Timed Data Release using Smart Contracts

Jingzhe Wang†, Haocheng Wang†, Chao Li*, Balaji Palanisamy†
† University of Pittsburgh, School of Computing and Information, Pittsburgh, PA, USA
*Beijing Jiaotong University, Beijing, China

***Abstract*—We present a decentralized secure data release application built on Ethereum, named Timed Data Release (TDR). TDR allows users to encrypt sensitive data by enabling the decryption of the data only after a predetermined period of time has elapsed. We present the technical foundation for TDR and its implementation on Ethereum. Our demonstration features a microblogging application that enables scheduled publication of microblogs, showing a practical application of timed data release using smart contracts.**

***Index Terms*—Timed Release, Blockchain, Smart Contract**

I. Introduction

Consider a scenario where a data owner would like to publish their message at a future time point, while hiding the message until the release time. Several real-world applications employ this primitive; for example, secure auction systems require protection of important bidding information until all the bids arrive. Effectively protecting such messages before the release time is the central focus of the timed data release primitive. May [13] first introduced this problem. Later, Rivest et al. [14] suggested two promising solutions, namely *time-lock puzzle* and *timed encryption* with trusted time servers, which opened two directions of research in the literature. The most recent efforts in this line are those of [6], [16]. As a concurrent work, Liu et al. [12] had shown that incorporating witness encryption with blockchains is also feasible. Though these theoretical constructions ensure strong properties, their efficiency and scalability for large-scale data applications is still limited.

To resolve such tension, the notion of Timed Data Release (TDR for short) using blockchains was proposed, offering practical solutions for this problem. Recently, decentralizing TDR using blockchains has gained attention. Roughly, such a design constructs TDR on top of a blockchain network, where a group of participating nodes jointly protects the data. Since the first proposal of the blockchain-based design, there have been several efforts to develop this line of work, and their constructions have focused on the following aspects: (i) survivability in various adversarial contexts [5], [9], [10], [17], [18]; (ii) anonymization and efficiency [11]; (iii) enriching data control [19].

In this paper, we realize the notion of decentralized TDR using smart contracts. Our toolkit represents an engineering effort of the existing TDR protocol [11], [18] in such a way that users can use TDR in a black-box fashion. It consists of two key components: front-end interface and back-end TDR protocol. In detail, the front end allows potential users to provision key service parameters, required by the TDR protocol. The back end, upon observing service requirements, starts the TDR protocol. Later, when the release time arrives, users can issue requests to the front-end interface for releasing the protected data, realized through the interaction between the interface and TDR protocol.

Our toolkit emphasizes several features of our design. First, our design is not novel in theory; instead, it is a re-engineering of the status quo TDR, in a black-box fashion, offering the notion of "TDR as a Service"; second, our design is application-agnostic and can be integrated into potential applications seamlessly. To demonstrate the effectiveness, we showcase a demo system, capturing a microblogging application with an additional feature namely scheduled blog publication, supported by our TDR design. Through the demonstration, we show that our design is practical and we shed light on the effectiveness to other potential applications.

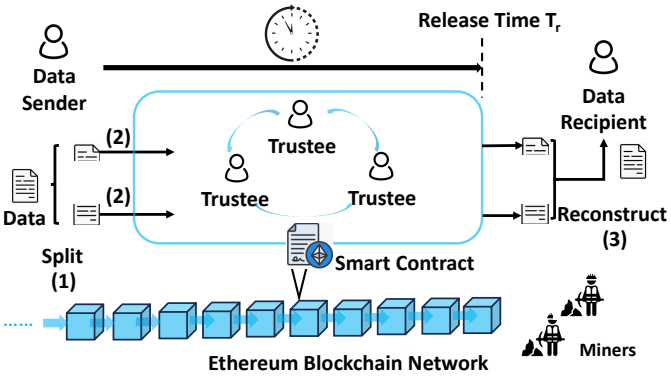


Fig. 1: Sketch of TDR

II. TDR Protocol

We introduce the preliminaries of TDR in this section. In Section II-A, we discuss the working of TDR and we present a concrete initialization of TDR in Section II-B.

