Comparison between Bit2 and other Payment Networks for Agentic Payments

Introduction

The next generation of the internet will be powered not only by humans, but by **autonomous agents and machines** continuously exchanging value. Software agents purchasing compute resources, IoT devices paying for bandwidth, robots buying energy, and AI services paying for data will require payment systems capable of executing **millions of small transactions reliably and instantly**.

Traditional payment networks were not designed for this environment. Most systems suffer from one or more structural limitations: capital must be locked to route payments, transaction data must be published to the base chain, routing failures are common, or centralized operators become regulatory choke points. While these designs may work for human commerce, they create serious constraints for **machine-to-machine (agentic) economies**, where payments must behave more like **deterministic API calls** than like probabilistic financial operations.

Bit2 is a new Layer-2 payment architecture based on **client-side validation and cryptographic state compression**. Instead of relying on liquidity routing or centralized sequencers, Bit2 allows users to exchange ownership of coins directly while anchoring minimal commitments to the base layer. This approach dramatically reduces reliance on base-layer bandwidth while enabling extremely high throughput, deterministic payment success, and strong confidentiality.

This article compares Bit2 with several existing payment networks and Bitcoin Layer-2 designs—including the Lightning Network, Ark, RGB, and rollup architectures such as Base—across the metrics that matter most for agentic payments.

Scope of the Comparison

Not all Bitcoin Layer-2 constructions have comparable security properties. This comparison therefore focuses on systems where **a centralized operator cannot steal user funds**.

For example, **statechains such as Mercury or Spark** are excluded because the server can collude with a previous owner and take custody of coins. RGB represents a partial exception: although a RGB transaction aggregator cannot steal funds, it may be able to **withhold state data and temporarily block access**.

Bitcoin rollups are also not analyzed separately due to the lack of public information about their performance. The Ethereum rollup **Base** is used as a representative benchmark because it currently represents one of the most optimized rollup architectures, and it's already integrated in the x402.org gateway for Internet-native payments. Any Bitcoin rollup inherits the same structural limitations while typically performing worse due to Bitcoin's data-availability constraints.

The comparison therefore focuses on the following categories of payment networks:

- **Client-side validated systems** (Bit2, RGB, Shielded CSV)
- **Payment channel networks** (Lightning)
- **Virtual-UTXO systems** (Ark)
- **Rollups** (represented by Base)

The goal is not to declare a universal "winner", but to analyze which architecture best supports **high-volume autonomous payment ecosystems**.

Bit2 bases its scalability on using batch time-stamping services that make commitments onchain. Therefore, Bit2 can be deployed on any chain that supports data commitments. This includes Bitcoin itself and any Bitcoin-compatible Layer-2 environments such as EVM sidechains like Rootstock or on Ethereum. When deployed on Rootstock, Bit2 inherits Rootstock's merge-mined security and therefore benefits from its fast economic finality time. The protocol can also operate on other Bitcoin L2 environments such as Citrea or Alpen. In fact, the core Bit2 protocol is largely L1-agnostic with respect to the anchoring layer: as long as clients can verify transaction inclusion on one of several supported Bitcoin L2s, time-stamping services can publish commitments on any of them. In this article we compare Bit2 deployed directly on Bitcoin, on Rootstock and on Ethereum to illustrate how the properties of the anchoring layer influence the system's performance and guarantees.

Evaluation Criteria

Agentic payments impose requirements that differ significantly from human-driven payments. The comparison therefore focuses on metrics that directly impact automated commerce:

- **Confidentiality**
  Whether transaction histories and balances remain private.
- **Payment success rate**
  Whether a payment attempt reliably succeeds once initiated.
- **Hardware Wallet Friendly**
  Whether a hardware wallet can fully confirm all transaction data in a secure display, and whether the funds can be stored in cold storage without wallet liveness assumptions.
- **Pre-confirmation latency**
  How quickly a receiver can safely assume a payment will be included.
- **Economic finality time**
  The point at which reversing a payment requires global economic cost.
- **Finality** (irreversibility)
  The point at which a payment is considered irreversible
- **Maximum Payment Throughput**
  The maximum sustainable number of payments per second assuming random senders and a single destination per payment.
- **Maximum Multi-receiver Payment Throughput**
  The maximum sustainable number of payments per second assuming random senders and multiple payments per sender.
- **Single-source throughput**
  How fast one device can continuously produce payments.
- **Average Transaction Fee**
  The typical cost paid by a user to execute a payment on the network.
- **L1 data footprint**
  How much base-layer blockspace each L2 transaction consumes.
- **Collateral-Free Operation**
  Whether capital must remain locked to enable payments.
- **Unilateral fund access**
  Whether a user can always recover or spend funds without cooperation.
- **Uncensorable Exit**
  Whether a user can exit to the L1 without a single actor or a committee majority being able to prevent it.
- **Regulatory resilience**
  Whether the system depends on identifiable operators who may be regulated.
- **Quantum-computing Safety**
  Whether the system is or can be upgraded to be safe from cryptographically relevant quantum computers (CRQCs).
- **Programmable Spending Controls**
  The ability to enforce programmable constraints on agent-controlled wallets—such as spending limits, rate controls, whitelists/blacklists, and conditional or human-in-the-loop authorization—directly at the protocol or account level.
- **Payment composability**
  The ability to embed payments within conditional logic or multi-step workflows, where execution depends on external events, proofs, or other payments.
- **Globally-verifiable non-interactive payment proofs**
  Comprises two properties: the sender can prove that the receiver accepted the payment and a receiver can prove that anyone has fulfilled the payment of a certain invoice.
- **Lightning Network Integration**
  The ability to connect to the Lightning Network or support a lightning network channel, to receive and send payments through the LN.

These metrics together determine whether a payment network can support **large-scale autonomous economic activity**.

