JOURNAL OF CYBERSECURITY

Journal of Cybersecurity, 0(0), 2017, 1–14
doi: 10.1093/cybsec/tyx010
Research paper

Research paper

Surveillance and identity: conceptual framework and formal models

Victoria Wang[1,*] and John V. Tucker[2]

[1]Senior Lecturer on Security and Cybercrime, Institute of Criminal Justice Studies, Faculty of Humanities and Social Sciences, University of Portsmouth, Portsmouth, UK; [2]Professor of Computer Science, Department of Computer Science, College of Science, Swansea University, Swansea, UK

*Corresponding author: E-mail: victoria.wang@port.ac.uk

Received 23 June 2016; accepted 26 October 2017

Abstract

Surveillance is recognized as a social phenomenon that is commonplace, employed by governments, companies and communities for a wide variety of reasons. Surveillance is fundamental in cybersecurity as it provides tools for prevention and detection; it is also a source of controversies related to privacy and freedom. Building on general studies of surveillance, we identify and analyse certain concepts that are central to surveillance. To do this we employ formal methods based on elementary algebra. First, we show that disparate forms of surveillance have a common structure and can be unified by abstract mathematical concepts. The model shows that (i) finding identities and (ii) sorting identities into categories are fundamental in conceptualizing surveillance. Secondly, we develop a formal model that theorizes identity as abstract data that we call *identifiers*. The model views identity through the computational lens of the theory of abstract data types. We examine the ways identifiers depend upon each other; and show that the provenance of identifiers depends upon translations between systems of identifiers.

Key words: surveillance, social sorting, identity, abstract data types, formal methods

Introduction

Surveillance is an integral part of everyday life as many technologies employed in our physical and virtual environments have long been capable of monitoring and recording our activities cf. [1]. The ubiquitous cameras that monitor our physical environment, in order to improve the safety and security of people and property, are but the most visible tip of the surveillance iceberg. The invisible bulk is made of software that record data about actions and events cf. [2]. Our professional lives have long been conducted through software systems, and recently, our personal lives have become dependent on software systems through social media. Our home and neighbourhood environments are next to succumb to software, through the internet of things, e.g. [3, 4, 5]. That our lives are being captured and represented by digital data, collected by many independent sources for different purposes, is an important sociological phenomenon. The translation of all kinds of data into digital form, and the aggregation and unification of all kinds of data sources through computer networks are important technological phenomena.

Surveillance is enormously controversial as it impacts on the multitude of notions that make up privacy and freedom for individuals; on the conduct of economic and social life of societies; and on the legal, political, and military foundations of the state [6, 7]. With this broad view, David Lyon has given a general description of surveillance as 'the focused, systematic and routine attention to personal details for purposes of influence, management, protection or detection' [8, 14]. In establishing surveillance as a general social issue, Lyon has proposed that surveillance has three main purposes [8]:

i. *keeping control*, which is the historic purpose pursued by employers, police, and government;
ii. *social sorting*, pursued by companies in marketing and managing customers; and

iii. *mutual monitoring*, pursued in peer to peer in social networks, real and virtual.

Thanks to the ubiquity of digital technologies, the aims and methods of social sorting — the categorization of personal data — is becoming most prominent.

In this article, we examine theoretically the general ideas of surveillance and one of its component concepts that of identity. Identity is fundamental to contemporary surveillance practices [9, 10, 11]. Surveillance technologies rely on identity management systems to provide information, which vary in accuracy.[1] For instance, for social sorting to work, identity needs to be just precise enough to enable categorizations to be useful in an application.

We seek completely abstract models that can be formalized and analysed mathematically. First, we develop a general definition of surveillance that captures the notion in diverse situations, and we illustrate the general definition with some disparate examples. This definition shows that the three main types of surveillance have the same structures, and that the essence of surveillance is indeed *sorting* and *categorization*. Our analysis applies to *entities* that are objects or people, real or virtual, belonging to a specific *context*.

A most important component idea of our definition of surveillance is that of the identity of the people or objects observed. We introduce the general concept of *identifiers*, which are data designed to recognize an entity. Here is our idea:

> **Informal Definition.** An *identifier* for an entity is data that is associated with the entity for the purposes of distinguishing it among other entities in some context and for some purpose.

Identifiers are the main focus of our paper. As a starting point for our conceptual analysis, we assume that:

> **Principle.** *Entities are recognised* ***only*** *through the data that act as their identifiers for a context. Entities are observed* ***only*** *through the data that represent their behaviour in a context.*

This hypothesis is widely applicable. First, the surveillance context is determined by selecting aspects of an entity's behaviour that *can* be captured in data, and by observations made by testing for *attributes* of the data. Secondly, the identity of an object is reduced to measurements, and the identity of a person is reduced to forms of evidence that are also data, including records of personal testimony and formal registration, as well as biometrics [15, 16, 17]. The idea is simpler and more palatable when one considers the virtual world, which creates hugely many more contexts that are, and can only be, made of data. Users have many identities, some of which they create in a state of anonymity. The Principle is perfectly at home in the virtual world of cybersecurity.

Technically, the operations and tests on identifiers combine to make systems of identifiers. Although designed for specific contexts, they often have unforeseen applications. Since identifiers are data, clearly the systems of identifiers are actually examples of what computer scientists call *abstract data types* [18, 19]. The theory of abstract data types characterizes data through its operations and tests, which may be specified by axioms to make them close to the application domain and independent of implementations. The theory uses algebra to model *any* form of data, and tools to design and build software. The idea of a general theory of identity based on abstract data types is new.

Foremost among identifiers are those that are supposed to identify people. The notion of a *personal identifier* proves to be as informative as it is subtle. To understand identity we need to examine the ways identifiers are issued and how they depend upon other identifiers. We show that the provenance of identifiers is an essential idea. We consider principles of how identifiers are to be compared and when they might be deemed equivalent; this requires notions of translations between different systems of identifiers.

All of these concepts are motivated by some informally described examples, and then formalized mathematically using elementary algebra. The examples of surveillance and identity we use refer to situations both in everyday life and in cybersecurity. The everyday examples make the point our concepts apply to traditional forms of identity and security. The cybersecurity examples give a glimpse of the abundance of identity issues in securing software systems. Identity is a central concept in hashing, encryption, communication protocols, certification, and their roles in the maintaining the trustworthiness of transactions, encoding of access controls, tracing events, forensics, etc. In their mathematical form, the concepts create precise and general definitions that cover a great range of examples.

The article is in two parts: on surveillance and on identity. The Section 'What is surveillance?' exemplifies the central ideas informally; their formalization begins in the Section 'A formal model of surveillance'.

But why formalize? Formalizations make notions precise. They uncover and classify the possible structures and interpretations of ideas. Our formalization can be likened to the way formal logic has long been used in philosophy to clarify the nature of truth, arguments and reasoning. Later, we reflect on the role of abstract concepts and formal methods to give insights in sociological contexts (the sub-Section 'Formalizing identity and social theory').

Our aim is to discover general concepts and principles associated with surveillance and to analyse them, mapping their interconnections and implications by means of mathematical ideas and structures. Formalization has proved to be a fundamental *practical* tool in the development of software. Formal models are needed by technologists designing surveillance systems and their safeguards, in order to develop tools for testing and reasoning. Such formal methods play a role in software engineering where security by design is an objective. We believe that raising the status of identity and modelling identity using abstract data types will be useful in making and maintaining cybersecurity software.

The mathematical pre-requisites are modest.[2] To address a readership from different disciplines and professions, we explain some very basic mathematical ideas. While this will seem laborious and unnecessary to those familiar with formal methods, others may benefit when ideas are shown to arise naturally in thinking about identity, and appreciate more readily the benefits that formalization delivers.

What is surveillance?

Let us begin with an abstract informal description of a large class of surveillance systems.

Informal Definition. A *surveillance system* observes the behaviour of people and objects in a context that may be real or virtual.

1 For example, in the UK, accurate identification of an individual usually depends on a passport [11], a driver's licence (DVLA) and, for some, the National Identity Register [12]. Accuracy increases if identification involves fingerprints [13], iris scans [14] and DNA [15]; see also [16].

2 The mathematical ideas we use use to launch our models are sets, functions and relations, which are described in any number of textbooks on discrete mathematics, e.g. [20, 21]. The theory of abstract data types is more demanding algebraically, e.g. [18].

The surveillance system classifies behaviours by means of attributes, and identifies people and objects with those attributes. A surveillance system consists of the following components and methods:

1. ***Entity***. Entities that possess behaviour in space and time.
2. ***Observable behaviour***. Methods for obtaining and recording data about behaviours.
3. ***Attribute***. Methods for defining and recognizing attributes of behaviour data.
4. ***Identity***. Methods for generating data that identifies entities in the context.

In practice, what is observable about the behaviour are the attributes of data; these characterize the context for the surveillance. Depending upon the context, we may expect the attributes to be based upon laws, rules, norms, conventions, policies, practices, expectations, etc. Indeed, when the purpose of surveillance is control, they may seek to catch deviations. The definition is neutral and does *not* imply deviance. The definition does require precise formulations of attributes for a process of categorization. The data that is used to identify entities can be numbers, texts, sounds and images. Here are three simple examples to prepare for our abstract formalizations.

