Towards Contractual Agreements for Revocation of Online Data

Theodor Schnitzler[1], Markus Dürmuth[1], and Christina Pöpper[2]

[1] Ruhr-Universität Bochum, Germany [2] NYU Abu Dhabi, UAE
{theodor.schnitzler, markus.duermuth}@rub.de, christina.poepper@nyu.edu

Abstract. Once personal data is published online, it is out of the control of the user and can be a threat to users' privacy. Retroactively deleting data after it has been published is notoriously unreliable due to the distributed and open nature of the Internet. Cryptographic approaches implementing data revocation address this problem, but have serious limitations when considering practical deployment, and they have not been broadly adopted.

In this paper, we tackle the problem of data revocation from a different perspective by examining how contractual agreements can be applied to create incentives for providers to conform to expiration regulations. Specifically, we propose a protocol to automate the handling of data revocation. We have implemented a prototype smart contract on a local Ethereum blockchain to demonstrate the feasibility of our approach. Our approach has distinct advantages over existing proposals: It can deal with a wide spectrum of revocation conditions, it can be applied retroactively after data has been published, and it does not require additional effort for users accessing the published data. It thus constitutes an interesting, novel approach to data revocation.

1 Introduction

Cloud infrastructures and cheap storage are changing how we handle and share data and how long-lasting data becomes. While promising opportunities go along with this storage and access to data, concerns about the negative impact on ever available data and resources arise. As more and more personal information is shared online, the concepts of *digital forgetting* [19] and *revocation of online data* [28] become increasingly important for protecting our online privacy.

From the jurisdictional and also regulatory perspective, the European Parliament has approved the General Data Protection Regulation (GDPR) [9] on the protection of natural persons with regard to the processing of their personal data and on the free movement of such data. This *Right to be Forgotten* [10] already received considerable attention in 2014 due to a ruling by the European Court of Justice (ECJ) [11]. The ECJ determined that online search engines need to provide an interface and procedures for EU citizens to request the removal of their personal information from search results.

Technical approaches to implement the Right to be Forgotten apply cryptographic erasure, i. e., publishing data only in encrypted form and making it irretrievable after expiration by a suitable key management [5, 12, 23, 24, 26]. In this work, we explore a mechanism for the revocation of online data that does not purely limit the availability of data in technical terms but provides monetary incentives such that providers take appropriate measures to support data revocation and to comply with expiration conditions. In particular, we propose to conduct an agreement between a user, who owns a specific data item, and a platform provider, who offers access to the data. The agreement defines the expiration conditions for the data items. In contrast to existing approaches, our technique can be applied not only for data at the point of publishing but also for data that has already been made available in the past. Technically, we explore how smart contracts, as available in certain cryptocurrencies such as Ethereum [4], can be used to realize agreements between users and providers. Compared to other forms of reaching formal agreements, e. g., digitally signed PDFs, smart contracts allow a high degree of automation. Using a distributed ledger system is attractive, as it comprises a trusted third party in a decentralized manner. Our approach is designed to handle the majority of agreements on data expiration. Violations and disagreements in particular cases can still be handled in the jurisdictional system. Data owners might even refer to the contract as a piece of evidence in a potential legal action.

We assume that services featuring a revocation mechanism can also be attractive from the provider's perspective. The broader adoption of services such as Snapchat [29] and Signal [21] over several years shows that there is a demand for online interaction granting a certain level of privacy.

In short, this paper makes the following contributions:

– We design a data revocation scheme based on contractual agreements that can be applied both to new data and to data that has been published in the past. Users can take action both during the lifetime of data and after its intended expiration. To the best of our knowledge, this is the first approach to cover all these aspects at the same time.
– Our proposal is particularly efficient for fast data access without unnecessarily burdening users legitimately reading the published data.
– We provide an extensive overview of the design space of solutions for the revocation of online data based on cryptocurrencies and contractual agreements.
– We provide insights into an instantiation of our protocol: We have implemented a prototype as an Ethereum smart contract to demonstrate the general feasibility of our approach.

2 Solution Overview

We will now provide an overview of the basic ideas of our approach, before giving detailed insights into our protocol in Section 3.

