RustProfit

Provably Fair Whitepaper

How every game outcome is generated, committed, and independently verified.

rustprofit.com

1. Introduction

RustProfit is a Rust skin gambling platform built on transparency and trust. Every game on the platform uses a provably fair system — meaning any player can independently verify any result after the fact, without relying solely on our word.

That said, any provably fair system involves some level of trust in the people running it. Anyone who tells you otherwise is not being honest. Our system is designed to be as simple and tamper-resistant as possible, but you must trust that we implement it correctly.

2. SHA-256 — The Cryptographic Foundation

Every game outcome on RustProfit is generated using the SHA-256 algorithm — the same cryptographic standard that secures the Bitcoin blockchain. It takes a string of inputs and always produces a unique 64-character hexadecimal hash.

SHA-256 has been the cryptographic standard since 2001 and has never been broken. It is open-source and publicly auditable — there is no secret or proprietary method involved.

Key properties of SHA-256:

— **Deterministic** — the same input always produces the same output.

— **Collision resistant** — different inputs produce different outputs.

— **One-way** — given an output, it is computationally impossible to derive the original input.

— **Avalanche effect** — even a tiny change to the input creates a completely different output.

3. Key Components

Every game outcome is produced by combining three inputs through SHA-256.

3.1 Server Seed

Generated by RustProfit at the start of each round. Before the round begins, we publish the **SHA-256 hash of the server seed** — this acts as a commitment. Because SHA-256 is one-way, we cannot change the seed after publishing its hash without it being detectable. Once the round ends, the original unhashed server seed is revealed so players can verify it.

3.2 Public Seed

An input entirely outside RustProfit's control. The exact source varies by game mode (see Section 4), but in all cases it is either publicly available data or data from the EOS blockchain that does not yet exist when bets are placed — making it impossible for us to predict or manipulate.

3.3 Round ID

The unique identifier for each round. This ensures that even if the server seed and public seed were identical across two rounds, each round still produces a distinct result.

4. Game Modes

4.1 Roulette

The result is generated **before** bets are placed.

— The **public seed** is a concatenation of five pairs of random numbers (01–39), generated daily and visible to all players.

— The **server seed** is a 32-byte SHA-256 hash generated at the same time as the public seed.

— The **round ID** is the unique identifier for the round.

These three values are combined and processed through HMAC-SHA256. The result is converted to a decimal and reduced modulo 15 to determine the roll (0–14). The roll maps to a colour: red, black, or green.

Roll mapping:

— Red — positions 0, 2, 4, 6, 9, 11, 13

— Black — positions 1, 3, 5, 8, 10, 12, 14

— Green — position 7

Verification script: https://3v4l.org/fPMeU

4.2 Jackpot

The result is generated **after** bets close.

— The **server seed** is a 32-byte SHA-256 hash generated at the start of the round.

— The **public seed** is the hash of a specific EOS blockchain block chosen before bets close but not yet generated — outside our control.

— The **round ID** is the unique identifier for the round.

The three values are combined through HMAC-SHA256, converted to a decimal, and reduced modulo 10^8 to produce the jackpot result. This is mapped to a ticket position to determine the winner, weighted by each player's share of the pot.

Verification script: https://3v4l.org/Xujdq

4.3 Coinflip

The result is generated **after** both players have joined.

— The **server seed** is a 32-byte SHA-256 hash generated at the start of the game.

— The **public seed** is the hash of a specific EOS blockchain block chosen before the flip, outside our control.

— The **round ID** is the unique identifier for the game.

The same HMAC-SHA256 process as Jackpot is used. The result modulo 10^8 determines the winner.

Verification script: https://3v4l.org/7JjEu

4.4 Giveaways

Giveaway winners are drawn without replacement — the same person cannot win twice.

— The **server seed** is a 32-byte SHA-256 hash generated when the giveaway is created.

— The **second input** is an incrementing counter (0 to n) representing each winner draw.

Each winner is determined by hashing the server seed with the current counter value and applying modulo to the total number of tickets.

Verification script: https://3v4l.org/4D3fd

5. Verifying a Result

After each round ends, the unhashed server seed is published. Any player can independently recreate the result by inputting the server seed, public seed, and round ID into the verification script for the relevant game mode.

If the script output matches the recorded result, the round has been mathematically proven fair and unchanged.

Verification scripts by game mode:

— Roulette — https://3v4l.org/fPMeU

— Jackpot — https://3v4l.org/Xujdq

— Coinflip — https://3v4l.org/7JjEu

— Giveaways — https://3v4l.org/4D3fd

Scripts are written in PHP and can be run at 3v4l.org without any setup required.

6. Trust Model

Our provably fair system minimises the trust required in RustProfit, but does not eliminate it entirely. You must trust that:

— We correctly implement the SHA-256 algorithm as described.

— We do not substitute seeds after committing to their hashes.

— The EOS blockchain block data used as public seeds is fetched accurately.

All of the above can be audited. The hashed server seed is published before each round so any substitution would be detectable. EOS block data is public. The verification scripts let you reproduce results independently.

We have nothing to hide, and the tools to check are always available on site.

7. Questions & Support

If you believe a round result is incorrect, visit the provably fair page, find the round in the game history, and use the verification tool. If the result still appears wrong after verification, contact us immediately:

contact@rustprofit.com

Include the game mode, round ID, and any relevant details. We investigate every report.

RustProfit staff will never contact you first. All support starts at rustprofit.com.