Example 1: Control Motor — Vehicles. Automatic Number Plate Recognition (ANPR) is a technology that observes vehicles and records registration marks. Typical applications are checking on vehicle speed, managing car parking and collecting tolls, cf. [22]. The technology was functioning in the late 1970s. Today, ANPR is a component of hundreds of thousands of surveillance systems owned by both public and private organizations.

Consider some ANPR applications in terms of our abstract definition. In such surveillance systems, the entities are vehicles at a particular location and time. The vehicles may be in motion (as with speed checks), or may be arriving or leaving a location (as with car parking and congestion zones in cities). The vehicles are observed by cameras that create images and the software that processes the images varies according to the behavioural attributes under observation (e.g. breaking an average speed limit over a stretch of road, or overstaying a parking time limit). In particular, optical character recognition establishes the registration mark of vehicles. A registration mark is an alpha-numeric name that identifies a vehicle uniquely in a human-centred way in a *national* register of vehicles. The mark links to information about the characteristics of the vehicle. Thus, to the surveillance system, the identity of an entity *is* this registration mark. For example, a surveillance system for car parking based on an ANPR has the form:

Entity: Cars

Observable Behaviour: The registration mark, its time of arrival and departure at the location

Attributes: Duration of stay above a particular limit

Identity: Registration marks

Following the ANPR stages described above, the registration mark is communicated to a database relevant to the application. For example, the database may be used to check an attribute, such as a payment (tax, charge or toll), having been made for that registration mark.

The surveillance system knows the identity of the car, but not necessarily the driver.

Suppose we take the entities to be people. To find the driver, an independent process involving *only* personal identities begins. The vehicle is registered to a person called the *keeper* of the vehicle, who must be located and contacted. In the UK, the operator of the surveillance system communicates the registration mark to the Driver and Vehicle Licensing Agency (DVLA), or to one of its approved agents, to determine the name and address of the keeper. The output of these actions is the identity of the keeper. Indeed, in this necessary second stage, there is a transformation of identity data from the registration mark to the name and address of the keeper. Note that finding the actual driver may require further independent action. In the case of speeding, where laws are involved, the driver's record will contain characteristics such as a driving penalty history.

Example 2: Social Sorting — Customer Accounts. Consider a client's e-account with some provider, such as a bank or shop. Typically, an account has the following components: an account number that identifies the account; a user name and password as a form of identity used to gain access to the account; a set of characteristics of the account, such as personal details and the scope and limits of services; and an account history that records past transactions and allows for new transactions, queries, preferences, etc. The account history is the behaviour of the account; it is observed to check that terms and conditions are met by the client, or that no unusual pattern of transactions has been carried out, or to generate suggestions for new products and services. Observations might also include standard monitoring data about user logins and login attempts, duration, location, etc. For example:

Entities: Credit card accounts

Observable Behaviour: Transactions: date, payee, location, sum, etc.

Attributes: Credit limit, minimum payments, unusual transactions

Identity: Credit card number

Example 3: Mutual Monitoring — Social Media Accounts. Social media connect people who have personal or professional interests in common. Systems such as Facebook, Twitter, WeChat, Instagram, LinkedIn, and Academia.edu attract large numbers of users. Individuals register with a system and create an account and a network of other users to suit their needs. Abstractly, an account has a structure similar to that of a customer account for a bank or shop. The behaviour of the account is a history of postings, status updates, linkages and interactions. In social networking, individuals voluntarily reveal very detailed information about themselves to their networks, including their personal history, tastes, opinions and activities; the behaviours could be termed *personas*. From the point of view of surveillance, two phenomena are of interest: (i) individuals can and do watch over people in their networks, and (ii) the data of the account holders belong to companies that can collect and use the information for commercial or other purposes. Illustrating the components:

Entities: Member accounts

Observable Behaviour: Personal declarations, posts, comments, connections, location, etc.

Attributes: Targeted opinions in posts on specific topics, interactions with other members, unusual interactions

Identity: Usernames

A formal model of surveillance

We have defined surveillance informally as a process that identifies entities on detecting certain properties of their behaviour. We will define this process formally.

Context: entities and their behaviour

Let *Entity* be a set of entities whose behaviour is to be observed. Let *Behaviour* be the set of all possible behaviours in space and time of all the entities of *Entity*. The nature of behaviour and its models we consider in stages.

Deterministic Behaviour. Suppose that each entity $e \in Entity$ has one and only one behaviour in space and time, i.e. its behaviour is deterministic. In this case, there is a single-valued mapping

$$[[\_]]: Entity \to Behaviour$$

such that

$[[e]]$ = the behaviour of the entity $e \in Entity$.

The mapping provides a formal model or semantics for the behaviour of the entity. Taken together we have formalized a *context* for the surveillance as an algebraic structure:

$$Context = (Entity, Behaviour \mid [[\_]]: Entity \to Behaviour).$$

Non-deterministic Behaviour. Suppose that each entity has more than one possible behaviour in space and time, i.e. its behaviour is non-deterministic. In this case, there is a relation

$$[[\_]]: Entity \times Behaviour$$

such that

$[[e, b]] \iff b \in Behaviour$ is a possible behaviour of the entity $e \in Entity$.

The *context* for the surveillance is a relational structure

$$Context = (Entity, Behaviour \mid [[\_]]: Entity \times Behaviour).$$

Alternately, in the non-deterministic case, if the elements of *Behaviour* are sets of possible behaviours of an entity, the relation can be replaced with a map returning sets.

We will focus on the deterministic case. The behaviours need to be modelled formally. How might this be done? There are several options.

Behaviour as streams of data

A way to formalize behaviours is to think of entities performing a sequence of actions or events taking place in time.

Let *Time* be a set of time points generated by a clock of some kind; for example, say $Time = \{0, 1, 2, \ldots, t, \ldots\}$. Let *Action* be a set of actions or events characteristic of the entities. The behaviour of an entity is conceived of as a stream

$$a(0), a(1), a(2), \ldots, a(t), \ldots$$

of actions or events in time, where $a(t) \in Action$ for all $t \in Time$. Such sequences will be termed *traces*:

Definition. A *trace* is an association of actions or events to time points and is formalized by a total mapping

$$a: Time \to Action$$

such that for all $t \in Time$

$a(t)$ = the action or event taking place at time $t \in Time$.

Let *Trace* be the set of all possible traces.

Now in many cases, the space *Behaviour* of all possible behaviours of the entities can be taken to be a subset of the set *Trace* of all possible traces; thus,

$$Behaviour \subseteq Trace.$$

When applying the behaviour mapping $[[\_]]$ to an entity $e \in Entity$ we get a trace, which is a map

$$[[e]]: Time \to Action.$$

Therefore, for $e \in Entity$ and $t \in Time$, we have

$[[e]](t)$ = the action or event of entity e taking place at time $t \in Time$.

Example: Twitter. Twitter processes data called *tweets*. At the heart of a tweet is a message made from at most 140 characters, but a tweet is composed of more data. For simplicity, a tweet can be thought of as a vector of data drawn from sets of the following kind:

Text	The text that is the status update.[3]
Identity	A string that uniquely labels the tweet.
Contributor	The author(s) of the tweet.
Time	The time when this Tweet was created.[4]
Location	The geographic location (longitude, latitude) of this Tweet as reported by the user or application.[5]
Retweet	Status and number of retweets.
Favourite	Number of favourites.

We let the set of all possible tweets be

$$Tweet = Text \times Identity \times Contributor \times Time \times Location \times Retweet \times Favourite.$$

Now Twitter generates and processes streams of tweets, i.e. sequences of tweets indexed by time. Thus, the behaviour can be modelled by traces that are streams of tweets of the form

$$a(0), a(1), a(2), \ldots, a(t), \ldots \in Tweet,$$

which is represented by a map a: $Time \to Tweet$. Let *Behaviour* be the set of all possible traces of these kinds. Typical user operations on tweets are 'embedding tweets', 'responding to tweets', and 'favouring', 'unfavouring', and 'deleting tweets', which induce operations on traces.

Depending upon the circumstances, monitoring tweet feeds is called 'curation', 'filtering', or 'surveillance'. Monitoring Twitter can be done in a number of ways via application programming interfaces (APIs), which define instructions for developers to build new systems. Twitter's *Search API* allows users to define criteria (keywords, usernames, locations, named places, etc.) to *search* among existing tweets. Twitter's *Streaming API* redirects a sample of tweets, based upon a user's criteria, as these tweets appear. The sample is less than 1% [23]. Twitter's *Firehose API* delivers 100% of all publicly available tweets that match users' criteria as they are made. The Twitter Firehose is complex and requires a subscription. Twitter's monitoring services have tools to detect non-compliance with Twitter policies (e.g. aggressive following and unfollowing).

Identity: identifying entities

To identify entities in a context whose behaviours have certain properties, the entities need to be labelled, marked or named in some

3 Using the UTF-8 representation for Unicode.

4 Measured by Coordinated Universal Time (UTC).

5 Using the geoJSON standard.

way. Our notion of identifier, defined in the Introduction, is designed to do just this. [6]

Each entity $e \in Entity$ has an identifier that is used to denote the entity in a context. The association of identifiers with entities can be complicated as we will see shortly. In order to formalize surveillance, we must formalize the assignment of identifiers to entities.

