Ideal Properties of Rollup Escape Hatches

Jan Gorzny, Lin Po-An, Martin Derka

October 18, 2022

Abstract

A rollup is a type of popular "layer two" scaling solution for slow-but-secure blockchains like Ethereum. A rollup perfoms computation of blockchain state updates off-chain but posts the inputs and the data to the underlying blockchain in order to benefit from its security. However, if rollup operators go offline, further state updates are no longer possible through the rollup; instead, state updates to the layer two state must be forced on the underlying blockchain. Such a mechanism is called an *escape hatch* as it allows state, and in particular digital assets, to escape from an inoperative rollup. We review the approaches from rollups developed by the community and highlight potential issues. We also establish a wishlist of properties that an escape hatch mechanism should have to be considered trustworthy and compatible with decentralization.

1 Introduction

The limited throughput provided by blockchains like Ethereum has motivated the design of scalability solutions for these systems [1]. These often take the form of so-called called *layer two* (or *off-chain*) protocols. This name comes from the fact that they are built on top of a blockchain, which is called the *layer one* for an ecosystem [1]. In a nutshell, layer two protocols allow transacting parties to utilize a high-throughput blockchain ecosystem and only fall back to the underlying layer one if there is a need to dispute behavior on the second layer (e.g., with respect to their cryptocurrency balances), interact with decentralized applications on layer one, or transact with accounts on other layer two instances.

There are several flavors of layer two protocols. The different design patterns offer an interesting tradeoff between several crucial aspects such as compatibility with the underlying blockchain, liquidity, collateralization, security and privacy, among others (see e.g., [1]). Protocols that leverage the notion of payment channels, which are generalized either in the form of payment-channel networks or payment-channel hubs are popular on the Bitcoin Network [2, 3, 4, 5]. Some layer two systems for Ethereum are based on side-chains (see e.g., [6]) or the *Plasma* protocol [7]. Recently, *rollups* (also called *commit-chains* [8]) have become a popular off-chain scaling solution for Ethereum (and other blockchains). Rollups overcome data-availability issues with Plasma (see e.g., [9, 10]), and also support *smart contracts*: automatically executing programs. We consider layer one blockchains, like Ethereum, that support smart contracts; such smart contracts are required to implement rollups.

Rollups that support smart contracts are essentially middleware. They often aim to mimic the behaviour of their underlying layer one, but may change the semantics of some system calls and add or remove features. For example, the Boba rollup intends to add support for calling off-chain oracles in smart contracts deployed on it [11]. More generally, the StarkNet [12] rollup will use the Cairo virtual machine, completely disjoint from the Ethereum virtual machine, which powers its underlying layer one; this will come will a host of new features, like additional cryptographic primitives [13].

Several implementations of rollups have been deployed and are seeing a growing number of users and transactions. Rollups come in two main flavors, defined in Section 2: *optimistic* and *zero-knowledge* (ZK).

In this work, we focus on so-called *escape hatches* for rollups. An escape hatch is method by which users of a rollup can recover digital assets or program state from a rollup when the operators are offline. Escape hatches for rollups are not required to deal with malicious operators as these are guarded against by so-called *fraud proofs* (in optimistic rollups) or *validity proofs* (in ZK rollups). However, if operators fail to publish updates to rollup state, user assets (like cryptocurrencies or ERC-20 tokens [14]) may be locked in the rollup system. These assets may be locked as the protocol no longer responds to transactions to transfer these assets. Therefore methods to unlock these funds — or more generally, transfer rollup state — are necessary to overcome the initial hardships of newer systems and the dangers of untrusted rollup operators. When a rollup is decentralized and anyone can act as an opera-

tor, this requirement may disappear, but there are no decentralized rollups operating at the time of writing.

Escape hatches have been underdeveloped in favor of advancing the core technologies of these systems. Some rollups, like Optimism [15], have implemented this feature. Optimism allows anyone to force a transaction to be included into (and update) the layer two state; submitting a withdraw transaction in this manner provides an escape hatch. In other cases, like for the StarkNet ZK rollup [12], escape hatches may need to be application specific [16]. We argue that this feature is critical for users and is technologically challenging.

