Fee Rate Derivation in Arkade via Brandt-Style Multiparty Computation

1 Overview

Arkade is a Layer 2 protocol for Bitcoin which batches transactions using a tree with K-radix structure. These batches are chained in a way that supports atomic swap from one batch to another. The atomic swap property requires liquidity to be prefunded and locked for a settlement interval. The more frequently batches are produced, the more liquidity must be locked at any moment.

If an operator wishes to remain neutral (i.e. not warehouse liquidity risk), it must source liquidity externally. External liquidity providers require compensation in the form of an interest rate for locking funds. This interest rate directly maps to Arkade's per-batch fee rate. Therefore, the effective fee rate for a batch is a function of the providers' borrowing cost.

Because liquidity supply and borrowing rates are dynamic, Arkade must discover a new fee rate every batch (or epoch). This is done by auctioning the right to provide liquidity.

However:

- Rate discovery in one batch MUST NOT leak information that can influence future bids.
- The operator MUST NOT be able to pick or bias the fee rate.
- Liquidity providers MUST get fairness: truthful bidding should be optimal.

Arkade plans to solve this by using a fully private, bidder-resolved, $(M + 1)$st-price auction based on the Brandt protocol [1]. The auction has no trusted auctioneer, and reveals only the clearing fee rate and which winners are obligated to lock liquidity for that batch. All other bid information remains cryptographically hidden.

2 Auction Goal in Arkade

At the start of batch t, Arkade requires a total liquidity commitment of $M \cdot k$ units. This is satisfied by selecting M liquidity providers, where each selected provider is expected to supply k units of liquidity (e.g. k units of notional locked for the batch's settlement interval, or k standardized liquidity obligations under Arkade's locking model).

There are n registered liquidity providers (bidders). Each bidder $i \in \{1, \ldots, n\}$ submits a sealed bid b_i which represents the interest rate (cost of liquidity) they require in order to lock one unit for the batch.

Arkade wants to:

G1. Select the M lowest-rate bidders as winners (i.e. the cheapest M liquidity providers).
G2. Set the *clearing fee rate* for all M winners to the $(M{+}1)$st lowest bid.[1]
G3. Reveal this clearing fee rate, but not reveal any other bid amounts or bidder identities to the public.

[1]For $M{=}1$ this reduces to a Vickrey / second-price auction.

Economically, this is just a single-round, sealed-bid, uniform-price auction. Cryptographically, the twist is that we run it *without an auctioneer*, using multiparty computation over homomorphically encrypted bids.

We denote the required clearing rate for batch t as

$$\rho_t \quad := \quad \text{the } (M{+}1)\text{st lowest bid in round } t.$$

Arkade uses ρ_t as the fee rate charged in that batch.

3 Threat Model and Requirements

Arkade adopts Brandt's strongest adversarial assumptions:

- Up to $n-1$ bidders may collude.
- The operator may collude with any subset of bidders.
- There is a public broadcast channel.
- We assume computational security of discrete-log–based cryptography and zero-knowledge proofs.

The following properties are required:

Full Bid Privacy No bid value or ordering information leaks to anyone except what is *strictly* implied by the final allocation and ρ_t. In particular, no losing bidder learns anything about ρ_t other than "I lost".

Correctness / Public Verifiability Everyone (including outside observers) can verify that ρ_t and the winner set are consistent with all committed bids.

No Auctioneer / Bidder-Resolved There is no trusted auctioneer. All cryptographic computation is jointly run by bidders themselves. This is called a *bidder-resolved auction* [1].

Weak Robustness A malicious bidder can be detected and expelled if they deviate. The auction can be rerun without them. Dishonest behavior is therefore attributable and punishable.

Temporal Independence Information revealed in batch t cannot be used to infer individual bids from batch $t-1$ or influence batch $t+1$ bidding. This is critical so the operator cannot shepherd rates by threatening to leak prior auction structure.

4 Cryptographic Primitives Used

Arkade instantiates the Brandt-style protocol using distributed ElGamal over a prime-order subgroup.