Definition. Let *Identifier* be a set of possible identifiers for the entities of *Entity*. There is a relation

$$id \subseteq Identifier \times Entity$$

such that

$$id(i, e) \iff \text{the data } i \in Identifier \text{ is assigned to entity } e \in E.$$

If *id(i, e)* then we say that identifier *i names* entity *e*. Let *anon* be a datum that is *not* in the set *Identifier* of identifiers for the entities; *anon* indicates anonymity, i.e., an entity not named. We will need the set $Identifier \cup \{anon\}$.

We will develop the notion of identifiers in the second part of the article (the Section 'What is identity'? onwards). Here, let us note that since the association of identifiers to entities is a relation, thus many identifiers can be allocated to an entity and, conversely, many entities can have the same identifier. Later, in the Section 'A formal model of identity?', we will simplify the discussion, focussing on the case that the association is a function id: $Identifier \rightarrow Entity$.

Surveillance: detecting attributes

The elements of *Behaviour* formalize the activity of the entities under surveillance. To formalize what it is we are to detect, we suppose that $Prop = Prop_1, \ldots, Prop_k$ is a collection of sets of behaviours, i.e. for $1 \leq i \leq k$,

$$Prop_i \subseteq Behaviour.$$

The entities of interest are those whose behaviours lie in some $Prop_i$; in symbols,

$$Entity(Prop_i) = \{e \in Entity: [[e]] \in Prop_i\}.$$

General Case. Entities in a context are known by their identifiers. Formulations of surveillance can seek for any entity *e* satisfying a $Prop_i$,

i. at least one identifier *i* for *e*;
ii. a subset of the identifiers of *e*; or
iii. all of the identifiers of *e*.

These options have the form of a selection or choice operation

$$select_{id}: Entity \rightarrow P(Identifier)$$

where *P(Identifier)* is the set of all subsets of *Identifier*, and

$$select_{id}(e) \subseteq \{i \in Identifier \mid id(i, e)\}.$$

Definition. Surveillance is formulated as follows: for $1 \leq i \leq k$ define,

$$Surv(Prop_i): Entity \rightarrow P(Identifier)$$

for $e \in Entity$ by

$$Surv(Prop_i)(e) = select_{id}(e) \text{ if } [[e]] \in Prop_i$$
$$= \varnothing \text{ if } [[e]] \notin Prop_i.$$

Note that entities whose behaviours do not lie in $Prop_i$ are mapped to the empty set $\varnothing$, and are ignored and not identified, i.e. they will remain anonymous.

Definition. An entity *e* in a context is *anonymous* under surveillance with attributes *Surv(Prop)* if $Surv(Prop_i)(e) = \varnothing$ for $1 \leq i \leq k$.

Minimal Case. Consider surveillance that seeks just one identifier for any entity whose behaviour satisfies some $Prop_i$. This view of surveillance is reformulated thus:

Definition. Surveillance is defined as follows: for $1 \leq i \leq k$ define,

$$Surv(Prop_i): Entity \rightarrow Identifier \cup \{anon\}$$

for $e \in Entity$ by

$$Surv(Prop_i)(e) = (some\ i)\ id(i, e) \text{ if } [[e]] \in Prop_i$$
$$= anon \text{ if } [[e]] \notin Prop_i.$$

Thus, given the collection $Prop = Prop_1, \ldots, Prop_k$ of properties, surveillance is specified by a collection of functions: for $1 \leq i \leq k$,

$$Surv(Prop_i): Entity \rightarrow Identifier \cup \{anon\}.$$

If convenient, these may be combined as a *k*-tuple,

$$Surv(Prop): Entity \rightarrow (Identifier \cup \{anon\})^k$$

where

$$Surv(Prop)(e) = (Surv(Prop_1)(e), \ldots, Surv(Prop_k)(e)).$$

Combining these ideas, we define formally a very general notion of a surveillance system.

Definition. A *surveillance system* for entities in a context is a structure of the form

$$SurvSys(Prop) = (Entity,\ Identifier \cup \{anon\},\ Behaviour \mid anon,$$
$$id,\ [[\_]],\ Prop_i,\ Surv(Prop_i)\ 1 \leq i \leq k),$$

consisting of the non-empty sets

$$Entity,\ Identifier,\ Behaviour,$$

the constant

$$anon,$$

and the $k + 1$ relations

$$id \subseteq Identifier \times Entity,$$
$$Prop_i,$$

and the $k + 1$ mappings

$$[[\_]]: Entity \rightarrow Behaviour$$
$$Surv(Prop_i): Entity \rightarrow Identifier \cup \{anon\}$$

for $1 \leq i \leq k$.

The definition expresses a minimal general form of a surveillance system as an algebraic structure, which is a semantic model of an abstract data type [18]. The theory of abstract data types was created to model the essential components of any computing system in a precise way. Thus, designers can use algebraic methods when thinking formally about the processes of user specification and subsequent technological implementation [25]. Of course, any actual surveillance system will involve many technologies to obtain and

6 In computing, the term 'identifier' is well established. It is data made of syntax that names or labels a computational entity; commonly, it is an alphanumeric string that defines components in a programming language, such as variables, operators, procedures, programs etc. Our adoption of the word for data associated with a context is essentially a large-scale generalization. The purpose of the notion is close to that of the idea of a pure name in [24]. The term is in use occasionally in some social discussions of identity.

process data. These technologies may suggest some new abstract components that need to be formalized and understood theoretically. Roughly speaking, system design has the following form:

Design Problem. The essence of the design problem is:

1. *Specification*. To define the desired surveillance system by specifying an abstract data type for *SurvSys(Prop)*.
2. *Implementation*. To choose technologies to generate data to
   a. represent the behaviours of the entities;
   b. represent the identities of the entities;
   c. observe behaviours and detect those behaviours having the attributes in *Prop*;
   d. recognize the identity of entities having the properties in *Prop*.

Surveillance and social sorting

In surveillance studies, social sorting is the categorization of people and results in a classification used to treat people differently [26]. Although originally formulated to understand the social impact of surveillance by companies and institutions, our formal definition shows that sorting is essential to the abstract conception of surveillance and, therefore, that sorting is inherent in the surveillance of entities of all kinds. We will formalize the sorting of entities using simple notions of *categorization* and *partition*; however, sorting can be problematic because the sorting of identifiers is more complex than the sorting of entities.

Sorting entities

In our definition of surveillance the collection *Prop* of properties of entities lead to a categorization of entities that can be treated differently. What is a categorization?

Definition. Let *Entity* be a set of entities. A *categorization of entities* is a collection of subsets

$$S_1, S_2, \ldots, S_k \subseteq Entity$$

that include all the entities, i.e.

$$S_1 \cup S_2 \cup \ldots \cup S_k = Entity.$$

An entity *e* lies in at least one of the sets and possibly several. In this loose idea, we may have categories overlapping and having interesting internal structure, e.g. they may be nested and form a hierarchy under the set inclusion ordering. Commonly, and most simply, we may want the sets not to overlap so that an entity *e* lies in one, and only one, of the sets:

Definition. The categorization is a *partition* if for $1 \leq n, m \leq k$, we have $S_n \cap S_m = \varnothing$.

Sorting identifiers

Surveillance observes data about behaviours of entities — not entities — and recognizes only identifiers for entities — not the entities themselves. Thus, surveillance delivers a categorization of identifiers, not a categorization of entities, which makes the notion subtle.

Definition. Let *Identifier* be a set of identifiers. A *categorization of identifiers* is a collection of subsets

$$S_1, S_2, \ldots, S_k \subseteq Identifier$$

that includes all the identifiers, i.e.

$$S_1 \cup S_2 \cup \ldots \cup S_k = Identifier.$$

Again, an identifier *i* lies in at least one of the sets and possibly several. A categorization of identifiers is less likely to be a partition. However, the structure must also be measured against the entities that the identifiers name. Given an entity *e* there can be identifiers *i* and *j* for *e* that lie in *different* sets. This means that the categorization of identifiers does not lead directly to a neat categorization of entities. Distinctions between different identifiers for the same entity may be ambiguities that are meaningful. For example, data integration combines sets of identifiers from different contexts that share the same entities. Categorizations of identifiers arise in many ways, not least by the analysis of data sets using clustering and classification techniques in machine learning [27]. Ideally, our categorization of identifiers can be transformed into one that corresponds with the entities:

Definition. The categorization $S_1, S_2, \ldots, S_k$ of identifiers is *complete* for the entities if for all $i, j \in Identifier$, and any $1 \leq n \leq k$,

$$\text{if } i \in S_n \text{ and } i \text{ and } j \text{ name the same entity then } j \in S_n$$

Our definition of surveillance delivers a categorisation of identifiers, namely:

$$S_n = image(Surv(Prop_n)) - \{anon\}.$$

To make a complete categorization is a process that depends upon knowledge of the equality of identifiers for entities (see the sub-Section 'Generating identifiers'). We now turn to theorizing identity.

What is identity?

Identity has become almost purely a matter of data. People and objects are named, numbered, labelled or otherwise denoted by data relevant to a context. People belong to many contexts: they can be citizens, patients, drivers, voters, employees, customers, crime suspects, etc., each with different identities managed by different kinds of identity management systems. Physical or virtual, each identity system is based on an abstract data type of some kind.

