AC/DC: Adaptive Cutoffs and Disputable Cutoffs for robust critical transactions in smart-contracts

Rami A. Khalil and Naranker Dulay
Imperial College London
{rami.khalil,n.dulay}@imperial.ac.uk

***Abstract*—To guarantee delivery of their intended functionalities in the presence of unresponsive parties, current smart-contracts cut users off from being able to commit their responses after a fixed period of time has elapsed. However, current blockchains have limited transaction processing capacities, so a fixed amount of time will not always be sufficient to receive every critical transaction (C-TX). This paper presents a mechanism for adaptive cutoffs (ACs) which ensures that users retain the opportunity to commit C-TXs despite blockchain congestion, and enables early cutoffs when the number of required C-TXs is low. A non-interactive argument system for setting adaptive cutoffs under the current Ethereum Virtual Machine is described. Additionally, disputable cutoffs (DCs) are presented, which are a more efficient approach used in parallel to ACs based on a bisection-based dispute. Furthermore, it's empirically demonstrated that an AC/DC-enabled smart-contract can receive a larger number of C-TXs than its non-adaptive counterparts when user responsiveness is slowed due to denial of service or congestion.**

I. Introduction

A central theme in smart-contract design is to continue operation even when the parties required to provide input become unresponsive. Usually, after a predetermined fixed time-out period passes, the smart-contract no longer waits for unresponsive parties to publish their transactions, and proceeds with its execution. The interpretation of unresponsiveness by the smart-contract is application-dependent, where the unresponsive party can be considered either to be maliciously withholding information, benignly uninterested in participating, or even proactively in agreement with the state of the smart-contract's execution [1]. In this work, transactions which must be published before such timeouts elapse are referred to as critical transactions (C-TXs).

The current state-of-the-art in enforcing timeouts on C-TXs follows only two main criteria: (i) a predetermined amount of time passes, and/or (ii) a predetermined number of blocks are generated, before a smart-contract cuts off its openness to receive C-TXs. This design pattern has become a de-facto standard employed today in several types of smart-contracts, including auctioning [2], voting [3], decentralized finance [4], and scalability solutions [5].

However, this paradigm of applying a fixed predetermined timeout does not gracefully coincide with the limited transaction-processing bandwidth of blockchains. As we will demonstrate, a fixed cutoff implies that only a limited number of critical transactions can be confirmed, and consequently smart-contracts may not operate as intended. This limit is based on: (i) the time allotted before the cutoff period, (ii) the transaction throughput of the blockchain, and (iii) the size of the C-TXs. Moreover, high blockchain transaction fees can hinder a user's ability to successfully publish a C-TX before the cutoff. Since blockchain processing power is limited, the maximum number of C-TXs can be calculated using Equation 1.

$$\max \text{ ctx} = \frac{\text{block size}}{\text{ctx size}} \times \frac{\text{cutoff period}}{\text{block delay}} \tag{1}$$

To avoid the potential impact of this limitation, "safe" fixed cutoff periods which provide appropriate opportunity for all interested parties to publish their transactions are set. For example, voting periods in MakerDAO [6] last for three days, or seven days, depending on the issue being raised[1]. Similarly, Optimistic Rollup scaling solutions only finalize transactions after waiting for 1-2 weeks [7]. Unfortunately, setting a long predetermined cutoff period as a defense may not always be a viable option, especially when accounting for large user-bases and adversaries with significant economic incentives.

While the inadequate length of a timeout may cause users to pay high fees, a poll to fail, or an auction to end prematurely, it can be harmful in other applications. Particularly, an imbalance between the potentially numerous C-TXs which may be required, and the length of the cutoff period provided by the smart-contract can severely affect the security of second-layer scalability solutions, which aim to onboard millions of users while substantially reducing the load on their underlying blockchains. This is achieved by locking user funds into a smart-contract which defines agreement terms for future withdrawal, and a protocol that allows users to transfer ownership of portions of their locked funds to each other by only exchanging signed off-chain messages. Users can unlock their funds using the smart-contract, even if other users are dishonest. Such payment agreements have been realized so far in two main ways [5]: (i) two-party agreements called **payment channels** [8]; and (ii) multi-party agreements, called **Plasma schemes**, where one party is a designated intermediary called the Plasma operator [9].

Referring to smart-contract blockchains as layer-one, and to off-chain systems as layer-two, the main security goal in layer-two is to prevent fraudulent layer-one withdrawals, whereby a user attempts to unlock funds it does not own. In payment channels, a party can attempt fraud by requesting to close

[1] https://makerdao.world/en/learn/governance/how-voting-works/

the channel and unlock its funds based on outdated balance information. In Plasma schemes, users can attempt to unlock their funds after having spent them [9], or collude with the Plasma operator to attempt to "counterfeit" their balances [10] and withdraw them.

To counter fraud, current platforms delay the completion of a withdrawal in layer-one by a fixed amount of time during which users can dispute the withdrawal. After the dispute period ends, the ability to contest a withdrawal is forfeit, and the withdrawn funds are unlocked from the smart-contract into the blockchain account of the withdrawal initiator [5].

The rigidity of the dispute period is problematic. First, regular transaction activity taking place in layer-one reduces its capacity to process layer-two dispute transactions. This forces payment-channel and UTXO-based Plasma-scheme users to pay high transaction fees for high priority dispute transactions. For example, the **OmiseGo Network** [11] platform users must broadcast their transactions with very high priority to successfully protect their funds, which requires high transaction fees. Second, a fixed dispute period constrains the number of users that can be safely registered in an account-based Plasma-scheme, as every user would issue a dispute in the worst case. For example, this means that roughly no more than 300,000 **Liquidity.Network** [12] platform users can be safely registered regardless of their transaction fees as of the time of writing.

As a concrete example, consider the account-based Plasma scheme NOCUST [10] when deployed on the Ethereum blockchain. Once its central operator publishes a commitment of the latest state of the layer-two ledger to the smart-contract, users only have 36-hours to dispute its correctness. Given an average Ethereum block creation delay of 12 seconds between blocks, a block size of 10,000,000 gas units, a dispute cost of 360,000 gas per user, and a dispute period of 36-hours as in [10], we can calculate using Equation 1 that no more than roughly 300,000 disputes can be processed at that block size.

In this paper we describe a mechanism for smart-contracts to set an adaptive cutoff (AC) period for critical transactions, such as layer-two fraud disputes. This mechanism enables smart-contracts to adjust the opportunity window for C-TXs in response to layer-one congestion, and the number of C-TXs successfully received, as illustrated in Figure 1. To integrate adaptive cutoffs with the Ethereum Virtual Machine, we give a non-interactive argument system using only currently deployable smart-contracts. Moreover, we describe a mechanism for contracts to operate using disputable cutoffs (DCs), which are more efficient than ACs in reaching cutoffs for smart-contracts. We also empirically demonstrate that AC/DC-enabled protocols can surpass their static counterparts in preventing fraud.

The remainder of this paper is structured as follows: Section II reviews related work. Section III overviews critical transactions. Section IV presents adaptive cutoffs. Section V introduces our proving system for ACs. Section VI presents disputable cutoffs. Section VII evaluates our AC/DC mechanisms, and Section VIII concludes the paper.

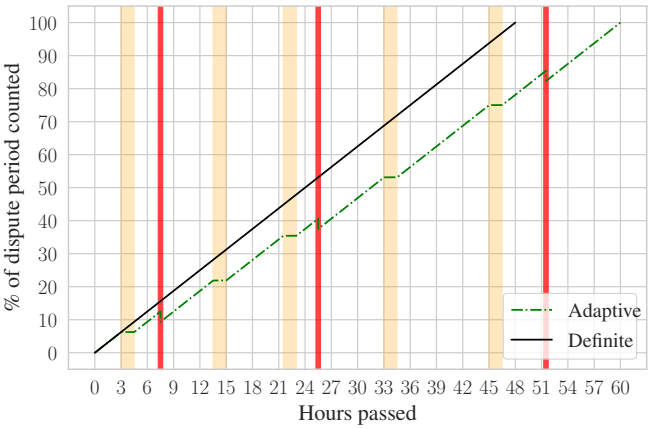

Figure 1: Plots of how an example 48-hour cutoff period is counted, where AC/DC starts with an initial 48-hours, and allows 12 additional hours for time-sensitive transactions, compared to a fixed cutoff period of only 48-hours. Yellow strips denote 1.5-hour periods of blockchain congestion, where AC/DC pauses the cutoff period. Red strips denote half-hour periods of continuous time-sensitive transaction publication, where AC/DC prolongs the cutoff period.

