22 April 2020 Study · Transparency · Privacy & Crypto · 15 min read

Gap Analysis of Data Transparency Logging Architectures

Archived. This content is no longer updated. It remains available for reference.

Source https://technology2.a-sit.at/en/publications/gap-analysis-of-data-transparency-logging-architectures/

The General Data Protection Regulation (GDPR) specifies a series of transparency obligations for data controllers concerning the processing and sharing of personal data. Transparency architectures focus on securely and efficiently logging private data events to enable either data subjects or trusted auditors to verify that the processing of personal data has happened In accordance with agreed privacy policies and the existing regulations. This project investigates the advantages and limitations of different candidate architectures for transparency logging. We evaluate existing solutions in terms of features such as Confidentiality, Integrity, Immutability, Interoperability, and Traceability. Furthermore, the project uses the results of the gap analysis to identify the primary open challenges and opportunities for future research.

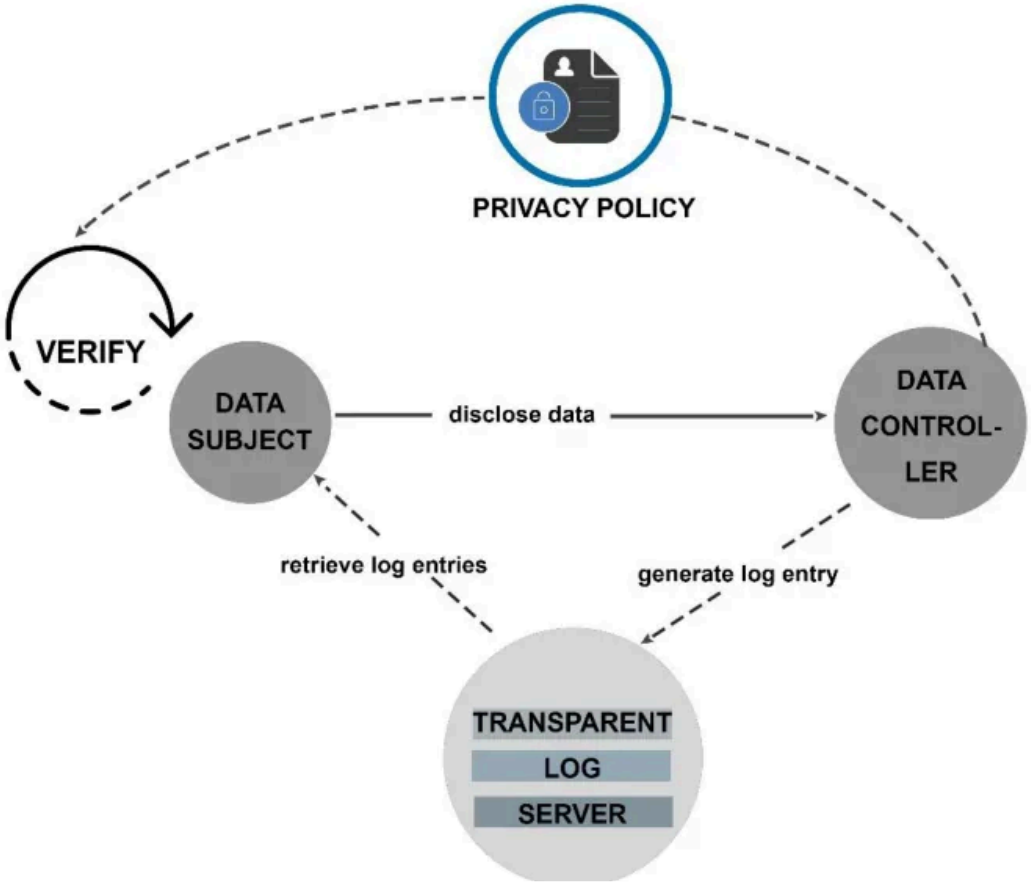
1. Introduction

Transparency is a core principle of the General Data Protection Regulation (GDPR) [1]. Under this regulation, data controllers must abide by transparency obligations through all the stages of data collection and processing. Articles 12-15 define specific requirements for data controllers concerning the provision of information to data subjects regarding the processing and sharing of private data. Furthermore, GDPR stresses that the comprehensibility and the accessibility of the information provided to the data subjects are as important as its content.