For users, there are no guarantees in centralized systems that operators will be online, despite their best efforts. In decentralized settings, there may be incentive for liveness, but this may also be insufficient; if the operator requires special hardware to generate a ZK proof, significant downtime may be possible and capable operators may be in short supply.

On the technical side of things, even in limited settings, existing techniques are not clearly applicable for a fully robust layer two. For example, available cross-blockchain payment solutions such as atomic swaps do not help the off-chain interoperability problem either since they require that both sender and receiver are part of both systems [17].

As a result of hurried rollup adoption, the security implications of these systems are not fully understood. While most rollup functionality aims to be equivalent or at least compatible with the functionality of the underlying layer one, this is not always the case. Early versions of the Optimism rollup changed the semantics of computation by changing how the native cryptocurrency was represented and could have resulted in minting additional cryptocurrency [18]. Moreover, if such changes also impact how decentralized applications are to be developed, as may be the case for escape hatches on StarkNet [12], there may be new risks to end-users if the changes are not well understood. Understanding the security implications of escape hatches is an important part of understanding the security of rollups at large.

In this paper, we outline a framework for evaluating escape hatches on rollups. We consider the approaches discussed by the community and highlight potential issues. We establish a wishlist of properties that an escape hatch mechanism should have to be considered trustworthy and to be compatible with decentralization, including decentralized governance. We emphasize that as middleware, rollups should have more features than their corresponding layer one, not fewer.

2 Preliminaries

We first introduce *optimistic* and *zero-knowledge* (ZK) rollups.

In an optimistic rollup, computation is assumed correct and users can submit so-called *fraud proofs* to challenge incorrect computations. Optimistic rollups operators are bonded so that they are incentivized to avoid losing such challenges (i.e., they are incentivized to operate honestly and correctly). Arbitrum is one example of an optimistic rollup [19].

Zero-Knowledge (ZK) rollups replace fraud proofs with so-called *validity proofs*. In a ZK rollup, system operators execute a state transition within a zero-knowledge proof framework (e.g., [20]) which generates the validity proof: an artifact that proves that a particular function was executed with particular inputs which resulted in the new state. The "zero-knowledge" aspect of these proofs are sometimes helpful for privacy, but mostly these systems are used because the proofs are also *succinct*. This property enables the proofs to be verified in a fraction of the time required to run the computation in the firt place, enabling verification directly on a layer one blockchain. zkSync is an example of a ZK rollup [21].

A rollup can be broken down into several components. A rollup typically has a *sequencer* component, which is responsible for ordering layer two transactions. Possibly separate, the rollup will have an *executor*, responsible for executing the transactions on layer two. For ZK rollups, the rollup may also have a disjoint *prover*, which generates a proof that the executor correctly computed the state update. Finally, the rollup will have various smart contracts to interface with these components.

A sequencer orders transactions for the layer two. The source of these transactions may be a user of the rollup or the layer one smart contracts of the rollup. As a result, sequencers are responsible for *cross-chain* communication, and may be considered to be a blockchain *bridge*. A bridge is a system or protocol for taking assets or blockchain state from one blockchain to another. As cryptographic assets cannot be literally moved from one blockchain to another, the bridge creates representations of assets on a *source* blockchcain on a *destination* blockchain. To avoid arbitrary minting of assets, bridges have a smart contract on the source blockchain called the *custodian*, which locks up the asset to be minted on the destination chain. Through an offchain *communicator* component, when assets are placed in the custody of the bridge, the corresponding *debt issuer* on the destination blockchain mints a representation of the asset in custody. The process can be reversed. Ad-

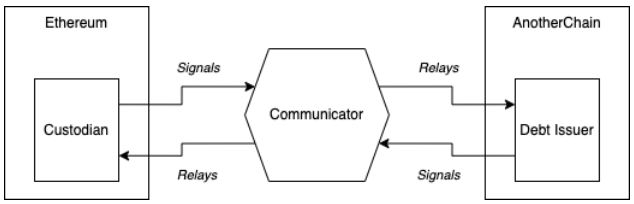

Figure 1: Components of a bridge, in this case from Ethereum to the fictitious "Another Chain" blockchain. The *custodian* (a smart contract on Ethereum) holds funds, the *communicator* relays signals between blockchains, and the *debt issuer* (a smart contract on Another Chain) creates representations of assets held by the custodian. The custodian signals the communicator when an asset is deposited in order to be issued on the destination blockchain, and the communicator relays this by calling a function on the debt issuer contract (or vice-versa, to move assets the other way).

vanced bridges can relay instructions to execute arbitrary functions on either blockchain. Bridges are illustrated in Figure 1.