Property	Bit2 on BTC/RSK/ETH	L. Network	Base	Ark	RGB	INTMAX2 (*14)
L1	Bitcoin/Rootstock/ Ethereum	Bitcoin	Ethereum	Bitcoin	Bitcoin	Ethereum
Collateral-Free Operation	Yes	No	Yes	No	Yes	Yes
Technology	Client Side Validation	PCN	Optimistic Rollup	Shared UTXOs	Client Side Validation	Client Side Validation
Confidentiality	Yes, opt-out	Partial	No (*18)	No	Partial	Partial (*19)
Hardware Wallet Friendly	Yes (*22)	No (*21)	Yes	No (*21)	Yes (*22)	Yes (*22)
Payment Success Rate (*7)	100%	89.5% (*2)	100%	100%	100%	100%
Pre-Confirma-tion Time	100-400 msec (*25)	~500 msec	200 msec	100 msec	100 msec	100 msec
Economic Finality Time	600/24/12 sec	~500 msec	2 sec	10 min	10 minutes	12 sec
Finality (*17)	30/10/15 min	1 Day (*10)	7 days (*10)	60 min	60 min	15 min
Max. Throughput	1.6K / 6K / 40K tps	2000 tps (*8)	300 tps	25 tps (*11)	8 tps (*12)	7500 tps (*20)
Multi-receiver Throughput	102K / 384K / 2.5M tps	2000 tps	300 tps	25 tps	Unknown	480K tps (*20)
Multi-receiver Transactions	Yes	No	Yes	No	Unknown	Yes
Sustained Single Source Throughput	10 to 100 tps (*3)	40 tps (*4)	~40 tps (*9)	~25 tps (*11)	~8 tps	~4 tps (*13)
Average Fee (*7)	<0,01	0,20	0,01 to 0,2 (*6)	0.002	unknown	0.005 (*14)
L1 Data Footprint	4 bytes	4 bytes (*1)	20 to 800 (*6)	zero	Low (*12)	4.15 bytes
USD Stablecoin Support	Yes	No (*5)	Yes, ERC-20	No	No	Yes
Bitcoin Token Support	Yes	Yes	Yes (wBTC)	Yes	No	wBTC
Unilateral Fund Access	Yes (*24)	Yes	Yes	No	Yes	Yes
Uncensorable Exit	Yes (*10)	Yes (*10)	Yes (*10)	Yes	No (*15)	Yes
Mass Exit Safety	No/Yes/Yes	No	No	No	No	Yes
Regulatory Resilience	High	Medium	Low	Low	Low (*16)	High
QC-Safety	Planned	Depends on Bitcoin	Follows Ethereum	Depends onBitcoin	Depends on Bitcoin	Planned
Prog. Spending Controls	Yes	No	Yes	No	Yes	No
Payment Composability	Yes	Partial with HTLCs	Yes	No	Yes	No
Globally-verifiable Payment Proofs	Yes	No	Yes	Yes	No	No
Lightning Network Integration	Yes (*23)	Yes	No	Yes	Yes	No

Notes:

- The figures presented in the table are estimates derived from publicly available information, inspection of current implementations, and the known theoretical limits of each technology. They should therefore be interpreted as indicative comparisons rather than precise measurements, and may change as implementations evolve or new optimizations are discovered.
- Bit2 throughput figures assume a high-performance batch time-stamping server implementation that exploits load balancing, address-space partition, parallelization and pipelining, capabilities that the protocol is explicitly designed to support.
- Payment Success rate is estimated based on 1 USD payments (~ 1450 sats).
- (*1) Lightning network L1 Data Footprint assumes 100:1 industry standard offchain:onchain ratio for channel management (top-ups, rebalancing and closures)
- Sustained single-source throughput is estimated based on a device hosting 1 wallet and using 1 processor
- (*2) The success probability is amount-dependent.
- (*3) Bit2 supports adding multiple recipients to a single transaction. The overhead of adding more receivers impacts a single balance proof. On top of this feature, Bit2 provides selective privacy. Non-private transactions can be created at ~100 tps, and client bandwidth is the limiting factor since client-side proof size can reach 1 Mb when using Nova or Stwo as the proving scheme, and each receiver must receive a copy of the balance proof. Using the same batching mechanism, we estimate that private transactions can be issued at ~10 tps, and ZKP compute becomes the limiting factor.
- (*4) Lightning Network nodes currently have practical limitations. The need for synced DB writes currently limits transfers to 40 tps. Without considering this bottleneck, the next bottleneck is the maximum number of HTLCs (483) which limits throughput to ~1500 tps. Exceeding the number of HTLCs is unsafe because a commitment transaction would become too large and cease to be standard.
- (*5) To transfer a USD stablecoin over the Lightning Network, the Taproot assets protocol is required, but this protocol is neither standard to Lightning Network nor widely accepted.
- (*6) The amount of L1 blob or calldata consumed by Base depends on several factors, including the compression ratio of the data, which depends on its use frequency. Normal transactions consume between 20 and 40 bytes, while private transactions such as those provided by Railgun usually consume more than 800 bytes of data to accommodate ZK proofs, nullifiers,commitments, etc. Private transactions can be up to 40 times more expensive than simple ETH transactions.
- (*7) Average fees and payment success rate are considered for 1 USD payments, which represent a middle-point between micropayments and retail payments. Bit2 fees can vary depending on the time-stamping fee market.
- (*8) While the Lightning Network is often described as supporting millions of payments per second at the global level, the maximum payment throughput between two specific nodes is bounded by the **minimum cut capacity** separating them in the channel graph. By the max-flow min-cut theorem, the maximum achievable flow equals the total capacity of the channels in the smallest cut that separates the sender and receiver. In practice, this bound is further reduced because channel balances tend to become imbalanced over time, shrinking the effective directional liquidity available for routing. Additionally, implementation constraints—such as limits on the number of concurrent HTLCs per channel and database write bottlenecks in node software—further restrict achievable throughput.
- (*9) The maximum throughput of Base is limited by the number of transactions that can be queued without confirmation. Base accepts up to 80 “future nonces”, and since blocks are issued every 2 seconds, this limits single-source throughput to 40 tps.
- (*10) The limiting factor is the challenge period.
- (*11) This estimation is based on random payments between users and 100M collateral used by the service provider. Ark Labs does not publish throughput information.
- (*12) While RGB documentation describes a batching service that can considerably increase RGB throughput, no such service is currently being offered.
- (*13) INTMAX2 can create transfers containing up to 64 destinations, which increases single-source throughput. However, creating the zero-knowledge proof requires 15 seconds of computing or more depending on the device. The limiting factor for single-source throughput is the need to compute succinct zero-knowledge proofs (source).
- (*14) The INTMAX FAQ states “INTMAX offers greater scalability than centralized servers. Even with a large user base, transaction fees can remain as low as 0.5 cents in a highly predictable and stable way”. We estimate that the transaction cost is lower (based on the L1 data footprint), but currently INTMAX charges much higher than the L1 cost, possibly due to operational costs.

