Public Accountability vs. Secret Laws: Can They Coexist?

A Cryptographic Proposal

Shafi Goldwasser
MIT and Weizmann

Sunoo Park
MIT

ABSTRACT

"Our Laws are not generally known; they are kept secret by the small group of nobles who rule us. We are convinced that these ancient laws are scrupulously administered; nevertheless it is an extremely painful thing to be ruled by laws that one does not know."
—Franz Kafka, Parables and Paradoxes.

Post 9/11, journalists, scholars and activists have pointed out that *secret laws* — a body of law whose details and sometime mere existence is classified as top secret — were on the rise in all three branches of the US government due to growing national security concerns. Amid heated current debates on governmental wishes for exceptional access to *encrypted* digital data, one of the key issues is: which mechanisms can be put in place to ensure that government agencies follow agreed-upon rules in a manner which does not compromise national security objectives? This promises to be especially challenging when the rules, according to which access to encrypted data is granted, may themselves be secret.

In this work we show how the use of cryptographic protocols, and in particular, the use of *zero-knowledge proofs* can ensure *accountability* and *transparency* of the government in this extraordinary, seemingly deadlocked, setting. We propose an efficient record-keeping infrastructure with versatile *publicly verifiable audits* that preserve *perfect (information-theoretic) secrecy* of record contents as well as of the rules by which the records are attested to abide. Our protocol is based on existing blockchain and cryptographic tools including commitments and zero-knowledge SNARKs, and satisfies the properties of indelibility (i.e., no back-dating), perfect data secrecy, public auditability of secret data with secret laws, accountable deletion, and succinctness. We also propose a variant scheme where entities can be required to pay fees based on record contents (e.g., for violating regulations) while still preserving data secrecy. Our scheme can be directly instantiated on the Ethereum blockchain (and a simplified version with weaker guarantees can be instantiated with Bitcoin).

1 INTRODUCTION

Twenty years ago, the clipper chip project which advocated that cryptographic keys for encrypted voice communications should be held in escrow by government agencies was discontinued. Today, brought to public attention by polarizing recent events such as the San Bernardino terrorist attack, the question of how (and if) access to plaintext should be made available to law enforcement agencies for encrypted digital information is back on.[1] Regardless of an eventual solution, it seems clear that a key factor in any workable solution will be the ability to trust government agencies to be *transparent* about their practices to the maximum extent possible consistent with their ability to enforce laws and national security. Secondly, and relatedly, it is important that formal procedures exist that ensure that government agencies can be held *accountable* under the law for their requests to access plaintext if required, so as to provide the guarantee that abuse of power can be detected *even in principle*.

These basic, natural requirements are further complicated by the increasing governance (or lack thereof) of intelligence operations by *secret laws*: that is, where *even the details of law itself are classified as top secret information*.[2] The very concept of secret law seems to go against the principles of accountability and transparency. Today's abundant use of secret law can render entirely indistinguishable to the public a lawfully behaving government from one that wantonly abuses power. Indeed, recent heated discussion among legal scholars challenges whether secret law is reasonable or even constitutional in the U.S.: for example, the American Civil Liberties Union (ACLU) recently filed a motion challenging the constitutionality of the secrecy of FISA court's secret rulings [21].[3] Our present purpose is not to delve into discussion of whether secret law is reasonable, but to consider the challenge of accountability and transparency in a world where secret law exists.[4] Recent work in the legal literature has begun to consider "rules of the road for governing secret law," and put forth pertinent accountability and transparency considerations: e.g., [20] refers to the importance of "public notification of secret law's creation, presumptive sunset and publication dates." The House of Representatives recently held a hearing on "Deciphering the Debate Over Encryption: Industry and Law Enforcement Perspective" [13] in which similar and more concerns were highlighted, e.g., [26].

[1] Technological development has wrought many important changes since twenty years ago: digital communication – encrypted or otherwise – has become prevalent and widely considered necessary to lead a normal life in the developed world; and alongside, government practices of monitoring, storing, and accessing ordinary citizens' digital data have expanded dramatically, overreaching what many thought acceptable constitutionally or from a civil liberties perspective [22]. Note that we refer to US government practices in this paper, while noting that evidence indicates that many other countries engage in similar behavior [22].

[2] Those not interested in legal subtleties may wish to skip this footnote, which provides clarification on the usage of the term "secret law" herein. In the U.S., laws themselves may not be secret. However, in the U.S. system of common law, the details of the way that a law shall be applied are often left ambiguous by the letter of the law, and are determined instead by the opinions and decisions issued by courts as cases arise. It is these opinions and decisions that may be classified as secret—but in such cases, it is effectively *the way the law is applied* that is kept secret. In this paper, we refer to this phenomenon as *secret law* even though it is not technically the law itself that is secret, because (a) writing "laws whose application is determined by classified court proceedings" every time would be confusing and cumbersome, and (b) in alternative legal systems where the application of the law is much closer to fully specified in the law itself, such as civil law systems that are predominant in much of the world, the closest equivalent might well be a *secret law*.

[3] The FISA Court is a U.S. federal court originally created in 1978 to oversee surveillance warrants against foreign spies, but whose oversight has since expanded to much more general intelligence operations, and by today "has created a secret body of law giving the National Security Agency the power to amass vast collections of data on Americans" [17]. Their operations are classified, and the court typically hears arguments only from the government [14].

[4] We refer to [20] for an examination of the evidence that secret law exists (it concludes that allegations of secret law in all three federal branches are well-founded).

The time-tested old method of reliable *record-keeping* and periodic audits is likely the most important and practical tool for accountability and transparency. This is true for individuals and businesses as well as for governments, and international organizations. The working assumption is that when record keeping is mandated by law or regulations, it will be followed.

However, there is only so much that traditional in-house record-keeping and audits can achieve. A main limitation is that much of the data kept in records is sensitive information that is not appropriate for routine disclosure, whether it be trade secrets, personal details of employees or clients or patients, or classified information pertaining to criminal investigations or national security. This means that irresponsible record-keeping tends to be revealed only in special cases where the disclosure of records are compelled (e.g., audits or court-ordered requests) rather than as a matter of routine. Public verifiability, which is desirable to achieve the goal of transparency, is usually out of the question in those special cases where records *are* checked. Another related limitation is that of timing and back-dating: even for those records which *are* later revealed (e.g., to some appropriate authority such as a judge), it is often impossible to be certain that the presented records are indeed the correct ones that were generated at the appropriate point in the past.

A naive approach to this problem might be to require organizations to hand over all their records as they are produced, e.g., each week or month. An immediate question is: *to whom?* An enormous—and, possibly misplaced—amount of trust and power would be placed in the hypothetical guardian of the sensitive internal records of all these countless organizations. Where would this vast and perpetually growing quantity of information be stored (and who would pay for it)? What if there were a hack or a leak? A breach of these records could be devastating to individuals' privacy nationwide (and likely internationally), and could cause far-reaching damage to businesses and economies. Aside from security concerns, integrity remains an issue: would there be any way for the public to be confident that the records used in future—e.g., for audits or released under freedom of information laws—really are the same as the data that was handed over by the company to the trusted party at the appropriate point in the past? This last question is particularly concerning in the case of the records of the record-keeping organization itself: *quis custodiet ipsos custodes?* And yet, naturally, in matters related to national security, the government may reasonably argue that transparency along these lines would be unacceptable.

1.1 Our contribution: publicly auditable records on secret data and secret laws

The focus of this work is to devise a solution that enables publicly auditable record-keeping while maintaining secrecy of recorded data and the secret laws that may govern it, without a trusted third party, which is compatible with the interests of all parties, whether governmental, corporate, or otherwise.

We propose a solution that makes use of a blockchain and other cryptographic methods to simultaneously achieve the following desiderata: *indelibility*, *secrecy*, *public auditability of secret data with secret laws*, *accountable deletion* and *succinctness*. A natural additional desirable feature of a record-keeping scheme (which we achieve but is of secondary importance) is to impose penalties for violations. Our construction can support penalties as well as versatile, secrecy-preserving "pricing schemes" based on record contents. This scheme can be instantiated on the Ethereum blockchain with no modification to the Ethereum protocol.

We now elaborate, informally, on each of the goals addressed.

Timestamping and indelibility. The creation time of a record should be public knowledge, agreed upon by all honest parties in the system.[5] Moreover, once records are created, it should be impossible to alter them without detection at a later date (that is, back-dating records should not be feasible).

Secrecy. Records kept by organizations contain sensitive information. Thus, any solution for the record-keeping problem must guarantee the strongest possible secrecy of the contents of records. In the best of worlds, a *perfect (information-theoretic) secrecy* guarantee should be sought, which would ensure secrecy of data even against arbitrarily powerful, computationally unbounded adversaries. Indeed, one instantiation of our construction efficiently achieves perfect (information-theoretic) secrecy of sensitive data.

