Bitcoin Poker

Robin Linus
Glass Coins

Abstract

We describe a two-player protocol for limit Texas Hold'em on Bitcoin. The players generate nine private, distinct cards without a dealer. Commitments to the cards determine public keys whose secrets become known to the recipient when the cards are opened. Bitcoin signatures under these keys authenticate the cards at showdown. Adaptor signatures enforce the release of the openings needed to continue play. A tree of presigned transactions encodes betting, reveals, hand evaluation, and payout; timeouts settle a hand if either player stops. Cooperative play can exchange the same witnesses off-chain, while the on-chain construction provides the fallback. The protocol uses existing Bitcoin operations and requires no consensus changes.

1 Introduction

Bitcoin's first release already contained a poker lobby and table interface, including controls for dealing, folding, calling, and raising [1]. These interface fragments were not a complete poker protocol. They leave an interesting question: how can two players deal private cards and enforce the result of a poker hand using Bitcoin alone?

Bram Cohen's work on Chia Gaming inspired this construction. His discussion of poker identifies both the difficulty of hidden dealing and a useful shortcut: generate only the cards needed for a hand and reject deals containing duplicates [2]. Chia Gaming also provides a concrete setting for two-player games over state channels [3]. Here we develop private dealing and game enforcement for Bitcoin's more restricted scripting system.

The intended mode of play is off-chain. Players exchange messages instead of waiting for blocks. But a cooperative transcript is useful only if a player can finish when the other stops cooperating. In the worst case for our construction, every move along the selected game path must be unrolled on-chain. We therefore construct the complete on-chain game first, then describe its cooperative use.

There are two main parts. The dealing protocol produces hidden cards with Bitcoin-verifiable identities. The settlement protocol turns the finite game into transactions, so that an honest player can enforce a payout without obtaining further signatures from the opponent.

2 Game and assumptions

Alice and Bob play heads-up limit Texas Hold'em. Each receives two private cards and shares a five-card board. We restrict the game to two players because several opponents at one table can secretly pool their cards and coordinate their decisions. Cryptographic dealing cannot detect this outside communication. Heads-up play removes that particular coalition.

The players agree on stacks, blinds, the dealer button, a cap on bets per street, reveal order, timeout delays, fee policy, and payout destinations. The reference rules use blinds u and $2u$, betting increments of $2u$ before and on the flop, and $4u$ on the turn and river. An opening bet and at most three raises are allowed per street. Effective-stack caps handle short all-ins; once no further betting is possible, the board runs out. All amounts are integers in Bitcoin base units.

At most one participant is malicious. The honest participant checks the full setup, retains its secrets and recovery data, and can observe and submit Bitcoin transactions. Confirmation and timeout arguments assume sufficient fees and eventual transaction inclusion. The relay transports messages; it holds no cards, decides no outcomes, and supplies no trusted chain state.

3 Private dealing

Only nine distinct cards are needed. We fix their positions as $(A_1, B_1, A_2, B_2, F_1, F_2, F_3, T, R)$ and sample those positions directly rather than shuffling a complete deck. A card identifier $c \in \{0, \ldots, 51\}$ encodes rank and suit as $c = 4r + s$, with $0 \le r \le 12$ and $0 \le s \le 3$.

3.1 Contributions and commitments

Let $\mathbb{G}$ be the prime-order secp256k1 group, of order q, with standard generator G. Choose a second generator M by domain-separated hash-to-curve using RFC 9380 [4]; nobody should know $\log_G M$. Each player $P \in \{A, B\}$ independently chooses, for each position i, a uniform contribution $v_{P,i} \in \{0, \ldots, 51\}$ and a random blinder $\gamma_{P,i} \in \mathbb{Z}_q$. It publishes the commitment

$$V_{P,i} = v_{P,i}M + \gamma_{P,i}G. \tag{1}$$

The raw sum and card are

$$s_i = v_{A,i} + v_{B,i} \in \{0, \ldots, 102\}, \qquad c_i = s_i \bmod 52. \tag{2}$$

One uniform independent contribution makes the modular sum uniform, even for a fixed other contribution. Commitments hide these small values because the blinding scalars span the group. Publishing $\gamma_{P,i}G$ separately would destroy that hiding: the 52 possible values could be enumerated.