II. Related Work

A. Layer-one

1) Transaction Metering: In the Ethereum Virtual Machine [13] (EVM), the computational, storage, and transmission efforts required to process a transaction are characterized by its *gas* consumption, where gas is a unit designed to capture the total cost of execution of a transaction. Consequently, an EVM transaction must specify both (i) a limit of how much gas the transaction may consume, and (ii) a price paid by the transaction sender per unit of gas the transaction consumes. Accordingly, the capacities of Ethereum blocks are decided by the network in terms of gas units. Some studies investigate the precision of gas as a measure of true transaction execution cost [14], and others investigate how transaction gas prices can be set effectively [15]. In AC/DC, we use blockchain gas consumption as a measure of C-TX publication opportunity.

2) Censorship: Because miners are not constrained on which transactions they can include in a block, or how they may order those transactions, external factors can affect their choices. Namely, miners may be threatened [16], or bribed [17, 18], to not mine transactions that do not meet an adversary's criteria, and moreover fork the canonical chain when other miners mine a block that includes transactions which do not abide by the adversary's standards [19]. While such attacks have been mainly analyzed under the premise that the goal of the adversary is to double-spend its balance in the ledger, the adversary may specify a blacklist of addresses, banning them from sending or receiving transactions, or specify a whitelist, permitting only certain addresses to be party to a transaction [20, 21]. Such a targeted attack can undermine the security of smart-contracts that rely on C-TXs [22].

3) Auctions: Blockchain auctions are built using a recipient smart-contract which remains open to receiving bids as long as the auction is deemed running, effectively rendering bids as

C-TXs. Different designs and implementations offer various privacy guarantees [23], transaction costs [24], and price mechanics [25]. However, to the best of our knowledge, all state-of-the-art such contracts do not automatically account for blockchain congestion, which may hinder their ability to receive bids, and remain cost-efficient.

4) Governance: Governance smart-contracts allow holders of certain tokens to propose and vote on referendum which affect these tokens. Usually, referendums first go through a proposal process, which narrows down which referendums will become open for voting. Referendums then become open for a fixed period of time, usually from a few days to a couple of weeks, during which token holders can cast their votes, through publishing C-TXs directed at the governance smart-contract [3]. Congestion can affect the costs associated with participating in governance contracts by forcing participants to publish their time-critical voting transactions at high fees. In contrast, AC/DC can enable participation in on-chain governance at fixed costs.

5) Decentralized Finance: Commonly known as DeFi, Decentralized Finance applications leverage smart-contracts to enable exchange, borrowing and lending, stablecoin maintenance, portfolio management, derivative trading, and mixers [4]. Their main highlight is their decentralized and permissionless design. However, because miners have an incentive to include transactions with higher fees before transactions with lower fees, the problem of Priority gas auctions (PGAs) arises [26], whereby bots compete to prioritize their transactions by paying higher fees than regular users in order to derive a guaranteed profit from their DeFi transactions (e.g. through arbitrage on decentralized exchanges). Prior to this work, the potential to operate a DeFi application with upper bounds on the maximum transaction fees (i.e. gas price) that can be utilized was not feasible.

B. Layer-two

1) Payment Channels: Channels are established between two parties and undergo three stages: (i) on-chain deployment, (ii) off-chain update and (iii) on-chain termination. Several designs secure the termination phase through the use of static-time locks [27–38], or a fixed dispute period [1, 39–50], leaving both exposed to the bandwidth limitations of layer-one, as leveraged in [51]. However, some designs secure their termination differently: (i) Brick relies on a committee's consensus on the latest state for termination [52], (ii) Teechain similarly uses committees called *Treasuries* that are secured by trusted hardware [53], (iii) and Teechan uses trusted execution environments to prevent channel termination with outdated balances [54].

2) Plasma: Plasma is an off-chain architecture managed by a central operator [9]. In Plasma MVP, the central operator periodically publishes a commitment to the state of the UTXO ledger on the parent-chain. Users must validate the full ledger with every commitment, and dispute any fake transactions or dishonestly minted funds within a limited amount of time. Plasma Cash reduces the user verification overhead of Plasma MVP. Each deposit creates a new coin whose ownership can be transferred. A user withdrawing m individual coins has to initiate m disputable withdrawals using the smart-contract, which must all be disputed within a fixed period. Plasma Debit amends Plasma Cash by adding a numeric value a to each coin to denote what portion of the corresponding deposit belongs to the owning user, while the rest is owed to the operator, bringing it closer to a payment-channel design. Plasma Prime reduces the transaction costs of disputes. In all aforementioned designs, the finality of each withdrawal is established after a fixed cutoff period passes. NOCUST is an account-based Plasma design where an operator that withholds the data behind its commitments must be forced to reveal it within a fixed-time period [10]. However, the fixed-time allotted for these queries means that not all users can perform them. In NOCUST-ZKP commitments are proven in zero-knowledge to be correct, but disputes to reveal data may only be performed within a fixed amount of time [55]. The lower bound on the number of disputes that must be made in Plasma protocols is analyzed in [56].

3) Rollups: Rollups leverage techniques that are similar to those of Channels and Plasma to combine several off-chain transfers into a single on-chain transaction submitted to a smart-contract [57]. However, while transfers are submitted to a Plasma-like operator directly in these models, a blockchain fee must always be paid for each transfer. Rollups therefore do not fully fall within the off-chain paradigm, but only offload a portion of the computational cost of transaction execution from layer-one to layer-two. To execute transfers, the operator submits the on-chain transaction to the smart-contract, which verifies that all transfers included in the transaction are valid. In ZK-Rollups, the on-chain transaction is includes a zero-knowledge proof of the validity of all off-chain transfers to avoid full verification by the smart-contract [58]. This design heavily borrows from the optimization methods of Coda [59–61], and does not depend on cutoffs. On the other hand, Optimistic Rollups require no immediate verification of the on-chain transaction, and instead wait for a proof of invalidity to be submitted, again only within a fixed time period [7].

III. Critical Transactions

In this section we provide an overview of C-TXs in smart-contracts. We first present C-TXs in layer-one, and emphasising some of the key conditions for C-TXs to work as intended. We then highlight how different layer-two protocol design elements lead to different security guarantees and to different requirements for C-TXs to successfully prevent fraud. We use the informal definitions presented here later in Section IV as the basis for adaptive cutoffs and in Section VII to compare the efficacy of adaptive cutoffs with fixed cutoffs.

A. Layer-one Polls

Critical transactions in fully on-chain layer-one smart-contract protocols are mostly utilized to tally responses for polls, elections, auctions, or other processes that should end after a finite amount of time. For brevity, we refer to all such processes as polls, whereby users participate with answers, votes, bids, or similar responses as transactions.

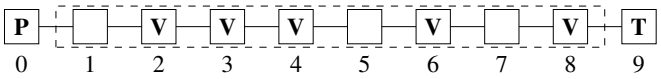

Figure 2: Timeline of an example polling process chronologically ordered by the block number in which each event occurs. In block 0, a **P**oll is initiated. Users publish votes in Blocks with a **V**. In block 9, the votes are **T**allied. Blocks 1 to 8, surrounded by a dashed rectangle, constitute the opportunity to participate in the poll.

Responding to polls can either be implemented in a restricted manner, where a single participant, or multiple participants, may provide up to n responses for example, or in an unrestricted manner where participants are free to add more responses as they wish. Furthermore, the tally may either consider all the responses of each participant, for example when aggregating all contributions made in a Dutch auction, or consider only the last response made by each participant, when tallying election votes for example.

1) Restricted polls: In a restricted poll the timeliness of issuing responses only falls on a participant until they exhaust their response quota. After reaching their quota, participants may then only observe the poll until it ends and the final results are tallied. For example, in an election, votes may be cast once, with no option of changing them.