- (*15) RGB could use a BitVM/BitVMX bridge to receive BTCs, but currently there is no plan to add such a bridge.
- (*16) RGB aggregators handle financial and payment information of users.
- (*17) For Bitcoin, we use the standard 6 block confirmations to reach Bitcoin transaction finality. For Rootstock, we use 25 blocks. For Ethereum, we use 2 epochs.
- (*18) Third party applications, such as Railgun, can provide confidential payments to Base.
- (*19) Sender identities (pseudonymous addresses) are included in aggregated signatures for block commitments and are visible to all users.
- (*20) INTMAX2 paper (2023) states that there is “an upper limit of 7500 transaction batches per second on Ethereum, where each transaction batch can transfer an unlimited number of tokens to an unlimited number of recipients”. However, the implementation limits the number of recipients per transfer to 64. The 7.5K figure is computed assuming the use of 0.375 Mb of sharded data (also called “blobs”) per block. However, using calldata instead of blobs, INTMAX2 throughput could increase to 40K tps, but the transaction cost would also increase ~200X, as calldata bytes are currently much more expensive than blob bytes. On the other hand, blobs must be paid in full (131072 bytes), and consuming one block per Ethereum block provides enough space for ~2600 tps. Therefore, during the network bootstrap period, assuming an average usage of 1 tps, each INTMAX2 transaction would need to cost 2600 times more to pay for the Blob, or be subsidized by a company.
- (*21) Time-sensitive reactions due to blockchain-originated events (liveness requirement) prevents the use of hardware wallets with these payment systems because hardware wallets are stored over long periods of time.
- (*22) Client-side validated systems require backup of the wallet data. If this data is lost, the funds cannot be accessed, even if the user possesses the seed. Therefore, the desktop or mobile application connecting with a CSV hardware wallet must store periodic encrypted backups of the wallet data in the cloud.
- (*23) Bit2 supports the creation of Lightning Network-compatible channels on top. However this is an extension that will not be available on the first release of the Bit2 network.
- (*24) When deployed on a blockchain with Turing-complete smart contracts, users can exit using a decentralized Bit2 bridge. When deployed on Bitcoin, the best trust-minimized bridges in existence rely on the 1-of-n honest security assumption (See BitVM / BitVMX).
- (*25) Each Timestamp Service Provider (TSP) defines the pre-confirmation time. A TSP may offer 100-msec pre-confirmations, while other TSPs may delay them. The procolol, and the cryptographic operations carried out by the TSP enable fast pre-confirmations.

Explanation of the Metrics

Technology

In **client-side-validated systems** such as *Bit2*, coin ownership evolves through **cryptographic state transitions that the receiver validates locally**, using commitments anchored periodically to Bitcoin. In this model, the sender transfers the right to spend specific coins, and the receiver verifies that the transfer is valid and not double-spent according to the history presented. Bitcoin is used only to leave a **private cryptographic trace** of all outgoing transfers produced by an account, binding them together in a *log chain*. This log chain ensures that a user cannot prove the existence of a particular outgoing transfer without also considering all previous transfers that were committed before it inside the proof. As a result, selective disclosure of transaction history becomes impossible, preventing equivocation while still preserving strong privacy.

Compared with **payment channel networks (PCNs)** such as the Lightning Network, client-side validation offers several advantages. It achieves comparable or higher throughput without requiring liquidity to be locked in channels, simplifies the protocol design by reducing the

number of interacting parties, and provides stronger confidentiality since transaction histories can be compressed into succinct cryptographic proofs.

Compared with rollups, the architecture is fundamentally different. Rollups rely on a centralized or semi-centralized **sequencer** that orders transactions and periodically publishes batches of transaction data (or validity proofs) to the base layer. This design makes the system dependent on sequencer availability and on significant L1 data bandwidth. While Bit2 also includes a **time-stamping service role**, this function is not unique or privileged: multiple independent time-stamping servers can coexist and compete to offer better quality of service and lower fees, and users can freely register with or leave any time-stamping service.

In contrast, most existing rollups rely on a **single active sequencer**, which introduces a potential single point of failure and censorship risk. Bit2 avoids this structural dependency because time-stamping servers do not control transaction validity or ordering. As a result, the system can remain operational even if individual time-stamping servers disappear or behave adversarially, while market competition can naturally regulate cost and service quality.

Collateral-Free Operation

In Bit2 security does not depend on maintaining liquidity inside channels or shared pools, **no capital must remain locked as routing or settlement collateral**. The only scarce resource consumed is Bitcoin blockspace for occasional commitments, so the marginal cost of transactions is primarily data availability and verification—rather than the opportunity cost of locked funds. Time-stamping servers may optionally post security bonds to provide fast time-stamp assurances, allowing them to economically guarantee that a blob of data will be time-stamped in a future commitment. However, these bonds are not required for the basic operation of the protocol. In addition, pegnatories (BitVMX committee members) may lock security deposits to ensure honest behavior in dispute resolution. Importantly, these deposits are not proportional to the total value of funds bridged into the system; instead, they are sized relative to the cost of executing disputes and enforcing correctness. As a result, Bit2 does not require proportional collateralization of the payment volume in order to operate securely.

Similarly, Base and other optimistic rollups—including proposed Bitcoin rollups—require security bonds from the parties that submit or challenge bridge withdrawals, so that incorrect withdrawal proofs can be economically penalized during the dispute period.

In contrast, **Lightning** and **Ark** rely on **pre-funded liquidity to guarantee instant settlement**. Lightning locks coins in payment channels so that intermediate nodes can forward payments without trusting each other, while Ark locks funds in shared UTXOs or virtual-UTXO pools managed by service providers. In both cases, the locked funds act as collateral ensuring that payments can be honored even if participants disappear or misbehave. However, capital that is locked cannot be used elsewhere, creating an **opportunity cost proportional to interest rates and payment volume**. Over time, routing nodes or Ark providers must recover this cost through fees. Therefore, even if early deployments subsidize fees to bootstrap adoption, the long-term

transaction price inevitably reflects **the capital cost of maintaining sufficient collateral to support the network's payment throughput**.

Confidentiality

Payment confidentiality is essential for both commerce and individual freedom. If every transaction is publicly visible, competitors can infer business relationships, suppliers, and pricing strategies, while individuals expose their spending habits, income patterns, and social connections. In a world of automated commerce and agentic payments—the type of ecosystem you are designing with Bit2—this transparency can leak enormous amounts of economic intelligence. Confidential payments therefore protect both **commercial privacy** and **personal autonomy**, preventing large-scale economic surveillance.

Bit2 provides confidentiality because it relies on client-side validation combined with zero-knowledge proofs (both STARKs and SNARKs). Each account maintains its own private transaction history, and the receiver of a payment only receives a STARK proof that the current state is valid and that the coins originate from legitimate genesis funds. The STARK proves that all past state transitions were valid without revealing the transactions themselves. As a result, observers can verify that the funds are authentic and that no inflation occurred, while learning nothing about prior transfers, balances, or counterparties. The transaction history is therefore **compressed into a proof of correctness rather than published as data**, preserving confidentiality. While Bit2 relies on techniques conceptually similar to INTMAX2, it offers stronger privacy. In Bit2, only the time-stamping server learns the publisher identifiers and can infer how many transactions each publisher generates. In contrast, INTMAX2 exposes the list of participating senders in each commitment, allowing any observer to see the pseudonymous identities of all participants and infer their activity levels.

In contrast, **Lightning Network**, **Ark**, and **Base** do not provide confidentiality because the transaction data or state transitions remain visible to intermediaries or the network. In the Lightning Network, each payment is routed through intermediate nodes using HTLCs; although the global network does not see the entire payment graph, the routing nodes learn partial information about the payment and amounts, and channel balances can leak through probing and channel updates. While Lightning privacy could improve with techniques such as Point Time-Locked Contracts (PTLCs), which remove hash reuse and reduce some correlation attacks, or route blinding, which hides the destination node, these improvements are only partial because payment amounts still leak information that can be used to correlate hops along a route.