To distinguish between entities in a context, identifiers need not reflect any aspect of the entity or have any meaning at all, however in practice they are loaded with information. Case studies reveal that the following processes are fundamental:

i. creation and re-creation of identifiers;
ii. comparison of identifiers;
iii. inter-dependence of identifiers;
iv. transformation of identifiers;
v. revocation of identifiers.

Identifiers are composite objects: identifiers are commonly built from other identifiers.

Personal identifiers are those that we rely upon to distinguish uniquely a human being. They are guarantees of peoples' identities in contexts that demand physical identity. In the UK, the basic, most rigorous personal identifiers are associated with birth, marriage and death certificates, passports, medical and dental records, driving licenses, National Insurance (NI) records, tax records, etc. Biometric data — such as photographs, fingerprints, iris scans, blood groups, and DNA — are also involved. Biometric data are physical measurements, but they are represented and processed digitally.

In this section, we examine informally some concepts, principles, and examples of identity prior to providing a formal definition and the outline of a theory in the next sections.

Recall from the Introduction that identifiers can be 'any data intended to separate entities in a context'. What is this data? For example, a name for an entity is an identifier. By a name for an entity we commonly mean data made from symbols. In computing systems, there are many syntactic schemes for naming hardware and software entities using alphanumeric strings; usually, the aim is to make a symbolic identifier unique to the entity in a context.

The relationship between entities and identifiers can be complicated. Consider these four *identifier-entity ratios*:

1. ***Many–One Associations***. Each identifier is assigned to one entity, but different identifiers can be assigned to the same entity.
2. ***One–One Associations***. Different identifiers are assigned to different entities.
3. ***One–Many Associations***. An identifier can be assigned to more than one entity but each entity has only one identifier.
4. ***Many–Many Associations***. An identifier is assigned to more than one entity and, vice versa, an entity can be assigned more than one identifier.

Surveillance returns identifiers that can narrow the search for entities but may not pin down the particular entity of interest. Searches take place on identifiers and, as we have noted, that an identifier can easily point to many distinct entities. Thus, many-to-one associations are important because:

> **Search Principle.** *If an association is many-one then to find an entity, we can search for any one of a set of alternate identifiers for that entity. If an association is one-one then there is one and only one identifier for that entity.*

The following point about narrowing the search for identifiers is obvious but certainly is profoundly important practically:

> **Enumeration Principle.** *The addition of a number, reference code, extension tag, time stamp, or hash code may turn a many-one association into a one-one association.*

The use of numbers to uniquely distinguish entities in a context is old and universal, helping to determine uniquely all sorts of entities, such as people (by membership numbers); invoices, orders and payments (by reference numbers); and consumer products (by serial and barcode numbers). Reference codes do the same using alphanumerical strings. The use of extension tags often structures identifiers as paths in a tree and, like time stamps, separate entities, narrow searches, and can isolate entities uniquely. Hashing produces long binary or hex numbers as code for an identifier.

Example 1: Cars. Recall Example 1 in the Section 'What is surveillance?', which illustrates one-one and many-one associations. In the UK, each car is assigned a registration mark; the current system was introduced on 1 September 2001. In general, each registration mark consists of seven characters with a defined format. From left to right, the characters consist of: (i) a local memory tag or area code, consisting of two letters that indicates the local registration office; (ii) a two-digit age identifier, which changes twice a year, in March and September; and (iii) a three-letter sequence which uniquely distinguishes each of the cars displaying the same initial four-character area and age sequence. The association of registration marks to cars is one-one at any time. However, with permission of the DVLA, registration marks can be transferred from one vehicle to another. Thus, the marks are unique identifiers that are *time-dependent*; they are not permanent unique identifiers for the vehicles. There are identifiers for vehicles that are permanent: in the UK, the *vehicle identification number* (VIN) consists of 17 characters that identify the manufacturer (three characters), the type of vehicle (six characters), and finally distinguishes each of the cars with these characteristics (eight characters). The VINs obey some international standards.

A car has one and only one registered keeper. The registered keeper is the person who is legally responsible for the car, and need not to be the owner of the vehicle. One purpose of the mark is to identify the keeper: thus, the association of a registration mark to a keeper is unique. However, a person can be a registered keeper of as many cars as he/she wants. Thus, the association of registration marks to keepers is many-one. The registration document (V5) for a car identifies the car by registration mark and VIN, and its keeper.

Many people have insurance policies that enable them to drive any car with the owner's permission. Thus, the driver of a car on a particular occasion may be only loosely connected to the keeper. The association between registration marks and drivers is complicated being one-many and time dependent, and incomplete in terms of formal documentation.

Example 2: Communications. This example demonstrates both many-one and many-many associations. When connecting a computer to the Internet, a number is needed called an Internet Protocol (IP) address that uniquely identifies the machine in the network; this number is 32 bits under Internet Protocol Version 4.[7] In some computer networks, such as networks local to an organization or company, there is an IP address for the machine that does not change; these are called static IP addresses. In this context, the association of computers to IP addresses is one-one. More commonly, at home, IP addresses are generated by an Internet Service Provider in response to a customer's need for Internet access. Thus, over time IP addresses can change and the association of IP addresses to a particular computer is many-one. Developing this example, if more than one computer is accessing the Internet at the same time in a period, from the same service, then the association between IP addresses and computers is many-many. The changing status seems to be natural in time-dependent associations of identifiers. However, each computer does have an identifier, called its MAC address (48 bits under IEEE 802), that identifies the device uniquely throughout its life. So, the association is one-one and time independent.

Example 3: Addresses. This example demonstrates a one-many association. In the UK, between 1959 and 1974, a system of postal codes was introduced to enable the automation of postal services. Typically, each address or location is assigned at most one postcode but a postcode can be assigned to more than one unit or building. The association between postcodes and buildings/addresses is one-many. Thus, postcodes are a system of identifiers that do not uniquely determine addresses. Local authorities determine addresses. Postcodes have found many uses and are used routinely in commercial transactions, navigation, and, more significantly, in calculating insurance, designing social policy and funding, and academic social studies — all of which are examples of social sorting.

For any system of identifiers for entities in a context, the questions arise:

> **Identifier Generation.** How does the system create and delete identifiers for entities?

7 In the Internet of Things, processors are embedded in products and places of all kinds. Thus, there is a need for many more IP addresses, prompting an upgrade of standards from Internet Protocol Version 4 to Internet Protocol Version 6 [28].

Identifier Authentication. Given two identifiers, how do we decide whether or not they are associated with the same entity?

Entity Authentication. Given an entity and identifier, how do we verify whether or not the identifier is associated with the entity?

Entity authentication is stronger than identifier authentication. The notion is attractive but not subtle for what does it mean to be 'given an entity'? In much theory and practice, the entity is actually 'given' by means of another identifier. We examine the relationship between identifiers in the Section 'Comparing identifiers'.

Example 4: Physical Verification of Entities. A biometric is an identifier that is designed to be verified by means of a physical process of identity authentication. The physical process involves instruments that make measurements, which are processed by software, and whose specifications involve probability theory. Questions arise about accuracy, equivalence across authenticating equipment, software portability, and, indeed, the probabilistic assumptions. However, the intention is clear: *through biometrics, physical reality verifies personal identity*.

A formal model of identity

We now consider formally the idea of a system of identifiers for the entities under observation. There are three aspects arising from our discussion of examples: *assigning identifiers, comparing identifiers* and *basic personal identifiers*. We will continue to use the formal notations introduced earlier in our formal definition of surveillance in the sub-Section 'Identity: identifying entities'.

Assigning identifiers

Definition. Let *Identifiers* be a non-empty set of identifiers and *Entity* a non-empty set of entities. Suppose that identifiers have been assigned to entities by means of a relation

$$id \subseteq Identifier \times Entity$$

such that

$$id(i, e) \iff \text{the data } i \in I, \text{ called an identifier, is assigned to entity } e \in E.$$

We define the set of entities named by identifier i by

$$ent(i) = \{e \in Entity \mid id(i, e)\}$$

and the set of all identifiers naming entity e by

$$id(e) = \{i \in Identifier \mid id(i, e)\}.$$

These sets are projections of the relation *id*.

The maps *ent(i)* and *id(e)* are needed to formalize the types of association in the Section 'What is identity?'. This idea is our most general definition:

Definition. A *system of identifiers* is a structure,

$$IdSys = (Identifier, Entity \mid id \subseteq Identifier \times Entity).$$

Example 1: Post Codes and Passwords. Recall Example 3 in the Section 'What is identity?': a postcode can be assigned to more than one building so the association is a one–many relation *postcode*: $Postcode \times Address$. Similarly, accounts are assigned one password, but passwords can be common to different accounts (e.g. proper names, birthdays, etc.). The association is a one–many relation *password*: $Password \times Username$.

Examples suggest that the following special case is most important.