2) Unrestricted polls: An unrestricted poll may require the attentiveness of participants until it ends. These polls are suitable when receiving more responses by the poll may cause some participants to submit further responses in turn. For example, in a first-bid auction, a participant which was outbid may need to make another bid to secure the auctioned item.

Generally, a blanket cutoff is set for each poll, allowing participants to submit their responses before a fixed deadline, as illustrated in Figure 2. One notable exception are Dutch auction smart-contracts, which adjust the deadline for receiving bids based on all bids received so far.

B. Layer-two Disputes

Layer-two protocols rely on a *spokesperson* publishing statements in smart-contracts about the states of layer-two data, such as accounts and payments. For example, a Plasma operator would publish a **statement** of the form: "*Commitment Y embodies the latest correct balances of all layer-two accounts*". Any errors in a statement are resolved using disputes. Disputes are critical transactions, designed to prevent fraudulent statements from becoming enforced, that we classify as one of (i) queries, (ii) claims or (iii) proofs of fraud. A published statement that has not been disputed is only considered **final** once the statement can no longer be disputed, i.e. the cutoff for dispute publication is reached.

1) Queries: Queries are requests for information about a published statement, to which a correct answer must be provided by the spokesperson in a timely manner to prevent the statement from being considered as disputed. For example, a query of the form: "*What is the layer-two balance for account X as of commitment Y?*", must be answered by returning the correct layer-two balance for the account in the NOCUST smart-contract [10].

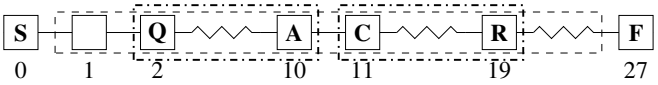

Figure 3: Timeline of an example dispute process chronologically ordered by event block number. In block 0, the spokesperson publishes a **S**tatement. In block 2, a user publishes a **Q**uery about the statement. In block 10, the spokesperson publishes the **A**nswer to said query. In block 11, a user publishes a **C**laim against the statement. In block 19, the spokesperson publishes a **R**efutation of said claim. In block 27, the statement is considered **F**inal. Blocks 1 to 26, surrounded by a dashed rectangle, constitute the opportunity to dispute the statement. Blocks 2 to 10, and 11 to 19, constitute the opportunities to answer the query, and refute the claim, respectively.

2) Claims: Claims are incriminating allegations in the smart-contract against a statement's validity. A claim must be refuted in a timely manner by proving the claim's incorrectness and the dishonesty of the claimant. For example, a claim of the form: "*The layer-two balance of account X as of commitment Y is incorrect*" can be refuted by proving that the account's layer-two balance is correctly derived in NOCUST [10]. A claim should only be made when the claimant is convinced that the allegation cannot be refuted, and a refutation that shows a claim to be false should penalize the claimant.

3) Proofs of fraud: Proofs of fraud are direct arguments which undeniably demonstrate that a statement is false and its spokesperson is dishonest. For example, a proof of the form: "*The spokesperson has incorrectly calculated the balance of account X as of commitment Y*" would undeniably prove that the balance is indeed incorrect, and that the operator is definitely at fault in NOCUST [10].

As query answers and claim refutations must be provided within a timely manner, individual cutoffs for their publication in the chain must also be defined with respect to the publication of each query or claim. These additional cutoffs are usually independent from that associated with disputing the statement in question. This is illustrated in Figure 3.

C. Publication Opportunity

Following the start of a poll, or the publication of a statement, on-chain smart-contracts are designed to grant some amount of time, or an opportunity, for users to submit their responses or initiate disputes. The opportunity to publish the necessary C-TXs depends on two main factors, (i) the ability of users to create their C-TXs after discovering a poll or statement, and (ii) the ability to publish created C-TXs on the layer-one blockchain within the amount of time granted by the protocol.

1) Transaction Creation: When speaking about the ability of a user to create a critical transaction, we refer only to the process of a user realizing that a poll has started, or that some statement cannot (yet) safely be left to become final. For a poll, this process includes understanding the poll parameters

and outcomes. For statements, this process includes creating a query about the statement, such as designating some data behind a commitment to be revealed, or constructing a claim or fraud proof against the statement. While such a process may be considered as simply a prelude to actually responding or launching a dispute, it is isolated and identified on its own in this discussion as it only depends on the awareness of users of published blockchain data. This ability depends on two main requirements: online presence, and decisional effort.

a) Online Presence: Strict online presence dictates that users constantly monitor for any spokesperson activity in layer-one, as the spokesperson is free to start polls or publish statements at any time. For example, payment-channel users must always watch the blockchain for any attempts by their counter-parties to withdraw funds using outdated state information. This requires remaining constantly online, downloading every layer-one block, and checking whether there exists a transaction which attempts to close the channel. A lower form of online presence only requires users to come periodically online, as the spokesperson may only publish a potentially disputable statement once every pre-defined time period. For example, a NOCUST operator may only publish a commitment to the state of the ledger once every pre-defined period, which means users only have to monitor the chain for a statement once per period [10]. The lowest online presence requirement would be to appear online a small constant number of times, or somehow not at all, such as through delegation to other parties. For example, once during a predetermined auction period, and once after the auction end to inspect the result. Consequently, the online presence requirements of a protocol constrain the opportunity to submit C-TXs only to users who meet a certain connectivity level to layer-one.

b) Decisional Effort: Layer-two protocols require varying degrees of computational from users before they can gain confidence in their disposition towards a poll or statement. For example, when payment-channel users verify a channel closure statement, they only need to compare the balance information being used to close the channel with the latest balance information they know, often a constant-time procedure. Similarly, a NOCUST user validates information proportional to how many transactions it has personally performed since the last commitment about the ledger, independent of how many transactions other users in the ledger have performed [10]. On the other hand, Plasma MVP extensively validates the entire ledger after every statement by the operator, which may require a non-trivial amount of time [9]. Furthermore, prediction market contracts may require end users to tally all bids before providing sufficient information for the user to participate. Ultimately, these requirements constrain dispute opportunity only to users who meet the necessary computational demands.

Because of a protocol's online presence and verification requirements, the opportunity available for issuing C-TXs can be undermined by an adversary who launches a denial of service attack that prevents users from realizing that a disputable statement is published, or by the user's own delay in learning of a poll. For example, blocks numbered 1 in Figures 2 and 3 contain no action by the user, which can be attributed to either of the aforementioned possibilities.

2) Transaction Publication: The publication of created user responses and disputes in a layer-one blockchain largely depends on the blockchain's transaction confirmation behavior. This process begins with the user broadcasting the critical transaction to the layer-one network of block proposers, and ends when a layer-one block that contains the transaction is confirmed. However, the process may end in failure if the target poll had ended, or the statement being disputed had already been considered final, before the dispute transaction was included in a confirmed block.

a) Gas Usage: As mentioned in Section I, only a limited number of transactions can be published per block, and consequently, C-TXs cannot always be immediately published in a block following their creation. Notably, when publishing polls or statements costs significantly less gas than publishing responses or disputes, a protocol may create an asymmetry, in terms of publication power, between spokespersons and users. For example, in account-based Plasma schemes, the central operator may affect the funds of all of its users by publishing a single small commitment statement. However, to dispute this statement, all of the affected users have to issue several queries and claims, each of which is significantly more expensive than the original statement. Essentially, the efficiency of dispute publication, in terms of gas cost, in comparison to that of statement publication, determines whether an opportunity to dispute all of the effects of a statement exists. This similarly holds for creating and responding to polls.

b) Transaction Priority: Recall that the current transaction ordering mechanisms in most layer-one blockchains incentivize miners to prioritize transactions based on how much fees the transactions pay to the miners upon their execution. Such mechanisms enable publishers to prioritize their transactions over those of others by paying higher transaction fees. Accordingly, such mechanisms enforce a minimum transaction fee for C-TXs, putting a price on the opportunity to respond to polls and dispute statements that is in line with the layer-one transaction fee market prices at the time of publication. Even when miners only receive part of the transaction fee while the remainder is burned as in Ethereum Improvement Proposal 1559, the expected behavior of miners is to prioritize transactions which directly benefit them the most [62]. Fundamentally, the difference in publication priority between spokespersons and users affects a protocol's usability and security, and determines the fairness of the time-critical response opportunity. For example, in a payment-channel protocol, a well-connected and wealthy spokesperson with a significant number of open channels can publish a flood of channel closure statements for all of its channels with a very high priority. Such a scenario only grants a dispute opportunity to channel users who can afford to publish disputes with higher priority than the statements of the wealthy spokesperson. Even when incentives are in place either to discourage publishing fraudulent statements or to encourage their dispute, such as penalties for attempting channel closure with outdated information [8], a spokesperson can estimate the inability of users to afford a costly dispute from their public balances, and attempt to leverage high priority transactions to successfully publish fraudulent statements.

