Atomic Information Disclosure of Off-Chained Computations using Threshold Encryption*

Oliver Stengele✉ and Hannes Hartenstein

Institute of Telematics, Karlsruhe Institute of Technology, Karlsruhe, Germany
{oliver.stengele,hannes.hartenstein}@kit.edu

Abstract Public Blockchains on their own are, by definition, incapable of keeping data private and disclosing it at a later time. Control over the eventual disclosure of private data must be maintained outside a Blockchain by withholding and later publishing encryption keys, for example. We propose the Atomic Information Disclosure (AID) pattern based on threshold encryption that allows a set of key holders to govern the release of data without having access to it. We motivate this pattern with problems that require independently reproduced solutions. By keeping submissions private until a deadline expires, participants are unable to plagiarise and must therefore generate their own solutions which can then be aggregated and analysed to determine a final answer. We outline the importance of a game-theoretically sound incentive scheme, possible attacks, and other future work.

Keywords: Consensus · Off-Chain Construction · Atomic Disclosure.

1 Introduction

Decentralised consensus systems like Bitcoin and Ethereum brought with them the prospect of widespread disintermediation. However, it was soon realised that not every interaction could feasibly be recorded on a Blockchain. By using off-chain constructions, a Blockchain can still serve as coordinator and final arbiter, thus maintaining all the security guarantees, while minimising the space and effort required for permanent records.

Recently, off-chain mechanisms have been used to execute tasks that are too complex for the Blockchain, or rather the Smart Contract execution environment [11]. A prominent example of this concept is TrueBit [15]. However, there is a class of problems that are not well suited for systems like TrueBit. These problems follow the concept of a seminal paper by Ken Thompson on trust [16], namely that they are best approached through independent reproduction and verification [17] rather than individual and sequential challenges.

To facilitate independence in producing and reproducing solutions to a given task, the disclosure of said solutions is crucial. After all, if a solution is known,

[*] The final publication is available at link.springer.com: https://doi.org/10.1007/978-3-030-00305-0_7

then actually performing work to reproduce the same result or just copying it without performing any work is indistinguishable for an outside observer. To preempt this problem, we propose the use of threshold encryption to allow an arbitrary number of submissions to be disclosed atomically. By that we mean that participants can publicly commit to a solution, without revealing it to anyone, in such a way that all submissions will be disclosed simultaneously.

The remainder of this position paper is structured as follows. The subsequent Section 2 opens with motivating examples for the application for the proposed pattern and follows with related work from various fields of research. In Section 3, we outline the Atomic Information Disclosure (AID) pattern and discuss it in Section 4, in which we also enumerate future work and conclude the paper.

2 Problem Motivation & Related Work

In this section, we describe three examples for the application of the AID pattern and outline how previous works relate to our approach.

Compilation of Software As mentioned previously, Ken Thompson famously described how compilers could be compromised to embed flaws into binaries compiled with them, without leaving any indications for said compromise in the source code of the compiler [16]. As a countermeasure, Wheeler presents the concept of *Diverse Double-Compiling* [17], where the outputs of a potentially compromised and a trustworthy compiler are compared against each other to determine a correct build. In lieu of a single trusted compiler, we can use the AID pattern to generalise the concept by Wheeler to an arbitrary number of compilers in the hopes that the majority of them are not compromised. An interesting pitfall specific to this example stems from the fact that benign and malicious versions of the same binary can be very hard to distinguish. Making use of similarity metrics when analysing the results of multiple independent compilations could therefore cause more harm than good. We address this further in Section 4.

Verification of Software Similar to the first example, we can also look towards the verification of software that has already been compiled for a possible application of the AID pattern. Asking a single company to check a given piece of software for certain (security) properties leaves one open to the possibility of fraud. The company could simply certify that every property is fulfilled, without actually performing any work. The AID pattern can be used to pit several companies against each other, forcing them to perform their duties faithfully.

Experimental Science Lastly, we also note similarities between the functionality of the AID pattern and the scientific process regarding experiments. Reproducibility of results is one of the hallmarks of science. Contrary to the current model of publishing results and expecting others to either reproduce or refute them, which is open to plagiarism, a lack of reproduction, and other issues, a methodology based on atomic disclosure would preempt these problems by enabling independent groups of scientists to perform experiments in parallel and disclose their results simultaneously.

