# Design and Usage of Transparency Enhancing Technologies

*Alexander Hicks*

A dissertation submitted in partial fulfillment

of the requirements for the degree of

**Doctor of Philosophy**

of

**University College London**.

Department of Computer Science

University College London

December 6, 2023

I, Alexander Hicks, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

# Abstract

As computer systems handle transactions and make decisions about our lives, the opaqueness of such systems means that when faults occur, the negative impacts of these faults are often unfairly passed on to the individuals that are subject to these systems. The aim of this thesis is to show how transparency enhancing technologies, technical mechanisms that support transparency about a system, can be used to address these issues. To do this, this thesis provides an analysis of transparency enhancing technologies from their technical design to their practical usage.

The first part of this thesis systematizes transparency enhancing technologies, providing an analysis of the threat models and technical mechanisms that can support transparency.

The second part of this thesis gives an example of a transparency enhancing technology designed to provide transparency that enables publicly verifiable audits of the kind of access to data requests that are made by, for example, law enforcement and medical practitioners. This work shows how distributed logs can be used to provide the infrastructure necessary for such audits and how publicly verifiable audits in the form of statistics (including multivariable statistics) can be produced with assurances that the privacy of individuals who relate to the information that is audited is respected.

The third part of this thesis discusses how transparency enhancing technologies can allow us to move beyond simply transparency and accountability. Drawing on the legal literature that has concerned itself with the legal effects of code as it enforces norms and the legitimacy of such effects, I discuss the ways in which code can be contested through the use of transparency enhancing technologies, in contrast

to other kinds of accountability technologies based on compliance, and, because it is code itself, how transparency enhancing technologies can be legitimate, unlike other ways of contesting code.

# Impact Statement

By showing how transparency enhancing technologies can be designed and implemented, and put to use, this thesis on transparency has the potential to shape how systems are designed, contested, and how system operators are held accountable for flaws in the system they operate and the harm that this can produce.

Some of the work in this thesis has been published in reputable academic venues, and the work that is the basis for Chapter 5 has been recognized by an award[1] determined by a panel spanning academia, industry, and public bodies.

[1] See https://www.ucl.ac.uk/computer-science/news/2018/jul/alexander-hicks-wins-ace-csr-pitch-competition

# Acknowledgements

First, I must of course thank Steven Murdoch for enabling me to do this thesis, as well as Wai Yi Feng who first suggested that I apply for this particular position. I am also grateful for the feedback I received from George Danezis, Sarah Meiklejohn, and Marie Vasek, as part of my first year viva and transfer viva. Administrative affairs are an unavoidable frustration these days, but I want to thank Dawn Bailey, Sarah Bentley, and Wendy Richards, who made processes simpler when others made them complicated.

I also want to thank Mustafa Al-Bassam, Sarah Azouvi, Guy Goren, Lioba Heimbach, Vasilios Mavroudis, Patrick McCorry, and Sarah Meiklejohn, for being my co-authors on the papers that were produced over the time it took to complete this thesis.

Pre-pandemic, it was also possible to get to know UCL colleagues better, by sharing an office, teaching together, and seeing them outside of work. I was fortunate to share some time with Tristan Caulfield, Lisa Chalaguine, George Kappos, Mary Maller, Enrico Mariconti, Bristena Oprisanu, Ania Piotrowska, Maria Schett, Alberto Sonnino, Mark Warner, and Haaroon Yousaf. Sorry if I have forgotten anyone!

Finally, I want to thank my wife Sarah Azouvi for supporting the completion of this thesis, I'm looking forward to better times with you.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Computational systems are ubiquitous in nearly all aspects of our lives. They handle financial transactions, communications, exchange data about us and for us, and are used to make or assist in making decisions that impact our lives.

As is the case with any system, things do not always go well. Computational systems can be used in a setting for which they were not designed or evaluated. Their specification can be flawed, ignoring edge cases or imposing faulty logic that discriminates against certain users. The programs they execute can be affected by bugs resulting from implementation errors.

Despite the fact that faults are practically inevitable, their impact is typically restricted to the individuals directly affected by the faults in the system, rather than to those that are responsible for these faults, who may avoid liability by relying on opaque systems and the presumption that the outputs of computational systems are correct. Moreover, because computational systems are primarily designed and operated by a small minority of states and private companies with little diversity (i.e., rich white men), their negative impacts typically fall upon marginalized people who do not have much power to contest the outputs of the systems they are subject to, sometimes without consent.

There are numerous examples of harms that can result from this, due both to legacy systems that have been computerized (e.g., surveillance, data sharing, payment systems, ...) and new technology that has allowed these systems to operate at a greater scale (e.g., algorithmic decision making systems). A payment system

can allow fraud to happen at the cost of the victim rather than the system operator, and ban legitimate users because of errors or deliberate over moderation. Law enforcement can access increasing amounts of sensitive data with limited oversight. Companies can obtain highly sensitive data without the consent of the people whose medical data they obtain and collate data from an increasing amount of sources.

This thesis evaluates the hypothesis that part of the solution to these problems is to make computational systems more transparent, based on the following intuition. The harms we have introduced above are the result of flaws in the system, whether that be the design, implementation, or operation of the system. The fact that these flaws affect individuals that have no (or limited) privileges over the system creates a classic moral hazard issue. One way of resolving this situation is to make it easier for those that may be victims of these errors in the system is to allow them to better evaluate the system as a whole, whether there has been an error that affected them, and to be able to contest the system and challenge the effects on the flaw that has affected them.

Thus, this thesis is also about power, because these issues revolve around the power that systems have over people that are subject to these systems, and therefore, the disproportionate power that those with privileged access to these systems have over those who do not. Because of the economic structure that determines how and who builds the system we are subject to, this power imbalance reflects the power imbalances that exist throughout society. Research in information security has always dealt with issues of power over a system, typically by ensuring privilege over a system (e.g., through access control mechanisms). This thesis considers the opposite approach.

## 1.1 Scope and Organization of The Thesis

Transparency is a wide-ranging subject so the scope of this thesis is necessarily restricted to specific forms of transparency, specifically transparency enhancing technologies that are based on distributed or decentralized cryptographic logs. The focus is, therefore, the analysis and design of technical mechanisms, in particular logging

mechanisms and privacy mechanisms, which can be used to support transparency, and how transparency can be put to use to contest systems and hold system operators accountable.

Chapter 2 introduces the primitives discussed and used in this thesis, while Chapter 3 outlines work that is related to the content of this thesis.

Chapter 4 presents a systematization of transparency enhancing technologies from a security point of view. This chapter systematizes log based transparency enhancing technologies. Based on established work on transparency from multiple disciplines we outline the purpose, usefulness, and pitfalls of transparency. We outline the mechanisms that allow log based transparency enhancing technologies to be implemented, in particular logging mechanisms, sanitization mechanisms and the trade-offs with privacy, data release and query mechanisms, and how transparency relates to the external mechanisms that can provide the ability to contest a system and hold system operators accountable. We illustrate the role these mechanisms play with two case studies, Certificate Transparency and cryptocurrencies, and show the role that transparency plays in their function as well as the issues these systems face in delivering transparency.

In Chapter 5 we propose VAMS, a system that enables transparency for audits of access to data requests without compromising the privacy of parties in the system. VAMS supports audits on an aggregate level and an individual level, by relying on three mechanisms. A tamper-evident log provides integrity for the log entries that are audited. A tagging scheme allows users to query log entries that relate to them, without allowing others to do so. MultiBallot, a novel extension of the ThreeBallot voting scheme, is used to generate a synthetic dataset that can be used to publicly verify published statistics with a low expected privacy loss. We evaluate two implementations of VAMS, and show that both the log and the ability to verify published statistics are practical for realistic use cases such as access to healthcare records and law enforcement access to communications records.

Chapter 6 branches out from a strict security point of view and addresses the role that transparency enhancing technologies can play in situations, such as legal

disputes, where it is necessary to contest the outputs of a system to which one is subject. Both technical security mechanisms and legal processes serve as mechanisms to deal with misbehaviour according to a set of norms. While they share general similarities, there are also clear differences in how they are defined, act, and the effect they have on subjects. This chapter considers the similarities and differences between both types of mechanisms as ways of dealing with misbehaviour, and where they interact with each other.

Taking into consideration the idea of code as law, we discuss accountability mechanisms for code, and how they must relate to both security principles and legal principles. In particular, we identify the ability to contest norms enforced by code as an important part of accountability in this context. Based on this analysis, we make the case for transparency enhancing technologies as security mechanisms that can support legal processes, in contrast to other types of accountability mechanisms for code. We illustrate this through two examples based on recent court cases that involved Post Office in the United Kingdom and Uber in the Netherlands, and discuss some practical considerations.

Finally, Chapter 7 summarizes and discusses the results presented in this thesis, outlines directions for future work, and ends with final thoughts transparency enhancing technologies.

## 1.2 Publications Resulting From This Thesis Work

Several papers have been produced over the course of working on this thesis. Some of these papers form the basis for Chapters 4, 5, and 6, while others are not included in this thesis because they are not directly related to the topic we explore here.

### 1.2.1 Papers used in this thesis

- Alexander Hicks, *Log Based Transparency Enhancing Technologies*, available as a preprint on arXiv [1], is used as the basis of Chapter 4.

- Alexander Hicks, Vasilios Mavroudis, Mustafa Al-Bassam, Sarah Meiklejohn, Steven J. Murdoch. *VAMS: Transparency for Audits of Access to Data*, available as a preprint on arXiv [2], is used as the basis for Chapter 5.

- Alexander Hicks, *Transparency, Compliance, And Contestability When Code Is(n't) Law* [3], which appeared at the New Security Paradigms Workshop (NSPW) in 2022, is the basis for Chapter 6.

### 1.2.2 Papers not included in this thesis

- Sarah Azouvi, Guy Goren, Lioba Heimbach, Alexander Hicks. *Base Fee Manipulation In Ethereum's EIP-1559 Transaction Fee Mechanism*, which will appear at the International Symposium on Distributed Computing in 2023 and is available on arXiv as a preprint [4]. This paper provides an analysis of the base fee in Ethereum's EIP1599 transaction fee mechanism, showing that there exists a rational deviation from the honest mining strategy to optimize revenue from the base fee by lowering it by publishing empty blocks.

- Sarah Azouvi, Alexander Hicks. *Decentralisation Conscious Players And System Reliability*, which appeared at the Financial Cryptography and Data Security conference in 2022 [5]. This paper presents a game theoretic model of decentralized systems as public goods and examines the role of decentralization conscious players; that is, players who prioritize some level of decentralization in the contributions to the system, in maintaining decentralization.

- Sarah Azouvi, Alexander Hicks. *Tools for Game Theoretic Models of Security for Cryptocurrencies*, which appeared at the Cryptoeconomic Systems conference in 2020 [6]. The paper systematizes existing work on algorithmic mechanism design and game-theoretic models of cryptographic and distributed systems security in the context of cryptocurrencies.

- Alexander Hicks, Steven J. Murdoch. *Transparency Enhancing Technologies to Make Security Protocols Work for Humans*, which appeared at the Security Protocols workshop in 2019 [7]. This paper looks at how the dispute resolution process around computer systems, such as the Post Office disputes, could be improved through the use of transparency enhancing technologies.

- Sarah Azouvi, Alexander Hicks, Steven Murdoch. *Incentives in Security Pro-*

*tocols*, which appeared at the Security Protocols workshop in 2018 [8]. The paper discusses the still undervalued role that incentives play in the real world security of protocols, based on examples from payment systems (EMV), cryptocurrencies, and anonymity networks, and how this could be addressed through the use of security models that explicitly take these into account.

- Patrick McCorry, Alexander Hicks, Sarah Meiklejohn. *Smart Contracts for Bribing Miners*, which appeared at the Bitcoin workshop in 2018 [9]. The paper introduces smart contracts that can be used by a malicious actor to bribe miners of a cryptocurrency to manipulate the underlying blockchain as well as attack another cryptocurrency, without the miners having to trust the briber.

## 1.3 Work Done in Collaboration

As a result of being based on a collaborative paper, Chapter 5 includes work that was done with other researchers. Steven Murdoch and Sarah Meiklejohn suggested the law enforcement and healthcare use cases, respectively, and provided some guidance throughout the completion of the paper. Vasilios Mavroudis first suggested the use of ThreeBallot which he and I then reworked to produce MultiBallot, with Vasilios Mavroudis focusing on the implementation of the scheme and the literature on verifiable statistics covered in Chapter 3, while I worked on the theorems that determined the bounds on the expected privacy loss and ballot reconstruction attacks. The Trillian based implementation of VAMS was done by Mustafa Al-Bassam, while I did the Hyperledger Fabric based implementation. The pseudonymous identifier scheme was designed in a meeting between Sarah Meiklejohn and myself.

The remaining chapters are solely the result of my work.

# Chapter 2

# Technical Primitives

This chapter introduces the technical primitives that we rely on or discuss throughout this thesis. Because these are well-established primitives and this thesis does not include work on these primitives, we choose to omit the most precise mathematical definitions in favour of readability for non-cryptographers. These can be found in freely available textbooks such as the draft of Boneh and Shoup's applied cryptography textbook [10] and Dwork and Roth's book on Differential Privacy [11].

## 2.1 Cryptographic Hash Functions

A hash function $h$ maps an arbitrarily sized input to a fixed length output that is referred to as the hash value (alternatively, the message digest) of that input.

In this thesis, we use the term hash function to refer exclusively to cryptographic hash functions, which are hash functions for which three security properties hold.

1. First pre-image resistance, which requires that given $h(x)$ an adversary cannot determine $x$ with non-negligible probability.

2. Second pre-image resistance, which requires that given an input $x$ an adversary cannot efficiently find another input $y$ such that $h(x) = h(y)$.

3. Collision resistance, which requires that an adversary cannot efficiently find two inputs $x$ and $y$ such that $h(x) = h(y)$. (This implies second pre-image resistance.)

The only case in which we refer to a specific hash function in this thesis is in Chapter 5 where we use the SHA256 hash function in the implementation of VAMS, which is standardized by NIST [12].

## 2.2 Public Key Cryptography

Public key cryptography is a set of cryptographic mechanisms that allow users to have a public key, by which they can be known to other users, as well as a private key that only they know and must keep secret, therefore removing the need for a pre-established shared symmetric key. (Public key cryptography is sometimes used as a way to establish a symmetric key.) In this thesis the systems we discuss require the use of public key encryption, digital signatures, and in some cases, zero-knowledge proofs, as well as public key infrastructures.

### 2.2.1 Public key encryption

Public key encryption allows parties to exchange encrypted messages without the need for a shared key.

More formally, a public key encryption scheme is a triplet of three algorithms.

1. A probabilistic key generation algorithm $G$ that takes no input and outputs a pair $(pk, sk)$, where $sk$ is called a secret key (sometimes simply private key) and $pk$ is called a public key.

2. A probabilistic encryption algorithm $E$ that taking as input a message $m$ and a public key $pk$ produces ciphertext $c$.

3. A deterministic decryption algorithm $D$ that takes as input a secret key $sk$, a ciphertext $c$, and returns the message $m$ that was encrypted using the corresponding public key, or a reject value distinct from all possible messages.

For a public key encryption scheme to be secure, it must be the case that an adversary that eavesdrops on encrypted communications between parties that encrypted their messages with a public key encryption scheme cannot learn anything about the messages that are exchanged.

### 2.2.2 Digital Signatures

Cryptographic digital signatures allow a party to sign a message with their private key and anyone with knowledge of that party's corresponding public key to verify the signature on that message and, therefore, authenticate the party that signed the message.

More formally, a digital signature scheme is a triplet of three algorithms.

1. A probabilistic key generation algorithm $G$ that takes no input and outputs a pair $(pk, sk)$, where $sk$ is called a secret signing key (sometimes simply private key) and $pk$ is called a public verification key (sometimes simply public key).

2. A probabilistic signing algorithm $S$ that taking as input the secret key $sk$ and a message $m$ produces a signature $\sigma$.

3. A deterministic verification algorithm $V$ that takes as input a public key $pk$, a message $m$, and a signature $\sigma$ then outputs $accept$ if the signature is valid (i.e., generated for the message $m$ using $V$ with the corresponding secret signing key) or $reject$.

The important security property for digital signature schemes is that it should be impossible for an adversary to forge a signature on a message. This also means that given a signature on a message, the party that signed the message cannot later claim that the message did not originate from them unless they argue that their secret signing key was compromised. This property is referred to as non-repudiation.

In the UK, a recent project by the Law Commission (referring to cryptographic digital signatures as electronic signatures) confirmed that "an electronic signature is admissible in evidence in legal proceedings" and "is admissible, for example, to prove or disprove the identity of a signatory and/or the signatory's intention to authenticate the document" [13].

### 2.2.3 Zero Knowledge Proofs

Zero-knowledge proofs are cryptographic tools that give a prover the ability to prove a statement without revealing anything but the truth of that statement. Introduced

by Goldwasser et al. [14], zero-knowledge proof techniques are popular because of the optimal trade-offs in confidentiality and utility that they seem to offer.

Confidentiality levels are of course very high given that nothing but the truth of the statement is revealed, and high levels of utility are assured by the completeness and soundness of the proof, which guarantees that if the statement is true then it is easy to verify that it is true and if the statement is false it is impossible to convince an honest person that it is true.

An important result for zero-knowledge proofs due to Goldreich et al. [15] is that any statement in a language in NP (i.e., a decision problem whose positive answers have proofs verifiable in polynomial time) can be proven in zero-knowledge. This implies that for a protocol that does something in a non-privacy-preserving way, it may be replaced by a zero-knowledge protocol that will be privacy-preserving.

The work of Eskandarian et al. [16] that proposes a way for certificate transparency (a system that logs SSL certificates) to allow users to report invalid certificates without compromising privacy (i.e., revealing browsing behaviour) is an example of this. Another example is the work of Frankle et al. [17] and Park and Goldwasser [18] that proposes using zero-knowledge proofs to show that confidential legal proceedings followed the correct legal procedures. More generally, zero-knowledge proofs offer the ability to efficiently verify information that may be suitable for accountability in legal settings [19, 20].

A zero-knowledge proof systems is said to be transparent if it does not require a trusted setup. Some zero-knowledge proof systems are not transparent such as the zk-SNARK construction originally used by the cryptocurrency Zcash, meaning that if the setup ceremony was compromised (despite extensive documentation [21]) then it would have been possible for the perpetrators to forge proofs and effectively generate additional units of Zcash in an undetectable manner. More recently, however, several ways of avoiding trusted setups without the need for complicated ceremonies have been introduced [22, 23, 24, 25, 26, 27, 28].

### 2.2.4 Public key infrastructure

The use of public key cryptography can entail the need for a public key infrastructure, which manages how identities are bound to public keys. In practice, this means that there are certificate authorities, designated trusted third parties, that verify the identity of a user and their possession of a set of cryptographic keys and produce a signed certificate (typically in the X.509 standard) that publicly associates the user with their public key.

## 2.3 Transparency Overlays

Following Chase and Meiklejohn [29], we use the term transparency overlay to broadly refer to any kind of distributed log that guarantees tamper evidence using cryptographic mechanisms. The term auditable data structures is sometimes also used, for example by Goodrich et al. [30], as well as the term authenticated data structures [31, 32].

By tamper evidence, we mean that any alteration to information on the log is detectable because changes to the log are recorded and their integrity is guaranteed by cryptographic mechanisms, and that the party responsible for tampering with the log can be held accountable because they will have signed the operation that resulted in the change to the log.

In this thesis, we rely on two instantiations of transparency overlays to implement VAMS in Chapter 5, Trillian which is based on Merkle trees and verifiable log-backed maps, and Hyperledger Fabric which is a permissioned distributed ledger with an underlying blockchain. However, we also refer to transparency overlays in general in other chapters.

### 2.3.1 Merkle trees and verifiable log backed maps

A Merkle tree, based on the work of Merkle [33], is a hash-based binary tree structure illustrated in Figure 2.1.

Every leaf in the tree corresponds to some data, and every non-leaf node up to the root node is the hash of its two child nodes. Given a hash function $h$, node $i$ stores the hash $h_i = h(h_{left(i)} \| h_{right(i)})$ based on its left and right children. Changing

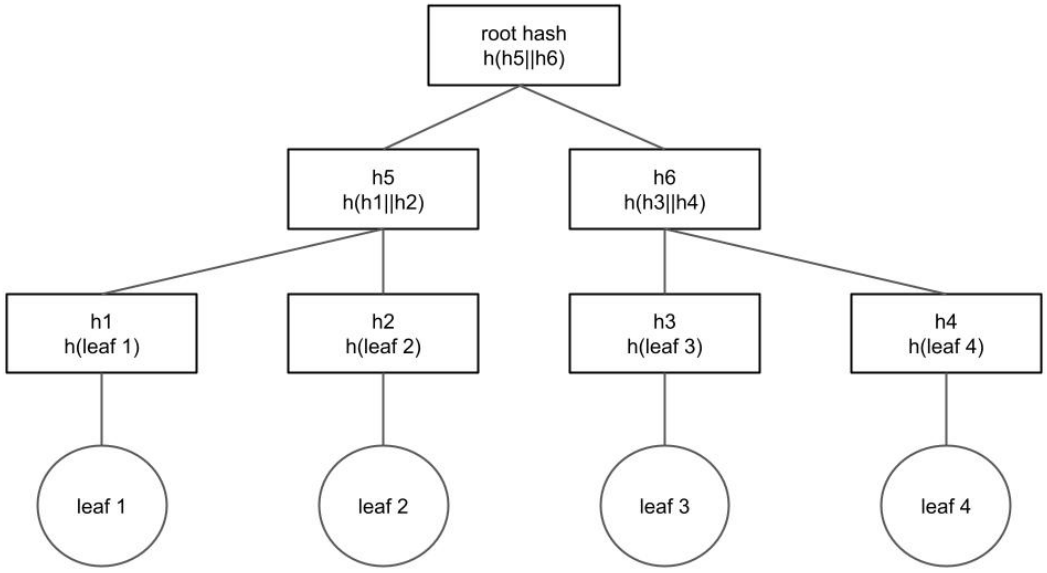


**Figure 2.1:** A Merkle tree with four leaf nodes.

a piece of data (i.e., a leaf) therefore changes every node of the tree's branch from the leaf to the root.

Assuming that the underlying hash function is collision resistant, Merkle trees can provide an efficient way of verifying the integrity of data that has been encoded in the tree (i.e., the value of a leaf) because one only needs to know the subset of nodes that, together with the leaf, are needed to reconstruct the root hash and verify that it matches the root hash of the tree. For a Merkle tree with $n$ leaves, this *Merkle proof* is of size $O(\log(n))$.

Looking at Figure 2.1, for example, given $h2$ and $h6$ we can verify that the value of $leaf$ 1 is included in the tree with root hash $h(h5||h6)$.

When a Merkle tree is updated, it is possible for the server hosting the Merkle tree to maintain a link with the previous Merkle tree by committing to a chain of signed tree roots where each new signed tree root references the hash of the previous signed tree root, as illustrated in Figure 2.2.

To check that two successive trees are consistent with each other, such that every value included in the past tree is included in the new tree, one needs to show that the present tree contains the past tree and that given the past tree, adding the leaves that have been added would result in the new tree.

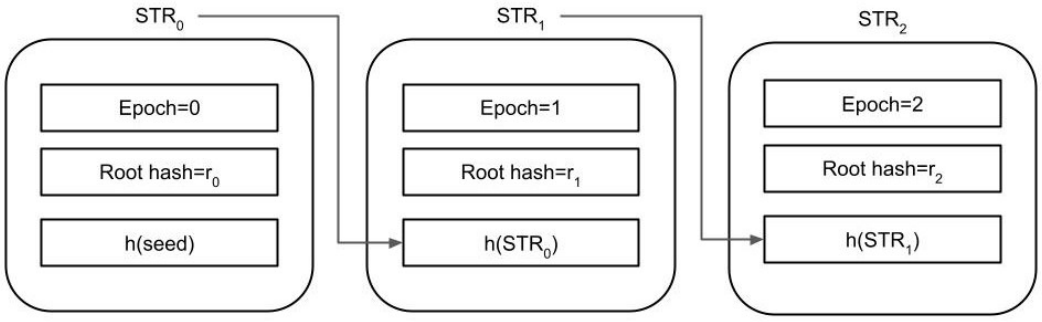


**Figure 2.2:** A chain of signed tree roots over three epochs. The first epoch is initialized with a seed as there is no previous root hash.

Looking again at Figure 2.1, if we take the past tree as being the tree containing only leaves 1, 2, and 3, and the new tree to result from the addition of leaf 4, then we can show the following given $h5$ and $h3$ from the past tree. The new tree contains the past tree because we can recompute the previous root hash $h(h5||h3)$. Adding $leaf\ 4$ to the past tree does result in the new root hash $h(h5||h(h3||h4))$ where $h(h3||h4))$ has replaced $h3$ in the past root hash.

### 2.3.2 Distributed Ledgers

Introduced in the context of Bitcoin [34], blockchains have now been used more widely for cryptocurrencies and other applications built on top of cryptocurrencies like Ethereum [35] that support *smart contracts* which can be used to execute arbitrary code.

A blockchain is a chain of *blocks* that are cryptographically linked to each other by having each block include a reference to the hash of the previous block. This means that it is not possible to modify past blocks in the blockchain as the hash values that link each block together will no longer be consistent. This makes the blockchain *append-only*, while consensus rules ensure that all nodes extend the same blockchain.

Blockchains use Merkle trees to store data (e.g., transactions) within each block, thus verifying the inclusion of data in a blockchain amounts to verifying its inclusion in a Merkle tree in a block. This makes systems both on Merkle trees or blockchains similar in the security guarantees they provide.

The important differences between the use of a blockchain rather than Merkle

trees are, therefore, the use of a consensus protocol rather than gossiping to ensure consistency (i.e., a global state agreed on by all nodes), and the ability to record the execution of smart contracts rather than just store data.

Blockchains can also be *permissioned* or *permissionless*, which determines the kind of consensus protocol that can be used.

Permissionless blockchains (e.g., Bitcoin) are open to anyone willing to participate (i.e., ay the cost of participating) in the system. This requires mechanisms that defend against Sybil attacks, which consist of an adversary generating many fake users controlled by the adversary to take control of the system. Sybil resistance mechanisms such as Proof-of-Work, Proof-of-Stake, Proof-of-Space, and so on, rely on users to commit some amount of a finite, difficult enough to obtain resources to perform actions in the system such as validating transactions. Permissionless blockchains also require consensus protocols that can incorporate these sybil resistance mechanisms scale to hundreds or thousands of nodes [36].

Permissioned blockchains alleviate the need for potentially wasteful sybil resistance mechanisms such as proof-of-work by requiring participants in the system to be known and agreed upon, making participation exclusive. This makes it possible to rely on more traditional and efficient consensus protocols.

## 2.4 Inference Control

Inference control refers to the set of mechanisms designed to limit how much can be inferred about an individual (or group) from data without sacrificing the utility of the data in learning about a population. Generally speaking, if data is released for a purpose, learning anything more from the data than what is intended by its release is a bad outcome. Although such concerns have existed since at least the seventies [37], database reconstruction and private attribute inference attacks continue to be shown to be practical [38, 39, 40, 41, 42, 43, 44].

A variety of approaches have been tried to deal with this, including statistical disclosure control mechanisms such as k-anonymity [45], l-diversity [46], t-closeness [47] and $\rho$-uncertainty [48], which do not achieve acceptable trade-offs

between privacy and utility [49].

### 2.4.1 Differential privacy

Introduced by Dwork et al. [50], differential privacy quantifies the impact that a single point in a dataset has on the distribution of the output of a randomized mechanism. Thus, it is a definition that applies to mechanisms, rather than data, and guarantees that the privacy loss associated with participating in (for example) a study is bounded by a known quantity without making assumptions about an adversary's knowledge. Definition 1 formally expresses this.

*Definition* 1 ($(\varepsilon, \delta)$-Differential Privacy [11]). A randomized algorithm $M$ with domain $\mathbb{N}^{|\chi|}$ is $(\varepsilon, \delta)$-differentially private if for all $S \subseteq Range(M)$ and for all $x, y \in \mathbb{N}^{|\chi|}$ such that $\|x - y\| \leq 1$:

$$\Pr[M(x) \in S] \leq \exp(\varepsilon) \Pr[M(x) \in S] + \delta. \tag{2.4.1}$$

This definition has several advantages. It is flexible, in the sense that the parameters $\varepsilon$ and $\delta$ can be tuned in order to balance trade-offs between utility and privacy.

It composes in a simple way with itself, so that the composition of two differential private mechanisms produces another differentially private mechanism, albeit with greater values of $\varepsilon$ and $\delta$.

It also does not depend on assumptions about an adversary's background knowledge, unlike previous attempts at privacy-preserving mechanisms with similar goals such as k-anonymity and its variants [45, 51, 46]

There are, however, some limitations to differential privacy. It also does not assume anything about an adversary's prior knowledge of the data, although assumptions about the way data is generated do matter if any utility is to be preserved [52]. It also does not protect very well against inference [53]

Several expository and systematizations papers on differential privacy and its applications exist [11, 54, 55, 56, 57, 58]. Differential privacy has become an extremely popular primitive, both in academic research where it has led to a large

body of work but also in its real-world applications. Google [59], Apple [60], and Microsoft [61], are examples of large tech companies that have deployed differential privacy. Excluding large tech companies, the 2020 census in the United States has also relied on differential privacy [62, 63, 64], following its use for On-TheMap [65, 66]. From a legal and regulatory perspective, differential privacy has also been studied in relation to privacy laws and regulations, such as the European GDPR requirements [67, 68], showing that differential privacy may also be a suitable privacy mechanism from that perspective.

## 2.5 ThreeBallot

ThreeBallot [69, 70] is a paper-based voting scheme proposed by Rivest for end-to-end auditable elections. Voters are given three blank ballots arranged as three columns, each row corresponding to a single candidate. Marking two of the three columns in a row is a vote *for* a candidate, while marking only one of the columns is a non-vote. In the standard version of ThreeBallot, each row must have either one or two marks and blank rows or rows with three marks are not allowed.

After the voting process, the collection of all ballots is placed on a public bulletin board, so that anyone can verify the outcome of the elections and check if their vote was represented correctly (while keeping their vote private).

The scheme's security properties (i.e., auditability and vote privacy) have been extensively studied in various works [71, 70, 72, 73, 74, 75, 76]. From all the attacks introduced in the literature, the *reconstruction* and *pattern-based* attacks apply to our use case in Chapter 5. Henry et al. [73] published a thorough analysis of these attacks against two-candidate races, extending previous work by Strauss [75]. These works provide a lower bound for security as a function of the ballot size (number of binary choices).

In Chapter 5, we rework ThreeBallot, which we call MultiBallot as we generalize the scheme to any odd number of ballots, as a primitive to generate a privacy-preserving synthetic dataset that can be used to publicly verify the statistics computed on the original dataset.

# Chapter 3

# Related Work

## 3.1 Work related to Chapter 4

A number of past surveys related to transparency enhancing technologies exist. Murmann and Fischer-Hübner [77] focus on the usability of transparency enhancing technologies. Hedbom [78], Janic et al. [79], and Zimmermann [80] focus on transparency tools that can be used to help users control or verify their privacy online. Spagnuelo et al. [81, 82] look at transparency enhancing technologies in the context of providing and complying with the transparency required by the GDPR.

In contrast to these papers, our focus is not specifically on existing tools (although we survey some and consider two use cases), but more generally on how to design and build transparency enhancing technologies based on cryptographic logs under realistic threat models that consider issues of editorial control and access to individual evidence.

## 3.2 Work related to Chapter 5

The work closest to ours is due to Frankle et al. [17], who propose a system that allows accountability of secret legal processes using zero-knowledge proofs and aggregate statistics computed through a multi-party computation (MPC) between courts. Previously, Goldwasser and Park [18] had also proposed using append-only ledgers and zero-knowledge proofs in the context of actions related to secret laws under the U.S. Foreign Intelligence Surveillance Act.

This approach provides less transparency than ours as they do not support in-

dividual transparency, only aggregate statistics, thereby reducing the potential for users to contest outcomes [3]. While the outputs of zero-knowledge proofs and MPC can be checked for correctness, the integrity of inputs (i.e., the integrity of the data used in audits) cannot be verified. Because the inputs can be manipulated, they must assume that judges (which are closest to the auditors of our context) are not malicious and would not publish an inaccurate report, making their threat model weaker than ours. Their proposed systems are also specific to a targeted use case where all parties could coordinate to perform the required multiparty computation, while our approach is more generally applicable as parties do not require as much coordination aside from the initial request for data (which is unavoidable).

Work by Panwar et al. [83] also addresses the problem of auditing surveillance orders, but differs from ours to a greater extent as it envisions an *enforcer* that verifies the interactions between agents and data providers, which are recorded on a blockchain using zero knowledge proofs, but does not support verifiable statistics.

Tamper evident logs have also been used in other auditability focused work. Bates et al. [84] look at accountable logs of wiretapping in the context of equipment implementing requirements of the US Communications Assistance for Law Enforcement Act (CALEA). This system permits simple counting queries, whereas VAMS allows broader analysis. CONIKS [85] deals with the specific case of key transparency, allowing users to monitor their key bindings, and does not deal with other problems that we address, in particular public audits.

In terms of privacy preserving statistics, techniques such as k-anonymity [45], l-diversity [46], t-closeness [47] and $\rho$-uncertainty [48] have been proposed. As discussed by Domingo-Ferrer and Torra [49], however, these techniques provide privacy only when the utility of the dataset is significantly reduced, whereas our solution enables accurate statistics.

Another line of work that attempts to address this limitation is privacy-preserving association rule mining [86] (we introduce association rule mining in Section 5.5). Such techniques generate randomized or perturbed datasets that protect the privacy of users while preserving some of the associations between the

variables that are of interest. Originally, privacy-preserving association rule mining was performed through uniform randomization of the dataset based on a public factor. As shown by Evfimievski et al. [87] this naive approach does not protect the users' privacy effectively. They instead proposed *randomization operators*[87] that were also proven ineffective and require an initial dataset of at least one million records [88]. Zhang et al. proposed a scheme that considers the existing association rules when perturbing the data and as a result provides better privacy bounds [88]. Unfortunately, this scheme has limited applicability as it severely distorts the strength of the association rules, overestimating strong relationships and under-representing less frequent ones. Overall, the weak privacy guarantees and the poor accuracy achieved by those schemes make them unsuitable for a system like VAMS.

A more promising line of work is based on differential privacy [50]. Such schemes have been studied extensively in the past years and have been proven to be secure in a variety of settings [89]. However, they still impose trade-offs between privacy and utility [90], as well as one-shot and continuous observation [91]. Achieving a meaningful privacy parameter can also be hard in practice [92], particularly when the aim is to provide a general solution like ours. This problem is tackled by Chen et al. [93], who take into consideration the underlying dataset to provide stronger privacy guarantees and increased utility.

None of these solutions provide verifiability, however, so the public cannot easily verify the integrity of the published data or statistics. In fact, the analyst who adjusts the noise term may accidentally or intentionally sample from distributions that drastically skew the statistics computed [94]. Narayan et al. [94] solve this problem with a scheme that uses a subset of Fuzz [95] to generate publicly verifiable validity proofs. Unfortunately, VerDP has limited expressiveness and severely constrains access to the dataset. More specifically, once the privacy budget of a particular dataset gets depleted, no further queries or analysis can be conducted. This may exclude researchers from using the data and prevent the application of novel analysis techniques on older, depleted datasets. It could also allow a malicious party

to intentionally deplete the privacy budget. In comparison, we allow the data to be used any number of times and without constraints.

## 3.3 Work Related to Chapter 6

There is a vast amount of work concerned with accountability, much of which is covered in the systematization of the topic by Wieringa [96] that is based on Boven's framework for accountability [97]. Transparency itself is also the focus of work across many disciplines. The book by Taylor and Kelsey provides a useful overview of applications of transparency in various contexts across the world, how it can succeed and how it can have counterproductive results if badly implemented and vulnerable to either unverifiable information or an inability to act on information (i.e., missing contestability) [98].

Specifically related to this paper, there is work that focuses on security models for accountability [99, 100, 99], interactions between security mechanisms that can provide assurances and the legal system have also been studied previously [101, 102], as well as the production of evidence by systems [103].

Our work differs from this existing body of work by considering how accountability mechanisms, in particular transparency enhancing technologies, can be used to contest norms enforced by code when designed to support existing processes such as legal disputes, rather than the predominant focus on obtaining assurances of compliance with a norm.

More recent work does address contestability, such as the work of Lyons et al. [104] who, like us, consider the ability to contest via legal processes but focus on higher level design principles. Our work is complementary to theirs, approaching contestability from the perspective of digisprudence and discussing specific technical security mechanisms.

# Chapter 4

# Log Based Transparency Enhancing Technologies

## 4.1 Introduction

As systems perform operations and assist decisions that can have an important impact on a person's life, transparency is often suggested as a way of identifying flaws in a system, enabling accountability, and making it more likely that flaws are rectified and their impacts mitigated.

Transparency, however, is a complex property to require from a system. It does not entail any specific meaning or way of implementing transparency, particularly in systems deployed in an environment that is adversarial to the accountability that transparency should enable. What information is revealed? In what form? By who? To whom? How? As a result, transparency does not always work as desired and is sometimes even counterproductive [98].

In this chapter, we consider achieving transparency based on logging mechanisms. This involves technical considerations, such as logging, sanitising, releasing and querying data, as well as non-technical external mechanisms that determine what can be done once transparency is in place. Our aim is to provide a systematization that brings the relevant aspect of each mechanism into one view of log based transparency enhancing technologies.

