International Journal of Engineering & Technology, 7 (3.34) (2018) 885-889

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Distributed Human Interaction Proof for Public Blockchain Platform

. **Mahesh Kumar [(1)], Sindhu M. [(2)], M. Sethumadhavan[(3)], Lakshmy K.V.[(4)],**

[1] *TIFAC CORE in Cyber Security,*
Amrita School of Engineering,
Amrita Vishwa Vidyapeetham,Coimbatore,
India

Abstract

We introduce the concept Distributed Human Interaction Proof also known as DHIP. DHIP is an adaptation ofCAPTCHAtechnology to the public Blockchain paradigm. CAPTCHAs are helpful to distinguish between humans and bots. All the centralized web applications use CAPTCHA technology to prevent bot attacks. DAPPs like Reputation systems, voting systems are prone to bot based ballot stuffing. Centralized CAPTCHA solutions cannot be applicable to DAPPs. In this paper, we have proposed ways to incorporate the CAPTCHA technology to the public Blockchain paradigm. Our DHIP utility offers CAPTCHA benefits to DAPPs in trustless and distributed way. Formulations of the classes and workflows are discussed. We separated out CAPTCHA provisioning from its validation. CAPTCHA provisioning is handled off the chain but its validation is openly done in a trust less way by our DHIP utility. We have developed the transaction flow for CAPTCHA provisioning and its validation. We have demonstrated our utility with the help of Ethereum platform.

Keywords: *DHIP; CAPTCHA; Blockchain; smart contract; DAPP*

1. Introduction

Blockchain has gained quite popularity in recent days. It is praised for its record immutability. Anydata that gotsettled in the ledger remains forever. It provides the cryptographic proof for alter resistant [11]. The distributed append only ledger keeps growing but it preserves single consistent state across all theparticipating nodes. The state of the ledger is achieved through consensus protocol. Consensus protocol forms the core part of Blockchain. There are many ways to achieve consensus in Blockchain [3, 1, 7] and they have been comparatively studied [9, 6]. We focus on protecting the integrity of the system from bot attacks. We have developed specification that abstracted away from underlying platform and its consensus algorithms.

Smart contracts take Blockchain technology to the next level. According to Christopher et al. [8], smart contracts are automatable and enforceable agreements. Once smart contracts are instantiated, it facilitates the transfer of value exchange in the shared ledger. Unlike storing data in ledger, it is an innovative approach that deploys executable code in the distributed ledger. The state of the contract is manipulated by calling its functions and state changes are recorded in the underlying ledger. In normal web applications, both the code and its state is invisible and resides behind theboundary of the organization that provides theservice. In contrast to that, DAPPs exposes the code and its state to theworld. This change in paradigm unleashes huge opportunity to the world.

To explain the necessity of our utility, we discuss the bot attacks in reputation system. Reputation is the measure of goodness and trust in a particular entity. It is the collective trust score obtained from the mass. It reflects the opinion of every single rater. The knowledge obtained from the reputation system could guide us to decide and compare among the choices. It is vital and can influence our choices. A well-crafted bot has the potential to shake the integrity of such systems and thus makes it more trustless.

Reputation system that is developed as DAPP could achieve decentralization as well as alter resistant benefit. Sincethe rating data is stored in public tamper proof database, it provides the proof that history is fixed and unalterable.Anybody with Internet connection could audit the rating history of a ratee. Blockchain provides a way to achievereputation system in a trust less way.