A. TDR in a Nutshell

We illustrate the working of TDR with the help of Figure 1. Typically, the blockchain-based TDR consists of the following three key entities, namely *data sender*, the *Ethereum blockchain*, and *data recipient*. Specifically, *data sender* specifies a message requiring timed release and a prescribed time after which the message can be published. The sender then delegates the protection of the data to the Ethereum Blockchain Network.

We describe the workflow of TDR as follows: A data sender first prepares the data, specifies the prescribed release time

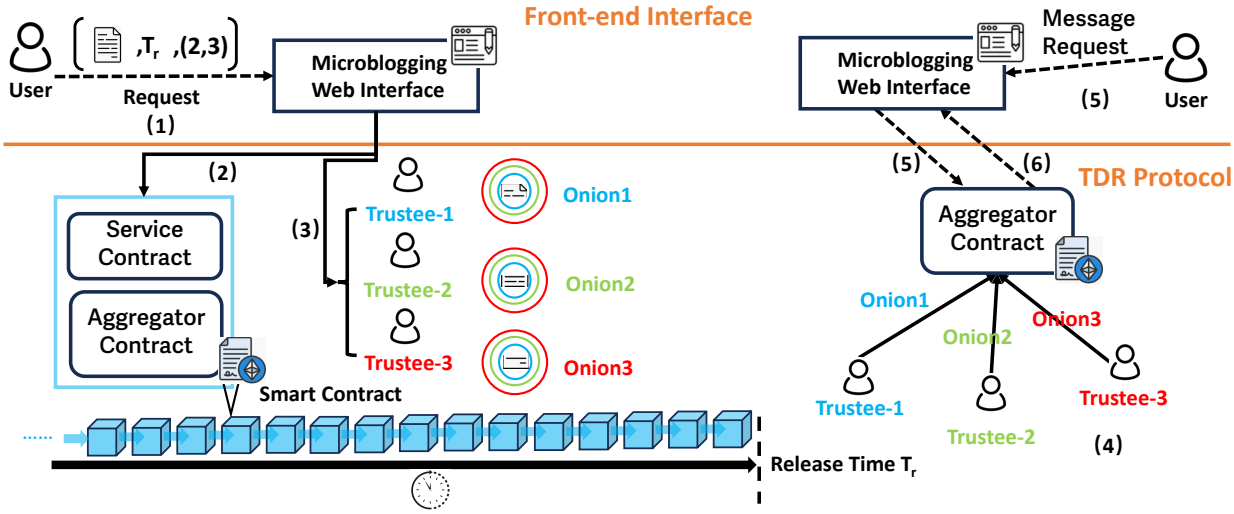


Fig. 2: Initialization of TDR

T_r of the data, and splits the data into multiple shares by adopting a specific coding scheme, such as *shamir secret sharing* [15](Step 1). The data sender then randomly recruits a set of nodes on the Ethereum network, called *trustees*, who are willing to provide service protecting the shares of the data. After that, the data sender disseminates the shares to the trustees(Step 2), which starts a TDR service. Later, when time hits the prescribed release time point, the recruited trustees publish their held shares. The data recipient can then reconstruct the data, by adopting a reconstruction algorithm from the specific coding scheme (Step 3).

Here, we would like to stress the role of the Ethereum blockchain network. First, Ethereum provides a decentralized context where a number of peers, holding Ethereum accounts, can actively participate in the decentralized TDR protocol. This feature enables Ethereum-based TDR to delegate the protection of the data to the crowd in Ethereum, avoiding the risks of a single point of trust; second, the TDR on Ethereum is typically governed by smart contracts, under the assumption that participating peers are rational, making any misbehaviors publicly verifiable and economically punishable [10].

