CHAPTER 4

Source code disclosure: A primer for trade negotiators

Cosmina Dorobantu, Florian Ostmann and Christina Hitrova[1]

The Alan Turing Institute and the Oxford Internet Institute; The Alan Turing Institute; Technical University of Munich

1 INTRODUCTION

In 1982, Therac-25 appeared on the market and was immediately hailed as a state-of-the-art medical machine. Produced by Atomic Energy of Canada Limited and sold to hospitals in the US and Canada, Therac-25 delivered computer-controlled radiation therapy. It was the most technologically advanced radiation therapy machine in use at the time, admired for its precision and reliance on an onboard computer. And yet, between June 1985 and January 1987, Therac-25 overdosed six patients with radiation doses that were hundreds of times higher than normal. Some patients died, while others were severely injured. Detailed investigations of the accidents followed, and their conclusions were clear: the deaths and injuries of these patients were, in no small part, due to errors made in the source code underlying Therac-25's software (Levenson and Turner 1993).

Since the tragic accidents caused by Therac-25, the use of software has become much more widespread. Nowadays, software is an integral part of our lives. Lines of code determine the news that we read online, power the smart speakers in our homes, recommend some of the medical treatments we receive, and help us navigate unfamiliar roads. Areas of our lives that, only a generation ago, we could not imagine would be touched by code – from our friendships to our work meetings – are now increasingly curated by software-powered technologies.

Software doesn't only control the devices in our homes and pockets. Businesses also increasingly rely on software to design their products, improve their operations, manage their employees, and advertise the goods and services they sell. In 2019, software made up over 10% of the UK's gross fixed capital formation – approximately £40.9 billion (Office for National Statistics 2020). Even industries with low levels of digitisation – from agriculture, forestry, and fishing to construction and real estate – are now investing heavily in software development.

1 The authors would like to thank Alan Winters, Ingo Borchert, David Leslie, and Olivier Le Gouanvic for insightful comments and valuable suggestions.

Software's presence in our daily lives and business operations is not only set to continue; it is poised to increase. As networked computing, digital interconnectedness, cost-efficient data storage, ever-increasing computing power, and financial investments continue their upward movement, software will only grow in importance. This growth will present individuals and businesses alike with unprecedented opportunities as well as significant challenges.

Policymakers have the difficult task ahead of striking the right balance between realising the benefits that software brings and minimising the risks. This task becomes more daunting when considering the international dimension. Code can easily cross borders, but it is not just code, on its own, that companies export. Products and services that incorporate or rely on software are also traded internationally. From watches and cars to medical devices and financial products, traded goods and services increasingly have software embedded in them. Software's increasing presence in internationally traded goods and services makes it an international matter, affected by future trade negotiations.

Recent trade agreements – from the Japan–Mongolia Economic Partnership Agreement to the United States–Mexico–Canada Agreement – incorporate specific provisions related to source code. These provisions prohibit governments and their agencies from requiring "the transfer of, or access to, source code of software owned by a person of the other Party."[2] This general prohibition on public authorities requiring the transfer of, or access to, source code matters. On the one hand, it encourages international trade by reassuring foreign software developers that they will not have to disclose the source code underlying their products and services. On the other hand, this general prohibition, even when accompanied by extensive exemptions, places limitations on the powers of governments and their agencies to examine source code.

Trade negotiators are increasingly having to take a stance on whether a general prohibition on governments requiring access to source code is desirable in a trade agreement, and if so, which exemptions should accompany it. Deciding on the best course of action requires a solid understanding of what source code is, how it functions, what can go wrong with it, and when governments and their agencies may have legitimate reasons to implement measures aimed at source code accessibility. It also requires a general understanding of how source code disclosure has been handled in existing trade agreements and what the limitations of the previous approaches are.

Surprisingly little has been written to help trade negotiators navigate the tricky – and expanding – territory that software occupies in our lives and trade agreements. While several publications dedicate a few pages to provisions on access to source code (e.g. Guglya

2 The Appendix contains the text of articles from recent trade agreements related to source code. This citation is common to most articles reflected in the Appendix.

and Maciel 2016, Wu 2017, McCann 2019), to our knowledge, no publication presents a holistic picture of how recent trade agreements have handled source code disclosure and why such disclosure might be necessary.

This chapter aims to fill this gap and help trade negotiators build the knowledge they need to confidently handle provisions related to source code in future trade agreements. The first section provides an introduction to source code. It defines what source code is, discusses our ability to understand what a piece of code does, and raises awareness of the role of humans when things go wrong. The second section provides an overview of possible motivations for government-mandated source code disclosure requirements and the forms that such requirements can take. The third section examines how source code disclosure has been handled in recent trade agreements. The final section concludes.

2 SOURCE CODE: BASIC CONCEPTS

In this section, we define source code and provide an example of what lines of code look like. We introduce two programming approaches: the traditional top-down, rules-based approach; and the more recent bottom-up, machine learning-based approach. We give a sense of what we can and cannot understand by looking at a piece of code. Finally, we end the section by reminding the reader that code does not act on its own volition. When things go wrong, they often do so as a result of cognitive limitations, errors, negligence, or ill intent on the part of humans.

2.1 Definition

Source code refers to the lines of code written by programmers to instruct a machine to perform a given task. Source code is usually written in a text file, it is readable by humans, and it uses a programming language, such as Python, C++, Java, or R.[3] For example, the simple lines of code below are written in Python and instruct a computer to prompt a user to enter their age and to convert the entered value into an integer number. If the resulting number is equal to or greater than 18, the code instructs the computer to display the message "You can vote in UK elections." If it is below 18, the computer will display the message "You cannot vote in UK elections."

```
age = int(input("Enter your age "))
if age >= 18:
      print("You can vote in UK elections")
else:
      print("You cannot vote in UK elections")
```

3 There are numerous other programming languages available.

Source code exists on a continuum, ranging from very simple code, like the lines above, to highly complex code. Pieces of code that contain a series of steps that need to be followed in order to solve a computational problem are often called algorithms.

2.2 Programming approaches

The traditional way of programming relies on explicitly programmed rules. Returning to the simple example above, if we wanted to produce a piece of code that lets a user know whether they are old enough to vote in the UK elections, we would rely on the top-down application of logical statements and rules, such as 'if this person's age is greater or equal than 18, then display the message "You can vote in UK elections"'.

A newer and more powerful way of programming relies on machine learning. In contrast to more traditional approaches, machine learning systems rely on data to 'learn' how to carry out a certain task in a bottom-up, inductive way, rather than being explicitly programmed to follow a set of predefined rules (Leslie et al. 2020: 36). For example, if we wanted to produce a piece of code that correctly identifies a dog in an image, instead of explicitly programming rules that describe what a dog looks like, we can write a piece of code that 'learns' what a dog looks like by identifying patterns that arise across the repetitious processing of thousands of labelled images.

The growing popularity of machine learning approaches is due, in no small part, to their ability to extract relevant features and properties from large datasets. To carry on with the dog image example further, in the past, creating a computer programme that had hard-coded rules of what a dog looked like was a monumental task. Dogs come in many shapes and sizes – a giant schnauzer, for example, looks very different from a westie. Pictures differ, too. Some pictures might be close-ups of a dog's face, while others might be landscape views of dogs running across a field. Some pictures might have good lighting, while others might have significant shading – or worse yet, other animals or people in them. While our human brains find it easy to recognise a dog in each one of these situations, it is difficult for human programmers to specify explicit rules that, when embedded in code, would equip a computer program with the same recognition capabilities. Machine learning has helped to solve this problem for us. Instead of requiring hard-coded rules of recognition, machine learning algorithms extract the relevant features and properties of what a dog looks like from large datasets.[4] This is a powerful and transformative technology that is fundamentally different in its approach from top-down, rule-based programming.

The reliance of bottom-up approaches on a 'learning' process gives rise to a consequential difference between them and top-down approaches. When it comes to the design of a system, what matters in top-down approaches is the way in which expert knowledge is translated into specific rules. In machine learning approaches, the 'learning' process is

4 For readers interested in the history, development, and risks of facial recognition technologies, Leslie (2020) provides an excellent review.

what guides a system to extract relevant information from the data. Questions such as ‘how do we pick an appropriate target variable?’, ‘how do we choose the input variables that the system relies on?’, and ‘how do we model the relationship between the input variables and the target variable?’ are of paramount importance in machine learning. The answers to questions like these form the basis for arriving at the mathematical formulation that underpins the machine learning system. We refer to this as the system’s ‘logic’ or rationale. When we try to understand the inner workings of a machine learning system, it is extremely useful if, in addition to the source code, we also have access to an expression and elucidation of the system’s underlying rationale.

2.3 Understanding what a piece of code does

There is a vast literature on the difficulties, limitations, and methodologies for analysing source code. We don’t replicate that body of knowledge here, but instead summarise some of the key points on which this literature agrees.[5]

First, there are two ways of examining code. One is through static analysis, which consists of analysing the code without running it. The other is through dynamic analysis, which entails observing the code ‘in action’ by running it and analysing its outputs.

Second, the complexity of the code affects our ability to ascertain how it functions. The lines of code at the beginning of this section are easy to interpret. A simple, static analysis reveals how the inputs (the user’s age) are turned into outputs (a message that tells the users if they can vote in UK elections). If, however, we were to examine highly complex code, such as the code that a search engine uses to rank results, it would be a monumental task to ascertain how the code generates the search results that a user sees for a given query. Such an examination would necessitate a dynamic analysis and the results of even the most thorough analysis would have limitations. For some highly complex machine learning systems, a complete understanding is impossible to achieve.

Finally, although a complete understanding of a given piece of software may be difficult or impossible to achieve, having access to the source code can make a big difference. For any software, ranging from the simplest to the most complex, we are able to paint a much clearer and fuller picture of how it functions – or malfunctions – if we can analyse the underlying code. In many situations, source code analysis can be an indispensable component for developing a sufficiently detailed understanding of how the software functions.