### Outline of the chapter

We motivate applying transparency to computer systems and give an overview of transparency and criticisms of transparency in Section 4.2, before outlining log based transparency enhancing technologies based on four essential mechanisms in section 4.3: logging, sanitization, release and query, and external mechanisms.

In Section 4.4 we discuss threats to transparency mapped to the essential mechanisms outlined in the previous section and editorial control and individual evidence.

We consider the infrastructure that supports logging in Section 4.5 and the interaction between transparency and privacy in Section 4.6. To illustrate our discussion we provide in Section 4.7 two case studies of transparency systems, Certificate Transparency and cryptocurrencies.

### Methodology

Transparency is a broad topic that many fields have independently studied, not all of which can be covered here. For work on transparency from other fields, we have, therefore, focused on work from Law, Philosophy, Business, and Economics, which provide a basis for thinking about transparency and computer systems.

Because of our focus on log based transparency enhancing technologies and the security of the mechanisms involved in such systems, we have endeavoured to find relevant papers from the information security literature by going through publications at major conferences like IEEE S&P, ACM CCS, NDSS, Usenix Security, PETS, and ACM FAccT, as well as searching for papers from other smaller conferences, workshops, and journals, including those in adjacent fields (e.g., HCI, STS). Work that relates to transparency but not directly to log based transparency enhancing technologies (e.g., work on transparent machine learning) is out of scope and, therefore, not included.

## 4.2 A Short Overview of Transparency

Transparency can be defined as “the quality of being done in an open way without secrets” [105]. Applied to an organization, it can mean that the organization is

"open, public; having the property that theories and practices are publicly visible, thereby reducing the chance of corruption" [106].

These definitions express the basic intuition that if something is being done transparently then it cannot be done badly without it being noticeable. As Brandeis put it, "sunlight is said to be the best of disinfectants" [107].

This should create an incentive to ensure that things are done well if there is a high likelihood of being held to account, making transparency an enabler of accountability or other ethical principles (e.g., safety, welfare) [108].

Transparency such as open data practices promoted by both governments [109, 110] and academics [111, 112], lead to the public release of data that is used to determine policy. Open data practices are also used in scientific research to allow results to be reproduced, further research to be conducted, and new algorithms to be benchmarked.

In a more bottom-up manner, freedom of information laws have enabled the media, NGOs, and the public, to make requests for information that can be used to hold public authorities to account. Other regulations, such as the GDPR [113] give individuals the right to request a copy of their personal data that is held by a controller (Article 15), require that personal data should be "processed [...] in a transparent manner in relation to individuals" (Article 5), and as general data processing principles that "it should be transparent to natural persons that personal data concerning them are collected, used, consulted or otherwise processed and to what extent the personal data are or will be processed" and "the principle of transparency requires that any information and communication relating to the processing of those personal data be easily accessible and easy to understand, and that clear and plain language be used" (Recital 39).

Access to data also helps mitigate information asymmetries. The work of Akerlof on information asymmetries in markets [114] has led to security economics re-framing many security issues (e.g., software security) as problems of information asymmetry [115, 116], which can be dealt with by requiring data standards and disclosures.

This ties into Saltzer and Schroeder's open design principle [117]. Most security mechanisms (e.g., cryptographic algorithms) are open, enabling users with the technical knowledge required to assess a system's code or specification to determine whether they want to rely on the system. Beyond the specification and code of a system, nutrition labels for datasets [118, 119, 120] and models [121], and privacy labels [122, 123], have been proposed.

### 4.2.1 Transparency matters for computer systems

Security mechanisms are designed to allow certain properties of systems (or the data they operate on) to hold; for example, integrity, confidentiality, or availability, but no system is perfect. However, designs can be flawed, implementations can suffer from software bugs or faulty hardware, and systems can be misused. Security Economics tells us that we should not expect perfect security in practice, even when technical mechanisms appear to be sufficient in principle [116, 115].

Even if we perfected the design of systems, designing and implementing complex systems that are entirely formally verified is currently unrealistic and would not prevent harms that occur because of a system that, operating as intended, applies harmful norms [3]. Information is routinely copied, aggregated, and analysed across networks operated by different parties, rendering strict enforcement mechanisms impractical compared to relying on accountability [124]. Notions of appropriate use may depend on the data itself as well as the context – an emergency that requires immediate access to medical data would render any strict security mechanism preventing this useless [100].

More generally, evaluating strict compliance with norms assumes that there are reliable norms, despite many systems operating in grey areas [3]. As systems grow in size, complexity, and scope of applications that impact people's lives, the ability to evaluate systems is increasingly important, not only for auditors or regulators but also for users who may change how they interact with the system [125].

Evaluating systems is not new, and system operators routinely do so internally but this does not always work to reduce the harm that a faulty system can cause. There can be issues with how the evaluation is done; for example, because of flawed

mechanisms or metrics. Even if a system operator detects faults in the system it operates, it still has to address these faults and may not do so if it does not have the incentive or the capacity (technical or economic) to do so.

Systems are not inherently inscrutable [126], but those to whom harm is caused cannot necessarily detect or show that the system is at fault, despite being those that have a greater incentive to do so. Access control mechanisms that regulate rights over a system tend to favour those who design or commission these access control mechanisms (e.g., system operators), rather than those subject to the system who have no ability to access useful information via the system itself.

Privilege over information about the system, such as known error logs, means that system operators can manipulate disclosure procedures to their advantage [127]. This includes many types of systems, such as accounting systems (e.g., Horizon, linked to one of the biggest miscarriage of justice in the UK [128]), breathalysers (See Bellovin et al. [129]), and newer data processing systems that result in unfair and harmful outcomes [130, 131].

Transparency enhancing technologies offer a way to not only provide trustworthy transparency through the use of security mechanisms but also to scale transparency. For example, the IPCO, which audits law enforcement requests for telecommunications data in the UK, perform local inspections of a limited amount of offices to produce their audit [132]. Transparency enhancing technologies could allow for larger and more efficient audits of many practices.

Moreover, while transparency can have negative effects on people if they respond to transparency with hiding behaviour, impacting their performance [133], the opposite could be true for computational systems with secure transparency mechanisms because the performance of such systems is determined by the code and infrastructure it runs on, not on whether or not it is being observed. Given two systems that perform similarly, if transparency is cheap enough to implement and expensive enough to cheat once implemented (e.g., breaking the logging mechanism's cryptographic properties), the honest transparent system will be cheaper to operate than the one that tries to cheat transparency, which should make it more

competitive. (That is unless the system is so broken in the first place that whatever is revealed by transparency condemns the system.)

Transparency can also be seen as a tool for efficiency. Decentralized systems are often desired because they do not rely on a central party, but centralized systems are typically more efficient to operate They can also make more sense logistically, for systems that either involve sensitive data that cannot be used in an encrypted form for operational reasons or simply to avoid the burden of coordinating many (sometimes unaligned) parties. A decentralized transparency enhancing technology, overlaid on top of a centralized system with a trustworthy interface between both, can provide a useful compromise between the inherent efficiency and logistical advantages of the centralized system and the lesser trust required by a decentralized system.

### 4.2.2 Forms Of Transparency

Transparency can take numerous forms based on the direction in which information flows, the type of information that flows, and when it flows.

Directions of transparency are reminiscent of basic access control models (e.g., Bell-LaPadula [134] and BIBA [135]), which determine in which direction (upwards or downwards) information can be read or written. Unlike many access control mechanisms, however, transparency requires that information leaves the system and be accessible by users with no privilege over the system, and restricts the write access of privileged users over this information.

Concerning the type of information, there can be information about inputs to a system, processes executed within the system, and outputs of the system, where different levels of transparency (or data granularity) matter. For example, when revealing the inputs to a system, the ordering of inputs can also be important as the ordering of data used to train a model can affect its performance [136].

Timing determines when information is made available. It is uncommon to have real-time transparency when humans are involved as knowingly being surveilled can affect behaviour [137]. A computer cannot be aware that its actions are being logged but a human user of the computer will be, so this can still be a

concern in some cases. Even for entirely computational systems, transparency may only be useful if there is enough information to obtain an aggregate view of the system's performance but systems such as cryptocurrencies offer a live transparent view of the system.

### 4.2.3 Criticisms of Transparency

#### Lack of effectiveness

The assumption that underpins much of the belief in transparency is that it will lead to accountability, better behaviour, and increased public trust. Criticism of this assumption is centred around the gap between the dissemination of information and its usefulness in enabling sanctions on a misbehaving party [138].

Etzioni has argued that there is little evidence that supports the view that transparency is an effective accountability mechanism [139]. The argument is that transparency is no alternative to regulation (it can only be complementary) because regulations cannot be replaced by offloading the responsibility of demanding and analysing data to citizens without the time or other resources to handle these tasks.

This is backed up by Ferry and Eckersley, who found that, in the UK, the replacement of formal audits with requirements for English local authorities to publish datasets (with little contextual information) weakened accountability [140]. In countries without regulations that implement effective accountability, however, transparency can be effective at bypassing corrupt official audit processes [140].

The issue is that information being transmitted about a bad outcome does not prevent it. Moreover, it does not prevent future bad outcomes either as it does not, by itself, mitigate their possibility. A practical example of this is mandated disclosures such as nutrition labels, which do not prevent any nutritional harms that, in any case, are linked to many factors beyond the nutritional value of a food item. The same is likely to be true with proposals for data and privacy nutrition labels. A label stating that a dataset has flaws does not prevent anyone from using the dataset and producing a flawed model trained on that dataset.

Research on the effectiveness of privacy labels has also shown that issues of judgement and misdirection could render transparency ineffective [141, 142]. De-

velopers themselves are not always well equipped to evaluate the labels they create, because privacy is not necessarily their expertise and they may not account for harms that are unknown to them [143]. If any harm is perceived as originating from the use of a problematic dataset or privacy-invasive system, a system operator will not be prevented from deploying such a system and may also rely on nutritional labels as cover if the process that produces these labels can be influenced.

Yu and Robinson have a similar view on open government technology and data, arguing that while it may allow the public to contribute in new ways, it does not create any government accountability [144]. Open government initiatives generally do not imply any effect on how government works (other than publishing data) so any faulty process is likely to remain in place. Thus, open data and transparency may be used as a trojan horse for other political goals [145].

If transparency by itself does not entail accountability, it follows that it also does not necessarily create trust. Despite greater access to information, for example in the case of government transparency and freedom of information, trust has not increased [146, 147, 148, 149]. If transparency only reveals systemic faults, why trust such a system?

### Restricted transparency

Obtaining information that is theoretically available, for example through Freedom of Information requests, can also be an issue that requires people to develop specific expertise. In other cases, the release of bulks of information may also obfuscate important information [150]. Even if a party is honest, the release of information implied by transparency does not necessarily imply the effective communication and understanding of that information [147] or that the information that is released is not chosen purposefully to serve a chosen narrative [151].

These criticisms extend to algorithmic transparency for black box computational systems [152, 153]. Burrell distinguishes three forms of opacity in the context of algorithmic systems, opacity as intentional corporate or state secrecy, opacity as technical illiteracy, and opacity as the way algorithms operate at the scale of application [154].

Rights such as data subject access requests may also not work well in practice [155]. This highlights the gap between transparency and other properties (e.g., fairness and explainability) of a computational system. Knowing the inputs, rules, and outcomes of a complex system may not be enough to understand its processes. Thus, while auditing is necessary and possible, auditing decisions that result from algorithms can still pose a significant challenge [156].

Even systems that are open source are not necessarily more or less secure than closed systems [157, 158] because there are many steps in between code being released in open source form and bugs in the code being identified and fixed, such as having the necessary resources and processes to fix bugs. Again, this highlights the gap between the availability of information and actions taken based on that information – in this case, auditing for and fixing vulnerabilities.

### Tension with privacy and confidentiality

Another criticism of transparency is that it can cause harm privacy or negatively affect businesses that rely on confidential components in their systems. This is particularly important for systems that process sensitive data, despite the fact that greater transparency about the sharing and processing of sensitive data may be desirable.

The potential privacy harms brought on by the release of information are also used to restrict transparency. Freedom of information requests may be refused if they involve the release of personal information that would contravene data protection principles [159, Chapter 36, Part II, Section 40].

Similar situations occur when it comes to challenging systems. For example, Uber invoked privacy concerns to impede a challenge by Uber drivers seeking to obtain information about the system that they were subject to [160]. More generally, unless compelled to, companies are often extremely reluctant to disclose anything that they can argue falls under commercial confidentiality.

## 4.3 Essential Mechanisms

This section introduces transparency enhancing technologies based on logging mechanisms, sanitization mechanisms that process the data into a format suitable

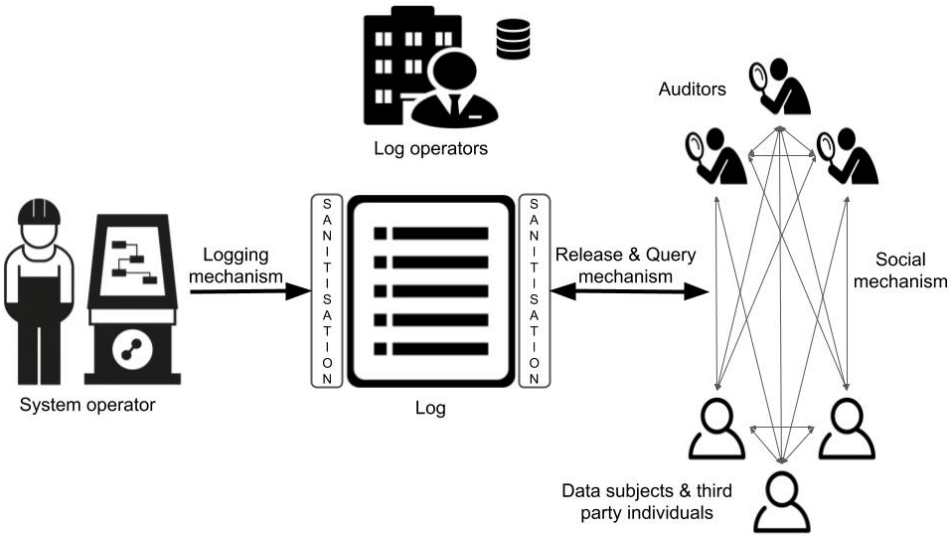


**Figure 4.1:** Summary of essential mechanisms for transparency enhancing technologies (logging, sanitization, release and query, external) and their place in a transparency process.

for release, release and query mechanisms, and external mechanisms to make use of transparency. Figure 4.1 illustrates where each mechanism takes place and the parties it relates to.

Logging involves the system operator of the subject system and log, which is maintained by log operators.

Sanitization takes place either between the logging mechanism recording information and committing it to the log (e.g., to protect commercially confidential information that even trusted auditors may not see) or before the release and query mechanism (e.g., to allow for both privacy-preserving releases of information and access to raw data depending on the party information is released to, and enforce access control to information).

The release and query mechanism relates the log to the users of transparency (auditors, data subjects, and other individuals) who then relate to each other and take action through external mechanisms.

### 4.3.1 Logging mechanism

Transparency requires information to be recorded and traceable [161], for example in the form of a chronological list of events or actions that have taken place, a record

of the data used by the system to operate, or even a complete record of any byte in a current or past state [162].

Secure logging mechanisms have been of interest to cryptographers for a long time [163, 164, 165, 166, 167, 168, 169, 170]. For the purpose of transparency, they have coalesced under authenticated data structures [31, 32] and transparency overlays [29], which are designed to broadly ensure that the log is verifiably append-only, can be used to lookup information, and is consistent in the sense that it shows the same information to everyone and does not equivocate. This is typically achieved with Merkle trees or blockchains, although more recent work has also explored the use of append-only dictionaries.

## Merkle Trees

Merkle Trees are binary trees based on a hash function $h$ such that each node $i$ takes the value $h_i = h(h_{left(i)}|h_{right(i)})$ based on its left and right children. Given that $h$ is collision-resistant, tamper resistance is guaranteed as modifying any node will result in a different root hash. This makes it possible to verify the integrity of any data encoded as a leaf in the tree.

A history tree, following the work of Crosby and Wallach [171], grows from left to right and is used by systems like Certificate Transparency [172]. This allows for logarithmic-sized proofs that the log is append-only as new values (e.g., the hashes of new certificates in Certificate Transparency) are added to the log by a log server. This addition results in a new Merkle tree and root hash, which is signed by the log server. Because the tree grows from left to right, it is then possible to efficiently verify that the new Merkle tree includes everything that was included in the old one, showing that it is append-only. Looking up specific certificates, however, requires linear-sized proofs.

As shown by Chase and Meiklejohn [29] the Certificate Transparency log satisfies *consistency*, meaning a potentially dishonest log server cannot get away with presenting inconsistent versions of the log to different parties, *non-frameability*, meaning that parties cannot blame the log server for misbehaviour if it has behaved honestly, and *accountability*, meaning that there exists evidence that can be used to

implicate log servers that promised to include events but then did not.

A prefix tree, as used by CONIKS [85] to allow users reliant on a PKI (e.g., for communication apps) to verify the consistency of the public keys of other users, has leaf nodes ordered in lexicographic order. This makes it efficient to look up values in the tree, although showing that the log is append-only now requires linear-sized proofs. For example, a client can register name-to-key bindings in the Merkle tree's leaf nodes, which other clients can then lookup on behalf of other users. To verify a name-to-key binding in the tree, a client checks the signed tree root (STR), which includes the root hash and a hash of the previous STR (successive STRs form a chain), and the inclusion of that name-to-key binding with the path from the root to the leaf node for that name-to-key binding.

Non-inclusion of a name-to-key binding can also be checked by verifying that given an index (i.e., a name), there is no key data mapped to it.

To prevent incidents, clients monitor their user's key bindings do not change unexpectedly and verify that the PKI's identity providers are presenting consistent versions of their key directories to all participants by checking that a provider has correctly signed the STR and that the hash of the previous STR matches what was previously seen.

### Combining Merkle trees

A prefix tree and a history tree can be combined to form a verifiable log-backed map [173, 174, 175, 176]. (The prefix tree can alternatively be a hash treap [177, 178].)

The prefix tree in a verifiable log-backed map, which can be in the form of a *sparse* Merkle tree pre-populated with all possible hashes (e.g., $2^{256}$ leaves to match all possible SHA-256 outputs) [179, 180], serves as a map (i.e., key-value store), while the history tree is used as a log that records all signed root hashes for the map, ensuring that clients can verify that the map they are shown has also been shown to others that have audited the log. This combination of both types of Merkle trees allows for a wider range of efficient proofs than either type of Merkle tree could support on its own (i.e., append-only for the history tree, look-ups for the prefix

tree) [173]. Users, however, still need to collectively check that both Merkle trees track the same keys and values.

A third Merkle tree can be added to construct an unequivocable log derived map [181], in which the first tree is a history tree log of operations, which are batched into a prefix tree that allows efficient lookups of operations, and the third tree records the root hashes of the second tree.

More recent work by Hu et al. [182] also combines history and prefix trees by proposing a history tree in which the internal nodes store the root hashes of prefix trees. At any given epoch, the root hash of the history tree summarizes the state of all prefix trees at that epoch, making it easier to monitor new changes, while the internal prefix trees make it easy to look up key values in the current epoch. Because the history and prefix trees are part of the same tree, checking that both trees track the same keys and values is easier.

Reijsbergen et al. [183] also combines several types of Merkle trees, this time a prefix tree in which all the leaves are the root of a Merkle sum tree in which nodes contain homomorphic commitments to the sum of the values of their child nodes, down to the *value* of each leaf. The prefix tree structure enables efficient lookups whilst the sum tree makes it possible to support a wider range of queries (sums, counts, averages, min/max, and quantiles) with integrity guarantees.

### Append-only dictionaries

Append-only dictionaries based on bilinear accumulators [184] have been proposed as an alternative to Merkle trees, enabling logarithmic-sized append-only proofs and polylogarithmic-sized lookup proofs, although high append times and memory usage, meaning this approach is not yet practical.

### Blockchains

Blockchains provide a decentralized and tamper-resistant way of updating and maintaining a global state. Transactions that update the state are logged on the blockchain, making it possible to replay all transactions and to verify that something has happened if it is included in the blockchain, as well as when it was included.

Beginning with Bitcoin [185], blockchains have been used by cryptocur-

rencies to provide a transparent record of transactions over a network. As Ethereum [35] and later projects have shown, it is possible to rely on blockchains to execute arbitrary programs (smart contracts) and record these executions on the blockchain. This allows a wide range of applications to run transparently on top of a blockchain or to use an existing blockchain to store evidence in a tamper-resistant way [18, 17, 186, 83, 2].

Blocks in a blockchain store data (including the state of a smart contract) in Merkle trees so transparency applications that run on top of Merkle trees can be adapted to a blockchain so that its consensus protocol replaces the need for gossiping between clients that is required in a Merkle tree based system to guard against equivocation [187, 188].

Blockchains can be permissionless or permissioned. For logging purposes, the effect of choosing one or the other is that in a permissionless setting, it is possible to use an existing public blockchain, such as Ethereum, in which case the blockchain will be maintained regardless of your use case because many other applications rely on it, as well of the value of the underlying cryptocurrency. Thus, any incentives to maintain (or not) a reliable log are taken care of (at a price determined by the underlying cryptocurrency).

On the other hand, relevant events may not appear in an accurate chronological order because their inclusion will depend on miners who will primarily care about including the transactions that maximize their revenue rather than the needs of a single transparency application.

The effort required to use an existing public blockchain and write a smart contract for it may also be much less than deploying an entire system like Certificate Transparency, allowing for more applications of transparency.

In a permissioned setting, known pre-determined parties will have to ensure that the log is maintained but, because there is no need for an underlying cryptocurrency, the system could be set up to include new events to the chain as they arrive rather than at the wishes of an uninterested miner. In this case, because all parties are known and the blockchain is more likely to be application specific than a

general-purpose blockchain, this setting is also much closer to deploying a Merkle tree based system like Certificate Transparency, with the benefit (or cost) of having a consensus protocol.

### 4.3.2 Sanitization mechanism

The information recorded on a log will often be sensitive, in the sense that it affects the privacy of an individual or that it reveals confidential information about the system it is pulled from. For this reason, sanitising the information that is logged will be necessary but must be done in a way that does not compromise the desired transparency.

The sanitization mechanism determines how logged information is processed, in plaintext (i.e., *unsanitized*), through a privacy-preserving form of data release (e.g., by adding noise or generating a synthetic data [2]), in an encrypted form to be decrypted by specific parties (e.g., designated auditors being given access to raw data, individuals accessing individual evidence [2]), or using cryptographic techniques such as zero-knowledge proofs to assert relevant properties of the logged information without revealing the underlying data [17, 18, 83].

Access to unsanitized information may be required if no sanitization mechanism exists that is compatible with the desired transparency. For example, there may be no way to satisfy reasonable differentially private bounds without adding excessive noise, to produce zero-knowledge proofs that assert the necessary properties of the logged information, or simply to rely only on cryptographic proofs about data. In such cases, it may be necessary to permit access to unsanitized data by designated auditors, while the public is given access only to sanitized data that can be used to verify the results of an audit published by the designated auditors.

Beyond the data itself, identifiers (and other metadata) that allow users to verify their individual data may also need sanitization. CONIKS, for example, uses a verifiable random function to produce a user identifier for the log that does not reveal the identity of the user to others [85], and more recent work has introduced append-only zero-knowledge sets that minimize the leakage from queries [189, 190].

### 4.3.3 Release and query mechanism

Once data is logged, it must also be possible to release the data or perform queries on it. As shown by Reijsbergen et al. [183], it is possible to implement (Merkle tree) logs in such a way that they natively support broader queries than simple lookups, but more can also be done.

Given a database, it is possible to store the hash of the database on a log, enabling users to verify that the database they are querying is the same as the one indicated by the log if they can download the entire database, but this does not guarantee the integrity of a query on that database.

Work on single client authenticated databases [191, 192]; that is, outsourced databases that guarantee the integrity of queries and updates to the database, has led to work combining authenticated databases with a log such as a blockchain on which a smart contract is running [193]. The log ensures consistency and allows clients to verify that the database they are querying (without needing to go through the blockchain) is the database that has been recorded on the log, allowing for a broader set of queries than what is natively supported by the log itself.

Specialized formal languages, similar to TILT [194] (developed for the GDPR transparency requirements), could also be developed to produce application-specific transparency APIs that return human-readable answers to queries.

As discussed in the case of sanitization mechanisms, data may appear in different forms to different parties. For example, only some designated auditors may be able to access raw data. One way of doing this is simply to encrypt data under the relevant parties' public keys so that only they can decrypt the raw data, but another possibility is for the release and query mechanism to implement access control that determines who can query the log. Depending on the type of log, this may be more or less simple. For example, a blockchain based system can implement access control via a smart contract. This could also be set up to log queries if necessary. For a Merkle tree based system the access control mechanism would have to be built on top of the logs.

### 4.3.4 External mechanisms

Transparency cannot be expected to be effective by itself, it must work to enable action based on what it reveals. For example, if transparency produces evidence that a system has malfunctioned, it can allow aggrieved parties to take legal action, governance decisions about a platform or network [195], and the removal of parties from a network if they cause a fault [196]. This entails supporting processes such as public discussions about the system to which transparency is applied and, for practical accountability purposes, legal processes that resolve disputes about a system or more automated processes that similarly make it possible to contest actions taken by the system. This is a key difference between tools that evaluate the compliance of a system with preset norms (e.g., the correct execution of a program) and transparency enhancing technologies that can allow the norms enforced by a program to be contested [3].

This process starts with users being able to check information that is relevant to them or being notified about such information. Notification tools [197, 198, 199, 77, 200, 201, 202] are a useful way to keep the user in the loop, without needing them to perform queries, when their explicit consent for an action is not required, but this does not necessarily allow a user to contest any action that is taken.

For an action to be contested, there must first be evidence of that action. Often a program is assumed to have been correctly executed unless there is evidence of the contrary, but systems often fail to produce such evidence [103]. Transparency should address this, and gossip and consensus protocols can also play a part in spreading evidence and reaching a conclusion about evidence. What is then important is that the evidence be useful.

For an automated process, the proof must fit the requirements of the program that will evaluate it. For a non-automated process, such as a legal process, evidence being useful means that it should be *admissible* in the relevant jurisdiction. Admissibility involves the data itself and also the authentication of the data, its integrity, the network over which the data is exchanged, and how it is then stored [203].

In both cases, this requires the form of the evidence and the process in which

it will be used to be taken into account before it is produced for it to be useful. In the non-automated case in particular, evidence is not sufficient to contest a system by itself (unlike automated processes) and the outcome of the dispute process can vary much more, up to contesting the existence and norms of the system.

In such cases, it may not always be clear when considering a single event, why the system failed [7]. This can require a broader discussion about the system and both the individual evidence and aggregate evidence (e.g., error rates) about the system to be considered to see which is more likely. To act on information also requires the ability to understand that information, which can be made easier via explanations [204], context [112], and labels [119, 122]. This is particularly important, but also challenging, because disclosure practices are not always well designed [205].

## 4.4 Transparency and Security

Although many transparency enhancing technologies have come from security and cryptography research (e.g., cryptographic logs) and, therefore, have involved a security-focused approach, this is not always the case. Moreover, even for cryptographic mechanisms, threats are typically expressed in terms of the cryptographic properties of the mechanisms, particularly when these mechanisms are introduced as abstract primitives, useful for applications outside of transparency, rather than as part of a system focused on transparency, which is our approach here.

### 4.4.1 Assets and beneficiaries of transparency

The inputs, processes, and outputs, of systems are assets for the parties that own and operate them. The value of these assets can depend on their confidentiality. Datasets, a codebase, a machine learning model and its outputs, can all contribute to a competitive advantage, and their confidentiality can also help avoid liability for flaws in the system, or give the illusion of technical sophistication.

Transparency can benefit system operators if it increases public trust. This can be true regardless of whether or not the system is good by any measure because an organization operating a flawed system may engineer a form of transparency that

does not reveal these flaws by, for example, limiting transparency to only reveal favourable information.

Because transparency does not necessarily increase trust, however, operators of reliable systems may feel they have little to gain and operators of unreliable systems may have little to lose. That is unless transparency is deployed in such a way that, for example, it harms those who operate unreliable systems by enabling consequences.

For the public, transparency should be a valuable asset, revealing useful information about a system over which they have no control and allowing them to take action by choosing whether to use the system, contest it, and hold the system operator to account for any faults. Privacy concerns over the public release of sensitive data that pertains to them may, however, be an important drawback.

Thus, transparency can be both beneficial and a drawback for system operators and the public, and, importantly, the ways in which the public may benefit from transparency may be a drawback for the system operator. When this is the case, it should be ensured that blame avoidance strategies (e.g., avoidance of record keeping, gaming performance metrics) are not put into place [206].

### 4.4.2 Threats based on essential mechanisms

#### Logging

The logging mechanism relates to the system operator of the system, from which information is recorded, and the log operators that maintain the log. Assuming that the logging mechanism is based on sound cryptography (e.g., a secure hash function, public key encryption scheme, and digital signature scheme) then what remains as a threat is the ability of a malicious system operator (or whichever party is responsible for logging information) that attempts to compromise what makes it to the log in the first place.

#### sanitization

As sanitization can take place before or after information is logged, threats can come from either the system operator (before logging) or from data releases and

queries (after logging).

A system operator could try to compromise a sanitization mechanism just as they would the logging mechanism itself. A sanitization step taking place before the information is committed to the log would be intended to work towards the confidentiality of commercially sensitive information about the system or to respect the privacy of users who relate to logged data. This could be abused by the system operator to hide other information without having to compromise the logging mechanism.

For a sanitization mechanism that takes place after information is logged, threats are posed by parties attempting to learn private information about others from the information they have access to.

The sanitization mechanism could also be used by log operators, if sanitization is done at the interface between the log and users of transparency, or auditors, if they are given access to raw information that they sanitize for public release, to compromise the information that is released. This can be achieved either by producing sanitized information that does not relate to the original information (e.g., releasing wrong statistics) or relying on an honest use of a sanitization mechanism that obfuscates some information as part of its use (e.g., by adding noise).

### Release and query

The form of the information made available by release and query mechanisms will depend on the sanitization mechanism, so the threats that are specific to release and query mechanisms will be those that target the access control it implements and the integrity of the information (sanitized or unsanitized) that is released. Given that information should broadly be released to everyone except for individual evidence (available only to data subjects) and unsanitized information (available only to trusted auditors), the threat is that any other party may try to pose as an individual or trusted auditor to gain access to their privileged information. The right to access under the GDPR has been abused for this purpose [207], as well as to infer information about the organization answering the query [208].

If information is simply released, without the need for queries, threats could be

**Table 4.1:** Threats for transparency enhancing technologies based on editorial control (EC) and individual evidence (IE).

| Mechanism | Threat | Affected transparency property | Threat actor(s) |
|---|---|---|---|
| Logging | Compromised logging mechanism (EC, IE) | Integrity | System operator |
| | Compromised log server (EC, IE) | Integrity, Availability | Log operator |
| | Collusion between system operator and log operators (EC, IE) | Integrity, Availability | System operators, log operators |
| sanitization | Loss of privacy for data subjects | Respect of privacy and confidentiality | Users of transparency |
| | Control over logging (EC. IE) | Availability | System operator |
| | Control over release and query responses (EC, IE) | Integrity | Log operators, auditors |
| Release & Query | Access to raw data or individual evidence | Respect of privacy and confidentiality | Users of transparency |
| | Restricted releases (EC, IE) | Availability, interpretability | Log operators, auditors |
| | Constraints on queries (EC, IE) | Availability, interpretability | Log operators |
| External mechanisms | Misinformation & disinformation | Interpretability | Auditors, data subjects, third parties |
| | Lying about individual evidence (IE) | Trustworthiness | Data subjects |
| | Discrediting individual evidence (IE) | Actionability | Third party individuals |

posed by having only a partial release of information, or a different release of information to different users. When queries are involved, the threats are that the query mechanism could constrain acceptable queries to queries that are not practically useful. It could even do so for a priori valid reasons such as limiting the privacy loss associated with queries, as in a differential private query model once the privacy budget is used up. A limited query mechanism could also serve to require an impractically large number of queries to obtain any useful information.

### External

External mechanisms (not necessarily technical mechanisms) represent the interactions between users of transparency and the actions that they can take based on it. The threat in this case is misinformation and disinformation and the threat actor can be any user of transparency giving (mistakenly or intentionally) inaccurate information.

This can be seen as an attack on the integrity of the information made available through transparency, which can be mitigated by ensuring that the same information (barring individual evidence) is available to all. In the specific case of individual evidence, it should be ensured that an individual cannot lie about their individual evidence, but also that they can use that to show that any individual evidence they disclose is correct.

### Editorial control and individual evidence

Examining different attempts to implement transparency around the world, Taylor and Kelsey found that the two general threats to transparency were editorial control,

the ability to control what is made transparent, and individual evidence, the ability to suppress the ability of a person to find information that relates to themselves through transparency [98].

We relate this to the mechanism-specific threats we have outlined above in Table 4.1. Both editorial control and lack of available individual evidence can occur through the system operator (logging mechanism), and the log operators and auditors (sanitization and release and query mechanism), resulting in effects on the external mechanisms.

## 4.5 Transparency Infrastructure

### 4.5.1 Requiring and maintaining transparency

Deploying transparency requires an infrastructure that supports the operation of logs and the storage of any data required, including data that may not be stored on the log. Because logs (and any other data) may be used after the system (or its operator) it originates from stops operating, they must be stored independently from the system. Thus, although a centralized approach could be sensible on the basis that only the system operator has a business reason to store that information, it may not be reliable for transparency.

Relying on distributed storage, however, raises questions about how to distribute it. Parties such as NGOs monitoring government activities or public institutions monitoring some businesses may have a strong incentive to support transparency infrastructure that relates to issues that they investigate as it directly supports their goals.

This can also be the case in commercial settings. Google, for example, is responsible for the design and deployment of Certificate Transparency. Because Google Chrome is the dominant browser [209], it has a direct interest in keeping Certificate Transparency operational, requires that any certificate appears in at least two logs, and operates some of the logs itself. (Google previously required one of the two logs to be a log operated by Google [210].)

Unfortunately, this example does not generalize well. In most cases, the parties

that design the transparency enhancing technology may not be those that operate it, or may not have a direct incentive to ensure its success or the resources both in terms of influence on the ecosystem and technical resources (e.g., in the case of NGOs) to guarantee it. Proponents of blockchains and cryptocurrencies argue that they offer the possibility of designing decentralized systems that, via mechanism design, can ensure that participants in the system have incentives – typically financial – that are aligned with maintaining the system. Blockchain can then serve as logs, requiring only a smart contract to deploy, and services such as Filecoin [211] could also offer decentralized storage when it is necessary to store more than logs.

Users themselves could drive businesses to provide greater transparency as they do react to, for example, being shown the extent to which they are tracked [212] and how moderation is applied [213]. However, they often have to rely on tools set up by system operators that do not provide complete transparency, or transparency that users can understand [214, 215]. As we already noted in Section 4.4, system operators may not be incentivized to provide effective transparency, leading to a market for lemons.

Regulation could also play a part by imposing a statutory requirement to provide transparency could be through enforcement action of a regulator such as the Federal Trade Commission or a data protection authority. The European GDPR, which effectively applies globally to any service that has users who are citizens of the EU, notably includes several articles concerning transparency.

Designated auditors may also have the power to ask for the infrastructure needed to operate a transparency enhancing technology. For example, the IPCO in the UK is tasked with auditing how law enforcement access telecommunications data (a yearly report is published [132]) and can require that public authorities and telecommunication operators provide any assistance required to carry out audits, which could include implementing IT infrastructure [216, Section 235(2)].

Some regulations, like the German Network Enforcement Act (NetzDG), do include require transparency requirements about, for example, how unlawful content is dealt with and have resulted in fines for companies such as Meta. Companies

differ in how they implement their compliance with this regulation [217] and are likely to differ in implementing any other kind of transparency requirement. Standardization may, therefore, be required if there is any hope of achieving reliable transparency across different types of systems, and this should be done taking into account threat models and mechanisms to deal with these threat models, and still allow enough flexibility to adapt to, for example, case-specific sanitization needs.

In particular, because regulators are not the people affected by flawed systems and can typically only levy fines on system operators who treat these as a cost of business, transparency that provides information to regulators is unlikely to offer much progress. Transparency that is user-facing, and can inform users in a way that allows them to take action on the basis of that information may be more effective.

### 4.5.2 Truth

A limitation of logs is that their security properties cannot ensure that any logged data or event is true. Dealing with this depends on how the logging mechanism can ensure that the recorded value matches that of the object of interest, and what the logging mechanism actually records.

In Bitcoin, miners reach consensus on which public keys own each bitcoin. A user may want to send bitcoins to another user but if the transaction is dropped by the network the transaction fee was too low, then the transaction is never executed or recorded. Thus, the Bitcoin network is transparent about how the miners view the network, not about every action of the users in the network.

Moreover, not all real-world transactions are logged because Bitcoin private keys may be exchanged offline with no mapping between keys and identities to restrict this.

Likewise, Certificate Transparency is transparent with respect to the set of certificates accepted by log servers, not with respect to all certificates emitted by certificate authorities as some may not be logged. Browsers can reject certificates that do not appear in Certificate Transparency logs, however, which ensures that log servers that are operated by, for example, Google, have the incentive to log all valid certificates sent to them by certificate authorities.

The interface between the device that records information that is logged and the log is also important.

A malicious recording device would be a clear weakness so a trusted hardware interface could be used. The security of trusted hardware components may, however, be centralized if all units are the same. If one unit is broken then, for example, the attestation key could leak [218], rendering all other units worthless. This is a case of weakest-link security that depends on the party with the lowest benefit-cost ratio in securing their unit [219], in a scenario where that party may be adversarial and have full physical access to their hardware.