B. Initialization of TDR

Our initialization of TDR, detailed in Figure 2, consists of two modules: front-end interface and the TDR protocol. The front end provides a unified interface that welcomes any user who is interested in using the TDR to schedule his/her future publication. Specifically, a user can provide the interface the data that needs to be scheduled and protected, the time that he/she would like to publish the data, and security parameters relevant to the Shamir secret sharing scheme (Step 1). Upon finalizing the parameters, an event will be triggered, which results in the following actions: (i) two smart contracts, Service Contract, and Aggregator Contract, will be deployed in Ethereum (Step 2). The former files and publishes the critical service parameter mentioned above while the latter will be in charge of releasing the protected data ; (2) the data will be split into multiple shares and encapsulated into onions by following the notion of *onion routing* [7], and distributed to the selected trustees (Step 3). After that, a TDR service normally starts and the time clock starts ticking. We refer the interested readers to [11], [18] for additional details on this protocol.

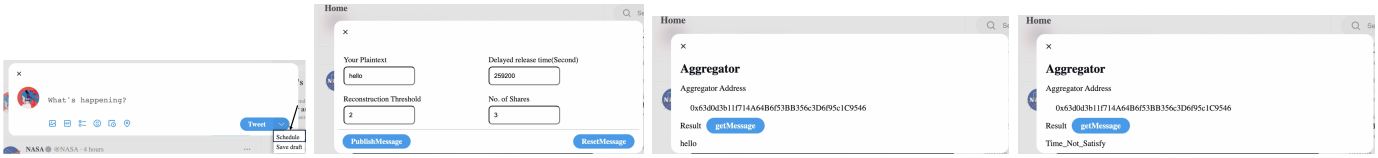
When the data is on-the-fly, it is protected jointly by the selected trustees. Later, once the time clock hits the release time, the reconstruction procedure will start. It consists of the following steps: the trustees submit their own onions and their private keys to Aggregator Contract (Step 4). If a data recipient would like to release the data, the recipient can trigger a release by requesting the web interface (Step 5). Based on the eligibility of releasing the data, the Aggregator Contract will reconstruct the message and send it to the web interface, which makes the data public (Step 6).

III. Demonstration: Microblogging using TDR

In this section, we provide the implementation details of our toolkit based on the design outlined in Section II-B and we further expand the use of our protocol to support scheduled blog publications using a microblogging application.

To support the interactions with Ethereum [1], we implement our design in the Truffle framework [4]. We program the smart contracts in Solidity [3] and deploy them using Ganache. We implement our web front end in Node.js [2].

We next demonstrate a concrete microblogging application for supporting scheduled blog publication. Here, for ease of demonstration, we build a Twitter-like web interface. As Figure 3a shows, any user can start scheduling their message by hitting the Schedule button. It will prompt a new page that requests the TDR parameters, including Message, Delayed Release Time, Reconstruction Threshold, and Number of Shares. Specifically, the Message field allows users to attach the blog content that requires a scheduled publication. In the example shown in Figure 3a, “hello” is the content



(a) **Schedule Blog** (b) **Service Provisioning** (c) **Legal Message Release** (d) **Illegal Message Release**

Fig. 3: Microblogging Application using TDR

of the scheduled blog. The Delayed Release Time is also specified by the user, which captures the time elapsed until the release of the message. We represent time in seconds. In the example, the time value "259200 secs"(3 days) indicates that the user would like to release the message after 3 days. To finalize the service configurations, the user should tell the interface the scheme parameters of *shamir secret sharing*, including Reconstruction Threshold and the number of shares. In the example, the user requires a (2,3) *shamir secret sharing* scheme to protect their message. After such provisioning, by clicking PublishMessage, the back-end TDR service starts.