4.1 Group Setup

Let p, q be large primes such that $q \mid (p-1)$ and let G_q be the order-q subgroup of $\mathbb{Z}_p^\times$. Let $g \in G_q$ be a generator. All arithmetic below is in G_q.

4.2 Distributed Key Generation

Each bidder i:

1. Samples a private share $x_i \in \mathbb{Z}_q$.
2. Publishes $y_i = g^{x_i}$ along with a Schnorr-style ZK proof of knowledge of x_i.

The joint public key is

$$y = \prod_{i=1}^{n} y_i = g^{\sum_i x_i}$$

and the *joint* private key is the sum

$$x = \sum_{i=1}^{n} x_i \quad \text{(never explicitly reconstructed).}$$

No single party knows x, unless all n collude. This is what enforces full $(n-1)$-privacy.

4.3 ElGamal Encryption

To encrypt a group element $m \in G_q$ under y, a bidder samples fresh $r \in \mathbb{Z}_q$ and publishes

$$(\alpha, \beta) = (m \cdot y^r,\ g^r).$$

ElGamal here is multiplicatively homomorphic:

$$(\alpha_1, \beta_1) \cdot (\alpha_2, \beta_2) = (\alpha_1\alpha_2, \beta_1\beta_2)$$

is an encryption of $m_1 m_2$ if (α_j, β_j) encrypt m_j.

4.4 Distributed Decryption

Given a ciphertext (α, β):

1. Each bidder i publishes a *partial decryption share* β^{x_i} and proves in zero knowledge (Chaum-Pedersen equality-of-discrete-logs proof) that it was correctly formed using the same x_i committed above.
2. Anyone can combine all shares:
$$m = \frac{\alpha}{\prod_i \beta^{x_i}}.$$

No strict "auctioneer" is needed to decrypt; any authorized party can assemble the shares once all of them are revealed.

4.5 Random Exponentiation (Rerandomization / Blinding)

Suppose we have a ciphertext (α, β) encrypting m. We want to transform it into an encryption of m^M where M is a fresh random exponent that *no strict subset of bidders knows.*

Protocol:

1. Each bidder i samples a private mask $r_i \in \mathbb{Z}_q$.
2. Bidder i computes $(\alpha^{r_i}, \beta^{r_i})$ and proves via Chaum-Pedersen that they exponentiated consistently.

3. Multiply all published exponentiations:

$$\prod_i (\alpha^{r_i}, \beta^{r_i}) \;=\; \left(\alpha^{\sum_i r_i}, \beta^{\sum_i r_i}\right)$$

which is an encryption of $m^{\sum_i r_i}$.

We call this *joint random exponentiation.* It lets the group:

- apply bidder-unknown blinding factors,
- permute semantic meaning of ciphertext slots,
- and zero-test specific conditions,

all without revealing any intermediate plaintexts.

Arkade uses these blinded zero-tests instead of ever publishing unblinded intermediate "matrix products."

5 Bid Submission Phase

Arkade discretizes the interest rate interval $[r_{\min}, r_{\max}]$ into k admissible ticks

$$p_1,\ p_2,\ \ldots,\ p_k \quad \text{(e.g. basis point grid or stepped APR quotes).}$$

Bidder a chooses one tick p_{b_a} they are willing to accept. This is their bid.

To commit to this bid without revealing it, bidder a does the following for every tick index $j \in \{1, \ldots, k\}$:

1. Define a "one-hot" component

$$b_{a,j} \;:=\; \begin{cases} Y & \text{if } j = b_a \\ 1 & \text{otherwise} \end{cases} \quad \text{where } Y \in G_q \text{, fixed and } Y \neq 1.$$

   Intuition: Y marks "this is my chosen rate".

2. Encrypt each $b_{a,j}$ under the joint key y:

$$(\alpha_{a,j}, \beta_{a,j}) \;=\; (b_{a,j} \cdot y^{r_{a,j}},\ g^{r_{a,j}})$$

   with fresh randomness $r_{a,j}$.