2.4 The role of humans when things go wrong

In 2020, GCSE and A-level examinations were cancelled due to the Covid-19 pandemic. Ofqual, the organisation tasked with regulating qualifications, exams, and tests in England, decided to produce an algorithm that would compute an exam grade for each

5 Some of the prior literature on this topic is written for a general audience (e.g. CMA 2021), while other parts of it are written for an academic audience (e.g. Desai and Kroll 2018).

student. The intention was for the algorithm to ensure that qualification standards were maintained and that grade inflation was kept under control. Yet, when students received their algorithmically generated results, it started to become clear that the code was more likely to award higher grades to children from private schools and lower grades to children from state schools. Faced with a growing public backlash, the government decided against using the grades generated by the algorithm. As the dust started to settle on the exam grade fiasco, the prime minister visited Castle Rock High School in Coalville, Leicestershire. Addressing the assembled students, he said "I am afraid your grades were almost derailed by a mutant algorithm and I know how stressful that must have been" (Coughlan 2021).

As this quote illustrates, we have a tendency to think of code as having agency. We assign it human-like powers: we speak of algorithms that judge us, that lie to us, or that learn from us. When things go wrong, as they did last summer, we prefer to give code a life – and will – of its own: the algorithm went rogue or mutated, much like a living virus would when adapting to changing conditions.

Code, however, does not act on its own volition. It is written by humans,[6] according to specifications defined by humans. When things go wrong, our malicious intentions, errors, or limitations are often at play. In particular:

- **Software can be deliberately programmed to serve illicit purposes.** This is where malicious intentions are at play. Malware, for example, refers to code that is "specifically designed to disrupt, damage, or gain unauthorised access to a computer system."[7] Apart from malware, code can and has been used to evade existing standards, laws, and regulations. For example, engineers at Volkswagen and other car manufacturers specifically programmed code to manipulate the results of emissions tests.

- **Software can have unintended consequences.** This is where human errors and limitations come into play, despite our best intentions. We make mistakes when designing software, specifying the requirements, writing the code, and testing the performance of a programmed system. As the Therac-25 example shows, the errors that we make can have disastrous consequences. We also fail to foresee all the consequences or uses of a particular piece of code. Even when designed and developed in accordance with best practices, pricing algorithms that monitor and react to competitors' pricing strategies could, for example, lead to price collusion (Ezrachi and Stucke 2016). And well-intended software, like Microsoft's Twitter bot

6 Code can also be automatically generated now, but humans are very much involved there, too. It is humans that program how a system can generate code.

7 Source: Oxford Languages from Oxford University Press.

named Tay, can be sabotaged by Twitter users and turned into a sexist and racist chatbot – a far cry from Microsoft's original intention of having a friendly chatbot that would converse with millennials (Schwartz 2019).

The examples above shine light on our failings, as humans. We sometimes have malicious intentions. We make mistakes. We fail to foresee how a piece of code will behave when deployed – or how other human beings will interact with it. The code that we write and the software systems that we produce simply reflect these failings. As the prevalence of software increases, our ability to understand where humans made deliberate attempts to cause harm or unintended mistakes becomes ever more important. Source code disclosure can help build that understanding, making it an essential tool for the detection of foul play or negligence. This is one important reason, among others, why access to source code can be desirable in general, and why governments may choose to enact source code accessibility requirements in particular. We turn to a more comprehensive discussion of such reasons in the next section.

3 POSSIBLE SCENARIOS OF GOVERNMENT-MANDATED SOURCE CODE DISCLOSURE

The landscape of possible government-mandated requirements on technology providers to disclose source code can be surprisingly complex. The wide range of purposes that such requirements can serve and the fact that the disclosure itself can be made to a variety of recipients – inside or outside the government – contribute to that complexity. In this section, we consider the range of possible scenarios involving government-mandated disclosure of source code. The discussion is not meant to be exhaustive, but to provide an overview of the reasons for disclosure that have played a prominent role in policy discussions to date. These reasons can be structured around three main categories:

- meeting regulatory and judicial needs
- meeting procurement needs
- promoting innovation and economic development.

3.1 Meeting regulatory and judicial needs

Software is increasingly relevant to questions of regulatory compliance and lawfulness. The ever-expanding use of software in products and services means that countries' regulatory and judicial authorities face a growing number of situations where they need to determine whether software is consistent with regulatory or legal requirements. Conclusive assessments may require the analysis of source code and, even where they could in principle be achieved through other means, may prove cheaper and easier to complete by accessing the source code. Governments may therefore establish rules and schemes that ensure that relevant authorities and actors related to them are able to access source code for purposes of assessing compliance and lawfulness.

Questions of compliance and lawfulness in relation to software arise in virtually every domain. The examples below serve to illustrate just how widespread these questions and concerns are.

Competition law. The way in which a piece of software is programmed can be central to different types of competition law violations. As an example, software can be used in ways that – intentionally or unintentionally – amount to unfair treatment of competitors. Google, for instance, has been found to prioritise its own products and services over those of competitors when ranking search results.[8] Such anti-competitive practices are not uncommon. The European Commission and EU member states levied more than €20 billion in fines on Big Tech companies for anti-competitive practices between 2016 and 2019 (Scott and Larger 2019). When used for pricing purposes, software can also lead to collusive market outcomes, either purposefully (facilitation of explicit collusion) or as an unintended emergent result (tacit collusion) (Ezrachi and Stucke 2016, OECD 2017, CMA 2018).

Equality law. Software is increasingly used in the context of making decisions that affect individuals, be it in the private sector (e.g. algorithms used to inform hiring decisions, determine credit eligibility, or provide targeted pricing) or in the public sector (e.g. algorithms to inform policing practices, determine the allocation of health resources, identify children at risk of harm, or predict households likely to enter into fuel poverty). In such cases, the way in which the software is programmed can come into conflict with equality law requirements, including the avoidance of unlawful discrimination and, when it comes to software use in the public sector, the Public Sector Equality Duty.

Data protection law. Digital ways of collecting, curating, processing, sharing, and storing personal data invariably involve a variety of software solutions. Some of these solutions may have weaknesses that raise questions related to compliance with data protection legislation. Software can also be at play in the deliberate flouting of data protection requirements. For example, there is evidence to suggest that companies use cookies to track individuals' online habits and share that information with multiple advertising partners around the world in ways that violate the provisions of the European Union's General Data Protection Regulation (e.g. Libert 2015, ICO 2019, Murgia and Harlow 2019).

8 See the 2017 case by the European Commission against Google (https://ec.europa.eu/commission/presscorner/detail/en/IP_17_1784).

Product safety. The details of a given piece of software can be relevant from the perspective of product safety regulation, be it through software being embedded in physical products or software tools constituting products in their own right. In either case, software can be the cause of product safety violations. Software-related product safety issues can arise in any sector. We give two examples here to illustrate the forms that such issues can take:

- *Transport.* Software plays an increasingly important role in the operation of motor vehicles such as cars and lorries, passenger airplanes and trains, commercial ships, as well as the transport infrastructure itself. This includes the use of software to operate specific parts (e.g. a vehicle's brakes) or to automate the operation of entire systems (e.g. automated trains or driverless cars). As manufacturers increasingly rely on software to control devices, errors in the code or in the design of the code are bound to arise. In the most extreme cases, such as the two plane crashes caused by poorly designed flight control software aboard Boeing's 737 Max, software can lead to devastating loss of life.

- *Health.* Software is also increasingly used in the medical field. This includes software that is embedded in machines (e.g. the Therac-25 radiation therapy machine), used as part of operations (e.g. to triage patients in A&E), or deployed to support diagnostic decisions (e.g. as a software tool for reading X-rays or analysing CT scans). When used in any of these situations, inadequate code can harm or even threaten the lives of patients.

Integrity and stability of financial markets. In the financial sector, software is used in applications that can have implications for market integrity and financial stability. For example, software used for financial trading may have features that – intentionally or unintentionally – result in trading patterns that amount to insider trading or illicit forms of market manipulation. Similarly, trading software or software used for risk management can produce destabilising market dynamics (e.g. flash crashes) or the violation of capital requirements, either through errors or as a result of purposeful programming.

Environmental protection. In cases where source code is embedded within products with a regulated environmental impact, such as automobiles, software features may be relevant to questions of compliance with the relevant requirements. For example, software tweaks were used to manipulate the results of emissions tests by several car manufacturers.[9]

Gambling regulations. Software-related issues have been around for decades in the gambling industry. In the 1990s, for example, Ronald Harris, a software engineer for Nevada's Gaming Control Board, inserted fraudulent code into more than 30 slot machines. The code would trick the slot machines into triggering jackpots when users

9 In the past, many large automobile manufacturers have been fined for cheating on emissions tests by manipulating technology in their cars, including Volkswagen (in 2015), Fiat Chrysler Automobiles (2017), General Motors (2015), and Daimler (2019).

inserted coins in a specific order (Koeppel 2006). Although this is an example of a rogue employee intentionally inserting malicious code into a small number of slot machines, concerns related to more systemic use of software to rig gambling machines or applications abound.

Consumer protection. Several reports outline the challenges of consumer protection in the digital age (e.g. OECD 2019). As software is embedded in more and more products and services that we use in our lives and homes, it gives rise to concerns related to privacy, security, hidden deficiencies or aftermarket support. A well-known example of the interplay between consumer protection laws and software embedded in physical devices relates to Apple's iOS updates for iPhones. Towards the end of 2017, it became increasingly clear that the updates that Apple was pushing for iPhones limited the battery performance of older iPhone models. There have been multiple class-action lawsuits against Apple, accusing it of undermining the performance of its older products in order to encourage customers to purchase newer phones. The lawsuits against Apple are likely to be only the tip of the iceberg when it comes to software that might come in conflict with consumer protection law.

As these examples show, needs for source code analysis for regulatory or judicial purposes can arise in many different contexts. For each of the examples above, conclusive assessments of compliance and lawfulness may not be possible without access to source code or may be easier and cheaper to achieve by accessing the underlying code.

Governments and their agencies may require access to source code either before a product or service containing software enters the market or after it was sold and problems appeared. The first situation can be thought of as a need for *ex-ante* disclosure of source code – accessing source code for conformity assessments and regulatory approvals. These assessments and approvals can be a precondition for the sale, distribution, or use of certain types of software. The second situation can be thought of as a need for *ex-post* disclosure of source code – accessing source code as part of investigations, enforcement activities, and legal proceedings in response to incidents or suspected violations of regulatory or legal requirements. We discuss *ex-ante* and *ex-pos*t disclosure in greater detail below.