Ark relies on a centralized or semi-centralized coordinator that aggregates and reassigns virtual UTXOs; this coordinator necessarily observes the transaction flows and balances of participants.

Base, as an optimistic rollup, publishes transaction data (or data availability commitments) to the underlying chain, meaning the full transaction history is reconstructible and visible to

observers. In all these systems, validation requires access to transaction data, whereas Bit2 replaces data visibility with **zero-knowledge proofs of correctness**, enabling verification without disclosure. While additional privacy layers can be built on top of rollups—such as the Railgun protocol for Ethereum and Base—these approaches increase transaction costs and may offer limited protection if the anonymity set remains small. In contrast, Bit2 provides confidentiality by default, and its anonymity set naturally grows with the entire population of users, approaching the maximal possible set.

Hardware Wallet Friendly

Not all payment networks are hardware-wallet friendly. To securely support hardware wallets, a payment protocol must satisfy two fundamental requirements.

First, the signer must know exactly what it is signing so that the hardware wallet can display meaningful information to the user: the destination address, token, amount, fees, and any other security-critical parameters. Until recently, Ethereum offered limited support for hardware wallets because devices could not interpret the payloads of the smart contract methods used to transfer tokens. Even today, hardware wallets cannot infer essential parameters—such as the decimal precision—of unknown user-defined tokens, forcing users to perform blind signing. In contrast, Bit2 transactions use standardized formats for destinations, token identifiers, amounts, and fees that can be interpreted by lightweight firmware and presented to the user before signing. Because transaction semantics are part of the protocol rather than embedded inside arbitrary smart contract calls, hardware wallets do not need to understand application-specific contract logic in order to safely authorize payments.

Second, a payment network should not require a liveness assumption from the signer. Since most users store digital assets in hardware wallets for long periods of time, often disconnected or even kept in secure storage, the protocol should not assume that the signing device is continuously online. For example, the Lightning Network requires users to monitor the Bitcoin blockchain for attempted unilateral channel closures or delegate this responsibility to watchtowers. A hardware wallet stored in a safe deposit box cannot satisfy such a requirement.

In contrast, Bit2 imposes no liveness requirement on the hardware wallet. Users can safely receive payments while their hardware wallet remains offline by running a mailbox and wallet-monitoring application on a phone or computer. The hardware wallet is only needed when authorizing new outgoing transactions, while synchronization, timestamp retrieval, and payment reception are handled by online companion software. This naturally separates cold key management from online wallet operation, which closely matches the security model for which hardware wallets were originally designed.

Liveness Requirements

Many modern payment systems implicitly assume that wallets remain online, responsive, and

continuously monitoring the network. This creates a fundamental tension between usability, security, and long-term storage.

This condition, called **liveness**, is closely related to cold storage and hardware wallets: liveness is the requirement that a wallet must react within a bounded time to external events, often originating from the blockchain or a coordinating service.

In systems with liveness requirements, failing to act in time can result in partial or total loss of funds. This assumption is incompatible with hardware wallets, which are typically stored offline for weeks, months, or even years.

The Lightning Network requires periodic monitoring of the blockchain. Users must detect unilateral channel closures and respond within a dispute window. If they fail to do so, a counterparty can broadcast an outdated state and claim funds. While watchtowers mitigate this risk, they introduce additional trust, complexity, and operational overhead.

Ark also imposes a form of liveness. Users must periodically interact with the Ark server to migrate funds into new virtual UTXO trees. If the user remains offline for too long, they risk losing access or being forced into more complex recovery paths.

Rollups occupy an intermediate position. Users can keep funds in a hardware wallet, but they must remain aware of **exit windows**. If the sequencer halts permanently, users have a limited time to submit exit transactions on L1. Missing this window can lead to funds being locked.

Bit2 users have no liveness requirements. No requirement to monitor the blockchain for adversarial actions. No requirement to stay connected to a server. No bounded reaction windows that threaten funds. A user can safely store funds in a hardware wallet, disconnect it completely, and return at any time without risk of losing funds due to inactivity.

This distinction is crucial. With client-side validation: A user can move funds to cold storage (paper wallet or hardware wallet) and remain offline indefinitely.

Payment Success Rate

A very low payment failure rate is essential for any payment system because every failed payment introduces friction, uncertainty, and operational cost. For human commerce this already degrades user experience, but for **agentic payments** it is even more critical: automated agents cannot easily negotiate retries, resolve ambiguity, or tolerate nondeterministic outcomes. If a payment attempt can fail unpredictably, applications must implement complex retry logic, liquidity probing, and error handling, which increases latency and infrastructure costs and can break automated workflows. In a large-scale agentic economy—where millions of micro-transactions may be executed autonomously—payments must behave like deterministic API calls: once accepted, they must succeed.

In Bit2, the sender submits the transaction to hash as an opaque blob to a time-stamping service, and once the sender receives confirmation from the time-stamping service the sender can consider the transaction to have an economic finality guarantee: the time-stamping service commits to including performing the time-stamping in the next commitment and faces significant penalties if it fails to do it. This mechanism ensures a very high payment success rate. Once the corresponding time-stamping commitment is anchored in a block, the L2 transaction contained in the blob becomes as irreversible as an L1 transaction.

The correctness of a payment is enforced by client-side validation and the recursive STARK-compressed state history, not the time-stamping service. Therefore the success probability after acceptance is effectively **100%**, except for catastrophic time-stamping service failure. In contrast, Lightning Network payments are routed through multiple liquidity-constrained channels using HTLCs. Each hop must have sufficient inbound liquidity and must successfully forward the payment within time constraints; if any hop fails, the whole payment fails. Because liquidity is fragmented and routing information is imperfect, payments frequently fail even when a valid path exists, producing observed success rates around **~89–90%** and often appearing to fail “for no reason” from the sender’s perspective.

Also, in Bit2 the payment success rate is independent of the payment amount, while in Lightning it is strongly amount-dependent. In Bit2, the transfer is simply a state update and does not depend on the availability of liquidity across intermediate nodes, so the probability of success does not change with the payment size. In the Lightning Network, however, payments must traverse a path of channels where each channel must have sufficient outbound liquidity to forward the amount. Larger payments are therefore harder to route because the required liquidity must simultaneously exist across all hops, which significantly reduces the probability of finding a valid route. As a result, **the larger the payment, the lower the success rate** in Lightning.

Pre-Confirmation Time (bonded)

Fast **pre-confirmations** **(≈100–200 ms)** are extremely important for payment systems that serve **agentic interactions**, where software agents must decide instantly whether a payment succeeded before continuing an automated workflow. Examples include API access, compute rental, bandwidth purchases, IoT services, or autonomous trading agents. In these environments, waiting seconds—or experiencing unpredictable failures—breaks the interaction model, because the receiving agent must either stall the process or risk providing service without payment. A deterministic pre-confirmation allows the receiver to safely treat the payment as completed almost immediately and continue execution, enabling synchronous protocols between agents and making micropayment-based APIs feasible. Bit2 pre-confirmations are cryptographically signed by the time-stamping service, allowing the sender to forward them to the receiver as a verifiable payment guarantee backed by the time-stamping server’s economic commitments.