3. Prove in zero knowledge:
   - each $b_{a,j}$ is either 1 or Y, and
   - exactly one position is Y (i.e. $\prod_j b_{a,j} = Y$),

   using standard OR-proofs / proofs of partial knowledge (e.g. Cramer-Damgård-Schoenmakers).

All these ciphertexts and ZK proofs are broadcast. At this point:

- Everyone is convinced the bids are well-formed.
- Nobody (including the operator) learns b_a.

6 Secure Computation of the Clearing Rate

Now Arkade must jointly derive:

- which bidders are in the cheapest M slots,
- the $(M+1)$st cheapest quote ρ_t,
- and who actually has to lock liquidity.

In Brandt, this is done by computing *indicator ciphertexts* that become 1 exactly at the correct rate level and become a random group element everywhere else. We summarize the adapted flow here, replacing the matrix notation by cryptographic operations.

6.1 Aggregating Demand at Each Rate Level

For each rate tick j, all bidders have published an encryption of $b_{a,j} \in \{1, Y\}$.

Because ElGamal is multiplicatively homomorphic, anyone can compute the ciphertext

$$({}_{j}, {}_{j}) \;=\; \prod_{a=1}^{n} (\alpha_{a,j}, \beta_{a,j})$$

which is an encryption of

$$B_j \;=\; \prod_{a=1}^{n} b_{a,j}.$$

Interpretation:

- $B_j = Y^t$ if t bidders chose tick j.
- $B_j = 1$ if nobody chose tick j.

So, *without decrypting*, we have per-tick ciphertexts encoding "how many bidders want this rate," but encoded multiplicatively as Y^t.

6.2 Cumulative Supply Curve (Encryption-Only)

To know whether we've already satisfied M units of liquidity by some tick j, Arkade needs a cumulative count of bidders who bid *at or below* j.

Critically: we **do not** reveal these counts. Instead, we build encrypted tests of the form "is the cumulative count strictly greater than M?"

Sketch:

1. Form encrypted partial products over ticks:

   $$C_j \;=\; \prod_{d \le j} B_d \quad \text{(still encrypted via homomorphic multiplication of ciphertexts).}$$

   Intuition: in the exponent of Y, C_j encodes the *number of distinct winning-capacity bids* up to tick j.
2. For each j, jointly apply *random exponentiation* (Section 5.4) with bidder-secret exponents to blind C_j into C'_j.
3. For each j, jointly compute (again via homomorphic ops and blinding) an encrypted value that will decrypt to 1 *iff* the cumulative count first exceeds M at tick j, and decrypts to a random non-1 element otherwise.

This produces, for each tick j, a ciphertext WIN_j such that:

$$\mathrm{Dec}(\mathsf{WIN}_j) = \begin{cases} 1 & \text{if } p_j \text{ is the } (M{+}1)\text{st (i.e. clearing) rate} \\ \text{random in } G_q \setminus \{1\} & \text{otherwise.} \end{cases}$$

6.3 Per-Bidder Winner Indicator

Independently, each bidder a also derives a blinded indicator vector of length k, call it

$$\mathsf{IND}_{a,j},$$

such that after final decryption:

$$\mathrm{Dec}(\mathsf{IND}_{a,j}) = \begin{cases} 1 & \text{if bidder } a \text{ is obligated to supply liquidity at tick } j \\ \text{random} & \text{otherwise.} \end{cases}$$

Construction outline:

1. Start from bidder a's encrypted $\{b_{a,j}\}_j$.
2. Combine those with the global encrypted allocation test ("is this bid among the cheapest M?") using only homomorphic multiplication.
3. Apply joint random exponentiation to blind all non-relevant entries.

Result:

- Each winning bidder a will have exactly one position $j^\star$ where $\mathrm{Dec}(\mathsf{IND}_{a,j^\star}) = 1$. That $p_{j^\star}$ is the clearing rate they must honor.
- A losing bidder will get no position that decrypts to 1.

This gives bidder-local clarity without revealing anything global.