Alternatively, it may be possible to rely on non-colluding parties to cross-verify information.

Problems may also occur if there is no ground truth for the logged data. For example, wage transparency could identify wage gaps but if the party that logs salaries is the business itself, the logging mechanism (or any computation used to identify a wage gap [220]) can execute correctly regardless of the data (and the resulting analysis) being true if individuals cannot verify their inclusion in the computation.

Problems can also occur when dealing with physical objects, because this requires a secure way of mapping physical objects to digital objects that can be authenticated once logged. Mechanisms that provide cryptographic-like mechanisms to authenticate certain physical objects do exist, however. There is a body of work that studies how paper documents could be authenticated based on their physical characteristics [221, 222, 223, 224, 225, 226, 227, 228, 229]. This would allow the document to be logged with its fingerprint, allowing it to be authenticated later if required.

## 4.6 Balancing Transparency With Privacy

Because privacy concerns can create legitimate restrictions on transparency, privacy enhancing technologies that preserve privacy while retaining the utility of information can enable transparency. (In turn, transparency can help users identify privacy risks [230, 231, 232, 233].)

There are two types of information to consider, aggregate information related to a population and information related to individuals. Aggregate information makes it possible to determine how the system is functioning as a whole and whether it is, for example, (un)fair, (un)biased, or error-prone. By itself, this can be enough to reach a conclusion about the system such as whether the system should be modified, shut down, or to make the choice of participating in the system. For individuals, it is also important to be able to determine how they are personally affected by the system as, for example, a biased system will not impact all users in the same way.

In the case of aggregate information, the privacy requirement is that the aggregate information should not leak information about an individual, including the inclusion of an individual's data in the data that was used to produce aggregate information. This often involves differentially private mechanisms that determine the kind of perturbed data that can satisfy data protection requirements [67, 68], and zero-knowledge proofs, which allow the execution of a process to be verified without revealing anything else about the process [19, 14, 15].

For individual information, controlling access to information also matters since revealing information only causes a loss of privacy if it is revealed to someone other than the individual it relates to.

While differentially private mechanisms and zero-knowledge proofs appear necessary to balance transparency and privacy requirements, there are concerns tied to editorial control and individual evidence that we consider here.

### 4.6.1 Editorial control

Editorial control encompasses not only the ability to prevent access to information (e.g., information being logged by the transparency enhancing technology) but also any way of influencing what is or is not recorded, the format in which it is recorded, what is shared with who, and the terms under which information is shared.

Differential privacy does this by changing the information that is shared, for example through the addition of noise or by sharing a synthetic dataset rather than the original one. While differentially private mechanisms work to preserve as much

utility as possible, this is nonetheless a form of editorial control that can work in favour of an adversarial system operator. This is because the addition of noise disproportionately affects less represented groups in the data. For example, the adoption of differential privacy for the U.S. Census could effectively erase smaller towns from census data [234]. More generally, differential privacy could be used, under the cover of it being a required privacy enhancement, as a way of masking bad outcomes on minority groups, or to make low-frequency faults disappear.

Another way in which differential privacy can lead to editorial control is by limiting the number or type of queries that can be made as part of the query mechanism of the transparency enhancing technology. Differential privacy assigns a privacy budget that dictates how many queries can be made (based on their sensitivity), placing a limit on what and how much data subjects, third-party auditors, and third-party individuals can do through a query mechanism. It could also allow an adversarial auditor (perhaps colluding with the system operator wanting to work against transparency) to exhaust the privacy budget by performing high-sensitivity queries that do not reveal anything unwanted.

However, this can be avoided by relying on a release mechanism that generates synthetic data (although not a general solution [235]) that can be queried ad infinitum, rather than relying on a query mechanism that serves differential private answers to queries on the database of original data.

Zero-knowledge proofs can also act as a form of editorial control. A zero-knowledge proof reveals nothing but the truth of a statement, which can remove context from a query. Requiring that any query by an auditor be expressed as a provably true or false statement within the constraints of a formal language may also restrict the range of possible queries, and prevent necessarily vague queries.

Querying for provable statements can also be made inefficient this way as queries must be designed without access to data. This could mean iterating over queries of the type "is the number of data points with attribute $\alpha$ greater than $x$". The result of this is that practically speaking, it is only possible for auditors and individuals to verify statements that are given to them by those who control the in-

formation that is queried, rather than being able to perform their own investigation.

Moreover, detecting a flawed implementation of a zero-knowledge proof system that allows counterfeit proofs to be produced can be hard. Flaws in zero-knowledge proof systems have only happened by accident so far [236, 237], but there is a precedent for cryptosystems that could plausibly be exploitable by design [238]. A malicious system operator could attempt to introduce an intentionally flawed zero-knowledge proof system that would allow them to appear compliant with any desired norm.

### 4.6.2 Individual evidence

Individual evidence is desirable for the simple reason that a general overview of a system may reveal issues with the system (e.g., it is biased against certain attributes or has bugs) but fail to show their impact on individuals (e.g., if one was discriminated against or affected by a bug). This requires not only knowledge of the system's outcome for that individual, which usually will be known for the outcome to have any effect although this may not always be the case (e.g., for confidential processes) but also some form of ground truth for what the outcome could have been, which in general may be harder to obtain.

For example, the covid-19 pandemic caused secondary education exams in the UK to be cancelled in 2020 and grades to be awarded based on an algorithm using results of past students as input. The population outcome was normal by design – the distribution of grades matched historical distributions for each school – but it meant that students who performed outside the historical norm could be awarded lower or higher grades than expected for the sake of preserving the historical grade distribution. Individual evidence in the form of teacher predicted-grades, however, made it possible to easily identify how students had been affected (e.g., a student with a high teacher-predicted grade being awarded a low grade) and the algorithmic marking scheme was quickly replaced with teacher-predicted grades [239].

Individual evidence can also be useful when there is a dispute about whether an individual has made an error when using a system or has been a victim of a bug. Human errors and bugs can happen at reasonably low frequencies so conclusively

determining whether one is more likely than the other can be impractical, and neither the presence of bugs in the system nor the possibility of a human error can be used to invalidate the other [7]. Individual evidence that makes it possible to identify the error in an event log and a record of actions by the individual could make it much more efficient to determine whether the error was human or due to a bug in the system.

The role of privacy enhancing technologies, however, is often to make it impossible to link an individual to an input or output of the subject system's process.

Differential privacy guarantees that an individual does not have too much of an effect on outputs so that it cannot be determined their data was used to obtain that output without an additional mechanism that deals with this.

Zero-knowledge proofs remove the relation between the output of the computation it verifies and its inputs. If individual evidence exists, however, a zero-knowledge proof could be used to show an individual that their individual evidence was used in the computation. Without this, an adversarial system operator or auditor could simply use inputs that they choose or generate to obtain valid zero-knowledge proofs for whatever they want.

This means that the use of these privacy enhancing technologies to allow the release of aggregate information requires that additional mechanisms be used for individuals to obtain the individual evidence necessary to contextualize the aggregate information and the effect the system has had on them.

## 4.7 Case Studies

### 4.7.1 Certificate Transparency

SSL certificates are an essential part of web security, allowing a user's browser to verify the owner of a website. Certificates are issued and signed by trusted third parties, certificate authorities, who can be the source of security incidents [240, 241] An example of this is the DigiNotar hack [242], which led to hundreds of rogue certificates being issued with DigiNotar's signing key and DigiNotar certificates being rejected by most browsers [243, 244, 245].

Certificate Transparency [172, 246] was developed to address this type of incident. Acknowledging that it is not possible to prevent rogue certificates from being issued, Certificate Transparency works by making certificate issuance transparent and working against malicious certificate issuance by helping reveal cases where this happens. This is achieved by using logs based on Merkle history trees that ensure the list of logged certificates is a secure append-only transparency overlay [247, 29].

Certificate authorities submit certificates to the logs themselves and browsers will only accept certificates that come with a signed certificate timestamp from log servers, so a malicious certificate authority cannot compromise the efficacy of the logging mechanism by not submitting certificates that they issue to logs and collusion between a certificate authority and a log server is mitigated by requiring multiple signed certificate timestamps from different logs.

Certificate Transparency is widely deployed, with the percentage of mainframe HTTPS page loads and HTTPS connections with at least two valid signed certificate timestamps reaching above 60% as of 2018 for Chrome users [248]. There is significant infrastructural backing from organizations like Google, Mozilla, and Cloudflare, and free services such as Let's Encrypt [249].

There is no sanitization mechanism involved in Certificate Transparency, although some interactions involve privacy concerns for users. For example, when their browser queries a proof of inclusion in a log, it reveals the website that the user is browsing. As a result, most clients do not directly request proofs of inclusion, although solutions based on fuzzy ranges, private set intersection, and private set membership protocols have been proposed [210].

Reporting that a certificate has not been included in a log also reveals a user's browsing activity for that website. This can be mitigated by using zero-knowledge proofs to allow the browser to prove to a browser vendor (e.g., Google) that it knows a signed certificate timestamp signed by a log server (without revealing it) despite the log omitting this certificate, therefore showing that the log does not have integrity [16]. This approach has downsides, however, as it would require changes to

log implementations and APIs, and obfuscate details in investigations of log misbehaviour [250], showing the tension between transparency and privacy goals.

Other issues exist with the certificates themselves and logs, which can be used to identify potentially vulnerable websites because websites with expired certificates tend to more outdated software that may be vulnerable to CVEs [251]. The volume of information available through Certificate Transparency also makes it possible to monitor logs to identify new DNS names (i.e., service endpoints) that may be vulnerable to an attack, rather than inefficiently scanning the IP space [252]. Logs can also be mined to detect subdomains, as well as other sensitive information including names, usernames, email addresses, business relationships, and unreleased products [253].

The volume of logged certificates poses scalability issues as well. Monitors, who fetch and try to spot suspicious certificates, cannot guarantee that fetching certificates returns a complete set of certificates, meaning that fraudulent certificates may be logged but not spotted [254].

External mechanisms play an important role in Certificate Transparency. Certificates must be revoked as time passes or in the event of an incident (e.g., DigiNotar). In such a case, a human decision must be made based on the information available and the potential to act on that information. The latter means that power is concentrated in browser vendors (e.g., Google, Mozilla, Microsoft, Apple, Brave) which are the only parties who can act on certificate transparency revealing a malicious or compromised certificate authority by blocklisting it. Expert users can in principle also inspect logs, but represent a tiny minority of users.

Gossip protocols should play a role in enabling clients to exchange messages containing warnings or inconsistencies between signed tree heads of logs [255], but gossiping is not widespread [256]. There are several ways to work around this, replacing gossiping as a type of external mechanism with a protocol that is integral to the transparency overlay.

The first way is to use a blockchain and rely on its consensus protocol for consistency [187, 188], but this can be expensive because of transaction costs and

has slow finality if relying on a slow blockchain (e.g., Bitcoin or Ethereum).

The second way is to rely on witnesses (e.g., the different Certificate Transparency log servers) could collectively sign a checkpoint of a log, producing some form of consensus that the log has been verified up until the checkpoint [257], but this could suffer from liveness issues if there are too few witnesses.

### 4.7.2 Blockchain based cryptocurrencies

Cryptocurrencies, such as Bitcoin [185], Ethereum [35], and many others, aim to enable decentralized peer-to-peer transactions between users that do not rely on any centralized institutions such as banks, Paypal, and VisaNet [185].

This requires solving the problem of currency minting and double-spending such that no single user can unilaterally determine the amount of tokens they control, or spend the same tokens multiple times. This is achieved by relying on a blockchain, which records blocks of transactions (that refer to the previous block in the chain), which are mined (i.e., validated) by miners expending a scarce resource such as computational work (e.g., proof-of-work, proof-of-storage) or stake in the currency (proof-of-stake) for the right to mine blocks. The state of the blockchain is public and agreed upon by the nodes in the network through a consensus protocol, allowing anyone to track any asset on the network.

Chase and Meiklejohn [29] considered the Bitcoin blockchain as one of their two case studies (the other being Certificate Transparency) in their formalization of transparency overlays. The important difference between the two systems that emerged is that miners in permissionless blockchain systems are not known and, therefore, cannot be held responsible for faults and are not trusted to provide consistent views of the blockchain. This can be dealt with through penalties and *slashing* mechanisms that exist in proof-of-stake cryptocurrencies, such as Ethereum [258], to directly fine or remove from the network block validators that misbehave because being elected to be a block proposer or validator requires staking funds.

Nonetheless, although it is possible to see what is going on with blockchain explorers (e.g., `https://www.blockchain.com/explorer`) that display the latest block information, users must download, store, and verify the entire

blockchain to assure themselves they have the correct information.

As blockchains record an increasing number of transactions they become larger and more expensive to download, store, and verify. For example, the Bitcoin and Ethereum blockchains now amount to hundreds of gigabytes of data, making it difficult for most users to operate a node that independently verifies the state of the blockchain. As a result, users often run light clients that verify only block headers and the transactions inside blocks, decreasing security.

Transparency in this setting, whether at the stage of validating blocks or later auditing past transactions, is useless if it is not used to verify the system's consistency and ensure that only valid transactions are processed, so this is a problem that relates to the transparency of the system.

One approach to solving this issue is based on succinct blockchains that reduce the computational costs of verifying the blockchain [259, 260]. Recursive succinct arguments of knowledge can be produced in time proportional only to the number of transactions added since the previous block and verified in constant time [260]. To verify the blockchain, this allows blockchains to effectively be compressed from hundreds of gigabytes (the size of a blockchain after a few years) to a 22 kilobyte proof that verifies transactions and consensus rules, which can be verified in milliseconds.

Another approach is based on fraud proofs, which involve full nodes producing proofs of invalid transactions that light clients can efficiently verify to narrow the security gap between full nodes and light clients [261, 262]. Fraud proofs also play a role in enabling scaling solutions such as optimistic rollups on Ethereum [263], which process transactions off the main chain (reducing congestion and transaction fees) and then post only compressed transaction data on the main chain. The transparency obtained from the transaction data posted on the main chain makes it possible to verify the validity of transactions and produce fraud proofs for any invalid transactions. (Zero-knowledge rollups, the alternative to optimistic rollups, rely instead on proofs of validity to prevent invalid transactions [264].)

Another commonality with Certificate Transparency is that blockchains do

not necessarily offer much in terms of sanitization mechanisms, and there is no right level of privacy that is agreed upon, between full transparency that compromises basic privacy expectations and fully obfuscated transactions that rely on the blockchain as an integrity check rather than a transparency mechanism.

Early systems, such as Bitcoin and Ethereum, do not offer any privacy because, although they are pseudonymous, it is easy enough to identify unique users by studying the public transaction flows recorded on the blockchain [265] and trace coins that have been used as part of some unwanted activity [266, 267], a practice that has been commercialized by companies such as Chainalysis, TRM, and Elliptic.

More recent systems have attempted to provide greater privacy [268] through the use of zero-knowledge proofs (e.g., Zcash [269]), ring signatures (e.g., Monero [270]), coin mixing services (e.g., Tornado cash [271], sanctioned by the US Treasure since August 2022 [272]), and network level mixing (e.g., Nym [273]). Not all attempts have been successful in achieving their privacy goals because of low adoption, design flaws, and the inherent availability of auxiliary information available via blockchain analysis that can be exploited [274, 275, 276, 277, 278, 279].

Balancing privacy goals with the goal of stopping tainted funds (e.g., stolen funds) from being laundered through, for example, mixing services has also been shown to be possible. One possible solution is to produce a zero-knowledge proof that the funds one has put through the mixing service did not come from any address that is publicly associated with tainted funds. In this case, the transparency that allows the addresses containing stolen funds to be identified would allow other addresses to use privacy services without the risk of facilitating the laundering of stolen funds [280].

Another possible solution is collaborative deanonymization [281], which would allow users to contribute information that helps identify a source of coins processed by a mixing service, enabling transparency that can be determined by users themselves rather than system designers.

External mechanisms also play an important role in blockchains and their gov-

ernance. The blockchain can show miner behaviour such as front-running [282], evidence of hacks, trace stolen funds, and so on. This has led to important debates about, for example, whether the 2016 DAO hack on Ethereum should be reversed with a hard fork (leading to the split between Ethereum and Ethereum Classic) [195], or whether the size of Bitcoin blocks should be increased (leading to Bitcoin Cash and Bitcoin SV).

Social influence also plays a role in such discussions as public figures (e.g., Vitalik Buterin for Ethereum) and influential companies (e.g., Blockstream employed many Bitcoin Core developers) can sway public opinion. In principle, anyone can suggest improvements and fork a blockchain to implement their suggested improvements and publicly showcase them. Thus, although miners have the power to enforce changes as they run the software and validate transactions, and the few developers with write access to the software repositories have privilege over the code, transparency enables some redistribution of power as discussions can be based on entirely public information.

## 4.8 Conclusion

This chapter provides a systematization of log based transparency enhancing technologies, identifying the requirements and essential mechanisms of transparency enhancing technologies, and showing how threat models relate to issues of editorial control and individual evidence.

There are many use cases for transparency: Certificate and Key Transparency [246, 172, 175, 85, 187, 186, 190, 189], cryptocurrencies [185, 35, 269, 270], binary transparency [283, 284], decentralized authorization [181], and socially driven applications such as transparency about wage gaps [220], financial markets [285], legal processes [18, 17, 83], data sharing [2] and usage [286], data mining [287], inference [288], advertising [289], and open government data [290, 291]. Many of these rely (or could as they adapt their threat models) on logs and sanitization mechanisms as we have described.

There are clear challenges to tackle, relating to the infrastructure that would

enable transparency, and balancing transparency with privacy and confidentiality concerns. The two case studies we have provided, Certificate Transparency and cryptocurrencies, show how many of these challenges arise in practice for each essential mechanism and, in some cases, how they can be addressed.

Several additional challenges must also be resolved for transparency enhancing technologies to be practically useful in supporting users and processes such as legal disputes, in which they will engage based on what transparency reveals, and regulations that require transparency.

As we have discussed, there are many possible use cases and approaches that can be taken in designing and deploying transparency enhancing technologies. Based on the history of transparency, effectiveness is not guaranteed. The design of transparency enhancing technologies should, therefore, ensure that any technological attempt to enable greater transparency focus on making transparency not a goal in itself but a tool that serves a broader aim in the system in which it is put in place.

# Chapter 5

# VAMS: Transparent Auditing of Access to Data

## 5.1 Introduction

Personal data plays an important role in activities where there is a high cost of failure, as is the case in healthcare, preventing and detecting crime, and legal proceedings. Often, however, the organizations that need access to this data are not the ones who generate or hold the data, so data must be shared for it to be used. Such sharing must be done with care as improper sharing or modification of sensitive data can result in harm to the individuals whose data is involved, and others, whether through breaches of confidentiality or incorrect decisions as a result of tampered data. If there is widespread abuse of personal data, people may become unwilling to allow their data to be collected and processed even when it would benefit themselves and society.

Simple restrictions on sharing of personal data can be automatically enforced through access control and cryptographic protections, such as preventing unauthorized parties from accessing databases in which personal data is held. However, other equally important restrictions involve human interpretations of rules, consent, or depend on information not available to the computer system enforcing them. For example, access to medical records may be permitted only when it would be in the interests of the patient. Similarly, access to communication records may be permit-

ted only if it is necessary and proportionate to prevent crime.

In such cases, rules cannot reliably be automatically enforced in real-time so the approach commonly taken is to keep records of access attempts and subject the actions to audit. Provided that the audit can detect improper activities and violations are harshly punished, abuse can be effectively deterred. Statistics published about the audit can also provide confidence to society that access to data is being controlled and that organizations who can access data will be held to account.

This raises questions about who performs the audit and how the auditor can be assured that the records they see are accurate. If individuals at risk of their personal data being misused do not trust that the auditor is faithfully carrying out their duties then the goal of the audit will not be achieved. However, because of the sensitivity of personal data and the records containing the justification for data being processed, not everyone can act as an auditor. Even if it was possible to find an organization whose audit would be widely accepted, an audit based on tampered records would not be reliable.

The integrity of the data that is accessed is also important when actions are taken based on this data. When making a medical decision or conducting legal proceedings, relying on tampered data can have severe consequences. It may be possible to refer back to the organization that collected the data to verify its integrity, but if that organization no longer holds the data or has gone out of business, such verification is not possible. Digital signatures can provide some confidence that data is genuine, but if the private key is compromised then any data signed by that key is subject to doubt, even if it was created before the point of key compromise.

To improve on the current situation, we propose VAMS, a system that enables transparent audits of access to data requests. This is achieved by allowing auditors to verify the integrity of the data they see and publish audits that can be publicly verified without compromising the privacy of the parties involved, as well as allowing individuals to audit requests for data that relates to them.

### 5.1.1 Outline of the Chapter

Section 5.3 introduces our setting, threat model, and goals, which address transparency (verifiable audits of aggregate and individual outcomes) and privacy (the verifiability of audits cannot reveal more than what is intentionally revealed by audits).

We describe in Section 5.4 the three mechanisms that we use to build VAMS. Tamper-evident logging provides integrity for the information they see on the log. A log entry tagging scheme allows users to efficiently find log entries that are relevant to them. MultiBallot, a novel adaptation of ThreeBallot [70] as a rule-based way of generating a synthetic dataset, allows published audits to be publicly verified with only a small expected privacy loss.

The operation of VAMS is described in Section 5.5, while Section 5.6 argues that it achieves the goals stated in Section 5.3, and Section 5.7 shows that the two implementations of the log, based on Hyperledger Fabric (HLF) and Trillian, show sufficient scalability and functionality, as well as the ability to accurately verify statistics with MultiBallot. Our results show that VAMS can serve as a lightweight overlay applicable to many use cases.

## 5.2 Motivating Scenarios

To motivate the design of our system, we consider two challenging scenarios: controlling the access of law-enforcement personnel to communication records and the access of healthcare professionals to medical data.

### 5.2.1 Law-enforcement access to communications data

In the UK 95% of serious and organized crime cases make use of communications data [292] – metadata stored by telecommunications providers in their billing system about account holders or their use of communications networks (e.g., phone numbers called, address associated with an account, location of a mobile phone). Telecommunications providers are required to store this data for up to 2 years, but once this period has expired and there is no business reason to store this personal data, they are required to delete it. Within the period that data is stored,

law-enforcement personnel is permitted to request access, provided that they can demonstrate that their actions are legally justified[1]. At the time a request is made, there is, however, no external oversight. Instead, information about the request and the justification for access are stored and made available for audit by the Investigatory Powers Commissioner's Office (IPCO)[2]. IPCO then assess whether law enforcement personnel make appropriate use of the powers they were given, and publishes reports with statistics of how these powers were used [293].

Communications data plays an important role in the investigation of criminal offences, but may also be used as evidence in legal proceedings, for the prosecution or defence. If the integrity of the evidence is questioned, a representative of the telecommunications provider will be asked to appear in court to verify the evidence and attest to its accuracy. If technical issues arise related to this evidence, one of the parties to the case may also request that the court request assistance from an expert witness. This process is expensive, time-consuming, and even impossible if the provider has deleted the original data in the time between the law enforcement agency requesting it and the data being required in court.

To improve the process, industry standards allow providers to sign or hash communications data when it is provided in response to a request from law enforcement. Someone who needs to verify an item of data can compare the hash to the one stored by the provider, or verify the digital signature using the provider's public key [294]. However, if the provider's private key or hash database is compromised, any evidence presented after to the compromise will be brought into doubt, even if it was generated before the time of compromise.

Our system can be applied in this scenario, allowing the integrity of communications data evidence to be demonstrated, even if the communications provider which produced the data no longer exists or has been compromised. Furthermore, the system will give assurance to the auditor that records of requests to access com-

[1]Similar legal powers are available in the US through the use of administrative subpoenas, but as there are no publicly available statistics for their use and there is no centralized oversight, we focus on the UK case.

[2]Before to September 2017 this role of IPCO was the responsibility of the Interception of Communications Commissioner's Office (IOCCO).

munications data have not been tampered with, and assure society that reported statistics have not been improperly manipulated by the auditor. We also show how the system protects the privacy of individuals whose data is requested and also protects the confidentiality of ongoing law-enforcement investigations.

### 5.2.2 Access to healthcare records

In our second scenario, we consider how to empower individuals by giving them control over how their medical records are used and shared. In a healthcare system, once consent has been given by a patient, various actors should be able to access various records associated with that patient. For example, their general practitioner should be able to access scans that were run at a hospital, and researchers running academic studies or clinical trials in which the patient has enrolled should be able to access records relevant to the study.

Currently, patients can only give permission for broad types of activities and may have legitimate concerns that their information is being used inappropriately. Conversely, patients with serious diseases (e.g., cancer, motor neuron disease) often have trouble getting the treatment they need, as universities conducting studies are legally blocked from contacting them, and patients are unaware that such studies are going on.

Opening up access to medical databases may fulfil the needs of some patients but would also open up the potential for abuse, so it is important for patients to have visibility into how their data is being used to understand the implications of their consent. For clinical practice, the default could be that patients opt-in to sharing their data, although they can always opt out if they wish. For academic studies and clinical trials, the default should be that they are opted out, but can opt-in. They can even choose at some granular level (e.g., according to the type of study) which studies they want to opt in to.

One issue with having patients opt in individually is that for some studies this process may not result in a large enough sample. Equally, if patients are deluged with requests for consent, they are likely to resort to some default behaviour ("click-through syndrome") without understanding what they have consented to. As such,

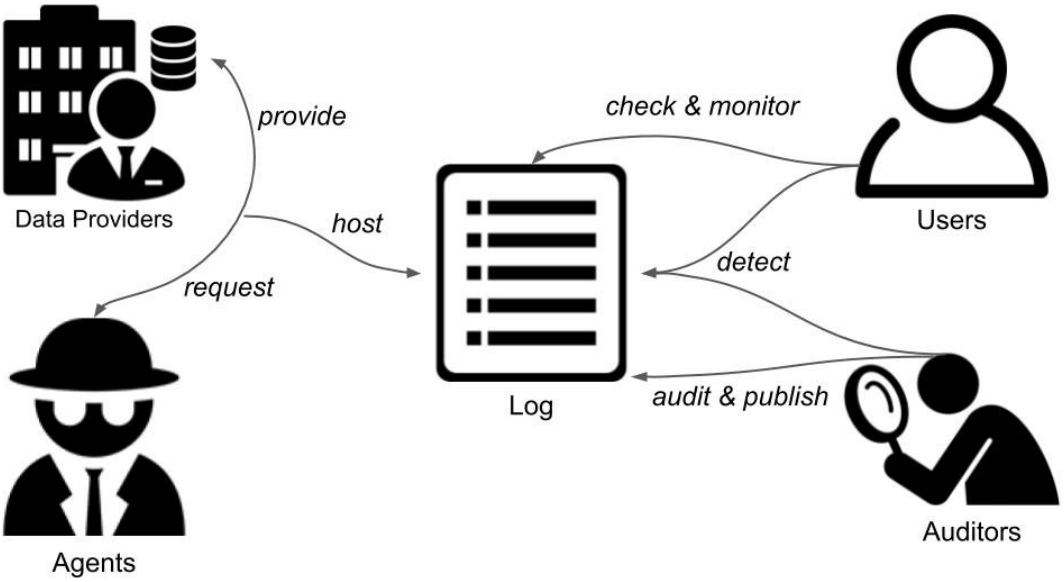


**Figure 5.1:** The parties in VAMS and their functionalities. The optional data broker would act as a user.

patients could outsource these decisions to data brokers; that is, organizations that pay attention to the studies being conducted and are authorized to provide consent on behalf of patients registered with them.

Our system can be applied to allow patients to share their data in such a way as to protect their privacy while ensuring that unauthorized parties are prevented from having access and that authorized parties abusing their access can be detected.

## 5.3 Threat Model and Goals

Our setting (illustrated in Figure 5.1) involves agents, data providers, users, auditors, log servers, and (optional) data brokers. Table 5.1 summarizes the functionalities and malicious behaviours for each party, which we describe below.

The *log* is a key-value store of access to data requests. The values of log entries are *records*, tuples of *elements* such as the attributes of a data request (e.g., the type of data requested by law enforcement) or answers to a medical questionnaire. The log can also contain datasets and statistics published by auditors or a link to the datasets and statistics along with a hash to verify their integrity. *Log servers* host the log of requests (*host*). A malicious log server would aim to give inconsistent views of the log to auditors and users (i.e., attack the availability of logged information).

**Table 5.1:** The parties in the system, the functions they perform and their malicious behaviour.

| Party | Function | Malicious behaviour |
|---|---|---|
| Agent | *request*: append a request to a data provider to the log | Provide an invalid request |
| Data provider | *provide*: answer a request that is on the log | Provide invalid data |
| | *detect*: detect if log servers are behaving dishonestly | |
| User | *check*: look for relevant log entries | Access requests relevant to other users |
| | *monitor*: verify the statistics published by auditors | Infer information from the statistics |
| | *detect*: detect if log servers are behaving dishonestly | |
| Auditor | *audit*: check the log entries for misuse or errors | Infer information from the log |
| | *publish*: publish statistics about entries on the log | Publish inaccurate statistics |
| | *detect*: detect if log servers are behaving dishonestly | |
| Log server | *host*: return the log to parties wishing to inspect it | Provide inconsistent views of the log |
| Data broker | *broker*: respond to requests in place of a user according to their preferences | Misrepresent the preferences of the user |

*Agents* (e.g., law enforcement, medical researchers) request access to user data from data providers (*request*). A malicious agent would aim to access data without it being logged or submit an invalid request, and try to tamper with a logged invalid request before an auditor or user audits it. In other words, a malicious agent would aim to attack the integrity of the log.

*Data providers* (e.g., telecommunications providers, healthcare providers, users) collect user data and receive requests from agents (*provide*). A malicious data provider would aim to give access to data without it being logged.

*Auditors* audit the log (*audit*) and publish statistical reports (*publish*). In the UK, the IPCO is an example of this kind of auditor. They must be able to detect if log servers are behaving dishonestly (*detect*). A malicious auditor would aim to publish an inaccurate report (i.e., compromise the integrity of the audit).

*Users* are members of the public that check requests for their data (*check*) and verify audits (*monitor*). They must also be able to detect if log servers are behaving dishonestly (*detect*). A malicious user would aim to learn information about another user (i.e., attack their privacy) by using the log or published audits.

*Data brokers* are non-essential intermediaries that users can rely on to deal with requests if they are willing to serve as a data provider by, for example, providing data to a study. A data broker can then deal with requests (*broker*) according to pre-set rules from the user. A malicious broker would aim to misrepresent the user's preference accepting (or rejecting) requests that the user's rules would prohibit (or allow).

**Table 5.2:** Transparency and privacy goals that address the malicious behaviours defined in our threat model.

| Goal | Supports | Protects against |
|---|---|---|
| Log availability (T1) | Agents and users (*detect*) | Log servers providing inconsistent views of the log |
| Log integrity (T2) | Agents (*audit*) and users (*check*) | Agents tampering requests |
| Verifiability of inputs used to compute statistics (T3) | Users (*monitor*) | Auditors releasing inaccurate statistics |
| Verifiability of published statistics (T4) | Users (*monitor*) | Auditors inaccurate statistics |
| Transparency of the system (T5) | Auditors and users | Reliance on agents, data providers, or (for users) auditors |
| The log itself does not reveal any sensitive information (P1) | User, agent, data provider privacy | Parties wanting to infer information from the log |
| Verifying an audit is privacy preserving (P2) | User privacy | Parties wanting to infer information from the statistics |

### 5.3.1 Threat Model

We allow every party to act maliciously except for colluding data providers and agents as they could simply exchange data without it being logged. This cannot be prevented with cryptographic techniques because the data must generally exist in an unencrypted form for its primary use (e.g., routing calls or providing healthcare). We, therefore, require that agents log their requests and that data providers do not answer requests without ensuring that the request has been logged. If one of these parties is malicious, their misbehaviour will be caught by the other.

In practice, VAMS as described here would be augmented by procedural and technical access controls that prevent confidential data from leaving a system without being logged. The goal of VAMS is to help ensure that requests represented by the log are compliant with policy and have not been tampered with.

#### Transparency goals

Our motivation is to provide more agency to users, who in many current systems have data requested and provided about them but cannot evaluate this process. To help resolve this asymmetry, VAMS is designed to provide verifiable aggregate statistics (population outcomes) about requests for data and the use of data that can be verified by users, and allow users to check for log entries that are relevant to them (individual outcomes). This is achieved through five transparency goals, which are summarized in Table 5.2 .

**Log availability (T1).** It must be possible for users and auditors to access the information they require for their respective audits.

**Log integrity (T2).** Users and auditors must also be able to check the integrity of the information they access because the information on the log could have been

modified or they could be given an incorrect view of the log.

**Verifiability of inputs used to compute statistics (T3) and of published statistics (T4).** Due to the sensitivity of personal data and the records containing the justification for data being processed, not everyone can act as an auditor with wide-ranging access to log entries. Auditors are therefore relied on to compute and publish aggregate statistics. To minimize the trust required in the auditors, users must be able to verify both the input and the output of the computation of the statistics. Auditors could otherwise publish bogus statistics by miscomputing them or by computing them on a fake dataset that gives the results they desire.

**Transparency of the system (T5).** Users and auditors should only have to rely on VAMS itself to perform their functions and not a potentially malicious party.

## Privacy goals

Our transparency goals must be complemented by *privacy* goals to ensure that parties in the system do not learn private information about one another in the process of performing their audits. This may seem contradictory, but information relating only to a single party is not necessarily required to evaluate the system as a whole. VAMS does not control the contents of a published audit so it cannot control the privacy loss associated with an audit, which will vary based on the requirements and privacy concerns of auditors. As a result of this, our privacy goals (summarized in Table 5.2) focus on requiring that VAMS does not lead to a greater loss of privacy than what is released through a published audit.

**The log itself does not reveal any sensitive information (P1).** This requires that the log entries themselves do not reveal any sensitive information, but also that the log as a whole does not reveal links between requests so the log entries should be unlinkable.

**Verifying an audit is privacy-preserving (P2).** It should not be possible to learn information about individuals from statistics published by an auditor; that is, verifying an audit should not reveal more than the correctness of the audits themselves, although the audit itself may reveal sensitive information.

## 5.4 Building VAMS

VAMS is built using technical mechanisms that we describe below, based on the rationale that follows.

Central to VAMS is the log containing requests submitted by agents through *request*. For the log, key-value stores are a natural choice as log entries (key-value pairs) include keys (i.e., identifiers) that can easily be queried by users performing *check*. Log integrity requirements mean that auditors and users should be able to verify that the records that they access are those submitted by agents, so the log must be tamper-evident.

Auditors and users should also be able to detect log misbehaviour; that is, equivocation in the form of an altered log or a split-view attack in which different versions of the log to different parties. The need for a tamper-evident log can also be seen in cases where evidence is required; for example, when an agent must show that they accessed data with a valid request. In some cases, *urgent* requests that are authorized orally (with paperwork authorized only retroactively) are necessary, such as in the case of a medical or security emergency, so attempting to block invalid requests as they are issued is not enough.

Requests should be signed so that they can be used as evidence to assign liability and to hold the relevant parties accountable. This would work only if the evidence produced is robust so that liability can be properly assigned. Evidence should also exist even if the party that produced it is no longer active; for example, if a data provider declares bankruptcy, or if some servers fail, or are destroyed. Thus, logs should not depend solely on the party tied to the evidence.

Once requests are recorded in the log, auditors perform their audits and publish the resulting statistics through *publish*. Users must be able to verify these statistics through *monitor* (requiring the published statistics and the data necessary to verify their results) without learning more than what is revealed by the statistics themselves (i.e., specific information about other individual users).

To summarize, we need the following. First, we need some kind of tamper-evident log, which requires that the state updates of the log be tied to a blockchain,

a history tree [171], or more generically a transparency overlay [29] with efficient proofs of inclusion and consistency. Second, we need a mechanism that allows the log to be efficiently queried (i.e., identifying relevant requests) without revealing any links between entries on the log. Third, we need a mechanism to publish statistics that can be verified without revealing more information than what can be learned from the statistics themselves.

### 5.4.1 Using Hyperledger Fabric and Trillian as tamper-evident logs

Existing transparency overlays come in the form of distributed ledgers and verifiable logs. We have implemented VAMS twice, using Hyperledger Fabric, a distributed ledger with an underlying blockchain, and Trillian, a verifiable log-backed map.

In both cases, using HLF or Trillian guarantees that the log is tamper-evident due to the underlying blockchain or verifiable log that records state updates, that the availability of information on the log can be assured by making log equivocation detectable, and that the log is easy to query as it is in the form of a key-value store.

#### Hyperledger Fabric

Hyperledger Fabric [295, 296, 297] is a modular open-source system for deploying and operating permissioned distributed ledgers whose state updates are recorded on a blockchain.

A HLF network is composed of peers, who maintain a key-value store that is updated through transactions on the underlying blockchain, and an ordering service (i.e., a consensus protocol). Because updates to the ledger's state (i.e., VAMS's log) are recorded on the underlying blockchain that is append-only, the ledger's state benefits from availability guarantees against the log equivocating as a log server that equivocates results in a fork of the blockchain. Integrity guarantees against tampering of the log are also guaranteed as changes to the ledger's state appears on the blockchain, which can be replayed or queried through a key history function.

Peers have identities in the form of X.509 public-key certificates and a *Membership Service Provider* (a PKI). These identities allow peers to be split up into

organizations on the network (e.g., agents, data providers, ...). This provides a way of implementing basic access control for operations on the network.

Updating the state of the ledger requires *endorsing peers* to execute *chaincode* (smart contracts) and sign the transaction containing the resulting state update. This ensures that an endorsing peer can be held accountable for the transactions they endorse; for example, the requests they make as agents or the requests they accept as data providers.