The players first establish a fresh joint ElGamal key $Y = X_A + X_B$, where $X_P = x_P G$. Each proves knowledge of its secret share. Alongside every commitment, the player encrypts the same contribution:

$$C_{P,i} = (R_{P,i}, S_{P,i}) = (r_{P,i}G,\ v_{P,i}M + r_{P,i}Y). \tag{3}$$

Fresh nonzero encryption randomness is used for every position. Neither player alone can decrypt. These ciphertexts are needed only to test whether the nine hidden cards are distinct.

3.2 Proving valid inputs

Each contribution must be in the stated range, and its ciphertext must encrypt the value in its commitment. Otherwise a player could pass a uniqueness test for one deal while using another at settlement.

For the range proof, commit to six bits of v and six bits of $51 - v$. Each bit commitment $B = bM + \rho G$ carries an OR proof of knowledge of the discrete logarithm of either B or $B - M$ to base G [5]. Thus its value is either zero or one. Verify the weighted sums

$$\sum_{j=0}^{5} 2^j B_j = V, \qquad \sum_{j=0}^{5} 2^j \bar{B}_j = 51M - V. \tag{4}$$

Both represented integers lie between 0 and 63. Commitment binding then forces their sum to be 51, so v lies between 0 and 51. Proving six bits of v alone would also admit the invalid values 52 through 63.

A generalized Schnorr proof establishes knowledge of (v, γ, r) satisfying all three equations $V = vM + \gamma G$, $R = rG$, and $S = vM + rY$. The range and link proofs refer to the same public objects. Fiat–Shamir challenges bind the game, player, attempt, protocol stage, and complete statement. Players commit to their key and contribution bundles before either opens them, preventing the second sender from choosing inputs after seeing the first sender's bundle.

3.3 Rejecting duplicate cards

ElGamal ciphertexts add componentwise. Let $C_i = C_{A,i} + C_{B,i}$ encrypt $s_i M$. For every pair $i < j$, compute

$$D_{i,j,d} = C_i - C_j - (O, dM), \qquad d \in \{-52, 0, 52\}. \tag{5}$$

Two cards coincide modulo 52 exactly when one of these three ciphertexts encrypts zero. There are $3\binom{9}{2} = 108$ tests.

Decrypting the differences directly would leak relations between cards. Instead, Alice multiplies each test ciphertext by an independent nonzero scalar α, and Bob multiplies the result by an independent nonzero scalar β. Each supplies proofs that both ciphertext components were scaled by the same scalar, together with a nonidentity commitment to that scalar. The order may be reversed; both scaling rounds are required.

For a final ciphertext (R', S'), each player supplies $Z_P = x_P R'$ with a proof of the same discrete logarithm as its public key X_P. Both commit to their decryption messages before opening them. The public result is

$$S' - Z_A - Z_B = \alpha\beta(s_i - s_j - d)M. \tag{6}$$

Zero survives scaling. A nonzero value is blinded by the honest player's unknown random factor. Under the DDH-based privacy argument, the test reveals whether the difference is zero without revealing its value. Scalars must not be shared between tests, since shared factors preserve relations.

If any result is zero, discard the entire attempt and start with fresh contributions and randomness. An invalid proof is a protocol failure, not a normal collision. With independent uniform contributions, the success probability is

$$p = \prod_{k=0}^{8} \frac{52-k}{52} \simeq 0.48025, \qquad \mathbb{E}[\text{attempts}] = p^{-1} \simeq 2.08223. \tag{7}$$

Conditioning uniform draws on distinctness gives the same distribution as the first nine cards of a uniform shuffle. Rejected attempts expose their collision pattern and must never be reused. This calculation does not bound deliberate aborts.

