Decentralised domain authentication

Shehyaaz Khan Nayazi*,
Mohammed Sufiyan Aman, Shakshi Pandey,
Riyanchhi Agrawal and M. Namratha

BMS College of Engineering,
Bull Temple Rd., Basavanagudi, Bengaluru,
Karnataka 560019, India
Email: shehyaaz.cs17@bmsce.ac.in
Email: mohammedsufiyan.cs17@bmsce.ac.in
Email: shakshi.cs17@bmsce.ac.in
Email: riyanchhi.cs17@bmsce.ac.in
Email: namratham.cse@bmsce.ac.in
*Corresponding author

Abstract: Transport layer security protocols ensure secure and encrypted communications on the internet. The security of TLS protocols relies on properly validating digital certificates containing the public keys of domains during the handshake authentication. These certificates are issued by certificate authorities (CAs). However, mis-issued certificates have been used to attack users. To monitor CA behaviour, certificate transparency (CT) and a decentralised system, instant karma public key infrastructure (IKP) were proposed, however, they do not tackle domain misbehaviour. Based on CT and IKP, a decentralised system is proposed using the Ethereum framework to handle the misbehaviour of domains and enhance the security of TLS. The proposed system utilises CT logs to detect and respond to domain misbehaviour.

Keywords: certificate transparency; online certificate transparency protocol; OCSP; signed certificate timestamp; SCT; blockchain; smart contract; transport layer security; instant karma PKI; IKP; decentralised domain authentication.

Reference to this paper should be made as follows: Nayazi, S.K., Aman, M.S., Pandey, S., Agrawal, R. and Namratha, M. (2022) 'Decentralised domain authentication', *Int. J. Blockchains and Cryptocurrencies*, Vol. 3, No. 1, pp.24–40.

Biographical notes: Shehyaaz Khan Nayazi is working as a Software Engineer at Cisco Systems (India) Pvt. Ltd. He has received his BE degree in the Department of Computer Science and Engineering from BMS College of Engineering in 2021. He is currently working on storage systems, Dockers & Kubernetes and automation projects. He has organised and attended various workshops spanning fields such as web and mobile app development, data science and blockchain and has also worked on various projects in these domains during his academic years.

Mohammed Sufiyan Aman is working as an Embedded Engineer at Honeywell Technology Solutions Lab Pvt. Ltd. He has received his BE degree in the Department of Computer Science and Engineering from BMS College of Engineering in 2021. He is currently working on flight management engines and systems acting as a core member of Flight Systems and Controls Centre of

Excellence. During his academic years, he has worked in various projects including database management systems, mobile app development and augmented and virtual reality.

Shakshi Pandey is working as a Software Engineer in the Aerospace Field at Honeywell Technology Solutions Lab Pvt. Ltd. She has received her BE degree in the Department of Computer Science and Engineering from BMS College of Engineering in 2021. During her academic years, she has been one of the student coordinators for a technical symposium called Phaseshift, where she has managed and attended various workshops like UI/UX, blockchain and has also worked in the areas of frontend development, database, cloud services and so on.

Riyanchhi Agrawal is working as an Associate Technical Consultant at BlueYonder Group Inc. She has received her BE degree in the Department of Computer Science and Engineering from BMS College of Engineering in 2021. She has worked in areas like blockchain, AI/ML and so on. She is currently functioning as a core member in the JDA cloud services and has also performed various tasks on environment provisioning and configuration in BY cloud/Azure. She has also worked in various projects in the fields of web and mobile app development.

M. Namratha is working as an Assistant Professor in the Department of Computer Science and Engineering, BMS College of Engineering from 2010. She is currently pursuing her PhD in the area of Blockchain at NIT Tiruchirappalli. She received her BE in Computer Science and Engineering in 2010 (5th Rank–VTU) and MTech in Software Engineering (Gold Medalist – VTU) in 2013. Her current research interests include blockchain, cryptology, image processing. She has published 16 papers in international journals and nine conference papers in her domain. She has taught subjects such as C Programming, microprocessors, and many more. She is the recipient of the 'Best Scientist National Award' at the 2nd International Conference on ERIR-2020 on 12th December 2020.

This paper is a revised and expanded version of a paper entitled 'Decentralized domain authentication – exploratory literature survey' presented at International Conference on Computing Methodologies and Communication (ICCMC), Surya Engineering College, Erode, Tamil Nadu – 638107, India, 8–10 April 2021.

1 Introduction

Data communication security on the Internet is ensured by TLS protocols. These protocols are widely adopted and involve the verification of digital certificates received by TLS clients from domains using the public keys provided by certificate authorities (CAs) (Namratha et al., 2021). However, the reliance on CAs to verify domain certificates validates the rogue certificates issued by them as well. Such certificates can be used to attack unsuspecting clients and there is no mechanism to warn them against such attacks (Namratha et al., 2021). Incidents of CAs being compromised have been reported in the past, resulting in domains receiving invalid certificates. Also, there have been incidents of CAs having mis-issued certificates by mistake (Namratha et al., 2021). Thus, the fragile CA system is a vulnerability of TLS (Namratha et al., 2021).