Transactions are then sent to the ordering service that packages them into blocks with which *validating peers* update the state of the ledger. Only endorsing peers are required to execute code for a transaction, so other peers do not handle any computational burden other than receiving events from the network. The endorsement mechanism also allows for endorsement policies that limit which peers can invoke or sign transactions for a certain chaincode, for example, based on their organization.

## Trillian

Trillian [174] is an open-source project that implements a generalization of Certificate Transparency [298] based on three components: a verifiable log, a verifiable map, and a log of map heads.

Trillian's verifiable log (not to be confused with VAMS's log) is an append-only log implemented as a Merkle tree that allows clients to efficiently verify that an entry is included in the log (with a proof showing the Merkle path to the tree's entry), detect log equivocation (i.e., conflicting tree heads), and verify that the log is append-only (through Merkle consistency proofs).

The verifiable map (i.e., VAMS's log) is a key-value store implemented as a sparse Merkle tree pre-populated with all possible keys as leaves (e.g., all $2^{256}$ possible SHA-256 hashes). Although a tree with $2^{256}$ unique leaves would not be practical to compute, only the non-empty leaves have to be computed because all others will have the same value (e.g., zero) [179]. Clients can then verify that a certain value is included (or not) in the map at any point in time, with proofs containing Merkle paths. Combining a verifiable log with a verifiable map leads

to a verifiable log-backed map, where the log contains an ordered set of operations applied to the map. Clients can then verify that the entries in the map they view are the same as those viewed by others by replaying the log and detecting any change in values of the key-value store.

The log of map heads records the root hash of the log, signed by the log's server so that if it equivocates there is a cryptographic proof that it has done so. Because Trillian does not involve a consensus protocol, it instead relies on gossip between clients (e.g., auditors and users performing *detect*) to detect misbehaving servers by comparing the views of the log that they have received.

As in the case of HLF, the fact that updates to the verifiable map (VAMS's log) are recorded on Trillian's append-only verifiable log provides availability guarantees again VAMS's log equivocating as this will lead to different three heads in the log of map heads, and integrity guarantees against tampering of VAMS's log as updates will appear on the underlying append-only Merkle tree that is Trillian's log.

### 5.4.2 Tagging log entries with common identifiers

In order to gain useful information from VAMS, users must be able to efficiently identify the log entries that contain information that is relevant to them. The values of log entries (i.e., records) will be encrypted for privacy reasons so this requires a mechanism that allows agents and data providers to derive keys that users can also compute only if the entry is relevant to them.

A strawman solution would require users to ask agents for the log entries relevant to them. This would reveal to agents which users are monitoring the log, and require users to trust agents that could lie about the log entries that are relevant to them. To remove the need for users to interact with agents, we use *common identifiers* that can be computed only with information known to a user, the data provider, and the agent that is involved in a request. By relying on shared information we remove the need for interaction, and by having the information known to not only the agent and user but also the data provider, a data provider that is not colluding with the agent can check that the tag will be correct.

We assume that agents and data providers refer to a given user using (private)

agent and data provider identifiers, $id_a$ and $id_{dp}$, which are also known to the user they correspond to but not to others. It is then possible to obtain common identifiers $id_c = Hash(id_a \| id_{dp} \| n)$ by using a secure hash function such as SHA-256, where $n$ is a *session identifier* that changes deterministically with every request involving the same pair $(id_a, id_{dp})$.

This ensures different requests involving the same parties are unlinkable as common identifiers will appear random, but allows users to check requests that are relevant to them communicating with agents or data providers, which also reduces the risk of information leaking. The private contents of the requests are then simply encrypted under the keys of users and auditors so that they can access them.

We assume that $id_a$ and $id_{dp}$ are pseudorandom strings like, for example, the German Electronic Identity Card that can be used for online authentication and has been analysed from a cryptographic point of view [299]. This is not unreasonable for purpose-built identifiers, and although it adds the burden of managing them, software such as password managers or data brokers could be relied on.

An agent or data provider could in principle leak $id_a$, $id_{dp}$, or common identifiers, but they could just as well leak the data attached to it in the first place. As we have remarked in the threat model, this cannot strictly be prevented as they control the data much like they must know the common identifiers to tag the log entries. Moreover, they would not gain anything from doing so.

### 5.4.3 Generating synthetic data and verifying statistics with MultiBallot

To allow published statistics to be publicly verified without incurring more of a privacy loss than is already caused by the statistics themselves, we introduce a way for auditors to generate a synthetic dataset $D_{priv}$ from the dataset $D$ used to compute the statistics. We call this randomized mechanism MultiBallot and denote $\mathscr{M}_n : D \mapsto D_{priv}$ the mapping of $|D|$ records in D to $n|D|$ shares in $D_{priv}$.

MultiBallot can be used to support either exclusively univariate statistics or multivariate statistics. Figure 5.2 illustrates how a record in a dataset $D$ that has $e$ binary elements can be transformed into *shares* of a synthetic dataset $D_{priv}$ accord-

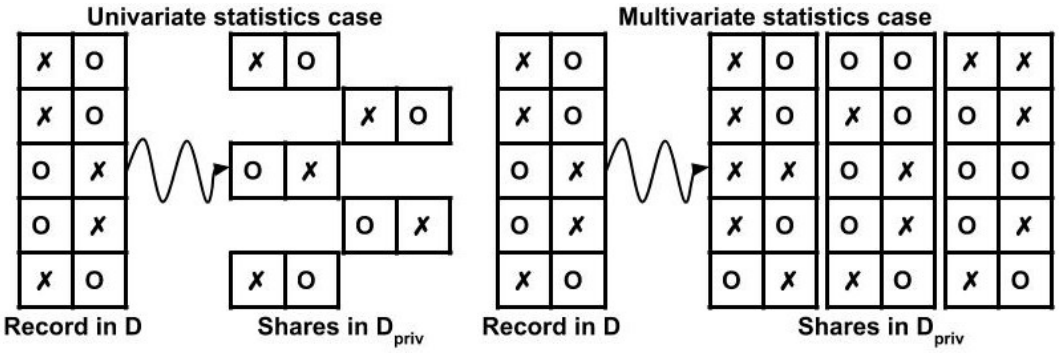


**Figure 5.2:** Example transformation of records in $D$ to shares in $D_{priv}$ for univariate and multivariate statistics. In the univariate case, the record is split into individual elements. In the multivariate case, the record is used to generate shares with the same number of elements that are then split from each other.

ing to the rules we describe for each case. Elements, in this case, could be attributes of a request for data in the law enforcement case (e.g., "request type:urgent/not urgent"), or of a patient in a medical study (e.g., "has gene X:yes/no").

In the univariate case, the elements of a record are simply split and shuffled, which allows univariate statistics to be re-computed without the possibility of computing multivariate statistics. In the multivariate case, the record in $D$ generates a combination of shares that introduce noise into the $D_{priv}$ but preserve the relative counts between elements in $D$ and therefore allow multivariate statistics to be re-computed.

In both cases, shares in $D_{priv}$ can be tagged so that the published statistics and their inputs can be verified. The privacy loss associated with the verification of the statistics through a public dataset $D_{priv}$ is also mitigated by ensuring $D_{priv}$ cannot be used to reconstruct $D$ or learn more information than what is learnt from the statistics in the first place, and (in the multivariate case) by generating it in a way that guarantees only a small expected privacy loss from having one's data included.

## Generating $D_{priv}$ for univariate statistics

Support for univariate statistics, and in particular a restriction to univariate statistics is important in settings where multivariate statistics are not used due to privacy concerns. This is for example the reason that IPCO reports contain only univariate statistics. When only univariate statistics are required, it is, therefore, important that

$D_{priv}$ should not be useful to compute multivariate statistics.

To handle the univariate case, shares in $D_{priv}$ are obtained by splitting records in $D$ into shares that each have one element only and then shuffling them, as in Algorithm 1. Each share is tagged with the element type and a unique share identifier $id_{share} = Hash(id_c \| i)$ (using a secure hash function like SHA-256) derived from the common identifier $id_c$ of the user and the index $i$ of the share.

**Algorithm 1:** $\mathscr{M}_1 : D \mapsto D_{priv}$ for univariate statistics.

**Input:** $D = \{r_i = (r_{i,1}, \ldots, r_{i,|r_i|}) | i \in [1, |D|], r_{i,j} \in \{\square, \boxtimes\}^2\}$
**Output:** $D_{priv} = \{shares \in \{\square, \boxtimes\}^2\}$
list $shares = []$
**for** $i = 1, \ldots, |D|$ **do**
  **for** $j = 1, \ldots, |r_i|$ **do**
    $shares = shares + r_{i,j}$
$shuffle(shares)$
**return** *shares*

Splitting up the shares ensures that the original record cannot be reconstructed without knowing the common identifiers used to generate the share identifiers, so the only thing that can be learnt is the individual counts of elements, which were already revealed by the statistics themselves. The univariate statistics (i.e., counts) will be unchanged as they do not depend on more than one element so they are unaffected by the split of elements and can be verified. Multivariate statistics on the other hand will not be computable.

## Generating $D_{priv}$ for multivariate statistics

In cases where multivariate statistics are needed (e.g., medical studies), records in $D$ are split into $n$ shares that have as many elements as the records. Our approach is inspired by Rivest's ThreeBallot voting scheme [69, 70], which works by giving voters three ballots (instead of one) and having them fill them according to a set of rules. We extend this to any odd number of ballots (combinations of shares in $D_{priv}$) where a vote for or against an element corresponds to having an attribute or not. We show how to use this as a way of generating a synthetic dataset ($D_{priv}$) that can be used to re-compute multivariate statistics originally computed using $D$.

Shares are generated such that the correct value of each element $e$ (i.e., $\boxtimes\square$ or $\square\boxtimes$) appears $s \in [1, k+1]$ times and the "false" value $\bar{e}$ (i.e., $\square\boxtimes$ or $\boxtimes\square$) appears $s-1$ times. The remaining elements in the shares are neutral and take the form $\boxtimes\boxtimes$ or $\square\square$. Algorithm 2 summarizes this process. Once the shares are generated, they are tagged as in the univariate case with a share identifier, split up, and shuffled.

**Algorithm 2:** Generating valid combinations of shares for $D_{priv}$ in the multivariate case.

**Input:** $n = 2k+1, k \in \mathbb{N}$
**Output:** Valid combinations of $n = 2k+1$ shares for elements $\square\boxtimes$ and $\boxtimes\square$
**for** *e in* $[\boxtimes\square, \square\boxtimes]$ **do**
  list $shares_e = []$
  **for** $s = 0, \ldots, k-1$ **do**
    tuple $shares = (e)$
    **for** $i = 0, \ldots, s-1$ **do**
      $shares = shares + (e) + (\bar{e})$
    **while** $length(shares) < 2k+1$ **do**
      $shares = shares + (\boxtimes\boxtimes) + (\square\square)$
    $shares_e = shares_e + permutations(shares)$
**return** $shares_{\boxtimes\square,n}$, $shares_{\square\boxtimes,n}$

**Algorithm 3:** $\mathscr{M}_n : D \mapsto D_{priv}$ for multivariate statistics.

**Input:** $D = \{r_i = (r_{i,1}, \ldots, r_{i,|r_i|}) | i \in [1, |D|], r_{i,j} \in \{\square, \boxtimes\}^2\}$
$n = 2k+1, k \in \mathbb{N}$
$shares_{\boxtimes\square}$
$shares_{\square\boxtimes}$
**Output:** $D_{priv} =$
list $shares = []$
**for** $i = 1, \ldots, |D|$ **do**
  tuple $shares_i$
  **for** $j = 1, \ldots, |r_i|$ **do**
    **if** $r_{i,j} = \boxtimes\square$ **then**
      $\_ \leftarrow shares_{\boxtimes\square}$
      $shares_i = shares_i + \_$
    **else**
      $\_ \leftarrow shares_{\boxtimes\boxtimes}$
      $shares_i = shares_i + \_$
$shuffle(shares)$
**return** *shares*

The number of valid combinations $B$ of shares, given by Equation 5.4.1, is obtained by considering the number of multiset permutations of the shares for each possible value of $s$, summing over $s$, and multiplying by a factor of 2 to account for both elements.

$$B = 2\sum_{s=1}^{k+1} \frac{(2k+1)!}{s!(s-1)!(k+1-s)!(k+1-s)!} \tag{5.4.1}$$

As an example we look at the record in Figure 5.2, which corresponds to $[\boxtimes\square, \boxtimes\square, \square\boxtimes, \square\boxtimes, \boxtimes\square]$. With $n = 3$, an element $\square\boxtimes$ can generate a combination of shares $(\square\boxtimes, \square\boxtimes, \boxtimes\square)$ and its 6 permutations, and $(\square\boxtimes, \boxtimes\boxtimes, \square\square)$ and its 3 permutations. Similarly, an element $\boxtimes\square$ can generate a combination of shares $(\boxtimes\square, \boxtimes\square, \square\boxtimes)$ and its 6 permutations, and $(\boxtimes\square, \boxtimes\boxtimes, \square\square)$ and its 3 permutations. For each element, a valid combination of shares is picked at random. In Figure 5.2 this results in $[(\boxtimes\square, \square\square, \square\square)$, $(\boxtimes\square, \boxtimes\square, \square\boxtimes)$, $(\boxtimes\boxtimes, \square\boxtimes, \square\square)$, $(\boxtimes\square, \square\boxtimes, \square\boxtimes)$, $(\square\boxtimes, \boxtimes\square, \boxtimes\square)]$. Split up in $D_{priv}$, these will look like three records of the form $[\boxtimes\square, \boxtimes\square, \boxtimes\boxtimes, \boxtimes\square, \square\boxtimes]$, $[\square\square, \boxtimes\square, \square\boxtimes, \square\boxtimes, \boxtimes\square]$, $[\square\square, \square\boxtimes, \square\square, \square\boxtimes, \boxtimes\square]$.

Valid combinations of shares can be pre-computed, so all that is required given $D$ is to randomly pick combinations of shares for all records. The shares are therefore picked from a distribution given in Equations 5.4.4 and 5.4.5 for each element, which is derived in more detail in Section 5.6, where it is also shown that the process of generating $D_{priv}$ incurs only a small expected privacy loss.

$$S_{\boxtimes\square} = S_{\square\boxtimes} = \sum_{s=1}^{k+1} (2s-1) \frac{(2k+1)!}{s!(s-1)!(k+1-s)!(k+1-s)!} \tag{5.4.2}$$

$$S_{\boxtimes\boxtimes} = S_{\square\square} = 2\sum_{s=1}^{k+1} (k+1-s) \frac{(2k+1)!}{s!(s-1)!(k+1-s)!(k+1-s)!} \tag{5.4.3}$$

$$\Pr(\boxtimes\square) = \Pr(\square\boxtimes) = \frac{S_{\boxtimes\square}}{(2k+1)B} \tag{5.4.4}$$

$$\Pr(\boxtimes\boxtimes) = \Pr(\square\square) = \frac{S_{\boxtimes\boxtimes}}{(2k+1)B} \tag{5.4.5}$$

The level of privacy depends on the original distribution of elements in $D$ and

the number of shares generated (i.e., the parameter $k$). This means that the level of privacy can be tuned by varying $k$ (see Section 5.6). Reconstructing a record in $D$ will also be highly improbable as given even all but one of the shares generated from a specific record, finding the correct last share will be improbable as no link will be revealed by the share identifiers. Verifying multivariate statistics will be possible (as we will see next), as well as their inputs by using the share identifiers.

## Verifying statistics with MultiBallot

Verifying univariate statistics is straightforward because this just involves counting the number of occurrences of an element having values 0 or 1 in $D_{priv}$, which will be the same as the number of occurrences in $D$. Because shares in $D_{priv}$ are tagged with an element type, the total number of shares that correspond to a specific element is also the same as the number of records in $D$ that include that element.

The multivariate case is a bit more involved and we show how to do it explicitly for association rule mining, although a similar approach could be used for other ways of computing multivariate statistics. This allows the results of an analysis of $D$ to be estimated from $D_{priv}$ by computing a matrix populated with the expected counts of shares in $D_{priv}$, which translates between $D$ and $D_{priv}$.

Association rule mining [86] is one of the most commonly used approaches to identify *if-then* rules and relationships between variables in large datasets. Given an element set $E$ of binary elements of a record and a dataset $D$ of records containing elements that form a subset of $E$, rules such as $\varepsilon \Rightarrow \varepsilon'$ where $\varepsilon, \varepsilon' \subseteq E$ are used to find interesting relationships between variables; for example, linking a set of genes with a particular disease. Two measures are commonly used to select interesting rules: *support* and *confidence*.

Support, defined in Equation 5.4.6, indicates how frequently a subset of elements appears in the dataset; that is, the proportion of records $R \in \delta$ (where $\delta \subseteq D$) that contain a subset of elements $\varepsilon \in E$. Confidence, defined in Equation 5.4.7, indicates how often a rule is found to be true. Given a rule $\varepsilon \Rightarrow \varepsilon'$, it is defined using

the support of the rule $\varepsilon \Rightarrow \varepsilon'$ and the support of $\varepsilon$.

$$supp(\varepsilon) = \frac{|\{R \in \delta : \varepsilon \in R\}|}{|\delta|} \tag{5.4.6}$$

$$conf(\varepsilon \Rightarrow \varepsilon') = \frac{supp(\varepsilon \Rightarrow \varepsilon')}{supp(\varepsilon)} \tag{5.4.7}$$

Our goal is to estimate the true counts of $\varepsilon$, $\varepsilon'$ and $\varepsilon \cup \varepsilon'$ in $D$ based on observations from $D_{priv}$, which also contains noise in the form of elements $\bar{e}$. Computing the support and confidence measures defined above is then straightforward. This process is often referred to as *support recovery*. For simplicity, we represent both the original records and the shares as bitstrings. For example, the record and shares in Figure 5.2 can be represented as $[10,10,01,01,10]$ and $[(10,10,11,10,01),(00,10,01,01,10),(11,01,00,01,10)]$. This is the same as the previous notation with $\boxtimes = 1$ and $\square = 0$.

We define $o_D$ and $o_{D_{priv}}$, which contain the number of occurrences of all possible bitstring permutations in $D$ and $D_{priv}$ in Equation 5.4.8. (The number of occurrences may be 0 for some permutations.)

$$o_D,\ o_{priv} = \begin{bmatrix} \#[0] \\ \vdots \\ \#[2^t - 1] \end{bmatrix}_{D, D_{priv}} \tag{5.4.8}$$

We also define $M$ in Equation 5.4.9. This matrix stores the expected bitstring occurrences $\mathbb{E}(\#[s])$ of any bitstring $s \in [0, 2^t - 1]$ in $D_{priv}$ (i.e., $\mathbb{E}[o_{priv}]$) for all possible bitstring permutations and a fixed number of bits $t$. The value $\mathbb{E}(\#[s])$ is obtained from the distribution of shares given in Equations 5.4.4 and 5.4.5.

$$M = \begin{bmatrix} \mathbb{E}[\#[0]]_0 & \dots & \mathbb{E}[\#[0]]_{2^t-1} \\ \vdots & \vdots & \vdots \\ \mathbb{E}[\#[2^t-1]]_0 & \dots & \mathbb{E}[\#[2^t-1]]_{2^t-1} \end{bmatrix} \tag{5.4.9}$$

A relation between $o_{priv}$, $M$, and $o_D$, can be established from the fact that for each record in $D$ its elements contribute an expected amount of each element in

$D_{priv}$. Therefore, the number of occurrences of any given bitstring $s$ in $o_{priv}$ is, on expectation, the sum of the expected amount of that bitstring due to a bitstrings in $D$ times the number of times these bitstrings (denoted $\delta$) occurred in D, as in Equation 5.4.10. Thus, we have that $\mathbb{E}[o_{priv}]$ is simply the result of multiplying $M$ with $o_D$, as in Equation 5.4.11.

$$\mathbb{E}[\#[s]_{D_{priv}}] = \sum_{\delta=0}^{2^t-1} \mathbb{E}[\#[s]]_{\delta} \cdot \#[\delta]_D \tag{5.4.10}$$

$$\mathbb{E}[o_{priv}] = M \cdot o_D \tag{5.4.11}$$

For the public to verify statistics using values in $o_{priv}$ obtained from $D_{priv}$, the aim is to reverse this process and infer the counts in $o_D$. Alongside $D_{priv}$, $M$ can also be safely released as it does not contain any private information. Computing its inverse $M^-1$, we can therefore estimate $o_D$ by multiplying $o_{priv}$ with $M^-1$, as in Equation 5.4.12.[3]

$$o_D \approx M^{-1} \cdot o_{priv} \tag{5.4.12}$$

Based on the inferred value of $o_D$, the support and confidence measures for any element sets $\varepsilon$, $\varepsilon'$ can then be computed in the usual way. This allows statistics to be accurately verified, as we show in the evaluation of our implementation in Section 5.7.

$D_{priv}$ is used only for verification of the reported statistics, leaving plenty of room for minimizing its information content. If $D$ is composed of records with a large number of elements, but only a few of these have interesting relations that are relevant in the published statistics, then only these need to be included in $D_{priv}$. This can significantly reduce the size of $D_{priv}$ compared to $D$.

Our technique can in certain cases support statistics involving continuous variables. During the rule mining phase, the researcher may need to examine the exact

[3] If $M$ is not invertible, it can be made invertible with a change that would not significantly affect the results.

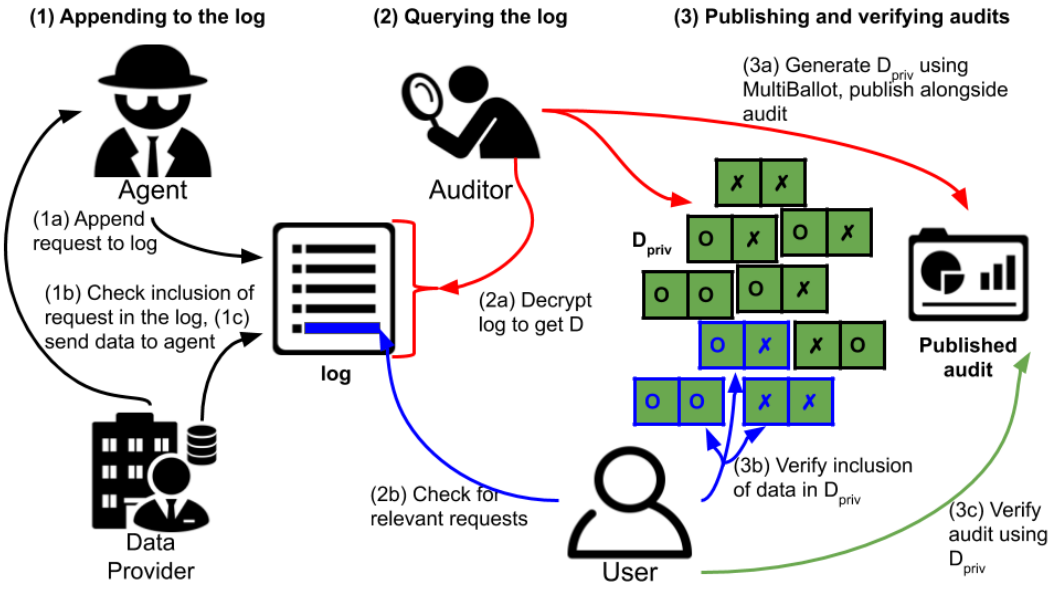


**Figure 5.3:** The three stages in the operation of VAMS. Red, blue and green boxes indicate information available to auditors, users, and the public. Similarly, red, blue and green arrows indicate operations that require being an auditor, a user, or anybody.

values (e.g., blood pressure) but once a relevant threshold is identified, all the values can be expressed as larger or smaller than that threshold (e.g., blood pressure over 140/90mmHg). This practice is common in machine learning algorithms. For example, C4.5 (an extension of ID3 [300]) builds decision trees from sets of data samples containing both continuous and discrete attributes. Alternatively, continuous variables can be split into multiple binary elements.

## 5.5 Operating VAMS

VAMS involves three stages that are illustrated in Figure 5.3: appending requests to the log, querying the log for audits, and publishing and verifying audits. In this section, we describe each stage and argue that VAMS achieves its transparency and privacy goals.

### 5.5.1 Appending to the log

As part of *request*, agents append requests to the log as values tied to the relevant common identifiers. If the value is temporarily sensitive then a cryptographic commitment can be used to ensure that correct logging can be verified after the fact.

Our transparency goals require that the requests be auditable by the parties they pertain to (i.e., users performing *check* and auditors performing *audit*) without relying on other parties. Our privacy goals also require that the private contents of the requests are not visible to any other parties, and that information cannot be inferred about the requests by linking them with other requests, users, agents, or data providers. This is assured by encrypting the log entries so that only auditors and relevant users can decrypt them, and using unlinkable common identifiers.

Once a request is appended to the log it can be answered by a data provider. A user acting as a data provider that is relying on a data broker to answer requests for their data could check if the data broker was misrepresenting their preferences by checking access' to their data themselves, or simply receiving notifications for the requests appended to the log that the data broker accepts.

### 5.5.2 Querying the log

Once requests are logged, users and auditors can verify that the log servers are not malicious by performing *detect*, then perform *audit* and *check* as required. Both users and auditors can assure themselves that the information obtained from the log is correct due to the availability and integrity properties of the log, and audit the entries of the log. Auditors perform their task over the entire log, or a subset of the log. Users look only for specific requests by iterating over their common identifiers until no request is found to determine the possible requests relevant to them.

Users that do not wish to take on this task can choose to outsource it to a data broker. A downside of this is that the data broker must then be trusted with the private identifiers tied to requests that the user positively answers. No other trust is required as VAMS allows users to check the activity of their broker, which can be logged and audited under the same guarantees as other log entries.

### 5.5.3 Publishing and verifying audits

Auditors perform *publish* to release the statistics computed as a result of their audit. Examples of what might be published are the statistics provided by the IPCO in its annual report [301] (e.g., the number of urgent requests) or the results of a medical

study (e.g., the association of some attribute with a disease).

Our transparency goals require that these statistics be verifiable, but for operational and privacy reasons the original data used to compute statistics cannot be published. Instead, the synthetic dataset $D_{priv}$ generated by MultiBallot (or its hash) can be published on the log and used to verify the statistics by users performing *monitor*, as we have shown in Section 5.4.3. Users whose data was used to compute the statistics can verify the inclusion of their data in $D_{priv}$ as part of *monitor*.

Assuming all users will take on the burden of verifying statistics is unrealistic, but the system does not require them to do so. Users that wish to check for access to their data can do so regardless of others. They can also verify published statistics even if their data was not used in the computation. Verifying the integrity of a dataset benefits from more users doing so, but a limited amount of users doing so will already be beneficial. Others could rely on data brokers, which would be acting in a way similar to organizations that currently perform Freedom of Information requests.

## 5.6 Achieving Transparency and Privacy Goals

### 5.6.1 Goal $T1$: log availability

We have assumed that agents and data providers do not collude so requests will be logged so availability only requires online log servers. The remaining threat is then a malicious log server that equivocates, in which case users and auditors could perform *detect* as follows.

In the HLF case, equivocation would result in a fork of the blockchain. Both the main chain and the forked chain would be visible, so equivocation can be detected.

In the Trillian case, a log server that equivocates would have to produce signed tree heads and Merkle consistency proofs for the alternative Merkle trees. Different Merkle consistency proofs leading from the same Merkle tree generate different views of the log, but these differing logs can no longer accept the same Merkle consistency proofs to extend the logs because the leaves are different. As tree heads

are signed by the log server, two inconsistent tree heads can be used as evidence to implicate the log server [247, 29].

### 5.6.2 Goal $T2$: log integrity

This argument is based on the fact that updates to the key-value store are recorded on an append-only blockchain (for HLF) or a verifiable log (for Trillian), resulting in VAMS's log being tamper-evident.

(*HLF case*) We rely on the underlying blockchain that records state updates. Auditors can use the key history function to obtain the state updates that have modified the value of a key. If they do not trust the integrity of that function (the code for which is public), they can replay the blockchain's transactions to detect a party's misbehaviour as they will have signed the relevant transactions.

(*Trillian case*) We rely on the underlying Merkle trees and the Merkle consistency proofs that give the append-only property of the trees. If a malicious party has tried to tamper with requests, they will have to update a request value, which will appear in the append-only log. If the log server produces a new tree head for a tree that modifies requests in the tree associated with the previous tree head, it will be evident as there cannot be a Merkle consistency proof between the two trees. Similarly, if a leaf of an existing tree is removed, the Merkle root of the tree will no longer match the leaves.

Auditors can then perform *audit* by querying the state of the ledger or log-backed map containing the requests (which are encrypted under their public keys) and performing their analysis. Requests that cannot be decrypted can be classed as invalid and reported. The same argument can be used for users performing *check*.

### 5.6.3 Goal $T3$: verifiability of inputs to audits

In the case of a user performing *monitor*, we again have that the user has a correct and complete view of the log by following through the arguments previously presented. A malicious auditor could nonetheless perform *publish* maliciously, publishing incorrect statistics or the wrong dataset, but this would be detected by a user performing *monitor*. A user that was included in the used dataset $D$ used can check

the integrity of the transformed dataset $D_{priv}$, identifying their shares using the share identifier derived from their common identifiers and checking that they reconstruct their original record.

### 5.6.4 Goal $T4$: verifiability of published audits

Using $D_{priv}$, any user can compute the same statistics that are contained in the published audits in the way that was described in Section 5.4.3. If the results are acceptably close then they can conclude that the statistics computed on $D$ were correctly computed.

### 5.6.5 Goal $T5$: transparency of the system

All the information that auditors require is by definition the information that is logged, which they can access with access only to VAMS, and without interaction with any other party. For users, the information relevant to themselves will be accessible by finding and decrypting the records relevant to them, which does not require the help of any other party, and audits must be made available by the auditors, but a hash of $D_{priv}$ on the log can assert the integrity of $D_{priv}$. Verifying the statistics from $D_{priv}$ and the inclusion of their data does not require any interaction either.

### 5.6.6 Goal $P1$: The log itself does not reveal any sensitive information

The values of the log entries are encrypted so that no party can gain any information from these unless they have the decryption key controlled by either a relevant auditor or user. Identifying related log entries could reveal sensitive information but log entries are unlinkable. It is not possible to link either common identifiers (outputs of a hash function such as SHA-256) or the values of log entries (outputs of a secure encryption scheme) together.

### 5.6.7 Goal $P2$: verifying an audit is privacy preserving

Verifying an audit involves verifying known inputs (i.e., privately known records in $D$) and the public outputs (i.e., statistics computed on records that are released).

Verifying the inputs only involves checking that known shares in $D_{priv}$ reconstruct a known record in $D$. No privacy loss can occur by doing this because the record is, by assumption, already known. We, therefore, focus on arguing that the access to $D_{priv}$, which is necessary to verify the statistics, does not lead to a greater privacy loss than the release of the statistics themselves.

The privacy risk associated with the release of $D_{priv}$ comes in three forms.

First, the share identifiers used by users to verify the inputs to the statistics could reveal links between the shares, or the common identifiers used as their inputs. For this, we rely again on the security of the hash function used to generate the share identifiers. Taking SHA-256 as providing sufficiently random outputs, it will not reveal links between the shares, and taking it as pre-image resistant it will not reveal the common identifiers used as input.

Second, $D_{priv}$ itself may leak sensitive information, allowing a record to be reconstructed or allowing the presence of a record to be inferred more than already possible from the publicly released statistics.

Third, $D_{priv}$ could be used to compute not only the statistics released through the published audit but also other statistics that were not intended to be released.

To verify univariate statistics, $D_{priv}$ needs only to contain single element shares so $D_{priv}$ can only be used to compute univariate statistics. Moreover, if some elements of the records in $D$ were considered too sensitive to publish statistics about, they can simply be excluded from $D_{priv}$ without affecting the ability of users to compute the statistics that were published. This means that only the published statistics and their inputs can be verified, so there is no risk of privacy loss from allowing the statistics to be verified.

In the case of multivariate statistics, we rely on the fact that generating $D_{priv}$ incurs only a small expected privacy loss (Theorem 2) and that, given $D_{priv}$, it is not possible to reconstruct records on the log (Theorem 1). This ensures that publishing $D_{priv}$ does not enable an adversary to infer whether or not a certain log entry was in $D$ and that given some information about a record in $D$ (i.e., some shares from that record), the remaining shares cannot be identified using $D_{priv}$.

## Bounds on ballot reconstruction attacks

*Theorem* 1. The probability that an adversary who knows $\alpha \in [1,2k]$ shares of a ballot can reconstruct the entire ballot is $\Pr(Reconstruct)) = (1 - \Pr(Valid)^e)^{\binom{(2k+1)r-\alpha}{2k+1-\alpha}-1}$.

*Proof.* Each element of a share can take the form of a *single* (i.e., $\boxtimes\square$ or $\square\boxtimes$) or a *double* (i.e., $\boxtimes\boxtimes$ or $\square\square$). Initially, we restrict ourselves to ballots of one element, so a share simply corresponds to an element. Given a ballot of $n = 2k+1$ elements, it must contain $s \in [1,k+1]$ singles that correspond to the record, and thus $s-1$ copies of the other single. The rest of the elements are filled up using $k+1-s$ of each double. The number of permutations, denoted $P(s)$, of a ballot with $s$ singles corresponding to the record using the standard formula for multiset permutations, which takes into account repeated elements in a ballot, is given in Equation 5.6.1.

$$P(s) = \frac{(2k+1)!}{s!(s-1)!(k+1-s)!(k+1-s)!} \tag{5.6.1}$$

To compute the total number of possible ballots $B$, given in Equation 5.6.2, we just sum over $s$ to add up ballots corresponding to each number of singles matching the record and multiply by a factor 2 as ballots are symmetric under an interchange of singles.

$$B = 2\sum_{s=1}^{k+1} P(s) \tag{5.6.2}$$

This result can be used to compute the probability distribution of the shares by counting their appearances in the $(2k+1)B$ shares that make up all the possible ballots. This amounts to taking, for each element, the sum of permutations of a ballot weighted by the number of appearances of that share in the ballot, and taking into account the fact that doubles appear the same amount of times in ballots corresponding to either record, and singles appear either $s$ or $s-1$ times depending on whether they match the record. We denote the number of $\boxtimes\square$, $\square\boxtimes$, $\boxtimes\boxtimes$ or $\square\square$ shares by $S_{\boxtimes\square}$, $S_{\square\boxtimes}$, $S_{\boxtimes\boxtimes}$ or $S_{\square\square}$, which are given in Equations 5.6.3 and 5.6.4. The probability of each share, given in Equations 5.6.5 and 5.6.6, is then obtained by

dividing the number of shares for each form by the total number of shares.

$$S_{\boxtimes\square} = S_{\square\boxtimes} = \sum_{s=1}^{k+1}(2s-1)P(s) \tag{5.6.3}$$

$$S_{\boxtimes\boxtimes} = S_{\square\square} = 2\sum_{s=1}^{k+1}(k+1-s)P(s) \tag{5.6.4}$$

$$\Pr(\boxtimes\square) = \Pr(\square\boxtimes) = \frac{S_{\boxtimes\square}}{(2k+1)B} \tag{5.6.5}$$

$$\Pr(\boxtimes\boxtimes) = \Pr(\square\square) = \frac{S_{\boxtimes\boxtimes}}{(2k+1)B} \tag{5.6.6}$$

With the probability distribution obtained we can obtain the probability $\Pr(Valid)$, given in Equation 5.6.7, of the event $V$ that occurs when randomly chosen shares form a valid ballot, by summing over the possible ballots weighted by the probability of each share. More generally, when shares involve $e$ elements the probability is $\Pr(Valid)^e$.

$$\begin{aligned}\Pr(Valid) = 2\sum_{s=1}^{k+1} P(s)\Pr(\boxtimes\square)^s \Pr(\square\boxtimes)^{s-1}\cdot \\ \Pr(\boxtimes\boxtimes)^{k+1-s}\Pr(\square\square)^{k+1-s}\end{aligned} \tag{5.6.7}$$

The above is the probability of success for a weak adversary that starts with no prior knowledge and wants only to reconstruct a ballot, regardless of whether it belongs to someone. An adversary that knows up to $\alpha \in [1,2k]$ shares of a ballot and wishes to figure out the last shares required to reconstruct that ballot chooses $2k+1-\alpha$ other shares from the dataset, giving $\binom{(2k+1)r-\alpha}{2k+1-\alpha}$ possibilities, where $r$ is the number of records from which we subtract 1 as there must be at least one valid ballot. This gives Equation 5.6.8, which expresses the probability of success $\Pr(Reconstruct)$ of that adversary.

$$\Pr(Reconstruct)) = (1-\Pr(Valid)^e)^{\binom{(2k+1)r-\alpha}{2k+1-\alpha}-1} \tag{5.6.8}$$

□

**Table 5.3:** Upper bounds on the number of elements in 3Ballot and 5Ballot shares such that the probability of a successful reconstruction is less than 0.01%. The numbers in brackets next to the scheme indicate the number of shares known to the adversary and the numbers in brackets next to the number of elements indicate the probability of success.

| Scheme | 10 users | 100 users | 1000 users | 10 000 users |
|---|---|---|---|---|
| 3Ballot (1) | 3 $(3 \cdot 10^{-5})$ | 6 $(5 \cdot 10^{-13})$ | 10 $(6 \cdot 10^{-10})$ | 14 $(1 \cdot 10^{-7})$ |
| 3Ballot (2) | 1 $(8 \cdot 10^{-5})$ | 2 $(2 \cdot 10^{-12})$ | 4 $(2 \cdot 10^{-10})$ | 6 $(5 \cdot 10^{-9})$ |
| 5Ballot (1) | 6 $(4 \cdot 10^{-6})$ | 11 $(5 \cdot 10^{-20})$ | 17 $(3 \cdot 10^{-12})$ | 23 $(2 \cdot 10^{-13})$ |
| 5Ballot (4) | 1 $(5 \cdot 10^{-5})$ | 2 $(3 \cdot 10^{-9})$ | 3 $(2 \cdot 10^{-17})$ | 5 $(3 \cdot 10^{-7})$ |

Table 5.3 gives an upper bound on elements that can be included in shares while maintaining a probability of a reconstruction attack under 0.01 when an adversary knows one share or all but one share. Different bounds can be chosen depending on the acceptable probability of a reconstruction. In practice, only a few elements may be relevant to the results of an audit or study, and only those need to be published for the relevant statistics to be publicly verifiable.