3.1.1 Ex-ante disclosure for regulatory purposes

Requirements of ex-ante regulatory approval or independent conformity assessments for certain types of products, services, projects, or processes are a well-established practice in many jurisdictions. Medical devices and vehicles, for example, commonly need to undergo independent assessments and obtain regulatory approval before being admitted to the market. Depending on the context, such assessments and approvals may be carried out by regulatory authorities themselves or may rely on delegation, for example in the form of certification schemes administered by accredited organisations.

Software may become subject to ex-ante regulatory approvals or conformity assessment requirements in one of two ways. First, software may be part of products, services or processes that are themselves subject to regulatory approvals or conformity requirements (e.g. cars, medical devices, or voting machines). Second, software can be a standalone product or tool that, in its own right, is subject to regulatory approvals or assessment requirements – for example, in the case of a software tool for interpreting CT scans or a model used by a financial institution to calculate capital requirements.

Insofar as regulatory approvals or conformity requirements touch on software features, there can be cases in which an adequate analysis of the software involved may be difficult or impossible to achieve without access to source code. At the time of writing, regulatory approval and conformity assessment procedures rarely involve the analysis of source code. However, the vehicle emissions scandals mentioned above or recent controversies about voting machines[10] provide vivid evidence of the limitations of this approach and illustrate why governments may decide to introduce requirements for source code disclosure to regulatory authorities or other entities conducting conformity assessments.

Beyond these present-day examples, the growing adoption of machine learning is set to increase the role of software disclosure in regulatory approval and conformity assessment procedures. Technical capabilities enabled by machine learning will lead to the introduction of software-based solutions in domains and sectors in which software has played a less prominent role in the past. Whether used in new or established areas, machine learning will contribute to software becoming increasingly complex.

The complexity of software based on machine learning approaches can make it harder to manage risks and can also lend itself to the concealment of the intentional violation of regulatory and legal requirements. Both of these factors are potential reasons for the introduction of regulatory approval and conformity assessment requirements in areas where such requirements have not existed in the past. Increases in software complexity can also mean that there is a growing need to analyse the software's source code in order to arrive at sufficiently conclusive results for regulatory approval and conformity assessment purposes: performance tests, the review of software specifications, and other assessment approaches that do not require source code access can be less conclusive as the complexity of software systems increases.

10 The software used in Dominion voting machines in the 2020 US presidential election was blamed incorrectly by Donald Trump for mistakes in vote counts coming from the states of Michigan and Georgia (Nicas 2020). Although the software was not deemed to be at fault in the end, important questions were raised about certifying voting machines and ensuring trust in the electoral process. Prior to the 2020 election, researchers uncovered vulnerabilities in online voting systems and apps. For example, Specter et al. (2020) conducted research on a mobile voting app used in the 2018 midterm elections in West Virginia, US which had vulnerabilities that made it possible to alter, stop, or expose a user's vote; Halderman and Teague (2015) conducted research related to an internet voting system used in the 2015 state election in New South Wales, Australia which had vulnerabilities that could be exploited to change votes, identify voters, or bypass the verification mechanism.

Recent years have seen a rapidly growing debate about regulatory questions that arise in the context of machine learning systems. Requirements for technology providers or users to participate in certification schemes or otherwise seek regulatory approval play a prominent role in these debates. While it is too early to judge their outcome, it is easily conceivable that these debates will lead to novel regulatory approval and conformity assessment requirements; and that some of those requirements will involve ex-ante disclosures of source code.

Countries are increasingly looking at regulatory solutions to ensure that software innovation based on machine learning approaches is responsible and safe. In the US, the Commodity Futures Trading Commission has contemplated rules that would require financial institutions to make source code used for high frequency trading generally accessible to the regulator.[11] Similarly, regulators in the UK are increasingly devoting resources to understanding how to regulate software based on machine learning approaches. Ambitious agendas and strong calls for greater transparency have been set out by the Information Commissioner's Office in publications such as *Explaining decisions made with Artificial Intelligence*[12] and *Guidance on AI and Data Protection*;[13] the Competition and Markets Authority in *Algorithms: How they can reduce competition and harm consumers*;[14] and Ofcom in *Regulating video-sharing platforms*.[15] Publications from other regulators will soon follow. The Financial Conduct Authority, for example, is collaborating with The Alan Turing Institute on a report regarding AI transparency in financial services. The Equality and Human Rights Commission are also working alongside The Alan Turing Institute and the Centre for Data Ethics and Innovation to produce guidance on compliance with the Equality Act 2010 when using complex software solutions.

In the UK, regulation of technologies based on machine learning is dynamic, fast moving, and ambitious. The UK has a tremendous opportunity to lead the global conversation on the regulation of these powerful technologies. As software-based technologies, part of that conversation will inevitably be around the need for ex-ante source code disclosure.

3.1.2 Ex-post disclosure for regulatory and judicial purposes

Setting aside the need for ex-ante source code disclosure as part of regulatory approvals or conformity assessments, source code disclosure may also be needed to enable authorities to determine relevant facts in response to incidents, suspected violations of regulatory and legal requirements, or other disputes involving software. As with ex-ante disclosure, access to source code will not always be necessary to determine the facts in question, but

11 In the end, these rules were not adopted.
12 https://ico.org.uk/for-organisations/guide-to-data-protection/key-data-protection-themes/explaining-decisions-made-with-artificial-intelligence/
13 https://ico.org.uk/for-organisations/guide-to-data-protection/key-data-protection-themes/guidance-on-ai-and-data-protection/
14 www.gov.uk/government/publications/algorithms-how-they-can-reduce-competition-and-harm-consumers/algorithms-how-they-can-reduce-competition-and-harm-consumers
15 www.ofcom.org.uk/__data/assets/pdf_file/0021/205167/regulating-vsp-guide.pdf

in some cases the relevant facts are difficult or impossible to establish without access to source code. The likelihood of this being the case will increase with the adoption of more complex software solutions.

Relevant facts that may need to be determined include questions of compliance and lawfulness across virtually all domains, as the examples at the beginning of this section illustrated. Such questions may arise in the context of investigations and enforcement activities carried out by regulatory bodies or in the context of legal proceedings. For example, source code analysis may be needed to determine whether:

- a given piece of software contributed to anti-competitive outcomes and whether it was deliberately programmed to do so;
- the use of a given decision-making algorithm is inconsistent with equality law requirements;
- personal data is processed and shared in ways that violate data protection law;
- accidents are attributable to failures to conform to product safety requirements;
- a financial trading algorithm is designed to pursue strategies that amount to unlawful market manipulation.

When it comes to legal proceedings, disputes may also concern facts that go beyond compliance and lawfulness. These include, for instance, questions such as determining whether an accident caused by a partly automated vehicle or some other software-reliant product is due, say, to a manufacturing defect or to inappropriate use. Source code can also be key evidence in intellectual property infringement cases.

In contrast to ex-ante source code disclosure requirements, which are relatively rare in the existing regulatory and legal landscape, ex-post disclosure requirements have clearly established precedents in many jurisdictions. Often, such requirements will have a basis in general regulatory or legal powers. It is worth noting, however, that there are also examples of dedicated legal provisions that give authorities specific powers to request the disclosure of source code in certain contexts. In the US, for example, the Internal Revenue Code explicitly provides for the possibility of authorities requesting access to the source code of tax software where they cannot otherwise reasonably ascertain the accuracy of an item on a return.[16]

Ex-post source code disclosure requirements can involve making the relevant code accessible to the appropriate regulatory, administrative, or judicial authorities or to other organisations or individuals involved in carrying out the needed assessments. The analysis of source code by independent experts played an important role, for example,

16 26 U.S. Code §7612 – Special procedures for summonses for computer software.

in Toyota's unintended acceleration lawsuits. *Bookout and Schwarz v. Toyota* is a case brought against the car manufacturer following a 2007 car accident that led to the death of the passenger and the serious injury of the driver. Faulty software caused the car to continue accelerating despite the driver's attempts to engage the hand and foot brakes. NASA was initially tasked with examining the source code of the software embedded in the 2005 Toyota Camri involved in the crash. NASA software engineers found 7,134 violations when they checked Toyota's source code against MISRA-C, a coding standard for software embedded in cars. Michael Barr, a software specialist, and Phillip Koopman, a professor in Electrical and Computer Engineering at Carnegie Mellon University, were subsequently tasked with reviewing Toyota's source code further. Their review uncovered 81,514 violations against the coding standard. Barr and Koopman's testimonies proved pivotal, convincing the jury not only of the fact that Toyota's software was defective, but also that the company acted in "reckless disregard of the rights" of the plaintiffs (Safety Research & Strategies 2013). Cases like this underline the importance of technical specialists and academic researchers having access to source code in order to determine relevant facts in response to incidents, suspected violations of regulatory and legal requirements, or other disputes involving software.

3.2 Meeting procurement needs

Government-mandated source code disclosure requirements can also serve the purpose of meeting information needs that arise in procurement contexts. When organisations procure software or software-driven products and services, the ability to access the software's source code can matter for a variety of objectives, including due diligence, transparency and accountability, or strategic aims. In light of the potential dependence of these objectives on source code accessibility, governments may decide to enact rules that require procurement contracts to include source code accessibility conditions or that provide a legal basis (where needed)[17] for procuring entities to give preference to contracts that include such conditions.

The most salient procurement context in which government-mandated source code disclosure requirements may exist relates to procuring goods and services for the public sector. Across jurisdictions, it is common practice for governments to set specific rules for public procurement. Such rules often require public bodies only to enter into – or to give preferential treatment to – procurement contracts that include certain conditions. When it comes to procurement that involves software, these conditions may touch on source code. In particular, conditions might include the sharing of the software's source code with the procuring public sector body and other government entities for purposes

17 For example, in the context of public sector procurement where such preferential treatment could otherwise be open to legal challenge.

of scrutiny; granting licences to share the source code publicly; or a more comprehensive assignment of intellectual property rights in the software and its source code to the procuring body.

Such conditions can be implemented through bespoke contractual provisions. Alternatively, they can be achieved through adherence to independently established default arrangements. A particularly prominent example of the latter approach relates to requirements or preferential treatment for technology that is provided on the basis of open source licences, which allow software to be freely used, modified, and shared.[18]

The UK government made a commitment in 2016 to have source code be open by default.[19] This commitment is reflected in the *Government Design Principles*,[20] where the 10th principle states:

> "We should share what we're doing whenever we can. With colleagues, with users, with the world. Share code, share designs, share ideas, share intentions, share failures."

