heaney, and David Pitt. *Fundamentals of futures and options markets*. Pearson Higher Education AU, 2013.
- 8 Harry Kalodner, Steven Goldfeder, Xiaoqi Chen, S Matthew Weinberg, and Edward W Felten. Arbitrum: Scalable, private smart contracts. In *USENIX Security Symposium*, pages 1353–1370, 2018.
- 9 Patrick McCorry, Chris Buckland, Bennet Yee, and Dawn Song. Sok: Validating bridges as a scaling solution for blockchains. Technical report, Cryptology ePrint Archive, 2021.
- 10 Sarah Meiklejohn. An evolution of models for zero-knowledge proofs. In *EUROCRYPT (invited talk)*, 2021.
- 11 Mahsa Moosavi and Jeremy Clark. Lissy: Experimenting with on-chain order books. In *Workshop on Trusted Smart Contracts (WTSC)*, 2021.
- 12 Paul Sztorc. Truthcoin. Technical report, 2015.

## 22:18 Fast and Furious Withdrawals from Optimistic Rollups

- 658 13 Louis Tremblay Thibault, Tom Sarry, and Abdelhakim Senhaji Hafid. Blockchain scaling using  
659 rollups: A comprehensive survey. *IEEE Access*, 10:93039–93054, 2022.
- 660 14 Alexei Zamyatin, Mustafa Al-Bassam, Dionysis Zindros, Eleftherios Kokoris-Kogias, Pedro  
661 Moreno-Sanchez, Aggelos Kiayias, and William J Knottenbelt. Sok: Communication across  
662 distributed ledgers. In *Financial Cryptography*, 2021.