IV. Adaptive Cutoffs

In this section we describe the adaptive cutoff (AC) mechanism and the criteria with which this mechanism considers a poll closed or a statement final. We follow an incremental approach by first defining basic conditions under which an adaptive critical transaction publication period should be cut off to provide improved publication opportunity, and subsequently adding further conditions that account for gas price, congestion, and dynamic block sizes.

The *requirements* for the AC mechanism are (i) a smart-contract blockchain with gas-based transaction metering, such as Ethereum, (ii) that users possess sufficient layer-one funds for publishing C-TXs, (iii) that users publish C-TXs when necessary. The latter two requirements imply that unless a user trusts another party to launch C-TXs on its behalf, as in PISA, a completely passive account with no blockchain balance is not granted opportunity to respond or dispute [40].

Note that, AC does not tackle privacy, and offers no censorship resistance against powerful layer-one block proposers. The identities of parties involved in C-TXs, and the application smart-contract that integrates AC, are assumed to be public. This leaves AC vulnerable to a powerful layer-one adversary preventing critical transactions from being ever confirmed.

A. Statement Finalization

Assuming that layer-one is resistant to censorship, AC considers gas that was unused for responses or disputes as an indicator that potential C-TX publication opportunities were unnecessary. For example, if after the publication of some statement in layer-one, no disputes about this statement were published in any of the blocks following the one containing the statement, then the full gas of all these blocks is considered to have added to the credibility of the statement, and is referred to as **ratification-gas**, or "***r*-gas**" for short. Consequently, the end of a poll or a statement's dispute period is determined in AC using the amount of *r*-gas accumulated over time.

$$r(d) = \Delta \times \text{gps} + \alpha \times d \times c \tag{2}$$

Equation 2 defines the **required *r*-gas** for an adaptive cutoff to be reached as $r(d)$, a function of d, the number of C-TXs issued towards the poll, or against the statement, before the cutoff is reached. The **maximum gas cost**, in gas units, for a C-TX is denoted by c, while the **minimum cutoff** is denoted by Δ, a pre-defined amount of time. The **layer-one throughput**, in terms of average gas per second, is denoted by *gps*. The **magnification factor** α adjusts the increase of the required *r*-gas remaining in response to the number of C-TXs published so far. This allows tolerance against degradation in the online presence of users, e.g. due to a service provider outage [63], and adaptively delays the cutoff of a poll or statement in response to C-TXs. For example, consider a layer-two protocol that requires users to be constantly online to issue disputes. If users appear online randomly before the minimum cutoff, Δ, instead of being constantly online, and a disputable statement is published, then users would not publish their disputes in perfect sequence. Instead, some amounts of *r*-gas would accumulate in between each user dispute. In turn setting the magnification factor α to a value larger than 0 would make up for the expected average gap between consecutive disputes. Furthermore, if the minimum cutoff Δ were not late enough for all possible disputes to be processed, then setting the magnification factor α to a value at least 1 would allow one more user dispute to be processed for each user dispute that is processed, and allow the dispute period to be prolonged enough for all possible disputes to be issued.

In AC, starting at time t_0, the cutoff is reached at the first point in time $t_1 > t_0 + \Delta$ if at least $r(d)$ of *r*-gas was accumulated in the layer-one blockchain between t_0 and t_1. When there are no C-TXs published, such that $d = 0$, or the magnification factor $\alpha = 0$, the cutoff is reached after the minimum cutoff of Δ seconds on average. However, as disputes are issued and d increases, a delay is incurred. With some fraction, denoted by e, of the gas *not* going towards processing dispute transactions, the delay would amount to $r(d) \div ((1 - e) \times \text{gps})$. Optimizing the maximum gas cost of a response or dispute c can mitigate this delay.

B. Priced Statement Finalization

The risk that not all C-TXs are published before the cutoff is a burden for a protocol's users. This risk creates an incentive to publish C-TXs with high fees to increase their priority, which may lead to a surge in layer-one transaction fees. The *r*-gas driven approach improves this scenario by granting users more time based on layer-one capacity. However, if layer-one fees surge, then *r*-gas would not represent an unused dispute opportunity for users with insufficient layer-one balance.

To remedy the aforementioned downside of *r*-gas, a fixed **dispute pricing factor** p is incorporated in AC. Accordingly, the adaptive cutoff then is additionally only reached when enough *r*-gas *priced less than* p, referred to as "***p*-gas**", is accumulated after the publication of a poll or statement. Accounting for the layer-one gas pricing required for C-TXs allows users to plan ahead their publication costs, and provides an equitable opportunity for users who cannot afford to win a layer-one transaction fee bidding war.

Empirically, the average daily gas price in Ethereum has not so far risen above 1000×10^{-9} ETH[2]. For Ethereum, p would determine the transaction gas price. With this, we can estimate that $p = 1000 \times 10^{-9}$ ETH would give C-TXs a reasonably high priority in Ethereum. However, such static pricing does not guarantee absolute protection from unpredictable increases in gas fees, which may cause a statement to suffer delays in accumulating sufficient *p*-gas for finality, or to never finalize if gas prices become permanently higher than *p*.

C. Constant-Price Statement Finalization

One additional concern is the fragmentation of accumulated *p*-gas over several blocks. If publishing C-TXs costs a significant amount of gas, then *p*-gas may not reflect the actual opportunity that had been available to issue such a C-TX.

[2]https://etherscan.io/chart/gasprice

For example, consider a C-TX which requires 10,000,000 gas to execute. If exactly 5,000,000 of gas in two consecutive blocks was accumulated as *p*-gas, then the total *p*-gas accumulated would be the gas required for a single C-TXs (10,000,000). Relying on *p*-gas would lead to a false indication of a sufficient opportunity to issue the C-TXs, despite there being no single block containing 10,000,000 *p*-gas.

To remedy this, the maximum gas cost c of a C-TX is incorporated into the AC mechanism, such that the cutoff is reached only when enough c-sized consolidated units of *p*-gas have been accumulated. The term "***c*-gas**" is used to refer to the accumulation of such consolidated units, which must be accumulated only in multiples of c within a single block. To elaborate, the total *c*-gas accumulated in the example of the two blocks in the previous paragraph, where c is equal to 10,000,000, would be zero, as no single block contained the required amount of *p*-gas.

D. Bounded Statement Finalization

The last restriction we set protects AC against potential changes in layer-one transaction capacity per block. Primarily, Ethereum is known to have a dynamic gas limit, which has been increasing since its creation, and such an increase may lead to faster accumulation of *c*-gas that would lead to unstable minimum cutoffs. Consequently, we set an upper-bound on the amount of *c*-gas that maybe accumulated in each block to the pre-defined *gps* value multiplied by the expected time in between consecutive layer-one blocks. This restriction means that if the layer-one block gas limit increases, the adaptive cutoff is not reached on average before the pre-defined minimum cutoff duration Δ.

E. Summary

Lastly, in Table I we summarize all the adaptive cutoff parameters introduced in this section, and overview how they affect poll closure and statement finalization.

	Finality Time	Fraud Resilience	Transaction Cost
Δ	$\nearrow$	$\nearrow$	-
α	$\nearrow$	$\nearrow$	-
c	$\nearrow$	$\searrow$	$\nearrow$
p	$\searrow$	-	$\nearrow$

Table I: Adaptive Cutoff parameter effect summary. $\nearrow$ denotes a positive correlation, $\searrow$ denotes a negative correlation, while - denotes noncorrelation.

