Towards Privacy by Design for Data with STRM privacy

Bart van Deenen
strmprivacy.io

Pim Nauts
strmprivacy.io

Robin Trietsch
strmprivacy.io

Bart Voorn
strmprivacy.io &
University of Amsterdam

Abstract

Both societal and regulatory pressure (GDPR) increasingly challenge organizations and engineering teams to balance privacy and innovation. Striking this balance can be costly in terms of effort, data utility and computation costs. Moreover, current approaches in scalable data systems often treat privacy as an access problem, which is at odds with important legal and design principles. A plethora of privacy preserving- and enhancing- technologies are available, yet their adoption in production data systems still faces challenges. In particular, their focus is often on narrow use cases such as external data sharing, on mostly existing data sets, rendering them unusable in real-time data architectures. In this paper we argue engineering teams should "shift left" with their data privacy efforts, to the point of data collection. We show how privacy challenges in production architectures can be addressed without compromising speed, data quality or privacy. We provide a detailed yet practical explanation of an architectural set-up that allows users to launch privacy streams in seconds.

1 Introduction

If the industrial revolution ran on fossil fuel, modern-day organizations run on data. From governments to hospitals, manufacturing to digital marketplaces: data systems are everywhere and have quickly become a core asset and capability to deploy for any modern organization in order to innovate. With"data" driving both considerable value creation and grounded concern about the impact to private individuals, legislators developed and implemented new regulations like the General Data Protection Regulation (GDPR) [26] in the European Union to foster innovation without foregoing individual rights to privacy [7] .

But even in *data-native* organizations, precisely this "balancing" of privacy and data driven innovation is not an easy task for data engineers, privacy officers and data governance experts alike. While the first foundational work on *Privacy by Design* from Ann Cavoukian dates back to the 90s [9], building modern, scalable data systems around privacy principles is still a nascent field due to a lack of (technical) standards and a prolonging gap between the policy perspective and operational reality of data systems [1]. This gap leads to a very real "Cost of Privacy" to organizations beyond just legal costs [5]. Structuring privacy operations in organizations, at scale, requires time and effort. Increased coordination across departments, unclarity around requirements, longer development cycles, additional staffing for e.g. RTBF requests [16] and "privacy paralysis"[1] all mean organizations incur additional- and opportunity costs. For example, a structured approach to privacy is often

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

[1] the observation that taking a next step with application development is delayed or even refrained from due to privacy concerns

lacking, and the GDPR requirements around data collection and user opt-ins lead to reverse-engineering the legal ground and related purpose for collecting and processing data, with engineering teams often pulled into lengthy legal discussions on purpose. Moreover, requirements like strong anonymization pose a practical challenge to data utility and -usability for (data) engineering teams.

Despite this apparent *Cost of Privacy*, available solutions are still limited. We see the gap between the policy perspective and real-world systems is most apparent in the way privacy is embedded in many data architectures. Often, privacy is treated as an access problem on data that is already collected, stored and used[2], not as a first principle of systems design (often without traceability to the collection ground and collection purpose). While this is convenient as it requires limited re-architecting of existing systems, we believe this is an "early efficiency win [that causes] late stage headaches" [5]. Moreover, it is at odds with (if not in violation of) legislative directions. This often leads to (i) organizations accepting an increased risk profile or (ii) refraining from engaging in new use cases (and so innovation) altogether. All in, the current approach to dealing with privacy often poses another real cost and risk to organizations, both financial and in reputation.

1.1 Privacy Enhancing Technologies

One of the domains where these challenges are being (partly) addressed, is in Privacy Enhancing or Privacy Preserving Technologies (PET's) [21]. PET's allow data practitioners to transform sensitive data into usable input for consumption or support detecting sensitive data [5], limiting risk of unsolicited data consumption. Yet, they have important shortcomings in the practical sense: they often center around external data sharing as the dominant use case, are computationally heavy, best suited to aggregate data operations only and/or rely on "complete" data sets. This makes them better suited to use cases where data is already collected and stored. Hence, PET's provide a solution for a limited set of use cases and can still lead to non-compliance with privacy regulations if data was not properly collected or bounded to data purpose [5].