There are some similarities between these use cases that deserve explicit mention. In all cases, the effort necessary to produce and reproduce results is significant and comparable to each other. There is also an entire set of possible results from which a final answer is to be selected. And lastly, the trustworthiness or reliability of results grows with independent reproduction. As mentioned previously, it is the independence of reproduced solutions that we aim to achieve through the AID pattern.

Now, we will go over previous works from rather different areas of research that our approach is related to.

With Bitcoin [12], the feasibility of a Blockchain-based consensus system was first demonstrated. Ethereum [5] later generalised this concept to build a global state machine complete with a programmatic method for interacting with it, namely *Smart Contracts* [14].

Both in Bitcoin and in Ethereum, *miners* compete against each other to verify transactions submitted by users and publish them in newly mined blocks that are appended to the Blockchain. The miner that finds a new block is rewarded by the system for providing an integral service[1]. The remaining miners are then expected to check the newly published block for validity and to decide whether to build upon this new block, thereby proclaiming their accordance with it, or fork the Blockchain by building on top of the previous block in the event that they disagree with the new block. This verification of published blocks is not rewarded in any way, but is nevertheless a critical requirement to ensure the security properties of the Blockchain.

In the case of Bitcoin, the verification of new blocks is negligible compared to the process of mining. In Ethereum, however, this is not necessarily the case considering that arbitrarily complex Smart Contracts have to be executed in order to check the validity of any given block. Luu et al. [11] demonstrated that honest miners can be presented with a *verifier's dilemma* if the verification of published blocks requires non-negligible effort: If the miners choose to perform the verification, they put themselves at a disadvantage with regard to finding new blocks; if they forgo the verification, they risk mining on an invalid branch of the Blockchain. To combat this, Ethereum limits the amount of computation that can be executed and that must be verified within a block.

In general, it would seem that Blockchain-based consensus systems with mutual verification inevitably place an upper bound on the complexity of computations that can be executed and on whose results the system provides consensus. To circumvent these limitations, several mechanisms have been proposed in the past that allow processes to run *off-chain* while still relying on the Blockchain as the coordinator and final arbiter in case of dispute. Eberhardt and Tai [9]

[1] Note that this reward requires the agreement of other miners. If they fork the Blockchain and the block in question becomes *stale*, then the miner in question will not receive any reward.

present an overview of common off-chain patterns. While their list was compiled from past experiences, we look towards a new set of problems with our approach that has not been tackled yet.

Similar to off-chain patterns, in the sense that they enable the circumvention of the previously mentioned complexity bound, but vastly larger in terms of scale, are entire platforms like TrueBit [15]. In order to ensure the validity of outsourced computations, TrueBit employs a *verification game*, where a solver and a verifier narrow down the point of contention in the outsourced computation until it can be executed by Ethereum miners who subsequently resolve the dispute. While TrueBit would appear quite suitable for the problems at hand, we note that problems with a set of valid solutions could be challenged ad infinitum. Concurrently and independently generated solutions appear to be a valid approach to overcome this problem.

As previously mentioned, we propose to use a threshold encryption scheme to facilitate the submission of solutions in such a way that they become public simultaneously, thus allowing each solver to work independently. Threshold encryption was first introduced by Desmedt [7,8] and later improved by Pedersen [13] and Boneh [4]. Broadly speaking, threshold encryption enables the sharing of decryption capabilities among a group of parties such that t of them, called a *threshold*, have to cooperate in order to perform the decryption. The contribution by Pedersen [13] is especially noteworthy for demonstrating that a threshold encryption scheme can be constructed without a trusted dealer who would generate and distribute the individual key shares. Boneh [4] then improved the efficiency of threshold encryption schemes without trusted dealers to the point where an Ethereum Smart Contract could potentially verify the correctness of the setup and later perform decryption operations if enough parties publish their respective key share. The necessary operations fall within integer arithmetic that should be practical within Smart Contracts.

Very recently, Kokoris-Kogias et al. [10] have also employed threshold encryption to achieve distributed access control on a public ledger. While their construction of *One-Time Secrets* appears functionally similar to the AID pattern, their architecture requires the sender of a secret to be the trusted dealer in the threshold encryption setup. Our approach, by contrast, requires a dealerless threshold encryption scheme so that multiple senders can encrypt their secrets into one atomically disclosable pile. Our more open use-case also necessitates an incentive scheme and other security mechanisms.