The secrecy requirement is of paramount importance especially in the context of national security, where the cost of failures can be catastrophic. However, it is also very important in the context of businesses which hold secret information about business practices (this is desirable since the ability to keep trade secrets can benefit the economy), and sensitive information about employees, clients, and other individuals. Release of individuals' private information is not only an ethical issue, but also of legal significance as organizations are subject to complex regulations on handling of individuals' data, and in particular the sharing of such data with other parties. In some (though arguably, too few) cases, explicit consent from the individuals is a prerequisite to use of their data in certain ways; an effective accountability/auditing system must moreover be robust against reasonable individual exercise of privacy rights.

Public auditability of secret data with secret laws. In high-stakes settings like surveillance for national security, it is important for the public to be confident that governmental agencies are abiding by the law in their investigations. This holds even for secret law: the government should be able to assure the public, at the very least, of the fact that it is adhering to well-defined laws. This assertion in itself—that the government is behaving lawfully—is of great public interest and does not seem to pose a credible threat to national security in any circumstance. The interesting challenge that our work addresses is to devise a system allowing credible assertion by record-keepers that their recorded data adheres to specific regulations, while revealing *nothing else* about the data contents *or* the content of the regulations, in a provable (and information-theoretic) way.

Accountable deletion. Certain regulations (or an organization's internal policies) may require that recorded data be deleted, e.g., if a user closes their account, or simply because a certain amount of time has elapsed. Deletion of data is important for privacy and

[5] We acknowledge that there may be certain situations in which even the creation time of a record, i.e., the time of occurrence of an unknown event, might be sensitive information that is not considered suitable to be publicly known; in such (rare) cases, our system would not be an appropriate solution and alternatives should be sought.

security. Data deletion events must be logged in any organization's records, and the public audits mentioned above can be used to attest that organizations' records show compliance with regulations about data deletion (without, of course, revealing any further details about the data). Sometimes, deleted data could include the contents of past records: accordingly, we require that upon being served with a request to reveal a particular record, an organization must always be able to *either* verifiably reveal the requested data, *or* reveal the fact that it has been legitimately deleted (as evidenced by a later record).

Succinctness. The volume of data that is generated by organizations is growing every day, and the amount of data that is stored for record-keeping purposes is growing alongside. While record-keeping organizations themselves must find solutions to archive their own growing quantities of data, a system for accountable record-keeping which allows auditors, and ideally also the public, to verify compliant record-keeping, must process orders of magnitude less data. The importance of this requirement is compounded due to the number of record-keeping organizations. It would be unmanageable to process data in volumes comparable to the raw records.

Pricing schemes. It may be desirable to enforce fees based on the content of records. A simple example is to enforce a fine for violation of certain regulations, but payments need not necessarily be based on violations. For example, a local police department might be required to pay some fees depending on how much surveillance technology they deploy (e.g., over the course of a month), a quantity that should be logged in their records. As above, secrecy must be preserved: no information beyond the fact that the correct fee has been paid should be revealed by a secrecy-preserving pricing scheme. (In certain settings, it may be acceptable to reveal the fee amount too.)

1.1.1 *Overview of the proposal.*

We propose an infrastructure for timestamped record-keeping on a blockchain integrated with a system for routine, publicly verifiable audits that satisfies all the above desiderata. We consider the regulatory framework *together* with the technical component to comprise a full *record-keeping scheme*.

Participating organizations are required to keep data records by publishing cryptographic *commitments* to the records in the blockchain at regular intervals. When a participating organization is later required to reveal a record—e.g., at regular intervals according to their business practice, or during an audit—they can verifiably open the commitment to the record in question. The cryptographic commitments published on the blockchain will hide the record contents, but make it impossible for the organization which published the commitment to reveal anything different from the original record committed to. Refusal to open the record, if presented with a request from appropriate authorities, will result in legal or contractual consequences (as applicable in the circumstances). Thus, the only choices for a government or a business using the infrastructure would be: reveal the record (i.e., reveal its contents or attest to its legitimate deletion) *or* publicly refuse to open the commitment and face consequences.

Different types of cryptographic commitments could be used depending on the application, including ones which achieve perfect (information-theoretic) hiding. The latter would likely be the method of choice for high-stakes settings such as national security. With perfectly hiding cryptographic commitments, a commitment mathematically contains *no information* about the contents of the record, thus posting it on a blockchain reveals no information to an adversary regardless of computational power (except for the fact of the record's existence). At the same time, it will be impossible for a business, government, or organization to falsify a record (i.e., reveal a different record which is consistent with the commitment) under standard cryptographic hardness assumptions. Moreover, using standard techniques, the size of the commitment to a record does not grow with the size of the record.

So far, the technical component of the scheme bears substantial similarity to existing Bitcoin-based time-stamping services (see Section 1.2 for more discussion of related work). Next, we discuss how to achieve the novel and richer functionality of public auditability of secret data with secret laws and pricing schemes. To achieve these, we leverage additional cryptographic tools: notably, zero-knowledge arguments.

Secrecy-preserving audits. In a system where commitments to records are routinely published, publicly verifiable and secrecy-preserving audits can be implemented elegantly with zero-knowledge arguments. We propose a system for routine secrecy-preserving audits based on zk-SNARKs, a particularly efficient type of zero-knowledge argument which is also publicly verifiable. zk-SNARKs have already been useful in blockchain-based systems: they are used to achieve anonymity in recently launched cryptocurrency Zcash [3].[6] Under our scheme, organizations are required to post zk-SNARKs on the blockchain along with the commitments to their recorded data, as incontrovertible evidence that they are complying with applicable regulations. If the regulations themselves are secret, organizations must additionally publish *commitments to the regulations*, and prove compliance with *specific* regulations while keeping contents of both the regulations and the records secret. Further details are in Section 3.3.

Putting a price on records. Building upon the SNARK-based scheme, fines for non-compliance with regulations can be implemented automatically via transactions on the blockchains. This variation can be extended to support versatile system of fees (not necessarily fines) based on arbitrary properties of committed records. For example, a local police department might be required to pay some fees depending on how much surveillance technology they deploy, a quantity that should be logged in their records (and the fees could even go back to the local community). Using SNARKs, this can be implemented in a way that reveals nothing more than the amount of the fee paid and the fact that it is the correct fee amount for the committed record. Moreover, as above, compliance with fee-paying regulations can be asserted in a publicly verifiable manner even

[6] Note that zk-SNARKs require a one-time trusted setup, which can be implemented using multi-party computation (MPC) between a number of carefully chosen "trustees," and will be secure as long as not *all* of the trustees collude to subvert the protocol. Exactly such a trusted setup based on MPC was run in order to launch Zcash, as documented by [27]. See Section 3.3.1 for more discussion of SNARK setup.

when the regulations themselves are secret. Further details are in Section 3.3.2.

Our scheme would be able to be implemented on the Ethereum blockchain with no modification to the Ethereum protocol, satisfying all of the above properties simultaneously. In the body of the paper, we discuss how this instantiation would work, as well as discussing: a simplified scheme (satisfying indelibility, secrecy, and accountable deletion but without audits) that could be implemented on the Bitcoin blockchain with no modification to the Bitcoin protocol; and an enhanced scheme that also hides *fee amounts* paid in a pricing scheme, and could be deployed on a hypothetical version of Zcash/Ethereum that combines Zcash's anonymity features with Ethereum's flexible scripts and transaction formats.[7]

The general framework for publicly accountable record-keeping on secret data and regulations may find utility in other domains as well. To illustrate this point, we briefly describe one potential such application.

Whistleblowers. Public timestamping schemes could also serve as a tool for timestamped record-keeping to lend credibility to whistleblowers. Potential whistleblowers could commit to their data in a timely fashion and record the commitments on the blockchain, either with the explicit intent of later revealing it, or "just in case." A common tactic used against whistleblowers is to cast doubt upon their authenticity or even their sanity, and the presence of a timestamped trail of evidence could an helpful tool to boost their credibility. Moreover, we believe that the presence of *infrastructure* for "just-in-case whistleblowing" could cause an interesting and beneficial shift in the incentives of corrupt organizations and individuals therein. Currently, "good" individuals in corrupt organizations are often incentivized to keep quiet for fear of retribution. Though such individuals may not take the initiative to blow the whistle, it is strictly beneficial for them to anonymously post commitments to whistleblowing data in the blockchain, which they can come out and reveal in case of any future investigation in which the corruption is brought to light or the cause for fear is removed (e.g., due to change of employer). In the long run, perhaps this landscape of individual incentives could incentivize organizations to rein in corruption.

Finally, some remarks are in order.

Remark 1. *While it remains possible that an organization could simply refuse to open their commitments, this is analogous to the long-existing reality that an entity served with a subpoena can choose either to comply or face punishment under the law. In comparison to the existing system, our proposal provides stronger guarantees on the authenticity and public verifiability of records, and better incentivizes organizations to keep regular records as required by the law.*

Remark 2. *Our system does not provide a way to check that all generated records are committed on the blockchain. That is, a government or business could choose not to comply with regulations and not keep records. This is also the case today.*

[7] While we believe fee amounts would not be sensitive information in many settings, the ability to hide amounts could potentially enable (discussion of) more controversial pricing schemes, e.g., pay-per-wiretap.