The Lightning Network cannot provide deterministic pre-confirmations because a payment must traverse a **multi-hop HTLC route**, and success depends on the liquidity and responsiveness of several independent nodes. Even if an intermediate node initially forwards the HTLC, the payment can still fail later due to liquidity constraints, channel policy changes, or timeouts, and the sender only learns the final result after the route attempt completes. Therefore Lightning payments are **probabilistic and path-dependent**, and nodes cannot credibly promise that a payment will succeed before the whole routing process finishes.

Bit2, in contrast, allows secure pre-confirmations from time-stamping servers. A time-stamping service can cryptographically commit to including a blob and attach economic guarantees to that promise. If the time-stamping service fails to include the blob, the user can either **slash a security bond** posted by the time-stamping server or claim compensation through a **quality-of-service contract** that enforces reimbursement plus a penalty when time-stamping deadlines are missed. Because these guarantees are enforced by smart contracts and signed commitments, the time-stamping server's pre-confirmation becomes economically binding, enabling **deterministic, sub-second payment assurances** that agents can safely rely on.

Economic Finality Time

Economic Finality Time is the minimum time after which reversing a transaction requires bearing a non-local, consensus-enforced economic cost, with no single party able to unilaterally nullify that guarantee. This definition cleanly excludes pre-confirmations backed only by bonds, because a sequencer or time-stamping service can over-promise to many users at once, so the same bond may be insufficient against aggregate cheating.

Economic Finality Time matters because it is the first moment at which a payment is protected by the system's shared security budget rather than by the promise, reputation, or collateral management of one operator. For commerce, and especially for agentic trade, this is the difference between "someone said the payment will probably stick" and "reversing it now requires paying a real protocol-wide price". On Rootstock, Bit2 inherits that property once the Bit2 commitment is included in a Rootstock block, because after that point reversal is no longer a matter of one time-stamping server changing its mind: it would require defeating or outcompeting Rootstock's consensus, so the Economic Finality Time is roughly one Rootstock block, currently about 24 seconds on average. By contrast, Bitcoin rollups such as Citrea do not give that same fast Bitcoin-level economic finality for ordinary off-chain transaction ordering, because users first depend on the rollup/sequencer pipeline and only later on Bitcoin anchoring, proof publication, and, in failure cases, force-inclusion or exit mechanisms.

Lightning is the strongest, because within an already funded channel the latest state is economically enforceable immediately: a party trying to reverse by publishing an old state risks losing funds through Bitcoin's revocation mechanism.

Maximum Payment Throughput

A high **Maximum Payments per second** is critical for any serious payment system because congestion is not just a performance issue: it directly becomes a reliability, cost, and product-design problem. Once demand approaches the system's ceiling, queues grow, fees rise, latency becomes unpredictable, and small payments stop being economical. This matters even more for **agentic** commerce, where large fleets of agents may generate bursts of tiny payments continuously and in parallel. In that setting, the system is not competing only on average throughput, but on whether it can sustain massive concurrency without forcing capital lockups, routing retries, or expensive fallback to the base layer.

The maximum throughput on Bitcoin is 1.6K tx/sec. The throughput is constrained mainly by **time-stamping service efficiency** and by how many commitments can be packed into an L1 transaction, not by per-payment collateral or path liquidity. Each payment consumes only a **few bytes** of L1 data, while the time-stamping service can **pipeline and parallelize** many commitments and publish them together. Because senders are deterministically assigned to commitment buckets, a proof for one bucket does not need to reason about the others, and double-spends cannot hide across buckets. That gives Bit2 a scaling path that is fundamentally different from the Lightning Network, where throughput is limited by **channel liquidity, routing, channel management, and HTLC/channel constraints**; different from Base, where fast preconfirmations exist but the system is still bottlenecked by a **single sequencer plus L1 batch posting bandwidth**; different from Ark, where user experience is good but the design still runs into **server coordination, round structure, refresh/offboarding mechanics, and liquidity/capital economics**; and different from RGB in its current non-aggregated form, where recipients must validate the relevant client-side history and each transfer still depends on individual witness/consignment handling rather than a highly parallel shared time-stamping service layer.

Sustained Single-source Throughput

Sustained Single-source Throughput matters because many real applications are bottlenecked not by the global network ceiling, but by how fast **one device** can repeatedly authorize and deliver payments. A laptop, phone, or embedded controller may need to pay dozens of counterparties per second during bursts: API calls, sensor data purchases, ad impressions, energy metering, robotic coordination, or market-making between agents. In those cases, the question is not "can the network eventually absorb the load?" but "can one wallet keep paying continuously without waiting on round trips, route discovery, channel rebalancing, or a centralized sequencer's pacing?" For this metric to be fair in comparisons, we restrict the payment latency to 1 second. Therefore, we limit the amount of batching the sender can do. Bit2 is strong here because it supports private transactions having one sender but multiple receivers. The sender can also batch time-stamp many independent transactions in the same time-stamp server commitment., Therefore, the sender is dominated mainly by **local computation** rather than communication latency. Bit2 provides a faster mode without fresh privacy proof generation

and the standard fully-confidential mode. According to our tests, we estimate that a standard home laptop connected to the Internet will be able to deliver 100 non-private Bit2 payments per second with bandwidth being the limiting factor, while the same laptop will be able to create 10 fully private payments per second, with local computations being the limiting factor, and using recursion-optimized systems such as Nova.

By comparison, the **Lightning Network** makes a single sender depend on route finding, remote liquidity, channel state updates, and receiver/path availability, so throughput is constrained by interaction and liquidity, not only by the sender's CPU. Also, all implementations are currently practically constrained by the DB I/O bottleneck.

Base can offer very fast preconfirmations through a sequencer, but it has a future nonce restriction of only 80 transactions, afterward no additional transaction is accepted. Private transactions on Base (i.e. using Railgun) are highly limited by local SNARK proof computation, and a single transaction can take more than 30 seconds to prove.

Ark may eventually provide fast single-source throughput but currently sustained throughput is partly tied to server coordination and capital deployment rather than only the wallet's local compute.

L1 Data Footprint per L2 Transaction

The **L1 Data Footprint per L2 Transaction** is a fundamental scalability metric for any Layer-2 system because L2 security ultimately relies on anchoring data or commitments on the base layer. If each L2 transaction consumes many bytes on L1, the L2 becomes tightly coupled to the availability and price of L1 blockspace. During periods of L1 congestion, the cost and throughput of the L2 degrade proportionally. In contrast, an L2 that compresses a very large number of transactions into a tiny L1 commitment becomes almost independent from L1 congestion. This dramatically improves scalability and cost predictability because millions of L2 transactions can share a negligible amount of L1 bandwidth.