There are three main entities in the system: the *user* U, who initially holds and owns the data and wants to publish it online. The data should be published

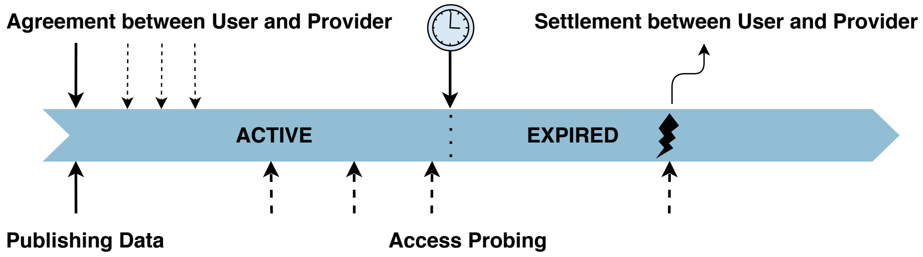
Agreement between User and Provider
Settlement between User and Provider
ACTIVE
EXPIRED
Publishing Data
Access Probing

Fig. 1. Timeline of our approach. User and provider conclude the agreement at the time of publishing or later. During its lifetime, the data is freely accessible on the provider's platform. A successful access after expiration leads to a settlement process.

with a *provider* P, who makes it publicly available on a large *online platform* in the WWW, such as a social network, or image sharing website.

Whereas previous work has tried to enforce automatic deletion of data, we pursue an idea where both the user and the provider establish a contractual agreement. This agreement states that P will take measures to limit the distribution of the data (e. g., by enforcing a limited lifetime on the data) and that U is entitled to compensation if the provider violates this agreement.

For this process, we distinguish the following four actions: (1) registering the agreement, (2) publishing the data, and (3) probing if the data is still available. When a violation is recognized we (4) enter a settlement process. We illustrate the chronology of these actions in Figure 1.

(1) First, the user and the provider need to reach a formal agreement. This agreement will include (i) the data d it concerns, by means of a unique identifier, (ii) the expiration condition on which the data should become unavailable, e. g., after a specific time t or an event e, and (iii) the penalty p for the provider in case the agreement is violated by the data being available after time t or event e. Similar agreements can be reached in the form of a traditional contract, but enforcing such agreements incorporating very small penalties, e. g., in the range of a few cents, is prohibitively expensive.

(2) The actual publishing of data typically involves sending the data to the provider, who will make it publicly accessible. One interesting property of this approach is that Steps (1) and (2) can be initiated in arbitrary order. In other words, it is also possible to reach such an agreement even after the data was published. While most other approaches require deciding on using protective measures and even fixing specific parameters such as the expiration time upon publishing, our system can also be invoked retroactively.

(3) After publishing and before expiration by meeting the expiration time t or the event e, the data is freely available. Accessing the data requires no additional tools or measures. The user can easily probe whether the data is still accessible before and after expiration.

(4) If the user detects a violation, i. e., the provider fails to handle the data correctly in making it available other than specified in the initial agreement, a penalty mechanism is triggered. A typical settlement could be a small financial compensation.

Adversary Model

The security notion for data revocation considered in previous works is security against a *retrospective attacker*: Basically, the retrospective adversary becomes active only after the data has been revoked. The attacker should not be able to reconstruct the original data after the expiration. This strong notion can typically only be achieved when the users are entirely honest and refrain from re-uploading the accessed data without protection to a different location. One can argue that re-uploading the data while still available constitutes an explicit act of archiving, in which case the data should indeed be available over time [19].

Additional aspects consider how the protocol can be exploited by the participants to obtain an advantage over the opposite party. The use of financial compensations might tempt the user to claim violations in cases in which the provider complied with the agreement. The provider is therefore interested in a guarantee that the data have actually been accessible at the time of the claim. At the same time upon violation claim, the provider might be interested in false statements, i. e., contending that the data do not exist. Thus, the user is interested in a decision finding that can objectively determine whether the data exists and that is resilient to a provider tampering with it.

3 Revocation Contract Scheme

We use Ethereum smart contracts [4] as a convenient method to specify and validate agreements between the data owner and the service provider on the expiration of data. However, Ethereum smart contracts comprise certain limitations, such as accessing data outside the blockchain and strictly enforcing payments between parties. For the specification of our protocol, we therefore require (i) the presence of a data feed for accessing external data and (ii) that a fraction of a potential penalty is placed as a deposit in the contract. The interactions in the stages of our protocol are illustrated in Figure 2.