The government's *Technology Code of Practice*[21] echoes that advice ("publish your code and use open source software to improve transparency, flexibility and accountability"), as do the recently published *Guidelines for AI procurement*[22] ("ensure your work is open and available to others for reuse"). Such commitments – and extensive guidance – highlight the relevance of open source licensing arrangements to recent public procurement policy debates.

When it comes to possible motivations for public procurement rules that require or give preference to contracts that allow source code access, the following purposes can be distinguished.

3.2.1 Due diligence

Access to source code can enable procuring bodies to perform their own assessment of a given piece of technology at a level that is sufficiently detailed for due diligence purposes. Relevant dimensions of assessment include questions of regulatory compliance and lawfulness across the various areas outlined at the beginning of this section, including requirements that apply specifically to the public sector, such as the Public Sector Equality Duty. They also include broader aspects of fitness for purpose, safety, and trustworthiness, which are important to any responsible procurement process and essential for procurement processes in the public sector.

18 https://opensource.org/osd
19 www.gov.uk/government/publications/open-source-guidance
20 www.gov.uk/guidance/government-design-principles
21 www.gov.uk/government/publications/technology-code-of-practice/
22 www.gov.uk/government/publications/guidelines-for-ai-procurement/guidelines-for-ai-procurement

One context with elevated requirements of scrutiny is the procurement of technology that is considered *critical national infrastructure*, with examples ranging from communication network equipment to technology used in elections.[23] Recent controversies about the use of Huawei kit in 5G networks in the US and the UK and allegations of election irregularities in the US have highlighted the particularly pronounced need to ensure the trustworthiness of software involved in operating such critical national infrastructure. Other contexts include technology that has national security implications, such as software used in the defence sector, and situations that involve the deployment of software within protected digital environments, which exist in many areas of the public sector. For example, software is sometimes built to analyse valuable, sensitive, or highly personal data which may be held in protected environments. The ability to scrutinise source code is essential to ensuring that the software used within these protected environments does not compromise data security.

When it comes to due diligence, open source licensing arrangements have a key advantage in that they entail the ability to publish source code, which means that it can be scrutinised by a wider audience, outside of the procuring entity. As the *Government Design Principles*[24] state when advocating for open source software, "the more eyes there are on a service the better it gets – howlers are spotted, better alternatives are pointed out, the bar is raised". Due diligence processes are enhanced as a result of making source code widely available.

3.2.2 Transparency and accountability

The accessibility of source code for software procured by public sector bodies can also be important in order to meet demands of transparency and accountability concerning the use of a given piece of software. Relevant demands can take two forms. First, beyond the need to ensure that software is compliant, fit for purpose, safe, and trustworthy, software users will often face a need to *demonstrate publicly* that the software used has these properties. Second, in cases where software is used for decision-making purposes, there can be a need to explain and justify these decisions (ICO 2020).

While these two types of needs are not limited to the use of software by public sector bodies, they can be particularly pronounced in this context, in part due to legal reasons. And both needs can have implications for the accessibility of source code that may go beyond those associated with due diligence needs. More specifically, public demonstrations of safety and trustworthiness may include making source code publicly available, regardless of whether there is a case for the publication of source code from a due diligence perspective; and an organisation's ability to explain and justify software-based decisions may require the analysis of source code even if such analysis is not deemed necessary for due diligence purposes.

23 www.dhs.gov/news/2017/01/06/statement-secretary-johnson-designation-election-infrastructure-critical
24 www.gov.uk/guidance/government-design-principles

3.2.3 Strategic considerations

Finally, source code accessibility requirements can be rooted in strategic considerations faced by organisations procuring technology that go beyond the purposes of due diligence or transparency and accountability. In particular, open source licensing arrangements or other contractual provisions that enable the procuring entity to modify, re-purpose, or share source code can have significant financial and technological benefits.

From a financial perspective, such arrangements can significantly reduce the cost of updating or improving a given piece of software by allowing for these activities to be carried out in-house or based on open tendering. It can also make it possible to re-deploy a given software tool in new contexts without additional cost. These benefits are particularly salient for the public sector, where value-for-money considerations play a prominent role in procurement processes.

In terms of technological benefits, such arrangements can reduce the likelihood of organisations being tied into particular types of operating infrastructure for the use of a given software tool. The public sector has a long history of procurement contracts that result in government departments and agencies being locked into long-term relationships with software providers. Source code accessibility requirements can help mitigate these long-standing issues.

***

In the UK, the procurement of software in the public sector is subject to numerous policies and extensive guidance. As software-based technologies evolve, these policies and guidelines are also changing and placing greater emphasis on the need for transparency. Recent documents such as the government's *Guidelines for AI procurement*[25] provide horizontal guidance for software solutions that rely on machine learning, while documents such as *A buyer's guide to AI in Health and Care*[26] provide sector-specific guidance. The need for transparency is underlined in all recent guidelines related to the public procurement of software-based products and services.

While our discussion here focuses on source code accessibility provisions in public-sector procurement rules, it is worth noting that governments may in principle also issue rules where such provisions apply to private-sector procurement practices. In particular, in response to due diligence and transparency/accountability needs, governments may decide to enact measures designed to require or prioritise source code accessibility for certain types of technology procurement in the private sector.

25 Available at www.gov.uk/government/publications/guidelines-for-ai-procurement/guidelines-for-ai-procurement
26 www.nhsx.nhs.uk/ai-lab/explore-all-resources/adopt-ai/a-buyers-guide-to-ai-in-health-and-care/

3.3 Promoting innovation and economic development

Finally, governments may introduce source code disclosure requirements that are intended to promote innovation and economic development. The availability of source code developed by a technology provider to other organisations can be an impactful mechanism to accelerate technology adoption, spur new inventions, ensure interoperability between technology solutions, and foster the growth of a country's industrial ecosystem.

Source code disclosure requirements motivated by these considerations can take various forms. They may involve source code being made available publicly or to specific organisations, with or without the permission to use or modify it. Relevant examples include, once again, measures that require or incentivise arrangements that make source code available on an open source basis, for example through rules for procurement contracts in the public sector or elsewhere. Another prominent example with particular relevance to trade agreements are measures specifically dedicated to technology transfer at the international level. For instance, governments may enact investment rules that require foreign technology developers investing in a given country to form joint ventures with domestic companies, entailing domestic ownership of the intellectual property for imported technology (e.g. through provisions in investment treaties). Considerations of technology transfer can be particularly relevant in the context of developing economies, with the possibility of trade agreement provisions including exceptions for countries depending on their level of economic development. Considerations of international technology transfer can also be relevant in other contexts, however. For example, governments may seek to require or encourage the transfer of technologies that are deemed essential to the mitigation of emergencies or crises.

4 SOURCE CODE DISCLOSURE IN RECENT TRADE AGREEMENTS

Trade agreements and negotiations have taken notice of the important role of source code in international commerce. As a result, several recent agreements have specific provisions related to source code disclosure. In this section, we introduce the legal mechanisms for protecting source code and note that the limitations of these mechanisms are pushing countries to introduce provisions related to source code disclosure in bilateral trade agreements. We then examine the provisions related to source code disclosure in seven recent trade agreements, highlighting some of their differences and limitations. We end the section by analysing the enforcement mechanisms in each of the seven agreements we examined.

4.1 Legal mechanisms for protecting source code

The legal mechanism that is most widely used to protect source code is trade secret law. Trade secrets, however, are not the only type of IP protection available to software creators. The expression of a programmer's ideas in source code can be protected through copyright law, for example, and truly innovative software can be protected through patents.

Yet, these alternative protections leave gaps. Copyright law, for example, protects the specific way in which a piece of code is written, but not the more general idea of how the software functions. Patent protections are limited in time, and often not available for all parts of a software-based system.[27] Machine learning approaches, for instance, might not be patentable where they are considered too abstract.[28] Moreover, patent applications generally require disclosure of the innovation in question, which might make the information disclosed ineligible for trade secret protection.[29] The limitations of copyright law, combined with the cost and uncertainty of patenting software, have meant that software producers and innovators rely mostly on trade secret protection rather than other IP protections available.

Article 39 of the WTO Agreement on Trade-Related Aspects of Intellectual Property Rights (TRIPS) offers protection for trade secrets.[30] The software code disclosure provisions included in recent bilateral agreements, however, go further than Article 39 of TRIPS. In particular – and of interest to us in this chapter – the provisions place limitations on the power of public authorities to mandate access to source code.

4.2 Provisions related to source code disclosure in recent trade agreements

In this section, we examine the source code disclosure provisions in the following agreements:[31]

- the US–Japan Digital Trade Agreement (US–Japan) (2019)[32]
- the United States–Mexico–Canada Agreement (USMCA) (2018)[33]
- the EU–Japan Economic Partnership Agreement (EU–Japan) (2018)[34]
- the EU–UK Trade and Cooperation Agreement (EU–UK) (2020)[35]
- the Comprehensive and Progressive Agreement for Trans-Pacific Partnership (CPTPP) (2018)[36]

27 *Alice Corp. v. CLS Bank International*, 573 U.S. 208 (2014)
28 Article 52, European Patent Convention, and Guidelines for Examination by the European Patent Office (www.epo.org/law-practice/legal-texts/html/guidelines/e/g_ii_3_3_1.htm).
29 Despite transparency requirements seemingly putting patents and trade secrets at odds, there are legal avenues for innovators to maintain the secrecy of their inventions during the process of patent applications (e.g. McGurk and Lu 2015).
30 www.wto.org/english/docs_e/legal_e/27-trips.pdf
31 We selected these agreements based on (1) the wide coverage they provide of negotiating parties and their respective interests; and (2) the fact that they allow us to see how provisions have evolved over time. The years in parentheses represent the year when each agreement was signed.
32 https://ustr.gov/sites/default/files/files/agreements/japan/Agreement_between_the_United_States_and_Japan_concerning_Digital_Trade.pdf
33 https://ustr.gov/trade-agreements/free-trade-agreements/united-states-mexico-canada-agreement/agreement-between
34 https://trade.ec.europa.eu/doclib/press/index.cfm?id=1684
35 https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/948119/EU-UK_Trade_and_Cooperation_Agreement_24.12.2020.pdf
36 www.dfat.gov.au/sites/default/files/tpp-11-treaty-text.pdf; to be read in conjunction with the text of the Trans-Pacific Partnership (TPP) (www.dfat.gov.au/trade/agreements/not-yet-in-force/tpp/Pages/tpp-text-and-associated-documents).