Once 3 days have passed, the message becomes eligible to be published. Our interface, as shown in Figure 3c, prompts another page to support the message release. When a user wants to get the protected message, he/she can hit the getMessage button to trigger the reconstruction handled by the Aggregator Contract, yielding the resultant message "hello" as scheduled originally. However, if someone aims to publish the message at a time prior to the prescribed release time, our interface will intercept the request and give an exception message, as shown in Figure 3d.

IV. Conclusion and Future Work

We provide a practical implementation of TDR and demonstrate how to adopt its decentralized design for a microblogging application. Our future work aims at incorporating TDR in other data-oriented decentralized applications. Another direction of future work is focused on enhancing current designs of TDR by considering a more powerful mobile adversary [8], who could threaten the protocol by continuously corrupting trustees during their participation. In addition, further research may also develop more rigorous TDR constructions with additional provable security guarantees while retaining the efficiency and scalability offered by the smart contract-based decentralized approach.

Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant #2020071. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

References

[1] Ethereum. https://ethereum.org/en/, [Online].
[2] Nodejs. https://nodejs.org/en, [Online].
[3] Solidity. https://soliditylang.org/, [Online].
[4] Truffle suite. https://trufflesuite.com/, [Online].
[5] Enrico Bacis, Dario Facchinetti, Marco Guarnieri, Marco Rosa, Matthew Rossi, and Stefano Paraboschi. I told you tomorrow: Practical time-locked secrets using smart contracts. In *Proceedings of the 16th International Conference on Availability, Reliability and Security*, pages 1–10, 2021.
[6] Leemon Baird, Pratyay Mukherjee, and Rohit Sinha. i-tire: Incremental timed-release encryption or how to use timed-release encryption on blockchains? In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 235–248, 2022.
[7] David Goldschlag, Michael Reed, and Paul Syverson. Onion routing. *Communications of the ACM*, 42(2):39–41, 1999.
[8] Amir Herzberg, Stanisław Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing or: How to cope with perpetual leakage. In *Advances in Cryptology—CRYPT0'95: 15th Annual International Cryptology Conference Santa Barbara, California, USA, August 27–31, 1995 Proceedings 15*, pages 339–352. Springer, 1995.
[9] Chao Li and Balaji Palanisamy. Decentralized privacy-preserving timed execution in blockchain-based smart contract platforms. In *2018 IEEE 25th International Conference on High Performance Computing (HiPC)*, pages 265–274. IEEE, 2018.
[10] Chao Li and Balaji Palanisamy. Decentralized release of self-emerging data using smart contracts. In *2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS)*, pages 213–220. IEEE, 2018.
[11] Chao Li and Balaji Palanisamy. Silentdelivery: Practical timed-delivery of private information using smart contracts. *IEEE Transactions on Services Computing*, 15(6):3528–3540, 2022.
[12] Jia Liu, Tibor Jager, Saqib A Kakvi, and Bogdan Warinschi. How to build time-lock encryption. *Designs, Codes and Cryptography*, 86:2549–2586, 2018.
[13] Timothy C. May. Timed-release crypto. http://www.hks.net/cpunks/cpunks-0/1460.html, 1993.
[14] Ronald L Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release crypto.
[15] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
[16] Shravan Srinivasan, Julian Loss, Giulio Malavolta, Kartik Nayak, Charalampos Papamanthou, and Sri AravindaKrishnan Thyagarajan. Transparent batchable time-lock puzzles and applications to byzantine consensus. In *IACR International Conference on Public-Key Cryptography*, pages 554–584. Springer, 2023.
[17] Jingzhe Wang and Balaji Palanisamy. Securing blockchain-based timed data release against adversarial attacks. *Journal of Computer Security (to appear)*.
[18] Jingzhe Wang and Balaji Palanisamy. Attack-resilient blockchain-based decentralized timed data release. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 123–140. Springer, 2022.
[19] Jingzhe Wang and Balaji Palanisamy. Ctdrb: Controllable timed data release using blockchains. In *International Conference on Security and Privacy in Communication Systems*, pages 231–249. Springer, 2022.