1+2) Registration The registration process incorporates the first two steps as proposed in Section 2, since publishing data is a key subject of the agreement registration. A user U can publish a data item d along with an associated expiration condition exp_d on the platform of the provider P and will receive an identifier id_d for the data from the provider.

The user, who is identified by the address $addr_U$, can then initiate a transaction invoking the registration function of the smart contract C, passing id_d and exp_d to the contract.

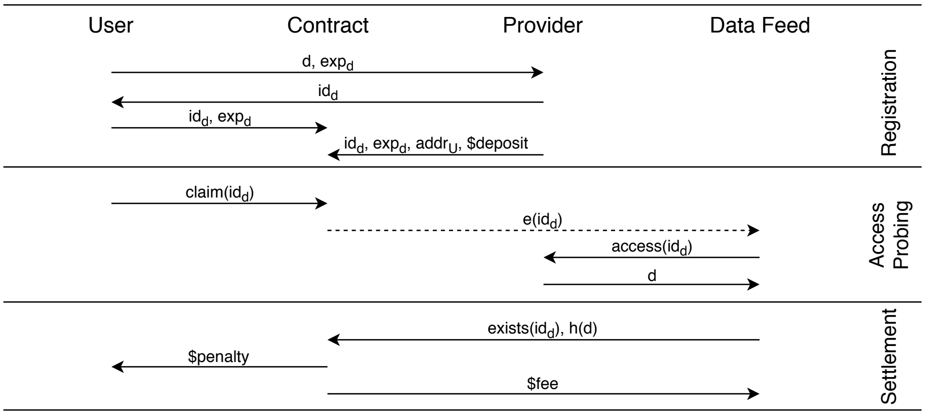

Fig. 2. Protocols of the registration, access probing, and violation settlement processes.

The registration process is finalized as soon as the provider also invokes the contract, providing id_d, exp_d and $addr_U$. Along with the confirmation transaction, the provider places a deposit in the contract, from which a potential penalty can be paid out. We assume that the deposit can be significantly smaller than the prospective penalty, obviously separated for each provider since we expect that contract violations and the resulting settlement processes will only occur in exceptional cases. Deposits are not bound to specific data items, but instead, the contract balance should cover penalties for a certain proportion of all data registered in the contract. In practice there is no fixed order for the last two steps, i. e., it is also possible that the provider is the first party to pass the registration information to the smart contract.

During the registration process, the contract may check the conditions for general plausibility. In case the expiration condition is, e. g., a time t, it should be checked whether t lies in the future. The registration process does not necessarily have to be completed immediately after the publication of the data. It can generally be initiated at any time after the data has been published on the provider's platform.

3) Access Probing After its publication, the data is freely accessible on the provider's platform. There is no need to encrypt the data prior to legitimately reading it during its lifetime. In addition, the owner can probe access in direct interaction with the provider without the contract being involved. We can also delegate this task to a third party, e. g., the owner can assign the task of access probing to a trusted service. This requires that the data is publicly available during its lifetime – our scheme does not cover non-public data.

4) Violation Settlement Ideally, the provider would have revoked the data according to the expiration condition specified in the contract. Nonetheless, if the owner detects that the data is yet available, a transaction can be initiated to notify the smart contract about the violation, passing the corresponding identifier along with the function call. Violations can be reported not only by the user but by everyone who has access to the data. Thus, the user can even conclude out-of-band agreements at will with dedicated services to check the availability. These services regularly observe the provider's compliance with the contract and initiate settlement on behalf of the user in the case of a contract violation. When the contract receives such a violation claim, it must be checked whether the data is available online. Therefore, the smart contract will request an external data feed service S to retrieve the data from the platform of the provider. Finally, S will initiate a transaction notifying the smart contract that the access attempt has been successful and will provide a hash $h(d)$ of the retrieved data. As the last step of the violation settlement, the smart contract will pay out the penalty to the user and a small fee to reward the data feed service for taking part in the protocol.

