قرارداد هوشمند برگزاری مناقصه (Tender)

هادی ثقفی

نحوه اجرا: روی Hardhat در سیستم عامل لینوکس Ubuntu و روی شبکه localhost تست شده (در ویدئو دمو) ولی روی تست نت sepolia هم قابل اجراست (براساس نوع شبکه باید آدرس مناسب کانترکتها در فایل constants.js در فولدر tender-front انتخاب شود)

فایلها و فولدرهای مهم:

فولدر Contracts : سورس کد کانترکت اصلی (Tender) و کانترکت TestUSDT

فولدر test/unit : تستهای قرارداد به زبان javascript

فولدر tender-front : front-end قرارداد به زبان html و javascript

فایل tender-hashes.txt : نمونه های هش پروپوزال و هش قیمت استفاده شده جهت تست

فایل README.md : شرح کلی پروژه

فایل design_pattern_decisions.md (و pdf.): مهمترین دیزاین پترنهای مورد استفاده

فایل avoiding_common_attacks.md (و pdf.): مقاوم بودن قرارداد در مقابل حملات متداول

شرح پروژه:

این قرارداد هوشمند مراحل برگزاری یک مناقصه فنی جهت انتخاب مجری یک پروژه را به شرح زیر مدیریت می کند. مجری پروژه باید مشخصات فنی قابل قبول داشته باشد و حداقل قیمت را پیشنهاد دهد. با توجه به متغیر بودن قیمت اتریوم، برای پیشنهاد قیمت و وثیقه شرکت در مناقصه از استیبل کوین USDT استفاده می شود. بدین منظور با استفاده از کتابخانه OpenZeppelin یک توکن تستی ایجاد و deploy می شود.

۱- مرحله Hashed Proposal Send

در این مرحله شرکت کنندگان در مناقصه پیشنهاد فنی (پروپوزال) خود را مثلا در یک سرور ipfs بارگزاری می کنند و آدرس آن را بصورت هش شده برای قرارداد هوشمند ارسال می کنند. این کار جهت مخفی ماندن پیشنهادات از دید سایر شرکت کنندگان انجام می شود. همچنین جهت جلوگیری از سوء استفاده و یا شرکت سوری افراد، هر شرکت کننده لازم است مبلغی که توسط ادمین (owner) تعیین شده است را به عنوان وثیقه به کانترکت بفرستد

۲- مرحله Revealed Proposal Send

در این مرحله اصل آدرس پیشنهاد فنی ارسال می گردد. در این مرحله فقط پروپوزالی قبول می شود که هش آن با مقدار ارسال شده در مرحله قبل یکسان باشد.

۳- مرحله Review and Approve/Reject the Proposals

در این مرحله پیشنهادات فنی توسط ادمین (owner) سیستم بررسی می شود و پیشنهاداتی که قابل قبول باشد تایید می گردد.

۴- مرحله Hashed Price Send

در این مرحله، شرکت کنندگانی که در مرحله قبل تایید شده اند اجازه ارسال قیمت دارند. جهت مخفی ماندن قیمت ارسالی، قیمت در این مرحله بصورت هش شده ارسال می شود. برای اینکه قیمت ارسالی قابل حدس توسط سایر شرکت کننده ها نباشد، قیمت به همراه یک کلمه دلخواه به عنوان salt هش می شود.

۵- مرحله Revealed Price Send

در این مرحله شرکت کنندگانی که در مرحله قبل قیمت هش شده را ارسال کرده اند، باید قیمت واقعی به همراه salt انتخاب شده را ارسال کنند. مقدار ارسال شده در صورتی قابل قبول است که هش آن (قیمت + salt) با مقدار ارسال شده در مرحله قبل یکسان باشد. قیمت پیشنهادی نباید از حداکثر قیمت تنظیم شده توسط ادمین بزرگتر باشد.