1.2 Towards Privacy by Design for data

To bridge the gap from the realities of privacy professionals and data (system) engineers to available technologies, we propose an architecture to encode privacy in data that helps to structure privacy efforts, and a system to implement this at lower real-world costs. Both the architecture and system come from a set of clear principles, derived from legal and privacy by design requirements:

1.2.1 Requirements

- ***Explicit contracting*** - define both data shape (the schema) and privacy implications as metadata (including key legislative concepts such as collection ground) in data contracts that describe each datapoint.
- ***Enforce transformation*** - define inside the contracts, set per-field privacy transformations, such as de-identification or anonymization, for sensitive (PII[3]) data fields, so that data collection and processing are tightly coupled.
- ***Shift left*** - Enforce these data contracts at ingestion, so data is (i) always bound to purpose and (ii) welded to the original collection ground.
- ***Retain utility*** - transform data in such a way that the trade-off between privacy and utility is limited or even removed.

[2] e.g. the way many modern data warehouses deal with privacy is through role-based access

[3] Personally Identifiable Information, such as a name or email

- ***Dedicated distribution*** - provide a dedicated interface for every specific collection ground through privacy streams, so that applications and/or downstream users are guaranteed to only receive (i) necessary and (ii) properly transformed data.
- ***Interoperable*** - the system should naturally complement existing architectures and accommodate easy integration and deployment (aligned with industry standards).
- ***Cheap and Fast*** - extend interoperability by aiming to add minimal additional latency to existing systems and limit computational cost of operations.

In effect, a solution should allow data teams to build with confidence, cut costly coordination time, and enable them to quickly deploy a contemporary (streaming) data infrastructure with a focus on high quality data at low effort at lower legal and reputation risks.

2 Limiting the Cost of Privacy: privacy, utility and design challenges

In this section we identify the challenges that surround building scalable data systems from privacy forward, and the design principles that should shape them. We will approach these as drafting the list of requirements for building these *Privacy by Design* data systems.

2.1 Privacy challenges

2.1.1 Key requirements from GDPR

While often viewed as a complex legislation, the core of what GDPR seeks to achieve is straight forward: protection of individuals' private data. A few key concepts set the boundaries for responsible (and lawful) data behavior, such as accountability (you are accountable for the data you control), and data subject rights (what an individual can require from a controller). We will use these as a treasure map in hunting for requirements.

Data minimization and consent Of key interest to our challenge are the governing principles on how to collect and use data: data minimization and consent obligations. Simply put, they state one should not collect more data than necessary for a purpose, and can only collect data under *either* explicit consent for that purpose or a justified cause[4], like legitimate interest or contract fulfillment (e.g. an e-commerce marketplace simply needs your home address to deliver your order).

In practice, these legal grounds are oftentimes liberally interpreted. Historical data – i.e. data that has been collected in the past and stored in e.g. a data warehouse – is often persisted without storing the purpose under which it was acquired. This is clearly problematic in the privacy sense: it is (nearly) impossible to determine the purpose of a data point after the data collection took place.

It becomes even more interesting once we acknowledge the dynamic nature of consent. For example, how do we handle changing preferences of end-users, and how is consent linked to data over time? By collecting purpose up-front, and ensuring that this purpose stays with the data, two problems are handled instantly: purpose limitation and consent [24]. This yields the first challenge we should take into account:

Req. 1: ***We should encode how a data point was obtained and for what purpose***

Purpose limitation As data in practice is often collected once and then up- and recycled often, there is another key principle to take into account: purpose limitation. In the official text, GDPR article 5 section 1(b) [26], we see the following:

[4]Consent, purpose and legal ground are used interchangeably

> *Personal data shall be collected for specified, explicit and legitimate purposes and not further processed in a manner that is incompatible with those purposes; further processing for archiving purposes in the public interest, scientific or historical research purposes or statistical purposes shall, in accordance with Article 89(1), not be considered to be incompatible with the initial purposes ('purpose limitation').*

This is directly at odds with the common practice of hoarding data into a warehouse. If the purpose of data collection is not clear, the legal ground cannot be determined and consent cannot legally be obtained [24]. While the legalese and exact wording of what purposes can be considered legal is far outside of the scope of many data engineering professionals, this is a clear area of interest for designing systems around privacy: data usage without determining if it is legal to use for a certain purpose is already violating this very principle. In practice we see this judgement is often made by a data consumer team without (sufficient) knowledge of Personal Data regulations. This paves the way to our second requirement:

Req. 2: ***Data consumers should only receive data they are entitled to, given the consent of the data-owner, for the specific purpose of the data consumer*** [5]

So e.g. while a data-team building a recommender system has a vested interest in knowing what a user does, they do not need to know who the user is - and so could do with any random value representing the user as long as its consistent over time for that specific person or user account (provide consent or legal ground is present!).

