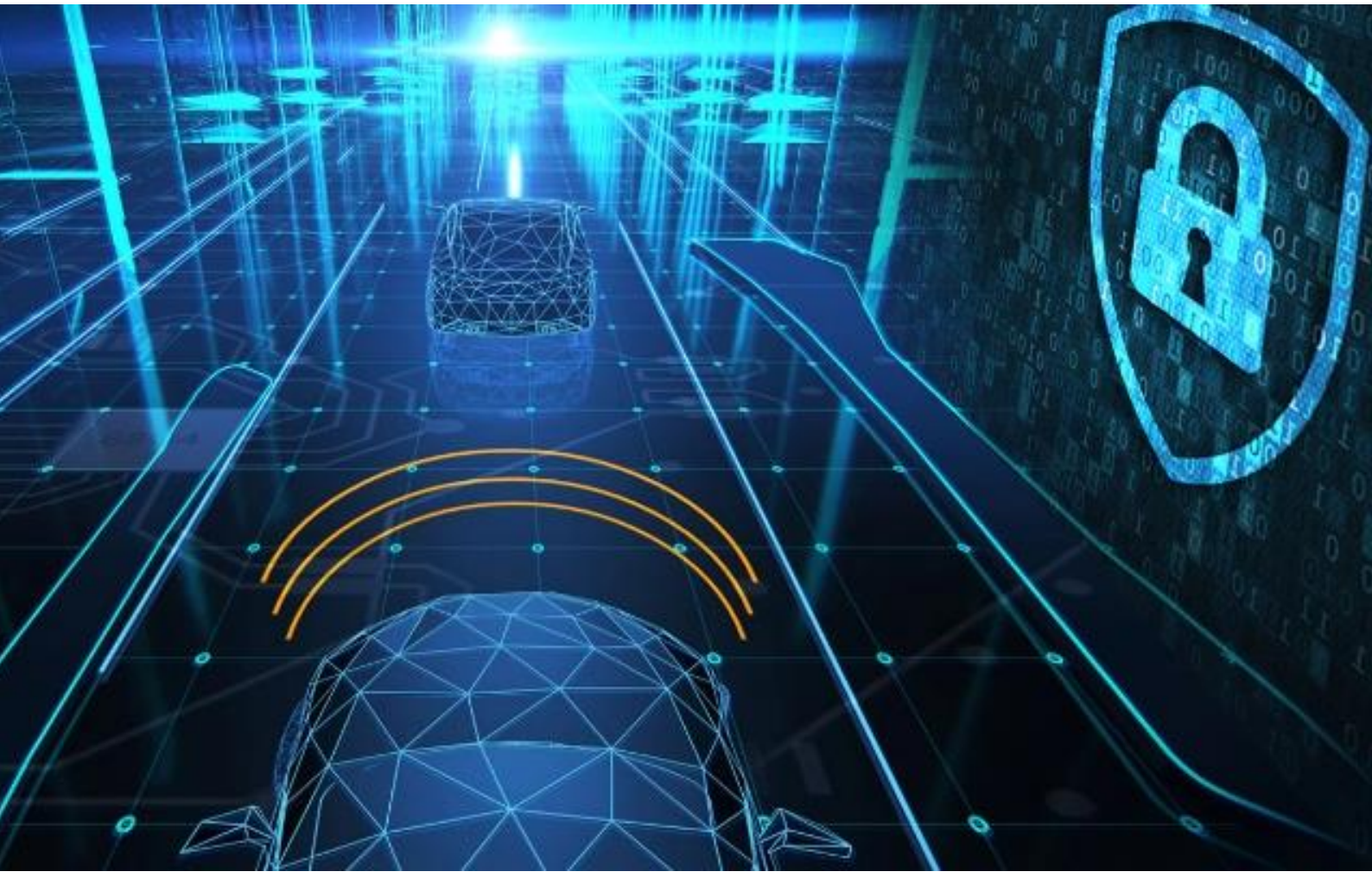
smashHit

Smart Dispatcher For Secure And Controlled Sharing Of Distributed Personal And Industrial Data

Optimised smashHit Framework

Public Deliverable D5.4



December 2022

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 871477

Foreword

Welcome to our smashHit Optimised Framework! From the very beginning of the project, we were absolutely convinced that the growing Data Economy has to become more attractive for its key stakeholders (data subjects, data providers and data controllers) to overcome existing barriers, as e.g. the complicated and time-consuming consent/contract processes, hindering to build-up innovative services using data from multiple sources.