CT was proposed to facilitate the quick identification of rogue certificates. CT created a framework to make the web public key infrastructure (PKI) transparent by providing public logs to monitor and audit digital certificates issued by CAs. However, CT suffered some limitations due to its centralised nature. To mitigate these short-comings, instant karma PKI (IKP) was proposed which provided a decentralised, blockchain-based solution to detect CA misbehaviour (Namratha et al., 2021).

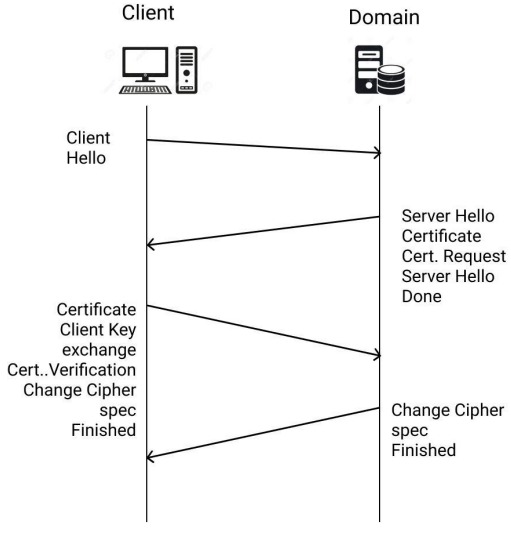
CT and IKP focus primarily on detecting and responding to CA misbehaviour, however, domain misbehaviour in using digital certificates is not considered (Namratha et al., 2021). To monitor domain behaviour, a decentralised, blockchain-based system is proposed. Such a system could use CT to detect domain misbehaviour and enhance the security of TLS (Namratha et al., 2021).

In this paper, we make the following contributions:

- We propose decentralised domain authentication (DDA), a decentralised system based on CT and IKP to handle domain misbehaviour.
- We discuss the system design and various entities involved in the system.
- We implement a system prototype using Ethereum blockchain, utilising CT logs and online certificate transparency protocol (OCSP) (Santesson et al., 2013) to validate digital certificates.
- We analyse the feasibility of the prototype in terms of gas fee charged for each operation involved in the system.

The paper is structured as follows: Section 2 discusses certain key terminologies, explores the problem statement and existing solutions. Section 3 discusses the proposed solution, including the evaluation methodology, key components, high level design and prototype implementation. Section 4 discusses the implementation results. Section 5 presents a summary of the paper, whereas Section 6 discusses future work, including possible system enhancements.

Figure 1 TLS 1.2 handshake



2 Literature survey

2.1 Key terminologies

1. TLS handshake authentication: This initiates the TLS connection between client and domain by establishing a cipher suite for the session and verifying the certificate provided by the domain (Bhargavan et al., 2014) (Namratha et al., 2021). Figure 1 illustrates the steps involved in the handshake (Cloudflare) (Namratha et al., 2021).

2 Smart contract: Smart contracts are user-defined, self-executing programs stored on the blockchain that automatically enforce consensus and also automate the execution of agreements in the blockchain network (Delmolino et al, 2016; Namratha et al., 2021). A blockchain is an append-only linked list-like data structure consisting of a chain of individual records called blocks (Nakamoto, 2008; Namratha et al., 2021).

3 Ethereum: Ethereum is one of the most popular blockchain frameworks. It is a decentralised, open-source, global platform for decentralised applications (dApps) with provisions for stateful smart contracts to be built and run, and uses a cryptocurrency known as ether (Namratha et al., 2021). It can be used to create business, financial and entertainment dApps. Values can be persisted in the blockchain through the smart contracts and can be used in various invocations (Ethereum) (Namratha et al., 2021).

2.2 Existing systems

A combination of symmetric and asymmetric key cryptography is used by TLS protocols to establish cipher suites and ensure a secure and encrypted channel on the internet (Rescorla, 2018; Namratha et al., 2021). Clients and domains establish a secure session using asymmetric key cryptography, and symmetric key cryptography is used to exchange data in the secure session (Namratha et al., 2021). To establish the secure session, TLS clients receive digital SSL certificates when a secure, encrypted connection to a domain is requested (Namratha et al., 2021). SSL certificates contain the information about the domain, including its public key and are signed using the private keys of CAs. The CA signature is verified using its public key which is embedded in the client browser's key store (Namratha et al., 2021). Since the signature verification is done using the public key of the CA, a mis-issued certificate is technically valid and trusted.

Several cases of CA failures, misbehaviours and man-in-the-middle (MITM) attacks have been reported in the past. These include the Comodo Fraud Incident (Comodo, 2011), the Verisign Microsoft incident (Microsoft Security Bulletin, 2001), the Google ANSSI incident (Langley, 2013), the DigiNotar breach (Hoogstraaten et al, 2012), the MITM attack targeting popular websites such Google, Yahoo and Skype (Mills and McCullagh, 2011), and the Spanish CA Camerfirma incidents (Mozilla, 2021).