۶- مرحله withdraw

در این مرحله می توان برنده نهایی مناقصه را با فراخوانی تابع مربوطه از کانترکت مشخص نمود (حداقل قیمت از بین کسانی که در مرحله ۳ تایید شده اند). سایر شرکت کنندگان می توانند با فراخوانی تابع withdraw وثیقه خود را برداشت کنند ولی برداشت وثیقه برای برنده مناقصه امکان پذیر نیست.

همچنین شرکت کنندگان می توانند در هریک از مراحل پروژه با فراخوانی تابع withdraw از ادامه مناقصه انصراف دهند و وثیقه خود را دریافت کنند.

شرح توابع موجود در قرارداد:

constructor(address USDT_address)

آدرس قرارداد توکن USDT را دریافت و ذخیره می کند (جهت راحتی deploy در شبکه های مختلف). همچنین آدرس deployer را به عنوان ادمین ثبت می کند

set_new_tender(uint256 col, uint256 p_max, uint256 t_h_st, uint256 t_h_end ,

uint256 t_r_st, uint256 t_r_end, uint256 p_h_st, uint256 p_h_end ,

uint256 p_r_st, uint256 p_r_end, uint256 w_st, uint256 w_end)

تعریف مناقصه جدید و تنظیم مشخصات آن. فقط توسط ادمین قابل اجرا است. این تابع فقط بعد از ریست کردن قرارداد هوشمند توسط یکی از دو تابع *reset_tender* یا *forced_reset_tender* قابل اجراست

ورودی ها به ترتیب:

- مقدار وثیقه مورد نیاز برای هر شرکت کننده
- حداکثر قیمت قابل قبول
- زمان شروع مرحله ۱
- زمان پایان مرحله ۱
- زمان شروع مرحله ۲
- زمان پایان مرحله ۲ و شروع مرحله ۳
- زمان پایان مرحله ۳ و شروع مرحله ۴
- زمان پایان مرحله ۴
- زمان شروع مرحله ۵
- زمان پایان مرحله ۵
- زمان شروع مرحله ۶
- زمان پایان مرحله ۶

reset_tender(address balance_receiver)

این تابع را فقط می توان بعد از پایان مرحله ۶ اجرا نمود. فقط توسط ادمین قابل اجراست. همه متغیرهای حالت و آرایه ها استفاده شده (شامل فهرست آدرسهای شرکت کننده، آدرسهای تاییده در مرحله ۳ و آرایه های مپ شده برای ذخیره هش آدرس پروپوزال، آدرس پروپوزال، هش قیمت و قیمت واقعی) را تخلیه و به مقادیر اولیه باز می گرداند. همچنین باقیمانده موجودی قرارداد (شامل وثیقه برنده و هر شرکت کننده ای که تا پایان مرحله ۶ برداشت نزده باشد) را به آدرس داده شده به عنوان ورودی انتقال می دهد.

forced_reset_tender()

این تابع را فقط ادمین در هرزمانی می تواند اجرا کند و با اجرای آن کل فرایند مناقصه متوقف شده و وثیقه همه شرکت کنندگان به ولت آنها بازگشت داده می شود و همچنین شبیه ریست معمولی همه متغیرهای حالت به وضعیت اولیه برگردانده می شوند و قرارداد آماده تعریف مناقصه جدید می شود. با توجه به حساسیت این کار، لازم است ابتدا تابع *approve_forced_reset* جهت تایید *forced reset* فراخوانی شود.

approve_forced_reset()

باتنظیم یک *flag* تابع *forced_reset_tender* را آماده اجرا می کند و فقط توسط ادمین قابل اجراست

cancel_forced_reset()

عملکرد قبلی تابع *approve_forced_reset* را خنثی و کنسل می کند

send_proposal_hash (bytes32 proposal_h)

چنانچه در زمان مرحله ۱ باشیم، هش پروپوزال را دریافت و آن را در آرایه ای مپ شده (*proposal_hashed*) ذخیره می کند. همچنین آدرس شرکت کننده را در آرایه دیگری (*proposal_participants*) ذخیره می کند

send_proposal_reveal(bytes32 proposal_r)