3.4 Turning a card into a signing key

The accepted commitments determine 103 public keys per position:

$$K_{i,t} = V_{A,i} + V_{B,i} - tM, \qquad 0 \le t \le 102. \tag{8}$$

For the actual raw sum $t = s_i$, the message terms cancel:

$$K_{i,s_i} = (\gamma_{A,i} + \gamma_{B,i})G = \kappa_i G. \tag{9}$$

A player who receives both openings knows s_i and κ_i, and can produce a Bitcoin Schnorr signature under this key. For a wrong candidate, the remaining term $(s_i - t)M$ has unknown discrete logarithm. Knowing both κ_i and a wrong candidate's scalar would reveal $\log_G M$.

All 927 candidate points are derived locally from the verified setup. Before acceptance, reject identity points, equal x-only keys across candidates or positions, and collisions with protocol signing keys. Full point signs are retained during algebra; BIP340 parity normalization is applied at signing [6]. Ciphertext degeneracies that expose small plaintexts are also rejected. Both players sign the resulting setup certificate before the deal is used. They then erase threshold decryption shares and proof randomness, retaining the contribution openings needed for play.

At showdown, a witness supplies t and a signature. Script checks $0 \le t \le 102$, selects the corresponding embedded x-only key, and verifies the signature on the spending transaction. It obtains the card identifier by subtracting 52 if $t \ge 52$. No elliptic-curve arithmetic or modular division is performed inside Script: the key catalogue is prepared off-chain.

4 Enforcing card delivery

An ordinary card signature proves knowledge of a signing secret but does not reveal it. Later transactions need reusable openings. Simply replacing a hashlock with a card signature would therefore leave the game unable to continue without another message from the opponent.

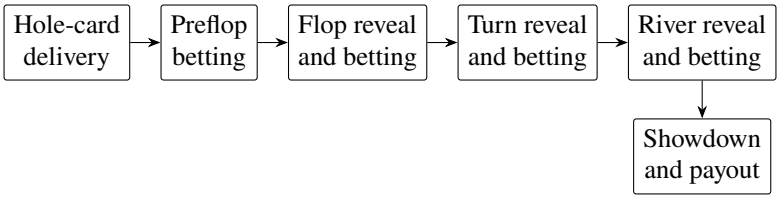

Figure 1: A showdown path through one hand. Each reveal contains both players' required deliveries. Betting nodes also have fold exits; every outstanding obligation has a timeout exit. All-ins omit further betting.

Consider a contribution commitment $V = vM + \gamma G$ that Alice must open. For each possible value w, define

$$T_w = V - wM, \qquad w \in \{0, \ldots, 51\}. \tag{10}$$

Bob provides a Schnorr adaptor presignature under a dedicated authorization key, encrypted to each T_w, for the exact reveal transaction. Alice verifies all 52 candidates before activation. She knows $\log_G T_v = \gamma$ for her actual contribution, so she can complete that presignature. The resulting ordinary BIP340 signature satisfies the reveal script.

Bob extracts γ from the completed signature and its presignature. He recovers v by checking the 52 possibilities in Equation (1). The full signed point and the final signature are verified during extraction. The extracted opening can then be combined with his own opening to derive the card key in Equation (9). Bitcoin need only verify the completed signature; the connection between signature and disclosure was checked during preparation.

Each contribution position has a separate opponent-controlled reveal key. This prevents a signature for one position from satisfying another position's check. A three-card reveal requires three such signatures plus the revealer's live authorization. Presignatures bind the hand, node, position, commitment, key, and transaction digest. Nonces must also be separated by these contexts and by the adaptor point. No ordinary signature under a reveal key is distributed in place of its adaptor package.