V. Probabilistic AC

In this section we show how a spokesperson can create a non-interactive computationally-sound proof [64](i.e. a non-interactive argument) to convince a smart-contract that an adaptive cutoff has been reached. Under this probabilistic approach, the smart-contract is first convinced, with overwhelming probability, that some *p*-gas was accumulated after the publication of a statement. Subsequently, the smart-contract calculates the minimum *c*-gas value possible for that value. If at a time $t > \Delta$ the smart-contract receives the AC argument, the cutoff is reached. For a poll, the smart-contract would stop accepting responses. For a statement, the smart-contract would accept it as final if there were no successful disputes and no open queries or claims as of time t. However, the cost of verifying this argument is non-negligible, and this probabilistic approach *increases* the expected dispute period duration due to its imprecision. However, this approach is compatible with the existing Ethereum Virtual Machine.

A. Proving System

Algorithm 1: ProvePGas

Input :
- maxGasPrice: p value for accumulating *p*-gas
- blocks: blocks from which to accumulate *p*-gas
- maxSize: maximum size of transaction to include
- numSamples: number of samples to open

Output:
- pGas: accumulated *p*-gas
- commitment: Merkle-sum commitment root
- samples: random samples opening information

```
1  pGas ← 0;
2  pgValues ← [];
3  pgData ← [];
4  for block ∈ blocks do
5      remGas ← block.gasLimit − block.gasUsed;
6      pGas ← pGas + remGas;
7      pgValues.append(remGas);
8      pgData.append((block.number, -1, 0));
9      for tx ∈ block.transactions do
10         if |tx| > maxSize
11         or tx.gasPrice > maxGasPrice then
12             continue;
13         end
14         pGas ← pGas + tx.gasUsed;
15         pgValues.append(tx.gasUsed);
16         data ← (block.number, tx.number, tx.gasPrice);
17         pgData.append(data);
18     end
19 end
20 commitment, openings ←MSMMCommit(pgData, pgValues);
21 samples ← [];
22 for i = 0 to numSamples − 1 do
23     g_i ← RO(commitment, pGas, i) mod pGas;
24     s ← 0;
25     for j = 0 to |pgValues| − 1 do
26         if s + pgValues[j] ≥ g_i then
27             samples.append((pgData[j], openings[j]));
28             break;
29         else
30             s ← s + pgValues[j];
31         end
32     end
33 end
34 return pGas, commitment, samples
```

Using Algorithm 1, a prover creates an argument that n random *p*-gas units out of the *p*-gas units claimed to have been accumulated are valid. The prover prepares the argument as follows: **In lines 1 to 19** the prover calculates the *p*-gas contributions of all reasonably-sized transactions following the statement. All transactions with a non-zero *p*-gas contribution

Algorithm 2: VerifyCGas

Input :
- maxGasPrice: p value for accumulating p-gas
- segmentSize: c value for accumulating c-gas
- blocks: blocks from which to accumulate p-gas
- pGas: accumulated p-gas
- commitment: Merkle-sum commitment root
- samples: random samples opening information
- λ: probability of error
- ρ: p-gas fraction claimed

Output:
- cGas: proven c-gas

```
1  n ← |samples|;
2  for i = 0 to n − 1 do
3      g_i ← RO(commitment, pGas, i) mod pGas;
4      weightPrefix, weightLeaf, data ←
         Open(commitment, samples[i]);
5      blockNum, txNum, gasPrice ← data;
6      block ← getBlock(blockNum);
7      remGas ← block.gasLimit − block.gasUsed;
8      tx ← getTransaction(blockNum, txNum);
9      if g_i < weightPrefix
10     or weightPrefix + weightLeaf < g_i
11     or gasPrice > maxGasPrice
12     or (txNum = −1 and remGas ≠ weightLeaf)
13     or tx.gasPrice ≠ gasPrice
14     or tx.gasUsed ≠ weightLeaf
15      then
16         return -1
17     end
18 end
19 if ρ^n > λ then
20     return -1
21 end
22 pgProven ← ρ × pGas;
23 incompleteSegments ← ⌊pgProven / (segmentSize − 1)⌋;
24 remainder ← pgProven mod (segmentSize − 1);
25 m_1 = |blocks| × (⌊incompleteSegments / |blocks|⌋ − 1);
26 m_2 = incompleteSegments mod |blocks|;
27 if m_1 ≤ 0 then
28     return 0
29 else if remainder = 0 then
30     return m_1 + m_2
31 else
32     return m_1 + m_2 + 1
33 end
```

are collected into a list sorted by order of transaction execution in the blockchain. **In line 20** the prover then creates a Merkle-sum-min-max (MSMM) tree commitment over this list, where the leaf weights are the p-gas contributions of each transaction, and the leaf values are (block number, transaction number) pairs, over which the minimum and maximum value annotations are calculated. Denoting the p-gas value claimed by b, **in lines 21 to 33** the prover generates a deterministic pseudo-random sequence of n elements from $\mathbb{Z}_b$ using any cryptographically acceptable realization of a Random Oracle (RO). With g_i denoting the i^{th} value in the generated sequence, the prover appends to the argument the opening information for the transaction which contributes the $g_i{}^{\text{th}}$ p-gas unit.

Given the argument, the smart-contract can use Algorithm 2 to derive a c-gas value from it. **In lines 1 to 18**, the smart-contract recalculates the list of g_i values using the same Random Oracle and commitment. It validates all the openings provided in the argument using the min-max tree annotations, while ensuring that all opened transactions in the commitment are relatively sorted by their order of execution in the blockchain, and are published after the target statement. **In lines 19 to 21**, with some security threshold λ fixed for the smart-contract, such as $\lambda = 2^{-128}$, the smart-contract then calculates the maximum possible number of valid p-gas units that the smart-contract is confident exist in the measurement. As the argument contains no information about how the p-gas units are distributed across continuous segments, the smart-contract calculates **in lines 22 to 33** the minimum amount of c-gas derivable from the provided blocks.

Notably, the smart-contract needs access to block and transaction information in order to verify the claimed p-gas values. In Ethereum, this can be accomplished using transaction and block inclusion proofs as the Ethereum Virtual Machine does not provide such built-in introspection.

B. Parameterization

The two primary parameters in this argument are (i) the number of samples in an proof, which is a dynamic parameter that the prover can decide per argument, and (ii) the security level λ, which dictates the level of confidence the smart-contract must have in the argument. In this section we describe the relationship between these two parameters and how they affect the percentage of p-gas the smart-contract can be confident to have been accumulated.

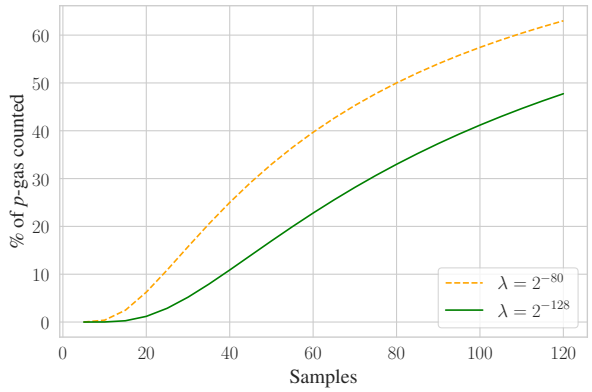

Figure 4: Number of correct samples provided in a proof versus the percentage of p-gas that may be safely assumed to exist with probability $1 - \lambda$ by the verifier.

Interestingly, by setting "maxSize" as input to Algorithm 1, the prover can control the maximum cost of verification for each sample, including transaction membership proofs, and can consequently derive the maximum cost of verification of the argument in the smart-contract using the information in Figure 4. This allows the prover to perform a trade-off between how much it is willing to wait for more p-gas to accumulate, and how much gas it is willing to spend to prove to the smart-contract that a certain portion of the p-gas that has been accumulated exists.

For example, by limiting the maximum cost of verification of a single sample to 100,000 gas, the prover can include any

transaction of size 1 kilobyte or less in its argument, and by including 40 samples, it can control the argument verification cost to be on the order of 4,000,000 gas. Smart-contracts with $\lambda = 2^{-80}$ would accept roughly 25% of the claimed *p*-gas value, while those with $\lambda = 2^{-128}$ would accept roughly 10%.