2.1.2 Privacy by Design, the principles

Our next source of system requirements are the original Privacy by Design principles laid out by Ann Cavoukian in her capacity as the Information and Privacy Commissioner of Ontario [9].

The first principles[6] revolve around timing - when privacy comes into play in both designing systems and the flow of data. Clearly, it is only private by *design* if "characterized by proactive rather than reactive measures, [anticipating and preventing] privacy invasive events before they happen". Moreover, the "maximum degree of privacy" is in warranting the protection of personal data in "any given" data system (aligning with our first requirement: encode privacy into data). This suggests a profound shift of the point at which privacy becomes part of the data flow:

Req. 3: ***Privacy should be embedded right where data is collected.***

More recent views on best practices like Bhajaria's (2022) [5] also suggest that both classifying and tagging data before it hits data pipelines (i.e. at ingest) mitigates privacy risks more optimally.

2.2 Data utility

2.2.1 Bringing structure to data improves data quality

Even before privacy comes into play, many data practitioners will recognize the challenge of data quality in and of itself. In organizations with lots of operational legacy systems (which does not imply they are "old"!), it is common to see the creative re-use of fields/columns in a data set. Dealing with confusing field names, undocumented changes in lineage and empty fields are common practice in the real world [4]. Moreover, data consumers generally have *no direct control* on what is accepted as valid data, and will often have to resort to heuristics to determine what to do with a certain data field.

[5] processor in GDPR parlance

[6] 1. Proactive not Reactive; Preventative not Remedial, 2. Privacy as the Default Setting, 3. Privacy Embedded into Design

Think of this as *data debt* or negative interest building inside data flows. Adding and enforcing more structure in data will improve data utility over the long run considerably as it addresses this kind of real-world challenges (see for an overview of data quality frameworks [10]. In any case, we should strive to:

Req. 4: ***Make structure explicit and retain as much of the data quality and utility as privacy permits.***

2.2.2 The opportunity of privacy transformations for data utility

One of the key challenges of applying privacy regulations and principles to data systems lies in the (supposed) limitations they impose on data utility. Any personal data collected and used is subject to the GDPR provisions, limiting collection ground and purpose - e.g. you cannot simply use an acquired e-mail for marketing purposes, as used to be common practice. This creates another practical cost, as different data types need to be shielded from general use inside systems.

There are however circumstances under which collected data may be used for different purposes than originally requested: in anonymized form, with "data rendered anonymous in such a way that the data subject is not or no longer identifiable." (Recital 26 of [26]). When done properly, and this is an extremely high standard many organizations fail to achieve (Article 29 Working party, in [14]), this even places personal data outside of GDPR, hence allowing to retain utility. This is especially attractive for pattern-driven tasks such as aggregated analytics and applied machine learning that do not rely on identification of the data subject persé.

Along that axis is a much lighter interpretation of what it means to de-identify data: pseudonymization, "the processing of personal data in such a way that the data can no longer be attributed to a specific data subject without the use of additional information." (Recital 4(5) of [26]). Hence, by separating the necessary link between a data point (such as a click event) and the data subject, GDPR allows data to be used with much more degrees of freedom (even beyond the original purpose limitation!) [14]. In summary, applying lighter or heavier de-identification methods helps to retain the data utility many believe is lost under GDPR.

Req. 5: ***Immediately transform data in such a way that the trade-off between privacy and utility is limited or even removed***

2.3 Technical design challenges

Building and maintaining scalable data systems requires deep knowledge of a wide set of technologies and domains. On top of that, adding privacy does not simplify these already complex designs and systems. A great example of the complexity GDPR brings to systems design is fulfilling *Right To Be Forgotten* (RTBF) requests:

Under RTBF, a data subject can request a data controller to deliver or remove their (personal) data. This has the potential to be a daunting and expensive task, as legally obtained personal data can be *everywhere* in or even outside an organization, from local copies to data shared externally with vendors. In order to completely fulfill an RTBF request, one has to conduct either a full table scan on every day of data for each request in any location, or build and update an index of the users present in every data file or storage, including derivatives of personal data. Both are expensive operations, even when automated. We would have to repeat this at least every 30 days (minus operation time), as that is the response window GDPR permits. RTBF is just one of many challenges (and opportunities) GDPR imposes. Various challenges on Integration, Architecture, and Performance exist.