Definition. A system of identifiers *IdSys* is said to satisfy the *many-one property* if each identifier is assigned to one and only one entity but an entity may have more than one identifier. In this case, the relation becomes a single-valued mapping

$$id: Identifier \to Entity$$

such that

$$id(i) = \text{the entity } e \in Entity \text{ named by the data } i \in Identifier.$$

The structure becomes an algebra:

$$IdSys = (Identifier, Entity \mid id: Identifier \to Entity).$$

Recalling the Search and Enumeration Principles in the Section 'What is identity?', we will focus on systems having this many-one property. Since the purpose of the identifiers is to recognise the entities that we are interested in, the following equivalence relation on *Identifier* is basic:

Definition. For any i_1 and $i_2 \in Identifier$, we say that they are *entity-equivalent* if they are associated with the same entity: in symbols,

$$i_1 \approx_{en} i_2 \text{ if, and only if, } id(i_1) = id(i_2).$$

The identifier captures and narrows down detection of entities. Thus, we can strengthen the system of identifiers if we can satisfy this condition:

Definition. A system of identifiers *IdSys* is said to satisfy the *one-to-one property* if the map *id* satisfies: for any i_1 and $i_2 \in Identifier$,

$$\text{if } id(i_1) = id(i_2) \text{ then } i_1 = i_2.$$

The map *id* is *one-to-one* or *injective, and entity-equivalence* $\approx_{\text{en}}$ is $=$.

Example 2: Cars. Recalling Example 1 in the Section 'What is identity?', the association of registration marks to cars is one–one.

Generating identifiers

How are identifiers generated for a set of entities in practice? First, some input data is presented to the system that has to be examined and approved according to some set of rules.

Definition. Let the initial data presented to a system in order to create an identifier be called a *form*. Let *Form* be the set of all possible forms for the system. The creation of an identifier is a mapping of the type:

$$generate: Form \to Identifier.$$

A form $f \in Form$ is the background data needed to create the identifier *generate(f)*.

We can refine this idea by separating the processing of the data from the release of the identifier. Let the processing of the form be represented by a function

$$check: Forms \to \{0, 1\}$$

that tests the data in a form $f \in Form$ for consistency against the system's rules. We assume that $check(f) = 1$ means the form is accepted and $check(f) = 0$ means the form is rejected.

We represent the next stage — if and when an identifier is to be issued — by a function

$$issue: Forms \to Identifier$$

which uses some or all of the data in $f \in Form$ to make an identifier.

The two stages are represented by composing the functions to make the new function

generate: Forms → *Identifier* ∪ *{reject}*

where

generate(f) = **if** *check(f)=1* **then** *issue (f)* **else** *reject*.

The idea of the form is seen in the familiar procedures of enrolment and registration required when applying to join organizations, schemes and services etc.

Personal identity

Of greatest interest is surveillance in which the entities are people. A fundamental problem is how identifiers can actually identify a specific individual. An individual's identity involves many characteristics — social, biographical, psychological and biometric — all of which can be presented digitally. A person identifier is very special data as it is fundamental to theories of trust, privacy and surveillance. Consider some examples of assigning data to individuals.

Examples

Example 1: Biometrics. Biometric identifiers are measurable qualities that can be used to describe and label the physical characteristics of individuals and enable the *automatic* recognition of people. Physiological and behavioural characteristics are related to the body, and there are many: some 9 leading biometrics, and a further 17 biometrics under development, are discussed in [16]. All of these physical measurements end up in software. The association of a biometric to people is expected to be highly reliable because it is expected to be one-one. Biometric digital technologies emerged in the 1960s with automatic fingerprint recognition [29, 30] – perhaps, the best understood automatic process [31].

The operational tests used to measure biometrics are of course, approximate, due to technical constraints, error margins and costs. Thus, that biometric data manifests a one–one identity association is a matter of *probability*, especially *high* probability. Increasingly accurate measurements are desirable or necessary. Although identical twins share very similar DNA, they are not identical [32]. The environment affects the genetics, possibly even in the womb. But the complexity of testing is considerable and is a research area [33]. Recently, public attention was drawn to these points when identical twins were identified by DNA evidence as suspects in a series of sexual assaults, in Marseille, France, and soon after in Reading, England. In the case of Marseille, after 10 months incarceration, one of the twins confessed [34]; in the case of Reading, mobile phone evidence revealed the offender [35]. At the time advanced DNA tests were not applied to separate the twins.

Example 2: Citizenship. In the UK, for example, an individual can or must register with state organizations devoted to health, employment, citizenship, and transport, and with local government organizations devoted to residence and elections. Everyone registered with the National Health Service has his/her unique number, which is linked to his/her health record. Each NHS number is made up of 10 alpha-numerics. Everyone gets a National Insurance (NI) number just before he/she turns 16. An individual's NI number makes sure his/her NI contributions and taxes are only recorded against her/her name. The format of the number is two prefix letters, six digits, and one suffix letter. In the new style red passport, in addition to the biometrics, there is a passport number that must be nine characters and all characters must be numeric. Finally, each driving licence has a number made up of 18 alpha-numerics, which codes part or all of (i) the surname; (ii) the date of birth; (iii) the first names; (iv) sex; (v) licence issue; and (vi) checks. In these cases of registration, numbers are added to identifiers in order to ensure that each of these associations is one–one. The ways in which the British state knows its citizens is complicated; plans in 2006 for (re-)introducing a national identity register were abandoned in 2011 [36, 37].

Formal personal identifiers

We have emphasized how systems of identity are designed to separate entities in contexts, how they are established with widely varying standards of rigour, and that they are combined and compared in all sorts of unanticipated ways. The fundamental personal identifiers mentioned in the sub-Section 'Examples' are much used because they carry weight: with the authority of the state, people are identified in basic contexts for citizenship, employment, tax and health.

Definition. A *personal identity system* has the form

PIdSys = *(Identifier, Person | pid: Identifier→ Person)*

and satisfies the uniqueness property, namely *two different people are assigned different data* and the function *pid* is one–one.

In practice, the data assigned to a person invariably includes a number or alpha-numeric code precisely in order to *enforce* the uniqueness property. All systems of identity need to be analysed by studying comparisons that involve mapping between different systems of identity, but this is especially true of personal identity systems.

Provenance of identifiers

Generating identifiers using other identifiers

Creating identifiers is an everyday occurrence: we open accounts, register for services, buy products, etc. For many of these actions, we rely on a handful of pre-existing identifiers. In the UK, to open a bank account, we give a proof of our identity and our current address, e.g. using a passport and a recent utility bill. To order a product or service, an address and a credit card account number are usually sufficient for the vendor to dispatch: notice the dependency on the bank identifier. At face value, the quality of a bank identifier is guaranteed by the databases of the state (passport, driver's licence) and local organizations (utility providers, local authorities). The passport provides a high quality identifier based on a birth certificate, a photograph and possibly other biometric data. Example after example, illustrates the general point that:

> **Principle.** *The creation of new identifiers is dependent upon pre-existing identifiers.*

The quality of an identifier is essentially a matter of its reliability, which in turn depends on

i. its provenance, i.e. the process involved in establishing the identifier; and
ii. scope, i.e. the context(s) in which it is accepted.

In the case of people, a passport and a driving licence are standard examples of high-quality identifiers with a rigorous provenance and wide application [38, 39]. In the case of a bank, where it is a now a priority to check on identity of existing customers, the process of identification can be clumsy and discriminatory, as women can experience when using both their maiden name (in their profession) and married name (in their personal life), which are often not linked rigorously in practical ways.

The dependence of one identifier upon another may be illustrated in an *identity dependence tree*.

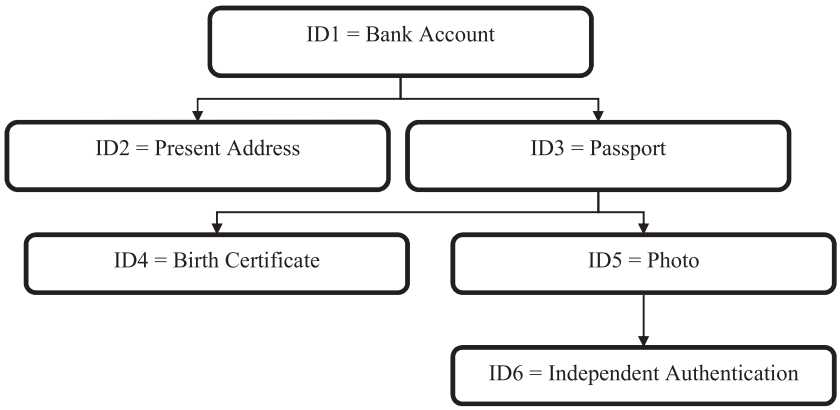

Figure 1: Dependency tree of identifiers

Example 1: Bank Account. Consider the role of identifiers in opening a bank account (in the UK), which is depicted in Fig. 1. Establishing the identifier ID1 of the account holder involves providing evidence using five other identifiers: the validity of ID1 depends upon, or is reduced to, the validities of ID2–ID6. Some of these identifiers have a special status, in that they are designed to reliably denote an individual. In the example, these personal identifiers are guaranteed by the state (ID4) and biometric data (ID5); in the latter case, ID6 is used to allow a passport to be issued by post, without face-to-face interaction. ID2 is used to confirm the validity of the account holder's address.

The identifiers that appear in the nodes of the tree suggest that there can be quite complicated dependencies between *systems* of identifiers for the same or, more commonly, different contexts. The identifier is made by aggregating pre-existing identifiers: the bank identifier in Fig. 1 is the sum of the identifiers for current address, birth and image, etc.