Bit2 consumes 3-4 bytes per transaction. This is an order of magnitude better than typical rollups. For example, optimistic rollups such as **Base** must publish transaction data for data availability, which results in tens of bytes per L2 transaction (around ~40 bytes). Client-side validated systems such as **Shielded CSV** require approximately **64 bytes per transaction** to anchor commitments. Stateless rollup designs such as **INTMAX2** (now deprecated) significantly reduce this cost to roughly **4–5 bytes per transaction**. However, Bit2 still improves on these systems.

Protocols such as **Ark** and **RGB** theoretically achieve an even smaller L1 footprint because a fixed L1 commitment can anchor an arbitrarily large number of off-chain transactions. In practice, however, the difference is negligible because Bit2's footprint is already extremely small. Once the L1 cost per transaction is reduced to a few bytes, further improvements provide

little economic benefit—the dominant scalability limits shift from L1 bandwidth to computation, networking, or time-stamping service throughput.

USD Stablecoin Support

Bit2 supports two methods for bringing stablecoins into the ecosystem. First, an issuer can issue the stablecoin directly on Bit2. Second, stablecoins can flow from the underlying Layer 1 blockchain into Bit2 through the canonical bridge. For this second option, Bit2 must be deployed on top of a blockchain where the stablecoin has already been issued.

Bit2 is designed to provide a set of technological capabilities that may assist stablecoin issuers in implementing compliance controls, sanctions programs, and lawful-order response mechanisms. It includes tools that allow issuers to blacklist addresses, preventing agents and users from accepting stablecoins originating from those addresses, as well as tools to freeze addresses so they cannot transfer stablecoins. Importantly, a stablecoin issuer can only exercise control over its own stablecoin. It cannot block the transfer of Bitcoin or any other decentralized asset. The issuer's authority is limited to the assets it has issued.

This distinction is important. Bit2 preserves the neutrality of the underlying network while enabling issuers to exercise the controls necessary to meet their regulatory obligations. The network remains decentralized even when individual assets implement issuer-specific compliance policies.

Average Transaction Fee

Average Transaction Fee measures the typical cost paid by users to execute a payment on the network. This metric is crucial for agentic commerce because automated agents often exchange extremely small payments. If transaction fees are high or unpredictable, many economically useful interactions—such as paying for milliseconds of compute, kilobytes of bandwidth, or micro-services—become infeasible. A payment network designed for large-scale automated commerce must therefore keep the marginal cost per transaction extremely low and stable, even under high load.

In Bit2, transaction fees are expected to remain very small because the only scarce resource consumed is the occasional L1 commitment, whose cost is amortized across a very large number of payments. Since thousands or millions of L2 transactions can share the same commitment, the effective cost per payment becomes a few bytes of the underlying L1 fee. The time-stamping server must also generate zero-knowledge proofs (SNARKs) to compress and verify the correctness of state transitions, and one of these proofs is published on-chain with each commitment. However, these proofs are produced per commitment rather than per user or per transaction, so their cost is shared across all included payments. Moreover, modern proof systems allow massive parallelization using GPUs, which further reduces the amortized proof-generation cost per transaction. In contrast, systems such as Lightning and Ark must

recover the opportunity cost of locked liquidity through routing or service fees, while rollups such as Base must pay for publishing transaction data for data availability on the base layer. As a result, their long-term transaction fees are structurally tied either to capital costs or to the price of L1 blockspace, whereas Bit2 primarily amortizes those costs across extremely large transaction batches.

Unilateral Fund Access

It measures whether a user can always access or spend their funds **without relying on the cooperation of any single party**. Unilateral fund access is critical in payment systems because users must be able to recover or spend their funds even if service providers become malicious, unavailable, or censor transactions. Systems that depend on a single party for data or authorization introduce a blocking risk where funds remain technically owned by the user but practically unusable.

In Bit2, in the event of time-stamping service failure, the sender can publish a deregistration transaction on the L1 and continue transacting using a meta-protocol directly on the base layer or simply register with another time-stamping service. The recovery operation requires only small L1 transactions and therefore remains inexpensive.

In contrast, if an Ark service provider fails, the user must exit the L2 using the unilateral exit mechanism before joining another provider. Ark exits are expensive because they require publishing transaction chains (tree branches) that prove the user's position in the shared UTXO structure.

A similar situation occurs in Base: if the sequencer fails or becomes unavailable, users must initiate a forced withdrawal to the L1, wait through the challenge period, and then deposit their funds again into another L2 or sequencer pipeline. This recovery path is significantly slower and more costly than switching time-stamping servers in Bit2.

Uncensorable Exit

Uncensorable Exit measures whether a user can always withdraw funds to the base layer without a single actor—or even a majority of a committee—being able to prevent it. Although this property is slightly weaker than **Unilateral Exit** (where a user can independently complete the withdrawal process without relying on any other party), it still provides a very strong safety guarantee. In practice, it ensures that funds cannot be permanently trapped by service providers or infrastructure operators, while allowing architectures that do not rely strictly on virtual UTXO ownership models.

Bit2 provides uncensorable exits because neither the payment time-stamping service nor the bridge operators can block withdrawals. The Bit2 bridge relies on a **1-of-N honesty assumption** enforced through BitVMX: as long as a single honest participant in the pegnatory

set exists, incorrect bridge behavior can be challenged and the withdrawal can be enforced. Furthermore, once commitments are published, payments themselves do not require time-stamping server cooperation, so users can always recover their funds through the bridge mechanism.

In contrast, RGB aggregators can withhold state data, preventing users from proving ownership of their assets and effectively blocking access to their funds. Additionally, RGB in its current form does not provide a trust-minimized Bitcoin bridge, meaning users cannot enforce withdrawals to Bitcoin without relying on trusted intermediaries. As a result, RGB does not currently provide uncensorable exits.

Mass Exit Safety

Mass Exit Safety measures whether a payment network can preserve user funds and withdrawal guarantees when a very large fraction of users tries to exit to the base layer at the same time. This metric matters because many systems advertise unilateral exit in the normal case, but fail to preserve that guarantee under stress. If exits depend on narrow time windows, optimistic challenge periods, or a large number of individual L1 transactions, then a panic event can turn theoretical self-custody into a race for scarce blockspace. In that situation, users may face expired challenge windows, forced de-pegs, or transaction fees so high that withdrawing small balances becomes irrational.

Bit2 is comparatively strong on this metric. When Bit2 is deployed over an EVM chain, the bridge contract can directly verify exit proofs on-chain, so exits do not need to rely on an optimistic challenge game with delayed finalization.

The Lightning Network is much weaker under mass exit conditions. The Lightning paper itself identifies “forced expiration spam” as perhaps the greatest systemic risk for the network: if many channels are forced to close at once, blockchain capacity can be overwhelmed, delaying confirmation until other time-sensitive transactions become valid. The paper explicitly warns that users must be given sufficient time for their transactions to confirm when interacting with non-cooperative peers.