Ballot stuffing is a serious threat to integrity of the online based reputation system as well as e-voting system.Bots escalate the threat to massive scale. They can create millions of digital identities in a minute unless the system isprotected by CAPTCHAs. These instantaneously created identities used by bots to boast the reputation of a targetedentity. The targeted entity may be an individual or company. Centralized reputation systems adopt CAPTCHAs toprevent bot based ballot stuffing. The commonly followed solution for bot attacks are CAPTCHAs. It has been usefulin lot of practical security applications ever since the term CAPTCHA was coined in 2003 [12]. CAPTCHAs arebeing used in preventing the bot attacks. Those attacks include email spams, automatic account signups and DDOSattacks against online resources. It also prevents indexing of website resources by search engine bots. It also saves theintegrity of online voting system and reputation system from ballot stuffing by bots. The use cases of CAPTCHAs arenot limited to the above mentioned but applies to broad idea of using unsolved AI (Artificial Intelligence) problemsfor security purposes. In general, any use case that requires user to be tested for bot can make use of the CAPTCHAs.

Alex et al [10] developed decentralized reputation system for e-commerce web applications that mainly focus onpreserving the privacy of raters by anonymous rating. It addresses the bot based

white washing by billing ratee forevery rating they received. But still rich ratee can spend money to boast his rating via bots. DAPPs are (Web 3.0 Apps)are different than web 2.0 Apps. They are trust disrupted [4]. Like traditional CAPTCHA system, role of CAPTCHAprovisioning should not be handled by a single entity. We have designed a CAPTCHA system that goes hand in handwith the trust disrupted principal of DAPP.

We have developed a CAPTCHA system for DAPP platform. It gives Human Interaction Proof (HIP) to DAPPs.HIPs are found to be useful in DAPPs like voting, reputation system and many more. Any DAPP that requires humaninvolvement can use DHIP utility. Rate operation performed by the reputation system needs its user to be tested asHumans. The same baseline requirement goes for voting application. So, we have developed it as a separate utilitycalled DHIP.We have designed it to run over any public blockchain platform. It is loosely coupled with the underlyingplatform and its consensus mechanism.

After passing the CAPTCHA test, DHIP utility generates proof to the user. The user account is associated withthe proof. Thus, accounts are assured to be created with human involvement. DHIP proofs are timestamped and can be looked upon by any other DAPP. Since the HIPs are stored in Blockchain database, they are tamper resistant. The user invokes the DHIP utility and solves the challenge to get HIP. DAPPs can see DHIP attested user (users proof) and take necessary action.

2. Existingwork

To Prevent bot based ballot stuffing attacks, in non-blockchain world (centralized environment), the reputation system relies on CAPTCHA service. It can incorporate CAPTCHA service in two ways. It can either implement the CAPTCHA service itself or outsource it to third party. The latter one is commonly followed. At the time of account sign up, users are tested with the CAPTCHA. The user who solved the challenge can secure an account with the system. Further operations invoked from the account are believed to be done by human.

In the outsourced CAPTCHA service, third party CAPTCHA Provider (CP) takes the responsibility of CAPTCHAgeneration and validation of the user submitted CAPTCHA-texts. The client gets CAPTCHA image from the thirdparty CP and shows it on the screen. The user solves the CAPTCHA challenge and the client get the users solutionvalidated by the CP. The CP provides token back to the client in response. The client sends the token back to Reputationsystems server. The server can run token validation with the CP. The CP returns the proof that whether the claimeduser is human or bot. In this solution, since the responsibility of CAPTCHA provisioning and validation is outsourcedto the third party, the whole trust is placed on the third party CP. In the centralized environment, trust concentrationon the third party is followed practice.

3. Proposed Work

To have CAPTCHA model in the Blockchain environment, we have proposed three solutions. The first two solutionshave shortcomings. We have developed DHIP utility from the third solution which is a combination of first twosolutions. We have implemented the third solution.

3.1 CAPTCHA Service Implemented as a Smart Contract

Every state change in the contract is recorded in the blockchain and openly accessible to all. If captcha-text is generated by the contract, it is immediately reflected in the ledger. CAPTCHA-text should be kept as secret until it is solved. So, the CAPTCHA service should not be implemented as smart contract.