- the Indonesia–Australia Comprehensive Economic Partnership Agreement (IA-CEPA) (2020)[37]
- the Japan–Mongolia Economic Partnership Agreement (Japan-Mongolia) (2015).[38]

The agreements above establish a general prohibition on public authorities requiring the transfer of, or access to, source code of software owned by a person of the other Party (also referred to as source code disclosure in this document). The prohibition aims to encourage trade by banning source code disclosure mandates as a condition for market access.

The agreements allow for a number of exceptions. A holistic reading of each agreement is necessary to develop a full picture of the exceptions that relate to the source code provision. First, the articles related to source code in each agreement contain some safeguarding exceptions. Second, each agreement includes exceptions that apply to the scope of the treaty's various provisions taken together which, by implication, also apply to the provisions contained in the article on source code disclosure. Both types of exceptions provide potential bases for the parties to the agreements to require the disclosure of source code under certain conditions. Typically, the burden of proof to show that a given disclosure requirement is justified pursuant to one of the exceptions falls on the party that is enacting the requirement.

The exceptions in the trade agreements that we examined build on earlier exceptions in WTO agreements. Some of the agreements we looked at incorporate the well-known Articles XX GATT 1994 & Article XIV GATS (general exceptions for public interest measures) or Article XXI GATT & Article XIV bis GATS (national security exception). In general, these provisions provide a basis for justifying government conduct that goes against general treaty obligations, including the general prohibition on source code disclosure requirements, by requiring that:

1. the government conduct pursues a legitimate public interest;
2. the measures taken are either necessary or relevant to the achievement of this public interest (depending on the provision);
3. the measures implemented in order to pursue the goal do not constitute arbitrary or unjustifiable discrimination or a disguised restriction on international trade.

Table 1 provides a summary of the legitimate public goals that the trade agreements we examined recognise as justifying government measures that diverge from treaty provisions.[39] Since the focus of this chapter is source code disclosure, the table concentrates

37 www.dfat.gov.au/trade/agreements/in-force/iacepa/iacepa-text/Pages/default
38 www.mofa.go.jp/files/000067716.pdf
39 The exact wording of the articles in each trade agreement that specifically refer to source code is available in the Appendix.

on the legitimate public purposes that are explicitly mentioned in the agreements and that have a clear relationship to the disclosure of source code embedded in software services or products using software.

TABLE 1 EXPLICITLY STATED EXCEPTIONS TO THE GENERAL PROHIBITION ON SOURCE CODE DISCLOSURE REQUIREMENTS[40]

	US-Japan[1]	USMCA[2]	EU-Japan[3]	EU-UK[4]	CPTPP[5]	IA-CEPA[6]	Japan-Mongolia[7]
Disclosure to a public or judicial authority during specific targeted investigations, inspections, or proceedings	X*	X*		X			
General disclosure requirements necessary to secure compliance with/ enforcement of laws or regulations, explicitly including:	X[8]	X[9]	X	X	X**	X**	X[10]
Competition law enforcement			X	X			
IP protection and enforcement			X	X[11]	X*	X*	X[12]
Tax law and customs law enforcement	X[13]	X[14]	X	X[15]			X[16]
Prevention of deceptive practices or deal with effects of default	X[17]	X[18]	X[19]	X[20]			X[21]
Privacy and data protection	X[22]	X[23]	X[24]	X[25]			X[26]
Safety	X[27]	X[28]	X[29]	X*			X[30]
Protect public morals, order or safety	X[31]	X[32]	X[33]	X[34]			X[35]
Protect human, animal or plant life or health	X[36]	X[37]	X[38]	X[39]			X[40]
Financial system integrity and stability	X[41]	X[42]***	X[43]	X[44]			

40 An in-depth reading of the text of each of these treaties, as well as other relevant documents, may be necessary to provide greater clarity on whether the treaties can be interpreted in a manner to justify source code disclosure in a particular case. Another preliminary note to make is the fact that financial and investment services are treated separately from electronic commerce in many of these agreements. The row on 'financial stability and integrity' includes a mark (X) only where there is an explicit exception to the general prohibition on source code disclosure for the purpose of financial stability and integrity. Footnotes allow the reader to consult the original source in the legal texts. Unless otherwise indicated through footnotes, the relevant provisions can be found in the articles referred to in the top row of the table.

	US-Japan[1]	USMCA[2]	EU-Japan[3]	EU-UK[4]	CPTPP[5]	IA-CEPA[6]	Japan-Mongolia[7]
Limit prohibition of source code disclosure requirements to mass market software or products using such software and excluding critical infrastructure					X	X	X
Government procurement		X[45]	X[46]	X[47]	X[48]	X[49]	
Essential security interests / national security	X[50]	X[51]	X[52]	X[53]		X	X[54]
Reassurance disclosure does not relate to proprietary or confidential information held, collected, processed by public authorities		X[55]	X[56]****	X[57]****	X[58]		
Commercially negotiated contracts			X	X	X	X	
Voluntary transfer of source code			X	X			

Notes: * Subject to appropriate safeguards to prevent unauthorised disclosure of source code. ** Requiring modification of source code for the purposes of compliance with the law rather than requiring source code disclosure. *** Only restrictive measures related to payments or transfers are included in this exception. **** No disclosure of information that is contrary to the essential security interests of a party to the agreement can be mandated. [1] Article 17, US-Japan. [2] Article 19.16, USMCA. [3] Article 8.73, EU-Japan, referring to Article III Government Procurement Agreement and Articles 1.5, 8.3 and 8.65 of EU-Japan. [4] Article DIGIT.12, EU-UK, referring to Article DIGIT.4 of EU-UK and Article III Government Procurement Agreement as incorporated by Article PPROC.2 of EU-UK. [5] Article 1, CTPP referring to Article 14.17, Trans-Pacific Partnership Agreement. [6] Article 13.13, IA-CEPA. [7] Article 9.11, Japan-Mongolia. [8] Article 3, US-Japan, referring to Article XIV GATS. [9] Article 32.1.2, USMCA, referring to Article XIV GATS. [10] Article 1.10.1, Japan-Mongolia, referring to Article XIV GATS and Article XX GATT. [11] Article DIGIT.12, EU-UK. In the context of public procurement, Article PPROC.2 of EU-UK referring to ANNEX PPROC-1 of EU-UK, Section A, referring to Article III Government Procurement Agreement. [12] Article 1.10.1, Japan-Mongolia, referring to Article XX(d) GATT, reinforced by Article 12.1.1, Japan-Mongolia. [13] Article 3, US-Japan, referring to Article XIV GATS. [14] Article 32.1.2, USMCA, referring to Article XIV GATS. [15] Article EXC.1.1, EU-UK, referring to Article XX GATT. [16] Article 1.10.1, Japan-Mongolia, referring to Article XIV GATS and Article XX GATT. [17] Article 3, US-Japan, referring to Article XIV GATS. [18] Article 32.1.2, USMCA, referring to Article XIV GATS. [19] Article 8.3.2(c)(i), EU-Japan. [20] Article EXC.1.1, EU-UK, referring to Article XX GATT; Article EXC.1.2.(c)(i), EU-UK. [21] Article 1.10.1, Japan-Mongolia, referring to Article XIV GATS and Article XX GATT. [22] Article 3, US-Japan, referring to Article XIV GATS. [23] Article 32.1.2, USMCA, referring to Article XIV GATS. [24] Article 8.3.2(c)(ii), EU-Japan. [25] Article EXC.1.2(c)(ii), EU-UK. [26] Article 1.10.1, Japan-Mongolia, referring to Article XIV GATS. [27] Article 3, US-Japan, referring to Article XIV GATS. [28] Article 32.1.2, USMCA, referring to Article XIV GATS. [29] Article 8.3.2(c)(iii), EU-Japan. [30] Article 1.10.1, Japan-Mongolia, referring to Article XIV GATS. [31] Article 3, US-Japan, referring to Article XIV GATS. [32] Article 32.1.2, USMCA, referring to Article XIV GATS. [33] Article 8.73, EU-Japan, but also Article 8.1.2 of EU-Japan, Article 8.3.2(a) and Article 8.3.1 of EU-Japan referring to Article XX GATT; Article 8.73.2(c) of EU-Japan referring to Article III Government Procurement Agreement. [34] Article EXC.1.1, EU-UK, referring to Article XX GATT; Article EXC.1.2(a) of EU-UK; Article EXC.1.2.(c)(iii) of EU-UK; In the context of public procurement – Article PPROC.2 of EU-UK referring to ANNEX PPROC-1 of EU-UK, Section A, referring to Article III Government Procurement Agreement. [35] Article 1.10.1, Japan-Mongolia, referring to Article XIV GATS and Article XX GATT. [36] Article 3, US-Japan, referring to Article XIV GATS. [37] Article 32.1.2, USMCA, referring to Article XIV GATS. [38] Article 8.3.2(b), EU-Japan; Article 8.73.2(c) of EU-Japan referring to Article III GPA. [39] Article EXC.1.1, EU-UK, referring to Article XX GATT; Article EXC.1.2(b) of EU-UK; in the context of public procurement - Article PPROC.2 of EU-UK referring to ANNEX PPROC-1 of EU-UK, Section A, referring to Article III Government Procurement Agreement. [40] Article 1.10.1, Japan-Mongolia, referring to Article XIV GATS and Article XX GATT. [41] Article 5, US-Japan. [42] Article 32.4, USMCA. [43] Article 8.65, EU-Japan. [44] Article SERVIN 5.39, EU-UK. [45] Article 19.2.3(a), USMCA. [46] Article 8.73.1, EU-Japan, but also Article 8.73.2(c) of EU-Japan referring to Article III Government Procurement Agreement. [47] A set of justifications (incorporated into the table), found in Article PPROC.2 of EU-UK referring to ANNEX PPROC-1 of EU-UK, Section A, referring to Article III Government Procurement Agreement. [48] Article 1, CPTPP, referring to Article 14.2, Trans-Pacific Partnership Agreement. [49] Article 13.2.3, IA-CEPA. [50] Article 4(b), US-Japan. [51] Article 32.2.1, USMCA. [52] Article 8.73.2(c), EU-Japan, referring to Article III Government Procurement Agreement. [53] Article EXC.4(b)(i), EU-UK; Article PPROC.2 of EU-UK referring to ANNEX PPROC-1 of EU-UK, Section A, referring to Article III Government Procurement Agreement. [54] Article 1.10.1, Japan-Mongolia, referring to Article XIV bis GATS. [55] Article 19.2.3 (b), USMCA, with the exception of provisions relating to open government data. [56] Article 1.5.1(a), EU-Japan. [57] Article EXC.4(a), EU-UK. [58] Article 13.2.4, IA-CEPA.