2.3.1 Architecture

One of the first challenges to discuss is in the (dominant) architecture underlying many data systems: batch processing, where (often millions) of rows of data are created and processed at set intervals.

But if data "exists" already, we run into a clear challenge for adhering to the aforementioned principles or to be able to *shift [privacy] left* to the point of data collection. The presence of personal data in raw form needs to be prevented up-front. A sensible way to achieve this is by shifting data collection to an event-based approach. If we capture and treat each datapoint individually, e.g. through following the exchanging API's design, we can enrich and add metadata at much higher resolution. A stream processing-based[7] event gateway allows for coupling specific metadata to each and every single data point (the event) and so allows to enforce "embedding" privacy.

Although a good starting point for privacy by design systems, creating (and maintaining) an end-to-end streaming architecture with data enrichment centered around strict requirements is technically challenging. Technical difficulties and limited maturity to fully benefit from real-time architectures in data consumption has caused many companies to avoid or even abandon them. Executing transformation on a data stream is technically more demanding in both development and reliability as compared to batch processing, and standards (both technical and implementation) are not as widespread. When applied to business challenges in the right way, processing and enrichment of streaming data can create highly valuable real-time insights, for example into customer behavior (see for an example [12] and the remainder of the Dec 2015 Bulletin) and real-time machine learning use cases [27].

In order to conduct the type of transformations necessary for privacy processing, popular tools such as Apache Kafka, Apache Calcit e(streaming SQL), Kafka Streams or Apache Flink can be leveraged. While powerful, in our experience they are also characterized by a steep learning curve, and are non-trivial to bring into production (in both platform maintenance and usage), especially at scale. Other popular components and solutions (self-hosted or managed) generally provide the necessary strict event formats and support for stream processing, but are agnostic to use cases or specific application domains and therefore require a lot of engineering on top of the bare metal solutions. For instance, retaining data purpose like data subject consent is still left to the data consumers.

2.3.2 Integration

Another key challenge when building for privacy lies in the integration to existing data technology. Enterprises and SMEs are often (deeply) invested in existing data architectures, and are likely unwilling or not capable of switching technologies very easily. As mentioned, when designing for privacy, the goal is to prevent having to deal with privacy *after* data collection, preferably agnostic to the configuration and architecture of existing data systems [5] that handle data downstream.

To many engineers, even when leveraging cloud or OSS[8] building blocks, designing a platform for data processing from privacy forward is a daunting prospect, precisely because the aforementioned gap between the legal and policy perspective persists. However, drawing upon our experience, it is possible to achieve Privacy by Design and implement its principles by framing it as an augmentation instead of a replacement challenge . In fact, many basic building blocks are provided by various technologies that have proven themselves over the past years (e.g. like Apache Kafka and processing engines). We therefore argue that evaluating how to augment and evolve an existing stack is likely a more effective strategy than re-architecting everything for privacy.

Moreover, an event-based approach provides the opportunity to simply aggregate and transform data to match existing downstream (batch) processes at low cost, while retaining the metadata necessary for encoded privacy. This limits scope of implementing such systems: it would simply precede existing processes instead of replacing them and is valuable even when there's no immediate need or value to obtain and consume data at (near) real-time latency.

[7] Often referred to as streaming or real-time data

[8] Open Source Software

2.3.3 Performance

The last key challenge to point out is in system performance. While end-to-end stream enrichment with meaningful data can bring value and suits the privacy challenges well, these opportunities can be offset by decreased systems performance (e.g. increases in system latencies can cause decreased conversion in online retail [2]. A lot of the value of enrichment occurs if the result of the enrichment is available as soon as possible. Hence, latency is critical to take into account when preventing raw form PII data at the moment of collection.

A major factor in the performance of such systems is how they respond to increased scale. Bottlenecks are highlighted quickly when data processing systems are put under high load. All sorts of challenges should be taken into account when designing for scalability of data processing systems: distribution of the data, average size, different types of events, and many other factors. Orchestration platforms, such as Kubernetes, allow for fine-tuning in order to deal with the requirements of scalability and flexibility.