Remark 3. *A corrupt organization could record false information in its logs. This is true when they are required to keep traditional/paper logs, and also true in the blockchain-based record-keeping scheme that we propose. Existing measures to deter and mitigate corruption, such as audits, could be applied to blockchain-based record-keeping too.*

Remark 4. *The question of what should be done if an instance of non-compliance is discovered by way of an accountable record-keeping scheme is beyond the scope of this paper, and is likely to be context-dependent. For example, when it is desirable that the identity of the creator of a record not be known, then it is problematic for the public to directly approach the creator of the non-compliant record with a complaint; in such cases, one solution could be to designate a specific (judicial) committee to receive and arbitrate complaints.*

1.2 Relation to existing services

The potential of blockchains for timestamping documents has long been recognized. A number of websites offer a timestamping service on the Bitcoin blockchain (and some also on other blockchains, notably that of Ethereum), wherein users submit documents of their choice to be timestamped, upon which the service creates a transaction containing the hash of the document and adds it to the Bitcoin blockchain. Some other services provide more complex frameworks for timestamped data management with the aim of selling their services as a generic or special-purpose data management solution to organizations. Features marketed include "proof of existence"; assurance of data "privacy", "integrity", "attribution", and "auditability"; and "scalability" to large volumes of data. To our knowledge, the auditability claimed by existing systems refers only to audits where the underlying data is revealed in an audit. We are not aware of existing proposals which put forward an inter-organizational regulatory framework (like our work), rather than targeting data management solutions within organizations. The desiderata for the two cases overlap to a degree, but have ultimately different goals: e.g., latency is a major concern of many existing services, and is not a limiting factor for us; whereas information-theoretic security and publicly verifiable audits on secret data with secret laws are paramount in our setting and not addressed by existing solutions. More detailed discussion of existing services is in Appendix A.

2 PRELIMINARIES

We write "PPT" to denote probabilistic polynomial time. For a randomized algorithm A, when we refer to the algorithm's randomness explicitly, we write $A(x; r)$ to denote the output of A on input x and randomness r.

2.1 Blockchains

In this subsection, we give a brief overview of the high-level structure of a blockchain with a stylized focus on the properties of blockchains that are relevant to this work.

A blockchain is a finite sequence of blocks $\mathbf{B} = (B_1, B_2, \dots)$ maintained by a decentralized network of parties, in the form of a public, append-only ledger whose contents all the parties are guaranteed to agree upon. Each block B_i contains some data including a set of *transactions*. In Bitcoin, the transactions typically record

monetary transfers of the form "Alice transfers x bitcoins to Bob."[8] More generally, a blockchain-based record-keeping system may instead support non-monetary "transactions" that serve to record certain data.

The entities that create blocks are typically called "miners," and we will refer to the entities that create transactions as "participants." Participants need not be miners, and vice versa. In order to have a transaction τ added to the blockchain, a participant broadcasts τ to the entire network of miners (we refer to this action as "sending τ to the blockchain network"); a miner who is creating a block will insert into the block some number of broadcasted transactions that have not yet been added to the blockchain.

Transaction fees. Transactions that are added to the blockchain may be accompanied by an optional transaction fee, i.e., a small amount of currency that is designated to be transferred to the party who creates a block containing the transaction in question and successfully adds that block to the blockchain. Having a transaction fee is usually advantageous for the transaction creator, as it incentivizes miners to include the transaction (quickly) in the blockchain.[9] The default presence of a nominal transaction fee is also important to prevent denial-of-service attacks by adversaries flooding the network with transactions (sometimes called "penny-flooding").

Record transactions. We describe our proposed scheme in the context of an abstract blockchain that includes a special transaction type called a *record transaction*. A record transaction does not entail any monetary transfer, and can store some arbitrary data δ of the transaction creator's choice. A record transaction $\mathsf{RecordTx}(pk, \delta) = (\zeta, \delta)$ contains the data D as well as any auxiliary information ζ that is required by the transaction format of the abstract blockchain. While we recognize that Bitcoin transactions can be used to instantiate our scheme, we prefer to first present our scheme in terms of an abstract blockchain since the underlying ideas are not Bitcoin-specific.

On the use of blockchains vs. other types of ledgers. Though it is presented as using a blockchain and we give discussion of instantiations based on Bitcoin and Ethereum, our main construction could also be instantiated based on other types of append-only ledgers, including ones which are not fully decentralized. Such options might be preferable in situations where a small set of stakeholders can plausibly be trusted not to collude together to falsify records. Building atop an existing decentralized blockchain such as Bitcoin or Ethereum can have the disadvantage that the miners on whom the integrity of the blockchain's content depends are typically unknown entities whose motives may be harder to ascertain than those of known reputable stakeholders.

An extension feature of ours — specifically, the system for automatic penalties for violations — relies more on the "smart contract" capability of the Ethereum blockchain specifically. For such a feature, use of an existing blockchain-based cryptocurrency might be advantageous in order to enforce penalties in a currency with exogenous monetary value.

[8] In fact, Bitcoin's scripting language, Script, allows for more complex monetary transfers that impose conditions upon the recipients, e.g., "Alice transfers x bitcoins to anyone who can find a SHA-256 preimage of value y."

[9] Indeed, although the transaction fee is technically optional in Bitcoin, in practice, transactions with fees below 0.00001 BTC are typically discarded as spam (according to https://en.bitcoin.it/wiki/Transaction_fees).

2.2 Cryptographic commitments

This subsection presents standard definitions of cryptographic commitments (including SHA-based commitment, Merkle commitments, and Pedersen commitments), and may be skipped by those familiar with these concepts.

Throughout this work, we use the term "commitment scheme" to refer only to *non-interactive* commitment schemes. Next, we define the syntax of a commitment scheme and informally describe the required security properties. We now recall the standard definition of a commitment scheme.

Definition 2.1 (Commitment scheme). A commitment scheme (for a message space $\mathcal{M}$) is a triple of probabilistic polynomial-time algorithms $\mathbb{C} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ as follows.

- $\mathsf{Setup}(1^\lambda)$ takes as input a security parameter λ (in unary) and outputs public parameters pp.
- $\mathsf{Commit}(pp, m)$ takes as input public parameters pp and a message $m \in \mathcal{M}$ and outputs a commitment c.
- $\mathsf{Open}(pp, (c, m', \omega'))$ takes as input public parameters pp, a message m', and a string ω' and outputs

$$\begin{cases} 1 & \text{if } c = \mathit{Commit}(pp, m'; \omega') \\ 0 & \text{otherwise} \end{cases} .$$

A *secure* commitment scheme is required to satisfy the following:

- *Hiding:* $\forall$ PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, $\exists$ negligible ε s.t. $\forall \lambda \in \mathbb{N}$,

$$\Pr\left[\begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda) \\ (m_0, m_1, \mathrm{state}) \leftarrow \mathcal{A}_1(1^\lambda) \\ b \leftarrow \{0,1\} \\ c \leftarrow \mathsf{Commit}(pp, m_b) \\ b' \leftarrow \mathcal{A}_2(c, \mathrm{state}) \end{array} : \quad b' = b \right] \leq 1/2 + \varepsilon(\lambda) . \tag{1}$$

  If $\mathbb{C}$ moreover satisfies (1) for computationally unbounded adversaries and $\varepsilon = 0$, then $\mathbb{C}$ is said to satisfy *perfect hiding*.
- *Binding:* $\forall$ PPT adversaries $\mathcal{A}$, $\exists$ negligible ε s.t. $\forall \lambda \in \mathbb{N}$,

$$\Pr\left[\begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda) \\ (c, m, \omega, m', \omega') \leftarrow \mathcal{A}(1^\lambda) \\ b \leftarrow \mathsf{Open}(pp, (c, m, \omega)) \\ b' \leftarrow \mathsf{Open}(pp, (c, m', \omega')) \end{array} : \quad m \neq m' \wedge b = 1 = b' \right] \leq \varepsilon(\lambda) . \tag{2}$$

  If $\mathbb{C}$ moreover satisfies (2) for computationally unbounded adversaries and $\varepsilon = 0$, then $\mathbb{C}$ is said to satisfy *perfect binding*.

For our purposes, the public parameters pp are assumed to be generated in an initial setup phase and thereafter publicly known to all parties, so we sometimes leave them implicit and write simply $\mathsf{Commit}(m)$ and $\mathsf{Open}(c, m, \omega)$ for brevity.