Escape hatches are considered to be part of rollup functionality (called validating bridges) in [22]. Three important mechanisms are considered relating to escape hatches:

- A "forced transaction inclusion" mechanism. This is primarily discussed in the context of censorship resistance, but we note that an offline operator is akin to an operator who is censoring all transactions.
- A "value-transfer's escape hatch" mechanism, which allows accounts to withdraw digital assets through the aforementioned forced transaction inclusion mechanism. This requires users to submit a transaction to withdraw, and as such, cannot deal with transferring smart contract state in general.
- A "smart contract functionality and enforced liveness" mechanism. This is the most general form of escape hatches, and allows users to first withdraw balances from smart contracts on layer two before invoking the aforementioned value-transfer escape hatch. However, this may be infeasible if those executions cannot be replayed via forced inclusion on the layer one.

Ideally, the escape hatches we will discuss enforce a kind of *liveness*: the property that every transaction will eventually execute [22]. For a rollup

that satisfies the liveness property, escape hatches are not necessary. Thus we will consider rollups which fail to be live, and discuss escape hatches in their full generality, i.e., beyond mere asset withdrawal.

3 Idealized Properties

In this section, we highlight idealized properties for rollup escape hatch. Many of these properties may not be feasible with off-the-shelf ideas and may require novel solutions. For each property, we suggest some possible approaches and we reference existing implementations. The reference implementations aim to justify particular approaches and allow others to understand the existing technology when considering new methods. Table 1 lists several rollups and their escape hatch mechanisms, which will be discussed in the context of the following properties.

3.1 Basic Properties

Modular It may be beneficial to make escape hatches modular in order to replace such a mechanism. For example, a forced transaction inclusion may be simplest to implement and attractive for a first implementation but may be replaced by more sophisticated approaches.

Secure Escape hatches should be reviewed and tested. Bridges are increasingly targeted for exploitation as these systems hold more and more funds. As an escape hatch may interact with the custodian, communicator, and the debt issuer, there is a large surface area for attacks.

Correcting Escape hatches should not reintroduce the issues of escaping from a rollup. The state should not be migrated to another blockchain where another escape hatch may soon be required.

3.2 Support for Arbitrary State Escape

Ideally, more than a user's balance should be exit-able. This is particularly concerning as a user's balance is not always clear. It is likely that users may expect that a native currency on the rollup, like (wrapped) Ether, along with any ERC-20 tokens, constitute a part of the user's balance on the rollup.

Rollup	Escape Hatch Mechanism
Arbitrum Nova [23]/One [24]	Transact Using L1
Aztec [25] (Connect [26])	Propose Blocks* (ZK)
Boba Network [11]	Transact Using L1
dYdX [27]	Force Exit to L1
Fuel (v1) [28]	Propose Blocks
ImmutableX [29]	Force Exit to L1
Layer2.Finance [30]	*None*
Layer2.Finance-zk [31]	Force Exit to L1
Loopring [32]	Force Exit to L1
Metis Andromeda [33]	Transact Using L1
Optimism [15]	Transact Using L1
Polygon Hermez [34]	Force Exit to L1*
rhino.fi [35]	Force Exit to L1
Sorare [36]	Force Exit to L1
StarkNet [12]	*None*
ZKSpace (ZKSwap) [37]	Force Exit to L1
zkSync (v1) [21]	Force Exit to L1

Table 1: Escape hatches for various layer two solutions according to L2Beat.com [38] as of August 2022. We do not distinguish between so-called *Validium* solutions and ZK rollups, as they are similar except that the former is not required to store data on-chain along with their validity proofs.