7 Fairness / Final Decryption Phase

Brandt notes a subtle fairness issue: someone could learn the result early and abort before others learn it. Arkade resolves this using the operator as a fairness relay without trusting them with bid privacy.

Protocol endgame:

1. Each bidder a sends their partial decryption shares *only* for their own $\mathsf{IND}_{a,j}$ entries (and the global WIN_j entries) to the operator.
2. The operator collects *all* partial decryptions for each ciphertext. Because each partial share is ZK-proofed against the published y_i, the operator cannot forge.
3. Once all shares for a given ciphertext are in, the operator (or anyone else, since at that moment they're public) can compute the plaintext.

This yields two things simultaneously at the end of the round:

- The clearing rate $\rho_t = p_{j^\star}$, where $\mathsf{WIN}_{j^\star}$ decrypts to 1.
- For each winning bidder a, evidence that $\mathsf{IND}_{a,j^\star}$ decrypts to 1, i.e. they are on the hook to lock liquidity at that rate.

Critically:

- No bidder can selectively refuse to reveal a losing indicator share, because losing indicators are just random and don't affect allocation.
- The operator learns *the clearing rate and the winning identities* (needed to coordinate liquidity lock-in), but not the exact losing bids, and not the full rank order of all bids.

8 Tie Handling and Liquidity Rollover

In uniform-price auctions, ties are common and cryptographically it introduces edge cases.

Suppose multiple bidders all submit the same marginal rate that defines ρ_t (i.e. several bids lie exactly at the $(M+1)$st slot). In classical Brandt, naive handling of this can cause the protocol to produce "no unique winner" or leak that a tie occurred at that exact price level.

The mechanism adopts Brandt's "Determining Ties" (`Det`) method, adapted to liquidity rollover:

- Instead of trying to *break* the tie inside this batch, Arkade identifies (still under encryption + blinding) that the $(M+1)$st rate level corresponds to a tie.
- All tied bidders at that marginal rate are considered "conditionally admitted." Arkade can admit as many of them as liquidity requirement actually needs for this batch. The surplus capacity is marked as *rollover liquidity.*
- Rollover liquidity is treated as already-precommitted supply for some future batch, at the same rate ρ_t, *without re-exposing the tie structure.* That is:
  - Next round's auction treats some liquidity slots as already satisfied at cost ρ_t.

  - New bidders in batch $t+1$ only compete to fill the residual demand above that committed rollover.

This has two consequences:

1. The system does not need to publicly reveal how many bidders tied at ρ_t or who they were.
2. The fee rate discovery in round $t+1$ is *not forced* to anchor to ρ_t: new bids are still sealed, and the MPC computation is rerun fresh. The rollover only affects how much *additional* liquidity needs to be purchased, not the cryptographic privacy of the new bids.

Thus we preserve temporal independence: knowing ρ_t does not leak which bidders created that rate, nor how aggressive future bidders must be. The only visible economic signal is the final clearing rate ρ_t used to set the batch fee.

9 Result: Fee Rate and Settlement Enforcement

At the end of the protocol for batch t:

- Arkade obtains ρ_t, the clearing fee rate, and publishes it as the fee coefficient for that batch.
- Each winning bidder has a publicly verifiable proof (from the decrypted $\mathsf{IND}_{a,j^\star}$) that they are obligated to provide liquidity at rate ρ_t.
- The operator (and only the operator plus the winners) learns which public keys correspond to the winning liquidity providers. This mapping need not be revealed globally.
- A Merkle root of all ciphertext transcripts, ZK proofs, and final decrypted indicators can be committed on-chain. Anyone can audit correctness ex post, but cannot reconstruct losing bids.

No other bid amount, ordering, or loser identity is leaked.

10 Economic Properties

Truthfulness / Incentive Compatibility This construction is a sealed-bid uniform-price auction. Under standard private-value assumptions, bidding one's true required rate is a dominant strategy: each bidder either wins and is paid (in effect) the clearing rate ρ_t or loses with no obligation.