Systems can scale in two ways: vertical (more resources) and horizontal (more instances). Designing systems for horizontal scalability is more complex, as applications should be designed stateless and resilient to restarts, but often cost effective as dynamically scaling instances keeps idle resources to a minimum. All these factors (architectural, integration and performance challenges) add up to our final requirement:

Req. 6: ***Complement existing data systems, focus on scalability and aim to add as little latency and computational costs.***

3 From Requirements to Solution: STRM Privacy

The aforementioned requirements are the boundaries for a Data System designed for Privacy. The core premise of privacy regulations like GDPR is that every data item should have a legal ground to gather and process it. The only way to guarantee proper processing of the data item is to tie the owner and permissions based on legal grounds directly and irrevocably into the data item. Our solution adds a standard set of attributes to *every data item* to achieve this.

Structure is imposed before any data is transferred through a data contract[9] defining the data shape, the privacy implications and the data validations. As can be seen in figure 1, a central Event Gateway receives data from many different applications, where a data contract is enforced, and *Personal Data Attributes* are encrypted[10]. Basic data validation is performed upon ingest, resulting in an encrypted data stream (where only the PII fields in data items are encrypted). It should only be possible to decrypt these attributes provided the data collection purpose and *specific usage or consumption purpose* match. By encrypting on ingest, and only putting encrypted *privacy safe* data items in long-term storage, the long-term storage does not become data *toxic waste*.

While taking purpose into account, a decrypter step then creates derived privacy streams, including only events with the required consent for that stream as decrypted. If a user did not provide consent for their data being used for a specific purpose, the data does not end up as private data in these streams.

We apply encryption methods beyond just obfuscating underlying field values. Through key rotations on links inside the data items, privacy transformations such as masking or anonymization can be performed in real-time. Combined with the purpose and consent binding, we can split the incoming data and create purpose-driven data interfaces that downstream applications or teams can consume without privacy concerns.

The encryption mechanism will assign the same encryption key to the same value of a specific data item field. As the same value for a field in the data item yields the same cipher-text value, the integrity between events is retained. With a time-based encryption key rotation (i.e. the privacy algorithm) we can consider the resulting encrypted data stream to be *not privacy sensitive anymore*. Next to that, in case a data owner requests to *be*

[9] for more information on data contracts: `https://docs.strmprivacy.io/docs/latest/concepts/schemas-and-contracts/`

[10] through a symmetric encryption algorithm, with a fixed initialization vector

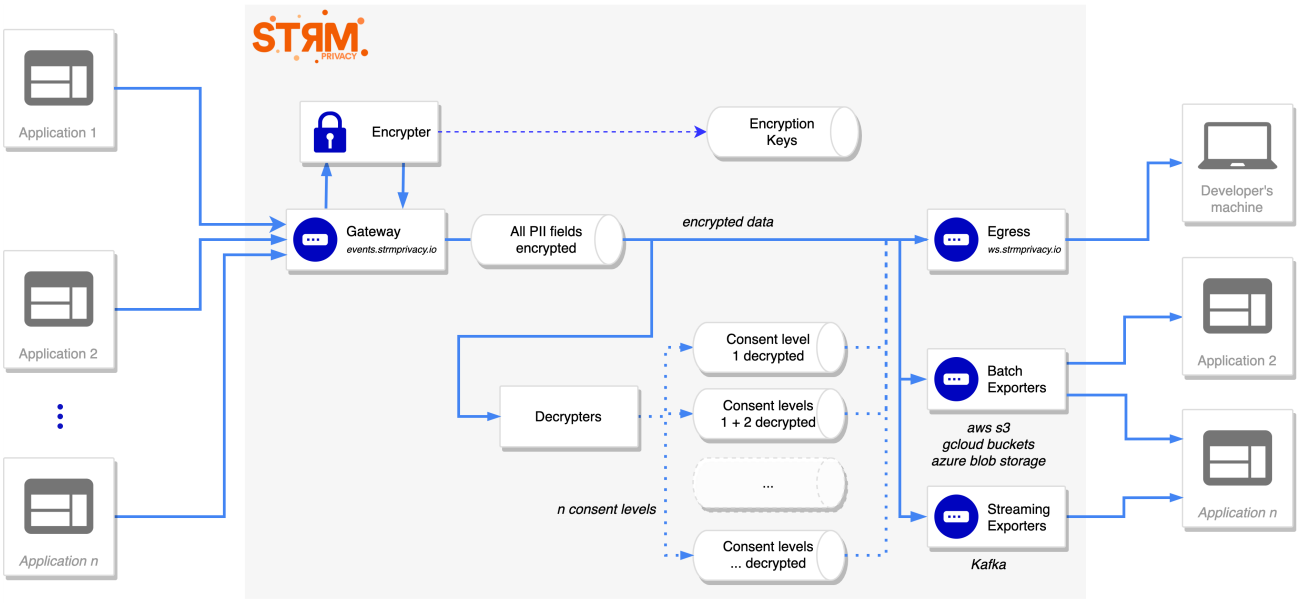

Figure 1: STRM Privacy High Level Architecture

forgotten, throwing away the encryption key destroys all personality dimensions inside the data, but helps to retain the utility of the non-sensitive fields. This process is also known as crypto-shredding [11].