4 Protocol Design Space

Next, we sketch how extensions of our protocol and alternatives in its design influence specific properties of the protocol.

4.1 Data Identification

In a basic approach, identifiers assigned by the providers are stored in the smart contract for data identification. Usually, this identifier does not change during the lifetime of data, regardless where it is referenced. To improve user privacy, the identifier should not be stored in plain text, but in a cryptographically secure hashed form. However, uploading the data to an external website or even under a new identifier is sufficient to circumvent the protection.

Alternatively, data can be identified using hashes of the original data. Robust hashes [32, 33] are indented to tolerate minor modifications of data, such as rescaling, compression artifacts, and similar in the case of images. The use of such alternative identifiers is generally feasible within our proposed protocol, it only requires the location information as additional input in the first step of the settlement process.

Such extensions appear more desirable for the user, but put a significantly stronger burden on the provider. Location-independent data identification raises additional challenges with regard to the accountability of contract violations. Moreover, by storing images while they are available and re-uploading them at a later point, a malicious user may be able to force compensation payments by the provider, even though the provider has behaved correctly. However, users are not entirely protected from malicious providers or third parties. If the script is transparently available, an adversary can gradually introduce slight changes into the data until they are classified as different to circumvent protection.

4.2 Data Feeds

Our system requires the presence of a data feed to incorporate real-world events from outside the blockchain. However, the actual request processing, contacting external resources such as a website, raises concerns regarding the trust required in these services and challenges in validating the delivered results.

First, the response delivered by the data feed should be correct, i. e., the data feed should process the data as they are present at the time of the request. Such a response must be time-bound, because it is not sufficient to show that the data under consideration have existed on the website at any point in time before. Second, transactions performed on the blockchain are publicly visible and irrevocable. The data feed cannot simply access a website and write its contents (i. e., the data) to the blockchain, as this is effectively opposed to the goals of data revocation. We require the data feed to access the data from the provider website, perform an additional step such as determining whether the retrieved data comprise an image, and write the result to the blockchain.

Oraclize [22] and Town Crier [35] both use cryptographic means such as *TLS Notary Proofs* or *Software Guard Extensions (SGX)* to deliver a proof that certain data exist on a source website. From the perspective of the provider, such cryptographic attestations are attractive to prevent spurious claims from adversarial users. However, the use of a single service appears to be susceptible to malicious providers, as they could identify a particular data feed and deliver false responses to them to avoid the settlement process.

The concept of ChainLink [7] comprises multiple data feeds that access data independently from each other and are capable of performing computations on the retrieved data before responding. Its consensus mechanism for aggregating a result from the distributed responses (which can be built on trusted hardware such as SGX on an individual basis) reduces the trust required in the individual data feeds. Moreover, the use of a reputation mechanism that records false responses and a deposit-based penalty mechanism incentivize data feeds to deliver true responses. We imagine this as a crowd-sourcing approach in which even end-users can serve as verifiers in individual cases. Thus, massive misbehaving from the provider appears impossible, which ensures a high level of availability for such a system.

4.3 Complex Revocation Conditions

While the majority of prior work on data revocation simply use time-based expiration mechanisms, we suppose that smart contracts can be used to create almost arbitrary expiration conditions. However, these conditions need to be objectively observable through reliable data sources providing appropriate interfaces. Such scenarios would require an additional step in the verification step of our protocol, whereas the general procedure would be similar and security requirements would not be stricter than for the data access verification.

In a naïve approach, the expiration condition is represented as an expression in natural language. There is a need for reliable information sources that are capable of processing the information in their represented form, e. g. providing an interface for natural-language processing. The trust in these services (e. g., Google, Wikipedia, Wolfram Alpha) can be reduced through the use of decentralized verification as proposed by ChainLink, assuming that the service cannot distinguish verification attempts from regular requests to their services. In addition, the stability of information must be taken into account, i. e., the verification must be resilient to short-time changes, e. g., malicious users manipulating the information to their advantage and then triggering the settlement process. This is feasible, if Wikipedia is used as a knowledge base, but can be prevented by also considering the information history.