چنانچه در زمان مرحله ۲ باشیم، آدرس واقعی پروپوزال را دریافت و آن را در آرایه ای مپ شده (*proposal_revealed*) ذخیره می کند به شرط آنکه هش آن با مقدار دریافتی قبلی یکسان باشد.

approve_proposal(address participant)

چنانچه در مرحله ۳ باشیم، آدرس دریافت شده به عنوان ورودی را در آرایه شرکت کنندگان تایید شده (*approved_participants*) ذخیره می کند. این تابع فقط توسط ادمین قابل اجراست.

reject_proposal(address participant)

چنانچه در مرحله ۳ باشیم و آدرس دریافت شده به عنوان ورودی قبلا در آرایه شرکت کنندگان تایید شده (*approved_participants*) ذخیره شده باشد، آن را حذف می کند. این تابع فقط توسط ادمین قابل اجراست.

send_price_hash(bytes32 price_h)

چنانچه در زمان مرحله ۴ باشیم، هش قیمت پیشنهادی را دریافت و آن را در آرایه ای مپ شده (*price_hashed*) ذخیره می کند. قیمت به شرطی دریافت می شود که از طرفی آدرسی باشد که قبلا در آرایه شرکت کنندگان تایید شده (*approved_participants*) ثبت شده باشد.

send_price_reveal(uint256 price, string calldata salt)

چنانچه در زمان مرحله ۵ باشیم، مقدار واقعی قیمت پیشنهادی (*price*) و *salt* مورد استفاده را به عنوان ورودی دریافت می کند و در صورتی که هش آنها با مقدار ارسال شده قبلی تطابق داشته باشد، قیمت پیشنهادی را در آرایه ای مپ شده (*price_revealed*) ذخیره می کند. همچنین در مقایسه با قیمتهای قبلی، برنده را مشخص و یا در صورت نیاز به روزرسانی می کند.

withdraw()

جهت دریافت وثیقه در پایان مناقصه توسط شرکت کنندگان برنده نشده و یا انصراف از ادامه مناقصه در مراحل اولیه توسط هر شرکت کننده قابل فراخوانی است.

remove_from_approved_participants(uint256 index)

حذف شرکت کننده انصرافی از آرایه *approved_participants*. این تابع *private* بوده و فقط از داخل قرارداد قابل فراخوانی است.

remove_from_proposal_participants(uint256 index)

حذف شرکت کننده انصرافی از آرایه *proposal_participants*. این تابع *private* بوده و فقط از داخل قرارداد قابل فراخوانی است.

delete_all_data()

حذف تمام داده ها و بازگرداندن مقادیر متغیرها و آرایه ها به مقادیر اولیه. این تابع *private* بوده و فقط از داخل قرارداد قابل فراخوانی است (توسط توابع *reset_tender* و *forced_reset_tender*).

getStage()

دریافت مرحله فعلی مناقصه براساس *block.timestamp* (عدد خروجی این تابع کمی با مراحل تعریف شده در این داکیومنت تفاوت دارد ولی بصورت کامنت در سورس کد قرارداد معرفی شده است)

get_number_of_participants()

دریافت تعداد شرکت کنندگان در مناقصه (تعداد عناصر آرایه *proposal_participants*).

get_participant(uint256 index)

دریافت آدرس ولت شرکت کننده عنصر ***index*** از آرایه ***proposal_participants***

get_proposal(address participant)

دریافت پروپوزال شرکت کننده با آدرس ولت ***participant***

is_approved(address participant)

مشخص می کند آیا پروپوزال شرکت کننده با آدرس ولت ***participant*** تایید شده است یا خیر (خروجی ***true/false***)

getWinner()

دریافت آدرس ولت برنده مناقصه

getBestPrice()

دریافت حداقل قیمت پیشنهادی (توسط برنده مناقصه)

get_sha256(bytes32 input)

دریافت هش ورودی (با استفاده از تابع ***keccak256***)

get_sha256_salt(uint256 price, string calldata salt)