For Alice's hole cards, Bob releases his contributions; Alice keeps hers secret. Even if Bob's openings appear on-chain, outsiders still lack Alice's shares. Bob's hole cards are handled symmetrically. For a community card, both contributions are released in the agreed order. The second revealer learns the card first, but cannot skip its own reveal and proceed to betting. Refusal enables the opponent's timeout payout. At showdown, card signatures disclose the claimed card values; unused future cards remain unopened.

5 Representing the game in Bitcoin

The rules define a finite tree. A node fixes whose turn it is, the street, the outstanding obligation, committed wagers, remaining stacks, and fee reserve. An edge is a legal action, a card delivery, a showdown step, or a timeout. Both players independently derive the tree from the same accepted deal and agreed terms. Figure 1 shows its main phases.

5.1 Transactions as state transitions

Each nonterminal node is represented by a Taproot output. Its leaves enforce the conditions for leaving that state. The internal key is a point with no known signing scalar, so neither player has a key-path bypass. Each edge has a fixed transaction spending the node output into its successor or into final payouts. The transaction bodies, including destinations and amounts, are known before play; card openings appear only in witnesses.

For an ordinary action, the opponent supplies its signature in advance. The player whose turn it is chooses a legal edge by adding its own live signature. The opponent's signature fixes the successor output and fee using the Taproot transaction digest [7]. It cannot be redirected to an arbitrary payout. Since witnesses do not determine transaction identifiers, descendant transactions can be prepared before the cards are known.

This is how most game logic is enforced: the honest participant signs only the legal transitions, and the transaction graph restricts what those signatures authorize. Bitcoin executes the remaining witness predicates, including card authentication and hand evaluation. It does not rerun the off-chain dealing proofs. Their validity is checked before the honest participant authorizes activation.

All counterparty signatures and adaptor packages required by every reachable branch must be verified and saved before the hand becomes spendable. The player also saves the secrets needed for its own live signatures. A funding protocol must provide a refund if setup fails before this barrier. After activation, an honest player can fulfill its own obligations or take the applicable timeout without new cooperation. An incomplete signature inventory is not a playable hand.

5.2 Betting and accounting

Fixed limits keep the branch set finite. A node offers only the legal subset of fold, check, call, bet, and raise. It fixes the exact amount of each action, including short calls, effective-stack caps, and the big blind's option after a preflop call. Invalid actions have no preauthorized transaction.

The coins need not move between separate stack and pot outputs on each bet. Let a and b be the uncommitted balances, p all funds committed to the hand, and f the remaining fee reserve. The live output contains

$$U = a + b + p + f. \tag{11}$$

A bet of x by Alice changes (a, p) to $(a - x, p + x)$; an edge fee ϕ changes f to $f - \phi$. The output value decreases only by that fee. Street contributions are tracked separately to determine calls and raises. The symbol p includes current wagers, not merely chips already collected in the center of a displayed table.

At a fold, both uncommitted balances are returned and the pot goes to the other player. At showdown, the pot goes to the winner or is split; a fixed rule assigns an odd remainder. Unused fee reserve is returned under the agreed policy. Every terminal transaction binds the resulting exact amounts.

5.3 Timeouts

Every state waiting for a player also offers an opponent timeout spend. Its script uses a relative block delay with `CHECKSEQUENCEVERIFY` [8]. The defaulting player's authorization is already available; after the delay, the beneficiary adds its own signature. Delays are measured from confirmation of the relevant output, not from a relay message or local clock.

Timeouts return both uncommitted balances and award the committed pot to the nondefaulting player. They do not confiscate the opponent's unused stack. This applies to missing betting actions, card deliveries, and showdown witnesses. In particular, refusing a losing reveal does not recover the pot. The protocol can force settlement, but it cannot force someone to reveal a card; a timeout may end the hand with that card still hidden.