However, with more complex expiration conditions, we also see the possibility that data items can become valid again after they have expired. Examples are scenarios in which data are supposed to be available only on specific days of the week, or have a daily access limit (under the assumption that the number of accesses can be verified reliably). This property makes the use of smart contracts a much more powerful instrument than previous approaches.

4.4 Financial Reserve Model

In order to guarantee the payout of penalties, the provider needs to place a deposit in the contract that is paid out when the contract is violated. However, in the case of large-scale application of our system, this would lead to large amounts of provider capital locked in the smart contract. Thus, we propose that only a fractional reserve (e. g., 1%) has to be deposited in the contract, penalties are paid from the pool of all deposits, and that the provider has the possibility to withdraw money from the contract as long as the total amount is above the reserve threshold. This threshold is determined by the number of data items covered by the contract. If an item has expired and no violation has been reported for an adequate period of time, this item can be taken into account with less weight for calculating the threshold.

Large-scale violations that exceed the contract capacity determined by the total amount of deposits can still be handled resorting to the jurisdictional system. In this case, the contract can even be used as a piece of evidence to support that user and provider have agreed on the expiration of data beforehand.

5 Prototype Implementation

In this section, we describe our prototype implementation and evaluate the transaction cost incurring in its use and the scalability in terms of the numbers of data items that can be protected. Our contract employs time-based expiration conditions for data items that can be identified with a unique ID. We have implemented our prototype system using Ethereum with a local blockchain using the *Go Ethereum (Geth)* client. We have initialized Ethereum accounts representing a user, a provider, and the external service, as well as a smart contract in which items can be managed. For experimental interactions, i. e., registering or checking items with the smart contract, we utilized the *Ethereum Mist Wallet* application.

5.1 Smart Contract

Our Smart Contract consists of roughly 200 lines of Solidity code, is deployed on a local private blockchain, and available on GitHub[1]. The registration process consists of two steps that can be executed in arbitrary order. Both owner and provider have to commit information to create a valid entry. A user can register data in `addItem()`, providing as inputs its identifier and the remaining time it is intended to be available. The provider approves registration by using the function `confirmItem()`, also passing the identifier, the time left, and the owner's Ethereum address to the contract. We aim to ensure that the contract balance covers a minimum proportion of all registered data. After both parties have committed to the registration, the item agreement has become valid.

Data owners can be remove their data from the contract by calling the function `removeItem()`, which represents a cancellation of the agreement. Likewise, the provider can also withdraw confirmed items from the contract, as long as the owner has not added it.

The function `verifyAccess()` initiates the verification, whether data with a given identifier is available. In general, this function can be invoked by anyone and at any time, but the verification will only be triggered if the expiration date as stated in the record has been reached. However, access verification must be conducted by a data feed service external to the blockchain. For our prototype, we used an external script providing basic functionality. If the check is successful, the service can invoke the contract function `itemFound()`, which will initiate the compensation process. Thus, the contract will send the specified amount of Ether from its balance to the picture owner. In exceptional circumstances, i. e., if there is a widespread distribution of violations and many settlement processes are initiated at the same time, the Ether transfer may fail due to a low contract balance. The function `claimPending(id)` allows the user to initiate the compensation retroactively when the accessibility after expiration has already been verified before.

5.2 Evaluation

The feasibility of our approach mainly depends on the transaction cost arising from the use of the contract and the number of agreements that can be achieved in total.

Transaction fees in Ethereum, referred to as *gas*, generally depend on the complexity of the transaction, but can also be specified by the transaction sender. If a lower fee is selected, it may take a longer time-span for the transaction to be processed. As of December 2018, waiting times for transaction processing have been 45 seconds [8] on average. This time-span seems acceptable, as our application is not time-sensitive in terms of a few minutes.

[1] https://github.com/theoschnitzler/data-revocation-contract/

Table 1. Cost of contract execution. We assume gas fees of 1 GWei and an Ethereum exchange rate of $100.00.