In sum, our architecture allows to encode privacy into a data point from the moment of collection onward, enforcing that personal data is only available when the purpose allows it, while retaining important utility of the data. Our documentation [25] provides a deep dive into all the various components that drive this architecture.

4 Related Work

At this point, we positioned the need and guiding principles for building real-world data systems from privacy forward. We have described how the architecture of STRM Privacy follows these principles through an *event-based* component, a *privacy* component, and a *data quality* component.

Evidently, specific technologies exist for the various components such an architecture requires. First, for data processing in streaming fashion, there are various established technologies, such as Apache Kafka, Kafka Streams, Apache Beam, and many other Open Source and commercial solutions (see e.g. [15] or [19]).

Secondly, we emphasize that specific solutions in the realm of Privacy Enhancing Technologies are on the rise that warrant privacy through different approaches (see for an overview [21]). Their goal generally is to destroy identification in data, while maintaining the same (statistical) characteristics to sufficiently represent the original data to conduct operations (see for example earlier IEEE publications [17] or [28]). For example, solutions include Synthetic Data Generation, obfuscating data on existing data sets, and data encryption at rest. Some challenges remain, e.g. with obfuscation, one should ensure that the obfuscated data is not susceptible to linkage attacks combining obfuscated data with other data to de-anonymize the data [20]. An important limitation of these approaches is that they are currently not able to accommodate real-time transformations and embedding in operational data systems [3], and often require existing data sets that might be collected in violation.

Finally, *data quality* is considered to be of pivotal importance for any downstream application [8],[6]. Hence, both traditional ETL tooling and novel ML supported techniques (see for example [13]) enable both validating and enriching data at ingest or even in transit. Metadata management is an important adjacent field, which could further enrich *data usability* in data pipelines [22], although scalability of such systems is still challenging [23].

5 Conclusion

In this paper we argued that building data systems for privacy in real-world production settings should be guided by a core set of principles derived from legislative and *Privacy by Design* frameworks. This enables closing an existing gap between the policy perspective of privacy and the operational reality of data systems.

Existing *Privacy Enhancing Technologies* focus mostly on heavy computational tasks, such as synthetics, at the cost of latency and compute power. They are generally best suited to existing data sets, not in-flight data. Critically, data teams should not treat privacy as an access problem and shift their data privacy efforts to the point of data collection (i.e. "shift left") to better position for compliance. When employing data contracts to encode privacy and tight coupling to privacy transformations, scalable data systems that retain performance and data quality are achievable. They can be compatible with progressive data system philosophies (i.e. the Modern Data Stack or Data Mesh [18]). In effect, this lowers the *Cost of Privacy* through better data usability, reduced compliance risks and improved coordination between engineering teams and legal departments.

To the best of our knowledge, no other solution exists that focuses on encoding privacy inside data, combined with proven technologies for streaming infrastructure to minimize impact in latency and performance to complement existing data systems. Apart from a better modus operandi for data systems in general, such systems can drive valuable innovation like real-time machine learning while respecting end-user privacy (and thus privacy regulations), and bring privacy by design for data to a much wider audience. We happily invite the community to further extend the collective body of knowledge, by providing feedback or iterations to our work.

References

[1] Serge Abiteboul and Julia Stoyanovich. Transparency, fairness, data protection, neutrality: Data management challenges in the face of new regulation. *Journal of Data and Information Quality (JDIQ)*, 11(3):1–9, 2019.

[2] Akamai. Akamai Online Retail Performance Report. `https://www.akamai.com/newsroom/press-release/akamai-releases-spring-2017-state-of-online-retail-performance-report`, 2017.

[3] Jason W Anderson, Ken E Kennedy, Linh B Ngo, Andre Luckow, and Amy W Apon. Synthetic data generation for the internet of things. In *2014 IEEE International Conference on Big Data (Big Data)*, pages 171–176. IEEE, 2014.