3.2 Fixed CP for CAPTCHA Provisioning and Validation by Smart Contract

This approach is a two step approach. They are CAPTCHA generation and validation. The CAPTCHA generation can be done by CP to secure captcha-text confidentiality until it is solved by the user. The validation of CAPTCHA involves checking user submitted captcha text with actual CAPTCHA text. Its validation can be handled by a smart contract. By placing validation part on the contract, we can leverage Blockchains inherent advantage to generate HIPs in a trustless way. The steps are briefly discussed below. The steps are briefly discussed below.

CAPTCHA Generation: The CAPTCHA generation involves generating random text embedded CAPTCHA image and serving to the user. The CP accepts CAPTCHA request from user, it delivers CAPTCHA to them. The communication between CP and user is off-chain interaction. The software client displays the CAPTCHA image**.**

CAPTCHA Validation: The validation of CAPTCHA-text follows the CAPTCHA generation step. After delivery of CAPTCHA imageto the user, CP choose random nonce and keep it as secret. CP takes hash of the combination (captcha-text, secretnonce) called as commitment and sends the commitment to the DHIP contract. The user rekeys the captcha-text fromCAPTCHA image and submits the same to the contract for validation. Once user submits the answer, the CP revealsthe nonce value. The contract reconstructs the commitment from the users answer and CPs nonce. It compares newlycomputed commitment against the shared commitment for validation of CAPTCHA. The positive validation resultsare stored as HIP proofs against the user account.But there is shortcoming to this approach. Fixing the CP leads totrust concentration. It goes against the policy of trust disruption by Blockchain.

3.3 Randomly Electing CP for Every Request

This is an extension to our second solution. The right to provision CAPTCHA should not be given to a singlenode. It should be distributed to all the nodes on a fair basis. Election contract randomly chooses CP. CP is the servernode which generates and delivers CAPTCHA image to requested users. CP should register with the election contractto participate in the CAPTCHA service. Election contract maintains the pool of registered CPs. It adds or removes CP from the registered pool upon request. The user should create virtual CAPTCHA Request object with our DHIP utility. The utility invokes Election contract to choose CP for the request. Election contract chooses CP randomly. The elected CP for the request is called as Chosen CAPTCHA Provider For the Request(CCPFR). The rules defining randomness is written in election contract.

The value propositions of the Election contract are as follows as.

1. The right to be chosen for CCPFR is open and distributed to all nodes.
2. The choice of next CCPFR should be unpredictable until the current block is mined. If the next CCPFR.
3. Previous choices made for CCPFR should not influence current decision. The CCPFR forevery request is independent of the rest.

4. Formulationand Corrections

4.1 Stakeholders

We have discussed two stakeholders involved in our approach as follows as.

User: User is the one who request for CAPTCHA and solves it.

CaptchaProvider(CP): CP provides CAPTCHA to the requested users. From collection of CPs, Election contract elects CP

(CCPFR)for every request. The CCPFR will serve the CAPTCHA request.

4.2 Classes

We have outlined classes which could help us to realize our approach. The objects of these classes are instantiated by the smart contract to achieve the intended functionality. The instantiated objects are stored in the underlying ledger.

1. **CP**

a. Identifier Blockchain Identifier of the CP
b. ipAddress_port IP Address and port number of Captcha Service Provider
c. countOfService number of CAPTCHA requests served by the CP
d. registryIndex index of the CAPTCHA Registry kept in the Election smart contract

2. **Cp Registry Pool**

a. Index index of the CP
b. Cp array of CP objects with the index.

3. **Captcha Request**

a. Requestor identifier of the requestor used in Blockchain platform
b. Status current status of request including created, assigned, cleared and failed.
c. Answer answer object which contains the captcha-text submitted by user
d. fees amount of fee paid to ccpfr

4. **Captcha Response**

a. commitment commitment made by CP (hash of secretNonce with captcha-text)
b. secretNonce revealed by CP after user submission of answer
c. givenBy refers to the identifier of the CP

5. **Answer**

a. captchaText captcha-text submitted by the user
b. provider identifier of the captcha provider
c. givenBy identifier of captcha requestor or user

6. **Decision**

a. Requestor identifier of the requestor used in Blockchain platform
b. Ccpfr randomly elected CP data object
c. Response captchaResponse object

4.2 Smart Contracts

The DHIP utility is made up of Election and DHIP contracts. These contracts run in parallel independently.