Remark 5. *It may be that in the context of a specific commitment scheme, not all of the commitment randomness needs to be passed as input to the opening algorithm; rather, it may suffice for the opening algorithm to take as input some function of the commitment randomness. This could be preferable if the function output is smaller than the commitment randomness, or if the opening algorithm can*

be more efficient by operating on the function output rather than the commitment randomness directly. We have written Definition 2.1 as above to avoid the additional notational clutter of defining some "decommitment information" that is distinct from the commitment randomness.

It is well-known that a commitment scheme cannot satisfy both perfect hiding and perfect binding simultaneously.

Definition 2.2 (Succinctness). A commitment scheme $\mathbb{C}$ is *succinct* if the size of commitments is independent of the message size.

SHA as a commitment scheme. As remarked in Section 1, SHA can serve as a very efficient and succinct commitment scheme in practice. Instead of running SHA directly on the data D to be committed, one should use $D||r$ as the input to SHA, where $r \leftarrow \{0,1\}^\lambda$ serves as commitment randomness. Essentially, the use of r ensures that two commitments to the same D will be indistinguishable from two commitments to different values. Note that this SHA-based scheme implements a perfectly secure commitment scheme in the random oracle model.

Definition 2.3 (Merkle commitment). Given any commitment scheme $\mathbb{C} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ and a length-halving (or succinct) collision-resistant hash function family $\mathcal{H}$, the Merkle commitment $\mathbb{C}^{\mathcal{H},B}_{\mathrm{Merkle}}$ is defined as follows.

- $\mathsf{Setup}^{\mathcal{H},B}_{\mathrm{Merkle}}(1^\lambda)$ outputs (pp, h) where $pp \leftarrow \mathsf{Setup}(1^\lambda)$ and $h \leftarrow \mathcal{H}$.
- $\mathsf{Commit}^{\mathcal{H},B}_{\mathrm{Merkle}}((pp,h),m)$ divides the message m into $B/2$-bit contiguous chunks, $m_1, \ldots, m_{\lceil 2|m|/B \rceil}$, computes $c_i = \mathsf{Commit}(pp, m_i; \omega_i)$ for random ω_i, and considers the commitments $c_1, \ldots, c_{\lceil 2|m|/B \rceil}$ to be "labels" on the leaves of a binary tree. The label of any node v in the binary tree is then defined to be $\ell_v = h(\ell_{v,0}||\ell_{v,1})$ and $\ell_{v,0}, \ell_{v,1}$ are defined to be the labels of v's children. Finally, the output is ℓ_{root} where root denotes the root node of the tree.

Note that in the case of Merkle commitment, not all of the commitment randomness needs to be passed as input to the opening algorithm; rather, it would suffice for the opening algorithm to take as input the set of all labels in the tree computed by $\mathsf{Commit}_{\mathrm{Merkle}}$ (and check the equality of the root node label with the commitment, and also check that each node's claimed label is indeed the hash of its two children's labels). In practice, this may be the preferable implementation (see Remark 5 for more discussion).

Partial opening. It is possible to open any consecutive pair (m_{2i-1}, m_{2i}) of the $B/2$-bit chunks of the message m by revealing the labels of the nodes along its path to the root (together with the labels of immediate siblings of those nodes). This opening procedure provably preserves the hiding property for all chunks of the message apart from m_{2i-1} and m_{2i}.

Generalized Merkle commitment. In Definition 2.3, the chunk sizes are fixed at $B/2$, but this scheme would work equally well with arbitrary, variable-size chunks. That is, the data could be split into chunks according to logical divisions, and the minimum appropriate number of chunks can be decided adaptively, case by case. (Reducing the number of chunks is beneficial for efficiency.)

Use of SHA for Merkle commitment. To use SHA to instantiate Merkle commitment, the randomized SHA-based commitment scheme described above should be used to generate the commitments for leaf nodes, and SHA can be applied deterministically in place of h at each non-leaf node of the tree.

2.2.1 Perfect hiding. Possibly the best-known perfectly hiding commitment scheme is the *Pedersen commitment* [19], which is based on the discrete logarithm assumption over strong-prime-order groups. The scheme is described below.

Definition 2.4 (Pedersen commitment).

- $\mathsf{Setup}_{\mathrm{Ped}}(1^\lambda)$ outputs (p, g, y) where $p = 2q + 1$ is a strong $(\lambda+1)$-bit prime, g is a random generator of $G = QR(\mathbb{Z}^*_p)$,[10] and y is a random element of G.
- $\mathsf{Commit}_{\mathrm{Ped}}((p,g,y),m)$ takes a message $m \in \mathbb{Z}_q$ and outputs $(c,(r,m))$ where $r \leftarrow \mathbb{Z}^*_q$ is randomly sampled and $c = g^r y^m \bmod p$.

The Pedersen commitment is unconditionally perfectly hiding since the commitment randomness r is chosen randomly in $\mathbb{Z}^*_q$ and therefore $c = g^r y^m$ is distributed randomly in G, for any message m. The computational binding property follows from the hardness of discrete logarithm.

2.2.2 Timed commitments. Timed commitments are a variant on the standard notion of commitments, in which "a potential forced opening phase permits the receiver to recover (with effort) the committed value without the help of the signer" [8]. Under assumptions about the inherent sequentiality of certain computations, it can be ensured that no commitment can be "forced open" until a certain amount of time has elapsed (and until then, the standard hiding property of commitment schemes applies). This type of commitment could be useful in special circumstances where it is desired that committed data be inevitably revealed after a certain delay, even without the committer's later cooperation.

Unfortunately, timed commitments cannot be succinct, by a straightforward information-theoretic argument. We thus remark upon this as an interesting theoretical possibility—perhaps suitable for use in rare cases—rather than a generally practical option.

2.3 Zero-knowledge SNARKs

A *SNARK* is a cryptographic proof primitive; the acronym stands for "Succinct Non-interactive ARgument of Knowledge." Our construction uses preprocessing zero-knowledge SNARKs (zk-SNARKs) for arithmetic circuit satisfiability,[11] following the construction of [5, 7]. "Preprocessing" means that there is a potentially expensive algorithm (Gen) that must be run as a one-time setup, on whose output the subsequent generation and verification of proofs depends. We leave implicit the term "preprocessing" from here on.

Definition 2.5 (Zero-knowledge SNARK). A preprocessing zk-SNARK comprises a triple of PPT algorithms $\mathsf{SNARK} = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Verify})$ as follows:

- $\mathsf{Gen}(1^\lambda, R)$ takes as input the security parameter λ and a description of a binary relation R (represented as an arithmetic

[10] $QR(\mathbb{Z}^*_p)$ denotes quadratic residues in $\mathbb{Z}^*_p$.

[11] We consider arithmetic circuits over an implicit field $\mathbb{F}$. In the implementation of [?], circuits should be over a suitable prime field $\mathbb{F}_p$; for details, see [?].

circuit of size polynomial in λ), and outputs a pair (pk_R, vk_R) of a *proving key* and *verification key*.

- $\mathsf{Prove}(pk_R, (x, w))$ takes as input a proving key pk_R and an input-witness pair (x, w) and outputs a proof π attesting to $x \in L_R$, where
$$L_R = \{x \ : \ \exists w \text{ s.t. } (x, w) \in R\} \ .$$
- $\mathsf{Verify}(vk_R, (x, \pi))$ takes as input a verification key vk_R and an input-proof pair (x, π) and outputs a bit indicating whether π is a valid proof for $x \in L_R$.

SNARK is a zk-SNARK if the following four conditions hold.

- *Completeness:* Honestly generated proofs must verify with overwhelming probability. Formally, $\exists$ negligible ε s.t. $\forall \lambda \in \mathbb{N}$, $\forall R$, $\forall (x, w) \in R$,
$$\Pr\left[\begin{array}{ll} (pk_R, vk_R) \leftarrow \mathsf{Gen}(1^\lambda, R) & \\ \pi \leftarrow \mathsf{Prove}(pk_R, (x, w)) & : \quad b = 1 \\ b \leftarrow \mathsf{Verify}(vk_R, (x, \pi)) & \end{array}\right] \geq 1 - \varepsilon(\lambda) \ .$$
- *Perfect zero-knowledge:*[12] The distribution of keys and proof reveals no information about the witness (i.e., can be simulated without the witness). Formally, there is a stateful PPT algorithm $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ such that $\forall R$, $\forall \lambda$, and $\forall (x, w) \in R$, the following distributions are identical:
$$\left\{\begin{array}{l} (pk_R, vk_R) \leftarrow \mathsf{Gen}(1^\lambda, R) \\ \pi \leftarrow \mathsf{Prove}(pk_R, (x, w)) \\ \text{output } (pk_R, vk_R, x, \pi) \end{array}\right\} \equiv \left\{\begin{array}{l} (pk_R, vk_R, \tau) \leftarrow \mathsf{Sim}_1(1^\lambda, R) \\ \pi \leftarrow \mathsf{Sim}_2(\tau, x) \\ \text{output } (pk_R, vk_R, x, \pi) \end{array}\right\} .$$
τ, sometimes referred to as the "simulation trapdoor," is the state passed from Sim_1 to Sim_2. In general, τ may be considered to contain (pk_R, vk_R) as outputted by Sim_1.
- *Succinctness:* For any $\lambda \in \mathbb{N}$, binary relation R, and $(pk, vk) \in \mathsf{Gen}(1^\lambda, R)$, an honestly generated proof has size $\mathrm{poly}(\lambda)$ (but constant in other parameters), and $\mathsf{Verify}(vk, (x, \cdot))$ runs in time $\mathrm{poly}(\lambda) \cdot O(|x|)$.
- *Proof of knowledge:* For any PPT algorithm Prove^*, there is a PPT algorithm Extract and negligible function ε s.t. $\forall \lambda \in \mathbb{N}$, $\forall R$, for any auxiliary input $z \in \{0, 1\}^*$,
$$\Pr\left[\begin{array}{ll} (pk_R, vk_R) \leftarrow \mathsf{Gen}(1^\lambda, R) & \\ (x^*, \pi^*) \leftarrow \mathsf{Prove}^*(z, pk_R, vk_R) & \\ w^* \leftarrow \mathsf{Extract}(z, pk_R, vk_R) & : \ b = 1 \wedge (x^*, w^*) \notin R \\ b^* \leftarrow \mathsf{Verify}(vk_R, (x^*, \pi^*)) & \end{array}\right] \leq \varepsilon(\lambda) .$$