As stated in Section 1, CT was proposed to make the web PKI system transparent. It provided an open framework to log, monitor and audit digital SSL certificates publicly, facilitating the detection of rogue certificates (Laurie et al., 2013; Namratha et al., 2021). Laurie et al. (2013) defines a cryptographic component called Merkle hash tree used in CT and defines Merkle audit paths and Merkle consistency proofs which can be used to verify efficiently that a certificate has been logged. Laurie et al. (2013) also specifies the CT log format and API endpoints to add certificates to a log, retrieve consistency and

audit proofs from the log and retrieve log entries. While Laurie et al. (2013) defines the first version of CT, Laurie et al. (2021) proposes a second version of CT and specifies that CT logs must support either version 1 or version 2 as the versions are incompatible. Google's CT is a very popular and widely deployed log-based PKI and is supported by the major browsers such as Firefox and Chrome (Namratha et al., 2021).

Revocation Transparency (RT) was a supplementary system that was proposed to enhance CT (Laurie and Kasper, 2012; Namratha et al., 2021). This system provides an efficient mechanism to view a list of revoked certificates and examine the revocation of certificates (Namratha et al., 2021). The revocation status of a certificate can also be determined using OCSP, which stands for online certificate status protocol (Santesson et al., 2013). Using OCSP, the client can send an OCSP request to an OCSP responder. The OCSP response contains the revocation status of the certificate, and can be one of 'good', 'revoked' or 'unknown'.

As noted in Namratha et al. (2021) CT had the following limitations (Matsumoto and Reischuk, 2017):

1 CT requires a centralised and consistent source of information for secure operation, however, there is no protocol to periodically synchronise the logs securely (Namratha et al., 2021). Google's CT has specified a list of authorised logs (Certificate Transparency known logs, 2016a), and they must conform to certain operational standards, as noted in (Certificate Transparency log policy, 2016b).

2 CT does not provide sufficient incentivisation for monitoring CA behaviour, due to a greater cost of operating a log than the benefits for the log operator (Google: Certificate Transparency: Feb 2014 survey responses, 2014; Namratha et al., 2021). Also, no compensation is provided to those who discover unauthorised digital certificates (Namratha et al., 2021).

3 Though CT provides an efficient mechanism to monitor CA behaviour, responses to misbehaviour are delayed and not automatic (Namratha et al., 2021).

As stated in Namratha et al. (2021), an alternative approach called DANE or DNS-based Authentication of Named Entities was proposed to eliminate or replace CAs from the TLS ecosystem (Hoffman and Schlyter, 2012). This approach uses DNSSEC or domain name system security extensions (Arends et al, 2005) to store the public keys of domains. However, CAs could not be precluded by this approach (Namratha et al., 2021).

IKP is a decentralised, Ethereum blockchain-based system that was proposed to mitigate the defects of CT (Matsumoto and Reischuk, 2017). It is a system that not only provides automated responses to CA misbehaviour but also incentivises the monitoring of CA behaviour (Namratha et al., 2021). IKP's architecture proposes entities called reaction policies (RPs) and domain check policies (DCPs). These policies specify the CA's reaction to a detected misbehaviour, which is typically a compensation paid to the entity that detects the misbehaviour, and the domain's criteria to computationally determine the validity of a domains' certificate, respectively (Namratha et al., 2021). The system also includes entities known as detectors that report unauthorised certificates and monitor CA behaviour.

Despite the enhancements provided by CT and IKP, CT and IKP do not tackle the case of misbehaving domains (Namratha et al., 2021). Also, these systems do not propose any solution to mitigate denial-of-service (DoS) attacks (Xia et al., 2017; Namratha et al., 2021). Domain misbehaviour takes place when a domain deliberately does not use

certificates in the CT log for its own benefits, or when a domain deceives clients by fraudulently impersonating a genuine one, which could lead to a phishing attack (Namratha et al., 2021).

Xia et al. (2017) propose a decentralised, Ethereum blockchain-based system called enhanced TLS domain authentication (ETDA) that 'supplements CT and IKP and responds to domain misbehaviour by enforcing automatic punishments, in addition to compensating the client for such a misbehaviour during TLS handshake authentication' (Xia et al., 2017; Namratha et al., 2021). Xia et al. (2017) also provide a proof for the economic feasibility of the incentivisation mechanism using game theoretical models (Weibull, 1995) based on the Nash Equilibrium [Osborne et al., (1994), p.14]; Namratha et al., 2021).

3 Proposed solution

3.1 Evaluation methodology

As noted in Namratha et al. (2021), the evaluation of a proposed solution satisfying the requirements stated in Namratha et al. (2021) can be performed as follows:

1 The solution must define a deterministic, unambiguous criteria for domain misbehaviour.

2 The solution must provide a computationally efficient decentralised mechanism to detect the misbehaviour.

3 The solution must automate the responses to detected misbehaviour, and the responses must be immediate.

4 The solution must sufficiently incentivise those who monitor domain behaviour and punish the misbehaving domains.

5 The solution must avoid and prevent DoS attacks in all circumstances.

3.2 Key components

The following are the key components of the proposed system (Xia et al., 2017; Namratha et al., 2021):

1 Domains: Domains participate in TLS handshake with clients and issue domain reaction policies (DRPs) that specify the domain's reaction to the event of sending an invalid certificate to clients and breaking the clients' trusts. They are required to register DRPs and identity information to participate in the system (Namratha et al., 2021).