However, this may omit state that a user considers as part of their balance, explicitly or implicitly. For example, an explicit asset that may be a part of the user's balance are non-fungible tokens (NFTs). These NFTs may be digital art or other assets that are to be bought, held, and sold, and may be representations of deposits in DeFi protocols (e.g., as in Uniswap V3 [39]). However, some DeFi protocols accept and record deposits using address mappings rather than token issuance; in these cases, transferring that record is also crucial for users to recover all of their assets. This may require those applications' invovlment in the exit process, i.e., support for arbitrary transaction inclusion on the layer two. In short, exiting only fungible assets does not allow a user to completely escape the system.

To avoid having to make distinctions regarding relevant valuable state and irrelevant worthless state, the escape hatch should migrate any part of state of the rollup (see also the "Global" of Sec. 3.5). Otherwise, a state

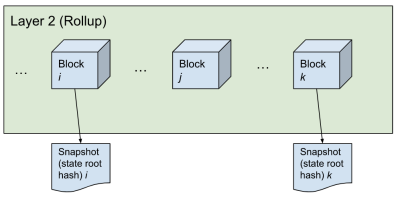

Figure 2: Proposing new blocks in a rollup. Snapshot i, if it exists, contains the state root (Merkle root). However, the data within the state is not obvious. The transactions of block j may change the state, but the (non-hashed) state updates are only known to someone verifying Snapshot k if they know the pre-images for Snapshot i and watch the data in block j. Without the non-hashed updates, it's not only impossible to provide an incorrect snapshot, but it's also unclear what the state contains.

exit should include state that is locked in additional layers (like a rollup in a rollup), games and applications that do not tokenize assets, or general data for applications built on the rollup, and chosen by users according to their needs.

The entries in Table 1 which allow new blocks to be proposed are the most similar to this goal. In these cases, the data and state are not actually moved; instead the chain continues to operate. In turn, data can be escaped through the usual methods; in this way, such an rollup is always live though the operator may change (see also Sec. 2 and Sec. 3.6). In [22], the authors suggest that honest operators can appoint themselves in this kind of situation, though the change of operators may be rate-limited. A rate-limited change process is important to enable some level of consensus and prevent griefing by always changing operators (and in turn, possibly system configurations).

However, not all rollups may wish to support this approach. Depending on the type of ZK proof system used within a ZK rollup, the hardware requirements to run a prover may be too much for end-users [40]. In times of sequencer failure, it may be difficult or impossible to quickly source the hardware (and configure it along with the proof generation software) required to propose new blocks. In the optimistic setting, even being able to update the last stateroot may be difficult. The state root commitment is known as part of the design, the explicit state in the Merkle tree is not known (see Figure 2). Without this information, it may be difficult to propose new blocks (without being considered malicious). In this case, a full archive node may

need to be sourced to collect the relevant data and rebuild the state, which can take a long time. Moreover, the escape hatch itself will need to publish the data for the state migration as it is used. Another approach that has not yet been attempted is to consider cross-chain state machine migration (see e.g., [41]) for duplication of state.

Rollups can handle the scenarios above from their outset by using a "permisionless set of executors" [22]. This approach would mean that a single sequencer failure may not even be noteworthy, as others are able to immediately play the part. However, this level of decentralization or coordination likely adds to the overall complexity of the system.

3.3 Built-In

The escape hatch should not be optional and should require minimal effort from third-party layer two application developers, if any. An escape hatch which is application-specific means that for every application, user state must be migrated. This decreases the likelihood that such an escape hatch is congestion resistant as the number of applications on the rollup grows. However, if every application has an escape hatch *for all of their users*, this may be a middle ground between application- and-user specific escape hatches and support for full state support on escape hatches.

All entries of Table 1 that support an escape hatch mechanism are built-in. For the StarkNet rollup [12], the design of escape hatches are currently left to applications developers aiming to build on StarkNet [16]. If application-specific escape hatches are to be used, they should utilize efficient techniques for state transfer; for example, NFTs can be transferred across chains efficiently using wrappers [42].