With the growing ability of Consumer Products (CP, such as cars, smart devices, etc.) to generate, gather and share data with third parties among different data-sharing platforms, there will be a general need for flexible and easily manageable procedures to handle data subject's consent and legal rules, to achieve effective and traceable contracting. The complexities of the General Data Protection Regulation (GDPR), for example, possess some challenges and require complex mechanisms to obtain, record and manage consent. Also, data subjects are afraid about improper use of their data. The combination of understanding and relating to the value proposition, consumer trust, and complex consent processes, results in a low opt-in rate for connected product data exchange (e.g. data from cars) and prevents the creation of innovative services (e.g. connected vehicle insurance programs, or smart city solutions).

Thus, we have conceptualised the smashHit system solution as a trusted, secure and integrating privacy-by-design reference Framework to simplify the consent/contract process, as well as to enable consent/contract tracing and sharing among multiple data platforms. In addition, smashHit will offer solutions to identify data misuse as well as to support stakeholders in the creation of legally binding contracts.

All along, we have followed the maxim to think about the needs of data subjects and data customers, but also to win CP manufacturers (e.g. car makers) to open up their products, by designing a convincing trustworthy ecosystem.

During the project lifetime we have finalized the smashHit system concept, its detailed specification and development by our software and RTD development partners and created a working prototype capable of scale. Several public presentations of our smashHit system concept and results have been presented (see smashHit project website).

In this public report you will find details about the optimised smashHit framework. It will cover the integration of services like the Identity Manager, the Automatic Contracting Tool, the Context Sensitivity Solution, the Data Use Traceability, the Privacy & Security Tool and the graphical user interface smashHit Web Application with the smashHit platform.

This report is based on internal deliverables produced during the development of the smashHit project, in particular the deliverable *D5.3 Optimised smashHit Framework* and its previous versions.

If you got curious about how all that is made possible, just continue on the following pages, enjoy the reading, and please contact us with your feedback or questions!

- smashHit support email: info@smashhit.eu
- smashHit project website: https://smashhit.eu

Executive Summary

The objective of smashHit is to assure trusted and secure sharing of data streams from both personal and industrial platforms, needed to build sectorial and cross-sectorial services, by establishing a Framework for processing of data subject consent and legal rules and effective contracting, as well as joint security and privacy preserving mechanisms. The vision of smashHit is to overcome obstacles in the rapidly growing Data Economy which is characterized by heterogeneous technical designs and proprietary implementations, locking business opportunities due to the inconsistent consent and legal rules among different data-sharing platforms actors and operators. The Framework will provide methods and tools (depicted in Figure 1), such as

- a set of loosely coupled, collaborating services that expose their functionality through a REST API, and
- the smashHit core component which coordinates the services and exposes the functionality through a REST API to the smashHit Web Application and the Business Case Applications.