دریافت هش ترکیب ورودیهای ***price*** و ***salt*** (با استفاده از تابع ***keccak256*** برای هش و ***abi.encodePacked*** برای ترکیب ورودیها)

شرح ***Modifier*** ها

onlyOwner()

چک می کند که صرفا ادمین سیستم، تابع را فراخوانی کند.

correctTiming(uint256 start_time,uint256 end_time)

براساس *block.timestamp* و ورودی های دریافتی به عنوان زمان شروع و پایان، بررسی می کند که در زمان مرحله خاصی از مناقصه باشیم (توسط توابعی که فقط در مراحل خاصی از مناقصه قابل فراخوانی هستند استفاده می شود.)

Event ها

تمام توابع اصلی قرارداد در پایان کار ایونت مشخصی را صادر می کنند که لیست آنها در ادامه آمده است و نام هرکدام مشخص می کند مربوط به کدام تابع است.

event newTender(uint256 indexed proposal_hash_start, uint256 proposal_hash_end ,

uint256 proposal_reveal_start, uint256 proposal_reveal_end,

uint256 price_hash_start, uint256 price_hash_end ,

uint256 price_reveal_start, uint256 price_reveal_end ,

uint256 withdraw_start, uint256 withdraw_end;(

event TenderReset;()

event TenderForcedReset;()

event TenderForcedResetApproved;()

event TenderForcedResetCanceled;()

event HashedPropsalRegistered(address indexed participant);

event RevealedPropsalRegistered(address indexed participant);

event PropsalApproved(address indexed participant);

event PropsalRejected(address indexed participant);

event HashedPriceRegistered(address indexed participant);

event RevealedPriceRegistered(address indexed participant);

event Withdraw(address indexed participant, uint256 value);

شرح مختصر تستهای نوشته شده برای قرارداد

تعداد ۴۷ تست به زبان جاوااسکریپت برای قرارداد نوشته شده که داخل فولدر تست قراردارد و داری توضیحات کافی بصورت کامنت است ولی اینجا هم بصورت مختصر توضیح داده می شود

describe "constructor"

it "sets the USDT contract address correctly"

بررسی میکند که آدرس کانترکت ***USDT*** به درستی توسط ***constructor*** ثبت شده باشد

describe "set_new_tender"

it "Fails if not Owner"

تست میکند در صورتی که اکانتی غیر از ***owner*** تابع را فراخوانی کند، تراکنش ***revert*** شود

it "Fails if sequence of timing is not correct"

تست میکند در صورتی که ورودیهای تابع توالی زمانی درستی نداشته باشد (زمانها متوالی نباشد)، تراکنش ***revert*** شود

it "Should correctly set value of variables"

بررسی میکند تابع مقادیر متغیرها را به درستی تنظیم کند

it "Fails if called before current tender finished"

تست میکند در صورتی که تابع قبل از اتمام مناقصه و ریست شدن آن فراخوانی شود، تراکنش ***revert*** شود

describe "reset_tender"

it "Fails if not Owner"

تست میکند در صورتی که اکانتی غیر از ***owner*** تابع را فراخوانی کند، تراکنش ***revert*** شود

it "Fails if tender not finished"

تست میکند در صورتی که تابع قبل از اتمام مناقصه فراخوانی شود، تراکنش ***revert*** شود

it "Should successfully transfer the balance of contract"

تست میکند با اجرای تابع، موجودی قرارداد به درستی به آدرس داده شده منتقل شود.

describe "approve_forced_reset"

it "Fails if not Owner"

تست میکند در صورتی که اکانتی غیر از ***owner*** تابع را فراخوانی کند، تراکنش ***revert*** شود

it "Should successfully approve the forced reset"

تست میکند که تایید ***forced reset*** به درستی انجام شود

describe "cancel_forced_reset"

it "Fails if not Owner"

تست میکند در صورتی که اکانتی غیر از ***owner*** تابع را فراخوانی کند، تراکنش ***revert*** شود

it "Should successfully cancel the forced reset"