As the table illustrates, the agreements we examined exhibit significant overlap in terms of their explicitly mentioned potential justifications for allowing governments and their agencies to require access to source code. They have a tendency to converge around similar sets of commonly accepted public interests, values, or circumstances that justify the disclosure of source code. At the same time, there are notable conceptual differences. For example, there is a shared recognition that source code disclosure may be required to ensure compliance and enforcement of laws, but differences when it comes to the particular legal or public interests explicitly recognised by individual agreements. There are also differences in terms of requirements to balance disclosure interests against other interests under the exceptions that agreements provide for – for example, the above-mentioned requirements (2) and (3) set out in Articles XIV GATS or Article XX GATT to demonstrate necessity or relevance and for measures to take forms that do not constitute arbitrary discrimination or a disguised restriction on international trade.

The exceptions contained in individual agreements resonate, to different degrees, with the range of possible reasons for government-mandated source code disclosure requirements discussed in Section 3. For example, source code disclosure for the purpose of compliance with or enforcement of laws and regulations is enshrined in many of the trade agreements we examined. Similarly, the agreements often exempt government procurement from the scope of the general prohibition on requiring source code disclosure and carve out some exceptions for the disclosure of source code in the interest of governmental interests such as the protection of national security or critical infrastructure.

In many agreements, however, the exceptions to the general prohibition on requiring access to source code are comparatively narrow, and none of the agreements that we examined covers all of the scenarios outlined in Section 3. While this finding might not come as a surprise – carving out exceptions for all relevant scenarios is not trivial – we find it worrisome that the exceptions in some agreements cover very few of the scenarios outlined in Section 3. Building future-proof provisions for the disclosure of source code in trade agreements is a challenging task. The starting point, however, must be what we know about software-based technologies today and the various scenarios that they give rise to, as outlined in Section 3. We recommend that trade negotiators give thorough consideration to all of these scenarios in formulating exceptions to general prohibitions on requiring access to source code.

For the rest of this section, we outline consequential differences in the wording of the articles relating to source code, as they appear in the agreements that we examined. Where relevant, we also provide recommendations for future negotiations.

4.2.1 The challenge of defining regulatory and judicial needs

The Japan–Mongolia Economic Partnership Agreement was the first to include an article related to source code. The article itself, 9.11, only included a single exception: it noted that "for the purposes of this Article, software [...] is limited to mass-market software or products containing such software, and does not include software used for critical infrastructure".

The agreements that followed the Japan–Mongolia Economic Partnership Agreement expanded the list of exceptions incorporated into the articles related to source code. The Comprehensive and Progressive Agreement for Trans-Pacific Partnership added wording to specify that Article 14.17, which relates to source code, "shall not be construed to affect requirements that relate to patent applications or granted patents, including any orders made by a judicial authority in relation to patent disputes". The EU–Japan Economic Partnership Agreement went much further, broadening the list of exceptions included in Article 8.73, which refers to source code, to competition law, intellectual property rights, essential security interests, and other legitimate public interests.

The fast-changing nature of software-based services and products – along with their widespread impacts on societies and economies – make it impossible to draft a complete and future-proof list of all areas of compliance and lawfulness where source code disclosure may be needed. Recognising this difficulty, the United States–Mexico–Canada Agreement and the US–Japan Digital Trade Agreement took a different approach from their predecessors. The two agreements note that their respective articles related to source code (19.16 in USMCA and 17 in US–Japan) do not "preclude a regulatory body or judicial authority of a Party from requiring a person of the other Party to preserve and make available the source code of software [...] for a specific investigation, inspection, examination, enforcement action, or judicial proceeding, subject to safeguards against unauthorized disclosure". This language provides a partial solution to the issue of having to enumerate all possible situations when source code disclosure might be required for regulatory and judicial needs.

The approach taken in Article 19.16 of USMCA and Article 17 of US–Japan is a promising way of getting around the challenging task of enumerating all possible situations when source code disclosure might be required. Should similar approaches become the norm in future trade agreements, we recommend that the wording in the source code articles allows for a wider range of actors to require access to source code. Organisations other than regulatory bodies or judicial authorities might have an interest in accessing source code, for example, for the purpose of conformity assessments.

4.2.2 Protection of source code vs. algorithms

The articles related to source code in the agreements that we examined establish a general prohibition on public authorities requiring "the transfer of, or access to, source code of software". The United States–Mexico–Canada Agreement and the US–Japan Digital Trade Agreement are, once again, notable exceptions here, as they extend the general prohibition on disclosure requirements to algorithms. In particular, Article 19.16 in USMCA and Article 17 in US–Japan ban mandatory transfers and access to "the source code of software [...] or an algorithm expressed in that source code".

As we saw in Section 2, algorithms are pieces of code that contain a series of steps that need to be followed in order to solve a computational problem. As such, the lines of code that specify how an algorithm functions are covered under the general prohibition on requiring "the transfer of, or access to, source code of software". It is unclear what additional protections, besides the lines of code themselves, the wording in Article 19.16 of the USMCA and Article 17 in US–Japan provides to algorithms. However, the existence of that wording opens the door to arguing for protections whose scope extends beyond the lines of code themselves and include, for example, more high-level descriptions of the operating logic of a given piece of software.

If the exceptions that an agreement provides for are unduly narrow, this kind of expanded scope of the general prohibition on disclosure requirements can aggravate the resulting implications in problematic ways. While often insufficient to develop a reliable understanding of a given piece of software, the analysis of general descriptions of algorithmic logic can be helpful in partly addressing the needs identified in Section 3. Compared to provisions that only preclude the accessibility of source code to meet such needs, provisions that additionally also preclude the accessibility of general descriptions of software logic should therefore be interpreted as entailing more severe undue constraints. Given the difficulties of ensuring that the exceptions specified in a proposed agreement are sufficiently comprehensive and future-proof, negotiators should exercise caution in expanding the scope of the general prohibition to include algorithms.

4.2.3 Access to versus modification of source code to secure compliance with the law

The safeguarding exceptions incorporated within the articles related to source code are generally couched in terms of *access* to source code. The Comprehensive and Progressive Agreement for Trans-Pacific Partnership and the Indonesia–Australia Comprehensive Economic Partnership Agreement are the only agreements among the ones that we examined to diverge from this. To secure compliance with law and regulations, the CPTPP and IA–CEPA agreements provide for the possibility to require the *modification* of source code but not to require access to it. This severely limits the ability of national authorities to ensure compliance with or enforcement of laws and regulations. In future trade negotiations, we strongly recommend against the approach taken by the CPTPP and IA–CEPA agreements. The possibility of requiring access to source code must be secured for the purpose of compliance with or enforcement of laws and regulations.

4.2.4 Voluntary or commercially negotiated transfer of source code

Among the agreements we examined, two of them explicitly state that voluntary source code disclosure is unaffected by the general prohibition (the EU–Japan Economic Partnership Agreement and the EU–UK Trade and Cooperation Agreement). Four of the agreements we examined explicitly acknowledge that the general prohibition does not affect source code disclosure as a result of commercially negotiated contracts. Yet, although the other trade agreements do not explicitly mention the voluntary or commercially negotiated transfer of source code, this does not mean that they prohibit such transfers.

The legal principle of freedom of contract, in particular, allows private parties to agree on the terms and conditions of their interaction in a legally enforceable document as long as they do not contradict existing laws. The trade agreements that do not explicitly recognise voluntary or commercially negotiated source code disclosure do not prohibit such conduct. However, the agreements that explicitly acknowledge the legality of voluntary or commercially negotiated source code disclosure provide an additional layer of transparency and clarity in this regard. We recommend that future trade agreements provide this clarity, especially for voluntary source code disclosure. Such clarity may be particularly valuable in encouraging practices like open source or open and reproducible science.

4.3 Enforceability of provisions relevant to source code disclosure

Enforcement is a key consideration when it comes to trade agreements. Without effective enforcement mechanisms in place, countries might not have an incentive to abide by the rights and obligations stipulated in international legal texts.

International trade disputes take place between legally equal sovereigns – be it nation states or trade blocs with a jurisdiction over certain territories. Because of this, dispute settlement procedures often include a diplomatic process of mutual consultation before any legally binding dispute settlement proceedings commence. This seeks to provide a forum for the parties to reach a mutually agreeable solution. The WTO Dispute Settlement Understanding similarly provides for this diplomatic step of consultation between the parties before turning to the Dispute Settlement Body for a binding decision.[41] In some trade agreements, parties are also invited to explore alternative methods of dispute resolution, such as good offices, conciliation, or mediation, although there is no obligation to do so.[42] This manner of dispute settlement – inviting parties to consultation and exploring alternative dispute resolution methods before establishing panels or tribunals – reflects the steps in the WTO's Dispute Settlement Understanding.[43]

If the parties do not reach a mutually agreed solution through consultations, they can request the establishment of a panel or arbitration tribunal, created for the purposes of the particular trade dispute. These dispute settlement tribunals or panels conclude their work with a decision or a report which is binding on the parties of the agreement.

41 Article 4, Dispute Settlement Understanding.
42 See, for example, Article 31.5 of USMCA.
43 For an overview of the process of the typical WTO dispute settlement, see https://www.wto.org/english/tratop_e/dispu_e/disp_settlement_cbt_e/c6s1p1_e.htm.