It is also important to note that we have modelled an attacker who completes a partial ballot by picking random shares in $D_{priv}$. In reality, an attacker may of course have better chances of reconstructing a ballot by inferring the remaining shares, particularly if they already know most of the ballot's shares, but this is done regardless of the availability of $D_{priv}$.

Finally, we have assumed statistical independence between elements, which may not always be true. ThreeBallot with correlated ballots was studied by Strauss [75] who showed that even heavily correlated elements had only a minor effect on the security of the scheme.

## Bounds on the expected privacy loss from membership of $D_{priv}$

To quantify the loss of privacy from membership of $D$ and, therefore, the published $D_{priv}$, we consider the privacy loss variable $\mathscr{L}^{\theta}_{M(D),M(D')}$, defined in Equation 5.6.9. This variable quantifies the privacy loss incurred by observing an output $\theta$ of the mechanism $M$, based on how much more (or less) likely that output is when $M$ takes $D$ as input rather than $D'$.

$$\mathscr{L}^{\theta}_{M(D),M(D')} = \ln\left(\frac{\Pr[\mathscr{M}(D)=\theta]}{\Pr[\mathscr{M}(D')=\theta]}\right) \tag{5.6.9}$$

If $M$ satisfied the definition of differential privacy, this would be equivalent to saying that the privacy loss variable would be bounded [302]. MultiBallot, however, cannot satisfy differential privacy because the share counts (which would define $D_{priv}$) that result from running $M$ on $D$ or $D'$ cannot ever match unless $D = D'$. This is because the share counts of ballots generated from different elements cannot be equal.

We, therefore, define in Definition 2 a relaxed alternative to differential privacy, replacing the distribution over outputs with an expected output. As we will show in Theorem 2, MultiBallot satisfies this definition such that given two datasets of the same size, $D$ and $D'$, differing in one entry, the expected outputs of $M_n$ running on either database remain close.

*Definition* 2 ($\zeta$-expected privacy loss). A randomized algorithm $M$ with domain $\mathbb{N}^{|\chi|}$ has a bounded expected privacy loss if given two input databases $D$, $D' \in \mathbb{N}^{|\chi|}$ where $D$ and $D'$ differ only in one element, there exists $\zeta \in \mathbb{R}$ such that

$$\zeta \geq \ln\left(\frac{\mathbb{E}[M(D)]}{\mathbb{E}[M(D')]}\right). \tag{5.6.10}$$

As in the case of differential privacy, $\zeta$-expected privacy loss also provides group privacy.

*Lemma* 1 (Bounded expected group privacy loss). Let $D$ and $D'$ be two databases that differ in $e$ elements. If a randomized algorithm $M$ satisfies bounded expected privacy loss, then we have that

$$e\zeta \geq \ln\left(\frac{\mathbb{E}[M(D)]}{\mathbb{E}[M(D')]}\right). \tag{5.6.11}$$

*Proof.* Iterating over Definition 2, if $M$ has bounded expected privacy loss then the expected privacy loss due to any single element on the output of $M$ is bounded by $\zeta$. Thus, the impact of any $e$ elements is bounded by $e\zeta$. □ □

We now prove in Theorem 2 that Multiballot satisfies this definition and compute values of *zeta* for different database sizes ($|D| = 10, 100, 1000, 10000$) and schemes (3Ballot, 5Ballot).

*Theorem* 2. The MultiBallot share generation mechanism $M_n : D \mapsto D_{priv}$ satisfies $\zeta$-expected privacy loss with $\zeta = \ln\left(\frac{|\boxtimes\square|_{D_{priv}}}{|\boxtimes\square|_{D_{priv}} - \sum_{s=1}^{k+1}\Pr(s\cdot\boxtimes\square)}\right)\Big|_{r_{\boxtimes\square}=0, r_{\square\boxtimes}=r}$.

*Proof.* Consider two databases $D$ and $D'$ that contain $r$ single element records and differ in one record. Without loss of generality, we take $D$ to contain $r^D_{\boxtimes\square}$ records of the form $\boxtimes\square$ and $r^D_{\square\boxtimes}$ records of the form $\square\boxtimes$ and $D'$ to contain $r^{D'}_{\boxtimes\square} = r^D_{\boxtimes\square} - 1$ records of the form $\boxtimes\square$ and $r^{D'}_{\square\boxtimes} = r^D_{\square\boxtimes} + 1$ records of the form $\square\boxtimes$.

Our aim is to determine and compare the share counts in $D_{priv} \leftarrow M_n(D)$ and $D'_{priv} \leftarrow M_n(D')$. Counting the different shares in $D_{priv}$ amounts to considering the probability that the ballot generated from a record will have $s$ shares of one form. This is given by the number of ballot permutations for a given $s$ over all possible ballots for that record, as expressed in Equation 5.6.12. Summing over $s$ gives the expected count for each share of ballots generated from a record, which we denote $|share|_{record}$. (The same analysis holds for $D'_{priv}$.)

$$\Pr(s\cdot(share = record)) = \frac{2}{B}P(S) \tag{5.6.12}$$

$$|\square\boxtimes|_{\square\boxtimes} = |\boxtimes\square|_{\boxtimes\square} = r^D_{\boxtimes\square}\sum_{s=1}^{k+1}\Pr(s\cdot\boxtimes\square)\,s \tag{5.6.13}$$

$$|\boxtimes\square|_{\square\boxtimes} = |\square\boxtimes|_{\boxtimes\square} = r^D_{\boxtimes\square}\sum_{s=1}^{k+1}\Pr(s\cdot\boxtimes\square)\,(s-1) \tag{5.6.14}$$

$$|\boxtimes\boxtimes|_{\square\boxtimes} = |\boxtimes\boxtimes|_{\boxtimes\square} = r^D_{\boxtimes\square}\sum_{s=1}^{k}\Pr(s\cdot\boxtimes\square)\,(k+1-s) \tag{5.6.15}$$

$$|\boxtimes\boxtimes|_{\square\boxtimes} = |\square\square|_{\boxtimes\square} = r^D_{\boxtimes\square}\sum_{s=1}^{k}\Pr(s\cdot\boxtimes\square)\,(k+1-s) \tag{5.6.16}$$

Adding the contributions from all the records together gives the total expected share count for each type of share in $D_{priv}$.

$$
\begin{aligned}
|\boxtimes\square|_{D_{priv}} &= r^{D}_{\boxtimes\square} \sum_{s=1}^{k+1} \Pr(s\cdot\boxtimes\square)\, s \\
&\quad + r^{D}_{\square\boxtimes} \sum_{s=1}^{k+1} \Pr(s\cdot\square\boxtimes)\,(s-1)
\end{aligned}
\tag{5.6.17}
$$

$$
\begin{aligned}
|\square\boxtimes|_{D_{priv}} &= r^{D}_{\boxtimes\square} \sum_{s=1}^{k+1} \Pr(s\cdot\boxtimes\square)\,(s-1) \\
&\quad + r^{D}_{\square\boxtimes} \sum_{s=1}^{k+1} \Pr(s\cdot\square\boxtimes)\, s
\end{aligned}
\tag{5.6.18}
$$

$$
\begin{aligned}
|\square\square|_{D_{priv}} = |\boxtimes\boxtimes| &= r^{D}_{\boxtimes\square} \sum_{s=1}^{k} \Pr(s\cdot\boxtimes\square)\,(k+1-s) \\
&\quad + r^{D}_{\square\boxtimes} \sum_{s=1}^{k} \Pr(s\cdot\square\boxtimes)\,(k+1-s)
\end{aligned}
\tag{5.6.19}
$$

We obtain a similar result for $D'_{priv}$ using the fact that $r^{D'} = r^{D'}_{\boxtimes\square} - 1$.

$$
\begin{aligned}
|\boxtimes\square|_{D'_{priv}} &= \left(r^{D}_{\boxtimes\square} - 1\right) \sum_{s=1}^{k+1} s\cdot\Pr(s\cdot\boxtimes\square) + \\
&\qquad \left(r^{D}_{\square\boxtimes} + 1\right) \sum_{s=1}^{k+1} (s-1)\cdot\Pr(s\cdot\square\boxtimes) \\
&= |\boxtimes\square|_{D_{priv}} - \\
&\qquad \left(\sum_{s=1}^{k+1} s\cdot\Pr(s\cdot\boxtimes\square) - \sum_{s=1}^{k+1} (s-1)\cdot\Pr(s\cdot\square\boxtimes)\right) \\
&= |\boxtimes\square|_{D_{priv}} - \sum_{s=1}^{k+1} \Pr(s\cdot\boxtimes\square)
\end{aligned}
\tag{5.6.20}
$$

$$
|\square\boxtimes|_{D'_{priv}} = |\square\boxtimes|_{D_{priv}} + \sum_{s=1}^{k+1} \Pr(s\cdot\square\boxtimes) \tag{5.6.21}
$$

$$
|\square\square|_{D'_{priv}} = |\square\square|_{D_{priv}} \tag{5.6.22}
$$

$$
|\boxtimes\boxtimes|_{D'_{priv}} = |\boxtimes\boxtimes|_{D_{priv}} \tag{5.6.23}
$$

Given the expected share counts in $D_{priv}$ and $D'_{priv}$ that we have derived, we can now compare them to obtain a bound $\zeta$ on the expected privacy loss.

**Table 5.4:** Values for the expected privacy loss parameters $\zeta$ and $e\zeta$ for different sizes of $D$. We take values of $e$ equal to the safe number of elements against reconstruction attacks taken from Table 5.3.

| **Scheme** | $\lvert D\rvert$ | $\zeta$ | $\exp(\zeta)$ | $e\zeta$ | $\exp(e\zeta)$ |
|---|---|---|---|---|---|
| 3Ballot | 10 | 0.36 | 1.43 | 1.08 ($e=3$) | 2.95 |
| 3Ballot | 100 | 0.03 | 1.03 | 0.18 ($e=6$) | 1.2 |
| 3Ballot | 1000 | 0.003 | 1.003 | 0.03 ($e=10$) | 1.03 |
| 3Ballot | 10,000 | 0.0003 | 1.0003 | 0.0042($e=14$) | 1.0042 |
| 5Ballot | 10 | 0.1335 | 1.143 | 0.801($e=6$) | 2.23 |
| 5Ballot | 100 | 0.0126 | 1.0127 | 0.1386 ($e=11$) | 1.149 |
| 5Ballot | 1000 | 0.00125 | 1.00125 | 0.02125 ($e=17$) | 1.0215 |
| 5Ballot | 10,000 | 0.000125 | 1.000125 | 0.002875 ($e=23$) | 1.0029 |

$$
\begin{aligned}
\zeta &= \max_{s\in shares} \ln\left(\frac{|s|_{D_{priv}}}{|s|_{D'_{priv}}}\right) \\
&= \max \ln\left(\frac{|\boxtimes\square|_{D_{priv}}}{|\boxtimes\square|_{D'_{priv}}}\right) \\
&= \max \ln\left(\frac{|\boxtimes\square|_{D_{priv}}}{|\boxtimes\square|_{D_{priv}} - \sum_{s=1}^{k+1}\Pr(s\cdot\boxtimes\square)}\right) \\
&= \ln\left(\frac{|\boxtimes\square|_{D_{priv}}}{|\boxtimes\square|_{D_{priv}} - \sum_{s=1}^{k+1}\Pr(s\cdot\boxtimes\square)}\right)\Bigg|_{r_{\boxtimes\square}=0, r_{\square\boxtimes}=r}
\end{aligned}
\tag{5.6.24}
$$

□

The final result from Equation 5.6.24 can be easily computed and is given for different values of $|D|$ in Table 5.4.

## 5.7 Implementation and Performance

We evaluate VAMS by comparing two implementations of the log based on HLF and Trillian (the code for which will be open-sourced after publication) and showing that statistics can be accurately verified with MultiBallot. Both log implementations

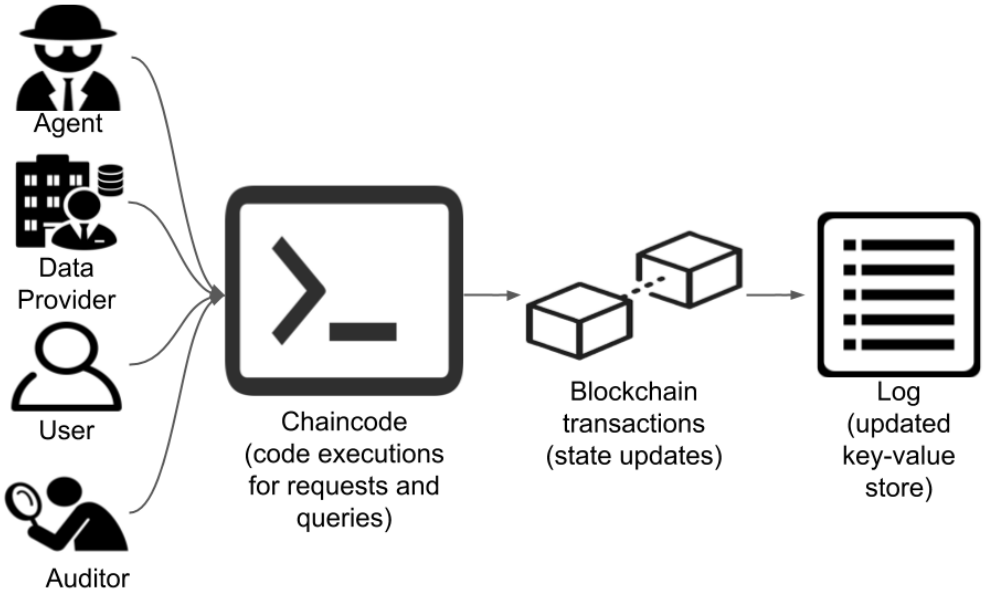


**Figure 5.4:** The HLF-based implementation.

are evaluated on very modest (and cheap) Amazon AWS t2.medium instances.[4]

### 5.7.1 Evaluating Hyperledger Fabric and Trillian based logs

#### Hyperledger Fabric based log

For our HLF-based log, we set up a test network of seven machines that represent four peers (an agent, a data provider, a user, and an auditor), an ordering service (an Apache Zookeeper service and a Kafka broker), and a client from which commands are sent. Log entries can be retrieved by querying specific common identifiers, and a key history function is also available to retrieve the state updates (i.e., transactions on the underlying blockchain) of a log entry. The execution of commands on the HLF network is summarized in Figure 5.4.

In this implementation, all peers are connected to one channel and there is one chaincode containing four functions that update the state of the ledger (as part of *request*), retrieve a range of key values (as part of *audit*), retrieve values for specific keys (as part of *check*) and retrieve a key's history (as part of *audit* and *check*).

Thus, auditors or users can check the transactions that updated the value of a key and easily determine the agent responsible for the update, as they will have

[4]Each instance has 2 vCPUs and 4GB of memory and is running Ubuntu Linux 16.04 LTS with Go 1.7, docker-ce 17.06, docker-compose 1.18, and Fabric 1.06 installed

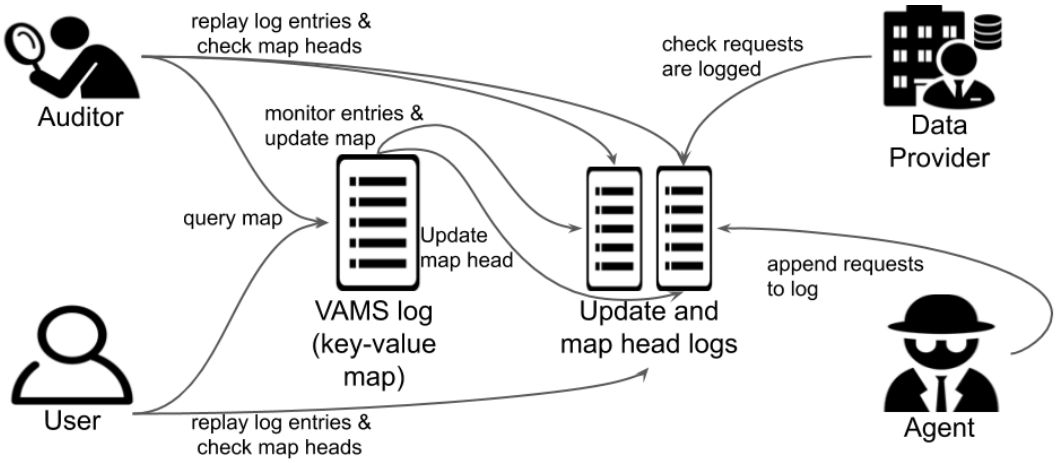


**Figure 5.5:** The Trillian-based implementation.

endorsed (i.e., signed) the transaction. Endorsement policies can require multiple signatures so they could hold multiple parties accountable. For example, if data providers were considered responsible for accepting invalid requests, they could be required to sign the corresponding request transactions. An ordering service of specific peers (e.g., auditors) could also be used to detect and flag invalid requests as they are initially processed (and endorsement policies are checked) before committing the requests. These are not present in our implementation, but give an idea of what improvements may be possible as Hyperledger Fabric undergoes continued development and implements further cryptographic tools.

## Trillian based log

Our second implementation of the log, illustrated in Figure 5.5, is based on Trillian's verifiable log-backed map. The map server (VAMS's log) monitors the log of updates for new entries and updates the map according to the new entries – common identifiers are used as the map's keys. It then periodically publishes signed map heads on the second verifiable log, solely responsible for keeping track of published signed map heads.

To perform *check*, users can query the map to efficiently check their possible common identifier values. The map will return a Merkle proof of non-inclusion for common identifiers that do not map to requests (i.e., the common identifier maps to 0), or a Merkle proof of inclusion for requests that the common identifiers do map

**Table 5.5:** Micro-benchmarks of basic operations for the Hyperledger Fabric and Trillian based implementations. The maximal throughput values are given for a batch size of 1 in the HLF case and a batch size of 300 in the Trillian case.

| **Measures** | HLF | Trillian |
|---|---|---|
| State update (average over 500 operations) | 65ms | 35ms |
| Request retrieval (average over 500 operations) | 66ms | 14ms |
| Max throughput | 40 | 102 |

to. Auditors performing *audit* can in turn check that the map is operated correctly by replaying all log entries, verifying that they correspond to the map heads on the second verifiable log.

## Performance evaluation

Table 5.5 presents benchmarks for state updates, state retrievals, and the maximal throughput for each system with a batch size of one. In both cases, the average for each operation is a few dozen milliseconds. For the HLF system, the results include the time required to create and submit 500 blocks; chaincode execution alone is under $10ms$. For state retrievals, HLF allows values to be retrieved for a range of keys. This operation scales linearly with the number of values retrieved and only requires one transaction.

Table 5.5 also includes the maximal throughput, which is 40 requests per second for the HLF system and 102 requests per second for the Trillian system. Figure 5.6 shows the throughput for different batch sizes. For the HLF-based log, the highest throughput is observed for smaller batch sizes. The bottleneck is simply the client sending requests. For the Trillian-based log, the batch size determines how many items at a time the map servers retrieve from the log to update the map's key values, until around batch size 300. The bottleneck is then the number of keys updated by the map server per second, and throughput levels out.

A greater throughput can be achieved with a larger batch size but having requests appear on the log sooner can be advantageous (e.g., in the case of urgent requests). In that case, a lower batch size is preferable, or a batch time-out that would ensure a request will appear after a time limit if the batch size limit is not

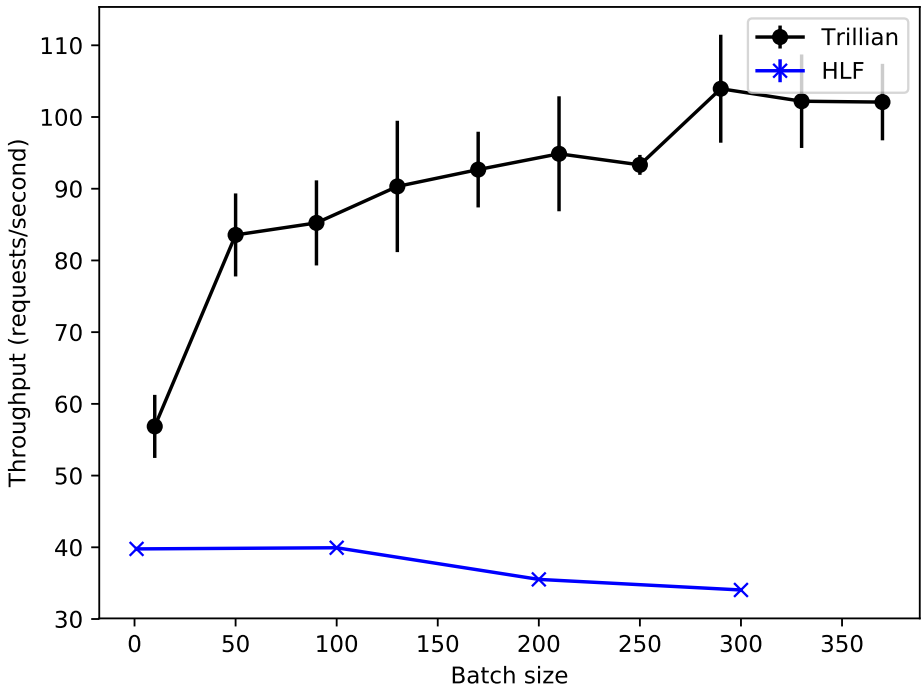


**Figure 5.6:** Throughput of both logs for different batch sizes.

reached.

Practically speaking, for example in the case of law enforcement access to communications data, the IPCO reports about 800 000 requests for communication data per year in the UK [301] or about 1 request every 9 seconds assuming that requests happen during work hours. (There are no equivalent publicly available statistics for other settings.)

A HLF-based log capable of 40 requests per second, placed at the interface for law enforcement (standardized by ETSI TS 103 307 [294]) would be more than sufficient, with an average waiting time of 25 ms assuming Poisson-distributed requests. For a Trillian-based system with 102 transactions per second, the average waiting time would be 10 ms.

## Trade-offs

Trillian has a higher throughput as no consensus is required among different nodes to agree on the ordering of transactions, and better user auditability as when a user

queries the map server for an $id_c$, the map server returns a Merkle proof of the key and value being included in the map. The key history function of HLF does not provide a cryptographic proof, so replaying the entire blockchain can be necessary to verify the inclusion of a key and value. Users could however outsource this task to a data broker.

HLF supports flexible chaincode policies to determine write access to the log and comes with built-in authentication and PKI services. However, this means that users must submit queries to audit the log using a pseudonymous identity. If they used the same identity for multiple queries, their common identifiers could be linked together. Authentication must be done separately in Trillian.

The two systems also differ in their architecture. HLF is decentralized (although it is permissioned) whereas Trillian is centralized. A decentralized approach is appealing because it reduces the trust required in single entities to maintain the log. In practice, however, there is only one organization that legitimately has reason to write records for a particular business relationship. Users will mostly only have a single data provider for a service, which may lend itself more towards the centralized approach.

Table 5.6 summarizes the features of both implementations. Ultimately, Trillian is easier to deploy and has less setup than HLF, which requires the setup of a network of multiple nodes to act as peers, and the maintenance of an identity service to allow nodes to interact with the network. HLF and other blockchain-based approaches may be preferable if an organization is already using the technology for some other purpose.

### 5.7.2 Evaluating the verification of statistics with Multiballot

The simplest case when verifying statistics is the univariate case. In this case, the exact counts for each value of every element are preserved, so statistics can be recomputed with 100% accuracy. This means that for applications like the IPCO report on law enforcement access to communications data [301], every statistic could be verified using our scheme. We, therefore, focus on the more complicated case of univariate statistics for the rest of this section.

**Table 5.6:** Summary of supported (full circles) and partially supported (half-circles) features of the HLF and Trillian based logs.

| **Features** | HLF | Trillian |
|---|---|---|
| User privacy | ● | ● |
| Agent privacy | ◐ | ● |
| Data provider privacy | ◐ | ● |
| Statistical privacy | ● | ● |
| User auditability | ◐ | ● |
| External auditability | ● | ● |
| Verifiability | ● | ● |
| Access control | ● | ◐ |

Our evaluation measures the accuracy of the association rule metrics computed on $D_{priv}$. For our experiments, we generate multiple synthetic datasets that follow the structure of $D$ described in Section 5.5, with several frequent element sets [303]. We mine these for association rules using the Apriori algorithm [304], identifying frequent elements in the dataset and extending them to larger element sets for as long as the element sets appear frequently enough in the dataset. We then compute the support and confidence measures on $D_{priv}$ for the previously extracted element sets, and compare those values with the reported values for the same element sets on $D$. We use the percent error $\%Err$, defined in Equation 5.7.1, to measure the disparity between statistics computed on $D$ (the ground truth value, $GV$) and $D_{priv}$ (the measured value, $MV$).

$$\%Err = \frac{|MV - GV|}{|GV|} \cdot 100 \tag{5.7.1}$$

We opt to use synthetic datasets to evaluate MultiBallot, by simulating scenarios with a known ground truth rather than relying on sanitized public datasets for which the ground truth is unknown. We also verify our results using commonly used public datasets, such as the Extended Bakery dataset [305] and the T10I4D100K dataset [306]. In all our experiments (repeated 100 times) we measure the error for both the support and the confidence metrics. Because these are identical, however, we only include the graphs for support here.

Our first experiment studies the percent error for the support over two elements when varying the number of rule occurrences for a dataset of $1M$ records. Figure 5.7 shows the results in the case of 3, 5, 7 and 9Ballot. Element sets that occur less often are prone to higher percent error, with a high variance in the reported support values. This is expected for rules with very low support as, for example, observing a rule twice in $D_{priv}$ when it occurs only once in $D$ gives a percent error of 100% despite the practically meaningless difference. As element sets become more frequent (up to around 11%), the percent error ($< 2\%$) and the variance both shrink. As the difference in percent error between MultiBallot schemes also shrinks we focus on the results for 3Ballot in the following experiments.

Our second experiment studies whether the accuracy for an element set depends on the number of times the element set occurs, or its occurrences relative to the overall number of users (i.e., support). We generate four datasets of size $1k$, $10k$, $100k$, and $1M$, and pick five element sets with support 0.1, 0.3, 0.5, 0.7, and 0.9, from each dataset. The percent error (shown in Figure 5.8) shrinks as the support increases, but the absolute size of the element set plays a bigger role in the accuracy of the statistics. In the cases of the $100k$ and $1M$ user datasets, the support has only a minimal effect on the accuracy. Our results are consistent with those of Blum et al [307].

Our third experiment evaluates MultiBallot for different element set sizes using a synthetic dataset of $100k$ users. Figure 5.9 shows that accuracy is sensitive to increases in the number of elements. This is expected as the scheme probabilistically estimates the field values of the original record based on the observed shares, and the inference error for each field adds up with the number of elements.

Our results show that, based on the type of statistics published by the IPCO [301], MultiBallot can provide publicly verifiable statistics in the context of law-enforcement access to telecommunications data.

To evaluate the applicability to healthcare data, we consider two types of studies: studies on genes and protein networks, and epidemiology studies. In studies on genes and protein networks, datasets commonly contain between $100k$ and

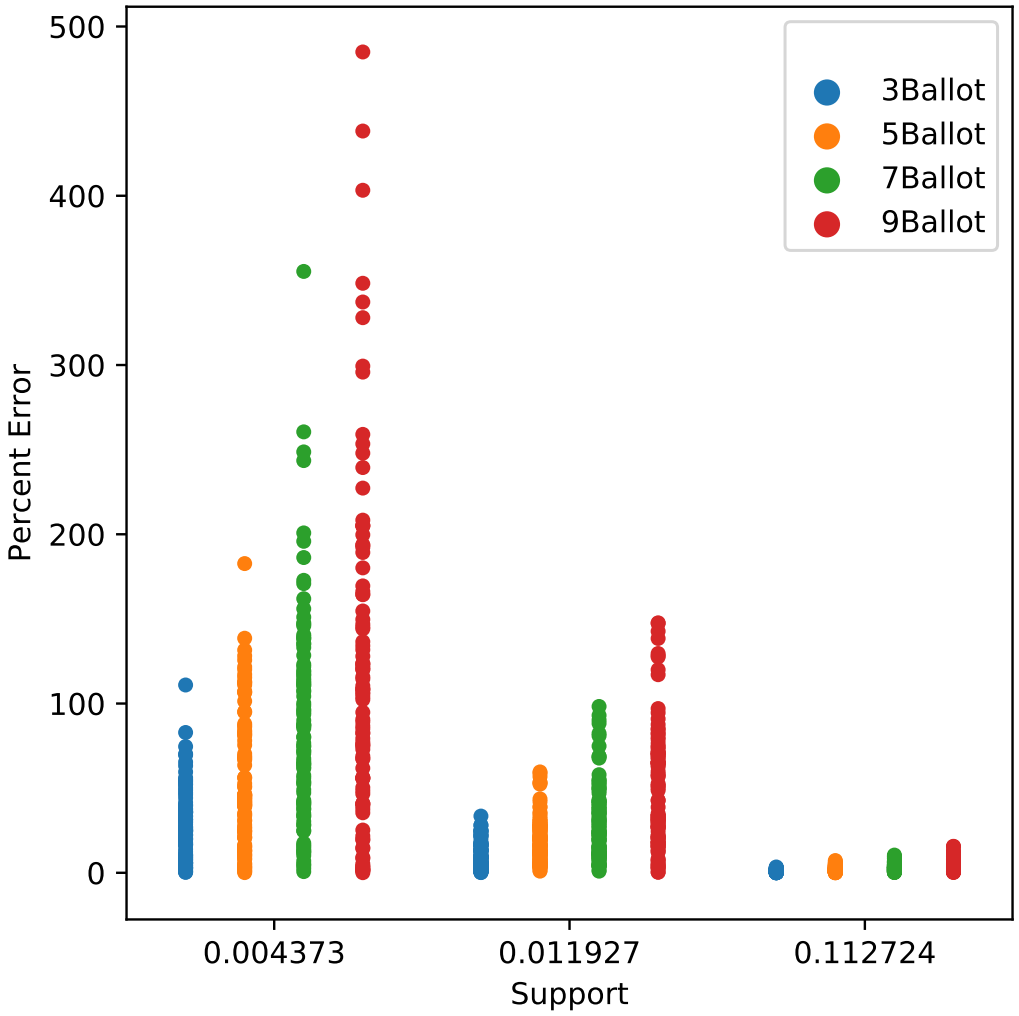


**Figure 5.7:** Percent error for the support over two elements as rule occurrences vary in the case 3, 5, 7 and 9Ballot.

a few million records, with a support threshold usually around 0.5%. In most cases, valid association rules are composed of only two elements and their support is greater than the minimum threshold. (The minimum threshold is relevant only during the rule-mining phase.) In the verification phase, the users compute measures over the relationships that are reported by the researcher as strongly associated [308, 309, 310, 311].

In epidemiology studies, the average element set size is 3, with a minimum support of around 1%. However, the support of relevant element sets identified is much higher and ranges from 1% to 16%, and datasets contain between 10,000 and

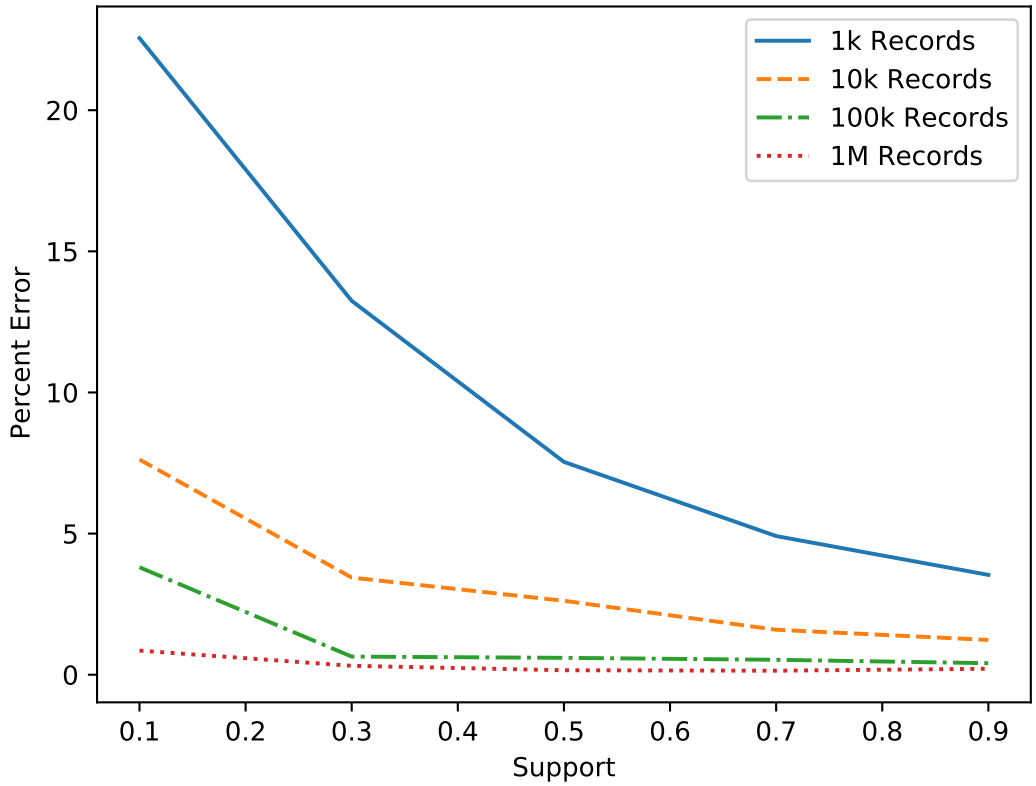


**Figure 5.8:** Percent error for elements that appear with varying frequency in datasets with different number of users, using 3Ballot.

250,000 records [312, 313, 314]. Our analysis of MultiBallot shows that acceptable accuracy can be obtained for such studies of healthcare data.

## 5.8 Deployability

For a system like VAMS to be deployed, agents and data providers would need to implement the necessary infrastructure. They may do so as part of transparency initiatives to increase public confidence [315]. As we have shown in the benchmarks presented in Section 5.7, this may be cost-effective as VAMS can achieve good enough performance on very cheap hardware.

Parties may also implement VAMS to allow them to demonstrate that data they submit as evidence in legal proceedings has not been tampered with. Alternatively, they may have a statutory obligation to provide transparency. For example, compliance with ETSI requirements may be a condition of providing a telecommunication service. Such standards do include provisions for requiring that access to personal

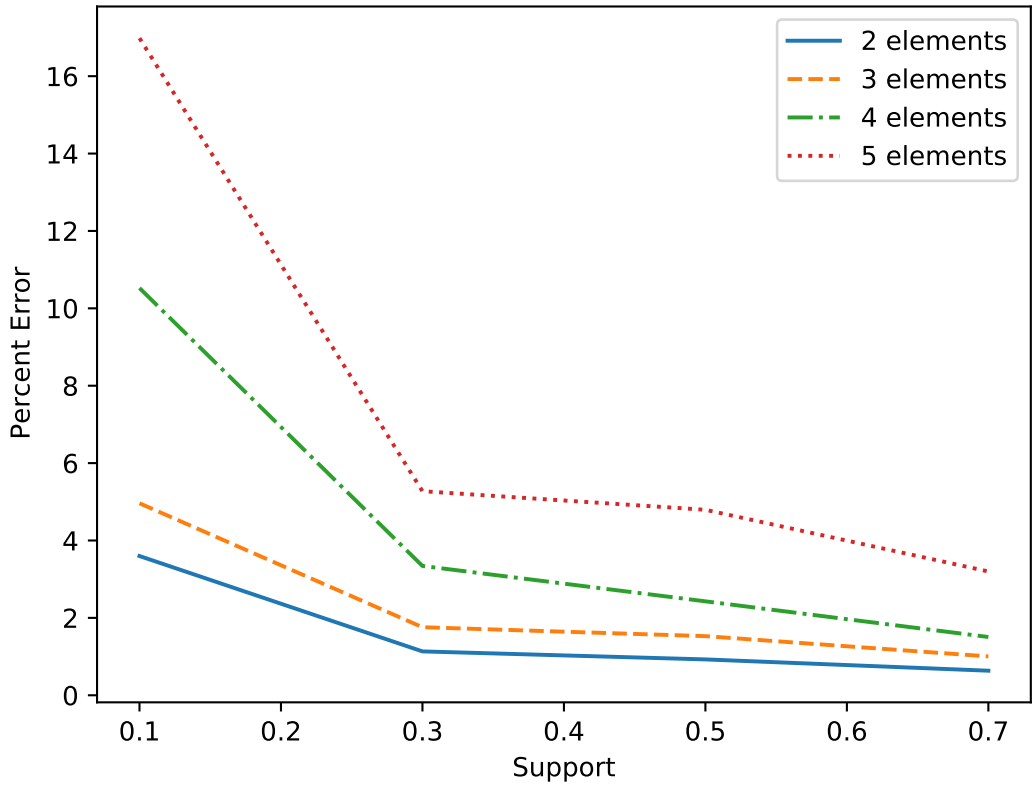


**Figure 5.9:** Percent error for element sets of varying size that have the same support, using 3Ballot.

data is auditable and that the authenticity of data can be established [294].