We may leave implicit the unary security parameter input 1^λ.

3 RECORD-KEEPING SCHEME

We begin by presenting a basic framework in which it is possible for organizations to cryptographically commit to data, and to post these commitments to a public ledger. We then overview the different types of parties in our record-keeping system, and the roles they play. After that, we proceed to describe the full-fledged version of our system which incorporates zero-knowledge proofs that the committed data adheres to certain regulations.

3.1 Committing to records

In the following, the algorithms Commit and Open denote the commitment and decommitment algorithms of a succinct commitment scheme.

Algorithm 1. Commit to a record

Public parameters:
- Blockchain **B**.
- Succinct commitment scheme $\mathbb{C}$ = (Setup, Commit, Open).

Input: (pk, D) defined as follows:
- Public key pk of data owner/record creator.
- Data D to be committed.

(1) Generate public parameters $pp \leftarrow \mathsf{Setup}(1^\lambda)$.
(2) Generate a succinct commitment $c \leftarrow \mathsf{Commit}(pp, D; \omega) \in \{0, 1\}^\lambda$.
(3) Generate a record transaction $\tau = \mathsf{RecordTx}(pk, c)$.
(4) Send τ to the blockchain network (with an appropriate transaction fee to ensure that τ will appear in the blockchain within the next 24 hours).
(5) Store D and ω locally (or on the cloud) for record-keeping purposes.

The public key pk may be thought of as both an identifier of the data owner/record creator and as a public verification key for others to confirm a transaction's authorship.

The routine record-keeping procedure required of an organization is to keep records on a regular (e.g., daily) basis, and run Algorithm 1 daily to record corresponding commitments in the blockchain. If the organization is later required to reveal the records (e.g., due to a routine audit, or when presented with a warrant), then it runs the decommitment algorithm to prove that the revealed records indeed match the commitment that was added to the blockchain at the appropriate point in the past.

Succinct perfectly hiding commitments. SHA-based commitments are succinct and thus can be used directly as $\mathbb{C}$ in our construction. However, for the case of *perfect hiding*, recall that parameter sizes in the standard Pedersen commitment grow with the message size, and in particular, the commitment is not succinct. We can obtain a commitment scheme that is both succinct and perfectly hiding by first creating a Pedersen commitment c' and then using a succinct, computationally hiding commitment scheme $\mathbb{C}$ = (Commit, Verify) to *create a succinct commitment* $c = \mathsf{Commit}(c')$.[13] In other words, the SHA hash of a Pedersen commitment is still perfectly hiding.[14]

3.2 Basic regulatory framework

Parties. In the basic framework, there are 3 types of parties (each modeled as interactive Turing machines). All types of parties have read access to the blockchain, and may communicate along authenticated channels.

[12] The zero-knowledge condition can be relaxed to statistical or computational variants; these definitions are standard so we omit the details of the variants. The SNARK construction of [7] was originally stated to satisfy a weaker notion of zero-knowledge than that given here, but in fact [7]'s construction also satisfies the stronger definition given here, and the proofs of [7] suffice unchanged to prove the stronger definition [25]. That perfect zero-knowledge can be achieved is moreover remarked in the appendix of [6]. We note that the conference version of this paper [12] contained the weaker notion of zero-knowledge than the one given here, i.e., that used in the main part of [7].

[13] The transformation is generic, in that it would work for any other perfectly hiding commitment scheme too.

[14] Because the underlying commitment already achieves hiding and binding (and is randomized), it actually suffices to apply SHA directly here, rather than use the randomized SHA-based commitment scheme described in Section 2.

- Organizations $O \in \text{Org}$ that create records and insert them in the blockchain. Each $O \in \text{Org}$ has the following associated public parameters:
  - Public key $pk_O \in \{0,1\}^*$.
  - Frequency parameter $\varphi_O \in \mathbb{R} \cup \{\perp\}$, indicating that O commits to a record to the blockchain every φ_O days. (Further discussion of this parameter is given below.)
- Auditors $A \in \text{Aud}$, who may initiate two types of interactions:
  - Reveal committed data
  - Request arbitration of disputes
- The Court, C, which is an arbitrator of disputes between organizations and auditors.

The frequency parameter. By default, organizations should post record transactions to the blockchain on a regular schedule, so that the time of record posting does not reveal any sensitive information. That is, all data from a given time interval should be aggregated to create a single record, and an "empty" record should be created even if no new data was produced during the time interval.

In some scenarios, where the timing pattern of record generation does not constitute sensitive information, the rate of record-keeping is irrelevant. In this case, we consider the frequency parameter φ_0 to have the value $\perp$.

Record types and format. Each record transaction contains a commitment $c \leftarrow \text{Commit}(D)$, as described in Algorithm 1. The data D that is committed to may be of one of two types: *log data* or *regulations*. Formally, a regulation is represented as an efficiently checkable predicate p which takes as input some log data and evaluates to 1 if and only if certain requirements are satisfied. Regulations could represent laws or other desirable checks.

3.2.1 Instantiation with Bitcoin. Bitcoin's OP_RETURN transaction type can store up to 80 bytes of arbitrary data of the transaction creator's choice, and is thus suitable to serve as a "record transaction" in our basic record-keeping scheme.

3.3 Zero-knowledge proofs of compliance

Secret log data. The most basic type of check that could be usefully performed on committed log data is a syntactic format check. More generally, zero-knowledge proofs or arguments could be used to prove that specific regulations are being adhered to (for example, quotas not being exceeded, or patient status being recorded at the required time intervals) without revealing any information about the log data other than the fact that it adheres to these regulations. If non-interactive zero-knowledge proofs of compliance were to accompany each record transaction containing log data in the blockchain, then adherence to regulations would be publicly verifiable by anyone viewing the blockchain.

Algorithm 2. Post proof that secret log data satisfies public predicate

Public parameters:
- Blockchain **B**.
- Succinct commitment scheme $\mathbb{C}$ = (Setup, Commit, Open).
- zk-SNARK SNARK = (Gen, Prove, Verify).

Input: $(pk, D, p, (pk_R, vk_R))$ defined as follows:
- Public key pk of data owner/record creator.
- Data D to be committed.
- Predicate p to be attested to.
- Key-pair $(pk_R, vk_R) \leftarrow \text{Gen}(R)$ where R is a circuit accepting the following relation:

$$\left\{((c', p'), (\omega', D')) : c' = \text{Commit}(D'; \omega') \wedge p'(D') = 1\right\}.$$

Algorithm:
1. Generate a succinct commitment: $c = \text{Commit}(D; \omega) \in \{0,1\}^\lambda$, where ω denotes the randomness used to generate the commitment.
2. Generate a SNARK proof: $\pi = \text{Prove}(pk_R, ((c, p), (\omega, D)))$.
3. Generate a record transaction $\tau = \text{RecordTx}(pk, (R, vk_R, c, p, \pi))$.
4. Send τ to the blockchain network.
5. Store D locally (or on the cloud) for record-keeping purposes.

Algorithm 3. Verify that secret log data satisfies a public predicate

Public parameters:
- Blockchain **B**.
- Succinct commitment scheme $\mathbb{C}$ = (Setup, Commit, Open).
- zk-SNARK SNARK = (Gen, Prove, Verify).

Input: $(pk, \mathfrak{r})$ defined as follows:
- Public key pk of data owner/record creator.
- Record $\mathfrak{r} = (R, vk_R, c, \tilde{c}, \pi)$.