Transaction	**Gas**	**USD**	**Actor**
Create	82,446	$0.0082	User
Confirm	35,107	$0.0035	Provider
CheckExpiry	27,199	$0.0027	User
VerifyAccess	28,621	$0.0029	User
ItemFound	22,071	$0.0022	Data Feed
ClaimPending	21,922	$0.0022	User

In Table 1, we illustrate the cost for the six transactions offered by our contract and the actor who needs to provide the fee on triggering the transaction. We assume an exchange rate of $100.00 per Ether and a transaction fee of 1 GWei. At the time of writing, miners of 30% of total blocks have accepted this or even a lower fee [8]. However, these numbers are constantly changing, due to the current network load. Registering a new data item to the contract requires a user to provide a transaction fee of 0.8 cents. For a user who uploads one item per day, this results in a total amount of roughly $3.00 per year. For each item confirmation, the provider has to bear costs of 0.35 cents. The cost of the other transactions will only incur in case of a contract violation. When we assume that the penalty for the violation is significantly higher (even in the range of a few dollars), the additional transaction cost will be negligible.

The amount of gas consumed by all transactions to be included in a new Ethereum block is limited to 8 million. When the contract registration requires 117,553 gas (see Table 1), and a new block is created every 15 seconds on average, this allows roughly 390,000 new registrations each day (2.75 million per week) under the assumption that the overall Ethereum network is not used for any other means. In 2017, Google has received roughly 2000 removal requests under European privacy law on a weekly basis [14], which is well below 0.1% of the limit that could be achieved within Ethereum.

6 Discussion

In this section, we discuss the effect of our proposal on trust assumptions and privacy aspects of practical implementations.

Trust Requirements Whereas we have introduced smart contracts for data revocation to reduce the trust required in providers when users upload personal data on their platforms, the use of data feeds still requires a certain level of trust. We consider the data feed a neutral adjudicator and, therefore, trust requirements in these entities are less critical than in providers who have — due to their business models — interest in making user content accessible as long as possible. However, trust in particular services is reduced by the presence of several alternatives users can ideally pick from, and also an approach based on crowd-sourcing. With regular other users serving as verifiers, it becomes harder for providers to make false attestations to data feeds to circumvent a violation detection, as regular accesses cannot be distinguished from accesses for verification purposes.

Metadata Privacy The data accessible to smart contracts is stored on the blockchain, a distributed and publicly readable and irrevocable data structure. This may lead to two challenges:

First, the arising database might be a valuable target for an adversary who aims to collect all the sensitive data prior to expiration. This mainly depends on the number of data items covered by the contract and can particularly become a problem when our system is only used for sensitive data. However, if there is also sufficient non-sensitive data covered by the contract, which we advocate for, an attacker cannot directly reason that the data is sensitive just from its presence in the contract.

Second, the contract data can reveal information about the privacy preferences of individual users. If many data items registered in the contract belong to the same owner, it is evident that this user attaches importance to privacy in common and data revocation in particular. On the other hand, we can also conclude that this user also shares a lot of data publicly. We assume that the blockchain interactions involving a specific user provide pseudonymity, without direct linkability to a concrete person. Thus, information derived from the contract does not comprise an additional privacy leak.

Provider Participation We see the use of smart contracts to register user data and specify expiration conditions similar to establishing a regular contract. Thus, a fundamental requirement is the willingness of service providers to support such a mechanism and active participation in the protocol. For a successful registration of a particular agreement, the provider must commit information to the contract. We cannot ultimately prevent that the provider might refrain from entering the agreement in specific individual cases. However, such a misbehavior can be observed, as there will be open registration requests by users, which are publicly visible on the blockchain. This makes it possible for a user to check upfront whether a provider actually complies with the system, before finally uploading new data.

Recent studies [3, 20] have found out that users of online social networks are actively employing privacy preferences as offered by the providers for their data. Therefore, we assume that applying a service for data revocation as we propose can make social networks more attractive to users, especially to those who are generally more concerned about privacy issues. Moreover, our scheme can be utilized as a reputation mechanism, in that providers use the system to demonstrate that they take user privacy seriously as they comply with the preferences defined in the contract. If more providers apply the system, they can even compete with each other in reaching the lowest violation rate. From a regulatory point of view, our approach can be considered as a technical instantiation for establishing the users' right to be forgotten. In the future, technical revocation mechanisms, but also our proposal for contractual agreements can provide directions towards the automated handling of such removal requests. This seems desirable not only for the users but also for providers, in that it renders the manual check of revocation requests and dealing with individual cases unnecessary.