In the UK, the IPCO can require that public authorities and telecommunication operators provide the commissioner's office with any assistance required to carry out audits, and this could include implementing IT infrastructure [216, Section 235(2)]. Another possible route for imposing a statutory requirement to provide transparency could be through enforcement action of a regulator such as the Federal Trade Commission or a data protection authority. NGOs that currently work with transparency as an objective (e.g., make Freedom of Information requests) could also have an interest in maintaining and operating a system like VAMS by, for example, hosting log servers and serving as data brokers or auditors.

## 5.9 Conclusion

We have proposed a system, VAMS, which achieves our transparency and privacy goals. Our work shows how existing transparency overlays used to provide tamper-

evident logging can be combined with our log entry tagging scheme and MultiBallot to support publicly verifiable individual and population level transparency about access to data requests. Our evaluation of two implementations of VAMS shows that the system also meets realistic performance requirements in practice, and not only on paper.

Our results illustrate that the current framework for requesting data can be greatly improved to benefit all parties involved. We have given two example use cases in Section 5.1 to illustrate how VAMS could be used. Its design does not depend on any particularities of these use cases so it could therefore be applied more generally. VAMS does not have to replace any existing component in the workflow of an organization. Instead, it serves as an overlay that can be used to achieve both transparency and privacy goals.

# Chapter 6

# Transparency, Compliance, and Contestability When Code Is Law

## 6.1 Introduction

Computer systems now have a broad, and increasing, role in people's lives, even when they do not interact with or have any privilege over these systems. The code that makes up these systems defines what these systems do and, therefore, the norms that they apply when functioning. The impacts of applying these norms can be negative and unfair, because they result from systems that are flawed (e.g., unreliable or discriminatory), or that, by optimising certain performance metrics at the cost of fairness, welfare, and other values, produce harmful externalities.

The harms of such systems are reserved for those that are subject to the system, and may not affect the entities that design and operate these systems – they may even benefit in some cases. These entities concern themselves with enterprise risks (liabilities) rather than societal risks (externalities), leaving the public which has little recourse to mitigate these harms to deal with them. Moreover, system faults that cause harm can happen silently in the sense that it is not always clear to someone with no control over the system that a fault has occurred (even if the victim might suspect that it is the case). This makes dealing with this source of harm difficult.

Security, which deals with ensuring that systems function as intended, should

prevent many of these harms, but because of the internal focus on enterprise risks, it can fail to prevent issues for the public. The law and the legal system are the recourse for individuals who deem they have been harmed, allowing victims of harm to be compensated, and misbehaviour that results in harm to be punished and disincentivized.

When dealing with harm that result from the application of code, both of these fields should come into play. Law should ensure that victims of code-enabled harm should be able to contest the systems that cause these harms. Security should ensure that people should not fall victim to flawed systems, and provide evidence that a system is (un)reliable.

In practice, however, this currently does not work. There are widespread issues with people suffering from flawed systems that have been applied to determine, among other applications, entry to buildings via facial recognition, jail sentences, and so on. Systems evolve quickly and the law (and security although it moves quicker than the law) has not kept up with the application of technology to these aspects of our lives, allowing harm to occur without sanction, and making it hard for victims of harm to contest the application of code-enforced norms that have caused harm.

To deal with this issue, the idea of algorithmic accountability, which studies how to design ways of making algorithms accountable, has gained popularity. In line with this field of work, this chapter works towards addressing the issue of reconciling the use of security mechanisms that can assert the behaviour of a system with legal processes that can be used to contest the norms enforced by a system.

### 6.1.1 Outline of the Chapter

We begin with the idea of norms, misbehaviour, how legal processes and security mechanisms work as two ways of dealing with misbehaviour, and the interaction between these two approaches in Section 6.2.

Looking at this through the lens of code as law and digisprudence in Section 6.3, we argue both the need for secure accountability mechanisms and how they must be designed to make them useful as tools to contest code-enforced norms,

which we expand on in Section 6.4.

This allows us to compare different approaches to auditing in Sections 6.5, where we make the case for transparency enhancing technologies against less transparent forms of audits based on assurances of compliance with norms. We illustrate this through two examples based on recent court cases that involved Post Office in the United Kingdom and Uber in the Netherlands.

We also discuss some practical considerations that relate to electronic evidence in Section 6.6, balancing transparency and privacy, and where transparency should be implemented with respect to the system it acts on.

## 6.2 Preventing Misbehaviour Through Legal Processes and Security Mechanisms

### 6.2.1 Norms and misbehaviour

Misbehaviour is the action of deviating from a norm. Following Hildebrandt's discussion of legal and technological normativity [316], we think of norms as regulative (mandating, permitting, or disallowing some pre-existing possible action) or constitutive (defining a possible action). The difference between both can be thought of in terms of how misbehaviour can occur in each case.

In the case of regulative norms, misbehaviour can occur by deviating from the regulative norm at hand by performing a disallowed action, which does not prevent the action from being performed but does entail possible punishment. For example, a regulative norm may stipulate that car should not be driven over a specified speed limit. This does not prevent driving a car at a higher speed (this is the driver's choice to make) but can lead to penalties enacted by the relevant authority if the speed limit is exceeded.

In the case of a constitutive norm, deviating entails not performing an action defined by the norm and, therefore, the expected result of adhering to the constitutive norm at hand does not occur, resulting in a form of failure for the user and the state of the system remaining unchanged. For example, a constitutive norm may be the rule that a password must be entered to login into an account. If no password,

or the wrong password, is entered, then it is simply not possible to login into the account - the user has no choice but to enter the correct password or they will fail to login and the state of the system will not change as the user's status will not change if they stay logged out.

There is of course the question of who defines what the norms are, and thus misbehaviour. For computer systems, the system's code often defines constitutive norms as it creates actions related to the system that did not exist before the system. (Different systems may of course share similar mechanisms and even re-use code; for example, a login mechanism, but logging into one system is not the same action as logging into another system.) Thus, whoever designs and implements (and in the case of a data-driven system, trains) the system has significant power to determine the norms that are put in place by the system.

The code itself is also the result of the social norms and practices of those who write it. This results in an expected behaviour model (explicitly specified or not) that the user should follow, with anything that deviates in a relevant way from this expected behaviour being thought of as misbehaviour. For example, in the US, the flawed design and training of an algorithm that produces risk assessments used to help determine whether an imprisoned person should be released resulted in black defendants being incorrectly labelled as higher risk compared to white defendants who were incorrectly labelled as lower risk [130]. This determined that being black constituted a deviation from the expected defendant model and was punished with harsher sentences – a reflection of a system that already disproportionately imprisons black Americans [317].

As we have seen, norms that are put in place by a system, as well as those that form the context in which the system was created and is operated, influence how that system functions and the experience of its users. They also influence the way the system is designed to respond in cases where it determines that users have deviated or misbehaved in some way. Our interest is in the design of mechanisms for the mitigation of harm that could result from the use of a system. This too will need to be informed by an understanding of how norms are put in place in technology and

how they can be challenged (and changed).

### 6.2.2 Law based disincentivization and punishment of misbehaviour

Law broadly defines the limits of acceptable behaviour and the consequences of unacceptable behaviour in everyday life. Its purpose is twofold. First, it disincentivizes people from acting in a manner that is defined as unacceptable by the law. Second, if people nonetheless act in such a manner, the law makes it possible to punish behaviour defined as unacceptable in law via legal processes that are themselves defined in law. The punishment can be financial (e.g., to compensate a victim following a civil litigation) or time and rights-based (e.g., a prison sentence following a criminal prosecution). These processes rely on the existence and availability of admissible evidence that shows beyond a certainty threshold that the person to be punished did, in fact, act unacceptably.

Processes are more fundamental to the law than specific laws are themselves. (Of course, processes are usually defined in the law itself, but we distinguish here between laws that are applied to determine the resolution of the question that results in a legal process from the laws determining how the legal process proceeds.) Although both vary across jurisdictions and are mutable, new laws that determine acceptable behaviour are introduced and changed much more frequently than, for example, the processes that are used to adjudicate trials. Moreover, the regular changes in laws show that they can be contested, as do the interpretation of the laws themselves, which is determined by courts and may be the subject of legal processes themselves.

The state institutions that legislate, enforce, and adjudicate laws are typically well-defined, although they vary across states. It is also possible for other organizations, such as private businesses, to act as rule makers and enforcers over the jurisdiction of a system they operate, for example through terms of service agreements, although these may, in turn, be subject to state-enacted regulations and the states legal system that would handle any dispute.

### 6.2.3 Security against threats and a posteriori security

Security, or more precisely information security in our context, works by defining mechanisms based on a defined threat model. There is no notion of absolute security, only security against a given threat model that relies on specified assumptions about the capabilities of an adversary and the difficulty (in a computational sense) or cost (in an economic sense) of performing certain tasks.

Traditionally, security mechanisms are constitutive and impose behaviour that honestly follows a protocol, implying that an adversary cannot possibly misbehave and break the security guarantees provided by the security mechanism (otherwise the mechanism would not be secure by definition). For example, a provably secure encryption scheme that is well implemented cannot be broken by an adversary with more than a negligible probability (in the formal mathematical sense of the term). Thus, security mechanisms are very different from how regulative legal mechanisms function. Misbehaviour cannot happen in principle and, therefore, there is no kind of accountability process or defined punishment for the adversary.

Not all misbehaviour by individuals or algorithmic systems can be stopped a priori, however, which has motivated work on security mechanisms designed instead to detect misbehaviour and produce evidence of that misbehaviour [124]. Moreover, the reliability of preventative security mechanisms must also be empirically examined on occasion. In the context of this chapter, this type of a posteriori security mechanism is what we focus on.

An example of this being successfully deployed in practice is Certificate Transparency [172]. This is a now widely adopted [248] system that provides tamper-evident transparency logs that record the issuance of SSL certificates for websites by certificate authorities. Certificates are either logged, making them easy to inspect, or not logged in which case a browser client that encounters such a certificate can report it. This allows misbehaving certificate authorities who (on purpose or due to compromise [242]) emit problematic SSL certificates to be detected, disincentivising such misbehaviour or, conversely, incentivising security on the part of certificate authorities.

The consequences of misbehaviour that is easily detected can be severe if sanctions are imposed. A certificate authority that is deemed to have misbehaved by Google, Mozilla, Microsoft, and other browser vendors may be blacklisted by their browsers, mitigating the harm done to users, and practically ensuring that the certificate authority quickly goes out of business. For example, DigiNotar went bankrupt shortly after being compromised and having its certificates deemed untrustworthy [318].

Security mechanisms like Certificate Transparency are defined as technical mechanisms that record evidence of misbehaviour and, therefore, function more like regulative mechanisms, there is no built-in notion of accountability process or punishment. That is left to whoever relies on these mechanisms, such as Google (who dominates the browser space [209]) and other browser vendors.

Thus, unlike law, it is the technical mechanism that is fundamental here, rather than the process of dealing with misbehaviour once it is detected. More often than not, there is no well-defined accountability process and it is instead determined by power relations around the system.

This implies a distinction between the security of the system (e.g., making sure that certificates are trustworthy) and the security of the parties in the system. DigiNotar vanished following its security incident, and browser vendors protected their products and users from future harm, but those affected by illegitimate certificates before measures were taken were not protected or compensated in any way due to this mechanism.

### 6.2.4 Economic considerations

Economics considerations play a role in both cases.

Harms caused by algorithmic systems often do not fall under criminal law and, therefore, the consequences are primarily financial, which leads to economic considerations of expected costs. As Wu puts it “laws impose costs upon regulated groups. Those groups that seek to minimize the costs of law face a fundamental choice between mechanisms of change and avoidance. Both mechanisms have the effect of lowering the expected costs of law, but the similarities end there. Mech-

anisms of change (principally lobbying) decrease the sanction attached to certain conduct and tend to require collective action. Mechanisms of avoidance, on the other hand, decrease the probability of detection and typically do not require that groups act collectively, but depend on specific vulnerabilities in the law." [319]. For example, Google has multiple times paid fines to the European Union that are greater than many of the contributions to the European Union by member states [244].

Similarly, the security of systems often relates to the underlying economics of securing the system [116]. Securing a system has a cost that, economically speaking, is only worth expending if it outweighs the expected loss due to the successful exploitation of the system by an attacker or, more generally, a system fault. When the costs of system fault can be passed on to the users who are harmed, there is a perverse incentive not to expend resources on making the system reliable.

The economic considerations related to both legal mechanisms and security mechanisms directly relate to each other when we consider cases where a system can harm users. If that is the case, the legal risk for the system operator is that they may be legally responsible for the harm done to users by faults in the system. The European GDPR, for example, makes these regulatory risks real for certain kinds of data protection failures. In these cases, the economic considerations of expected costs due to the risk of regulatory non-compliance are the economic considerations that can favour (or not) the implementation of reliable security mechanisms.

### 6.2.5 The interaction between security mechanisms and legal mechanisms

The overlap between legal processes and security mechanisms happens when a security mechanism is intended to ensure compliance with a legal norm. A common historical example of this taking place is the repeated attempts to apply copyright and digital rights management (DRM) to online content, which motivated both technical work (see, for example, Chapter 24 of Anderson's book [320]) and legal work on the interaction between code and law [321].

This put the focus on two things. First, technology could change the efficacy of a law and facilitate unwanted behaviour. For example, distributed online file sharing made it much easier to ignore intellectual property law. Second, technology could be used to deal with the change in the efficacy of a law by deploying mechanisms that prevent the unwanted behaviour enabled by technology (e.g., DRM mechanisms).

While security mechanisms and legal mechanisms are both ways of enforcing norms and interact in many situations, they are not interchangeable. Security mechanisms, in particular a posteriori security mechanisms, are technical mechanisms that enable the collection of evidence. Legal mechanisms are processes of determining the consequences that should be applied to parties in response to their behaviour based on evidence related to that behaviour.

Thus, the legal analogy for a posteriori security mechanisms is that of evidence collection while the accountability process is in the hands of those who can (i) access that evidence (which may be determined by technical access control mechanisms) and act upon it (which requires agency and authority). Re-iterating the previous example of Certificate Transparency, while everyone can monitor Certificate Transparency logs, it is effectively only browser vendors who can act upon the information they contain and enact some kind of accountability on the misbehaving certificate authority. Although this may manifest itself through code by, for example, blacklisting certificates signed by the misbehaving certificate authority, the process of accountability is a decision process within the organizations themselves, not one determined independently by code.

The interaction between security mechanisms and legal mechanisms for accountability is, therefore, centred on how security mechanisms can be leveraged to serve legal mechanisms.

This interaction is not necessarily frictionless, however, as there can be a "clash between rules and principles exacerbates the difference in perspective between system designers, who favour formal rules, and policymakers, who are more comfortable with the situational application of principles" [322]. Unlike Google with

Certificate Transparency, the legal system and many more organizations do not have the capacity to both design and make use of technical mechanisms that can support accountability processes. Without such capacity, however, dealing with systems that can produce harm is difficult.

## 6.3 Accountability Through The Lens of Code Is Law and Digisprudence

### 6.3.1 Code is Law and Digisprudence

The notion of code as law in academic work goes back to Reidenberg [323] who noted that "technological capabilities and system design choices impose rules on participants" and Lessig [324] who framed the issue as "we therefore don't see the threat to liberty that this regulation presents".

The use of code as a part of legal actions existed as transactions tied to contracts were already being executed through code at that time. Moreover, Szabo introduced the idea of smart contracts [325] that made explicit the possibility of contractual transactions that would execute entirely through smart contracts implemented in code. Smart contracts are now the basis for cryptocurrencies such as Ethereum, which are essentially decentralized smart contract platforms [35], and law scholars have studied their role as legitimate legal contracts [326].

More generally, however, code[1] that defines the operation of technical systems forms, like law, a way to regulate the behaviour of people subject to the system. Subjects of the system in this case include not only people operating the software or are users of the system, but also those on whom the system can have an effect. For example, someone who is run over by an autonomous vehicle operating software that did not determine the vehicle should stop once the person was identified will be affected by the software operating the car without ever interacting with it or consenting to be subject to it. This effect can also be mediated by a third party, including in legal matters, as is the case when judges make decisions based on the

[1]Here, code should refer to not only the actual code written by a software developer but also its compile environment because a compiler can interpret code in a way that undoes desired properties such as constant time execution [327].

outputs of (generally biased) automated decision-making systems [130].

As Diver [328] suggests, code is not law per se, even if its automation means that it governs the behaviour of people in the system more effectively, because it lacks law's mechanisms of ex-ante legitimation and ex-post remediation. Diver makes four claims about code, its effect, and its design [328].

First, code can have regulative effects on behaviour that are more pervasive and direct than law is capable of. Moreover, the regulatory effects of code do not need to be compatible with law.

Second, norms that regulate citizens, including those that are imposed by code, ought to be legitimate in that they ensure certain formal qualities in their design.

Third, attention should be paid not only to the legitimacy of code but also to the legitimacy of the design of code.

Fourth, legitimation of a code-imposed rule must be done at design time because there is little scope to re-interpret code after the fact.

Dealing with this requires a theory of what constitutes legitimate code that Diver names digisprudence, which is based on the following affordances: transparency about the provenance, purpose, and operation of code; oversight; choice; intelligibility supported by delay; and contestability as the overarching concern [328].

### 6.3.2 Digisprudence and Accountability

Digisprudence as a framework is aligned with the desire for accountability mechanisms that can provide the affordances we have just listed, beginning with transparency about the provenance, purpose, and operation of code. Oversight is required to make use of transparency to apply accountability. Choice is related to the norms enforced by the system, or simply the choice to be subject or not to these norms, which requires transparency about these norms and how they are applied in the first place. Intelligibility and the affordance of delay are, in turn, required for oversight and choice to take place.

Contestability is less integral to the discourse about accountability. For example, Wieringa's recent systematic review of the field does not mention contestabil-

ity [96]. Rather, accountability is often focused on whether or not a system has functioned correctly instead of the legitimacy of the norms the system applies – "trust but verify" as the saying goes (see Desai and Kroll for example [102]). This suggests that a choice must be made between wanting accountability for the performance of the system (which does not require contestability) or accountability for the norms enforced by the system (which requires contestability). We return to this in the next section.

Because the accountability mechanisms we are concerned with here also involve code and, indeed, accountability mechanisms are designed to apply norms, we must also pay attention to how these affordances are taken into account when designing and executing accountability mechanisms.

Fundamentally, accountability mechanisms must reveal information about the system and enable action to be taken based on that information (which may include legal action or some other process). Thus, they regulate access to information and the effects of access to that information.

The provenance, purpose, and operation of an accountability mechanism should make clear what the mechanism is intended to provide accountability for, to whom, and how. The incentives of the party that designs the accountability mechanism are important. An accountability mechanism designed for a system by the system's operator that primarily works to prove the correct execution of the system that, for example, does so in zero-knowledge as suggested by Kroll et al. [101], may not be considered as legitimate by the public as another mechanism for the same system that reveals more information about not only the system it provides transparency for but also itself. For example, a zero-knowledge proof, even if publicly verifiable, that is verified by a judge does not allow for any explanation beyond "computer says yes" or "computer says no", which may not be a satisfying explanation for the behaviour of complex systems.

More generally, the assumptions that underpin the design play an important role because they can determine the legal effect of the use of the accountability mechanism (e.g., it supports the production of admissible evidence to be used in

court) but also the type of misbehaviour that it can provide accountability for based on the threat model (e.g. whether the system operator and code are considered adversarial to accountability) that determines its security design.

Assumptions about the code that is subject to accountability are also important. Interpreting code as law generally entails considering code as a form of strong legalism, but this assumes that the code is reliable and secure, otherwise its effects can be bypassed and it fails to demonstrate strong legalism. Accountability mechanisms must take this into account by not assuming that the code is necessarily reliable and secure, and by being designed to function independently of the code so that it does not fail if the code fails.

There should be oversight over the use of accountability mechanisms, to make sure that they are effective in providing accountability, and that the way they regulate access to information and the effects of access to that information is aligned with its design and purpose. Of course, intelligibility (or usability in the context of designing a secure accountability mechanism) is necessary for this to be possible.

Likewise, choice must be given to be subject to the norms accountability mechanisms entail. Either for the system operator whose system will be subject to an accountability mechanism, in the case where there are no regulations requiring its use. (If there are regulations, there is a notional choice to abide by them and flexibility in how to implement them.) This point has been made under the guise of protecting commercially sensitive aspects of the system [102, 101]. This also applies to users of the system whose information may be revealed as part of transparency.

Contestability also matters because accountability mechanisms should enable consequences. The fact that, for example, transparency by itself is not always effective is that it can fail to enable further actions [152, 139]. Thus, it should be possible to contest accountability mechanisms so that the consequences (or lack of consequences) can be considered legitimate. A practical example of this is for mechanisms that serve as evidence producing mechanisms that enable legal dispute, the admissibility of the evidence produced can be contested according to the norms set out of law that regulates evidence. We explore this in greater detail in the next

sections.

## 6.4 From Accountability to Contestability

In Section 6.2.5 we highlighted a takeaway from the interactions between security mechanisms and the law, which is that technology can (i) serve to bypass and (ii) enforce law. If we take code as acting somewhat like law, this is still true.

Hacking, Privacy Enhancing Technologies (PETs), and Protective Optimization Technologies (POTs) [329, 330] show the existence of this interaction in practice.

Hacking attempts to do something that is not allowed by the norms of the system. This is often viewed through the lens of criminal hacking, but it can also fall in grey legal areas [331] or be done to contest norms that are reasonably considered illegitimate. In general, this is a solution that does not scale well because it can require technical skills that are not widespread among users, and typically does not entail any modification of the hacked system that would benefit users other than the hacker. An example where this is useful, however, is when it prevents the system from functioning (if this is not outweighed by some benefits the system might bring) or leads to greater transparency about the system (like whistleblowing) that can be leveraged to contest the system.

PETs constrain the capability of code that is designed to collect private information. For example, end-to-end encryption, which is widely deployed in messaging services, prevents the ability for someone to execute code that would eavesdrop on a conversation, which would otherwise be possible by default. After more than twenty years since PETS became an active topic [332], privacy engineering is now its own discipline [333, 334] backed by data protection regulations (e.g., the GDPR), although systems still routinely compromise user privacy to satisfy a logic of information accumulation and surveillance [335].

POTS attempt to overrule the effects of code-driven optimization, allowing users outside of the system to intervene without requiring cooperation from the system's operator. For example, using Sybil devices to generate fake traffic in an

area can stop traffic routing apps (e.g., Waze) from routing traffic to the area and mitigate the negative externalities that would otherwise ensue in said area [329]. There is, however, no guarantee that such interventions cannot in turn be optimized away by the target system once it is adjusted to take the existence of a POT into account.

These tools are available to individuals and can be effective (even if only to a limited extent) against code designed and deployed by states, companies, and other large institutions, showing that contesting code-imposed norms is sometimes possible (although these tools are not necessarily accountability mechanisms). Code not only enforces norms but can also be used to contest and bypass norms, and the fact that these tools are user-centric distinguishes contestability from traditional accountability that is centred on the system operator.

When code-imposed norms are discussed and determined to be harmful in some sense, through the use of secure accountability mechanisms, they can be changed. Even in the case of code that is intended to provide immutability by design, such as blockchains, these guarantees are void if other interests are deemed more important. Following the loss of $36,000,000$ ether due to an insecure smart contract, Ethereum users simply decided to fork Ethereum to revert the situation [195], creating Ethereum Classic (which did not revert the hack) and Ethereum (which did). Ethereum, the forked chain that decided that "code is law" was not worth it at that moment, has since been the dominant chain .

How did this happen? The realization that the loss of funds was (i) of great value, both financially, and in terms of the ability for users to trust the system with their funds; (ii) reversible because it was possible to introduce code that would transfer the stolen funds back to their original owners, at the cost of forking the chain; (iii) reverting the hack was supported by many powerful members of the community; for example, Vitalik Buterin, Ethereum's most important public figure [336] and idea contributor [337]. Ethereum was, therefore, clearly accountable to its users who (at least those that had more influence over the community) in turn were able to contest the norm applied to their ability to recover funds. Moreover,

because of the transparency offered by Ethereum, any interested user could see exactly what had happened, what could be done, and what was done in the end.

This example shows that transparency enabled accountability can be used to contest the effects of code and change them. This result is not necessarily generalizable, however, because it played out in favour of those with disproportionate power over the system. In many cases where we would like to introduce accountability to the extent that norms can be contested, those with power over the system (e.g., system operators) are not those that wish for the norms to be contested. Rather, they are those who want to enforce these norms in the first place. This brings back a common theme with accountability, which is the importance of power relations around systems.

This should inform how we design accountability mechanisms because, as mentioned in the previous section, accountability mechanisms can regulate the effects of access to the information that the mechanism makes available to some. This is because the format of that information plays a role in how it can be used. If any aspect of the system is to be contested, therefore, it must be determined how this will happen.

Some systems, such as Ethereum in the example above, afford more power to their user communities but this requires a level of decentralized governance that is rare. For the vast majority of systems, which are deployed by centralized private entities, there are no governance mechanisms that could allow an individual subject to the system to systematically influence it. Thus, in this chapter, we focus on contesting norms enforced by systems through legal processes with the intent of contesting the formulation of these norms. Although it is not ideal and can fail in loud (e.g., if there is media attention) and quiet ways (e.g., for groups society does not care about or actively discriminates against), the legal system is often the best chance of contesting a system an individual will have. As a result, the format of the information that accountability mechanisms provide should be usable as part of public disclosures of information about the system and admissible evidence to be used in court to support an argument in a dispute about the system.

## 6.5 Compliance and Transparency Based Auditing

### 6.5.1 Verification and compliance based auditing

In theory, systems could be formally verified and, therefore, treated as reliable assuming that no design flaws were presented (a strong assumption in itself). In practice, however, formal verification tools are of limited use because many systems involve multiple different protocols that interact with each other across different hardware, software, and network conditions, making formal verification of an entire system unrealistic.

Software is often continuously modified (as well as the operating system it runs on), in particular for new applications, which can involve millions of lines of code representing extremely complex protocols, with a non-zero rate of bugs in the code and logic flaws at the design level. Data is shared across networks operated by different parties, in varying network conditions (affecting reliability or synchrony assumptions required by distributed protocol design models), which makes strict enforcement mechanisms impractical [124]. Even hardware, at a scale at which some large-scale applications operate, may fail to be reliable for basic tasks such as encryption and decryption [338].

A weaker form of verification that is more realistic is based on compliance based auditing that checks the correct execution of a process in a system rather than the correctness of the system itself. For example, automated tools may work by checking for compliance with certain norms (e.g., certain specific clauses of the GDPR [339]). This is limited to cases where the desired norm is assumed, or simply required by law, which may not always be the case. In practice, many systems enforce norms that fall under a grey legal area, or like many clauses of the GDPR, are not related to compliance, system behaviour, or require interpretation, and cannot be encoded in logic and automatically checked for compliance.

The automated aspects of these tools do not provide any agency to any individual that would be harmed by the system, because there is no need for them to provide access to any information to unprivileged users of the system. For example, a system operator may be able to show that the system has complied with the de-

sired norms when it has, but when it hasn't a user may not be able to generate any evidence of this. This solution, therefore, benefits honest system operators but does not necessarily punish those that operate flawed systems.

Because the focus is on compliance with a pre-established norm, it does not leave much space to discuss the norm itself. A logical compliance test that returns a boolean pass/fail value will not be able to provide much information about edge cases or the cause of passes or fails that may be necessary to evaluate the norms, and the reason that the system satisfies a norm may be that the norm itself is specified erroneously. Having a human in the loop also brings its own challenges [340], and may risk the humans in the loop legitimising a system because it passes compliance checks that do not represent all the harms they may cause.

It is also important that the software be well designed to represent the norm it wishes to verify and secure enough to operate in an adversarial environment. For example, Volkswagen developed software that could detect when their cars were being tested so that they could change their performance accordingly [341].

Hardware that supports trusted execution environments and cryptographic tools that can be used to verify computations [342] can be applied to verify the execution of the assurance software can be applied in cases where the threat model requires it and to permit public verifiability. For example, methods of providing the public with cryptographic proofs that certain processes have followed have been proposed, based on zero-knowledge proofs and secure multiparty computation [18, 17]. Although the outputs of these systems can be verified, their inputs cannot. Thus, this amounts to assuming honesty on the part of those that control the inputs and, therefore, that the processes that are meant to be audited have been followed correctly. This is not an appropriate threat model for many cases where it can be assumed that processes may not be followed honestly and systems may be faulty. Moreover, because zero-knowledge proofs obfuscate practically all information, their use is very limited to investigate misbehaviour that would involve nuanced details [250].

Finally, as Kim points out [343] transparency and audits are still necessary

even if assurances exist, because the fault in the system that causes harm may not be in the code but in the design itself.

### 6.5.2 Transparency Enhancing Technologies

Transparency Enhancing Technologies, in contrast to compliance based solutions, focus on making information about the system available rather than evaluating the system. The evaluation is regarded as another process (which may or may not be automated) that is therefore more transparent because the information it is based on is more widely available.

In terms of technical mechanisms, this approach is therefore based on producing logs of operations in the system (transparency overlays) for which there are well-defined cryptographic security models [29] as well as implementations of reliable logs (e.g., the Certificate Transparency logs). Kroll provides a survey of traceability mechanisms [161]. Given a log of a program's actions in the system, it may also be possible to determine the program actions that were actual causes of the program deviating from its specified behaviour [344]. Likewise, for machine learning based systems, it can be possible to quantify the degree of influence of inputs on outputs of the system and release the information for transparency [345].

Transparency is based on recording and making information available, therefore, it does not assume a norm for the system like compliance based solutions. Thus, it makes it possible to explore what that norm is via the information it makes available. Moreover, it does so independently of the system's norm that may have been specified at its design stage. This is akin to adopting a stronger threat model that makes fewer assumptions about the system it audits and those that interact with the system. It can, therefore, identify discrepancies between norms that were desired at the design stage and those that are actually enforced as the system operates.

Transparency can also be more public facing and democratic than compliance based solutions. First, it is based on releasing information rather than checking it. Second, a transparency system (e.g., logs) can be maintained by various parties and relied on by others. Assurance software, however, must be possessed by those who execute it and are typically not publicly available. A broader audience invites a

broader critique.

An example comparison between a compliance based system and a transparency focused system can be made in this case. We have already mentioned the work of Frankle et al. [17], which uses cryptographic tools (zero-knowledge proofs and multiparty computation) to verify that secret legal processes to authorize surveillance, for example, have been followed. The output of this solution is a cryptographic proof that processes have been well followed, and statistics about these processes, but it does not reveal anything else.

VAMS, described in Chapter 5, addresses a similar problem, that of auditing requests for access to data made by law enforcement. This chapter proposes a solution that logs (similarly to Certificate Transparency) and releases the log of requests for access to data with read access reserved to auditors (for all requests) and individuals (to see requests for their data). This allows publicly verifiable statistics to the extent that individuals can verify the inclusion of requests for their data in the computation of the statistics, and recompute the statistics themselves based on a privacy preserving synthetic dataset.

The first system, proposed by Frankle et al. offers stronger confidentiality guarantees but is only useful if processes are followed correctly. If they are not, not much can be learned by design. The second system offers confidentiality guarantees that are weaker than those offered by zero-knowledge proofs because more information is revealed by the release of a synthetic dataset of logged requests. However, it can be used to identify errors (i.e., deviations from the specified "honest" norm) and abuse (i.e., the existence of a malicious norm) more effectively, and provide greater agency for those affected. Thus, if things go wrong, this solution may be more useful in contesting the system it looks at.

There are, therefore, trade-offs to consider, but if the ability to contest norms is required then the argument is in favour of transparency that can accurately produce evidence of the system producing behaviour that does not respect the desired norm, or correctly enforcing a harmful norm.

### 6.5.3 Examples of the usefulness of system transparency in court cases

**Post Office Limited and its unreliable accounting system** Post Office Ltd is a state-owned private company in the United Kingdom (UK) that provides a variety of services to customers including postal and financial services. Subpostmasters operate Post Office branches on behalf of Post Office Limited and are responsible for any losses at their branches.

The accounting at each branch, however, was handled by a centralized accounting system named Horizon, which was developed in the nineties. As it happens, Horizon, like most large IT systems, suffered from bugs that could lead to accounting errors. Over the years, Subpostmasters were accordingly requested to cover the losses or be criminally prosecuted.[2]

One important factor in these prosecutions was the legal presumption in the UK that, unless there is evidence of the contrary, the evidence produced by a computer was reliable. Post Office had access to a Known Error Log but did not disclose its contents [347], and because evidence was treated on a case-by-case basis, it was never possible to establish the unreliability of Horizon for a single defendant with limited resources. Thus, "a subpostmaster could be held responsible for losses they incurred as a direct result of failing to notice an error in a sophisticated computer system over which they had no control" [347].

More recently, however, a Group Litigation that allows a collection of cases to be examined in parallel took place. This allowed subpostmasters to contest Horizon as a group with pooled funds and more combined evidence to contest Horizon more effectively. As a result, it was possible to force more disclosures about Horizon that made it possible to establish that it was unreliable, with forced the government to put aside hundreds of millions of pounds to cover the payouts in what is considered the biggest single miscarriages of justice in British history [348].

It is instructive to consider this example, and how similar situations could be improved because it is a large system but one that is nonetheless less complex than,

[2] See Nick Wallis' book [346] on the subject for more details.

for example, machine learning based systems. It is also a typical kind of system that people will interact with daily. Many other faulty traditional systems have caused legal issues [129]. Reasoning about the responsibility of individual bug occurrences in a system is difficult because if the probability of a bug occurring is similar to the probability of a user committing fraud then we are left with biases [7].

As the Group Litigation showed, an approach based on transparency of the know error log and intelligible recordings of the system's operations could improve things by making accessible the information that was actually useful in practice [7]. This would make it possible for subpostmasters to (i) be aware of potential bugs (transparency about the system), (ii) analyse the logs of their system's operation (transparency about their interaction with the system), and (iii) have access to evidence that can be used to contest any faults in the system that may occur. Moreover, the security of such a system should be based on a threat model that assumes Post Office to be adversarial to transparency as they actively hid the contents of the known error log.

Relying on (zero-knowledge) proofs of correct execution would not solve the problem entirely because they only apply if the program executed entirely correctly, but this may not be the case if either the bugs that occur and cause the proof to fail are not responsible for faults (e.g., misrecording transactions) occurring, or if the program executes correctly but its logic is flawed. Moreover proving the correct (perhaps distributed) execution of a large program may simply be impractical. Focusing on only a small critical component is not enough because if, for example, the accounting executes correctly but the display is faulty, a subpostmaster might try to fix the error manually, leading to discrepancies.

**Uber's fraudulent activity algorithm** In another case, the Amsterdam District Court ruled that drivers from the UK were permitted to contest the norms applied to them by Uber's system (as well as other similar companies e.g. Ola) when they were banned from the service for fraudulent activity. Moreover, the court ordered to provide transparency about numerous aspects of its system, including the data used by Uber's algorithm to dismiss the drivers, which was not previously accessible to

the drivers [349].

The issue for drivers lies in the fact that they are subject to the ratings they receive from customers and Uber's system based on these customer ratings and other factors, which determine the service they receive from Uber and whether or not they are allowed to drive for Uber. Customer ratings may be biased, however, based on attributes such as the race of the driver, which then feeds into Uber's system determining that the driver should be banned if they fall under a certain rating. Other surveillance systems used to assess drivers are also in place such as facial recognition checks that may fail and lead to a driver being kicked off the platform [350].

Compliance based audits would not achieve much in these scenarios. When it comes to biased customers, there is no way to assess in advance whether customers will be more or less biased, or to produce a facial recognition system that functions such that there is a negligible probability of failure across all drivers. Inevitably, transparency will be required and must be available for drivers to allow them to contest such systems, without first having to go through lengthy, expensive processes to access the relevant information that is intentionally obfuscated.

While regulations, Article 22 of the GDPR that gives an individual the right not to be subject to a decision based solely on automated processing in this case, can enable an order to disclose aspects of the system, mechanisms to execute this are lacking, and it is not always possible for an individual to know that they are subject to such a system. There are suggestions for ways to audit the design [351] of AI systems as well as releasing information about the models themselves [121] and the datasets that they are trained on [118]. However, these are not designed with a threat model and, therefore, assume a fairly honest system designer and operator, whereas companies such as Uber have an incentive to obfuscate how their system functions to avoid scrutiny, and argue this is necessary for commercial confidentiality and customer privacy purposes.

## 6.6 Practical Considerations

### 6.6.1 Electronic evidence

The book by Mason and Seng [203] discusses many issues with electronic evidence in the legal context and makes clear that the topic touches upon many aspects of security, not only the authentication (typically handled through electronic signatures) and integrity of the evidence itself (typically handled through cryptographic hash functions), but also of the networks over which it is exchanged, and how it is stored. It also makes clear that when treating software as a witness, it must be taken into account that software can be written to deceive, as in the Volkswagen emissions case [341].

More recently, the Post Office case used as an example above has generated work discussing the presumption of reliability that evidence generated by computers often enjoy, and the issues this can cause [127, 352, 353, 354]. (Different jurisdictions adopt different standards of course.) This presumption that electronic evidence is reliable has also facilitated harm in cases where the party producing the evidence not only knows that it is unreliable, but also that it is essentially fabricated [355],

Related to this is also the necessity for expert witnesses to explain the evidence that is generated, so the explainability of the evidence plays an important role because the expert witness must be able to understand the evidence themselves and be able to explain it in a clear way to a judge or jury. Explainability has been investigated for machine learning based systems, sometimes with emphasis on explaining single decisions to individual users rather than explaining a system as a whole, which may be required to establish its reliability. The kind of explainability that is geared towards engineers [356] of the system may be more useful in this context, but may also be less accessible by design.