Algorithm:
1. Let $b = \text{Verify}(vk_R, ((c, p), \pi))$.
2. If $b = 0$, send $\mathfrak{r}$ to the Court C.

Secret regulations. To provide proof of compliance with a *secret* regulation p, an additional step is required: organizations must first create a commitment $\tilde{c}$ to the secret regulation p. Then, commitments to log data must be accompanied by a zk-SNARK attesting that the log data satisfies the unknown predicate committed to in $\tilde{c}$.

Algorithm 4. Post proof that secret log data satisfies secret predicate

Public parameters:
- Blockchain **B**.
- Succinct commitment scheme $\mathbb{C}$ = (Setup, Commit, Open).
- zk-SNARK SNARK = (Gen, Prove, Verify).

Input: $(pk, D, p, (pk_{\tilde{R}}, vk_{\tilde{R}}))$ defined as follows:
- Public key pk of data owner/record creator.
- Data D to be committed.
- Secret predicate p to be attested to.
- Key-pair $(pk_{\tilde{R}}, vk_{\tilde{R}}) \leftarrow \text{Gen}(\tilde{R})$ where $\tilde{R}$ is a circuit accepting the following relation:

$$\left\{((c', \tilde{c}'), (\omega', D', \tilde{\omega}', p')) : c' = \text{Commit}(D'; \omega') \wedge \tilde{c}' = \text{Commit}(p'; \tilde{\omega}') \wedge p'(D') = 1\right\}.$$

Algorithm:
1. Generate a succinct commitment: $c = \text{Commit}(D; \omega) \in \{0,1\}^\lambda$, where ω denotes the randomness used to generate the commitment.
2. Generate a succinct commitment: $\tilde{c} = \text{Commit}(p; \tilde{\omega}) \in \{0,1\}^\lambda$, where $\tilde{\omega}$ denotes the randomness used to generate the commitment.
3. Generate a SNARK proof: $\tilde{\pi} = \text{Prove}(pk_{\tilde{R}}, ((c, \tilde{c}), (\omega, D, \tilde{\omega}, p)))$.

(4) Generate a record transaction $\tilde{\tau}$ = RecordTx(pk, (R, vk_R, c, $\tilde{c}$, $\tilde{\pi}$)).
(5) Send $\tilde{\tau}$ to the blockchain network.
(6) Store (D, p) locally (or on the cloud) for record-keeping purposes.

Algorithm 5. Verify that secret log data satisfies a secret predicate

Public parameters:
• Blockchain **B**.
• Succinct commitment scheme $\mathbb{C}$ = (Setup, Commit, Open).
• zk-SNARK SNARK = (Gen, Prove, Verify).
Input: (pk, $\tilde{\mathfrak{r}}$) defined as follows:
• Public key pk of data owner/record creator.
• Record $\tilde{\mathfrak{r}}$ = ($\tilde{R}$, $vk_{\tilde{R}}$, c, $\tilde{c}$, $\tilde{\pi}$).
Algorithm:

(1) Let b = Verify($vk_{\tilde{R}}$, ((c, $\tilde{c}$), π).
(2) If b = 0, send $\tilde{\mathfrak{r}}$ to the Court C.

Reusing secret predicates across different log data items. When an organization attests to the satisfaction of a single secret predicate with respect to multiple log data items, it may be desirable that it be publicly known that the predicate attested to is the same across all the log data items concerned. In this case, the commitment $\tilde{c}$ to the secret predicate p could be reused for the zk-SNARK proofs generated for multiple log data items, rather than generating a fresb commitment $\tilde{c}$ for each log data item (as would be implied by naively applying Algorithm 5). Another alternative would be to allow certain records to reference previous records' commitments to secret predicates, rather than requiring a commitment to a predicate to be included explicitly in each record transaction.

The Court. The Court's task is to enforce publicly known policies for record-keeping under its jurisdiction, encompassing: the procedure by which an auditor can bring to the notice of the Court a failure of an organization to provide valid proofs of compliance regulations; the punishment associated with such failures; the circumstances in which recorded data can be legally compelled to be revealed; the steps that a requester must take to prove that his request is legitimate under the policies (e.g., presenting a warrant from a judge); the steps that the data owner can take to challenge the legitimacy of a request under the law; the time period within which requested data is required to be produced; and the punishment associated with non-cooperation with a legitimate request to reveal the data.

Mandatory data deletion. Certain regulations (or internal policies of specific organizations) require that recorded data is deleted after a certain period of time. Data deletion events should be recorded when they occur. Thus, an organization adhering to regulations, upon being served with a request to reveal a particular past data record, must always be able to either reveal the requested data by opening the commitment in the corresponding record transaction, or reveal a record that that data has been legitimately deleted, by (partially) opening a commitment in a subsequent record transaction. When a deletion event is included in a record, a Merkle commitment must be used to ensure that the deletion event can be revealed while keeping the rest of the record hidden.

Regulations as predicates. Not all laws can easily or succinctly be represented in the form of a well-defined predicate that log data must satisfy. The challenges of representing law in the form of code are interesting and have been recognized in the legal and computer science communities (e.g., [16]); while addressing these challenges is beyond the scope of our paper, we highlight that secret laws might be a setting in which purposely designing laws to be able to be represented in code might be of particular benefit, as it would facilitate the use of a secure accountability scheme such as our proposal. We note that in some cases, it may be useful to design the log format to include, at the very least, assertions that specific requirements have been satisfied, even if the adherence is too complicated to check directly.

3.3.1 Discussion about SNARKs. Using recent constructions [7] of zero-knowledge SNARKs (Succinct Non-interactive ARguments of Knowledge), proof sizes would be under 300 bytes at 128 bits of security on megabytes of record data, for natural checks such as quotas being respected, periodicity of certain types of logs, records being signed off by appropriate staff, or consistency between different parts of logged data. For [7], proof verification time is under 5ms regardless of the original program's running time. Each zk-SNARK would attest to the output (i.e., 0 or 1) of a specific predicate p evaluated on the input data (i.e., records) D. We discuss efficiency considerations in more detail shortly.

Keeping SNARK sizes small. Existing SNARK constructions work by converting a program to a circuit, and then generating a SNARK for the circuit. Since circuit size grows with the size of input data, the input data size can influence the succinctness of the proof and the efficiency of proof generation. Using the implementation and parameter settings of [7], megabytes of record data can be *directly* processed to yield proofs under 300 bytes at 128 bits of security. We believe this would be sufficient for many applications provided that organizations keep frugal plain-text logs of important records.[15] For cases when it is essential to process larger amounts of data in a single transaction, the recursive proof composition technique of [9] could be used to keep the proofs succinct (though proof generation time would grow).

Another factor that influences proof size and generation time is the complexity of the (circuit representation of the) predicate itself. For natural predicates such as quotas being respected, periodicity of certain types of logs, records being signed off by appropriate staff, or consistency between different parts of logged data, the circuit complexity can be linear in the data size (with small constants[16]), provided that "circuit-friendly" cryptographic primitives are used. Circuit-friendly constructions of collision-resistant hash functions are given by [1, 11] (and previously used in the context of SNARKs

[15] As opposed, for example, to storing all of their employees' HTML emails and word-processed reports, etc. The latter would seem to be an excess in many cases anyway, and potentially also disadvantageous as later references to the recorded data might have to search for a needle in a haystack.
[16] E.g., respecting quotas could be represented by "threshold" type predicates that would require just one gate per data bit; and using hash-and-sign with circuit-friendly signatures, signature verifications could be done with tens of gates per bit.

by [5]), and circuit-friendly constructions of a number of cryptographic primitives, including signature schemes, are given by [15]. Though we believe it likely that many natural predicates for routine audits would be relatively simple, we remark that in principle, our system would not be limited to such simple checks. Arbitrarily complex checks – described as predicates upon the committed data – could be performed, using recursive proof composition (as mentioned above) if necessary.

Finally, we note that [7] constructs *preprocessing* zk-SNARKs, in which the statement to be proven (or more precisely, the language to be attested to) must be known during the setup phase, and parameters established during the setup depend on the statement. This is compatible with our setting because we can consider the predicate to be an *input* to the statement to be proven, together with the data commitment. In the case of secret law, the input is just a commitment to the predicate, and the actual predicate serves as part of the witness.

Bitcoin's OP_RETURN transaction only holds up to 80 bytes, so the Bitcoin protocol could not be used unchanged for built-in audits. A factor of three or four increase in the data storage of an OP_RETURN transaction would suffice to support a SNARK-based audit scheme alongside our basic record-keeping scheme. The last increase of the OP_RETURN storage capacity was in late 2015 (Bitcoin Core 0.11.x), when it increased two-fold from 40 to 80 bytes. However, the Ethereum blockchain inherently supports storage of larger amounts of data, and we moreover suggest the use of Ethereum with our extended record-keeping scheme due to its more versatile scripting language, which can be useful for additional features such as pricing schemes, as discussed later.