Operator Neutrality The operator cannot alter ρ_t because ρ_t is derived from jointly committed ciphertexts plus ZK proofs. Any tampering would be publicly detectable via the transcript.

Temporal Independence Every batch runs a fresh MPC with fresh randomness, fresh joint exponents, and fresh partial decryptions. Only ρ_t leaks. Therefore, no batch can be used to "signal" acceptable future rates beyond revealing the achieved clearing rate itself.

Public Verifiability Auditors can check ZK proofs and homomorphic relations to confirm that:

1. all bids were well-formed (exactly one Y per bidder),
2. all computations were done via allowed homomorphic operations and joint random exponentiations,
3. and the finally decrypted ρ_t is consistent.

11 Overprovisioning of Liquidity as an Open Problem

In the proposed fee rate derivation mechanism, excess or *overprovisioned liquidity* refers to situations where more bidders qualify at or below the clearing rate than are actually required for the current batch. While such overprovisioning appears harmless at first, since surplus liquidity could theoretically be reused in subsequent batches, its existence introduces non-trivial challenges for both economic neutrality and temporal independence.

11.1 Economic Implications

If liquidity beyond the required M units is rolled forward to a future batch, those funds are effectively *anchored* to the clearing fee rate ρ_t of the previous round. This introduces a dependency between consecutive auctions: the subsequent fee rate ρ_{t+1} is no longer determined solely by bids in round $t+1$, but also by the magnitude and price of previously committed liquidity. Such anchoring can distort the intended market dynamics in several ways:

- A coalition of low-rate bidders could strategically over-supply liquidity in one batch to suppress future fee rates, by ensuring that cheap rollover liquidity dominates early allocations in subsequent rounds.
- Conversely, bidders requiring higher yields could attempt to prevent rollover accumulation to maintain persistent scarcity and elevated fee rates.
- The operator's neutrality may be challenged if rollover state is used to indirectly influence future rate discovery.

11.2 Conflict with Temporal Independence

One of Mechnaism's stated goals is to ensure that each batch's fee rate is derived independently of historical rounds. Overprovisioning conflicts with this principle because any carried liquidity effectively transmits information, and economic weight, from the past into the future. Even if the underlying bids remain private, the existence of residual liquidity priced at ρ_t imposes a structural correlation between ρ_t and ρ_{t+1}.

This coupling weakens *temporal independence* and complicates the theoretical analysis of incentive compatibility, since bidders may form expectations about future fee rates based on their ability to influence rollover capacity today.

11.3 Open Research Questions

At present, it remains an open question how to incorporate overprovisioned liquidity without compromising fee rate neutrality. Possible directions include:

1. Designing a mechanism that permits rollover liquidity purely as a *capacity buffer* while excluding it from future price formation;
2. Introducing stochastic or time-decaying weighting to discount the influence of older liquidity on new auctions;
3. Modeling rollover explicitly as part of a dynamic multi-period auction, potentially relaxing strict temporal independence in favor of fee stability.

12 Conclusion

Arkade intends to use a bidder-resolved, fully private $(M+1)$st-price auction (a uniform-price auction) to set its batch fee rate. Each liquidity provider submits an encrypted one-hot interest rate quote. Using only homomorphic multiplication, zero-knowledge proofs of correctness, joint random exponentiation, and distributed decryption shares, the set of cheapest M providers and the clearing rate ρ_t are determined without revealing any losing bids or bidder identities.

Because there is no trusted auctioneer and the operator cannot bias or learn full bid structure, the resulting fee rate is both economically fair and credibly neutral. Because only ρ_t is revealed per batch, and tie-handling is rolled into future batches without exposing structure, Arkade maintains temporal independence: fee discovery in round t does not leak forward into round $t+1$ in a way that lets the operator shape bidding behavior.

This satisfies Arkade's top-level requirement: determine a fair per-batch fee rate that internalizes the cost of external liquidity, while preserving privacy, verifiability, and neutrality.

References

[1] Felix Brandt. Fully private auctions in a constant number of rounds. *Financial Cryptography*, 2003.