2 Clients: Clients receive digital certificates from domains during TLS handshake and issue client check policies (CCPs), that specify the criteria for a valid certificate. Clients can monitor domain behaviour by purchasing the domain's DRP. Clients register CCPs in Ethereum blockchain to participate in the system (Namratha et al., 2021).

3 Smart contract: The smart contract provides different functions that perform the operations of each entity participating in the system. It also maintains a contract fund to escrow funds and provide rewards. The contract is accessible to all registered entities of the system (Namratha et al., 2021). The incentivisation mechanism of the system is implemented through the smart contract.

4 Ethereum blockchain: It is a 'decentralised platform providing a transaction framework and incentive nature to participants' (Namratha et al., 2021). It is a distributed ledger that can be validated by various nodes participating in the network.

3.3 Domain reaction policy

Figure 2 shows the proposed design of a DRP (Xia et al., 2017). The system requires that a domain must issue a DRP when registering in the system (Namratha et al., 2021). The various fields in a DRP are explained in Namratha et al. (2021).

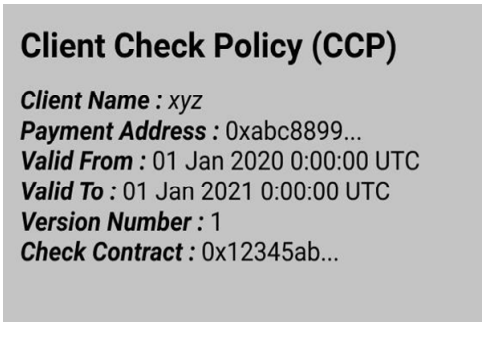
Reaction contract is a smart contract address that contains the implementation of the domain's reaction to its misbehaviour.

Figure 2 Structure of DRP

3.4 Client check policy

A client must register in the system by issuing a CCP (Namratha et al., 2021), whose proposed design is shown in Figure 3 (Xia et al., 2017).

Figure 3 Structure of CCP



The various fields in a CCP are explained in Namratha et al. (2021) Check contract is the smart contract address that checks a digital certificate's inclusion in CT logs, validity and revocation status. Through the Check contract, the client can establish the criteria for domain misbehaviour.

3.5 Operations

As explained in Namratha et al. (2021), the following are the fundamental operations provided by the system (Xia et al., 2017):

1 Client and domain registration: The system provides the necessary interfaces and operations for clients and domains to register CCPs and DRPs respectively in the blockchain using the smart contract (Namratha et al., 2021).

2 Purchasing DRP: To monitor domain behaviour, a client must purchase the domain's DRP. The purchase fee is transferred to the corresponding domain (Namratha et al., 2021).

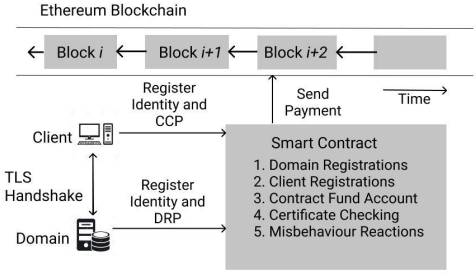
3 Certificate check: The client uses the check contract in its CCP to validate a domain's certificate (Namratha et al., 2021).

4 Misbehaviour reaction: In the event of detecting domain misbehaviour by a check contract, the Reaction contract in the domain's DRP is executed, which causes a series of transactions to occur (Namratha et al., 2021). The transactions and payments allow the system to implement its incentive mechanism, making the solution economically viable and feasible. These are explained in Section 4-G.

3.6 High level design

An overview of the high-level architecture of the system is given in Figure 4 (Namratha et al., 2021):

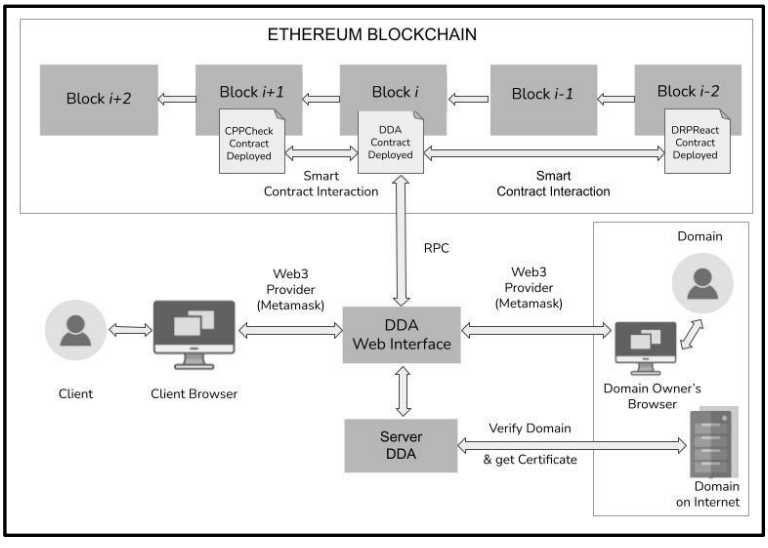
Figure 4 High level design of DDA



As explained in Namratha et al. (2021), the smart contract provides functions for client and domain registration. Client registration involves CCP registration, whereas domain registration involves DRP registration. The client and the domain participate in a TLS Handshake authentication to establish a secure connection, which includes the domain sending its digital certificate to the client. The smart contract also provides functions to purchase DRPs, check domain certificates, and trigger the misbehaviour reaction.