Since identifiers are often built from other identifiers, of central importance is the process of comparing identifiers and relating one type of identifier to another. Indeed, there must be translations between distinct systems for these identifiers for such methods to work. All of these observations and ideas can be formalized to make a precise and general mathematical framework for analysing identifiers. The identity dependence tree is a flexible notion with many more applications than proving personal identity.

Example 2: Namespaces. Namespaces are sets of identifiers that use symbols to label, organize and classify entities by names. The names can have a tree structure that enables them to be reused and to form a hierarchy. For example, the names for directories, folders, files, and web domains, etc. are made by concatenating names and denote paths in a tree: the web address

http://www.swansea.ac.uk/library/archive-and-research-collections/hocc

for the History of Computing Collection is a node belonging to the archives which in turn belong to the library of Swansea University. Indeed, there is no shortage of computing contexts where identity dependence trees are used. Domain name systems (e.g. URLs), directory services for networks (e.g. Microsoft's Active Directory), email addresses (e.g. X500), authentication systems (e.g. Kerberos), and public key infrastructures (e.g. blockchains) are natural sources of rules and structures for creating identifiers.

Example 3: Identity Fraud. The creation of new personal identities requires many identifiers to be fabricated: birth certificates, driving licences, employment histories, etc. The practicalities for the USA are discussed extensively in Kevin Mitnick's memoir [40]. When a fugitive, his method for creating a new identity in different states can be depicted as an identity dependence tree. More generally, Mitnick's success at social engineering is based on his extensive preparation, which focussed on researching identifiers that he would use in masquerades in the technical, commercial and government contexts of phone system companies, computer and phone manufacturers, and state agencies.

The complexity of computing systems suggests that tracing the provenance of a component may lead to circularity and so there may be a need for graphs of identifiers with cycles.

Generating identifiers from identifiers

Now suppose that to generate an identifier for an entity the input data involves other identifiers that must be presented to verify some of the new data (such as personal identity). The general ideas of the sub-Section 'Generating identifiers' can be reformulated with provenance in mind. We revise the processing of the form with a function with new variables:

$$check: Forms \times Identifier_1 \times \ldots \times Identifier_k \rightarrow \{0, 1\}$$

that tests the data in a form $f \in Form$ and the information available from identifiers $i_1, \ldots, i_k$ for consistency against the system's rules. Again, we assume that $check(f, i_1, \ldots, i_k) = 1$ means that the form is accepted and $check(f, i_1, \ldots, i_k) = 0$ means that the form is rejected.

The identity of an entity with identifier i depends upon the identifiers $i_1, \ldots, i_k$. This idea is formalised by re-representing the function

$$generate: Forms \rightarrow Identifier \cup \{reject\}$$

(in section 'Generating identifiers') by the new function

$$generate: Forms \times Identifier_1 \times \ldots \times Identifier_k \rightarrow Identifier.$$

There are now two ways of creating the identifiers and defining *generate*, defined by two principles:

Provenance Principle: Verification. *The data in $f \in Form$ is sufficient to create an identifier i. The data in the identifiers $i_1, \ldots, i_k$ are used only to confirm or validate the data in f.*

Here the function has the form

$$generate(f, i_1, \ldots, i_k) = \textbf{if } check(f, i_1, \ldots, i_k)=1 \textbf{ then } issue\ (f) \textbf{ else } reject$$

noting that *issue (f)* does not need to know the validation identifiers. Secondly, we have the more demanding case:

Provenance Principle: Inheritance. *The data in $f \in Form$ to create an identifier i is inherited from the data in the identifiers $i_1, \ldots, i_k$.*

In this case, the function has the form

$generate(f, i_1, \ldots, i_k)$ = **if** $check(f, i_1, \ldots, i_k)=1$ **then** $issue(f, i_1, \ldots, i_k)$ **else** *reject*.

Comparing identifiers

Access to data belonging to different contexts is desirable in surveillance, intelligence analysis, and academic research; it is undesirable in social and personal contexts as it undermines privacy and freedom. Access is regulated by legal instruments.

Reductions between systems of identifiers

Consider the case where a set *Entity* of entities has two systems of identifiers:

$$IdSys_1 = (Entity, Identifier_1 \mid id_1: Identifier_1 \rightarrow Entity),$$
$$IdSys_2 = (Entity, Identifier_2 \mid id_2: Identifier_2 \rightarrow Entity).$$

How can we relate or compare these systems?

One simple case is when the identifiers in $Identifier_1$ can be associated or matched with one or more identifiers in $Identifier_2$, and *vice versa*. This means that given an identifier $i \in Identifier_1$ of an entity $e \in Entity$, we can find corresponding identifiers in $Identifier_2$ that are *also* identifiers for e. This is formalized as follows:

Definition. Let $IdSys_1$ and $IdSys_2$ be systems of identifiers for *Entity*. A *matching relation*

$$r: Identifier_1 \times Identifier_2$$

for the systems of identifiers $IdSys_1$ and $IdSys_2$ compares identifiers as to whether or not they are associated with the same entity in the following sense: for every $i \in Identifier_1$ and $j \in Identifier_2$,

$r(i, j)$ if, and only if, $id_1(i) = id_2(j)$.

Different conditions on a matching relation can be found in examples, depending upon the properties of id_1 and id_2. An important and common case is: given an identifier $i \in Identifier_1$ of an entity $e \in Entity$, we can find *some* corresponding identifier in $Identifier_2$ that is also an identifier for e. This is formalized as follows:

Definition. Let $IdSys_1$ and $IdSys_2$ be systems of identifiers for *Entity*. The system of identifiers $IdSys_1$ is said to *reduce* to the system of identifiers $IdSys_2$ if there is a single-valued *reduction mapping*

$$f: Identifier_1 \rightarrow Identifier_2$$

that calculates for each identifier in $Identifier_1$ a corresponding identifier in $Identifier_2$ for the same entity in the following sense: for every $i \in Identifier_1$,

$$id_1(i) = id_2(f(i)).$$

We write $IdSys_1 \leq IdSys_2$ or, more simply and conveniently, $id_1 \leq id_2$ (see: Fig. 2).

This is but one formalization of the process of comparing the identifiers of $Identifier_1$ to those of $Identifer_2$. Another option would be to return a selection, or all, of the equivalent identifiers. Because the notion of identifier is so abstract, the notion of reduction is very general. Mappings between identifiers are ubiquitous in computing systems and employ many algorithmic techniques. Reductions can be found in situations where alternate terms like 'translating', 'binding', 'matching', and 'tracing' are used.

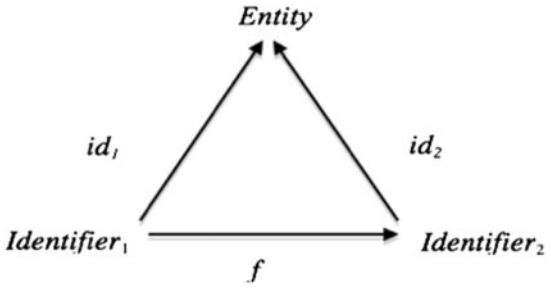

Figure 2: Transformation of identifiers

Example 1: Tracing. Consider the set *Keep* of keepers of vehicles in the UK and two systems of identity for this set of entities. Suppose, for simplicity, each keeper has one car and each keeper has a unique address. Each car has a registration mark. Let the first system be

$$Reg = (Keep, Regmk \mid reg: Regmk \rightarrow Keep).$$

Every keeper has an address assigned by the postal service so let

$$Add = (Keep, Address \mid addr: Address \rightarrow Keep).$$

Then the Driver and Vehicle Licensing Agency (DVLA) is responsible for the determining the keeper's address from the registration mark, which is defined formally by the reduction map $red: Regmk \rightarrow Address$ such that for every registration mark $r \in Regmk$,

$$reg(r) = addr(red(r)).$$

We say that the system of identities *Reg* is reducible to *Add*.

Example 2: Hashing. In cybersecurity, hashing techniques provide examples of reductions. For example, consider hashing in managing passwords. Hashing involves a one-way function $h: Password \rightarrow \{0, 1\}^k$ where $h(w)$ is a data used to separate w in some context.[8] There are many hashing algorithms, such as the secure hash algorithms SHA-256 and SHA-512; and there are methods to enhance their security such as *salting*, where random strings are added to the passwords to separate common passwords from each other. Thus, hash codes are identifiers and the hash function and salting qualify as reductions.

Example 3: Binding. Connections between computing entities require various degrees of reliability and, in secure contexts, trust. In computing, a *binding* is a mapping associating distinct entities in hardware or software. Commonly, bindings are mappings between syntactic spaces (e.g. namespaces) enabling binding to connect syntactic and semantic entities, or to create layers in software stacks, or create secure chains of identity in cryptography. The term binding has general application and several common forms of binding qualify as reductions between systems of identifiers in our sense.

Example 4: Certification. Certification is a security process that seeks to increase trust in identity. It is intended to reduce risks of man-in-the-middle vulnerabilities. In communications, such as calling a webpage, certification can flag doubts about a website. In cryptography, a public key certificate is used to confirm the ownership of a public key. The certificate validates the binding of a public–private key pair to an entity, using a digital signature generated by a certificate authority.