Obligations spelled out in trade agreements range from requiring the responding party to "whenever possible, eliminate the non-conformity"[44] to taking "the necessary measures to comply immediately with the ruling".[45]

States are subject to multiple trade agreements. Many of the agreements that we examined account for this fact. They recognise that an action by a state can be an alleged breach of more than one trade agreement to which that state is a party. In such circumstances, the agreements provide for the possibility of countries choosing which dispute settlement forum to use in order to address their grievance. This highlights that there may be more than one pathway for requesting an impartial decision on matters of compliance with the text of the trade agreements.

With the exception of the US–Japan Digital Trade Agreement, all of the agreements we examined include agreement-specific enforcement mechanisms. In the case of the US–Japan Digital Trade Agreement, there are still potential avenues for seeking to resolve disputes between the parties through appeal to an independent authority. Notably, the International Court of Justice (ICJ), within the United Nations system, is available to adjudicate on disputes between any states party to its Statute which willingly submit a case to it. According to Article 93(1) of the Charter of the United Nations, all UN member states are ipso facto parties to the Statute. Cases before the ICJ can fall within the scope of any international treaty,[46] which would include trade agreements such as these discussed in this chapter.

Table 2 summarises the procedural steps of the dispute settlement procedures in the trade agreements we examined. As the table illustrates, all but one of the trade agreements we examined provide for meaningful pathways to independent assessment and enforcement of rights and obligations arising out of the agreements. In all cases where there is a dispute resolution mechanism detailed in the agreement itself, the source code non-disclosure provision and relevant exceptions fall within the scope of this mechanism. Where no such mechanism is provided by the agreement, general fora like the International Court of Justice remain available to the parties to uphold the treaties.

44 Article 28.19.2 of TPP Agreement.
45 Article INST.21.1 of EU-UK Trade and Cooperation Agreement.
46 Articles 34-36, Statute of the International Court of Justice.

TABLE 2 DISPUTE SETTLEMENT PROCESSES RELEVANT TO THE GENERAL PROHIBITION OF REQUIRING SOURCE CODE DISCLOSURE AND RELEVANT EXCEPTIONS

	US-Japan	USMCA[1]	EU-Japan[2]	EU-UK[3]	CPTPP[4]	IA-CEPA[5]	Japan-Mongolia[6]
Parties must have good faith consultations prior to legally binding dispute settlement proceedings		X[7]	X[8]	X[9]	X[10]	X[11]	X[12]
Parties can choose alternative dispute resolution methods, like good offices, conciliation, mediation		X[13]	X[14]		X[15]	X[16]	X[17]
Parties can choose the dispute settlement forum if the subject matter is a breach of multiple agreements that both states are a party to		X[18]	X[19]	X[20]	X[21]	X[22]	X[23]
Binding decision by an ad hoc arbitration tribunal or panel		X[24]*	X[25]*	X[26]*	X[27]*	X[28]	X[29]
Complaining Party is allowed to suspend treaty obligations proportionately in case of the Responding Party's non-compliance with decision		X[30]	X[31]	X[32]	X[33]	X[34]	x[35]
No formal dispute resolution mechanism specified	X						

Notes: * Arbitrators or panellists are to be chosen from a pre-agreed list of qualified individuals. Some agreements provide for an exception where no qualified individuals are on the lists. [1] Chapter 31, USMCA. [2] Chapter 21, EU-Japan. [3] Article INST.10 on the scope of the Dispute settlement mechanism under EU-UK. [4] Chapter 28, TPP. [5] Chapter 20, IA-CEPA. [6] Chapter 16, Japan-Mongolia. [7] Article 31.4, USMCA. [8] Article 21.5, EU-Japan. [9] Article INST.13.1, EU-UK. [10] Article 28.5, TPP. [11] Article 20.5, IA-CEPA. [12] Articles 16.2 and 16.4, Japan-Mongolia. [13] Article 31.5, USMCA and Article 31.22, USMCA. [1 4]Article 21.6, EU-Japan. [15] Article 28.6, TPP. [16] Article 20.6, IA-CEPA. [17] Article 16.5, Japan-Mongolia. [18] Article 31.3, USMCA. [19] Article 21.27, EU-Japan. [20] Article INST.12.1, EU-UK. [21] Article 28.4, TPP. [22] Article 20.4, IA-CEPA. [23] Article 16.3, Japan-Mongolia. [24] Article 31.6, USMCA, on the establishment of a panel and Article 31.8, USMCA, on the pre-agreed roster of panelists to choose from. Article 31.9.3, USMCA, on the possibility to exceptionally nominate a panelist who is not from the roster. Articles 31.18, USMCA, on the impact of the panel's report. [25] Article 21.7 and Article 21.9, EU-Japan, on pre-established list of arbitrators; Article 21.15.8, EU-Japan, on the binding nature of the decision of the panel; Article 21.20, EU-Japan, on compliance with the final report of the panel. [26] Articles INST.14 and INST.15, EU-UK, on the establishment of the arbitration tribunal; Articles INST.27 and INST.28, EU-UK, on the pre-agreed arbitrators to choose from; Article INST.29.1, EU-UK, on the binding nature of tribunal decisions. [27] Article 28.7, TPP, on establishing the panel; Article 28.11, TPP, on the roster of panelists; Article 28.19, TPP, on the binding nature of the panel report. [28] Article 20.8, IA-CEPA, on establishing the panel; Article 20.12.1, IA-CEPA, on the binding nature of the panel's findings. [29] Article 16.6, Japan-Mongolia, on establishing an arbitration tribunal; Article 16.11.1, Japan-Mongolia, on the requirement to comply with the arbitral decision. [30] Article 31.19, USMCA. [31] Article 21.22, EU-Japan. [32] Article INST.24, EU-UK. [33] Article 28.20, TPP. [34] Article 20.14, IA-CEPA. [35] Article 16.11.4, Japan-Mongolia.

5 CONCLUSION

Software-driven technologies, made all the more powerful by machine learning approaches, have become a transformative social, political, and economic force. As they continue to improve and grow, they give rise to unprecedented opportunities but also to novel challenges for policymakers.

Trade negotiators, in particular, face the difficult task of establishing provisions that balance a multitude of competing interests, ranging from commercial interests, to national economic and security interests, and to the fundamental rights and freedoms of individuals. Compared to other issues covered by trade agreements, finding the right balance between these interests when it comes to the treatment of source code disclosure is particularly challenging. This is because of the rapidly evolving and partly unpredictable nature of software-based innovation, its increased importance in international commerce, and the multitude of reasons why governments and their agencies might require access to source code.

This chapter provides a primer on the issue of source code disclosure in the context of trade negotiations. We covered what source code is, the main programming approaches, and the crucial role of humans when things go wrong. We outlined the possible motivations for government-mandated source code disclosure requirements and the forms that such requirements can take. Finally, we examined how a number of recent trade agreements address the issue of source code disclosure, identifying along the way some consequential differences in the wording of articles relating to source code.

Despite the ground that we covered here, more work remains to be done. We focused on the question of when disclosure of source code may be needed. How the disclosure needs are best translated into legal text as well as how the disclosure itself should be managed in practice will require additional research and deliberation. Furthermore, in this chapter we primarily questioned whether the current provisions related to source code can be improved. We did not, however, ask the important question of whether a general prohibition on requiring the transfer of, and access to, source code is the best way to handle the protection of software innovations in trade agreements. Further research is needed to ascertain that this is the best path to take. Finally, looming large over any trade agreement are questions related to the economic impact of the provisions. At the time of writing, no research exists on the effect that provisions related to source code disclosure have on international trade.

Software is not only here to stay, but will play an ever-bigger role in our lives and businesses. Trade agreements are only one area where the fine balance between enabling innovation and mitigating risks needs to be found. In finding that balance, UK trade negotiators must ensure that trade agreements complement the country's ambitious regulatory and economic agenda in pursuit of responsible software-based innovation.

APPENDIX: ARTICLES FROM INTERNATIONAL TRADE AGREEMENTS RELATING TO SOURCE CODE

US-Japan Digital Trade Agreement

Article 17: Source code

1. Neither Party shall require the transfer of, or access to, source code of software owned by a person of the other Party, or the transfer of, or access to, an algorithm expressed in that source code, as a condition for the import, distribution, sale, or use of that software, or of products containing that software, in its territory.
2. This Article does not preclude a regulatory body or judicial authority of a Party from requiring a person of the other Party to preserve and make available[47] the source code of software, or an algorithm expressed in that source code, for a specific investigation, inspection, examination, enforcement action, or judicial proceeding, subject to safeguards against unauthorized disclosure.

United States-Mexico-Canada Agreement

Article 19.16: Source code

1. No Party shall require the transfer of, or access to, a source code of software owned by a person of another Party, or to an algorithm expressed in that source code, as a condition for the import, distribution, sale or use of that software, or of products containing that software, in its territory.
2. This Article does not preclude a regulatory body or judicial authority of a Party from requiring a person of another Party to preserve and make available the source code of software, or an algorithm expressed in that source code, to the regulatory body for a specific investigation, inspection, examination, enforcement action, or judicial proceeding,[48] subject to safeguards against unauthorized disclosure.

47 This making available shall not be construed to negatively affect the software source code's status as a trade secret, if such status is claimed by the trade secret owner.

48 This disclosure shall not be construed to negatively affect the software source code's status as a trade secret, if such status is claimed by the trade secret owner.

EU-Japan Economic Partnership Agreement

Article 8.73: Source code

1. A Party may not require the transfer of, or access to, source code of software owned by a person of the other Party.[49] Nothing in this paragraph shall prevent the inclusion or implementation of terms and conditions related to the transfer of or granting of access to source code in commercially negotiated contracts, or the voluntary transfer of or granting of access to source code for instance in the context of government procurement.

2. Nothing in this Article shall affect:

   a. requirements by a court, administrative tribunal or competition authority to remedy a violation of competition law;
   b. requirements by a court, administrative tribunal or administrative authority with respect to the protection and enforcement of intellectual property rights to the extent that source codes are protected by those rights; and
   c. the right of a Party to take measures in accordance with Article III of the GPA.

3. For greater certainty, nothing in this Article shall prevent a Party from adopting or maintaining measures[50] which are inconsistent with paragraph 1, in accordance with Articles 1.5[51], 8.3[52] and 8.65[53].