The blocks in Figure 4 represent the blocks in the Ethereum Blockchain network. These blocks store all the transactions in the Ethereum Blockchain network, providing a distributed ledger to all nodes in the network (Namratha et al., 2021).

Figure 5 System architecture of DDA



3.7 Design

The design of the proposed system is illustrated in Figure 5. The system consists of two entities – a client and a domain. Both the entities interact with the system through a web interface, which provides dashboards to register CCP and DRP, update them, purchase DRP, check certificate, check CCP and DRP status, and view account information. The system provides separate dashboards for the client and domain. The clients and domains are required to have a cryptocurrency wallet to interact with the system. Although there are many cryptocurrency wallets available, MetaMask has been used for the system prototype. The front-end of the web interface interacts with a back-end server to verify whether a domain can be accessed on the Internet and to fetch a domain's certificate and interacts with the deployed contracts on Ethereum blockchain through remote procedure calls or RPCs. Figure 5 also shows the interactions between the deployed contracts in the Ethereum blockchain. The system's web-interface interacts with the deployed DDA contract, which in turn interacts with the Check contract in the Client's CCP, and the react contract in the domain's DRP.

To realise the incentivisation mechanism of the system, various parameters and payments were defined (Xia et al., 2017):

1 Escrow parameter (α): determines the amount of ether escrowed by the system when a DRP is purchased by a client.

2 Termination parameter (δ): the minimum amount of ether a client receives for its lost trust in a domain.

3 Termination payment (t): the fund split between a client and a domain, when the client terminates the domain's DRP before its validity expires.

4 Internal misbehaviour payment (m): the fund received by a client when it detects an invalid certificate. This compensates the client for any security risks that it would suffer during TLS.

5 Contract fund payment (f): the fund paid to the DDA contract. This replenishes the contract fund and ensures that it always has sufficient funds to continue its operations. client terminates the domain's DRP before its validity expires.

Table 2 summarises the rewards and punishments for the various entities in the system in the domain behaviour and misbehaviour scenarios. Table 3 provides an overview of the various events and their corresponding transactions that occur in the system (Xia et al., 2017). It also provides the amounts that are exchanged in these transactions.

Table 1 List of parameters and their values

Parameters	*Values*
Escrow parameter (α)	0.3
Termination parameter (δ)	0.3
Internal misbehaviour (m)	0.9
Contract fund payment (f)	0.3
Termination payment (t)	0.4
Client registration fee (rC)	0.001 ether
Domain registration fee (rD)	0.01 ether
(Client and domain) update fee (u)	0.001 ether

Note: The parameters are multiplied by the DRP price (p) to get the value in ethers.

$$Total\ funds,\ S = m + t + f \tag{1}$$

$$Client\ Termination\ Payment,\ t_C = \delta + \theta * (t - \delta) \tag{2}$$

where θ is the proportion of termination (0 <= θ <= 1) calculated using the DRP valid from and valid to dates

$$Constraint\ 1:\ m + \delta > 1 \tag{3}$$

$$Constraint\ 2:\ (\alpha * S) > t \tag{4}$$

Table 2 Rewards in domain behaviour and misbehaviour scenarios

Events	*Entities*		
	Client	*Domain*	*DDA fund*
Domain behaves	–p	p	0
Domain misbehaves	(-1 + m + tC)*p	(1 – m – tC – f)*p	f

Note: p is the DRP price.

The user must also pay a registration fee when registering in the system, and an update fee to update the registered details. These parameters are used in calculating the total

funds (S) and the termination payment for the client (t_C) and must satisfy certain constraints to ensure rewards to the client and punishment to the misbehaving domains. Table 1 provides the list of parameters and their corresponding values used in the implementation of the system prototype.

Table 3 Transactions in various events in the system

Event	*From*	*To*	*Amount*
Register client	Client	CF	rC
Register domain	Domain	CF	rD + (m+f)*p
	CF	DRP	(m+f)*p
Purchase DRP	Client	CF	p
	CF	Domain	(1 - α*S)*p
Detect internal misbehaviour	DRP	CF	(m+f)*p
	CF	Client	(m+tC)*p
			α*S = α*S - tC
Expire DRP			
(Domain misbehaviour detected)	CF	Domain	α*S*p
Expire DRP			
(Domain misbehaviour not detected)	DRP	CF	(m+f)*p
	CF	Domain	(α*S)*p +
			(m+ f)*p

Notes: CF is the DDA contract fund and DRP is the registered Domain's reaction policy.
p is the DRP price.

The domain dashboard allows the user to register a DRP, update the details of a registered DRP, check the DRP status, view the amount escrowed by the system, and also expire the DRP. To register a DRP, the user enters a valid domain name, the name of the DRP issuer, the user's public wallet address, the DRP version, the DRP valid from and valid to dates, the DRP React contract address, and the price of the DRP in ether. The domain is verified by calling the verify API of the back-end server, and if valid, the details are sent to the deployed DDA contract. As shown in Table 3, the user pays rD + (m+f)*p ether when registering a DRP, and the DDA contract sends (m+f)*p ether to the DRP React contract. This ensures that the React contract has sufficient balance to pay the DDA contract when its trigger() method is called by a client. The user can also update the DRP issuer name and the valid to date through the dashboard.