Therefore, there is a need for technical means that enable data subjects, data controllers, and thirdparty auditors to either demonstrate or verify that the processing of personal data is compliant with the data subject's consent, the privacy policies, and the obligations set by the GDPR. This analysis takes as a basis the various existing transparency logging mechanisms. The study starts by identifying several requirements that apply to transparent personal data processing architectures. It then continues to examine state-of-the-art solutions with respect to these requirements. Finally, it identifies gaps and potential open research directions.

2. Requirements:

We start by defining the environment for data transparency logging. The main actors in this setting are data subjects, data controllers, and log servers. Data subjects use the services provided by data controllers and disclose their personal data to these services. In turn, data controllers record personal data processing transactions in a server dedicated to transparency logging. Examples of such transactions include which user shared data with which controller, with which purpose, and under what conditions. Data subjects can further retrieve log entries related to them from these log servers. Besides these primary actors, this setting might include several trusted third parties such as Monitors, Timestamping authorities, or Auditors. Figure 1 illustrates how data transparency logs can enable efficient verification of data processing and sharing events. A data subject discloses private data to a controller under the conditions specified in the privacy policy. Every time the data controller processes the data, it generates a log entry that records the details of the processing. Data subjects can then retrieve the log entries to *Figure 1 Transparency Logging Overview*



In real life, services and data processing transactions are often distributed. This implies that multiple data controllers (services) share the disclosed personal data. Figure 2 illustrates such a distributed setting with several users and data controllers. In this example, the data subjects first disclose their private data to the service provided by Data Controller 1 (DC1). DC1 uses the services provided by DC2, which in turn interact with services provided by data controllers DC3 and DC4. Hence, the personal data firstly disclosed to DC1 is further shared with DC2, DC3, and DC4. There are several options for the selection of log servers by data controllers. Each data controller might use one log server, or multiple data controllers might share logs servers. Data subjects can then query the log *Figure 2 Example of a Transparent Data Processing Architecture* We define the following requirements that apply to this setting:

Functionality: Data subjects should be able to identify and retrieve all the log entries that belong to them. In the case of a distributed environment where multiple data controllers share a log server, data controllers should be able to identify and retrieve all the log entries that were generated by them.

Integrity: Transparency logs must be protected from tampering (either accidental or malicious). Both data subjects and data controllers should be able to verify the integrity of the log. Furthermore, a trusted auditor should be able to verify the integrity of the log hosted by the server.

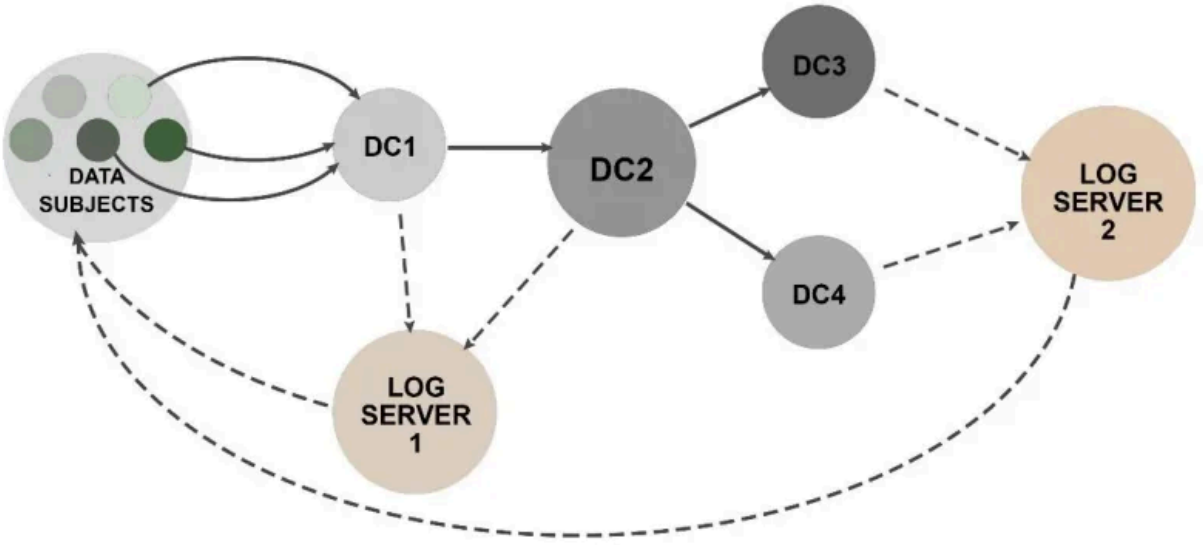
Immutability: Transparency logs should be immutable (tamper-proof, append-only). As such, it should not be able to modify or delete past log entries.