7 Related Work

Cryptographic erasure mechanisms to enforce digital forgetting have been well studied. One of the first approaches was Ephemerizer by Perlman [23]. Vanish [12] uses distributed hash tables for key distribution EphPub [5] and Neuralyzer [34] use the Domain Name System as infrastructure for key distribution, and Reimann and Dürmuth [26] leverage the continuous change of website contents for key expiration. Amjad et al. [1] propose an approach raising the effort required to access data to prevent large-scale adversarial data collection during the lifetime of data. Chen et al. [6] propose a method to delete not only data but also remaining structural artifacts on a hardware level.

Opposed to data revocation scenarios, approaches exist to keep data secret after its publication until a certain release condition is fulfilled. Li and Palanisamy [17] present a scheme leveraging distributed hash tables that is built on the cryptographic foundations of time-lock encryption [18, 27]. In this context, Jager [15] shows that blockchain technology can be leveraged to emulate real-world-time in a computational model.

Studies have explored how much users are willing to pay to recover data that have become inaccessible [30] and for improvements in data privacy in an online shopping scenario [31]. González et al. [13] present a tool for computing the value of personal data uploaded to Facebook based on the financial revenue generated from the data, which can be useful to determine appropriate penalty amounts in our contract.

Integrating blockchain technology with privacy scenarios raises challenges in handling sensitive personal information. In this context, approaches to change the blockchain history without omitting advantages such as public verifiability have been proposed [2, 25]. Kosba et al. [16] introduce a framework for privacy-preserving smart contracts by developing a blockchain model of cryptography.

8 Conclusion

In this paper, we have developed an approach for the support of data revocation. Different from previous work, we did not use cryptographic measures, but combined both technical and regulatory aspects in order to incentivize the provider to delete data as determined by its owner. Based on this idea, we have developed a protocol for the specification of revocation conditions in smart contracts and implemented a prototype that supports time-based revocation conditions and is processed on a local Ethereum blockchain. The contract incorporates a penalty mechanism for data that remains available deviating from the conditions in the contract. With our approach, users can take action both proactively in defining expiration conditions for data they have published, and also retroactively in that they can get compensation in case the data provider has failed to delete data as specified.

Acknowledgments. This research was supported by the BMBF InStruct project under grant 16KIS0581.

References

1. Amjad, G., Mirza, M.S., Pöpper, C.: Forgetting with Puzzles: Using Cryptographic Puzzles to support Digital Forgetting. In: CODASPY '18. pp. 342–353. ACM (2018)
2. Ateniese, G., Magri, B., Venturi, D., Andrade, E.: Redactable Blockchain–or–Rewriting History in Bitcoin and Friends. In: EuroSP '17. IEEE (2017)
3. Ayalon, O., Toch, E.: Retrospective Privacy: Managing Longitudinal Privacy in Online Social Networks. In: SOUPS '13. pp. 4:1–4:13. ACM (2013)
4. Buterin, V.: A Next-generation Smart Contract and Decentralized Application Platform (2014), https://github.com/ethereum/wiki/wiki/White-Paper
5. Castelluccia, C., De Cristofaro, E., Francillon, A., Kaafar, M.A.: EphPub: Toward Robust Ephemeral Publishing. In: ICNP '11. pp. 165–175. IEEE (2011)
6. Chen, B., Jia, S., Xia, L., Liu, P.: Sanitizing Data is Not Enough!: Towards Sanitizing Structural Artifacts in Flash Media. In: ACSAC '16. pp. 496–507. ACM (2016)
7. Ellis, S., Juels, A., Nazarov, S.: ChainLink – A Decentralized Oracle Network (2017), https://crushcrypto.com/wp-content/uploads/2017/09/LINK-Whitepaper.pdf
8. ETH Gas Station: ETH Gas Station (2017), https://ethgasstation.info/
9. European Parliament: Regulation (EU) 2016/679 (General Data Protection Regulation). Official Journal of the European Union **59** (2016)
10. European Union: Factsheet on the "Right to be Forgotten" Ruling (C-131/12) (2014)
11. European Union Court of Justice: Judgment in Case C-131/12 (2014), press release No. 70/14, available online https://curia.europa.eu/jcms/upload/docs/application/pdf/2014-05/cp140070en.pdf
12. Geambasu, R., Kohno, T., Levy, A.A., Levy, H.M.: Vanish: Increasing Data Privacy with Self-Destructing Data. In: USENIX Security '09. pp. 299–316. USENIX (2009)
13. González Cabañas, J., Cuevas, A., Cuevas, R.: FDVT: Data Valuation Tool for Facebook Users. In: CHI '17. pp. 3799–3809. ACM (2017)
14. Google, Inc.: European Privacy Requests for Search Removals (2017), transparency report, available online https://www.google.com/transparencyreport/removals/europeprivacy/
15. Jager, T.: How to build time-lock encryption. Cryptology ePrint Archive, Report 2015/478 (2015), http://eprint.iacr.org/2015/478
16. Kosba, A., Miller, A., Shi, E., Wen, Z., Papamanthou, C.: Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts. In: SP '16. pp. 839–858. IEEE (2016)