C. Periodic Aggregation

Unfortunately the present-day EVM does not allow smart-contracts to directly access hashes of blocks in the chain that are older than 256 blocks. This means that to utilize the our proving system, the AC implementing contract must be periodically invoked to aggregate the hashes of the last newly found canonical chain blocks for future proof verification. In case the suffix of blocks in the chain that are not currently aggregated grows beyond 256 in length, the bandwidth in unaggregated blocks older than 256 in height cannot be efficiently captured, as the entire blocks must be provided in order to manually recalculate their hashes.

VI. Disputable AC

The probabilistic approach to AC will slowly capture the gas that could have been used for C-TX publication. In this section we present a mechanism that allows a spokesperson to capture gas more efficiently using a statement that can be efficiently disputed using a bisection based approach, in combination with our non-interactive proof from Section IV.

A. Disputable Measurement Statements

To efficiently prove that sufficient c-gas had been accumulated, the spokesperson publishes our measurement statement, denoted by a_x, which is comprised of the hash h_x of aggregated block number x, and the c-gas that passed by the end of block x since the prior β number of aggregated blocks. We denote by β the measurement query upper-bound, which restricts how many aggregated blocks a single statement can cover. However, we also permit the spokesperson to make multiple non-intersecting statements simultaneously, and sum up their results. Once published, the measurement is unusable and disputable for a minimum time period of Δ.

Disputes against measurement statements follow a *bisection-based approach* similar to those employed in [1, 65], but specialized for disputing measurements. Under DC, a dispute against a block measurement is a query of the form x, y, to force publication of an intermediate measurement as of aggregated block number $\lfloor \frac{x+y}{2} \rfloor$. If an intermediate measurement is invalid, recursively applying this dispute method allows users to detect the earliest incorrect intermediate block measurement, such that another dispute procedure, which we will describe shortly, allows users to prove c-gas value incorrectness.

Consequently, using β, we place a limit on the maximum number of queries needed to dispute a measurement. The number of queries required by one user is at most $O(\log \beta)$. However, if many users query the same x, y range, we modify their redundant queries to instead inquire about an undisputed sub-range of x, y. This diversion means that if $O(\beta)$ users each query a measurement once, then the spokesperson will have to reveal all of the intermediate values, exposing the anomalous intermediate measurement.

Subsequently, when the earliest incorrect block measurement a_x is detected, users then dispute the transaction gas prices in block x. Transaction price disputes target the p-gas measurement after the execution of the i^{th} transaction in block x, in one of three ways: (i) a query of the form a, b, on the p-gas value after executing the i^{th} transaction in the block, where $i = \lfloor \frac{a+b}{2} \rfloor$, forcing the spokesperson to answer this query with a hash of the layer-one transaction, a hash of its receipt , and the p-gas measurement after its execution; (ii) a fraud proof that the published measurement after the i^{th} transaction is incorrect; or (iii) a claim that the i^{th} transaction size is too large to validate the measurement of in the smart-contract.

A single user only needs $\lceil \log(t) \rceil$ queries before being able to prove fraud or issue a claim, where t is the number of transactions in a block. Given a block size of 10,000,000 gas units in Ethereum, and a minimum transaction gas cost of 21,000 units, only 476 transactions can fit in a block. A user then would need 9 queries, or 476 users would need a single query each, when applying the redundant query diversion tactic. To prove fraud, a user reveals the data behind the transaction and receipt hashes, showing that the gas price was above p, or that transaction gas usage was less than claimed. If the operator publishes an incorrect hash, then a user can prove fraud by revealing the block header and a membership proof of the correct hashes of the i^{th} transaction.

Even though a transaction does not need to be re-executed, it could be too large to submit as a fraud proof. A claim that the verification cost exceeds c can be refuted by verifying the transaction at a cost at most c, a feat that can be done in Ethereum through leveraging the *gasleft* function.

B. Finalization

A questions that arises is how can we determine the dispute period on a measurement statement, which in itself determines the dispute period on other statements? We leverage the probabilistic AC method to finalize our disputable AC statements. Despite still depending on the probabilistic approach, this allows the decoupling of the gas price which the measurement statements target from the gas price which is used to cutoff disputes on the measurement statements themselves. The upside to this approach is that the p value for gas accumulation for measurement statements can be set to a higher level than the p value which the measurements themselves account for. Furthermore, the c value necessary for disputes in the bisection based approach is smaller than those necessary for some state-of-the-art protocols. Under larger p values, and smaller c values, this enables faster statement finalization using probabilistic AC. Furthermore, in a setting where different protocols utilize a single contract dedicated to pooling AC/DC p-gas proofs and c-gas measurements, this approach can be promising for multiple protocols to share block aggregation, measurement, and non-interactive argument costs.

VII. Evaluation

In this section we evaluate an implementation of the AC/DC mechanisms, focusing on the cost and effectiveness of the AC mechanism in protecting layer-two protocol users, and on the complexity of disputing DC measurements. The implementation is written in Solidity 0.6.6 and JavaScript, and is open-source[3]. Gas costs are sampled on a locally deployed test network. While our evaluation is focused on layer-two security, our implementation can be leveraged in other systems to determine and prove that sufficient opportunity was available to publish critical transactions through the *verifyCGas* method.

A. Probabilistic AC

1) Cost: The base smart-contract gas costs for AC proof verification is approximately 35,000 gas. It costs on average 426,000 gas per transaction inclusion sample. Furthermore, it costs roughly 490,005 to aggregate 256 blocks and commit their hashes. This brings the estimated total cost of an AC proof which includes m transaction inclusion samples and n aggregated blocks to $\lfloor \frac{n}{256} \rfloor \times 490005 + m \times 426000 + 35000$.

2) Effectiveness: In Figure 5 we plot the percentage of successful fraud attempts under AC, compared to the layer-two systems with static-dispute periods, NOCUST and OmiseGO. We simulate $n = 1{,}000{,}000$ registered users with a single layer-two operator, and an Ethereum layer-one blockchain where blocks have a limit of 10,000,000 gas units, and are produced once every 20 seconds. The variable $\mathcal{R}_S$ represents the average rate per block at which the spokesperson publishes statements in the smart-contract, and $\mathcal{R}_D$ represents the average rate per block at which these statements are disputed. We run the simulation with a fixed dispute period of $\Delta = 24$ hours for NOCUST and OmiseGO, and use the same Δ as the minimum cutoff for our AC ledger.

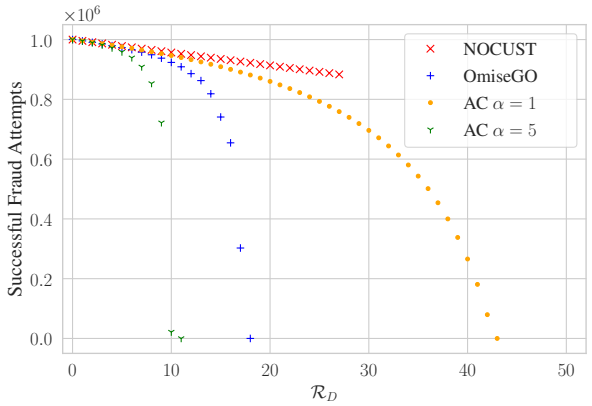

Figure 5: Plot of estimated successful fraud attempts for 1,000,000 attempts under different values for the average number of disputes per block $\mathcal{R}_D$ in AC, NOCUST and OmiseGO. NOCUST permits the operator to perform a high amount of fraud due to its over reliance on static-time queries, while OmiseGO is inflexible with how slowly withdrawals can be disputed.

[3]https://github.com/rami-github/adaptive-dispute-cutoffs

For AC and NOCUST, $\mathcal{R}_S = n$ in the first block, and $\mathcal{R}_S = 0$ thereon after, as the spokesperson can publish a single commitment statement which affects all registered user accounts at once. For NOCUST, we consider a maximum $\mathcal{R}_D = 27$, which is the maximum rate whereby all blocks are filled with queries and their responses. For AC we consider a maximum $\mathcal{R}_D = 52$, as at most 52 claims and their refutations can fit inside a block, while no queries are required.