A domain can check its DRP status through the dashboard. The DRP status can either be valid or terminated or validity expired, or both terminated and validity expired. As shown in Table 3, (1 – α*S)*p ether is escrowed by the DDA contract when a client purchases the domain's DRP. A domain can also view the total escrowed amount through the dashboard. Once the DRP validity has expired, the domain can expire its DRP to claim the escrowed amount and also the amount deposited in the DRP React contract, given it was not triggered. The web interface also provides a screen to view the domain details in brief.

The client dashboard allows the user to register a CCP, update the details of a registered CCP, check the CCP status, purchase DRPs, view the details of purchased DRPs and check a domain's certificate. To register a CCP, the user enters a client name,

the public wallet address, the CCP version, the CCP valid from and valid to dates and the CCP Check contract address. As shown in Table 3, the user sends rC ether when registering a CCP. The user can also update the CCP valid to date through the dashboard.

A client can check its CCP status through the dashboard. The CCP status can either be valid or validity expired. A client can purchase DRPs through the interface as well. Only those DRPs whose validity has not expired and React contract not triggered can be purchased. The client sends p ether to purchase a DRP, p being the DRP price. The dashboard also displays the details of purchased DRPs, that include the domain name, the DRP valid from and valid to dates, the DRP price, the date when this domain's certificate was last checked, and an option to check this domain's certificate. The web interface also provides a screen to view the client details in brief.

When a client checks a domain's certificate through the dashboard, the getsct API of the back-end server is called. The server then establishes a TLS connection with the domain and parses the domain's certificate received in the TLS handshake to extract the signed certificate timestamp (SCT) details (Laurie, Langley and Kasper, 2013), (How CT fits into the wider web PKI ecosystem). The SCT details are extracted using the OID (OID Registry) for SCT. The server also sends an OCSP (Santesson et al, 2013) request and parses the OCSP response to get the revocation status of the certificate. The API then returns these details to the web interface, which are then sent to the DDA contract. Further, the DDA contract sends these details to the CCP's Check contract which determines if the certificate is present in the CT logs or not and if it is revoked. If the certificate is not logged in the CT logs, or is revoked, the domain is deemed to have misbehaved and the trigger() method of the DRP's React contract is called. The transactions that occur in this scenario are shown in Table 3. If the Check contract finds that the certificate is valid, no transactions occur and the DDA contract updates the last checked field in the purchased DRP details.

The DDA contract maintains the state of the blockchain through mappings of client and domain details. It also maintains an array of registered domain names, allowing the contract to reject the registration of duplicate domains. The client details include the CCP details, the wallet addresses of the domains whose DRPs were purchased, and a mapping of the last checked dates for the DRPs. The domain details include the DRP details, the DRP price and the escrowed amount. To ensure ease of interaction among the DDA, CCP Check and DRP React contracts, two abstract contracts are defined – one for the CCP Check contract, and the other for the DRP React contract. Clients and domains are required to inherit their CCP Check contracts and DRP React contracts from these abstract contracts respectively, and the DDA contract verifies this at the time of client and domain registration. The abstract contracts provide some virtual methods which the inherited contracts can override, for example, the abstract contract for the CCP Check contract defines a virtual method check() which can be overridden by the inherited CCP Check contract, giving a client the freedom to specify the criteria to detect domain misbehaviour. The DDA contract also maintains an array of the log IDs of certain trusted CT logs, which are obtained from (Certificate Transparency Community Site-List of Known Logs). This array is passed to the CCP Check contract at the time of client registration.

4 Results

A prototype of the proposed system has been implemented using ReactJS (ReactJS Docs) and Material UI (Material-UI Docs) for the web interface, Node.js and Express for the back-end server and APIs, and Solidity v0.6 (Solidity Docs v0.6.0) for the smart contracts. The Truffle (Truffle Docs) package was used to compile and deploy the smart contracts to a local blockchain provided by Ganache. The prototype was also packaged using Docker. A list of known CT logs were used in the system (Certificate Transparency Community Site-List of Known Logs).

The smart contracts and the server APIs were tested using the Mocha and Chai JavaScript libraries. The tests included both unit and integration tests. We have used the following valid and invalid domains to test the domain behaviour and misbehaviour scenarios as shown in Table 2:

1 valid domains
   - google.com
   - github.com
   - facebook.com

2 invalid domains
   - no-sct.badssl.com
   - revoked.badssl.com
   - expired.badssl.com
   - self-signed.badssl.com.

Thus, the system prototype was able to identify invalid certificates such as those that are not logged in any CT logs, revoked certificates, expired certificates and also self-signed certificates. Using Truffle, the smart contracts were also deployed to the Rinkeby test network, and the web interface and the server were hosted on Heroku.