Ark also appears fragile under mass exit. Public Ark documentation describes unilateral exits as slow and expensive, and explains that exiting a VTXO requires broadcasting a chain of transactions. Ark implementations commonly use expiry parameters around seven days for the VTXO tree and about twenty-four hours for unilateral exit delays. If many clients attempt to leave simultaneously, mempool chain limits or insufficient fees can delay inclusion long enough that the Ark Service Provider may recoup the funds, causing user loss.

Base, and more generally optimistic rollups, are also weak on Mass Exit Safety. On OP Stack chains, withdrawals are not final immediately: users first initiate a withdrawal, then wait for the output root to be posted, then prove the withdrawal on L1, and finally wait out an approximately one-week challenge period before finalizing. The bridge is permissionless, but the exit path is

multi-step, delayed, and explicitly dependent on a challenge window. Under mass exit, this means many users must submit L1 transactions and pay L1 gas while also waiting through a fixed dispute period.

RGB lacks a trust-minimized canonical Bitcoin bridge comparable to the ones being evaluated here, so it does not provide mass exit safety.

Bit2 over an EVM chain is strongest because exit proofs can be verified directly by the bridge contract, avoiding optimistic challenge windows.

Regulatory Resilience

Financial Regulatory Resilience measures how resistant a payment network is to regulation targeting specific actors that are essential for its operation. This is important because payment networks tend to become critical infrastructure for commerce, and regulators often impose obligations—such as licensing, KYC, transaction monitoring, or reporting—on identifiable intermediaries. If the network depends on a small number of economically identifiable operators, regulatory pressure on those operators can reduce functionality, introduce censorship, or even halt the network entirely. Therefore, a resilient system is one where no participant is structurally forced into the role of a regulated financial intermediary.

Financial regulatory resilience does not imply regulation avoidance, it only means that it makes no sense for regulators to impose regulatory functions and obligations to network infrastructure providers. Those functions must be provided on the edges, where it should be, making the infrastructure only a fair and open technological backbone.

Bit2 has very high resilience to regulatory risk on the infrastructure because its time-stamping services are account-agnostic and transaction-agnostic: they simply aggregate opaque blobs and maintain a count of which publisher submitted each blob. Also, electronic mailbox services that may be used to receive and queue incoming payment messages, are use-case agnostic standard message dispatch services. All economic information—balances, transfers, and validation—flows peer-to-peer between users through client-side validation, and the blobs themselves do not reveal that they correspond to payments. This strongly reduces the feasibility of classifying the time-stamping servers as money transmitters or custodians. In contrast, the Lightning Network relies on routing nodes and hubs that actively facilitate payments and manage liquidity, making them economically identifiable actors potentially subject to money-transmission or bank-secrecy regulations. Systems such as Ark and rollups like Base also expose identifiable infrastructure—liquidity providers, coordinators, or sequencers—that directly participate in payment processing, creating regulatory choke points that could be targeted by compliance requirements or enforcement actions.

Quantum-computing Safety

We evaluate whether the system is or can be upgraded to be safe from cryptographically relevant quantum computers (CRQCs). At this point, none of the evaluated payment networks is quantum-safe. In its current version, Bit2 uses quantum-safe signatures and quantum safe STARKs, but it still makes use of SNARKs to reduce onchain footprint.

Lightning Network Integration

Unilateral payment channels remain the most efficient primitive for ultra-low-latency scenarios. A payment over a unidirectional payment channel requires only a signature from the sender, which the receiver stores. No routing, no coordination, no third-party interaction. Two machines can exchange value instantly, even across the globe, with minimal overhead. This model is ideal for nanopayments. Nevertheless, multi-hop payment channel networks such as Lightning still dominate in confirmation latency.

Bit2 was designed to support payment channels directly on top of it. These channels are extremely efficient: opening or cooperatively closing a channel requires only 4 bytes of L1 data, equivalent to a standard Bit2 transaction. In contrast, the Lightning Network requires a full Bitcoin transaction (~1000 weight units) to open or close a channel. While Lightning amortizes this cost across many off-chain payments (commonly estimated at ~100:1), a Bit2-based Lightning Network (Bit2-LN) reduces the L1 footprint of channel management by approximately two orders of magnitude.

In practice, this does not translate directly into 100× cheaper payments, because Lightning's dominant cost is not L1 bandwidth but locked capital. However, for nanopayments—where channel balances remain small—this efficiency becomes highly relevant.

Bit2 relies on timestamping service providers (TSPs) to anchor user commitments efficiently. These services will likely charge users. Unilateral payment channels are ideal to pay TSPs with minimal overhead and high efficiency. Beyond direct channels, Bit2 supports the construction of Hash Time-Locked Contract (HTLC) networks similar to the Lightning Network. This allows compatibility with existing Lightning-style protocols. Even though payment channels inherently require locked capital, a Bit2-based Lightning Network offers cheaper channel operations.

Programmable Spending Controls

Programmable Spending Controls are essential for machine-to-machine commerce because autonomous agents operate continuously and may control funds without direct human supervision. Without enforceable constraints, a compromised agent—or even a benign agent interacting with a malicious counterparty—could rapidly drain its entire balance. Attackers may exploit software bugs, manipulate APIs, or craft adversarial interactions that trick the agent into authorizing unintended payments. Spending policies provide a critical safety layer by limiting the

damage radius: rate limits can cap losses over time, whitelists can restrict counterparties to trusted entities, and human-authorization rules can gate high-value transactions. In large-scale agent ecosystems, these controls are not just optional safeguards but fundamental requirements for deploying capital safely, enabling organizations to delegate financial autonomy to software while maintaining predictable risk exposure.

Payment Composability

Programmable spending controls naturally extend to a broader concept: payment composability. Payment composability is the ability to embed payments inside conditional workflows whose execution depends on external events, cryptographic proofs, or other payments.

For example, Alice may create and timestamp a transfer containing several alternative recipients for the same funds. The final recipient is determined by evaluating a predefined condition at some point in the future. Alice distributes the transfer proof to all potential recipients, knowing that only one of them will ultimately satisfy the spending conditions.

This mechanism enables atomic protocols without requiring blockchain scripting. As an example, Alice and Bob may atomically swap two assets. Each party first creates a conditional transfer together with a refund path protected by a timeout. The first party chooses the longer timeout to guarantee safety. Each conditional payment references the proof generated by the other party, ensuring that either both transfers succeed or both participants recover their original funds.

More generally, client-side smart contracts allow entire payment protocols to be expressed as exchanges of cryptographic proofs rather than as globally executed smart contracts.

Bit2 is not the first client-side validation protocol to support programmable logic, but its approach differs from existing systems.

RGB supports client-side contract execution and offers expressive scripting capabilities attached to RGB assets. However, RGB contracts primarily describe asset state transitions, whereas Bit2's virtual machine is designed around programmable payment policies and payment authorization.