3.4 (Transaction) Efficient

A mass-exit should be feasible even if the underlying layer one is congested. A rollup that satisfies this property should also support exporting the entire state (Sec. 3.5), as otherwise the exporting of specific state fields for thousands of users will cause congestion on any layer one which is order of magnitudes less performant than the inoperative layer two in question. One can imagine that for every 1,000 users on a rollup are using 20 different applications, $1,000$ state fields will need to be exported manually. As a result, for every such $1,000$ users and the current rate of about 15 transactions per

second (tps) on Ethereum [43], there are $20,000$ transactions/15 tps $\approx 1333s$ or about 37 minutes worth of transactions. These transactions are on top of the usual transactions on Ethereum and may be poorly timed; for example, they may need to exit during a gas war started by minting a popular NFT. The ability to have state migrated even with a congested system likely means that the escape hatch does not require a lot of transactions, so that only a few transactions resulting in the state migration. This ensures that even when gas wars are on, users can recover value. The invocation of escape hatch functionality is likely to occur simultaneously among all users and application when a rollup begins to censor transactions or becomes inoperative.

This property is satisfied by every solution in Table 1 which uses new block proposition as an escape hatch mechanism. For entries which list "Transact Using L1" (modify the state via execution on layer one) or "Force Exit to L1" (force the rollup to execute a specific withdraw transaction), users may have a more difficult time escaping during congestion on the underlying layer one.

3.5 Global

The escape hatch should not be application specific. To maximize transaction efficiency, the ideal escape hatch should transfer the full state of the rollup. This may be difficult to implement and may be indistinguishable from a fork of the rollup. However, this means that users do not need to specify balances or state that they wish to exit, and individual applications are not required implement specific escape hatch mechanisms.

This property is satisfied only by Fuel (v1) in Table 1 which uses new block proposition as an escape hatch mechanism. Aztec is does not satisfy this property, as [25] suggests that blocks that anyone can propose for Aztec only include token withdraw transactions. Polygon Hermez aims to be decentralized, allowing anyone to propose blocks [34].

However, one concern is that blocks may not be constructed fairly. Depending on who has the authority to propose the blocks, they may only include transactions affecting the state that is of interest for the proposer. To avoid this, a escape hatch mechanism could enforce that every state update processes a "minimum number of transactions" [22]. In turn, this may raise concerns related to gas use if there are limits within a block, but narrowing the scope to some minimum number of *escape* transactions may mitigate them. For example, blocks could be limited to transfer transactions (but not

necessary to or from a particular set of addresses or applications). This is a trade off with general block proposition which may more readily support full state escape.

Since new blocks may unlock funds in the custodian, it may be simplest to involve social components. Custodians of bridges may be implemented so that ownership is changed via message signed by a multi-signature wallet. As a result decentralized governance concerns are also escape hatch concerns. Finally, to avoid new malicious operators (who are online but provide incorrect data). The use of suggest that "staked executors" to cover the cost of challenges for malicious data is one approach [22].

3.6 Automatic & Live

The escape hatch should be always be automatically available under certain conditions. A user should be able to trigger escape hatch functionality without waiting for manual intervention, provided some conditions exist. These are likely the proof of censorship or long periods of time between updates from the rollup operator. A proof of censorship may be a receipt for a transaction submission which has not been executed even after a sufficiently long time.

Moreover, once an escape hatch is opened, it should guarantee that every escape-able transaction is eventually escaped on the underlying layer one. It may be desirable to have the ability to open the escape hatch only after some period of sequencer inactivity has been detected.

All entries in Table 1 except those without an escape hatch mechanism satisfy this property. As an example, Optimism [15] provides a forced transaction inclusion system (see Sec. 2) to counter censorship and ensure that (token-defined) value can be transferred. This is achieved through smart contracts on Ethereum which forces their inclusion into rollup blocks after some delay, through its *Canonical Transaction Chain* logic. For zkSync (version 1) [21], a command-line tool exists to generate a proof of a forced exit of funds from the zero-knowledge off-chain exchange.