The deployment costs of the smart contracts, as well as the gas cost of the various contract operations have been estimated using Ganache and Remix IDE. It is to be noted that the gas cost of the write operations varies depending on the size of the data to be written, hence a reasonable estimate was obtained. The following tables summarise these costs: and do not consume any gas unless they are called from those functions that modify the Blockchain state.

Table 4 Deployment costs of the contracts in gas

Contracts	*Gas*
DDA	3967537
Check	466477
DRPReaction	215099

As seen in Table 4, the deployment cost of the DDA contract is significantly greater than that of the Check and DRPReaction smart contracts. This is due to the storage of the log IDs of the trusted CT logs when deploying the contract, which is a comparatively expensive operation. From Table 5, it can be seen that Register Client is the most expensive write operation. This is because when a client registers in the system, the

client's Check contract receives the list of log IDs of the CT logs used by the DDA contract. The Check contract then stores these values in a mapping data structure, which is a comparatively expensive operation. However, this significantly reduces the cost of the Check certificate operation, as searching for values in a mapping is an O(1) operation. This trade-off results in a higher gas cost when a client registers in the system, but makes the check certificate operation cheap and efficient, as can be seen in Table 5. The read operations shown in Table 6 do not consume any gas unless they are called from those functions that modify the blockchain state.

Table 5 Gas cost of the write operations in the dda contract

Operation	*Gas*
Register client	731,865
Update client	29,582
Register domain	241,629
Update domain	30,580
Purchase DRP	122,881
Check domain certificate (valid)	54,899
Check domain certificate (invalid)	69,536
Delete DRP from client list	24,069
Expire DRP	56,278

Note: These operations modify the blockchain state.

Table 6 Gas cost of the read operations in the dda contract

Operation	*Gas*
Check client registered	1,174
Check domain registered	1,219
Get client details	5,923
Get domain details	8,747
Get DRP purchase details	11,449
Get number of DRPs	1,110
Get number of client purchased DRPs	1,227
Get client purchased DRP details	11,567
Check CCP status	5,779
Check DRP status	6,652

Note: These operations read the state of the blockchain.

5 Conclusions

DDA is a decentralised system based on CT and IKP for detecting and responding automatically and immediately to domain misbehaviour. Smart contracts on Ethereum Blockchain are used to implement the various operations provided by the system. The system enhances the security of TLS protocols in establishing a secure, encrypted connection between clients and domains by monitoring domain behaviour in using

certificates. By monitoring domain certificates, the error behaviour of domains is restricted. The system also sufficiently incentivises clients and domains through rewards and punishments. Our system differs from ETDA in the sense that our system utilises a set of predefined CT logs that are supported by the major browsers today to verify digital certificates. Also, OCSP is used in conjunction with CT logs to validate digital certificates. Our system also differs in the transactions that occur during domain interactions with the system.

6 Future work

The following can be considered as future enhancements of the system:

- When registering in the DDA contract, a domain can use a DNSSEC based-proof to show its control over the DNS name. In the implemented prototype, any valid domain which is present on the Internet can register, but the system does not verify the association of the DNS name with the entity that is registering in the system.
- Clients and domains can register different versions of their CCPs and DRPs respectively. The CCP would call the corresponding DRP version to trigger the payments in the system.
- Certificate checking can include the verification of SCT signatures in the domain certificate. Verification of signatures in Ethereum is a prohibitively expensive operation which can be optimised in the future.
- Virtualised lists and tables can be used to effectively handle large data from the blockchain, thus improving the performance and efficiency of the dapp.
- More efficient implementation of the smart contracts to reduce the gas associated with the functions without compromising on security.

References

Arends, R., Austein, R., Larson, M., Massey, D. and Rose, S. (2005) 'DNS security introduction and requirements', *RFC*, March, Vol. 4033, pp.1–21.

Bhargavan, K., Delignat-Lavaud, A., Fournet, C., Pironti, A. and Pierre-Yves, S. (2014) 'Triple handshakes and cookie cutters: breaking and fixing authentication over TLS', *IEEE Symposium on Security and Privacy*, San Jose, United States, April, DOI: ff10.1109/SP.2014.14ff. Ffhal01102259f.

Certificate Transparency Community Site-List of Known Logs [online] https//www.certificate-transparency-community-site/known-logs.md at master • google/certificate-transparency-community-site • GitHub (accessed March 2021).

Certificate Transparency known logs' (2016a) April [online] http://www.certificate-transparency.org/known-logs (accessed October 2020).

Certificate Transparency log policy (2016b) October [online] https://www.chromium.org/Home/chromium-security/certificate-transparency/log-policy (accessed October 2020).

Cloudflare (2020) 'What happens in a TLS handshake', [online] https://www.cloudflare.com/learning/ssl/what-happens-in-a-tls-handshake(accessed November 2020).

Comodo (2011) 'Comodo fraud incident 2011-3-23', [online] https://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html (accessed 31 March 2011).

Delmolino, K., Arnett, M., Kosba, A., Miller, A. and Shi, E. (2016) 'Step by step towards creating a safe smart contract: lessons and insights from a cryptocurrency lab', Clark, J., Meiklejohn, S., Ryan, P.Y.A., Wallach, D., Brenner, M. and Rohloff, K. (Eds.): *FC 2016*, *LNCS*, Vol. 9604, pp.79–94, Springer, Heidelberg, DOI:10. 1007/978-3-662-53357-4 6.