Confidentiality: As the log content concerns personal data, it should remain confidential. Only data subjects and data controllers associated with particular log entries should be able to access and retrieve them.

Unlinkability: It should not be possible to link different log entries, as linking log entries related to a particular data subject might further sensitive data related to the user (such as service access patterns).

Completeness: It should not be possible to omit to record a data processing event. All data processing transactions should be logged in Transparency Logs.

Correctness: The log content should be correct with respect to the real data processing transactions.



Non-repudiation: It should not be possible to deny later that a specific data processing event took place in the past.

Traceability: In the distributed setting, it should be able to trace distributed data processing events.

Interoperability: It should be possible for data subjects to combine log entries generated by multiple data controllers.

Redaction & Erasure: It should be possible to either correct errors introduced to the transparent logs or completely erase the logged data of a particular data subject at the data subject's request.

Scalability: Log entry generation and retrieval, as well as proofs of integrity, should be scalable.

Storage: The amount of storage required to store the log content should be kept to a minimum to improve the log performance and scalability. This might imply storing only snapshots of the personal data processing events in the log instead of the data itself, and storing the full data elsewhere.

Performance: Optimization techniques should be employed to enhance the log's processing efficiency. A viable measure could be the identification and then optimization of the most costly subprocesses of the log operation.

Availability: Actors of the ecosystem (data subjects, data controllers, third-party auditors, or monitors) should be provided with optimal accessibility and usability of the transparency logs, regardless of the underlying technical architecture.

3. Data Transparency Logs

We can observe three main high-level architectural patterns in the existing proposals with respect to persisting sensitive-data transaction records:

i. Local Logs: ii. Global Log entrusted to Trusted Third Parties iii. Global Log distributed among peers.

Furthermore, these three patterns do not need to be mutually exclusive. This section goes into the details of existing solutions for each of these candidate architectural patterns.

Local Logs: In the Local Log setting, each data controller maintains a local log server to persist provenance data. Most of the early solutions proposed in the field of secure logging employ a local ledger. First examples include the work done by Bellare and Yee [2] and Schneier and Kelsey [3]. Both of them make use of hash chains generated using a secret key signing algorithm based on MACs (Message Authentication Codes) to protect the integrity and confidentiality of the log. Furthermore, Bellare and Yee formally defined the Forward Integrity (FI) property in the secure logging setting. They proposed to employ evolving MAC keys (keys derived by previously used keys) to authenticate the log entries. This ensured that in the case of a compromise, previously logged entries could remain secure.

Similarly, Schneier and Kelsey combine hash chains (in contrast to cipher-block chains used by Bellare and Yee) and evolving MAC Keys. Schneier and Kelsey also introduce a basic searchable encrypted log by adding metadata of the form of permission masks to determine access control rights of external verifiers. On the other hand, work from Holt [4] and Ma and Tsudik [5] employ public-key cryptography. Holt

[4] replaces hash chains with signature chains. In this solution, each log entry that is generated makes use of a new private key that corresponds to a public key stored in the previous log entry. Later, Ma and Tsudik [5] employ Forward-Secure-Sequential-Aggregate-Signatures to improve on the proposal from Schneier and Kelsey. Their work combines individual log entry signatures into a single aggregate signature to secure the log against log truncation attacks.

Sackman et al. [6] was the first work to propose using secure logs to store privacy-aware provenance records. In this proposal, secure data-processing logs are used by trusted auditors as 'evidence' for the comparison of the data processing to what is stated in the privacy policies. Unlike Sackman et al. [6], Hedbom et al. [7] give data subjects the role of the primary auditor and further discuss the unlinkability of entries in one local log in one server. A summary of these proposals can be found in Table 1:

Table 1 Local Log Candidate Architectures

Col1	Confidentiality	Integrity	Immutability	Unlinkability
Bellare and Yee	MAC	forward integrity, MAC security proofs	cipher chains	
Schneier and Kelsey	MAC	forward integrity	hash chains	
Holt	MAC, PKI	forward integrity	hash chains, signature chains	
Ma and Tsudik	FssAgg, PKI	forward integrity		
Sackman et al.	MAC	forward integrity		
Hedbom et al.		deletion- detection, forward integrity		Unlinkability among entries of the same log

Global Logs entrusted to Trusted Third Parties Another candidate architecture for transparency logging consists of a global log maintained by one or more trusted third parties.