In payment-channels and UTXO-based Plasma-schemes, fraudulent channel closures and fraudulent withdrawals can only be attempted per channel or transaction output respectively. Therefore, we examine the resulting successful fraud attempts while considering a maximum $\mathcal{R}_S = 29$ based the cost for a standard withdrawal initialization being on the order of an average 336,000 gas in the UTXO Plasma-based OmiseGO Network. We also use a maximum value of $\mathcal{R}_D = 49$ for the OmiseGo platform, since its challenge cost is on the order of an average 200,000 gas. As both ongoing withdrawal initialization and their challenges share the layer-one bandwidth, we vary the value of $\mathcal{R}_D$ and derive the value of $\mathcal{R}_S$ based on the remaining bandwidth.

Figure 5 illustrates the impact of the two different values of α on the estimated successful fraud attempts in AC. We then see that $\alpha = 5$ requires $\mathcal{R}_D = 10$ for zero fraud, while $\alpha = 1$ requires a higher user dispute rate of $\mathcal{R}_D = 42$.

B. Disputable AC

As we place an upper-bound β on the number of blocks that can be included in a measurement, this enforces a reasonable upper-bound on the number of C-TXs necessary to dispute a measurement. In Figure 6 we plot the number of blocks required for a measurement to be disputed under different user-base conditions in the worst-case. We assume a very small percentage of gas is available for measurement dispute ($e = 0.95$), while a large amount of repeated queries take place, stressing our dispute diversion tactic.

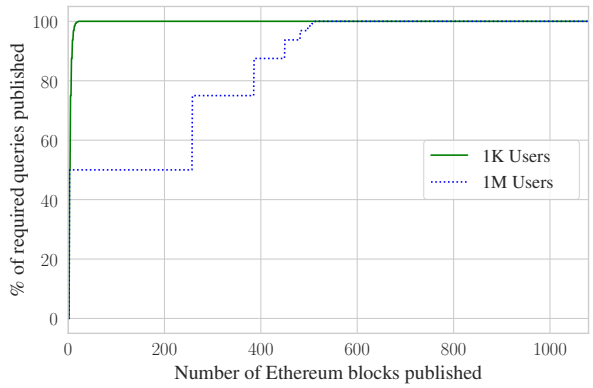

Figure 6: Plots of query publication simulations for disputable measurement statements with different user counts. The left axis represents the percentage of queries successfully published in the smart-contract. The bottom axis represents the number of published blocks in the layer-one blockchain Ethereum. New queries are evenly distributed across blocks. Query responses take precedence over new queries, while only 5% of block gas is used for disputes, and $\beta = 512$.

VIII. Conclusion

We have described two mechanisms that increase the robustness and security of smart-contracts that depend on critical transactions. The first mechanism enables adaptive cutoffs (ACs) using a non-interactive argument system that can be implemented in the Ethereum Virtual Machine. The second mechanism, disputable cutoffs (DCs), relies on efficient bisection-based disputes, and the AC mechanism, to provide more efficient critical transaction cutoffs. We showed that a second-layer account-based Plasma protocol with adaptive cutoffs can surpass non-adaptive counterparts in preventing fraud, securing $1{,}000{,}000$ accounts under layer-one congestion. Furthermore, we showed that DCs remain secure under a significant amount of layer-one congestion, and poor coordination between users on measurement disputes.

Limitations. Two primary bottlenecks exist in the design of AC/DC which affect its efficiency and cost-effectiveness. First, Algorithm 2 calculates the *minimum* amount of c-gas, and not the actual amount, which detracts from the effectiveness of the system in accounting for all of the publication opportunity available. Second, the sampling-based approach in Algorithms 1 and 2 results in relatively expensive to verify proofs, as shown in Section VII.

Future Work. One direction for future work is to use an efficient zero-knowledge proving system to reduce the gas cost of verifying c-gas accumulation. Another direction is to design a framework which enables AC/DC for payment-channels, such as Lightning [41]. Furthermore, the effects of transaction ordering schemes on dispute opportunity should be explored [66].

Acknowledgment

We'd like to thank our anonymous reviewers for their valuable feedback on this paper. This work was made possible with the generous funding and support of the Imperial College London President's PhD Scholarship program.

References

[1] H. Kalodner, S. Goldfeder, X. Chen, S. M. Weinberg, and E. W. Felten, "Arbitrum: Scalable, private smart contracts," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 1353–1370.

[2] Z. Shi, C. de Laat, P. Grosso, and Z. Zhao, "When blockchain meets auction models: A survey, some applications, and challenges," *arXiv preprint arXiv:2110.12534*, 2021.

[3] A. Kiayias and P. Lazos, "Sok: Blockchain governance," *arXiv preprint arXiv:2201.07188*, 2022.

[4] S. M. Werner, D. Perez, L. Gudgeon, A. Klages-Mundt, D. Harz, and W. J. Knottenbelt, "Sok: Decentralized finance (defi)," *arXiv preprint arXiv:2101.08778*, 2021.

[5] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais, "Sok: Layer-two blockchain protocols," in *International Conference on Financial Cryptography and Data Security*. Springer, 2020, pp. 201–226.

[6] "Makerdao whitepaper," https://makerdao.com/en/whitepaper/, 2020, online; Accessed 19-July-2022.

[7] "Minimal viable merged consensus," https://ethresear.ch/t/minimal-viable-merged-consensus/5617.

[8] "Bitcoin wiki: Payment channels." https://en.bitcoin.it/wiki/Payment_channels.

[9] J. Poon and V. Buterin, "Plasma: Scalable autonomous smart contracts," *White paper*, 2017.

[10] R. Khalil, "NOCUST-A non-custodial 2nd-layer blockchain payment hub," Master's thesis, Swiss Federal Institute of Technology, Zurich, 2018, https://pub.tik.ee.ethz.ch/students/2018-FS/MA-2018-24.pdf.

[11] J. Poon and O. Team, "Decentralized exchange and payments platform," 2017.

[12] "Liquidity.network whitepaper," https://www.allcryptowhitepapers.com/liquidity-network-whitepaper/, 2019, online; Accessed 19-July-2022.

[13] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, 2014.

[14] D. Perez and B. Livshits, "Broken metre: Attacking resource metering in evm," in *Proceedings of the 29th Annual Network and Distributed System Security Symposium, NDSS*, 2022.

[15] S. M. Werner, P. J. Pritz, and D. Perez, "Step on the gas? a better approach for recommending the ethereum gas price," in *Mathematical Research for Blockchain Economy*. Springer, 2020, pp. 161–177.

[16] R. Zhang and B. Preneel, "Lay down the common metrics: Evaluating proof-of-work consensus protocols' security," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 175–192.

[17] P. McCorry, A. Hicks, and S. Meiklejohn, "Smart contracts for bribing miners," in *International Conference on Financial Cryptography and Data Security*. Springer, 2018, pp. 3–18.

[18] J. Bonneau, "Why buy when you can rent?" in *International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 19–26.

[19] K. Liao and J. Katz, "Incentivizing blockchain forks via whale transactions," in *International conference on financial cryptography and data security*. Springer, 2017, pp. 264–279.

[20] A. Judmayer, N. Stifter, A. Zamyatin, I. Tsabary, I. Eyal, P. Gazi, S. Meiklejohn, and E. Weippl, "Pay to win: cheap, crowdfundable, cross-chain algorithmic incentive manipulation attacks on pow cryptocurrencies," *Cryptology ePrint Archive*, 2019.

[21] A. Judmayer, N. Stifter, A. Zamyatin, I. Tsabary, I. Eyal, P. Gaži, S. Meiklejohn, and E. Weippl, "Sok: Algorithmic incentive manipulation attacks on permissionless pow cryptocurrencies," in *International Conference on Financial Cryptography and Data Security*. Springer, 2021, pp. 507–532.

[22] F. Winzer, B. Herd, and S. Faust, "Temporary censorship attacks in the presence of rational miners," in *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2019, pp. 357–366.

[23] Y.-H. Chen, S.-H. Chen, and I.-C. Lin, "Blockchain based smart contract for bidding system," in *2018 IEEE International Conference on Applied System Invention (ICASI)*. IEEE, 2018, pp. 208–211.