Election contract: Election contract deals with electing CP for the captcha request. All captcha requests are recorded as captchaRequestobjects in the ledger. DHIP contract invokes election contract to choose CP for every new captchaRequest object.This newly elected CP for captchaRequest is called as Chosen Captcha Provider for Request(CCPFR). The CCPFRalong with the requestor identifier will be stored in the decision object and published into the ledger. The user canlook up the decision object for his identifier to know the CP information. Once decision object is generated, the statusof request object will be changed to assign. For randomness in choosing, we have used hash of the previous block.

DHIP contract: This contract captures interactions around captcha provisioning and validation. The user request for captcha challengeto the CCPFR goes as an off-chain interaction. The CCPFR chooses nonce for every captcha request he serves.CCPFR will keep the nonce as secret until user submits the answer. After serving the users request with captchaImage,CCPFR takes the hash of answer along with secret nonce. This computed hash is called commitment and itwill be submitted to the DHIP contract. Later, CCPFR reveals the secret nonce to evaluate the users answer. Thisoperational routine for Captcha validation is written into DHIP contract.

5. Contract Operations

5.1 Operations Used For Electing CCPFR

The following DHIP contract methods are either directly or indirectly used by CP to conduct election protocol.
chooseCP()
This method will choose CP based on the previous block hash. This function is only called by DHIP
contracts makeRequest method.

5.2 Operations Used for CAPTCHA Provisioning

The following methods are used by the user. Out of the 7 methods described below, getCaptcha is an off-chainmethod but other methods are on-chain methods. On-chain methods are methods written in contract and refers tointeraction that persists in the ledger. In contrast to that, off-chain method does not go over the underlying Blockchainplatform. Here, getCaptcha method captures the interaction between the user and the CCPFR.
makeRequest():creates an instance of captchaRequest object and decision object for the requestor.
getDecision(): This on-chain method will return decision object for the corresponding callers identifier
getCaptcha(): The user requests the CCPFR for captcha-image. This is an off-chain interaction.
submitAnswer(captcha-text)
when user submits the captcha-text(answer), DHIP contract will create the answerobject.
sendCommitment(commitment, requestor): The CCPFR made its commitment to this contract method. Thismethod will create corresponding captchaResponse object and add the response to the decision object.
openCommitment(nonce, requestor): This will update the nonce of the corresponding captchaResponse Object.
evaluateCaptcha(requestor): This is the final step of Captcha validation process for requestor. This function is onlytriggered by DHIP contract once the commitment is opened by the CCPFR.

6. Transaction Flow

6.1 Transaction Flow for Election

An honest administrator can add new CP to the DHIP utility by calling Election contracts register method. Thenew CP registration requires IP address and port details of the service. The administrator can add new CPs intothe system but he could not control or determine systems output.
DHIP contract calls the chooseCP method to choose CCPFR for captchaRequest object.