EU-UK Trade and Cooperation Agreement

Article DIGIT.12: Transfer of or access to source code

1. A Party shall not require the transfer of, or access to, the source code of software owned by a natural or legal person of the other Party.

2. For greater certainty:

   a. the general exceptions, security exceptions and prudential carve-out referred to in Article DIGIT.4 [Exceptions] apply to measures of a Party adopted or maintained in the context of a certification procedure; and
   b. paragraph 1 of this Article does not apply to the voluntary transfer of, or granting of access to, source code on a commercial basis by a natural or legal person of the other Party, such as in the context of a public procurement transaction or a freely negotiated contract.

49 For greater certainty, "source code of software owned by a person of the other Party" includes source code of software contained in a product.
50 Those measures include measures to ensure security and safety, for instance in the context of a certification procedure.
51 Security exceptions
52 General exceptions
53 Prudential carve-out

3. Nothing in this Article shall affect:

   a. a requirement by a court or administrative tribunal, or a requirement by a competition authority pursuant to a Party's competition law to prevent or remedy a restriction or a distortion of competition;
   b. a requirement by a regulatory body pursuant to a Party's laws or regulations related to the protection of public safety with regard to users online, subject to safeguards against unauthorised disclosure;
   c. the protection and enforcement of intellectual property rights; and
   d. the right of a Party to take measures in accordance with Article III of the GPA as incorporated by Article PPROC.2 [Incorporation of certain provisions of the GPA and covered procurement] of Title VI [Public procurement] of this Heading.

Comprehensive and Progressive Agreement for Trans-Pacific Partnership

Article 14.17: Source code

1. No Party shall require the transfer of, or access to, source code of software owned by a person of another Party, as a condition for the import, distribution, sale or use of such software, or of products containing such software, in its territory.

2. For the purposes of this Article, software subject to paragraph 1 is limited to mass-market software or products containing such software and does not include software used for critical infrastructure.

3. Nothing in this Article shall preclude:

   a. the inclusion or implementation of terms and conditions related to the provision of source code in commercially negotiated contracts; or
   b. a Party from requiring the modification of source code of software necessary for that software to comply with laws or regulations which are not inconsistent with this Agreement.

4. This Article shall not be construed to affect requirements that relate to patent applications or granted patents, including any orders made by a judicial authority in relation to patent disputes, subject to safeguards against unauthorised disclosure under the law or practice of a Party.

Indonesia-Australia Comprehensive Economic Partnership Agreement

Article 13.13: Source code

1. Neither Party shall require the transfer of, or access to, source code of software owned by a person of another Party, as a condition for the import, distribution, sale or use of such software, or of products containing such software, in its territory.

2. For the purposes of this Article, software subject to paragraph 1 is limited to mass-market software or products containing such software and does not include software used for critical infrastructure, or software that is specifically made for use by a Party.

3. Nothing in this Article shall preclude:

   a. the inclusion or implementation of terms and conditions related to the provision of source code in commercially negotiated contracts; or
   b. a Party from requiring the modification of source code of software necessary for that software to comply with laws or regulations which are not inconsistent with this Agreement.

4. This Article shall not be construed to affect requirements that relate to patent applications or granted patents, including any orders made by a judicial authority in relation to patent disputes, subject to safeguards against unauthorised disclosure under the law or practice of a Party.

5. Nothing in this Article shall prevent a Party from adopting or maintaining any measures that it considers necessary for the protection of its essential security interests.

Japan-Mongolia Economic Partnership Agreement

Article 9.11: Source code

1. Neither Party shall require the transfer of, or access to, source code of software owned by a person of the other Party, as a condition of the import, distribution, sale or use of such software, or of products containing such software, in its Area.

2. For the purposes of this Article, software subject to paragraph 1 is limited to mass-market software or products containing such software, and does not include software used for critical infrastructure.

REFERENCES

CMA – Competition and Markets Authority (2018), "Pricing algorithms: Economic working paper on the use of algorithms to facilitate collusion and personalised pricing", 8 October (https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/746353/Algorithms_econ_report.pdf).

CMA (2021), "Algorithms: How they can reduce competition and harm consumers", 19 January (www.gov.uk/government/publications/algorithms-how-they-can-reduce-competition-and-harm-consumers/algorithms-how-they-can-reduce-competition-and-harm-consumers).

Coughlan, S (2021), "A-levels and GCSEs: Boris Johnson blames 'mutant algorithm' for exam fiasco", BBC, 26 August (www.bbc.com/news/education-53923279).

Desai, D and J A Kroll (2018), "Trust but verify: A guide to algorithms and the law", *Harvard Journal of Law and Technology* 31(1).

Ezrachi, A and M E Stucke (2016), *Virtual Competition*, Harvard University Press.

Guglya, L and M Maciel (2016), "Addressing the digital divide in e-commerce JSI: From enabling issues to data and source code provisions", CUTS International (www.cuts-geneva.org/pdf/200615-E-Commerce-Issue_Paper.pdf).

Halderman, J A and V Teague (2015), "The New South Wales iVote system: Security failures and verification flaws in a live online election", *International Conference on e-Voting and Identity,* Springer, pp. 35–53.

ICO – Information Commissioner's Office (2019), "Update report into adtech and real time bidding", 20 June (https://ico.org.uk/media/about-the-ico/documents/2615156/adtech-real-time-bidding-report-201906-dl191220.pdf).

ICO (2020), "Explaining decisions made with AI", 20 May (https://ico.org.uk/for-organisations/guide-to-data-protection/key-data-protection-themes/explaining-decisions-made-with-ai/).

Koeppel, D (2006), "Casino hackers", CNN, 23 October (https://edition.cnn.com/2006/TECH/07/13/popsci.gambling/).

Leslie, D (2020), *Understanding bias in facial recognition technologies: An explainer*, The Alan Turing Institute (https://doi.org/10.5281/zenodo.4050457).

Leslie, D, L Holmes, C Hitrova and E Ott (2020), *Ethics of machine learning in children's social care,* What Works for Children's Social Care (https://whatworks-csc.org.uk/wp-content/uploads/WWCSC_Ethics_of_Machine_Learning_in_CSC_Jan2020.pdf).

Leveson, N G and C S Turner (1993), "An investigation of the Therac-25 accidents", *Computer* 26(7): 18–41.

Libert, T (2015), "Privacy implications of health information seeking on the web", *Communications of the ACM* 58(3): 68–77.

McCann, D (2019), *e-Commerce free trade agreements, digital chapters and the impact on labour: A comparative analysis of treaty texts and their potential practical implications*, New Economics Foundation (www.ituc-csi.org/IMG/pdf/digital_chapters_and_the_impact_on_labour_en.pdf).

McGurk, M R and J W Lu (2015), "The intersection of patents and trade secrets", *Hastings Science and Technology Law Journal* 7: 189.

Murgia, M and M Harlow (2019), "How top health websites are sharing sensitive data with advertisers", *Financial Times,* 13 November (www.ft.com/content/0fbf4d8e-022b-11ea-be59-e49b2a136b8d).

Nicas, J (2020), "No, Dominion voting machines did not delete Trump votes", New York Times, 11 November (www.nytimes.com/2020/11/11/technology/no-dominion-voting-machines-did-not-delete-trump-votes.html).

OECD – Organisation for Economic Co-operation and Development (2017), *Algorithms and collusion: Competition policy in the digital age* (www.oecd.org/daf/competition/Algorithms-and-colllusion-competition-policy-in-the-digital-age.pdf).

OECD (2019), *Challenges to consumer policy in the digital age* (www.oecd.org/sti/consumer/challenges-to-consumer-policy-in-the-digital-age.pdf).

Office for National Statistics (2020), "Annual gross fixed capital formation by industry and asset" (www.ons.gov.uk/file?uri=/economy/grossdomesticproductgdp/datasets/annualgrossfixedcapitalformationbyindustryandasset/current/bb20anngffindassetfinalqa.xls).

Safety Research & Strategies (2013), "Toyota unintended acceleration and the big bowl of 'spaghetti' code", 7 November (www.safetyresearch.net/blog/articles/toyota-unintended-acceleration-and-big-bowl-%E2%80%9Cspaghetti%E2%80%9D-code).

Schwartz, O (2019), "In 2016, Microsoft's racist chatbot revealed the dangers of online conversation", IEEE Spectrum, 25 November (https://spectrum.ieee.org/tech-talk/artificial-intelligence/machine-learning/in-2016-microsofts-racist-chatbot-revealed-the-dangers-of-online-conversation).

Scott, M and T Larger (2019), "To take on big tech, US can learn antitrust lessons from Europe", Politico.eu, 31 August (www.politico.eu/article/europe-us-big-tech-competition-antitrust-apple-google-facebook-amazon/).

Specter, M A, J Koppel and D Weitzner (2020), "The ballot is busted before the blockchain: A security analysis of Voatz, the first internet voting application used in US federal elections", *29th USENIX Security Symposium*, pp. 1535 – 1553.

Wu, M (2017), “Digital trade-related provisions in regional trade agreements: Existing models and lessons for the multilateral trade system”, RTA Exchange, International Centre for Trade and Sustainable Development and Inter-American Development Bank.

ABOUT THE AUTHORS

Cosmina Dorobantu is the Deputy Director of The Alan Turing Institute’s public policy research programme. She is a member of the Financial Conduct Authority and the Bank of England’s Artificial Intelligence Public-Private Forum, as well as the Trade and Economy Panel within the UK’s Department for International Trade. She holds a PhD in Economics from the University of Oxford and is a Research Associate at the University’s Oxford Internet Institute.

Florian Ostmann is the Policy Theme Lead within The Alan Turing Institute’s public policy programme. He is a member of the Royal Statistical Society’s Data Science Section Committee and the Law Committee for the IEEE Global Initiative on Ethics of Autonomous and Intelligent Systems. He holds a PhD in Political Philosophy from University College London and a Master’s degree in Public Policy from Harvard Kennedy School.

Christina Hitrova is a doctoral candidate with the Professorship for Law, Science and Technology at the Technical University of Munich. Christina spent two years at The Alan Turing Institute, researching AI ethics. She was also a trainee with the European Commission’s Legal Service, working within the Trade policy and WTO team. Christina holds Master’s degrees in Law from KU Leuven and the University of Zurich.