[24] C. Braghin, S. Cimato, E. Damiani, and M. Baronchelli, "Designing smart-contract based auctions," in *International conference on security with intelligent computing and big-data services*. Springer, 2018, pp. 54–64.

[25] C. Pop, M. Prata, M. Antal, T. Cioara, I. Anghel, and I. Salomie, "An ethereum-based implementation of english, dutch and first-price sealed-bid auctions," in *2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP)*. IEEE, 2020, pp. 491–497.

[26] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels, "Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 910–927.

[27] C. Decker and R. Wattenhofer, "A fast and scalable payment network with bitcoin duplex micropayment channels," in *Symposium on Self-Stabilizing Systems*. Springer, 2015, pp. 3–18.

[28] S. Werman and A. Zohar, "Avoiding deadlocks in payment chan-

nel networks," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Springer, 2018, pp. 175–187.
[29] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, and M. Maffei, "Anonymous multi-hop locks for blockchain scalability and interoperability." in *NDSS*, 2019.
[30] E. Tairi, P. Moreno-Sanchez, and M. Maffei, "A2l: Anonymous atomic locks for scalability and interoperability in payment channel hubs," Cryptology ePrint Archive, Report 2019/589, Tech. Rep., 2019.
[31] D. Hosp, T. Hoenisch, P. Kittiwongsunthorn *et al.*, "Comit-cryptographically-secure off-chain multi-asset instant transaction network," *arXiv preprint arXiv:1810.02174*, 2018.
[32] D. Piatkivskyi and M. Nowostawski, "Split payments in payment networks," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Springer, 2018, pp. 67–75.
[33] Y. Zhang, Y. Long, Z. Liu, Z. Liu, and D. Gu, "Z-channel: Scalable and efficient scheme in zerocash," *Computers & Security*, pp. 112–131, 2019.
[34] A. Zamyatin, D. Harz, J. Lind, P. Panayiotou, A. Gervais, and W. Knottenbelt, "Xclaim: Trustless, interoperable, cryptocurrency-backed assets," *IEEE Security and Privacy. IEEE*, 2019.
[35] G. Malavolta, P. Moreno-Sanchez, A. Kate, M. Maffei, and S. Ravi, "Concurrency and privacy with payment-channel networks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 455–471.
[36] E. Heilman, L. Alshenibr, F. Baldimtsi, A. Scafuro, and S. Goldberg, "Tumblebit: An untrusted bitcoin-compatible anonymous payment hub," in *Network and Distributed System Security Symposium*, 2017.
[37] C. Decker, R. Russell, and O. Osuntokun, "eltoo: A simple layer2 protocol for bitcoin," *White paper: https://blockstream. com/eltoo. pdf*, 2018.
[38] V. Sivaraman, S. B. Venkatakrishnan, K. Ruan, P. Negi, L. Yang, R. Mittal, G. Fanti, and M. Alizadeh, "High throughput cryptocurrency routing in payment channel networks," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, 2020, pp. 777–796.
[39] R. Khalil and A. Gervais, "Revive: Rebalancing off-blockchain payment networks," *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017.
[40] P. McCorry, S. Bakshi, I. Bentov, S. Meiklejohn, and A. Miller, "Pisa: Arbitration outsourcing for state channels," in *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, 2019, pp. 16–30.
[41] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," https://lightning.network/lightning-network-paper.pdf, 2016.
[42] M. M. Chakravarty, S. Coretti, M. Fitzi, P. Gazi, P. Kant, A. Kiayias, and A. Russell, "Hydra: Fast isomorphic state channels," Cryptology ePrint Archive, Report 2020/299, Tech. Rep., 2020.
[43] C. Buckland and P. McCorry, "Two-party state channels with assertions," in *International Conference on Financial Cryptography and Data Security*. Springer, 2019, pp. 3–11.
[44] E. Wagner, A. Völker, F. Fuhrmann, R. Matzutt, and K. Wehrle, "Dispute resolution for smart contract-based two-party protocols," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2019, pp. 422–430.
[45] A. R. Pedrosa, M. Potop-Butucaru, and S. Tucci-Piergiovanni, "Scalable lightning factories for bitcoin," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 2019, pp. 302–309.
[46] P. McCorry, C. Buckland, S. Bakshi, K. Wüst, and A. Miller, "You sank my battleship! a case study to evaluate state channels as a scaling solution for cryptocurrencies," in *International Conference on Financial Cryptography and Data Security*. Springer, 2019, pp. 35–49.
[47] S. Dziembowski, L. Eckey, S. Faust, J. Hesse, and K. Hostáková, "Multi-party virtual state channels," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2019, pp. 625–656.
[48] S. Dziembowski, S. Faust, and K. Hostáková, "General state channel networks," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 949–966.
[49] S. Dziembowski, L. Eckey, S. Faust, and D. Malinowski, "Perun: Virtual payment hubs over cryptocurrencies," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 106–123.
[50] A. Miller, I. Bentov, S. Bakshi, R. Kumaresan, and P. McCorry, "Sprites and state channels: Payment networks that go faster than lightning," in *International Conference on Financial Cryptography and Data Security*. Springer, 2019, pp. 508–526.
[51] J. Harris and A. Zohar, "Flood & loot: a systemic attack on the lightning network," in *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, 2020, pp. 202–213.
[52] Z. Avarikioti, E. Kogias, R. Wattenhofer, and D. Zindros, "Brick: Asynchronous incentive-compatible payment channels," in *International Conference on Financial Cryptography and Data Security*, 2021.
[53] J. Lind, O. Naor, I. Eyal, F. Kelbert, P. R. Pietzuch, and E. G. Sirer, "Teechain: Reducing storage costs on the blockchain with offline payment channels," in *Proceedings of the 11th ACM International Systems and Storage Conference, SYSTOR 2018, HAIFA, Israel, June 04-07, 2018*, 2018, p. 125.
[54] J. Lind, I. Eyal, P. Pietzuch, and E. G. Sirer, "Teechan: Payment channels using trusted execution environments," *arXiv preprint arXiv:1612.07766*, 2016.
[55] R. Khalil, P. Moreno-Sanchez, A. Zamyatin, A. Gervais, and G. Felley, "Commit-chains: Secure, scalable off-chain payments," Cryptology ePrint Archive, Report 2018/642, Tech. Rep., 2019.
[56] S. Dziembowski, G. Fabiański, S. Faust, and S. Riahi, "Lower bounds for off-chain protocols: Exploring the limits of plasma," in *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*, 2021.
[57] "On-chain scaling to potentially 500 tx/sec through mass tx validation," https://ethresear.ch/t/on-chain-scaling-to-potentially-500-tx-sec-through-mass-tx-validation/3477/5.
[58] "barrywhitehat. roll_up: Scale ethereum with snarks," https://github.com/barryWhiteHat/roll_up.
[59] I. Meckler and E. Shapiro, "Coda: Decentralized cryptocurrency at scale," 2018.
[60] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, "Recursive composition and bootstrapping for snarks and proof-carrying data," in *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM, 2013, pp. 111–120.
[61] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Scalable zero knowledge via cycles of elliptic curves," *Algorithmica*, vol. 79, no. 4, pp. 1102–1160, 2017.
[62] V. Buterin, E. Conner, R. Dudley, M. Slipper, I. Norden, and A. Bakhta, "Eip 1559: Fee market change for eth 1.0 chain," https://github.com/ethereum/EIPs/blob/master/EIPS/eip-1559.md.
[63] "Metamask stops working as infura suffers another service outage," https://www.theblock.co/post/143047/metamask-stops-working-as-infura-suffers-another-service-outage.
[64] S. Micali, "Cs proofs," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 1994, pp. 436–453.
[65] J. Teutsch and C. Reitwießner, "A scalable verification solution for blockchains," *arXiv preprint arXiv:1908.04756*, 2019.
[66] A. Orda and O. Rottenstreich, "Enforcing fairness in blockchain transaction ordering," *Peer-to-peer Networking and Applications*, vol. 14, no. 6, pp. 3660–3673, 2021.