تست میکند که کنسل کردن تایید قبلی ***forced reset*** به درستی انجام شود

describe "tender_forced_reset"

it "Fails if not Owner"

تست میکند در صورتی که اکانتی غیر از ***owner*** تابع را فراخوانی کند، تراکنش ***revert*** شود

it "Fails if forced reset not approved before"

تست میکند که اگر قبلا تایید ***forced reset*** انجام نشده باشد، تراکنش ***revert*** شود

it "Should successfully forced reset the tender"

تست میکند با تایید ***forced reset***، تراکنش به درستی انجام و قرارداد ریست شود

it "Should successfully refund the collateral to participants during the forced reset"

بررسی میکند که با انجام ***forced reset***، موجودی ولتهای شرکت کننده به درستی بازگشت داده شود

describe "send_proposal_hash"

it "Fails if not in the 'sending rehashed proposal' stage"

تست میکند که اگر در مرحله ارسال هش پروپوزال نباشیم، تراکنش ***revert*** شود

it "Should successfully register sent hashed proposal"

تست میکند که در صورت وجود شرایط ارسال هش پروپوزال به درستی انجام شود

it "Should successfully transfer collateral from participant's wallet to the contract"

بررسی میکند که با فراخوانی تابع، کسر کردن وثیقه از حساب شرکت کننده مناقصه به درستی انجام شود

it "Should prevent duplicating participant"

بررسی میکند که در صورت فراخوانی مجدد تابع توسط یک شرکت کننده، نام شرکت کننده در آرایه شرکت کنندگان تکرار نشود و صرفا هش پروپوزال جدید جایگزین قبلی شود

describe "send_proposal_reveal"

it "Fails if not in the 'sending revealed proposal' stage"

تست میکند که اگر در مرحله ارسال آدرس واقعی پروپوزال نباشیم، تراکنش ***revert*** شود

it "Fails if sent revealed proposal not matches with previously sent hash value"

تست میکند که اگر هش مقدار ارسال شده به تابع با مقداری که درمرحله قبل به تابع ***send_proposal_hash*** ارسال شده یکسان نباشد، تراکنش ***revert*** شود

it "Fails if sent revealed proposal registered before"

بررسی میکند که اگر مقدار ارسال شده به تابع قبلا هم ارسال و ذخیره شده است، تراکنش ***revert*** شود

it "Should successfully register sent revealed proposal"

بررسی میکند که در صورت وجود شرایط ارسال پروپوزال به درستی انجام شود

describe "approve_proposal"

it "Fails if not owner"

تست میکند در صورتی که اکانتی غیر از ***owner*** تابع را فراخوانی کند، تراکنش ***revert*** شود

it "Fails if address didn't send the proposal"

تست میکند در صورتی که اگر آدرس ارسال شده به تابع جزء شرکت کنندگان مناقصه و ارسال کنندگان پروپوزال نبوده، تراکنش ***revert*** شود

it "Should successfully approve a participant"

بررسی میکند که در صورت وجود شرایط تایید پروپوزال شرکت کننده به درستی انجام شود

it "Fails if an address approved twice"

تست میکند در صورتی که اگر آدرس ارسال شده به تابع قبلا تایید شده باشد، تراکنش ***revert*** شود

describe "reject_proposal"

it "Fails if not owner"

تست میکند در صورتی که اکانتی غیر از ***owner*** تابع را فراخوانی کند، تراکنش ***revert*** شود

it "Fails if address didn't send the proposal"

تست میکند در صورتی که اگر آدرس ارسال شده به تابع جزء شرکت کنندگان مناقصه و ارسال کنندگان پروپوزال نبوده، تراکنش ***revert*** شود

it "Should successfully reject a previously approved participant"

بررسی میکند که در صورت وجود شرایط ریجکت پروپوزال شرکت کننده به درستی انجام شود

describe "send_price_hash"

it "Fails if not in the 'sending hashed price' stage"

تست میکند که اگر در مرحله ارسال هش قیمت پیشنهادی نباشیم، تراکنش ***revert*** شود

it "Fails if address was not approved"