In a way, the AID pattern could be classified as a form of pseudonymous voting on a correct solution to a given task. Voting on Blockchains is a comparatively young but quite fruitful area of research [1,2,18]. In a similar vein, but related to a very different area of computer science, one could also draw parallels between our pattern and *MapReduce* [6], with the *Map* phase being the off-chained computation and the *Reduce* step as the election of a final solution by a Blockchain.

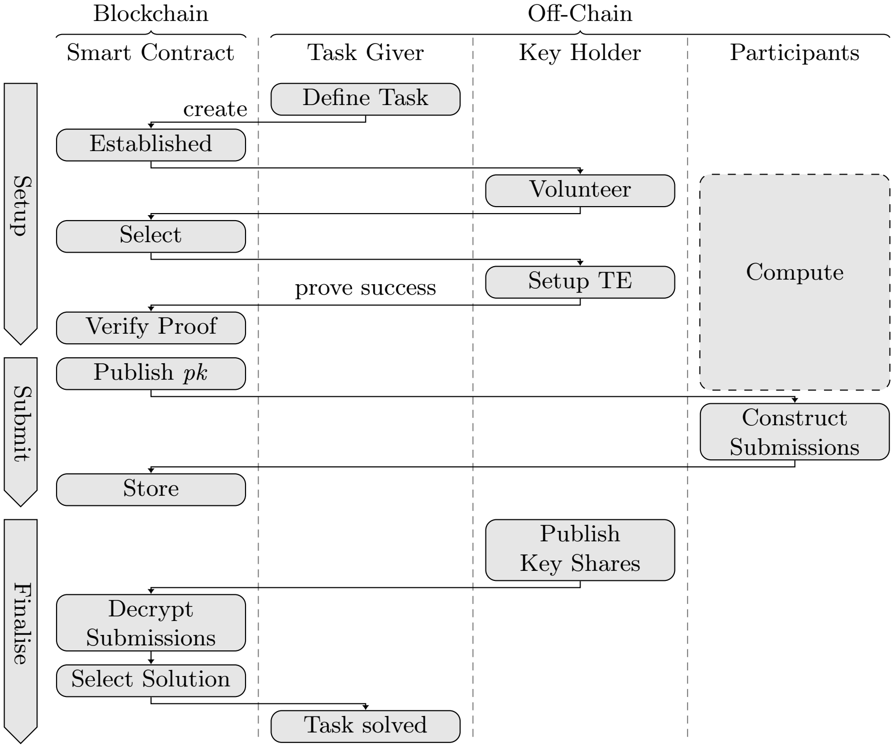

Figure 1. Sequence diagram of our off-chain pattern. Threshold encryption is abbreviated as TE.

3 Atomic Information Disclosure Pattern

In this section, we describe an off-chain pattern to solve a resource-intensive problem that requires parallel, independent reproduction of solutions through the use of threshold encryption. An overview of this pattern is presented in figure 1. The general idea of our pattern is to have an arbitrary number of participants generate solutions to a given task which are then published to a Blockchain where a final solution is elected from the candidates based on the number of times it was reproduced independently. Threshold encryption keeps the submissions private until the submission phase is over.

It is at this point important to note that we only discuss the general approach in this position paper. Crucial components such as the incentive scheme or implementation details are left as future work, see Section 4.

Setup Phase Initially, a task giver defines a Smart Contract that includes:

- The problem to be solved,
- A schedule for the subsequent phases,

- The logic for selecting a final solution from the submitted candidates.

This initial problem statement also includes integrity preserving references to any required data. In order to facilitate the independent generation of solutions, submissions need to be kept private until the stated submission deadline is expired. Since a public Blockchain is, by definition, incapable of executing this task, the privilege of revealing submissions has to be kept outside the respective Blockchain system.

While individual commitments by the participants could be used in this scenario, this would allow for a plethora of problems related to the opening of said commitments. For example:

- Participants could fail to open their commitments entirely due to crashes or human error, leading to ambiguity when it comes to electing the final result or at least wasted effort.
- Due to the sequential opening of commitments, participants may choose to not open theirs, erroneously assuming that they reached an incorrect solution that would not affect the outcome.