6.2 Transaction Flow for CAPTCHA Provisioning and Validation

1. User creates CaptchaRequest object by calling makerequest() method of DHIP contract. It creates CaptchaRequestobject and status marked as created. DHIP will invoke chooseCP() method to instantiate correspondingdecision object.
2. User gets decision object by calling getDecision method. He extracts CCPFR from the decision object. Hemakes request to the CCPFR by calling getCaptcha() method.
3. CCPFR constructs captcha-image and returns the same to the requested user. Meanwhile, CCPFR takes thehash of answer(captcha-text) along with the secret nonce. This hash is called commitment. CP publish thecommitment by calling sendCommitment method. It changes the status of the corresponding captchaRequestobject to received.
4. User rekeys the captcha-text from the captcha image. He submits the answer by calling submitAnswer methodof DHIP contract. DHIP contract stores the answer in the corresponding CaptchaRequest object.
5. CCPFR opens the earlier commitment by revealing the secret nonce associated with the decision object. CP callsopenCommitment method of DHIP contract. This will trigger the evaluateCaptcha method for the correspondingrequestor. The evaluate operation takes the hash of user submitted captcha-text along with the revealednonce and compares it against the commitment of the captchaResponse object. It changes the status of thecaptchaRequest object as cleared or failure based on the evaluate operation result.

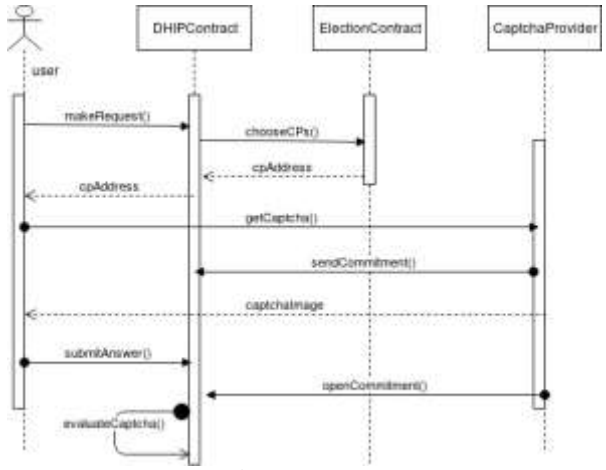

Figure 1: Transaction Flow for CAPTCHA Provisioning and Validation

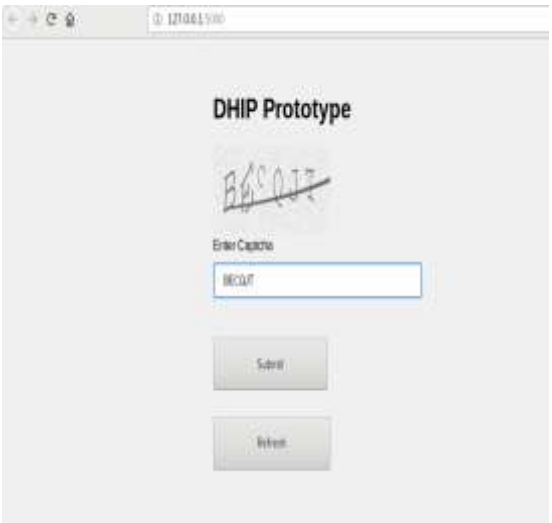

Figure 2: a) CAPTCHA from CP displayed on user page

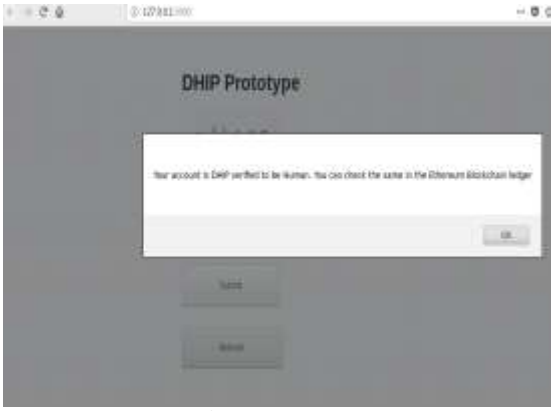

Fig3: b) After passing the CAPTCHA challenge

7. Incentivization and Probabilistic Attack

DHIP contract will deposit fees associated with captchaRequest objects to CCPFR at the end of captcha evaluationprocess. It is possible to get the bot controlled accounts as DHIP verified without even solving the captcha challenge ifthe attacker controls certain percentage of registered CPs in the system. Suppose, a malicious entity known as Malicewants to subvert the system. He creates bot accounts in the platform. To get these accounts recognized as DHIPverified, Malice compromises certain percentage of CP registered accounts to increase his chance to win the election.If Malice controlled CP is elected as CCPFR for the bot made captcha-request, he can help his bot account by sendingcaptcha plain text. This attack is probabilistic. Since the CCPFR changes for every subsequent request, Malice couldnot get bunch of bot accounts verified in a row. However, we have discussed our fourth solution in future work totackle this probabilistic attack.