تست میکند که اگر ارسال کننده هش قیمت جزء تایید شدگان پروپوزال نباشد، تراکنش ***revert*** شود

it "Should successfully register hash of proposed price (hashed with a secret salt)"

بررسی میکند که در صورت وجود شرایط، دریافت و ذخیره قیمت پیشنهادی هش شده به درستی انجام شود

describe "send_price_reveal"

it "Fails if not in the 'sending revealed price' stage"

تست میکند که اگر در مرحله ارسال مقدار واقعی قیمت پیشنهادی نباشیم، تراکنش ***revert*** شود

it "Fails if address was not approved"

تست میکند که اگر ارسال کننده هش قیمت جزء تایید شدگان پروپوزال نباشد، تراکنش ***revert*** شود

it "Fails if hash of sent (price+salt) not matches with previously sent hash value"

تست میکند که اگر هش مقادیر ارسال شده به تابع ***(price+salt)*** با مقداری که درمرحله قبل به تابع ***send_price_hash*** ارسال شده یکسان نباشد، تراکنش ***revert*** شود

it "Should successfully register proposed price"

بررسی میکند که در صورت وجود شرایط، دریافت و ذخیره قیمت پیشنهادی به درستی انجام شود

describe "withdraw"

it "Fails if withdraw() function called by a wallet didn't participate in the tender "

تست میکند که اگر کسی که تابع ***withdraw*** را فراخوانی کرده، جزء شرکت کنندگان مناقصه نبوده و درواقع وثیقه ای نگذاشته، تراکنش ***revert*** شود

it "Should successfully withdraw the collateral and remove the participant at early stages"

بررسی میکند که در صورت وجود شرایط، حتی در مرحله ارسال هش پروپوزال، انصراف از مناقصه و دریافت وثیقه شرکت کننده به درستی انجام شود

describe "withdraw"

علت جدا کردن این دو تست تابع ***withdraw*** از دو تست قبلی این است که دو تست قبلی در مراحل اولیه مناقصه انجام می شود ولی برای دو تست بعدی لازم است چند ولت تمام مراحل مناقصه را انجام دهند (در ***beforeEach*** این قسمت انجام می شود)

it "Fails if the winner of the tender wants to withdraw"

تست میکند که اگر کسی که تابع ***withdraw*** را فراخوانی کرده، برنده مناقصه بوده است، تراکنش ***revert*** شود

it "Should successfully withdraw the collateral and remove the participant at withdrawal stage"

بررسی میکند که در پایان مناقصه در صورت وجود شرایط، دریافت وثیقه شرکت کنندگانی که برنده نشده اند به درستی انجام شود

describe "get_participant"

it "Should successfully return the wallet address of desired participant"

بررسی می کند که آدرس ولت شرکت کننده خواسته شده به درستی توسط این تابع برگردانده شود

describe "get_proposal"

it "Should successfully return the wallet address of desired participant"

بررسی می کند که پروپوزال شرکت کننده خواسته شده به درستی توسط این تابع برگردانده شود

describe "is_approved"

it "Should successfully return the state of approve for different participants"

بررسی می کند که وضعیت تایید شرکت کننده خواسته شده در مرحله بررسی پروپوزالها به درستی توسط این تابع برگردانده شود

describe "getWinner & getBestPrice"

it "Should successfully return the winner of tender and the lowest proposed price"

بررسی می کند که آدرس ولت برنده مناقصه و حداقل قیمت پیشنهاد شده توسط برنده به درستی توسط توابع *getWinner* و *getBestPrice* برگردانده شود

describe "getStage"

it "Should correctly detects the stage of tender in the different priods of time"

بررسی میکند که مرحله فعلی مناقصه در بازه های زمانی مختلف به درستی توسط این تابع برگردانده شود (عدد خروجی تابع *getStage* کمی با مراحل تعریف شده در این داکیومنت تفاوت دارد ولی بصورت کامنت در سورس کد قرارداد معرفی شده است)