We propose to utilise threshold encryption to delegate and decentralise the privilege of disclosure without granting any party premature read access to the submissions to circumvent these issues. After the aforementioned problem statement is published on a Blockchain, a set of *key holders* is established through voluntary application and random selection. Alternatively, trustworthy key holders could also be vetted and hard-coded, similar to a permissioned Blockchain. We elaborate on this further in Section 4. These key holders then initialise a threshold encryption scheme among themselves, for instance the one presented by Boneh and Franklin [4], and prove the success of this setup to the Smart Contract by providing decryption shares to a predefined challenge, like the address of the coordinating Smart Contract. The Smart Contract can then check that all decryption shares are valid by attempting to combine them into a correct decryption. Note that decryption shares are distinct from key shares. If t is the threshold parameter of the underlying encryption scheme, any t decryption shares can be combined to decrypt a particular message whereas any t key shares can be combined to reconstruct the underlying private key which can subsequently be used to decrypt any message encrypted with this scheme, both past and future. We will employ the latter in the last phase of our pattern. Once the setup is complete, the key holders generate and publish a public key on the Blockchain that can be used to encrypt submissions. Note at this point that a threshold encryption scheme without a trusted dealer is essential for this application, as the dealer would otherwise be able to read all submissions and could therefore subvert the entire process.

Submission Phase Once the key holders have successfully initialised the threshold encryption scheme, participants can construct submissions by using the corresponding public key to encrypt their individual results. Depending on the expected number of submissions, they can either be recorded on the Blockchain itself or directed to off-chain storage like Ethereum Swarm or IPFS [3]. Keep in mind that participants could have started the necessary computation immediately after the task was published, parallel to the setup of the threshold encryption scheme. The submission phase lasts as long as the task giver specified in the beginning.

Finalisation Phase To begin the finalisation phase, the key holders are expected to publish their respective key share. This also serves as an irrevocable termination of the submission phase. If at least t key holders do this, all submissions will become readable to the public simultaneously. It is worth pointing out that once $t - 1$ key holders publish their key shares, the remaining key holders gain read access to the submissions. While this is certainly an advantageous position, it is very fleeting, since only one more key share suffices to extend read access to the public, and it is not very exploitable, as no new submissions by the key holders would be accepted at this point. The now public submissions can then be analysed and a final result can be elected based on the logic defined in the initial Smart Contract.

4 Discussion and Future Work

The purpose of this position paper is mainly to propose the use of threshold encryption in conjunction with Blockchains, especially those supporting Smart Contracts, to facilitate the concurrent and independent solving by multiple parties of certain problems that benefit from it. This benefit lies mainly in the increased certainty about the correctness of the solution. As such, several key aspects are left for future work. In this section, we elaborate on these and highlight possible pitfalls.

Probably the most crucial component after the functionally necessary primitives is the incentive scheme. Without a reason to both participate and to produce correct solutions, the whole scheme is futile. This issue becomes somewhat circular, given that we attempt to determine a correct solution through the process we try to incentivise based on the correctness of submitted solutions. Furthermore, the incentive scheme has to deal with possible collusions and bribery of task givers, participants, and key holders. This is doubly relevant since we do not enforce these roles to be disjoint. A task giver may also be a key holder and may also provide a solution. We intend to construct and game-theoretically analyse an incentive scheme in a future work.

Related to the incentive scheme but best viewed separate are possible attacks against the AID pattern. In general terms, we expect attacks that pursue any combination of these three goals:

- Influence the selection of the accepted solution
- Gain rewards disproportionate to the exerted effort
- Prevent the pattern from working entirely (*Denial of Service*)

Established off-chain mechanisms, like the Challenge-Response pattern mentioned by Eberhardt [9], may prove useful in stifling some of these attacks. One technique that deserves special mention at this point is sybil attacks. Without a widely adopted identity scheme, it is up to the incentive scheme to discourage participants from submitting the same solution multiple times though sybil accounts to either sway the final election or reap greater rewards compared to only submitting once. Similarly, a sybil attack on the threshold encryption scheme could enable the attacker to gain premature read access on the submissions. A collusion between sufficiently many key holders can accomplish the same goal without producing evidence on-chain. Here, a mechanism for rewarding the betrayal of such a collusion on-chain could be used as a countermeasure. The parametrisation of the encryption scheme and the selection of key holders is therefore a crucial line of defence. When deciding the parameters of the encryption scheme, availability and collusion resistance have to be weighed against each other carefully since they can be seen as opposing goals. The easier it is for the key holders to complete the protocol (availability), the lower might be the resistance to collusion, and vice versa.

The system used to store submissions is also a crucial component to mention at this point. During the entire process, it has to be available in addition to ensuring the integrity of submitted solutions. It would also be useful if the system could employ size and rate limits to impede denial of service attacks.