8. Results

At the time of writing this paper, Ethereum has been widely recognized as public Blockchain platform[4]. So,we have chosen it for developing our prototype. The solidity language has been used for writing smart contracts [2].We have developed a simple web client to prove DHIP utility. The CCPFR is implemented as a python service withaccess to smart contract through web3.py APIs[5].We have added screenshots of the user signup webpage which getscaptcha-image from the randomly elected CCPFR and displays it. When the user submits answer, web client starts theCAPTCHA validation workflow on DHIP utility and shows the result of CAPTCHA validation back to the screen.

9. Future Work

We think of extending the third solution further to rigid the security and lessen the impact of probabilistic attack.Instead of single CP for every captcha request, we can provide multiple CPs for every CAPTCHA request. TheCCFPR is a list of randomly elected CPs. The web client constructs captcha image from multiple image pieces andeach individual piece obtained from separate CPs in the CCPFR. Attacker should control all CPs for the captcharequest to make the bot pass the CAPTCHA challenge. This approach makes even harder for the bot to subvert the system.

10. Conclusion

Bots have the potential to shake the integrity of any online service until it is protected by CAPTCHA. Our approachis the first step taken in the line of developing CAPTCHA systems to DAPPs. We

have defined a specification andimplemented the same in Ethereum platform to prove the approach. Our specification abstracted away from underlyingBlockchain implementation and its consensus protocol. In general, our utility can serve all the DAPPs which requirehuman involvement. However, we have not tested the performance of the utility. We have planned to do it in further work. The design of the utility is simple enough to adapt to any Blockchain paradigm.

References

[1] Ripple website. URL: https://ripple.com/.

[2] Solidity documentation. URL: http://solidity.readthedocs.io/en/v0.4.21/.

[3] Tendermint website. URL: https://tendermint.com/.

[4] Web 3: A platform for decentralized apps. URL: http://ethdocs.org/en/latest/introduction/web3.html.

[5] Web3 python implementation. URL: https://web3py.readthedocs.io/en/stable/.

[6] Ambili, K., Sindhu, M., Sethumadhavan, M., 2017. On federated and proof of validation based consensus algorithms in blockchain, in: IOP Conference Series: Materials Science and Engineering, IOP Publishing. p. 012198.

[7] Cachin, C., 2016. Architecture of the hyperledger blockchain fabric, in: Workshop on Distributed Cryptocurrencies and Consensus Ledgers.

[8] Clack, C.D., Bakshi, V.A., Braine, L., 2016. Smart contract templates: foundations, design landscape and research directions. arXiv preprintarXiv:1608.00771 .

[9] Sankar, L.S., Sindhu, M., Sethumadhavan, M., 2017. Survey of consensus protocols on blockchain applications, in: Advanced Computing andCommunication Systems (ICACCS), 2017 4th International Conference on, IEEE. pp. 1–5.

[10] Schaub, A., Bazin, R., Hasan, O., Brunie, L., 2016. A trustless privacy-preserving reputation system, in: IFIP International Information Securityand Privacy Conference, Springer. pp. 398–411.

[11] Swan, M., 2015. Blockchain: Blueprint for a new economy. " O'Reilly Media, Inc.".

[12] Von Ahn, L., Blum, M., Hopper, N.J., Langford, J., 2003. Captcha: Using hard ai problems for security, in: International Conference on theTheory and Applications of Cryptographic Techniques, Springer. pp. 294–311.