Accorsi [8] tailored the scheme proposed by Schneier and Kelsey [3] to make it applicable to resource-restricted devices that remotely generate log entries. In this scheme, the log entries generated by devices are recorded by collectors. In [8], Accorsi also discusses the privacy aspects of their secure logging scheme, differentiating between inner (concerning private log data) and outer privacy (concerning observations of actions).

The rest of the candidate architectures detailed in this section consider the data subject to be the main verifier of the integrity of the logged private data access events. Wouters et al. [9] employ public-key cryptography to log private data transactions. The authors were the first to realize that such events are not isolated and should be logged as part of a trail of events. This enables the data subject to verify the status of a process that is using his/her private data.

Similarly, Peeters et al. [10] and Pulls et al. [11] propose MAC-based secure logging mechanisms distributed among several servers. In this work, the MAC is generated by a Trusted Third Party, which then encrypts it with the user's private key and signs it with its private key before sending it to the data subject through the data controller. Log entries are comprised of three blocks: the data subject block, the data controller block, and the encrypted content of the entry. The encryption of the log data is carried out such that only the data subject and the data controller are able to decrypt it. They also tackle data sharing among different data controllers. They address the unlinkability issue among the logs by generating a blinded key for the second

data controller such that the data subjects can decrypt log entries encrypted with this blinded key. Authors also evaluate the performance of their logging mechanism to find out that signing and encryption compose the most expensive operations, followed by decryption and verification. A summary of the candidate architectures detailed under this category is given in Table 2.

Table 2 Candidate Log Architectures: Global Log + TTP

Col1	Confidentiality	Integrity	Immutability	Traceability	Unlinkability
Accorsi	MAC	forward integrity			
Peeters et all.	MAC	forward integrity	Merkle Trees +Hash Treaps		
Pulls et al.	MAC	forward integrity	Merkle Trees + Hash Treaps		
Wouters et al.	PKI			Trail of log events	Unlinkability between logs on different servers
Hedbom et al.	MAC	forward integrity			Unlinkability among entries of the same log

Global Logs distributed across peers In their early proposal, Schneier and Kelsey [3] emphasized several vulnerabilities related to having a single Trusted Third Party is secure logging schemes. They further sketched a proposal to replace a single TTP with multiple untrusted machines, where a subset of these machines are required to reproduce the base MAC key. Their proposal highlighted a distributed architecture of secure logs – a global log distributed among several physical logs – where log contents are replicated between peers. Seneviratne and Kagal [12] refine the idea first sketched by Weitzner et al. [13] to have the log of transactions stored (in a replicated fashion) in distributed peers. The authors, however, fail to address the main technical and functional challenges introduced by this architectural pattern and focus on optimizing the availability and usability of the log.

Another alternative solution under this category is linked to the hyped blockchain technology. Zyskind et al. [14] propose to use the blockchain data model to store access to personal data. To overcome privacy-related issues, the data encrypted with a shared encryption key is stores in a separate keyvalue store, and a hash of the data is published in the public blockchain. Furthermore, they provide access to only data subjects that own the data and data controllers that generated the log entry by using compound identities.

In another proposal, Schaefer and Edman [15] propose to combine a private and a public blockchain to increase the transparency of personal data handling while preserving the integrity and immutability of the log. They use the private blockchain to store the actual log data and the public blockchain as a trust anchor for the private blockchain. Hashes of the last block of the private blockchain are periodically published in the public blockchain. Data subjects are given respective access permissions to the private blockchain. Their use case is, however, limited to logging personal data use within one enterprise. A summary of the proposals in this category is given in Table 3.

Table 3 Candidate Log Architectures: Global Log Distributed across peers

Col1	Confidentiality	Integrity	Immutability	Traceability	Unlinkability
Schneier and Kelsey	MAC Compound Identities	forward integrity			
Seneviratne and Kagal	PKI		Network of peers		
Zyskind et al.	Compound Identities		blockchain		
Schaefer and Edmain	PKI		Blockchain (private + public)		
Weitzner et al.			Network of peers		

4. Gap Analysis

After analyzing the different candidate architectures for private data access logs, we proceed to identify the main technical challenges that are not addressed by the existing proposals.

i. Research related to this topic can be mainly categorized into: a. designing a secure, efficient, privacy-preserving logging scheme that suits the application; b. modeling the content of transparency logs in order to enable c. interoperability and traceability between logs, and d. checking for compliance with previously agreed privacy policies.