Lastly, we plan to put our pattern into practice with a functional prototype. Of primary interest in this regard are the costs for its execution and the strain we put on the selected Blockchain system relative to the number of participants.

One interesting pitfall we have already identified is the use and abuse of similarity metrics when electing final solutions. If the concrete application allows for such a metric to be defined in the initial Smart Contract, one might be inclined to use it to cluster submissions together in order to not require exact replication. This way, a more robust and reliable selection process might be possible compared to just looking for the number of reproductions. However, we must remark that the opposite is also possible. Since the metric is part of the initial Smart Contract, and therefore public, an attacker might construct malicious submissions that are similar, in terms of the metric, to the likely majority solution but functionally nefarious. The aforementioned clustering mechanism could then lend credence to such a malicious solution and increase the chances of its election as the final answer. This example serves to demonstrate how crucial the selection logic in the coordinating Smart Contract is.

In conclusion, we believe that the combination of threshold encryption and Blockchain-based consensus systems holds great potential for various applications that have not been feasible before. The ability to delegate the disclosure of data not to a singular third party but to a collective of key holders without granting premature read access promises to find application in various contexts. The off-chain pattern we outlined here is hopefully only a first step.

Acknowledgements

This work was supported by the German Federal Ministry of Education and Research within the framework of the project KASTEL_ISE in the Competence Center for Applied Security Technology (KASTEL).

We would like to thank the anonymous reviewers for their feedback, especially for bringing the work by Kokoris-Kogias et al. [10] to our attention.

References

1. Bartoletti, M., Pompianu, L.: An Empirical Analysis of Smart Contracts: Platforms, Applications, and Design Patterns. Springer International Publishing, Cham (2017)
2. Bartolucci, S., Bernat, P., Joseph, D.: SHARVOT: secret SHARe-based VOTing on the blockchain. arXiv.org (Mar 2018)
3. Benet, J.: IPFS - Content Addressed, Versioned, P2P File System. arXiv.org (Jul 2014)
4. Boneh, D., Franklin, M.: Efficient generation of shared RSA keys. In: Advances in Cryptology — CRYPTO '97, pp. 425–439. Springer, Berlin, Heidelberg, Berlin, Heidelberg (Aug 1997)
5. Buterin, V.: A next-generation smart contract and decentralized application platform. white paper (2014)
6. Dean, J., Ghemawat, S.: MapReduce - simplified data processing on large clusters. Commun. ACM **51**(1), 107 (2008)
7. Desmedt, Y.: Society and Group Oriented Cryptography: a New Concept. In: Advances in Cryptology — CRYPTO '87, pp. 120–127. Springer, Berlin, Heidelberg, Berlin, Heidelberg (Aug 1987)
8. Desmedt, Y.: Threshold cryptosystems (1993)
9. Eberhardt, J., Tai, S.: On or Off the Blockchain? Insights on Off-Chaining Computation and Data. Springer International Publishing, Cham (2017)
10. Kokoris-Kogias, E., Alp, E.C., Siby, S.D., Gailly, N., Jovanovic, P., Gasser, L., Ford, B.: Hidden in Plain Sight - Storing and Managing Secrets on a Public Ledger. IACR Cryptology ePrint Archive (2018)
11. Luu, L., Teutsch, J., Kulkarni, R., Saxena, P.: Demystifying Incentives in the Consensus Computer. In: the 22nd ACM SIGSAC Conference. pp. 706–719. ACM Press, New York, New York, USA (2015)
12. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. bitcoin.org (2008)
13. Pedersen, T.P.: A Threshold Cryptosystem without a Trusted Party. In: Advances in Cryptology — EUROCRYPT '91, pp. 522–526. Springer, Berlin, Heidelberg, Berlin, Heidelberg (Apr 1991)
14. Szabo, N.: Formalizing and Securing Relationships on Public Networks. First Monday **2**(9) (Sep 1997)
15. Teutsch, J., Reitweißner, C.: A scalable verification solution for blockchains. people.cs.uchicago.edu (Mar 2017)
16. Thompson, K.: Reflections on trusting trust. Communications of the ACM **27**(8), 761–763 (Aug 1984)
17. Wheeler, D.: Countering Trusting Trust through Diverse Double-Compiling. In: 21st Annual Computer Security Applications Conference (ACSAC'05). pp. 33–48. IEEE (2005)
18. Zīle, K., Strazdiņa, R.: Blockchain Use Cases and Their Feasibility. Applied Computer Systems **23**(1), 12–20 (2018)