Ethereum (2021) *Ethereum White Paper* [online] https://ethereum.org/en/whitepaper (accessed March 2021).

Google (2014) *Certificate Transparency: February 2014 Survey Responses,* February [online] http://www.certificate-transparency.org/feb-2014-survey-responses (accessed October 2020).

Hoffman, P. and Schlyter, J. (2012) 'The DNS-based authentication of named entities (DANE) transport layer security (TLS) protocol', *TLSA*, *RFC*, August, Vol. 6698, pp.1–37.

Hoogstraaten, H., Prins, R., Niggebrugge, D., Heppener, D., Groenewegen, F., Wettink, J. et al. (2012) Black *Tulip: Report of the Investigation Into the Diginotar Certificate Authority Breach* [online] www.rijksoverheid.nl/bestanden/documenten-en-publicaties/rapporten/2012/08/13/black-tulip-update/black-tulip-update.pdf (accessed September 2020).

How CT fits into the wider Web PKI ecosystem' [online] *How CT Works: Certificate Transparency* [online] https://certificate.transparency.dev/howctworks/ (accessed March 2021).

Langley, A. (2013) *Further Improving Digital Certificate Security*, December, http://google onlinesecurity.blogspot.com/2013/12/further-improving-digital-certificate.html (accessed October 2020).

Laurie, B., Kasper, E. (2012) 'Revocation transparency,' *Google Research*, September [online] www.links.org/files/RevocationTransparency.pdf (accessed October 2020).

Laurie, B., Langley, A. and Kasper, E. (2013) 'Certificate Transparency', *RFC*, Vol. 6962, June, DOI: 10.17487/RFC6962, https://www.rfc-editor.org/info/rfc6962>.

Laurie, B., Langley, A., Kasper, E., Messeri, E. and Stradling, R. (2021) 'Certificate transparency version 2.0', *Work in Progress*, *Internet-Draft* [online] https://datatracker.ietf.org/doc/html/draft-ietf-trans-rfc6962-bis-39>.(accessed 17 May 2021).

Material-UI Docs (2021) [online] https://material-ui.com/ (accessed April 2021).

Matsumoto, S. and Reischuk, R.M. (2017) 'IKP: turning a PKI around with decentralised automated incentives', *2017 IEEE Symposium on Security and Privacy* (SP), San Jose, CA, pp.410–426, DOI: 10.1109/SP.2017.57.

Microsoft Security Bulletin MS01-017 – Critical (2001) *Microsoft: Erroneous VeriSign-Issued Digital Certificates Pose Spoofing Hazard* [online] https://docs.microsoft.com/en-us/security-updates/securitybulletins/2001/ms01-017 (accessed March 2021).

Mills, E. and McCullagh, D. (2011) 'Google Yahoo Skype targeted in attack linked to Iran', March [online] http://www.cnet.com/news/google-yahoo-skype-targeted-in-attack-linked-to-iran/ (accessed October 2020).

Mozilla, (2021) *CA: Camerfirma Issues – MozillaWiki*, 25 January [online] https://wiki.mozilla.org/CA:Camerfirma_Issues (accessed Nov 2021)

Nakamoto, S. (2008) *Bitcoin: A Peer-to-Peer Electronic Cash System* [online] https://bitcoin.org/bitcoin.pdf (accessed September 2020).

Namratha, M., Aman, M.S., Pandey, S., Agrawal, R. and Nayazi, S.K. (2021) 'Decentralized domain authentication – exploratory literature survey', *2021 5th International Conference on Computing Methodologies and Communication* (ICCMC), pp.42–47, DOI: 10.1109/ICCMC51019.2021.9418434.

OID Registry (2021) [online] https://www.hl7.org/Oid/information.cfm (accessed April 2021).

Osborne, M.J. and Rubinstein, A. (1994) *A Course in Game Theory*, 12 July, p.14, MIT, Cambridge, MA, ISBN 9780262150415.

ReactJS Docs (2021) [online] https://reactjs.org/ (accessed April 2021)

Rescorla, E. (2018) *The Transport Layer Security (TLS) Protocol Version 1.3*, RFC, 8446, pp.1–160.

Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S. and Adams, C. (2013) 'X.509 internet public key infrastructure online certificate status protocol – OCSP', *RFC,* Vol. 6960, DOI: 10.17487/RFC6960, https://www.rfc-editor.org/info/rfc6960.

Solidity Docs v0.6.0 (2021) [online] https://docs.soliditylang.org/en/v0.6.0/ (accessed March 2021)

Truffle Docs (2021) [online] https://www.trufflesuite.com/docs (accessed April 2021).

Weibull, J. (1995) *Evolutionary Game Theory*, MIT Press, Cambridge.

Xia, B., Ji, D., Yao, G. (2017) 'Enhanced TLS handshake authentication with blockchain and smart contract (short paper)', in Obana, S. and Chida, K. (Eds.): *Advances in Information and Computer Security*, IWSEC 2017, Lecture Notes in Computer Science, Vol. 10418, Springer, Cham.