8 A one-way function is easy to compute but hard to invert.

Definition. The system of identifiers $IdSys_1$ is said to be *equivalent* to the system of identifiers $IdSys_2$ if there are reduction mappings

$$f\text{: } Identifier_1 \to Identifier_2 \text{ and } g\text{: } Identifier_2 \to Identifier_1$$

that can exchange identifiers in $Identifier_1$ and corresponding identifiers in $Identifier_2$. We write $IdSys_1 \approx IdSys_2$ or, more simply and conveniently, $id_1 \approx id_2$.

Structuring the space of identifiers

Reductions are an important concept that occur widely. To conclude, we introduce some concepts and propositions to reveal the richness of the reduction notion and signal the possibility of advanced classification methods.

In the Section 'Provenance of identifiers', we discussed the combination of identifiers. The process of creating new identifiers from old introduces algebraic operations on *spaces* of identity systems. One choice of algebraic structure, the *semilattice*, organizes the space of all possible identity systems using reduction.

Lemma. Let *IdSys(Entity)* be the set of all identity systems for the non-empty set *Entity* of entities. The reduction relation $\leq$ on *IdSys(Entity)* is reflexive and transitive; and $\approx$ is an equivalence relation on *IdSys(Entity)*.

Proof. Let $IdSys = (Entity,\ Identifier \mid id\text{: } Identifier \to Entity)$. Trivially, $id \leq id$ using the identity function $Identifier \to Identifier$ as reduction map; so reduction is reflexive.

To show transitivity, let

$$IdSys_1 = (Entity,\ Identifier_1 \mid id_1\text{: } Identifier_1 \to Entity),$$
$$IdSys_2 = (Entity,\ Identifier_2 \mid id_2\text{: } Identifier_2 \to Entity),$$
$$IdSys_3 = (Entity,\ Identifier_3 \mid id_3\text{: } Identifier_3 \to Entity).$$

and suppose

$id_1 \leq id_2$ by $f\text{: } Identifier_1 \to Identifier_2$ and $id_2 \leq id_3$ by $g\text{: } Identifier_2 \to Identifier_3$.

Then, for $i \in Identifier_1$, and $j \in Identifier_2$, we have

$id_1(i) = id_2(f(i))$ and $id_2(j) = id_3(g(j))$.

Composing, f and g we have,

$$id_1(i) = id_3(g(f(i)))$$

and $id_1 \leq id_3$. It is easy to show that $\approx$ is symmetric.

Using the equivalence relation $\approx$ on *IdSys(Entity)*, we define the set of equivalence classes:

$$\textbf{\textit{IdSys(Entity)}} = IdSys(Entity)/\approx.$$

The equivalence classes have the standard form of $[id]$ for $id \in IdSys(Entity)$. The ordering relation $\leq$ on *IdSys(Entity)* induces ordering relation $\leq$ *on* ***IdSys(Entity)*** by

$$[id_1] \leq [id_2] \iff id_1 \leq id_2.$$

It is easy to check that $\leq$ is a partial ordering on ***IdSys(Entity)***. Furthermore, the ordering $\leq$ has the *least upper bound property*: for any $[id_1]$, $[id_2] \in$ ***IdSys(Entity)***, there is an element $[id]$ such that:

i. $[id]$ is an upper bound: $[id_1] \leq [id]$ and $[id_2] \leq [id]$;
ii. no lower element is a bound: if $[id_1] \leq [id_0] \leq [id]$ then either $[id_1] = [id_0]$ or $[id_0] = [id]$.

To show this we construct an identity system as follows. Given id_1, $id_2 \in$ ***IdSys(Entity)***, take the disjoint union $Identifier_1 \oplus Identifier_2$ of the sets of $Identifier_1$, and $Identifier_2$ and define

$$id_1 \oplus id_2\text{: } Identifier_1 \oplus Identifier_2 \to Entity.$$

wherein given $i \in Identifier_1 \oplus Identifier_2$,

$(id_1 \oplus id_2)(i) = id_1(i)$ if $i \in Identifier_1$
$(id_1 \oplus id_2)(i) = id_2(i)$ if $i \in Identifier_2$.

It is easy to show that $[id_1 \oplus id_2]$ satisfies conditions (i) and (ii). The construction

$$(Identifier_1 \oplus Identifier_2 \mid id_1 \oplus id_2)$$

is called a *co-product* of the identity systems. If the sets $Identifier_1$ and $Identifier_2$ are disjoint (as is often the case) then the carrier is their union.

Example: Combining Identifiers. Integration of identity data can be tentatively explored using coproducts. Consider making a system of identifiers for entities that are contracts, for which personal identity and current location must be validated. A space of identifiers may be built using the coproducts of pairs of validating systems of identifiers: passports, driver licences, identity cards for identity, and utility bills, local tax declarations for addresses.

A partial ordering with the least upper bound property is called an *upper semilattice* [41]. Thus, gathering together these arguments we have the theorem:

Theorem. *The reduction relation $\leq$ on IdSys(Entity) forms an upper semilattice.*

Corollary. *The process of creating new identifiers by inheriting existing identifiers forms an algebraic structure* ***IdSys(Entity)*** *that is an upper semilattice under the reduction relation.*

Equivalently, any upper semilattice can be reconstructed as an algebraic structure with a binary operation $\wedge$ that is associative, commutative, and idempotent [40]. In this form we would have the structure

$$\textbf{\textit{IdSys(Entity)}} = (IdSys(Entity)/\approx \mid \wedge)$$

with binary operation of least upper bound defined by

$$[id_1] \wedge [id_2] = [id_1 \oplus id_2].$$

Further properties of the upper semilattice ***IdSys(Entity)*** can be developed depending upon properties of the associations and reductions.

Concluding remarks

Employing simple examples and arguments from first principles, we have used formal methods to analyse precisely concepts involved in surveillance and identity. The formal analysis shows that disparate forms of surveillance can be unified by abstract mathematical definitions, and that (i) finding identities, and (ii) sorting identities into categories, are fundamental in conceptualizing surveillance. The formal analysis of identity shows that the idea of identity can be considered to be exclusively a matter of data, and its diversity can be unified by abstract mathematical definitions. It also shows that (i) comparing identifiers, and (ii) translating between systems of identifiers, are fundamental to understanding identity.

Developing a theory of identity

The theoretical analysis presented here is intended to analyse examples and formalize intuitions and ideas. This formal approach is new and inevitably modest, but it can serve as a basis for further conceptual and mathematical investigations relevant to the making and regulation of surveillance systems. Two conceptual and four technical further directions seem to us to be desirable. Conceptually, our theory of identifiers could be used to theorize privacy, interpreted as the *control identity*. (A formal notion of anonymity was included in the sub-Section 'Surveillance: detecting attributes'.) The second direction is to use identifiers to explore secure access control policies in computer systems (e.g. role-based access control).

Turning to technical directions, first the notion of context in the sub-Section 'Behaviour as streams of data' can be developed with various semantic models. Further behavioural features can be formalized, such as the interaction of entities. There are options for formalizing streams – infinite or finite, total or partial streams in discrete or continuous time – and for behaviours modelled as nondeterministic and concurrent processes [42]. Secondly, logics can be used to develop specification and reasoning about attributes. There are several candidates, such as many sorted first order logic and its many derivatives and its extensions – equational, Horn, and temporal logic [43]; and many valued logics [44]. Logics bring with them tools that would expand the scope of the theory and applications.

Thirdly, the idea of a system of identifiers needs to be developed mathematically. For example, identifiers commonly reference *characteristics* of the entities that may be essential to their deployment and application; this information is an additional component that would enrich the mathematical structure of contexts and their identifiers. The idea of the *form* does provide background initial information, but the characteristics of an entity may need updating because of the behaviour of the entity in time. Systems of identifiers are instances of abstract data types [25], whose extensive general theory based on many sorted algebras and equations [18, 19, 45, 46] can add significantly to the theory and practice of identity.

Fourthly, identifiers are assumed to be digital objects. To model this aspect, identifiers must themselves be coded by bits, i.e. by finite binary strings over {0, 1}. This introduces a new *digital layer* beneath the identifiers in which all computation actually takes place, simulating functions on user data by functions on binary numbers. This digital layer is a source of constraints on the theory of identifiers. The digital layer can be modelled by maps of the form

$$code\colon Numbers \to Identifier.$$

Classical computability theory is a mathematical theory of what can and cannot be computed on numbers [47], especially binary and decimal, etc. It has been applied to establish the scope and limits of computation on arbitrary data using such maps as *code*; these maps are called *numberings* in computability theory [48]. Thus, there is a second ready-made theory that can applied to develop this three-layer model:

$$id \circ code\colon Numbers \to Identifier \to Entity$$

and theorize what is, and what is not, computable about systems of identifiers. The conception of identity analysed here is inspired by and abstracts ideas about abstract data types and encodings cf. [49].

Formalizing identity and social theory

Ours is an investigation into ideas about surveillance and identity, wherein our models are developed from first principles. It may seem far from the world implied by the revelations of Snowden, with its surveillance tools (XKeyscore, Tempora, etc.) for target discovery and development. Let us observe that new surveillance contexts arise — or are recognized — as more of our professional and social activities are carried out by abstract technological systems rather than by direct face-to-face interactions. To make use of these systems, an individual needs to give over some of his/her identity to distinguish himself/herself from other users in the context. Thus, rather than having a single and holistic identity, individuals now have *many* separated and overlapping identities, which amplifies hugely the scope of a theory of identifiers.