ii. The main settings that can be identified for transparency logs are generally either local transparency logs or global transparency logs, possibly distributed among multiple TTPs or peers, without excluding a combination of approaches.

iii. The key properties of secure logging systems that should be ensured/improved when proposing a scheme are Confidentiality and Integrity, Availability, Scalability, and Storage Requirements. Most of the previous approaches employ MAC and hash/cipher chains for confidentiality and integrity. Furthermore, the evaluation results of related solutions such as [11] show encryption and signing processes to be the most time-consuming processes of the schemes. Faster cryptographic primitives can be investigated to improve the performance of the schemes. Storing the actual log entries separately and storing pointers to logs and their hash values in public transparency logs might prove beneficial to reduce the storage requirements of such schemes.

iv. Existing solutions fail to propose technical means that ensure the correctness and completeness of transparency logs, as it is almost impossible to prevent a data processor from entering false data or omitting to log a certain event. Applying non-repudiation mechanisms in private-data-sensitive processes, such as fair-exchange protocols, might be a viable option to fill in this gap.

v. Related work fails to address mechanisms that balance the requirements for immutability versus those for update and delete (right to be forgotten). Solutions from other areas such as cryptographic delete or versioning might be investigated to overcome such challenges.

5. References

[1] J. F. A. Murphy, “The General Data Protection Regulation (GDPR),” *Irish Medical Journal* . 2018, doi: 10.1145/3170427.3170632.

[2] M. Bellare and B. S. Yee, “Forward Integrity For Secure Audit Logs,” *Transactions on Information and Systems Security* . 1997, doi: 10.1.1.28.7970.

[3] B. Schneier and J. Kelsey, “Cryptographic support for secure logs on untrusted machines,” in *Proceedings of the 7th USENIX Security Symposium*, 1998.

[4] J. E. Holt, “Logcrypt: Forward security and public verification for secure audit logs,” *Conf. Res. Pract. Inf. Technol. Ser.*, 2006.

[5] D. Ma and G. Tsudik, “A new approach to secure logging,” *ACM Trans. Storage*, 2009, doi: 10.1145/1502777.1502779.

[6] S. Sackmann, J. Strüker, and R. Accorsi, “Personalization in privacy-aware highly dynamic systems,” *Communications of the ACM* . 2006, doi: 10.1145/1151030.1151052.

[7] H. Hedbom, T. Pulls, P. Hjärtquist, and A. Lavén, “Adding secure transparency logging to the PRIME core,” in *IFIP Advances in Information and Communication Technology*, 2010, doi: 10.1007/978-3-642-14282-6_25.

[8] R. Accorsi, “On the relationship of privacy and secure remote logging in dynamic systems,” *IFIP Int. Fed. Inf. Process.*, 2006, doi: 10.1007/0-387-33406-8_28.

[9] K. Wouters, K. Simoens, D. Lathouwers, and B. Preneel, “Secure and privacy-friendly logging for eGovernment services,” in *ARES 2008 - 3rd International Conference on Availability, Security, and Reliability, Proceedings*, 2008, doi: 10.1109/ARES.2008.41.

[10] R. Peeters, T. Pulls, and K. Wouters, “Enhancing Transparency with Distributed Privacy-Preserving Logging,” in *ISSE 2013 Securing Electronic Business Processes*, 2013.

[11] T. Pulls, R. Peeters, and K. Wouters, “Distributed privacy-preserving transparency logging,” in *Proceedings of the ACM Conference on Computer and Communications Security*, 2013, doi: 10.1145/2517840.2517847.

[12] O. Seneviratne and L. Kagal, “Enabling privacy through transparency,” in *2014 12th Annual Conference on Privacy, Security and Trust, PST 2014*, 2014, doi: 10.1109/PST.2014.6890931.

[13] D. J. Weitzner, H. Abelson, T. Berners-Lee, J. Feigenbaum, J. Hendler, and G. J. Sussman, “Information accountability,” *Commun. ACM*, 2008, doi: 10.1145/1349026.1349043.

[14] G. Zyskind, O. Nathan, and A. S. Pentland, “Decentralizing privacy: Using blockchain to protect personal data,” in *Proceedings - 2015 IEEE Security and Privacy Workshops, SPW 2015*, 2015, doi: 10.1109/SPW.2015.27.

[15] C. Schaefer and C. Edman, “Transparent logging with hyperledger fabric,” in *ICBC 2019 - IEEE International Conference on Blockchain and Cryptocurrency*, 2019, doi: 10.1109/BLOC.2019.8751339.