6 Showdown without a trusted evaluator

The remaining problem is to compare private hands on Bitcoin. Each player has seven cards: two hole cards and the common board. The scripts authenticate all seven, evaluate a selected five-card subset, and compare numerical scores.

6.1 Authenticate seven, evaluate five

The witness supplies one candidate index and transaction signature for each of the player's seven positions. Script selects the keys from Equation (8), checks the signatures, reconstructs the seven card identifiers, and checks distinctness. A subset index selects one of the $\binom{7}{5} = 21$ possible five-card hands.

The witness also supplies each selected card's rank and suit. Script checks their bounds and the equation $c = 4r + s$ using additions. Category-specific leaves verify pairs, trips, straights, flushes, and the other

hand patterns, including the ace-low straight. Tie-breaking ranks are checked against the cards, sorted or grouped as required by the category. A claimed score is packed as

$$h = k\,16^5 + r_1\,16^4 + r_2\,16^3 + r_3\,16^2 + r_4\,16 + r_5, \tag{12}$$

where $k \in \{0, \ldots, 8\}$ is the category and the remaining fields contain the category's ordered rank components, with unused fields set to zero. Integer order agrees with poker order. The score fits in 24 bits and hence within Script's arithmetic range. Multiplication by fixed powers of two is implemented by repeated doubling.

The script proves a lower bound on hand strength. It checks the positive facts for the chosen category without proving the absence of a stronger one. For example, a straight flush also supports a flush claim. Nor does it prove that the chosen subset is the best of all 21. Neither relaxation lets a player overstate its hand. Choosing a weaker claim can only worsen that player's result; the honest client chooses its maximum score. This avoids evaluating every subset and every excluded category on-chain.

6.2 Carrying a score between transactions

Alice proves her score first. Bob then proves his own score and spends to one of three fixed payout transactions: Alice wins, Bob wins, or a split. Bob's script cannot read Alice's previous witness. Enumerating a distinct successor transaction for every possible score would also be wasteful.

Instead, Alice's score is authenticated by a Lamport one-time signature [9]. Before activation, she commits to two secret preimages per score bit, publishing their hashes:

$$H_{j,0} = \mathrm{H}(z_{j,0}), \quad H_{j,1} = \mathrm{H}(z_{j,1}), \qquad 0 \le j < 24. \tag{13}$$

For a score with bits b_j, the certificate reveals z_{j,b_j} for each position. Alice's showdown script verifies these hashes, reconstructs the integer, and requires it to equal her card-backed score.

Bob copies that certificate into his payout witness. The next script checks it against the same committed hash pairs, computes Bob's card-backed score, and verifies the comparison required by the selected payout branch. Bob has a separate score key and certificate, checked against his own evaluation. Both score keys are bound to this hand and their distinct roles.

For an honest Alice, Bob cannot change her certified score without finding an unrevealed preimage. A malicious Alice can disclose extra preimages, but cannot remove the valid certificate Bob has already obtained from her confirmed showdown. Bob can always use that certificate with his best hand. Alice also cannot complete Bob's payout without his live signature. If Bob refuses, Alice takes the showdown timeout. This argument relies on an honest player signing only one score with a given key, even across crashes or retries; the chosen score and key usage must be persisted before release.

7 Security and cost

The intended security argument assumes discrete-log and DDH hardness in secp256k1, an unknown relation between G and M, sound Fiat–Shamir proofs, BIP340 unforgeability in this composition, and secure hashes for commitments and one-time signatures. The algebra explains why valid openings yield the right keys; it is not a formal proof of the complete malicious-party protocol. The proof composition and adaptor extension require independent review.

Privacy has explicit limits. A successful setup discloses that the nine cards are distinct. Failed attempts disclose collision relations. Public showdowns disclose both hands, and on-chain execution discloses the played path. A player may abort setup, delay play, or stop after learning information. Timeouts protect settlement after activation; they neither ensure timely cooperation nor establish unbiased completed-hand statistics against selective abort. Compromised endpoints are outside the model.