### 6.6.2 Balancing transparency and privacy

Whenever information that may be sensitive is made available, privacy and confidentiality concerns emerge.

This includes concerns for the privacy of the individuals who may be related to

the information that is released. This should be treated with care, using appropriate sanitization mechanisms; for example, by implementing data minimization and using differentially private data release mechanisms [54, 11], which are aligned with regulatory data protection requirements [67, 68]. Because different data carries different privacy risks, and different levels of usefulness in contesting the system, this is a problem that must be addressed on a case-by-case basis that takes into account the trade-offs between privacy, the consent of parties that relate to the information (or other bases for releasing that information), and the information that is necessary for transparency to be useful.

Often, a dispute may rely on both system-level information (e.g., error rates) and individual information (e.g., specific system events). System-level information such as univariate statistics may leak less sensitive information about individuals, while individual information is naturally more sensitive but may need only be accessible to the individual in question.

Commercial confidentiality can also be a concern. This motivated the reliance on tools such as zero-knowledge proofs suggested by Desai and Kroll [102] and Kroll et al. [101]. Some arguments support the idea of access to the source code of a system in the case of a dispute about the system [129], and as we have discussed above, relying on assurances rather than transparency may not enable contestability. Naturally, in cases where commercial entities benefit from information asymmetry, they are unlikely to want to provide greater transparency without an incentive or obligation that would provide trade-offs in favour of transparency. Thus, it may fall to evolving regulations and technical standards that govern the design and operation of systems to determine the right balance. As we have seen in the Uber example above, regulations can already force the disclosure of broad information about the system, even if they did not require that information to be public beforehand.

### 6.6.3 A system in one place, transparency in another

Systems are often designed and implemented in one place before being deployed internationally. Disputes around the system, however, often take place where the harm caused by the system has occurred, which may not be where the system has

been designed. Thus, transparency around the system, if it is to be useful in a dispute, should reflect the local context of the dispute, rather than the context in which the system was designed. The importance of the audience of transparency has been discussed by Kemper and Kolkman [357] and Felzmann et al. [358], highlighting the need for transparency solutions that reflect the population it interacts with.

## 6.7 Conclusion

In this chapter, we have argued for the necessity of employing secure accountability mechanisms to ensure the legitimacy of computational systems whose code enforces norms. In particular, we have argued the need for accountability mechanisms, based on transparency rather than compliance verification, to enable the ability to contest the norms that code enforces when these may be illegitimate.

This entails a culture shift to a user-centric notion aimed at giving users the agency to contest the systems they are subject to through channels such as legal processes, rather than a technical system-centric notion of accountable systems that do not entail any change in systems if they are flawed. Contestability is a human process with a human output, which should address any harm done to a person, regardless of any changes being made to the system that produced harm in the first place (even if such changes should also take place).

Because the best mechanisms to contest norms are those that can effectively pressure system designers or operators to change their system even if they are reluctant to do so, such as the legal system, we have analysed technical accountability mechanisms based on their ability to support the action of contesting computational systems via legal processes. From this perspective, transparency enhancing technologies understood as accountability mechanisms that include transparent logs of a system's operation, emerge as mechanisms that are more supportive of contestability than other accountability mechanisms based on providing assurances of compliance with given norms.

While work on designing technical systems has previously predominantly focused on building systems that match or comply with norms, there is scope to build

upon existing tools to create better transparency enhancing technologies that fill all the requirements that must for met to effectively enable accountability and, by extension, the ability to contest and change norms. Thus, this has implications for developers who wish to produce a change in existing systems and developers of new systems that may be designed with a model of decentralized governance that affords broader scope for changing the norms enacted by the system.

While there is justified scepticism of technical solutions to governance or regulatory issues, work in the field of law and policy that is concerned with the impact of computer systems should encourage and interact with the development of technical tools that can support their goals and empower the users that their work aims to help. Innovative new systems are not the only type of system that can cause harm, but there is necessarily a lag between technical innovation, the appearance of new systems, and of any effective governance or regulatory frameworks for these systems, which can leave users more exposed. Until such frameworks are put in place, it is all the more important for users to be able to contest the impact new systems can have, and this can also help guide the development of these frameworks by exposing system flaws or gaps in existing regulatory and governance approaches.

Finally, rather than simply allowing more legal cases to go forward, transparency and contestability can make security and reliability essential to system operators by making it harder for them to externalize the costs of a faulty system.

## Chapter 7

# Conclusion

This thesis has evaluated log based transparency enhancing technologies and the role they can play in making it possible to contest flawed systems and hold operators of these systems accountable.

First, as Chapter 4 shows, it is possible to systematize log based transparency enhancing technologies and characterize them based on the mechanisms they require to operate, and what security means for these mechanisms based on realistic threat models. For the most part, essential mechanisms like cryptographic logs exist and are usable. There are already transparency enhancing technologies like Certificate Transparency and cryptocurrencies that are built using these tools and work, securely, on a very large scale. This is encouraging and shows that building transparency enhancing technologies, even on a large scale, can be done and can bring positive change to a system.

Going beyond logs, however, issues quickly start to appear with existing sanitization, release and query, and external mechanisms. In the case of sanitization, privacy enhancing technologies are developed with privacy as their first concern, which creates trade-offs when it comes to satisfying the requirements of transparency.

In the case of release and query mechanisms, not much has been done in practice beyond querying logs for data (and verifying the inclusion and integrity of data in a log). Supporting an actual database with fully-fledged query capabilities on top of a log is something that can be done in theory but has not been done in practice.

Likewise, when it comes to integrating with external processes to make use of transparency enhancing technologies, much can be done in theory but little has so far been done in practice. Accountability in the Certificate Transparency ecosystem is down to whoever has decision-making power at Google, Mozilla, Microsoft, Brave, and other browser vendors. Regulations, and the development of transparency enhancing technologies for a broader range of systems, particularly those with which individuals directly interact, may positively impact this.

Chapter 4, which presents VAMS, illustrates these points quite well. We use two existing log primitives (HLF and Trillian) to build logs that satisfy the requirements of a system meant to provide transparent auditing of access to data records.

Concerning sanitization, however, our MultiBallot mechanism cannot satisfy the requirements of differential privacy, and no existing differentially private scheme supports the ability to easily find one's records in a privacy preserving dataset to verify its inclusion in the computation of statistics for an audit in the way that MultiBallot does. Thus, we have to recognize the trade-offs made with VAMS and use this as a starting point to develop better mechanisms that could satisfy all of our ideal requirements.

Nonetheless, VAMS is an example of how a transparency enhancing technology can be built to effectively improve an existing external process, in this the production of IPCO annual reports, and potentially extend this to users who could then also identify when they have been affected by mistakes and act on this information.

With Chapter 6 we address a more general question than how to build a transparency enhancing technology. What is the point of building a transparency enhancing technology when we could also just verify that a system functions correctly? The answer lies in the fact that transparency be useful for verifying more than simply the correct execution of a system, it can also be used to see what norms and system enforces and how it enforces them. Transparency then makes it possible to go further than accountability and test the legitimacy of these norms by contesting them via external processes.

## 7.1 Open Problems

**Requiring transparency** The question of how to require any kind of transparency remains open.

There may be requirements to implement some form of transparency to comply with, for example, ETSI requirements for providing a telecommunication service [294], a legal requirement to provide designated auditors with assistance (including IT infrastructure) [216], or the tight of access by the data subject to data held about them by a controller specified in the GDPR [359]. With the noticeable exception of the GDPR (which is very widespread), however, such transparency requirements are not public facing. Moreover, they are not as broad as what this thesis would argue for.

Laws and regulations can also work against transparency. Intellectual property (IP) law, in particular trade secrets that allow information about a system to remain confidential conflicts with requiring transparency and, as we have argued, an approach based on zero-knowledge proofs may not be practical or suitable to resolve this tension.

The argument could be made that any system operator that operates a system that could cause harm, especially systems that have been shown to have faults and that have caused harm (or are likely to have done so), should not be able to rely on IP law for obfuscation – vendors of electronic voting machines that have repeatedly been shown to have security or reliability issues exemplify this. Of course, many would argue against this because, as laid out in the introduction, operators of faulty systems would rather treat system faults as something that can be externalized to the public rather than a liability.

Thus, there is a debate involving the political and economic aspects of this question that must be resolved, but transparency must be considered more than it has been so far given the evidence that systems based on transparency do work (e.g., Certificate Transparency and cryptocurrencies).

**Implementing transparency** Like security and privacy, building transparency separately on top of an existing system (i.e., as an overlay) may not be as productive

as embedding it into the system itself by design. However, this naturally brings into question how this would change the workflow of system operators, many of which have yet to successfully embed security or privacy into their workflow and who might have a natural reluctance to implement transparency, and how this could be achieved with existing systems.

Aside from the open problem of requiring transparency, transparency engineering will have to be developed like security engineering [320] and privacy engineering [333, 334] before it and, like them, drawing on requirements engineering and other fields with extensive bodies of work to learn from, as well as developing an understanding of the contexts in which it is deployed.

**Interpreting transparency** If transparency is to be useful, it should be possible to find the evidence required to contest within the information it makes available. In court, evidence that is hard to interpret may mean relying on an expert witness to explain the evidence and someone to assert its reliability (e.g., a representative of the system operator or an expert witness depending on the legal system), but the focus in this thesis is also in facilitating access to evidence so that it is possible to get to the courtroom in the first place. It is still possible to rely on an expert before going to court, but they can also be difficult (and possibly expensive) to get in touch with without having already started the process of a legal dispute, so information made available through transparency should ideally be interpretable (at least to some extent) to anyone.

Starting with code, which is inscrutable to most people, it is unlikely that anyone who is not experienced with reading code (and the code base of the relevant system) in the first place will be unable to correctly assess how the code is intended to function and how it can fail simply by starring at it.

The Heartbleed Bug discovered in the OpenSSL cryptographic library makes this clear: years of using and occasionally inspecting code does not guarantee the discovery of one of the most impactful vulnerabilities in the history of a library on which the modern internet depends [360], even if the open-source nature of the code did ultimately result in the vulnerability being identified.

Searching for open source code itself is also a problem given the dependencies between different code bases, even if a software bill of materials (a list of software ingredients introduced to manage software supply chain risks) may help to outline these dependencies. GitHub can be thought of as providing a central repository of open source code that can be browsed and searched, but code is not indexed in the manner that other topics with centuries of library science devoted to their indexing are (e.g., the law, despite it's growing volume), and as a result, searches do not necessarily provide useful results given all the forks of a particular code repository (and its dependencies) that may exist, for example.

Going even further, code cannot be interpreted by a human in the same sense that a legal rule is interpreted (by a judge for example) and this is made more complicated by the role of a compiler in interpreting the code (and compiled code being even more inscrutable) and the presence of *comments* added by software developers to the code, which are not interpreted as code by the compiler, yet indicate what the code should do and how it is implemented, as well as the runtime environment.

Moving on to outputs, interpreting these can differ based on the existence of some ground truth.

In some cases, there may exist a ground truth that makes it clear the wrong output occurred. For example, A-level students in the UK were automatically assigned grades in place of having an exam because of the covid-19 pandemic in 2020 and the restrictions that were in place. The assigned grades were computed such that the grade distribution should match that of previous years, meaning that students were graded based on the performances of past students rather than their own and, as a result, some were penalized by a lower-than-expected grade if they had outperformed the norm. Because teacher predicted grades existed, however, it was obvious in some cases that students might be harmed by their assigned grades (e.g., if it meant they would not meet the conditions of their university offer) and that they could realistically have obtained a higher grade. This lead to protests and a government u-turn over the use of assigned grades [239]. (The system was not re-used so nothing was done to fix it.) More generally, in cases where code is used

purely to scale decision-making (e.g., applying a fixed decision tree) it may be feasible for a single or a few instances to go through the rules by hand and determine what the correct output should have been and interpret the system's output in that context.

In other cases, a ground truth may be harder to determine. if the code relies on large quantities of data and/or produces unexplainable outputs (as most complex neural networks do for instance), then the code's execution cannot meaningfully be compared to a manually computed output. In a legal setting, the presumption of innocence may also mean that there is no meaningful ground truth without going through the lengthy legal process that legal technology is deployed to replace. Interpreting an output may then be a very difficult task.

**Undoing harm** The key point that this thesis argues for is the idea that it must be possible to contest the outputs of a system and the effects they may have, particularly when they cause harm. Part of this should be to determine how harm can be undone.

When code causes harm, the harm is not in the code as it is written and executed, which changes the state of the system but not the state of the individual's world, but in the effect that the output of executing the code has, which can change the state of the individual's world. It makes sense, therefore, to focus on outputs as well as the code that produces these outputs.

Undoing harm could, therefore, mean two things. It could mean changing the code that produces harm, undoing the source of harm and preventing the same harm from occurring again. More importantly, for a victim of harm, it could mean reversing the state of the system to what it was before the harmful output and reversing the effect of that output on their life.

For example, a system that determines whether or not someone should be granted bail could include a function that allows an output to be erased from the system. In the physical world, such an output could be ignored if it is recognized as a fault because it only takes effect through other mechanisms such as a judge refusing to grant bail and law enforcement enforcing the decision. Of course, there

could be a cyber-physical system that also includes an automated judge and physical restraints, in which case the code that produces the output would have a physical effect, reinforcing the need to keep human decision-making in the loop so that decisions to ignore harmful outputs can be made.

More generally, an error is unlikely to be recognized and ignored. Automated systems are unlikely to be used if they involve supervision that is roughly equivalent to the task they are supposed to automate so they are more likely to operate in a mostly unsupervised way. As in civil litigation cases, compensation to the harmed individual could be based on the output that determines the harm done to them. Because harm is often not static in time (e.g., the harm that comes from a refused bail application increases with each day), the ability to identify and resolve faults, and the length of time this takes, should impact compensation. Punishment (as in criminal prosecutions) to the system operator could also be based on how the fault came to be; for example, because of accidental, negligent, or intentionally harmful operation of a system that could cause harm, and how it is resolved.

## 7.2 Closing thoughts

Like most academic work in this field, the hope is that this work can play a part in making systems better; for example, more secure, with better accountability for the inevitable flaws in systems, and better processes to deal with the impact of these flaws on people.

Redmiles, Bennett, and Kohno, have recently published an article that takes a critical view on power in computer security and privacy, which reflects on how the top-down power structures that govern systems and society impact how research is done [361].

It is easy to see this in practice. Funding for research and conferences flows from big tech companies or government agencies that produce the systems we are meant to critically analyse. This should include questioning whether a system should even exist, but can often be replaced by trying to simply mitigate the harms a system produces by, for example, making it privacy preserving when the lack of

privacy is not the primary issue or fairer when there should be no need to classify and discriminate in the first place.

As stated in the introduction to this thesis, research in information security has always dealt with issues of power over a system, typically by ensuring privilege over a system (e.g. through access control mechanisms), and this thesis has considered the opposite approach. If anything, transparency should allow systems to be questioned up to whether or not they should exist, and allow us to get a better idea of how to design systems that should exist.

# Bibliography


[1] Alexander Hicks. SoK: Log based transparency enhancing technologies. *arXiv preprint arXiv:2305.01378*, 2023.

[2] Alexander Hicks, Vasilios Mavroudis, Mustafa Al-Bassam, Sarah Meiklejohn, and Steven J Murdoch. Vams: Verifiable auditing of access to confidential data. *arXiv preprint arXiv:1805.04772*, 2018.

[3] Alexander Hicks. Transparency, compliance, and contestability when code is(n't) law. In *Proceedings of the 2022 New Security Paradigms Workshop*, NSPW '22, page 130–142, New York, NY, USA, 2023. Association for Computing Machinery.

[4] Sarah Azouvi, Guy Goren, Alexander Hicks, and Lioba Heimbach. Base fee manipulation in ethereum's eip-1559 transaction fee mechanism. *arXiv preprint arXiv:2304.11478*, 2023.

[5] Sarah Azouvi and Alexander Hicks. Decentralisation conscious players and system reliability. In *Financial Cryptography and Data Security: 26th International Conference, FC 2022, Grenada, May 2–6, 2022, Revised Selected Papers*, pages 426–443. Springer, 2022.

[6] Sarah Azouvi and Alexander Hicks. Sok: Tools for game theoretic models of security for cryptocurrencies. *Cryptoeconomic Systems*, 0(1), 4 2021. https://cryptoeconomicsystems.pubpub.org/pub/azouvi-sok-security.

[7] Alexander Hicks and Steven J Murdoch. Transparency enhancing technologies to make security protocols work for humans. In *Cambridge International Workshop on Security Protocols*, pages 3–10. Springer, 2019.

[8] Sarah Azouvi, Alexander Hicks, and Steven J Murdoch. Incentives in security protocols. In *Cambridge International Workshop on Security Protocols*, pages 132–141. Springer, 2018.

[9] Patrick McCorry, Alexander Hicks, and Sarah Meiklejohn. Smart contracts for bribing miners. In *International Conference on Financial Cryptography and Data Security*, pages 3–18. Springer, 2018.

[10] Dan Boneh and Victor Shoup. A graduate course in applied cryptography. *Available at https://toc.cryptobook.us/book.pdf*, 2020.

[11] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[12] National Institute of Standards and Technology. FIPS 180-4 Secure Hash Standard (SHS), August 2015.

[13] Law Commission. Electronic execution of documents, 2019.

[14] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

[15] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all np statements in zero-knowledge and a methodology of cryptographic protocol design. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 171–185. Springer, 1986.

[16] Saba Eskandarian, Eran Messeri, Joseph Bonneau, and Dan Boneh. Certificate transparency with privacy. *Proceedings on Privacy Enhancing Technologies*, 2017(4):329–344, 2017.

[17] Jonathan Frankle, Sunoo Park, Daniel Shaar, Shafi Goldwasser, and Daniel Weitzner. Practical accountability of secret processes. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 657–674, 2018.

[18] Shafi Goldwasser and Sunoo Park. Public accountability vs. secret laws: Can they coexist?: A cryptographic proposal. In *Proceedings of the 2017 on Workshop on Privacy in the Electronic Society*, pages 99–110. ACM, 2017.

[19] Kenneth A Bamberger, Ran Canetti, Shafi Goldwasser, Rebecca Wexler, and Evan J Zimmerman. Verification dilemmas in law and the promise of zero-knowledge proofs. *Berkeley Technology Law Journal*, 37(1), 2022.

[20] Joshua A Kroll, Solon Barocas, Edward W Felten, Joel R Reidenberg, David G Robinson, and Harlan Yu. Accountable algorithms. *U. Pa. L. Rev.*, 165:633, 2016.

[21] Zcash. Parameter generation, 2017.

[22] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. *IACR Cryptology ePrint Archive*, 2018:46, 2018.

[23] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 769–793. Springer, 2020.

[24] Sean Bowe, Ariel Gabizon, and Matthew D Green. A multi-party protocol for constructing the public parameters of the pinocchio zk-snark. In *International Conference on Financial Cryptography and Data Security*, pages 64–77. Springer, 2018.

[25] Sean Bowe, Jack Grigg, and Daira Hopwood. Halo: Recursive proof composition without a trusted setup. *IACR Cryptol. ePrint Arch.*, 2019:1021, 2019.

[26] Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-snarks. In *Annual International Cryptology Conference*, pages 698–728. Springer, 2018.

[27] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge snarks from linear-size universal and updatable structured reference strings. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2111–2128, 2019.

[28] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 315–334. IEEE, 2018.

[29] Melissa Chase and Sarah Meiklejohn. Transparency overlays and applications. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 168–179, 2016.

[30] Michael T Goodrich, Evgenios M Kornaropoulos, Michael Mitzenmacher, and Roberto Tamassia. Auditable data structures. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 285–300. IEEE, 2017.

[31] Roberto Tamassia. Authenticated data structures. In *European symposium on algorithms*, pages 2–5. Springer, 2003.

[32] Andrew Miller, Michael Hicks, Jonathan Katz, and Elaine Shi. Authenticated data structures, generically. *ACM SIGPLAN Notices*, 49(1):411–423, 2014.

[33] Ralph C Merkle. A digital signature based on a conventional encryption function. In *Advances in Cryptology—CRYPTO'87: Proceedings 7*, pages 369–378. Springer, 1988.

[34] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Web document., 2008.

[35] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[36] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. Sok: Consensus in the age of blockchains. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, AFT '19, page 183–198, New York, NY, USA, 2019. Association for Computing Machinery.

[37] Dorothy E Denning and Peter J Denning. The tracker: A threat to statistical database security. *ACM Transactions on Database Systems (TODS)*, 4(1):76–96, 1979.

[38] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 111–125. IEEE, 2008.

[39] Yves-Alexandre De Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports*, 3(1):1–5, 2013.

[40] Yves-Alexandre De Montjoye, Laura Radaelli, Vivek Kumar Singh, and Alex "Sandy" Pentland. Unique in the shopping mall: On the reidentifiability of credit card metadata. *Science*, 347(6221):536–539, 2015.

[41] Luc Rocher, Julien M Hendrickx, and Yves-Alexandre De Montjoye. Estimating the success of re-identifications in incomplete datasets using generative models. *Nature communications*, 10(1):1–9, 2019.

[42] Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. In *2009 30th IEEE symposium on security and privacy*, pages 173–187. IEEE, 2009.

[43] Andrea Gadotti, Florimond Houssiau, Luc Rocher, Benjamin Livshits, and Yves-Alexandre De Montjoye. When the signal is in the noise: Exploiting diffix's sticky noise. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1081–1098, 2019.

[44] Simson Garfinkel, John M Abowd, and Christian Martindale. Understanding database reconstruction attacks on public data. *Communications of the ACM*, 62(3):46–53, 2019.

[45] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[46] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 106–115. IEEE, 2007.

[47] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*. IEEE, 2006.

[48] Jianneng Cao, Panagiotis Karras, Chedy Raïssi, and Kian-Lee Tan. $\rho$-uncertainty: inference-proof transaction anonymization. *Proceedings of the VLDB Endowment*, 3(1-2):1033–1044, 2010.

[49] Josep Domingo-Ferrer and Vicenç Torra. A critique of k-anonymity and some of its enhancements. In *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, pages 990–993. IEEE, 2008.

[50] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. *Calibrating Noise to Sensitivity in Private Data Analysis*, pages 265–284. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[51] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3–es, 2007.

[52] Daniel Kifer and Ashwin Machanavajjhala. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 193–204, 2011.

[53] Arpita Ghosh and Robert Kleinberg. Inferential privacy guarantees for differentially private mechanisms. *arXiv preprint arXiv:1603.01508*, 2016.

[54] Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

[55] Damien Desfontaines and Balázs Pejó. Sok: Differential privacies. *Proceedings on Privacy Enhancing Technologies*, 2020(2):288–313, 2020.

[56] Michael Carl Tschantz, Shayak Sen, and Anupam Datta. Sok: Differential privacy as a causal property. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 354–371. IEEE, 2020.

[57] Alexandra Wood, Micah Altman, Aaron Bembenek, Mark Bun, Marco Gaboardi, James Honaker, Kobbi Nissim, David R O'Brien, Thomas Steinke, and Salil Vadhan. Differential privacy: A primer for a non-technical audience. *Vand. J. Ent. & Tech. L.*, 21:209, 2018.

[58] H Page, C Cabot, and K Nissim. Differential privacy an introduction for statistical agencies. *NSQR. Government Statistical Service*, 2018.

[59] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067, 2014.

[60] Differential Privacy Team. Learning with privacy at scale, 2017.

[61] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems*, pages 3571–3580, 2017.

[62] John M Abowd. The us census bureau adopts differential privacy. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2867–2867, 2018.

[63] Simson Garfinkel. Deploying differential privacy for the 2020 census of population and housing. 2019.

[64] Simson L. Garfinkel and Philip Leclerc. Randomness concerns when deploying differential privacy, 2020.

[65] Fredrik Andersson, John M Abowd, Matthew Graham, Jeremy Wu, and Lars Vilhuber. Formal privacy guarantees and analytical validity of onthemap public-use data. 2009.

[66] Ashwin Machanavajjhala, Daniel Kifer, John Abowd, Johannes Gehrke, and Lars Vilhuber. Privacy: Theory meets practice on the map. In *2008 IEEE 24th international conference on data engineering*, pages 277–286. IEEE, 2008.

[67] Aloni Cohen and Kobbi Nissim. Towards formalizing the gdpr's notion of singling out. *Proceedings of the National Academy of Sciences*, 117(15):8344–8352, 2020.

[68] Kobbi Nissim, Aaron Bembenek, Alexandra Wood, Mark Bun, Marco Gaboardi, Urs Gasser, David R O'Brien, Thomas Steinke, and Salil Vadhan. Bridging the gap between computer science and legal approaches to privacy. *Harv. JL & Tech.*, 31:687, 2017.

[69] Ronald L Rivest. The ThreeBallot voting system. 2006.

[70] Ronald L Rivest and Warren D Smith. Three voting protocols: ThreeBallot, VAV, and Twin. *USENIX/ACCURATE Electronic Voting Technology (EVT 2007)*, 2007.

[71] Harvey Jones, Jason Juang, and Greg Belote. ThreeBallot in the field. *Term paper for MIT course*, 6, 2006.

[72] Andrew W Appel. How to defeat Rivest's ThreeBallot voting system. *Manuskrypt, pazdziernik*, 2006.

[73] Kevin Henry, Douglas R Stinson, and Jiayuan Sui. The effectiveness of receipt-based attacks on ThreeBallot. *IEEE Transactions on Information Forensics and Security*, 4(4):699–707, 2009.

[74] Jacek Cichon, Miroslaw Kutylowski, and Bogdan Weglorz. Short ballot assumption and ThreeBallot voting protocol. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 585–598. Springer, 2008.

[75] Charlie EM Strauss. A critical review of the triple ballot voting system, part 2: Cracking the triple ballot encryption. *Unpublished draft,* `http://cems.browndogs.org/pub/voting/tripletrouble.pdf`, 74, 2006.

[76] Charlie Strauss. The trouble with triples: A critical review of the triple ballot (3ballot) scheme, part 1. *Verified Voting New Mexico*, 2006.

[77] Patrick Murmann and Simone Fischer-Hübner. Tools for achieving usable ex post transparency: a survey. *IEEE Access*, 5:22965–22991, 2017.

[78] Hans Hedbom. A survey on transparency tools for enhancing privacy. In *IFIP Summer School on the Future of Identity in the Information Society*, pages 67–82. Springer, 2008.

[79] Milena Janic, Jan Pieter Wijbenga, and Thijs Veugen. Transparency enhancing tools (tets): an overview. In *2013 Third Workshop on Socio-Technical Aspects in Security and Trust*, pages 18–25. IEEE, 2013.

[80] Christian Zimmermann. A categorization of transparency-enhancing technologies. *arXiv preprint arXiv:1507.04914*, 2015.

[81] Dayana Spagnuelo, Ana Ferreira, and Gabriele Lenzini. Accomplishing transparency within the general data protection regulation. In *ICISSP*, pages 114–125, 2019.

[82] Dayana Spagnuelo, Ana Ferreira, and Gabriele Lenzini. Transparency enhancing tools and the gdpr: Do they match? In *International Conference on Information Systems Security and Privacy*, pages 162–185. Springer, 2019.

[83] Gaurav Panwar, Roopa Vishwanathan, Satyajayant Misra, and Austin Bos. Sampl: Scalable auditability of monitoring processes using public ledgers. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2249–2266, 2019.

[84] Adam Bates, Kevin RB Butler, Micah Sherr, Clay Shields, Patrick Traynor, and Dan Wallach. Accountable wiretapping–or–i know they can hear you now. *Journal of Computer Security*, 23(2):167–195, 2015.

[85] Marcela S Melara, Aaron Blankstein, Joseph Bonneau, Edward W Felten, and Michael J Freedman. Coniks: Bringing key transparency to end users. In *USENIX Security Symposium*, volume 2015, pages 383–398, 2015.

[86] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, volume 22, pages 207–216. ACM, 1993.

[87] Alexandre Evfimievski, Ramakrishnan Srikant, Rakesh Agrawal, and Johannes Gehrke. Privacy preserving mining of association rules. *Information Systems*, 29(4):343–364, 2004.

[88] Nan Zhang, Shengquan Wang, and Wei Zhao. A new scheme on privacy preserving association rule mining. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 484–495. Springer, 2004.

[89] Cynthia Dwork. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation*, pages 1–19. Springer, 2008.

[90] Mário S Alvim, Miguel E Andrés, Konstantinos Chatzikokolakis, Pierpaolo Degano, and Catuscia Palamidessi. Differential privacy: on the trade-off between utility and information leakage. In *International Workshop on Formal Aspects in Security and Trust*, pages 39–54. Springer, 2011.

[91] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N Rothblum. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 715–724. ACM, 2010.

[92] Jaewoo Lee and Chris Clifton. How much is enough? choosing $\varepsilon$ for differential privacy. In *International Conference on Information Security*, pages 325–340. Springer, 2011.

[93] Rui Chen, Noman Mohammed, Benjamin CM Fung, Bipin C Desai, and Li Xiong. Publishing set-valued data via differential privacy. *Proceedings of the VLDB Endowment*, 4(11):1087–1098, 2011.

[94] Arjun Narayan, Ariel Feldman, Antonis Papadimitriou, and Andreas Haeberlen. Verifiable differential privacy. In *Proceedings of the Tenth European Conference on Computer Systems*, page 28. ACM, 2015.

[95] Andreas Haeberlen, Benjamin C. Pierce, and Arjun Narayan. Differential privacy under fire. In *20th USENIX Security Symposium, San Francisco, CA, USA, August 8-12, 2011, Proceedings*, 2011.

[96] Maranke Wieringa. What to account for when accounting for algorithms: A systematic literature review on algorithmic accountability. In *Proceedings of*

*the 2020 Conference on Fairness, Accountability, and Transparency*, pages 1–18, 2020.

[97] Mark Bovens. Analysing and assessing accountability: A conceptual framework 1. *European law journal*, 13(4):447–468, 2007.

[98] Roger Taylor and Tim Kelsey. *Transparency and the open society: Practical lessons for effective policy*. Policy Press, 2016.

[99] Joan Feigenbaum, Aaron D Jaggard, and Rebecca N Wright. Towards a formal model of accountability. In *Proceedings of the 2011 New security paradigms workshop*, pages 45–56, 2011.

[100] Joan Feigenbaum, James A Hendler, Aaron D Jaggard, Daniel J Weitzner, and Rebecca N Wright. Accountability and deterrence in online life. In *Proceedings of the 3rd International Web Science Conference*, pages 1–7, 2011.

[101] Joshua A Kroll, Joanna Huey, Solon Barocas, Edward W Felten, Joel R Reidenberg, David G Robinson, and Harlan Yu. Accountable algorithms. *University of Pennsylvania Law Review*, 165:633, 2017.

[102] Deven R Desai and Joshua A Kroll. Trust but verify: A guide to algorithms and the law. *Harv. JL & Tech.*, 31:1, 2017.

[103] Steven J Murdoch and Ross Anderson. Security protocols and evidence: Where many payment systems fail. In *International Conference on Financial Cryptography and Data Security*, pages 21–32. Springer, 2014.

[104] Henrietta Lyons, Eduardo Velloso, and Tim Miller. Conceptualising contestability: Perspectives on contesting algorithmic decisions. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW1):1–25, 2021.

[105] Cambridge Dictionary. Transparency.

[106] Wiktionary. Transparent.

[107] Louis Dembitz Brandeis. *Other peoples money, and how the bankers use it / by Louis D. Brandeis.* Stokes, New York, 1914.

[108] Matteo Turilli and Luciano Floridi. The ethics of information transparency. *Ethics and Information Technology*, 11(2):105–112, 2009.

[109] Department for Digital, Culture, Media & Sport. Uk open government national action plan, July 2019.

[110] Barack Obama. Transparency and open government. *Memorandum for the heads of executive departments and agencies*, 2009.

[111] David Robinson, Harlan Yu, William P Zeller, and Edward W Felten. Government data and the invisible hand. *Yale JL & Tech.*, 11:159, 2008.

[112] Auriol Degbelo and Tomi Kauppinen. Increasing transparency through web maps. In *Companion Proceedings of the The Web Conference 2018*, pages 899–904, 2018.

[113] Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation), 2016.

[114] George A Akerlof. The market for "lemons": Quality uncertainty and the market mechanism. In *Uncertainty in economics*, pages 235–251. Elsevier, 1978.

[115] Ross Anderson and Tyler Moore. The economics of information security. *science*, 314(5799):610–613, 2006.

[116] Ross Anderson. Why information security is hard-an economic perspective. In *Seventeenth Annual Computer Security Applications Conference*, pages 358–365. IEEE, 2001.

[117] Jerome H Saltzer and Michael D Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

[118] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021.

[119] Sarah Holland, Ahmed Hosny, Sarah Newman, Joshua Joseph, and Kasia Chmielinski. The dataset nutrition label: A framework to drive higher data quality standards. *arXiv preprint arXiv:1805.03677*, 2018.

[120] The Data Nutrition Project. The data nutrition project, 2021.

[121] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 220–229, 2019.

[122] Patrick Gage Kelley, Joanna Bresee, Lorrie Faith Cranor, and Robert W Reeder. A" nutrition label" for privacy. In *Proceedings of the 5th Symposium on Usable Privacy and Security*, pages 1–12, 2009.

[123] Patrick Gage Kelley, Lucian Cesca, Joanna Bresee, and Lorrie Faith Cranor. Standardizing privacy notices: an online study of the nutrition label approach. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems*, pages 1573–1582, 2010.

[124] Daniel J Weitzner, Harold Abelson, Tim Berners-Lee, Joan Feigenbaum, James Hendler, and Gerald Jay Sussman. Information accountability. *Communications of the ACM*, 51(6):82–87, 2008.

[125] Motahhare Eslami, Kristen Vaccaro, Min Kyung Lee, Amit Elazari Bar On, Eric Gilbert, and Karrie Karahalios. User attitudes towards algorithmic opacity and transparency in online reviewing platforms. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19,

page 1–14, New York, NY, USA, 2019. Association for Computing Machinery.

[126] Joshua A Kroll. The fallacy of inscrutability. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 376(2133):20180084, 2018.

[127] Paul Marshall, James Christie, B Ladkin, Bev Littlewood, Stephen Mason, Martin Newby, Jonathan Rogers, Harold Thimbleby, and M Thomas. Recommendations for the probity of computer evidence. *Digital Evidence and Electronic Signature Law Review*, 18, 2020.

[128] Jasper Jolly. Uk government sets aside up to £233m to cover post office payouts, 2021.

[129] Steven M Bellovin, Matt Blaze, Susan Landau, and Brian Owsley. Seeking the source: Criminal defendants' constitutional right to source code. *Ohio St. Tech. LJ*, 17:1, 2021.

[130] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias: There's software used across the country to predict future criminals. *And it's biased against blacks. ProPublica*, 23:77–91, 2016.

[131] Solon Barocas and Andrew D Selbst. Big data's disparate impact. *Calif. L. Rev.*, 104:671, 2016.

[132] Investigatory Powers Commitioner's Office. Annual report 2018, March 2020.

[133] Ethan S Bernstein. The transparency paradox: A role for privacy in organizational learning and operational control. *Administrative Science Quarterly*, 57(2):181–216, 2012.

[134] D Elliott Bell and Leonard J LaPadula. Secure computer systems: Mathematical foundations. Technical report, MITRE CORP BEDFORD MA, 1973.

[135] Kenneth J Biba. Integrity considerations for secure computer systems. Technical report, MITRE CORP BEDFORD MA, 1977.

[136] Ilia Shumailov, Zakhar Shumaylov, Dmitry Kazhdan, Yiren Zhao, Nicolas Papernot, Murat A Erdogdu, and Ross Anderson. Manipulating sgd with data ordering attacks. *arXiv preprint arXiv:2104.09667*, 2021.

[137] Ethan Bernstein. The transparency trap. *Harvard Business Review*, 92(10):58–66, 2014.

[138] Jonathan Fox. The uncertain relationship between transparency and accountability. *Development in practice*, 17(4-5):663–671, 2007.

[139] Amitai Etzioni. Is transparency the best disinfectant? *Journal of Political Philosophy*, 18(4):389–404, 2010.

[140] Laurence Ferry and Peter Eckersley. Accountability and transparency: a nuanced response to etzioni. *Public Administration Review*, 75(1):11, 2015.

[141] Idris Adjerid, Alessandro Acquisti, Laura Brandimarte, and George Loewenstein. Sleights of privacy: Framing, disclosures, and the limits of transparency. In *Proceedings of the ninth symposium on usable privacy and security*, pages 1–11, 2013.

[142] Alessandro Acquisti, Idris Adjerid, and Laura Brandimarte. Gone in 15 seconds: The limits of privacy transparency and control. *IEEE Security & Privacy*, 11(4):72–74, 2013.

[143] Tianshi Li, Kayla Reiman, Yuvraj Agarwal, Lorrie Faith Cranor, and Jason I Hong. Understanding challenges for developers to create accurate privacy nutrition labels. In *CHI Conference on Human Factors in Computing Systems*, pages 1–24, 2022.

[144] Harlan Yu and David G Robinson. The new ambiguity of open government. *UCLA L. Rev. Discourse*, 59:178, 2011.

[145] Karen EC Levy and David Merritt Johns. When open data is a trojan horse: The weaponization of transparency in science and governance. *Big Data & Society*, 3(1):2053951715621568, 2016.

[146] Onora O'neill. *A question of trust: The BBC Reith Lectures 2002*. Cambridge University Press, 2002.

[147] Onora O Neill. Transparency and the ethics of communication. In *Proceedings-British Academy*, volume 1, pages 75–90. Oxford University Press, 2006.