SNARK setup. All existing zk-SNARK constructions require some form of trusted setup, either in the form of a structured common reference string, or a common random string. The more efficient constructions, such as [7], rely on a common reference string.[17] The common reference string can be generated using an efficient MPC protocol between a number of carefully chosen "trustees," and will be secure as long as at least one trustee is honest (and not compromised) [4]. Exactly such a trusted setup based on MPC was run in order to launch Zcash, as documented by [27]. Notably, *even if the setup is compromised*, it is possible to ensure that only soundness is damaged (i.e., it may be possible to generate proofs of false statements), but secrecy of data is still preserved [24].

Perfect secrecy. While many zk-SNARK descriptions in the literature refer to statistical zero knowledge, existing constructions achieve *perfect* zero knowledge at the cost of a negligible error probability [3, 6]. In our setting, this ensures perfect (information-theoretic) secrecy of committed data.

3.3.2 Putting a price on records. Recall that transactions are usually accompanied by a transaction fee that is transferred to the party who mines the block containing the transaction in question. This means that there is a small cost associated with adding each record transaction to the blockchain. We propose two alternative pricing schemes which could be useful in the context of record-keeping: a "free" scheme – for use when it is in the public interest to remove the burden of paying fees for record transactions – and a "paid" scheme implementing the pricing schemes discussed in Section 1, using zk-SNARKs.

(Almost) free record transactions. If it were determined to be in the public interest to remove the burden of paying fees for record transactions, then adding record transactions to the blockchain could be made (almost) "free" in at least a couple of different ways, as described below.

- *Fee waivers:* If the benefit were universally deemed to be worthwhile by miners, then miners could simply agree to use mining software that favors record transactions, instead of (roughly) prioritizing by highest transaction fees, as is currently the norm. Note that this system could not be taken advantage of by parties interested in performing monetary transactions for lower fees, since a record transaction (or Bitcoin's OP_RETURN transaction) does not allow for monetary transfer.[18] However, since a truly free scheme would be open season for denial-of-service attacks, we do suggest imposing a nominal fee that is much smaller than the fee for a regular transaction. (Note that a single satoshi – the smallest transferrable denomination of Bitcoin – is worth only a few thousandths of a US cent, at the time of writing.)
- *Fee subsidies:* The simple fee-waiving scheme described above is arguably "unfair" to miners, as they have to shoulder the cost in lost transaction fees, whereas the benefit of the record-keeping system applies to the whole population (i.e., not just miners). An alternative scheme that addresses this issue would be to have the transaction fees of record transactions be subsidized by the public, whether that be all members of the blockchain network, or all taxpayers in a jurisdiction. In the former case, a candidate scheme would be that a small fraction[19] of all transaction fees of regular transactions are set aside to be used for record transaction fees. In the latter case, a special account would be maintained by the tax authorities and funded by taxpayer money allocated for the purpose of subsidizing record transaction fees. Given a sufficiently rich scripting language for transactions, such subsidies could be automatically implemented through scripts.

Paid record transactions. Taking a different perspective, we believe there are many situations where it would be of public *benefit* for record-keepers to pay fees for logging their records. In combination with the secrecy-preserving audits of Section 3.3, fees could be charged based on specific properties of the committed records.

- *Fines:* Fines for (minor) violations of regulations could be automatically imposed as follows: either the record-keeping organization must either post evidence (i.e., a SNARK) showing that their record complies with a specific regulation, or

[17] We remark that the very recent zk-SNARK construction of [2] could be considered preferable in some respects, as it relies on the weaker assumption of a common *random* string and works in the random oracle model without knowledge assumptions (whereas [7] requires a knowledge-of-exponent assumption). However, the proof sizes in [2] are orders of magnitude larger than those in [7].

[18] More precisely: OP_RETURN allows for the transaction creator to "burn" money (i.e., render it unspendable) and to pay a transaction fee, but not to transfer money to recipients of his choice.

[19] E.g., 1%; in practice, the fraction would depend on the rate of transactions processed by the currency.

accompany the record transaction with a monetary transfer of a specified amount.
- *Routine fees:* Moreover, it would be possible to implement a versatile system of fees based on arbitrary properties of committed records. For example, a local police department might be required to pay some fees depending on how much surveillance technology they deploy (e.g., over the course of a month), a quantity that should be logged in their records.

A way to implement a fee scheme, like the ones described above, would be for miners to routinely check the SNARK attached to a record transaction and only accept the transaction into the blockchain if the appropriate fee is attached. Instead of simply attesting to the binary output of a predicate on the committed data, the SNARK would attest that a particular fee v is the correct output of a *fee program* $\mathbb{P}$ evaluated on the input data D. Natural fee schemes which depend on adherence to certain thresholds or count the number of records of a particular type (e.g., measuring resource usage) would, as discussed in the previous subsection, have relatively efficient circuit representations allowing for succinct proofs by direct application of [7] on megabytes of data.

Note that with a small modification to the blockchain protocol, no real trust assumption need be placed on the miners: if a careless or malicious miner includes an invalid record transaction in the blockchain, then this can be detected by any member of the network by checking the corresponding SNARK (recall that it is publicly verifiable), and the modified blockchain protocol dictates that invalid transactions be ignored. Another variant protocol could furthermore impose a monetary penalty on miners who include invalid transactions in their block (e.g., by taking away their mining reward), thus incentivizing miners to perform the required checks and refrain from polluting the blockchain with invalid transaction that will end up being ignored. We refer to [18] for technical details of how such penalties could be implemented, as they define a "penalty transaction" that can be repurposed for our setting.

To whom are the fees paid? The traditional answer would be: to the appropriate governmental or regulatory agency. In the setting of blockchains, the fees could also be distributed among (an appropriate subset of) the members of the blockchain network, or given back to miners. We suggest that such systems could be impactful in emphasizing direct accountability. For instance, returning to the example of the local police department which is charged for the amount of surveillance technology they use: the fees could be directly shared among all members of the locality who register a blockchain payment address in a local registry. If the registry itself were maintained on the blockchain,[20] then the distribution of fees could again happen automatically via scripts.

3.3.3 Instantiation with Ethereum (or Zcash). Ethereum transactions can store larger amounts of data than Bitcoin transactions, and in particular, fit the zk-SNARKs required for our full record-keeping scheme. Ethereum script is Turing-complete, so can be used to implement the automatic payment schemes discussed above.

[20] E.g., a local resident could register their payment address by having the mayor's office digitally sign it, and post the payment address and signature on the blockchain in a special "registration transaction". This would be publicly verifiable using the verifiation key of the mayor's office.

In specific cases where it is required to implement secret law together with secret pricing schemes where the fee amounts cannot be revealed, one would need an anonymous cryptocurrency (such as Zcash). Unfortunately, Zcash transactions and scripting currently resemble Bitcoin's, and thus would not be suitable for deploying our full record-keeping scheme. A hypothetical extension of Zcash including more expressive scripting and transactions with enough data storage to fit zk-SNARKs would support our full scheme.

4 ACKNOWLEDGEMENTS

We are grateful for insightful and helpful conversations with Oded Goldreich, Mark Moir, Daniel Weitzner and Jonathan Frankle. We also thank Madars Virza for useful discussions about zk-SNARKs.

This research was supported by the following grants: NSF MACS (CNS-1413920), DARPA IBM (W911NF-15-C-0236), and SIMONS Investigator Award Agreement Dated June 5th, 2012.

REFERENCES

[1] Miklós Ajtai. 1996. Generating Hard Instances of Lattice Problems (Extended Abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, Gary L. Miller (Ed.). ACM, 99–108. https://doi.org/10.1145/237814.237838

[2] Eli Ben-Sasson, Iddo Bentov, Alessandro Chiesa, Ariel Gabizon, Daniel Genkin, Matan Hamilis, Evgenya Pergament, Michael Riabzev, Mark Silberstein, Eran Tromer, and Madars Virza. 2016. Computational integrity with a public random string from quasi-linear PCPs. *IACR Cryptology ePrint Archive* 2016 (2016), 646. http://eprint.iacr.org/2016/646

[3] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. 2014. Zerocash: Decentralized Anonymous Payments from Bitcoin. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*. IEEE Computer Society, 459–474. https://doi.org/10.1109/SP.2014.36

[4] Eli Ben-Sasson, Alessandro Chiesa, Matthew Green, Eran Tromer, and Madars Virza. 2015. Secure Sampling of Public Parameters for Succinct Zero Knowledge Proofs. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*. IEEE Computer Society, 287–304. https://doi.org/10.1109/SP.2015.25

[5] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. 2014. Scalable Zero Knowledge via Cycles of Elliptic Curves. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II (Lecture Notes in Computer Science)*, Juan A. Garay and Rosario Gennaro (Eds.), Vol. 8617. Springer, 276–294. https://doi.org/10.1007/978-3-662-44381-1_16

[6] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. 2014. Scalable Zero Knowledge via Cycles of Elliptic Curves. *IACR Cryptology ePrint Archive* 2014 (2014), 595. http://eprint.iacr.org/2014/595

[7] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. 2014. Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture. In *23rd USENIX Security Symposium (USENIX Security 14)*. USENIX Association, San Diego, CA, 781–796. https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/ben-sasson

[8] Dan Boneh and Moni Naor. 2000. Timed Commitments. In *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings (Lecture Notes in Computer Science)*, Mihir Bellare (Ed.), Vol. 1880. Springer, 236–254. https://doi.org/10.1007/3-540-44598-6_15

[9] Alessandro Chiesa, Eran Tromer, and Madars Virza. 2015. Cluster Computing in Zero Knowledge. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II (Lecture Notes in Computer Science)*, Elisabeth Oswald and Marc Fischlin (Eds.), Vol. 9057. Springer, 371–403. https://doi.org/10.1007/978-3-662-46803-6_13

[10] Adán Sánchez de Pedro Crespo and Luis Ivan Cuende García. 2016. Stampery Blockchain Timestamping Architecture (BTA). (2016). Available online: https://s3.amazonaws.com/stampery-cdn/docs/Stampery-BTA-v5-whitepaper.pdf.