Ark has introduced ArkadeOS, which explores programmable payment workflows around Ark's service-provider model. However, execution remains closely tied to Ark's server-assisted architecture, whereas Bit2's scripts are intended to remain valid even if Timestamp Service Providers become unavailable.

By contrast, the Lightning Network provides only a limited form of programmability through HTLCs, PTLCs, and timelocks. These primitives are sufficient for payment routing and atomic swaps but cannot express general spending policies, rate limits, authorization rules, or richer application logic.

Globally-Verifiable Non-Interactive Payment Proofs

Globally-verifiable non-interactive payment proofs capture the ability of a payment system to produce objective, third-party verifiable evidence that a payment has occurred, without requiring ongoing interaction between the involved parties. This property comprises two complementary guarantees: (1) receiver non-repudiation, where the sender can prove that the receiver accepted a payment, and (2) invoice fulfillment verifiability, where the receiver can prove that a specific invoice—identified by a payment hash or ID—has been paid by someone.

This property is particularly important for agentic commerce, where payments are tightly coupled with automated service delivery such as compute, bandwidth, storage, or API access. In these settings, smart contracts or autonomous agents often act as arbiters that must decide whether to release resources, deliver results, or enforce penalties. For such decisions to be reliable, they must be based on **objective and globally verifiable evidence of payment finality**, rather than on private communication or interactive protocols. Without non-interactive proofs, systems must rely on trust assumptions, callbacks, or dispute-prone coordination, which significantly complicates automation and weakens security guarantees.

Bit2 provides globally-verifiable non-interactive payment proofs in both dimensions. For receiver non-repudiation, the sender can obtain a signed receipt from the aggregator before on-chain time-stamping, which becomes binding contingent on the inclusion of the corresponding commitment. This receipt can be presented as verifiable evidence that the payment has been accepted and will be enforced. For invoice fulfillment verifiability, Bit2 allows receivers to embed invoice identifiers into payments, enabling them to later produce publicly verifiable proofs that a specific invoice has been satisfied, independently of who performed the payment. These proofs can be verified by any third party or smart contract without requiring interaction with the sender or the aggregator.

In contrast, systems such as the Lightning Network and INTMAX2 do not provide globally-verifiable non-interactive payment proofs. Lightning payments rely on the revelation of preimages and local channel updates, which do not produce globally verifiable evidence that a payment was completed or that a specific invoice was fulfilled. Similarly, INTMAX2 focuses on client-side validation and compression but does not expose publicly verifiable proofs linking payments to invoice identifiers or acceptance events. As a result, these systems are less suitable for fully automated environments where external parties must independently verify payment outcomes.

Strengths & Weaknesses of Payment Architectures

To complement the quantitative comparison presented earlier, the following table provides a qualitative summary of the main strengths and weaknesses of each payment architecture. Rather than focusing on specific performance metrics, this view highlights the fundamental

trade-offs that characterize each design, including scalability approach, privacy model, operational complexity, and dependency on infrastructure or capital. This perspective is particularly useful for understanding how each system behaves under real-world conditions and which architectural choices make them more or less suitable for large-scale agentic commerce.

System	Core Strengths	Key Weaknesses
Bit2	• Deterministic payments (≈100% success once accepted) • High throughput (up to very high with batching) • Strong confidentiality (STARK/SNARK-based) • No liquidity/collateral requirements • Very low L1 footprint • Uncensorable exit (1-of-N bridge assumption) • Strong support for agent policies & composability • Globally-Verifiable Non-Interactive Payment Proofs	• Relies on advanced cryptography (STARK/SNARKs, recursion, Nova/Stwo) • Requires client-side state/data management (backup burden)
Lightning Network	• Instant settlement within channels • No L1 data per payment (mostly off-chain) • Mature ecosystem and adoption • Strong economic finality inside channels	• Probabilistic success (routing failures) • Liquidity fragmentation and capital lock-up • Amount-dependent success rate • Privacy leaks via routing • Operational complexity (rebalancing, routing) • Payment repudiation and lack of payment receipts
Ark	• Good UX abstraction (virtual UTXOs) • No per-payment routing failures • Efficient batching	• Requires coordinated service providers • Capital/collateral requirements • Expensive exits (complex proofs) • Limited programmability and composability
RGB	• Client-side validation (strong theoretical privacy) • Minimal L1 footprint • No global data publication	• Data availability risks (aggregator can withhold state) • No trust-minimized Bitcoin bridge • Limited throughput in practice • Weak uncensorable exit guarantees
Base (Rollups)	• High throughput vs L1 • Strong composability (smart contracts) • Fast preconfirmations (sequencer)	• Centralized sequencer dependency • High L1 data costs (data availability) • No inherent confidentiality • Long withdrawal/challenge periods
INTMAX2	• High theoretical throughput • Low L1 footprint • Uses ZK compression techniques	• Deprecated • Heavy ZK proving cost (client-side) • Limited privacy (sender set exposure) • Practical throughput constrained by proof generation • Still evolving, unclear real-world performance • Lack of payment receipts

Summary

Agentic commerce introduces a fundamentally different set of requirements from traditional payments. Automated agents cannot tolerate frequent failures, unpredictable routing outcomes, or long confirmation delays. Payments must behave deterministically, scale to massive throughput, preserve confidentiality, and operate without requiring large amounts of locked capital.

Across these dimensions, Bit2 introduces a fundamentally different architecture compared to existing systems. By combining **client-side validation, cryptographic state compression, and minimal L1 commitments**, Bit2 eliminates the need for liquidity routing, dramatically reduces reliance on base-layer blockspace, and allows payments to succeed deterministically once accepted by the time-stamping server. At the same time, transaction histories remain confidential and users retain unilateral control of their funds.

Other architectures each solve parts of the scalability problem but introduce trade-offs. Payment channel networks rely on liquidity routing and suffer from probabilistic payment success. Rollups depend on sequencers and large data publication to the base layer. Virtual-UTXO systems and RGB rely on coordination services that may become operational bottlenecks or data availability risks.

Bit2's design aims to combine **the security of Bitcoin anchoring, the privacy of zero-knowledge proofs, and the scalability required for autonomous economic systems.**

Stay Informed

The agentic economy is approaching faster than many expect. As autonomous agents begin interacting economically, the underlying payment infrastructure will become as critical as the communication protocols that power today's internet.

Bit2 is being designed specifically for this future.

If you want to follow the progress of Bit2, learn about new technical developments, and stay informed about upcoming releases and research, **subscribe to our announcement channels and follow our updates.**

The next generation of programmable commerce is coming—and **Bit2** is being built to power it.

DISCLAIMER: This comparison is intended for informational and research purposes. The figures and conclusions presented are based on publicly available information, current implementations, internal analysis, and known theoretical limits, and should be interpreted as indicative rather than definitive benchmarks. Performance, costs, and technical characteristics may change as protocols and implementations evolve. This document was prepared by Fairgate, the team developing Bit2, and reflects its analysis of the technologies and architectures discussed. This document does not constitute financial, investment, legal, or technical advice.