4 Conclusion

We reviewed escape hatches for rollups and provided a quick overview of how some rollups may support this feature. We established a list of properties for

an idealized escape hatch before identifying some directions that may enable these properties to be satisfied. It may be that any feasible escape hatch only have a subset of these features. For example, an escape hatch which supports global state escape may be at odds with one that is automatic. This is because a global state escape is akin to a network fork and may require social consensus to determine several things, such as: where the new state is migrated, the exact state to be migrated (in terms of layer two block numbers), the development of new smart contracts, and crowd-sourced liquidity to facilitate the migration. Determining if these properties are mutually compatible with each other is left to future work.

Our list of idealized properties may also be incomplete and the approaches required to satisfy these properties may not be clear. Additional future work includes considering involve novel solutions to satisfy these properties, and to establish if any of these properties are infeasible, impossible, or require unpopular ideas. Leveraging the role of a rollup as middleware for blockchain ecosystems may be the most straightforward way to achieve these properties. Choosing to prioritize some of these properties may provide rollups with competitive advantages while also contributing to the increased security of the ecosystem at large.

Acknowledgement. Our survey and research of rollup escape hatches started at Quantstamp's Summer 2022 Research Retreat. Various ideas were discussed with many security researchers and engineers at Quantstamp. The authors would like to especially thank Kacper Bak for their discussions.

References

[1] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. SoK: Layer-two blockchain protocols. In Joseph Bonneau and Nadia Heninger, editors, *Financial Cryptography and Data Security - 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10-14, 2020 Revised Selected Papers*, volume 12059 of *Lecture Notes in Computer Science*, pages 201–226. Springer, 2020.

[2] Lukas Aumayr, Pedro Moreno-Sanchez, Aniket Kate, and Matteo Maffei. Blitz: Secure multi-hop payments without two-phase commits. In

Michael Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 4043–4060. USENIX Association, 2021.

[3] Giulio Malavolta, Pedro Moreno-Sanchez, Aniket Kate, Matteo Maffei, and Srivatsan Ravi. Concurrency and privacy with payment-channel networks. In Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 455–471. ACM, 2017.

[4] Giulio Malavolta, Pedro Moreno-Sanchez, Clara Schneidewind, Aniket Kate, and Matteo Maffei. Anonymous multi-hop locks for blockchain scalability and interoperability. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019.

[5] Erkan Tairi, Pedro Moreno-Sanchez, and Matteo Maffei. A^2l: Anonymous atomic locks for scalability in payment channel hubs. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pages 1834–1851. IEEE, 2021.

[6] Aggelos Kiayias and Dionysis Zindros. Proof-of-work sidechains. In Andrea Bracciali, Jeremy Clark, Federico Pintore, Peter B. Rønne, and Massimiliano Sala, editors, *Financial Cryptography and Data Security - FC 2019 International Workshops, VOTING and WTSC, St. Kitts, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers*, volume 11599 of *Lecture Notes in Computer Science*, pages 21–34. Springer, 2019.

[7] Joseph Poon and Vitalik Buterin. Plasma: Scalable autonomous smart contracts, 2017. `https://www.plasma.io/plasma.pdf`.

[8] Rami Khalil, Alexei Zamyatin, Guillaume Felley, Pedro Moreno-Sanchez, and Arthur Gervais. Commit-chains: Secure, scalable off-chain payments. Cryptology ePrint Archive, Paper 2018/642, 2018. `https://eprint.iacr.org/2018/642`.

[9] Vitalik Buterin. Sidechains vs Plasma vs sharding, 2017. `https://vitalik.ca/general/2019/06/12/plasma_vs_sharding.html`.

[10] Mustafa Al-Bassam, Alberto Sonnino, Vitalik Buterin, and Ismail Khoffi. Fraud and data availability proofs: Detecting invalid blocks in light clients. In Nikita Borisov and Claudia Diaz, editors, *Financial Cryptography and Data Security - 25th International Conference, FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part II*, volume 12675 of *Lecture Notes in Computer Science*, pages 279–298. Springer, 2021.

[11] Turing hybrid compute, 2022. `https://boba.network/turing-hybrid-compute/`.

[12] StarkNet. `https://starkware.co/starknet/`.

[13] Lior Goldberg, Shahar Papini, and Michael Riabzev. Cairo – a Turing-complete STARK-friendly CPU architecture. 2021. `https://eprint.iacr.org/2021/1063`.