The physical and the virtual are converging; indeed, it seems that the physical world is being sucked into the virtual and a virtual world is being created that is self-contained. It certainly exerts a strong influence on the physical world, and shows signs of autonomy. Thus, the components of monitoring and surveillance — context, entity, observable behaviour, attribute and identity — will seem natural in a world held together by data and software.

The multiplicity of contexts and identities, and the possibility of the autonomy of the virtual world, requires the nature of identity to be theorized. The formal framework we offer here is a rigorous analysis of the conceptual structure of surveillance; there ought to be others. What can formalization contribute? Guided by the theory of abstract data types, our formalization of identity aspires to:

i. establish and explore principles that assume identity is a matter of data and their implications;
ii. make precise essential concepts and classify abstractly methods of identification;
iii. provide a unified point of view that illuminates the design of many real systems;
iv. explore the role of identity in aspects of security studies, including monitoring and surveillance, personal privacy and trusted translations and interactions.

At this stage, these aims require a great deal of further work.

Finally, let us observe that if a social science topic is closely associated with abstract technologies that collect and process data effectively then the specification of the software tools — i.e. what the tools are designed to do for users — can be formalized in the same way as we have approached the problem here. Thus, sociological notions that motivate, shape and are ultimately represented in software, can be defined in a formal framework which can be mathematically analysed. In short, sociological theories about human activities that are closely associated with abstract software systems can be expected to have formal models, mathematical theories, as well as oodles of data arising from their use.

To isolate, define and analyse ideas is the *raison d'être* of formal methods, though in new areas their mathematical nature presents obstacles to their reception and appreciation. The use of formal methods to express and analyse general notions is established in areas of philosophy and linguistics but seems to be rare in social studies. Given software's colonization of professional and social life, and its promotion of monitoring and Big Data, the role of formal methods to theorize social concepts and problems is destined to grow.

Acknowledgement

We thank two anonymous referees for valuable suggestions that improved an earlier version of this paper. This research was partially supported by the EPSRC project *Data Release - Trust, Identity, Privacy and Security* (EP/N028139/1 and EP/N027825/1).

References

1. Haggerty K, Ericson R. The surveillance assemblage. *British J Sociol* 2000; **51**:605–662.
2. Introna L, Wood D. Picturing algorithmic surveillance: the politics of facial recognition systems. *Surveillance & Society* 2004; **2**:177–98.
3. Evans D. *The Internet of Things: how the next evolution of the internet is changing everything*. CISCO White Paper, 2011.
4. Hopper A. Sentient computing. *Phil Trans Royal Society* 2000; **358**: 2349–58.
5. Thrift N. The 'sentient' city and what it may portend. *Big Data & Society* 2014; **1**:1–21.
6. Foresight. *Future Identities Changing identities in the UK: The next 10 years (Final Project Report)*. Government Office for Science, 2013.
7. Anderson D. *A Question of Trust. Report of the Investigatory Powers Review*. HM Stationary Office, 2015.
8. Lyon D. *Surveillance Studies: An Overview*. Polity Press, 2007.
9. Ball K, Haggerty KD, Lyon D. *The Routledge Handbook of Surveillance Studies*. Routledge, 2012.
10. Wills D. *Surveillance and Identity: Discourse, Subjectivity and the State*. Ashgate, 2013.
11. Torpey J. *The Invention of the Passport: Surveillance, Citizenship and State*. Cambridge University Press, 2000.
12. UK Government. *The Identity Card Act 2006: Elizabeth II. Chapter 11*. The Stationary Office, 2006.
13. Cole S. *Suspect Identities: A History of Fingerprinting and Criminal Identification*. Harvard University Press, 2001.
14. Lyon D. Under my skin: from identification papers to body surveillance. In: Caplan C (ed.), *Documenting Individual Identity: The Development of State Practices in the Modern World*. Princeton University Press, 2001.
15. Wallace H. The UK National DNA Database – Balancing crime detection, human rights and privacy. *Science & Society* 2006; *EMBO reports 7*: S26–S30.
16. Vacca J. *Biometric Technologies and Verification Systems*. Elsevier, 2007.
17. van der Ploeg I. Biometrics and the body as information: narrative issues of the socio-technical coding of the body. In: Lyon D (ed.), *Surveillance as Social Sorting: Privacy, Risk and Digital Discrimination*. Routledge, 2003.
18. Ehrich HD, Loeckx J, Wolf M. *Specification of Abstract Data Types*. Wiley, 1996.
19. Goguen J, Thatcher J, Wagner E. An initial algebra approach to the specification, correctness and implementation of abstract data types. In: Yeh R (ed.), *Current Trends in Programming Methodology, IV*. Prentice-Hall, 1978.
20. Lipschutz S, Lipson M. *Discrete Mathematics*, 3rd edn. Schaum, 2009.
21. Makinson D. *Sets, Logic and Maths for Computing*. Springer, 2012.
22. Pieri E. Emergent policing practices: operation shop a looter and urban space securitisation in the aftermath of the Manchester 2011 riots. *Surveillance & Society* 2014; **12**:38–54.
23. Morstatter F, Pfeffer J, Huan L, *et al. Is the Sample Good Enough? Comparing Data from Twitter's Streaming API with Twitter's Firehose*. Arxiv, 2013. http://arxiv.org/abs/1306.5204v1 (14 March 2017, date last accessed).
24. Needham RM. Names. In: Mullender S. (ed.), *Distributed Systems*. ACM Press, 1989, 89–101.
25. Liskov B, Zilles S. Programming with abstract data types. In: *Proceedings of the ACM SIGPLAN Symposium on Very High Level Language*. ACM Press, 1974, 50–59.
26. Lyon D (ed). *Surveillance as Social Sorting: Privacy, Risk and Digital Discrimination*. Routledge, 2003.
27. Chen M, Mao S, Liu Y. Big data: a survey. *Mobile Network and Applications* 2014; **19**:171–209.
28. Minoli D. *Building the Internet of Things with IPv6 and MIPv6: The Evolving World of M2M Communications*. Wiley, 2013.
29. Trauring M. On the automatic comparison of fingerprint ridge patterns. *Nature* 1963; **197**:938–40.
30. Wayman JL. The scientific development of biometrics over the last 40 years. In: Leeuw K, Bergstra JA (eds), *The History of Information Security: A Comprehensive Handbook*. Elsevier, 2007, 263–74.
31. Maltoni D, Maio D, Jain A. *et al*. *Handbook of Fingerprint Recognition*. Springer, 2009.
32. Choi CQ. Copy that: identical twins are not genetically identical. *Scientific American* 2008;**298**:24–26.
33. Twin DNA test: Why identical criminals may no longer be safe, 5 January 2014. http://www.bbc.co.uk/news/magazine-25371014 (14 March 2017, date last accessed).
34. Mystery of which identical twin committed a series of rapes in France is finally solved as one brother confesses after he was given away by a stutter. http://www.dailymail.co.uk/news/article-3225467 (14 March 2017, date last accessed).
35. Identical twins need never be tried for same crime after DNA breakthrough. http://www.telegraph.co.uk/news/science/science-news/10511087/Identical-twins-need-never-be-tried-for-same-crime-after-DNA-breakthrough.html (14 March 2017, date last accessed).
36. Caplan J, Torpey J. (eds). *Documenting Individual Identity: The Development of State Practices since the French Revolution*. Princeton University Press, 2001.
37. Higgs E. *Identifying the English: A History of Personal Identification 1500 to the Present*. Bloomsbury, 2001.
38. Lloyd M. *The Passport: The History of Man's Most Travelled Document*. Queen Anne's Fan, 2016.
39. Castile M. *Driver's License*. Bloomsbury, 2015.
40. Mitnick K. *Ghost in the Wires*. Little Brown & Company, 2011.
41. Birkhoff G. *Lattice Theory*, 3rd edn. American Mathematical Society, 1995.
42. Bergstra JA, Ponse A, Smolka SA. (ed). *Handbook of Process Algebra*. Elsevier, 2001.
43. Manzano M. *Extensions of First-Order Logic*. Cambridge University Press, 2005.
44. Gottwaldov S. *A Treatise on Many-Valued Logics. Studies in Logic and Computation*, vol. 9, Research Studies Press, 2001.
45. Meseguer J, Goguen JA. Initiality, induction, and computability. In: Nivat M, Reynolds JC (eds), *Algebraic Methods in Semantics*. Cambridge University Press, 1986.
46. Meinke K, Tucker JV. Universal algebra. In: Abramsky S, Gabbay D, Maibaum, T (eds), *Handbook of Logic in Computer Science. Volume I: Mathematical Structures*. Oxford University Press, 1992, 189–411.
47. Griffor ER. *Handbook of Computability Theory*. Elsevier, 1999.
48. Ershov Y. Theory of numberings. In: Griffor ER (ed.), *Handbook of Computability Theory*. Elsevier, 1991, 473–503.
49. Stoltenberg-Hansen V, Tucker JV. Effective algebras. In: Abramsky S, Gabbay D, Maibaum T (eds), *Handbook of Logic in Computer Science. Volume IV: Semantic Modelling*. Oxford University Press, 1995, 357–526.