[148] Ben Worthy. More open but not more trusted? the effect of the freedom of information act 2000 on the united kingdom central government. *Governance*, 23(4):561–582, 2010.

[149] Vincent Mabillard and Martial Pasquier. Transparency and trust in government (2007–2014): A comparative study. *NISPAcee Journal of Public Administration and Policy*, 9(2):69–92, 2016.

[150] Cynthia Stohl, Michael Stohl, and Paul M Leonardi. Digital age— managing opacity: Information visibility and the paradox of transparency in the digital age. *International Journal of Communication*, 10:15, 2016.

[151] Arthur Harris Adelberg. Narrative disclosures contained in financial reports: means of communication or manipulation? *Accounting and Business Research*, 9(35):179–190, 1979.

[152] Mike Ananny and Kate Crawford. Seeing without knowing: Limitations of the transparency ideal and its application to algorithmic accountability. *new media & society*, 20(3):973–989, 2018.

[153] Adrian Weller. Transparency: motivations and challenges. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 23–40. Springer, 2019.

[154] Jenna Burrell. How the machine 'thinks': Understanding opacity in machine learning algorithms. *Big Data & Society*, 3(1):2053951715622512, 2016.

[155] Jef Ausloos and Pierre Dewitte. Shattering one-way mirrors. data subject access rights in practice. *Data Subject Access Rights in Practice (January 20, 2018). International Data Privacy Law*, 8(1):4–28, 2018.

[156] Brent Mittelstadt. Automation, algorithms, and politics— auditing for transparency in content personalization systems. *International Journal of Communication*, 10:12, 2016.

[157] Ross Anderson. Open and closed systems are equivalent (that is, in an ideal world), 2005.

[158] Guido Schryen. Is open source security a myth? *Communications of the ACM*, 54(5):130–140, 2011.

[159] Freedom of information act 2000, 2000.

[160] Reuters. Uber drivers consider appeal in dutch case over data access, 2021.

[161] Joshua A Kroll. Outlining traceability: A principle for operationalizing accountability in computing systems. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 758–771, 2021.

[162] David Devecsery, Michael Chow, Xianzheng Dou, Jason Flinn, and Peter M Chen. Eidetic systems. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pages 525–540, 2014.

[163] Mihir Bellare and Bennet Yee. Forward integrity for secure audit logs. Technical report, Citeseer, 1997.

[164] Bruce Schneier and John Kelsey. Cryptographic support for secure logs on untrusted machines. In *USENIX Security Symposium*, volume 98, pages 53–62. San Antonio, TX, 1998.

[165] Bruce Schneier and John Kelsey. Secure audit logs to support computer forensics. *ACM Transactions on Information and System Security (TISSEC)*, 2(2):159–176, 1999.

[166] Cheun Ngen Chong, Zhonghong Peng, and Pieter H Hartel. Secure audit logging with tamper-resistant hardware. In *Security and Privacy in the Age of Uncertainty: IFIP TC11 18 th International Conference on Information Security (SEC2003) May 26–28, 2003, Athens, Greece 18*, pages 73–84. Springer, 2003.

[167] Jason E Holt and Kent E Seamons. Logcrypt: forward security and public verification for secure audit logs. *Cryptology ePrint Archive*, 2005.

[168] Brent R Waters, Dirk Balfanz, Glenn Durfee, and Diana K Smetters. Building an encrypted and searchable audit log. In *NDSS*, volume 4, pages 5–6, 2004.

[169] Di Ma and Gene Tsudik. A new approach to secure logging. *ACM Transactions on Storage (TOS)*, 5(1):1–21, 2009.

[170] Tobias Pulls, Roel Peeters, and Karel Wouters. Distributed privacy-preserving transparency logging. In *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*, pages 83–94, 2013.

[171] Scott A Crosby and Dan S Wallach. Efficient data structures for tamper-evident logging. In *USENIX Security Symposium*, pages 317–334, 2009.

[172] Ben Laurie. Certificate transparency. *Communications of the ACM*, 57(10):40–46, 2014.

[173] Ben Laurie Adam Eijdenberg and Al Cutter. Trillian – verifiable data structures, 2017.

[174] Google. Trillian, 2017.

[175] Google. Key transparency, 2017.

[176] Mark Dermot Ryan. Enhanced certificate transparency and end-to-end encrypted mail. In *NDSS*, pages 1–14, 2014.

[177] Tobias Pulls and Roel Peeters. Balloon: A forward-secure append-only persistent authenticated data structure. In *European Symposium on Research in Computer Security*, pages 622–641. Springer, 2015.

[178] Roel Peeters and Tobias Pulls. Insynd: Improved privacy-preserving transparency logging. In *European Symposium on Research in Computer Security*, pages 121–139. Springer, 2016.

[179] Ben Laurie and Emilia Kasper. Revocation transparency. *Google Research, September*, 2012.

[180] Rasmus Dahlberg, Tobias Pulls, and Roel Peeters. Efficient sparse merkle trees. In *Nordic Conference on Secure IT Systems*, pages 199–215. Springer, 2016.

[181] Michael P Andersen, Sam Kumar, Moustafa AbdelBaky, Gabe Fierro, John Kolb, Hyung-Sin Kim, David E Culler, and Raluca Ada Popa. Wave: A decentralized authorization framework with transitive delegation. In *Proceedings of the 28th USENIX Security Symposium*. Univ. of California, Berkeley, CA (United States), 2019.

[182] Yuncong Hu, Kian Hooshmand, Harika Kalidhindi, Seung Jin Yang, and Raluca Ada Popa. Merklê 2: A low-latency transparency log system. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 285–303. IEEE, 2021.

[183] Daniel Reijsbergen, Aung Maw, Zheng Yang, Tien Tuan Anh Dinh, and Jianying Zhou. Tap: Transparent and privacy-preserving data services. *arXiv preprint arXiv:2210.11702*, 2022.

[184] Alin Tomescu, Vivek Bhupatiraju, Dimitrios Papadopoulos, Charalampos Papamanthou, Nikos Triandopoulos, and Srinivas Devadas. Transparency

logs via append-only authenticated dictionaries. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1299–1316, 2019.

[185] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[186] Hoang-Long Nguyen, Claudia-Lavinia Ignat, and Olivier Perrin. Trusternity: auditing transparent log server with blockchain. In *Companion Proceedings of the The Web Conference 2018*, pages 79–80, 2018.

[187] Joseph Bonneau. Ethiks: Using ethereum to audit a coniks key transparency log. In *International Conference on Financial Cryptography and Data Security*, pages 95–105. Springer, 2016.

[188] Alin Tomescu and Srinivas Devadas. Catena: Efficient non-equivocation via bitcoin. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 393–409. IEEE, 2017.

[189] Harjasleen Malvai, Lefteris Kokoris-Kogias, Alberto Sonnino, Esha Ghosh, Ercan Oztürk, Kevin Lewi, and Sean Lawlor. Parakeet: Practical key transparency for end-to-end encrypted messaging. *Cryptology ePrint Archive*, 2023.

[190] Melissa Chase, Apoorvaa Deshpande, Esha Ghosh, and Harjasleen Malvai. Seemless: Secure end-to-end encrypted messaging with less trust. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 1639–1656, 2019.

[191] Yupeng Zhang, Jonathan Katz, and Charalampos Papamanthou. Integridb: Verifiable sql for outsourced databases. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1480–1491, 2015.

[192] Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. vsql: Verifying arbitrary sql queries over dy-

namic outsourced databases. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 863–880. IEEE, 2017.

[193] Yanqing Peng, Min Du, Feifei Li, Raymond Cheng, and Dawn Song. Falcondb: Blockchain-based collaborative database. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 637–652, 2020.

[194] Elias Grünewald and Frank Pallas. Tilt: A gdpr-aligned transparency information language and toolkit for practical privacy engineering. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 636–646, 2021.

[195] Lucianna Kiffer, Dave Levin, and Alan Mislove. Stick a fork in it: Analyzing the ethereum network partition. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, pages 94–100, 2017.

[196] Devon O'Brien, Ryan Sleevi, and Andrew Whalley. Chrome's plan to distrust symantec certificates, 2017.

[197] Julio Angulo, Simone Fischer-Hübner, Tobias Pulls, and Erik Wästlund. Usable transparency with the data track: a tool for visualizing data disclosures. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, pages 1803–1808, 2015.

[198] Patrick Murmann. Usable transparency for enhancing privacy in mobile health apps. In *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*, pages 440–442, 2018.

[199] Patrick Murmann, Delphine Reinhardt, and Simone Fischer-Hübner. To be, or not to be notified: eliciting privacy notification preferences for online mhealth services. In *ICT Systems Security and Privacy Protection: 34th IFIP TC 11 International Conference, SEC 2019, Lisbon, Portugal, June 25-27, 2019, Proceedings 34*, pages 209–222. Springer, 2019.

[200] Patrick Murmann. Eliciting design guidelines for privacy notifications in mhealth environments. *International Journal of Mobile Human Computer Interaction (IJMHCI)*, 11(4):66–83, 2019.

[201] Simone Fischer-Hübner, Julio Angulo, Farzaneh Karegar, and Tobias Pulls. Transparency, privacy and trust–technology for tracking and controlling my data disclosures: Does this work? In *IFIP International Conference on Trust Management*, pages 3–14. Springer, 2016.

[202] Simone Fischer-Hübner, Julio Angulo, and Tobias Pulls. How can cloud users be supported in deciding on, tracking and controlling how their data are used? In *IFIP PrimeLife International Summer School on Privacy and Identity Management for Life*, pages 77–92. Springer, 2013.

[203] Stephen Mason and Daniel Seng. *Electronic Evidence and Electronic Signatures*. University of London Press, 2021.

[204] Emilee Rader, Kelley Cotter, and Janghee Cho. Explanations as mechanisms for supporting algorithmic transparency. In *Proceedings of the 2018 CHI conference on human factors in computing systems*, pages 1–13, 2018.

[205] Chris Norval, Kristin Cornelius, Jennifer Cobbe, and Jatinder Singh. Disclosure by design: Designing information disclosures to support meaningful transparency and accountability. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 679–690, 2022.

[206] Christopher Hood. What happens when transparency meets blame-avoidance? *Public Management Review*, 9(2):191–210, 2007.

[207] Mariano Di Martino, Pieter Robyns, Winnie Weyts, Peter Quax, Wim Lamotte, and Ken Andries. Personal information leakage by abusing the {GDPR}'right of access'. In *Fifteenth Symposium on Usable Privacy and Security ({SOUPS} 2019)*, 2019.

[208] Jatinder Singh and Jennifer Cobbe. The security implications of data subject rights. *IEEE Security & Privacy*, 17(6):21–30, 2019.

[209] Wikipedia. Usage share of web browsers, 2022.

[210] Sarah Meiklejohn, Joe DeBlasio, Devon O'Brien, Chris Thompson, Kevin Yeo, and Emily Stark. Sok: Sct auditing in certificate transparency. *arXiv preprint arXiv:2203.01661*, 2022.

[211] Protocol Labs. Filecoin: A decentralized storage network, 2017.

[212] Ben Weinshel, Miranda Wei, Mainack Mondal, Euirim Choi, Shawn Shan, Claire Dolin, Michelle L Mazurek, and Blase Ur. Oh, the places you've been! user reactions to longitudinal transparency about third-party web tracking and inferencing. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 149–166, 2019.

[213] Shagun Jhaver, Amy Bruckman, and Eric Gilbert. Does transparency in moderation really matter? user behavior after content removal explanations on reddit. *Proceedings of the ACM on Human-Computer Interaction*, 3(CSCW):1–27, 2019.

[214] Athanasios Andreou, Giridhari Venkatadri, Oana Goga, Krishna Gummadi, Patrick Loiseau, and Alan Mislove. Investigating ad transparency mechanisms in social media: A case study of facebook's explanations. In *NDSS 2018-Network and Distributed System Security Symposium*, pages 1–15, 2018.

[215] Tobias Urban, Martin Degeling, Thorsten Holz, and Norbert Pohlmann. " your hashed ip address: Ubuntu." perspectives on transparency tools for online advertising. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pages 702–717, 2019.

[216] Investigatory Powers Act, 2016.

[217] Ben Wagner, Krisztina Rozgonyi, Marie-Therese Sekwenz, Jennifer Cobbe, and Jatinder Singh. Regulating transparency? facebook, twitter and the german network enforcement act. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 261–271, 2020.

[218] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F Wenisch, Yuval Yarom, and Raoul Strackx. Foreshadow: Extracting the keys to the intel {SGX} kingdom with transient {Out-of-Order} execution. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 991–1008, 2018.

[219] Hal Varian. System reliability and free riding. In *Economics of information security*, pages 1–15. Springer, 2004.

[220] Andrei Lapets, Frederick Jansen, Kinan Dak Albab, Rawane Issa, Lucy Qin, Mayank Varia, and Azer Bestavros. Accessible privacy-preserving web-based data analysis for assessing and addressing economic inequalities. In *Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies*, pages 1–5, 2018.

[221] Ehsan Toreini, Siamak F Shahandashti, and Feng Hao. Texture to the rescue: Practical paper fingerprinting based on texture patterns. *ACM Transactions on Privacy and Security (TOPS)*, 20(3):1–29, 2017.

[222] Ashlesh Sharma, Lakshminarayanan Subramanian, and Eric A Brewer. Paperspeckle: microscopic fingerprinting of paper. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 99–110, 2011.

[223] William Clarkson, Tim Weyrich, Adam Finkelstein, Nadia Heninger, J Alex Halderman, and Edward W Felten. Fingerprinting blank paper using commodity scanners. In *2009 30th IEEE Symposium on Security and Privacy*, pages 301–314. IEEE, 2009.

[224] James DR Buchanan, Russell P Cowburn, Ana-Vanessa Jausovec, Dorothee Petit, Peter Seem, Gang Xiong, Del Atkinson, Kate Fenton, Dan A Allwood, and Matthew T Bryan. 'fingerprinting'documents and packaging. *Nature*, 436(7050):475–475, 2005.

[225] Wiwi Samsul, Henri P Uranus, and MD Birowosuto. Recognizing document's originality by laser surface authentication. In *2010 second international conference on advances in computing, control, and telecommunication technologies*, pages 37–40. IEEE, 2010.

[226] Zhengxiong Li, Aditya Singh Rathore, Chen Song, Sheng Wei, Yanzhi Wang, and Wenyao Xu. Printracker: Fingerprinting 3d printers using commodity scanners. In *Proceedings of the 2018 ACM sigsac conference on computer and communications security*, pages 1306–1323, 2018.

[227] Francesco Guarnera, Dario Allegra, Oliver Giudice, Filippo Stanco, and Sebastiano Battiato. A new study on wood fibers textures: documents authentication through lbp fingerprint. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 4594–4598. IEEE, 2019.

[228] Frerik van Beijnum, EG van Putten, KL Van der Molen, and AP Mosk. Recognition of paper samples by correlation of their speckle patterns. *arXiv preprint physics/0610089*, 2006.

[229] S. Wang, E. Toreini, and F. Hao. Anti-counterfeiting for polymer banknotes based on polymer substrate fingerprinting. *IEEE Transactions on Information Forensics and Security*, 16:2823–2835, 2021.

[230] Martin Henze, Daniel Kerpen, Jens Hiller, Michael Eggert, David Hellmanns, Erik Mühmer, Oussama Renuli, Henning Maier, Christian Stüble, Roger Häußling, et al. Towards transparent information on individual cloud service usage. In *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 366–370. IEEE, 2016.

[231] Max Van Kleek, Ilaria Liccardi, Reuben Binns, Jun Zhao, Daniel J Weitzner, and Nigel Shadbolt. Better the devil you know: Exposing the data sharing practices of smartphone apps. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 5208–5220, 2017.

[232] Yuanchun Li, Fanglin Chen, Toby Jia-Jun Li, Yao Guo, Gang Huang, Matthew Fredrikson, Yuvraj Agarwal, and Jason I Hong. Privacystreams: Enabling transparency in personal data processing for mobile apps. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(3):1–26, 2017.

[233] Sourya Joyee De and Daniel Le Métayer. Privacy risk analysis to enable informed privacy settings. In *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 95–102. IEEE, 2018.

[234] the New York Times. Changes to the census could make small towns disappear, february 2020.

[235] Theresa Stadler, Bristena Oprisanu, and Carmela Troncoso. Synthetic data–anonymisation groundhog day. *arXiv preprint arXiv:2011.07018*, 2021.

[236] Jim Miller. Coordinated disclosure of vulnerabilities affecting girault, bulletproofs, and plonk, 2022.

[237] Josh Swihart, Benjamin Winston, and Sean Bowe. Zcash counterfeiting vulnerability successfully remediated, 2019.

[238] Daniel J Bernstein, Tanja Lange, and Ruben Niederhagen. Dual ec: A standardized back door. In *The New Codebreakers*, pages 256–281. Springer, 2016.

[239] BBC. A-levels and GCSEs: U-turn as teacher estimates to be used for exam results, August 2020.

[240] Zakir Durumeric, James Kasten, Michael Bailey, and J Alex Halderman. Analysis of the https certificate ecosystem. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 291–304, 2013.

[241] Jeremy Clark and Paul C Van Oorschot. Sok: Ssl and https: Revisiting past challenges and evaluating certificate trust model enhancements. In *2013 IEEE Symposium on Security and Privacy*, pages 511–525. IEEE, 2013.

[242] Nicole Van der Meulen. Diginotar: Dissecting the first dutch digital disaster. *Journal of Strategic Security*, 6(2):46–58, 2013.

[243] Microsoft. Fraudulent digital certificates could allow spoofing, August 2011.

[244] Heather Adkins. An update on attempted man-in-the-middle attacks, August 2011.

[245] Johnathan Nightingale. Fraudulent *.google.com certificate, August 2011.

[246] R Stradling. Certificate transparency version 2.0 draft-ietf-trans-rfc6962-bis-39. 2021.

[247] Benjamin Dowling, Felix Günther, Udyani Herath, and Douglas Stebila. Secure logging schemes and certificate transparency. In *European Symposium on Research in Computer Security*, pages 140–158. Springer, 2016.

[248] E. Stark, R. Sleevi, R. Muminovic, D. O'Brien, E. Messeri, A. P. Felt, B. McMillion, and P. Tabriz. Does certificate transparency break the web? measuring adoption and error rate. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 211–226, 2019.

[249] Josh Aas, Richard Barnes, Benton Case, Zakir Durumeric, Peter Eckersley, Alan Flores-López, J Alex Halderman, Jacob Hoffman-Andrews, James Kasten, Eric Rescorla, et al. Let's encrypt: an automated certificate authority to encrypt the entire web. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2473–2487, 2019.

[250] Emily Stark, Joe DeBlasio, and Devon O'Brien. Certificate transparency in google chrome: Past, present, and future. *IEEE Security & Privacy*, 19(6):112–118, 2021.

[251] Stijn Pletinckx, Thanh-Dat Nguyen, Tobias Fiebig, Christopher Kruegel, and Giovanni Vigna. Certifiably vulnerable: Using certificate transparency logs for target reconnaissance. In *2023 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2023.

[252] Quirin Scheitle, Oliver Gasser, Theodor Nolte, Johanna Amann, Lexi Brent, Georg Carle, Ralph Holz, Thomas C Schmidt, and Matthias Wählisch. The rise of certificate transparency and its implications on the internet ecosystem. In *Proceedings of the Internet Measurement Conference 2018*, pages 343–349, 2018.

[253] Richard Roberts and Dave Levin. When certificate transparency is too transparent: Analyzing information leakage in https domain names. In *Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society*, pages 87–92, 2019.

[254] Bingyu Li, Jingqiang Lin, Fengjun Li, Qiongxiao Wang, Qi Li, Jiwu Jing, and Congli Wang. Certificate transparency in the wild: Exploring the reliability of monitors. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2505–2520, 2019.

[255] Laurent Chuat, Pawel Szalachowski, Adrian Perrig, Ben Laurie, and Eran Messeri. Efficient gossip protocols for verifying the consistency of certificate logs. In *2015 IEEE Conference on Communications and Network Security (CNS)*, pages 415–423. IEEE, 2015.

[256] Oliver Gasser, Benjamin Hof, Max Helm, Maciej Korczynski, Ralph Holz, and Georg Carle. In log we trust: revealing poor security practices with certificate transparency logs and internet measurements. In *International*

*Conference on Passive and Active Network Measurement*, pages 173–185. Springer, 2018.

[257] Sarah Meiklejohn, Pavel Kalinnikov, Cindy S Lin, Martin Hutchinson, Gary Belvin, Mariana Raykova, and Al Cutter. Think global, act local: Gossip and client audits in verifiable data structures. *arXiv preprint arXiv:2011.04551*, 2020.

[258] Ethereum. Proof-of-stake rewards and penalties, 2022.

[259] Aggelos Kiayias, Andrew Miller, and Dionysis Zindros. Non-interactive proofs of proof-of-work. In *International Conference on Financial Cryptography and Data Security*, pages 505–522. Springer, 2020.

[260] Joseph Bonneau, Izaak Meckler, Vanishree Rao, and Evan Shapiro. Mina: Decentralized cryptocurrency at scale. *New York Univ. O (1) Labs, New York, NY, USA, Whitepaper*, pages 1–47, 2020.

[261] Mustafa Al-Bassam, Alberto Sonnino, Vitalik Buterin, and Ismail Khoffi. Fraud and data availability proofs: Detecting invalid blocks in light clients. In *International Conference on Financial Cryptography and Data Security*, pages 279–298. Springer, 2021.

[262] Mingchao Yu, Saeid Sahraei, Songze Li, Salman Avestimehr, Sreeram Kannan, and Pramod Viswanath. Coded merkle tree: Solving data availability attacks in blockchains. In *International Conference on Financial Cryptography and Data Security*, pages 114–134. Springer, 2020.

[263] Ethereum. Optimistic rollups, 2023.

[264] Ethereum. Zero-knowledge rollups, 2023.

[265] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M Voelker, and Stefan Savage. A fistful of bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 127–140, 2013.

[266] Ross Anderson, Ilia Shumailov, Mansoor Ahmed, and Alessandro Rietmann. Bitcoin redux. 2019.

[267] Mansoor Ahmed, Ilia Shumailov, and Ross Anderson. Tendrils of crime: Visualizing the diffusion of stolen bitcoins. In *International Workshop on Graphical Models for Security*, pages 1–12. Springer, 2018.

[268] Ghada Almashaqbeh and Ravital Solomon. Sok: Privacy-preserving computing in the blockchain era. *Cryptology ePrint Archive*, 2021.

[269] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE, 2014.

[270] Kurt M Alonso et al. Zero to monero, 2020.

[271] Alexey Pertsev, Roman Semenov, and Roman Storm. Tornado cash privacy solution version 1.4. 2019.

[272] U.S. Treasury sanctions notorious virtual currency mixer Tornado Cash, 2022.

[273] Claudia Diaz, Harry Halpin, and Aggelos Kiayias. The nym network, 2021.

[274] George Kappos, Haaroon Yousaf, Mary Maller, and Sarah Meiklejohn. An empirical analysis of anonymity in zcash. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 463–477, 2018.

[275] Alex Biryukov, Daniel Feher, and Giuseppe Vitto. Privacy aspects and subliminal channels in zcash. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1813–1830, 2019.

[276] Alex Biryukov and Daniel Feher. Privacy and linkability of mining in zcash. In *2019 IEEE Conference on Communications and Network Security (CNS)*, pages 118–123. IEEE, 2019.

[277] Malte Möser, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava, Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, and Nicolas Christin. An empirical analysis of traceability in the monero blockchain. *Proceedings on Privacy Enhancing Technologies*, 2018(3):143–163, 2018.

[278] Younggee Hong, Hyunsoo Kwon, Jihwan Lee, and Junbeom Hur. A practical de-mixing algorithm for bitcoin mixing services. In *Proceedings of the 2nd ACM Workshop on Blockchains, Cryptocurrencies, and Contracts*, pages 15–20, 2018.

[279] Mike Wu, Will McTighe, Kaili Wang, Istvan A Seres, Nick Bax, Manuel Puebla, Mariano Mendez, Federico Carrone, Tomás De Mattey, Herman O Demaestri, et al. Tutela: An open-source tool for assessing user-privacy on ethereum and tornado cash. *arXiv preprint arXiv:2201.06811*, 2022.

[280] Ameen Soleimani. Privacy pools with opt-in or opt-out anonymity sets, 2022.

[281] Patrik Keller, Martin Florian, and Rainer Böhme. Collaborative deanonymization. In *International Conference on Financial Cryptography and Data Security*, pages 39–46. Springer, 2021.

[282] Shayan Eskandari, Seyedehmahsa Moosavi, and Jeremy Clark. Sok: Transparent dishonesty: front-running attacks on blockchain. In *International Conference on Financial Cryptography and Data Security*, pages 170–189. Springer, 2019.

[283] Kirill Nikitin, Eleftherios Kokoris-Kogias, Philipp Jovanovic, Nicolas Gailly, Linus Gasser, Ismail Khoffi, Justin Cappos, and Bryan Ford. {CHAINIAC}: Proactive software-update transparency via collectively signed skipchains and verified builds. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pages 1271–1287, 2017.

[284] Mustafa Al-Bassam and Sarah Meiklejohn. Contour: A practical system for binary transparency. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 94–110. Springer, 2018.

[285] Mark D Flood, Jonathan Katz, Stephen J Ong, and Adam Smith. Cryptography and the economics of supervisory information: Balancing transparency and confidentiality. 2013.

[286] Oshani Seneviratne and Lalana Kagal. Enabling privacy through transparency. In *2014 Twelfth Annual International Conference on Privacy, Security and Trust*, pages 121–128. IEEE, 2014.

[287] Daniel J Weitzner, Harold Abelson, Tim Berners-Lee, Chris Hanson, James Hendler, Lalana Kagal, Deborah L McGuinness, Gerald Jay Sussman, and K Krasnow Waterman. Transparent accountable data mining: New strategies for privacy protection. 2006.

[288] Daniel Weitzner, Hal Abelson, Tim Berners-Lee, Christ Hanson, Jim Hendler, Lalana Kagal, D McGuinness, Gerry Sussman, and K Krasnow Waterman. Transparent accountable inferencing for privacy risk management. In *AAAI Spring Symposium on The Semantic Web meets eGovernment*. AAAI Press, Stanford University, 2006.

[289] Giridhari Venkatadri, Alan Mislove, and Krishna P Gummadi. Treads: transparency-enhancing ads. In *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*, pages 169–175, 2018.

[290] Nigel Shadbolt, Kieron O'Hara, Tim Berners-Lee, Nicholas Gibbins, Hugh Glaser, Wendy Hall, et al. Linked open government data: Lessons from data. gov. uk. *IEEE Intelligent Systems*, 27(3):16–24, 2012.

[291] Kieron O'Hara. Transparent government, not transparent citizens: a report on privacy and transparency for the cabinet office. 2011.

[292] Home Office. Operational case for the use of communications data by public authorities. `https://www.gov.uk/government/publications/investigatory-powers-bill-overarching-documents`, 2016.

[293] Interception of Communications Commissioner's Office. Report of the interception of communications commissioner - annual report for 2016, December 2017.

[294] ETSI. TS 103 307: Security aspects for LI and RD interfaces, 2018. V1.3.1.

[295] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. Weed Cocco, and J. Yellick. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. *ArXiv e-prints*, January 2018.

[296] Christian Cachin. Architecture of the Hyperledger blockchain fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*, 2016.

[297] Marko Vukolić. Rethinking permissioned blockchains. In *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, pages 3–7. ACM, 2017.

[298] Ben Laurie, Adam Langley, and Emilia Kasper. Rfc 6962 – Certificate transparency. Technical report, 2013.

[299] Özgür Dagdelen. *The cryptographic security of the German electronic identity card*. PhD thesis, Technische Universität Darmstadt, 2013.

[300] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[301] Investigatory Powers Commitioner's Office. Annual report of the investigatory powers commissioner 2017, January 2019.

[302] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[303] J. Heaton. Comparing dataset characteristics that favor the apriori, eclat or fp-growth frequent itemset mining algorithms. In *SoutheastCon 2016*, pages 1–7, March 2016.

[304] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. *Proc. of the 20th VLDB Conference*, pages 487–499, 1994.

[305] A Dekhtyar and J Verburg. Extended bakery dataset. `https://wiki.csc.calpoly.edu/datasets/wiki/ExtendedBakery`, 2009.

[306] Frederic Flouvat, F De March, and Jean-Marc Petit. A thorough experimental study of datasets for frequent itemsets. In *Data Mining, Fifth IEEE International Conference on*. IEEE, 2005.

[307] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the sulq framework. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 128–138. ACM, 2005.

[308] Pietro Hiram Guzzi, Marianna Milano, and Mario Cannataro. Mining association rules from gene ontology and protein networks: Promises and challenges. *Procedia Computer Science*, 29:1970–1980, 2014.

[309] Anurag Nagar, Michael Hahsler, and Hisham Al-Mubaid. Association rule mining of gene ontology annotation terms for sgd. In *Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), 2015 IEEE Conference on*, pages 1–7. IEEE, 2015.

[310] Daniel Faria, Andreas Schlicker, Catia Pesquita, Hugo Bastos, António EN Ferreira, Mario Albrecht, and André O Falcão. Mining go annotations for improving annotation consistency. *PloS one*, 7(7):e40519, 2012.

[311] Anand Kumar, Barry Smith, and Christian Borgelt. Dependence relationships between gene ontology terms based on tigr gene product annotations. In *Proceedings of CompuTerm 2004: 3rd International Workshop on Computational Terminology*, 2004.

[312] So Hyun Park, Shin Yi Jang, Ho Kim, and Seung Wook Lee. An association rule mining-based framework for understanding lifestyle risk behaviors. *PloS one*, 9(2):e88859, 2014.

[313] Peter B Jensen, Lars J Jensen, and Søren Brunak. Mining electronic health records: towards better research applications and clinical care. *Nature Reviews Genetics*, 13(6):395, 2012.

[314] Giulia Toti, Ricardo Vilalta, Peggy Lindner, Barry Lefer, Charles Macias, and Daniel Price. Analysis of correlation between pediatric asthma exacerbation and exposure to pollutant mixtures with association rule mining. *Artificial intelligence in medicine*, 74:44–52, 2016.

[315] Mustafa Suleyman and Ben Laurie. Trust, confidence and verifiable data audit, 2017.

[316] Mireille Hildebrandt. Legal and technological normativity: more (and less) than twin sisters. *Techné: Research in Philosophy and Technology*, 12(3):169–183, 2008.

[317] Ashley Nellis. The color of justice: Racial and ethnic disparity in state prisons. 2021.

[318] Kim Zetter. Diginotar files for bankruptcy in wake of devastating hack, 2011.

[319] Tim Wu. When code isn't law. *Va. L. Rev.*, 89:679, 2003.

[320] Ross Anderson. *Security engineering: a guide to building dependable distributed systems*. John Wiley & Sons, 2020.

[321] Lawrence Lessig. Law regulating code regulating law. *Loy. U. Chi. LJ*, 35:1, 2003.

[322] Joan Feigenbaum and Daniel J Weitzner. On the incommensurability of laws and technical mechanisms: Or, what cryptography can't do. In *Cambridge International Workshop on Security Protocols*, pages 266–279. Springer, 2018.

[323] Joel R Reidenberg. Lex informatica: The formulation of information policy rules through technology. *Tex. L. Rev.*, 76:553, 1997.

[324] Lawrence Lessig. Code is law. *Harvard magazine*, 1(2000), 2000.

[325] Nick Szabo. Formalizing and securing relationships on public networks. *First monday*, 1997.

[326] Max Raskin. The law and legality of smart contracts. 2016.

[327] Laurent Simon, David Chisnall, and Ross Anderson. What you get is what you c: Controlling side effects in mainstream c compilers. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 1–15. IEEE, 2018.

[328] Laurence Diver. Digisprudence: the design of legitimate code. *Law, Innovation and Technology*, pages 1–30, 2021.

[329] Bogdan Kulynych, Rebekah Overdorf, Carmela Troncoso, and Seda Gürses. Pots: protective optimization technologies. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 177–188, 2020.

[330] Seda Gürses, Rebekah Overdorf, and Ero Balsa. Stirring the pots: protective optimization technologies. In *Being profiled*, pages 24–29. Amsterdam University Press, 2018.

[331] Ivan Evtimov, David O'Hair, Earlence Fernandes, Ryan Calo, and Tadayoshi Kobno. Is tricking a robot hacking? *Berkeley Tech. LJ*, 34:891, 2019.

[332] Ian Goldberg, David Wagner, and Eric Brewer. Privacy-enhancing technologies for the internet. In *Proceedings IEEE COMPCON 97. Digest of Papers*, pages 103–109. IEEE, 1997.

[333] Seda Gürses, Carmela Troncoso, and Claudia Diaz. Engineering privacy by design. *Computers, Privacy & Data Protection*, 14(3):25, 2011.

[334] Seda Gürses, Carmela Troncoso, and Claudia Diaz. Engineering privacy by design reloaded. In *Amsterdam Privacy Conference*, pages 1–21, 2015.

[335] Shoshana Zuboff. Big other: surveillance capitalism and the prospects of an information civilization. *Journal of information technology*, 30(1):75–89, 2015.

[336] Jeff John Roberts. Hoax over 'dead' ethereum founder spurs $4 billion wipe out, 2017.

[337] Sarah Azouvi, Mary Maller, and Sarah Meiklejohn. Egalitarian society or benevolent dictatorship: The state of cryptocurrency governance. In *International Conference on Financial Cryptography and Data Security*, pages 127–143. Springer, 2018.

[338] Peter H Hochschild, Paul Turner, Jeffrey C Mogul, Rama Govindaraju, Parthasarathy Ranganathan, David E Culler, and Amin Vahdat. Cores that don't count. In *Proceedings of the Workshop on Hot Topics in Operating Systems*, pages 9–16, 2021.

[339] Emma Arfelt, David Basin, and Søren Debois. Monitoring the gdpr. In *European Symposium on Research in Computer Security*, pages 681–699. Springer, 2019.

[340] Ben Green. The flaws of policies requiring human oversight of government algorithms, 2021.

[341] Russell Hotten. Volkswagen: The scandal explained, 2015.

[342] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 238–252. IEEE, 2013.

[343] Pauline T Kim. Auditing algorithms for discrimination. *U. Pa. L. Rev. Online*, 166:189, 2017.

[344] Anupam Datta, Deepak Garg, Dilsun Kaynar, Divya Sharma, and Arunesh Sinha. Program actions as actual causes: A building block for accountability. In *2015 IEEE 28th Computer Security Foundations Symposium*, pages 261–275. IEEE, 2015.

[345] Anupam Datta, Shayak Sen, and Yair Zick. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *2016 IEEE symposium on security and privacy (SP)*, pages 598–617. IEEE, 2016.

[346] Nick Wallis. *The Great Post Office Scandal*. Bath Publishing Ltd, 2021.

[347] Tim McCormack. The post office horizon system and seema misra. *Digital Evidence & Elec. Signature L. Rev.*, 13:133, 2016.

[348] Jasper Jolly. Uk government sets aside up to £233m to cover post office payouts, 2021.

[349] Ekker. Dutch court rules on data transparency for uber and ola drivers, 2021.

[350] The App Drivers and Couriers Union. Uber under pressure over facial recognition checks for drivers; call for suspension of checks.

[351] Inioluwa Deborah Raji, Andrew Smart, Rebecca N White, Margaret Mitchell, Timnit Gebru, Ben Hutchinson, Jamila Smith-Loud, Daniel Theron, and Parker Barnes. Closing the ai accountability gap: Defining an end-to-end framework for internal algorithmic auditing. In *Proceedings*

*of the 2020 conference on fairness, accountability, and transparency*, pages 33–44, 2020.

[352] Peter Bernard Ladkin, Bev Littlewood, Harold Thimbleby, and Martyn Thomas. The law commission presumption concerning the dependability of computer evidence. *Digital Evidence & Elec. Signature L. Rev.*, 17:1, 2020.

[353] Peter Bernard Ladkin. Robustness of software. *Digital Evidence & Elec. Signature L. Rev.*, 17:15, 2020.

[354] Nicholas Bohm, James Christie, Peter Bernard Ladkin, Bev Littlewood, Paul Marshall, Stephen Mason, Martin Newby, Steven Murdoch, Harold Thimbleby, and Martyn Thomas. Briefing note: The legal rule that computers are presumed to be operating correctly–unforeseen and unjust consequences. *Digital Evidence and Electronic Signature Law Review*, 19:123–127, 2022.

[355] Todd Feathers. Police are telling ShotSpotter to alter evidence from gunshot-detecting AI, July 2021.

[356] Umang Bhatt, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, Ruchir Puri, José MF Moura, and Peter Eckersley. Explainable machine learning in deployment. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 648–657, 2020.

[357] Jakko Kemper and Daan Kolkman. Transparent to whom? no algorithmic accountability without a critical audience. *Information, Communication & Society*, 22(14):2081–2096, 2019.

[358] Heike Felzmann, Eduard Fosch Villaronga, Christoph Lutz, and Aurelia Tamò-Larrieux. Transparency you can trust: Transparency requirements for artificial intelligence between legal norms and contextual concerns. *Big Data & Society*, 6(1):2053951719860542, 2019.

[359] General data protection regulation, 2016.

[360] Zakir Durumeric, Frank Li, James Kasten, Johanna Amann, Jethro Beekman, Mathias Payer, Nicolas Weaver, David Adrian, Vern Paxson, Michael Bailey, et al. The matter of heartbleed. In *Proceedings of the 2014 conference on internet measurement conference*, pages 475–488, 2014.

[361] Elissa M. Redmiles, Mia M. Bennett, and Tadayoshi Kohno. Power in computer security and privacy: A critical lens. *IEEE Security & Privacy*, 21(2):48–52, 2023.