[11] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. 1996. Collision-Free Hashing from Lattice Problems. *Electronic Colloquium on Computational Complexity (ECCC)* 3, 42 (1996). http://eccc.hpi-web.de/eccc-reports/1996/TR96-042/index.html

[12] Shafi Goldwasser and Sunoo Park. 2017. Public Accountability vs. Secret Laws: Can They Coexist?: A Cryptographic Proposal. In *Proceedings of the 2017 on*

Workshop on Privacy in the Electronic Society, Dallas, TX, USA, October 30 - November 3, 2017, Bhavani M. Thuraisingham and Adam J. Lee (Eds.). ACM, 99–110. https://doi.org/10.1145/3139550.3139565

[13] House Energy and Commerce Committee. 2016. Deciphering the Debate Over Encryption: Industry and Law Enforcement Perspectives. (April 2016). https://energycommerce.house.gov/hearings-and-votes/hearings/deciphering-debate-over-encryption-industry-and-law-enforcement.

[14] Dia Kayyali. 2014. What You Need to Know About the FISA Court – and How it Needs to Change. (August 2014). https://www.eff.org/deeplinks/2014/08/what-you-need-know-about-fisa-court-and-how-it-needs-change.

[15] Ahmed E. Kosba, Zhichao Zhao, Andrew Miller, Yi Qian, Hubert Chan, Charalampos Papamanthou, Rafael Pass, Abhi Shelat, and Elaine Shi. 2015. How to Use SNARKs in Universally Composable Protocols. *IACR Cryptology ePrint Archive* 2015 (2015), 1093. http://eprint.iacr.org/2015/1093

[16] William Li, Pablo Azar, David Larochelle, Phil Hill, and Andrew W. Lo. 2015. Law Is Code: A Software Engineering Approach to Analyzing the United States Code. *Journal of Business and Technology Law* 10 (2015). Issue 2.

[17] Eric Lichtblau. 2013. In Secret, Court Vastly Broadens Powers of N.S.A. (2013). http://www.nytimes.com/2013/07/07/us/in-secret-court-vastly-broadens-powers-of-nsa.html.

[18] Sunoo Park, Albert Kwon, Joël Alwen, Georg Fuchsbauer, Peter Gaži, and Krzysztof Pietrzak. 2015. SpaceMint: A Cryptocurrency Based on Proofs of Space. Cryptology ePrint Archive, Report 2015/528. (2015). http://eprint.iacr.org/2015/528.

[19] Torben P. Pedersen. 1991. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings (Lecture Notes in Computer Science)*, Joan Feigenbaum (Ed.), Vol. 576. Springer, 129–140. https://doi.org/10.1007/3-540-46766-1_9

[20] Dakota S. Rudesill. 2015. Coming to Terms with Secret Law. (November 2015). Harvard National Security Journal 241 (2015); Ohio State Public Law Working Paper No. 321. Available at SSRN: https://ssrn.com/abstract=2687223.

[21] David A. Schulz, Hannah Bloch-Wehba, John Langford, Patrick Toomey, Brett Max Kaufman, Alex Abdo, Arthur B. Spitzer, and Scott Michelman. 2016. Motion of the American Civil Liberties Union for the Release of Court Records. (October 2016). Original document at: https://www.aclu.org/legal-document/aclu-motion-filed-foreign-intelligence-surveillance-court-fisc-requesting-release. Associated blog post at: https://www.aclu.org/blog/speak-freely/constitution-leaves-no-room-secret-law.

[22] Edward Snowden. 2013. Snowden Surveillance Archive. (2013). Archive of documents leaked by Edward Snowden, maintained by Canadian Journalists for Free Expression and the Politics of Surveillance Project at the Faculty of Information at the University of Toronto. Available at: https://snowdenarchive.cjfe.org.

[23] Peter Todd. 2016. Preventing Consensus Fraud with Commitments and Single-Use-Seals. (December 2016). https://petertodd.org/2016/commitments-and-single-use-seals.

[24] Madars Virza. 2016. (December 2016). Private communication.

[25] Madars Virza. 2017. (November 2017). Private communication.

[26] Daniel J. Weitzner. 2016. Testimony of Daniel J. Weitzner. (April 2016). http://docs.house.gov/meetings/IF/IF02/20160419/104812/HHRG-114-IF02-Wstate-WeitznerD-20160419.pdf.

[27] Zooko Wilcox. 2016. The Design of the Ceremony. (October 2016). https://z.cash/blog/the-design-of-the-ceremony.html.

A MORE DISCUSSION OF RELATION TO EXISTING SERVICES

A number of websites offer a timestamping service on the Bitcoin blockchain (and some also on other blockchains, notably that of Ethereum), wherein users submit documents of their choice to be timestamped, upon which the service creates a transaction containing the hash of the document and adds it to the Bitcoin blockchain. Some of these services[21] embed the hash in the transaction by setting it as the "recipient address" of a tiny monetary transfer. However, repurposing the recipient field in this way is frowned upon because it bloats the unspent transaction output (UTXO) database with bogus addresses; it is preferable to use a special-purpose transaction type which can store arbitrary data without "pretending" to be a monetary transfer. Bitcoin has a special transaction type called OP_RETURN which can serve this purpose. Numerous existing services[22] use Bitcoin's OP_RETURN transaction to store hashes of data on the blockchain. Moreover, many of these services use Merkle trees to generate a single succinct hash of large quantities of data, often aggregated across different users:[23] this is an important measure given the limited throughput of the blockchain. Merkle trees are a basic and versatile cryptographic tool that will be an important in our constructions too.

While many of the existing services offer a simple service allowing users to timestamp any data of their choice, some other services provide more complex frameworks for timestamped data management with the aim of selling their services as a generic[24] or special-purpose[25] data management solution to organizations. Features marketed include "proof of existence"; assurance of data "privacy", "integrity", "attribution", and "auditability"; and "scalability" to large volumes of data. We are not aware of existing proposals which put forward an inter-organizational regulatory framework (like our work), instead of targeting data management *within* organizations as do the above examples. The desiderata for the two cases overlap but have important differences: we prioritize the design of a framework that can be compatible with the interests of all parties involved, and incorporate the regulatory framework as part of the protocol, which we then prove satisfies well-motivated security definitions. Moreover, latency is not a limiting concern for our setting as we consider the aggregation of records over long periods of time, whereas many of the data management offerings consider it essential to have latency much faster than the growth rate of the blockchain [10], and that is one of the main technical challenges (perhaps *the* main one) that their systems aim to address.

In contrast, the primary concern in our setting (unaddressed by existing solutions) public auditability of secret information to enable proof of compliance with arbitrary regulations, even in cases where the regulations themselves are kept secret. To our knowledge, the auditability claimed by existing systems refers only to audits where the underlying data is revealed in an audit; we are not aware of any scheme that targets privacy-preserving, publicly verifiable audits, which constitute an integral feature of our system.[26]

[21] E.g., BitcoinProof, OriginStamp, SatoshiProof, among others.

[22] E.g., Bitproof, Eternity Wall, Factom, OpenTimestamps, Stampery, and others.

[23] We remark that sharing a single Merkle hash across multiple users can violate data privacy if done naively. This can be acceptable in cases where the timestamped data is intended to be public; and in other cases, some services appear to address the problem by encrypting the data before hashing it.

[24] E.g., Factom, Stampery, Tierion, among others.

[25] E.g., Dipl.me specializes in educational diplomas.

[26] We note that there do exist schemes that involve proving properties of certain secret data for specific applications, such as anonymous currency transfer (Zcash [3]) or preventing consensus fraud ([23]). These address substantially different use cases from our scheme, and moreover, they do not consider settings where the statement to be proven is itself secret.