17. Li, C., Palanisamy, B.: Timed-Release of Self-Emerging Data Using Distributed Hash Tables. In: ICDCS '17. pp. 2344–2351 (June 2017)
18. Mahmoody, M., Moran, T., Vadhan, S.P.: Time-Lock Puzzles in the Random Oracle Model. In: CRYPTO. vol. 6841, pp. 39–50. Springer (2011)
19. Mayer-Schönberger, V.: Delete: The Virtue of Forgetting in the Digital Age. Princeton University Press (2011)
20. Mondal, M., Messias, J., Ghosh, S., Gummadi, K.P., Kate, A.: Forgetting in Social Media: Understanding and Controlling Longitudinal Exposure of Socially Shared Data. In: SOUPS '16. pp. 287–299. USENIX (2016)
21. Open Whisper Systems: Signal (2010), https://signal.org/
22. Oraclize, Ltd.: Oraclize – Blockchain Oracle Service, Enabling Data-Rich Smart Contracts (2017), http://www.oraclize.it
23. Perlman, R.: The Ephemerizer: Making Data Disappear. Tech. Rep. SMLI TR-2005-140, Sun Microsystems Laboratories, Inc. (2005)
24. Pöpper, C., Basin, D., Capkun, S., Cremers, C.: Keeping data secret under full compromise using porter devices. In: ACSAC '10. pp. 241–250. ACM (2010)
25. Puddu, I., Dmitrienko, A., Capkun, S.: μchain: How to Forget without Hard Forks. Cryptology ePrint Archive, Report 2017/106 (2017), http://eprint.iacr.org/2017/106
26. Reimann, S., Dürmuth, M.: Timed Revocation of User Data: Long Expiration Times from Existing Infrastructure. In: WPES '12. pp. 65–74. ACM (2012)
27. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock Puzzles and Timed-release Crypto. Tech. rep. (1996)
28. Shein, E.: Ephemeral Data. Communications of the ACM **56**(9), 20–22 (2013)
29. Snap Inc.: Snapchat (2011), https://www.snapchat.com/
30. Spiekermann, S., Korunovska, J.: Towards a Value Theory for Personal Data. Journal of Information Technology **32**(1), 62–84 (2017)
31. Tsai, J.Y., Egelman, S., Cranor, L., Acquisti, A.: The Effect of Online Privacy Information on Purchasing Behavior: An Experimental Study. Information Systems Research **22**(2), 254–268 (2011)
32. Venkatesan, R., Koon, S.M., Jakubowski, M.H., Moulin, P.: Robust Image Hashing. In: ICIP '00. pp. 664–666. IEEE (2000)
33. Yang, B., Gu, F., Niu, X.: Block Mean Value Based Image Perceptual Hashing. In: IIH-MSP '06. pp. 167–172. IEEE (2006)
34. Zarras, A., Kohls, K., Dürmuth, M., Pöpper, C.: Neuralyzer: Flexible Expiration Times for the Revocation of Online Data. In: CODASPY '16. pp. 14–25. ACM (2016)
35. Zhang, F., Cecchetti, E., Croman, K., Juels, A., Shi, E.: Town Crier: An Authenticated Data Feed for Smart Contracts. In: CCS '16. pp. 270–282. ACM

All URLs have been last accessed Feb. 28th, 2019.