[4] Mahmoud Barhamgi and Elisa Bertino. Special issue on data transparency—data quality, annotation, and provenance. *Journal of Data and Information Quality (JDIQ)*, 14(1):1–3, 2022.

[5] N. Bhajaria. *Data Privacy: A Runbook for Engineers*. Manning, 2022.

[6] Eric Breck, Neoklis Polyzotis, Sudip Roy, Steven Whang, and Martin Zinkevich. Data validation for machine learning. In *MLSys*, 2019.

[7] Mind the Bridge. Gdpr and beyond recommendations from a transatlantic perspective, Dec 2018.

[8] Li Cai and Yangyong Zhu. The challenges of data quality and data quality assessment in the big data era. *Data science journal*, 14, 2015.

[9] Ann Cavoukian. Privacy by design: The 7 foundational principles, 1997.

[10] Corinna Cichy and Stefan Rass. An overview of data quality frameworks. *IEEE Access*, 7:24634–24648, 2019.

[11] Andrea Compton. Assured deletion in the cloud. `https://www.cs.tufts.edu/comp/116/archive/fall2014/acompton.pdf`.

[12] Maosong Fu, Sailesh Mittal, Vikas Kedigehalli, Karthik Ramasamy, Michael Barry, Andrew Jorgensen, Christopher Kellogg, Neng Lu, Bill Graham, and Jingwei Wu. Streaming@ twitter. *IEEE Data Eng. Bull.*, 38(4):15–27, 2015.

[13] Stefan Grafberger, Julia Stoyanovich, and Sebastian Schelter. Lightweight inspection of data preprocessing in native machine learning pipelines. In *Conference on Innovative Data Systems Research (CIDR)*, 2021.

[14] Looking to comply with gdpr? here's a primer on anonymization and pseudonymization. `https://iapp.org/news/a/looking-to-comply-with-gdpr-heres-a-primer-on-anonymization-and-pseudonymization/`.

[15] Haruna Isah, Tariq Abughofa, Sazia Mahfuz, Dharmitha Ajerla, Farhana Zulkernine, and Shahzad Khan. A survey of distributed data stream processing frameworks. *IEEE Access*, 7:154300–154316, 2019.

[16] Paulan Korenhof, Jef Ausloos, Ivan Szekely, Meg Ambrose, Giovanni Sartor, and Ronald Leenes. Timing the right to be forgotten: A study into "time" as a factor in deciding about retention or erasure of data. In *Reforming European data protection law*, pages 171–201. Springer, 2015.

[17] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd international conference on data engineering*, pages 106–115. IEEE, 2007.

[18] Inês Araújo Machado, Carlos Costa, and Maribel Yasmina Santos. Data mesh: Concepts and principles of a paradigm shift in data architectures. *Procedia Computer Science*, 196:263–271, 2022.

[19] Erum Mehmood and Tayyaba Anees. Challenges and solutions for processing real-time big data stream: A systematic literature review. *IEEE Access*, 8:119123–119143, 2020.

[20] Martin M Merener. Theoretical results on de-anonymization via linkage attacks. *Trans. Data Priv.*, 5(2):377–402, 2012.

[21] Jules Polonetsky and Tim Sparapani. A review of the privacy-enhancing technologies software market. *IEEE Security Privacy*, 19(6):119–122, 2021.

[22] Pegdwendé Sawadogo and Jérôme Darmont. On data lake architectures and metadata management. *Journal of Intelligent Information Systems*, 56(1):97–120, 2021.

[23] Harcharan Jit Singh and Seema Bawa. Scalable metadata management techniques for ultra-large distributed storage systems–a systematic review. *ACM Computing Surveys (CSUR)*, 51(4):1–37, 2018.

[24] Jordi Soria-Comas and Josep Domingo-Ferrer. Big data privacy: challenges to privacy principles and models. *Data Science and Engineering*, 1(1):21–28, 2016.

[25] STRM Privacy documentation. `https://docs.strmprivacy.io/`.

[26] European Union. General Data Protection Regulation. `https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679&from=EN`, 2016.

[27] Eugene Yan. Real-time machine learning for recommendations. `https://eugeneyan.com/writing/real-time-recommendations/`, 2021. Post published on 10 Jan 2021.

[28] Bin Zhou, Jian Pei, and WoShun Luk. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *ACM Sigkdd Explorations Newsletter*, 10(2):12–22, 2008.