[14] Fabian Vogelsteller and Vitalik Buterin. ERC-20 token standard, 2015. `https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md`.

[15] Optimism. `https://www.optimism.io/`.

[16] TimG. Starknet escape hatch research, 2019. `https://community.starknet.io/t/starknet-escape-hatch-research/1108`.

[17] Alexei Zamyatin, Mustafa Al-Bassam, Dionysis Zindros, Eleftherios Kokoris-Kogias, Pedro Moreno-Sanchez, Aggelos Kiayias, and William J. Knottenbelt. SoK: Communication across distributed ledgers. In Nikita Borisov and Claudia Diaz, editors, *Financial Cryptography and Data Security - 25th International Conference, FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part II*, volume 12675 of *Lecture Notes in Computer Science*, pages 3–36. Springer, 2021.

[18] Jay Freeman. Attacking an Ethereum L2 with unbridled optimism, 2022. `https://www.saurik.com/optimism.html`.

[19] Harry A. Kalodner, Steven Goldfeder, Xiaoqi Chen, S. Matthew Weinberg, and Edward W. Felten. Arbitrum: Scalable, private smart contracts. In William Enck and Adrienne Porter Felt, editors, *27th USENIX*

Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018, pages 1353–1370. USENIX Association, 2018.

[20] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 305–326. Springer, 2016.

[21] Alex Gluchowski. Introducing zkSync: the missing link to mass adoption of ethereum, 2021.

[22] Patrick McCorry, Chris Buckland, Bennet Yee, and Dawn Song. SoK: Validating bridges as a scaling solution for blockchains. Cryptology ePrint Archive, Paper 2021/1589, 2021. `https://eprint.iacr.org/2021/1589`.

[23] Arbitrum Nova. `https://nova.arbitrum.io/`.

[24] Arbitrum One. `https://bridge.arbitrum.io/`.

[25] Ariel Gabizon, Zac Williamson, and Tom Walton-Pocock. Aztec Network yellow paper. `https://hackmd.io/@aztec-network/ByzgNxBfd`.

[26] Jon Wu. Private DeFi with the Aztec Connect bridge, 2021. `https://medium.com/aztec-protocol/private-defi-with-the-aztec-connect-bridge-76c3da76d982`.

[27] Antonio Juliano. dYdX: A standard for decentralized margin trading and derivatives, 2018. `https://whitepaper.dydx.exchange/`.

[28] Fuel (v1). `https://www.fuel.network/`.

[29] Immutable X. `https://www.immutable.com/`.

[30] Layer2.Finance. `https://app.l2.finance/`.

[31] Layer2.Finance (ZK). `https://zk.layer2.finance/`.

[32] Loopring. `https://loopring.org/`.

[33] Metis. https://www.metis.io/.

[34] Polygon ZK: Deep dive into Polygon Hermez 2.0, 2022. https://blog.polygon.technology/zkverse-deep-dive-into-polygon-hermez-2-0/.

[35] Rhino.fi. https://rhino.fi/.

[36] Sorare. https://sorare.com/.

[37] ZKSpace. https://zks.org/.

[38] L2beat. https://l2beat.com/.

[39] Hayden Adams, Noah Zinsmeister, Moody Salem, River Keefer, and Dan Robinson. Uniswap V3 core. https://uniswap.org/whitepaper-v3.pdf.

[40] Johannes Sedlmeir, Fabiane Völter, and Jens Strüker. The next stage of green electricity labeling: Using zero-knowledge proofs for blockchain-based certificates of origin and use. *SIGENERGY Energy Inform. Rev.*, 1(1):20–31, dec 2022.

[41] Yingjie Xue and Maurice Herlihy. Cross-chain state machine replication. *CoRR*, abs/2206.07042, 2022.

[42] Vitalik Buterin. Cross-rollup NFT wrapper and migration ideas, September 2021. https://ethresear.ch/t/cross-rollup-nft-wrapper-and-migration-ideas/10507.

[43] Coinbase. Scaling ethereum & crypto for a billion users. https://blog.coinbase.com/scaling-ethereum-crypto-for-a-billion-users-715ce15afc0b.