Preparation costs cover the whole tree, while chain costs cover only one path. A reference configuration with 100 big blinds per player and four bets per street has 56,132 logical nodes and a maximum path of 33 gameplay transitions [10]. Its recorded single-tree inventory contains 54,855 ordinary signatures and 1,306

reveal packages. Each reveal package contains 52 adaptor signatures. These are concrete reference counts, not asymptotic bounds or counts for a two-owner channel.

An accepted dealing certificate occupies 102,070 bytes; a reveal package, 3,412 bytes. The reference public preparation snapshot is 8,416,186 bytes. Three recorded four-worker desktop browser trials prepared and persisted the single tree in 26.39–27.26 seconds, excluding relay latency and funding confirmation [10]. These are fixture measurements, not startup guarantees.

For a chosen fee schedule, reserve at least

$$F \geq \max_{\pi} \sum_{e \in \pi} \phi_e, \tag{14}$$

where π ranges over all reachable settlement paths, including timeouts. Fees must account for actual witnesses, especially the larger showdown scripts, and any channel contest transactions. A fixed presigned schedule does not ensure timely inclusion if fee conditions change. Recovery must also handle reorgs and competing spends. Existing implementation tests exercise card gates, reusable reveals, payouts, and timeout paths with Bitcoin Core; test coverage is not a security proof.

8 Cooperative play

During cooperative play, the players exchange and verify transaction witnesses, retaining them for recovery without broadcasting them. Several hands can share one funding output. A cooperative close pays the agreed balances directly from it.

A player who has learned a future card must not be able to return to an earlier decision. The channel prepares one owner-specific transaction tree per player. Retirable outputs combine a delayed continuation with an immediate counterparty spend enabled by a revocation secret. Accepting a move retires abandoned alternatives in both trees while preserving the selected path. Terminal outputs need the same protection.

Interrupted exchanges must remain recoverable, including after partial card disclosure. Witnesses and retirement evidence are saved before acknowledgment. A new hand uses fresh card and score secrets and exact settled balances; its recovery paths must be ready before the old hand is retired. Players monitor the chain during contest windows; the relay provides no recovery service.

Chia Gaming likewise combines off-chain play with a chain-enforced exit [3]. Our Bitcoin fallback uses the prepared tree and revocation paths. Unrolling it can require every selected game transaction and repeated contest delays. Off-chain play avoids this cost only while the players cooperate.

References

[1] S. Nakamoto. *Bitcoin v0.1.0 source code*, poker interface classes in `uibase.cpp`. Archived source, 2009.

[2] B. Cohen. *Ongoing Chialisp Development*, 2023. bramcohen.com.

[3] Chia Network. *Chia Gaming: Overview of State Channels*. Project documentation.

[4] A. Faz-Hernandez et al. *Hashing to Elliptic Curves*. RFC 9380, 2023.

[5] R. Cramer, I. Damgård, and B. Schoenmakers. *Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols*. CRYPTO, pp. 174–187, 1994. CWI archive.

[6] P. Wuille, J. Nick, and T. Ruffing. *Schnorr Signatures for secp256k1*. BIP 340, 2020.

[7] P. Wuille, J. Nick, and A. J. Towns. *Taproot: SegWit Version 1 Spending Rules*; and P. Wuille, A. J. Towns, and J. Nick, *Validation of Taproot Scripts*. BIP 341 and BIP 342, 2020.

[8] BtcDrak, M. Friedenbach, and E. Lombrozo. *CHECKSEQUENCEVERIFY*. BIP 112, 2015.

[9] L. Lamport. *Constructing Digital Signatures from a One Way Function*. Technical Report CSL-98, SRI International, 1979. Research archive.

[10] R. Linus. *Bitcoin Poker: implementation and reference measurements*. Accompanying source distribution, September 2026. Source locations and measurement scope are recorded in the whitepaper README.