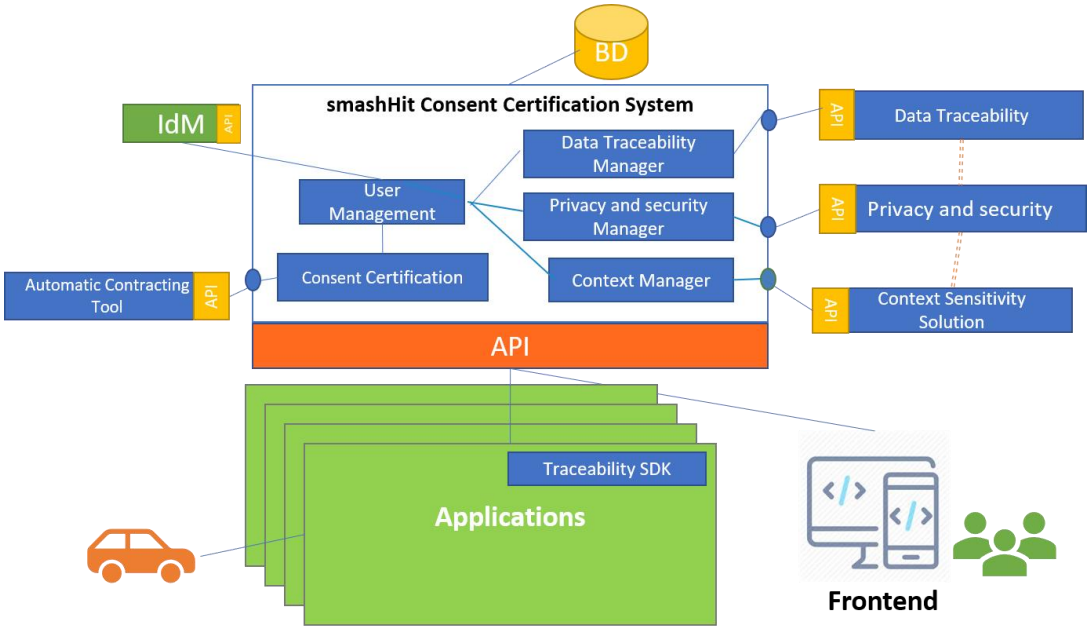


Figure 1: smashHit Framework architecture

The development cycles of the features have evolved so that initially the core functionality was implemented and integrated into the smashHit platform. Subsequently, additional functionalities were developed, that have extended and improved the basic functionality, resulting in the Full Prototype of the smashHit framework. Individual features and the integrated smashHit platform as a system were further developed and optimised based on feedback from the Business Cases using the smashHit framework features.

This report is based on internal deliverables produced during the development of the smashHit project, in particular the deliverable *D5.3 Optimised smashHit Framework* and its previous versions.

Table of Content

List of Figures

1 User Management Component with Identity Manager Integration

The User Management component as shown in Figure 2 is responsible for the life cycle of the actors involved in the smashHit framework. The functionality includes actor creation, update of actor's information, actor deletion, actor validation, and user/application authentication and authorization. The actor may be an end-user, an application or an organisation user.

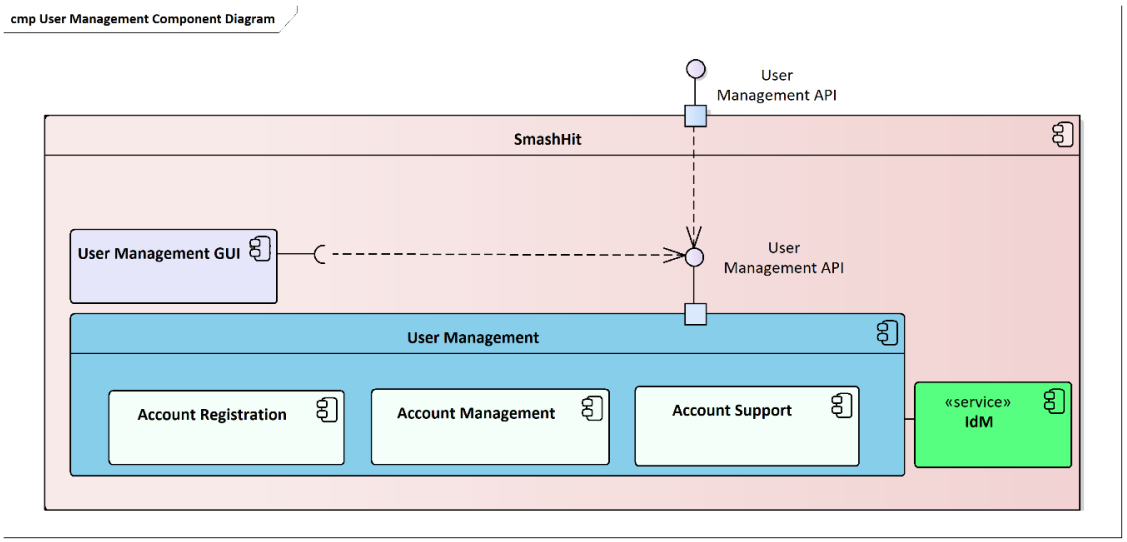


Figure 2: User Management component

The FIWARE Identity Manager (IdM) "Keyrock[1]" is used for providing the User Management functionality and OAuth2[2]-based authentication and authorization to protect services and applications. Organizations can choose whether to default to the bearer tokens and rely on Keyrock for validation or use of JSON Web Tokens (JWT)[3] and handle the token in the client-side.

The functionalities which are utilized from the IdM include:

- User registrations and logins
- Account management: user management and credential renewal
- Mapping organizations' users with smashHit users
- User authentication at login
- User validation for smashHit internal modules

The entire functionality of the component is exposed via a REST API to interact with applications, internal smashHit components and the smashHit Web Application. The API is a RESTful web service over the HTTPS protocol that uses JSON data serialization formats. The OpenAPI specification[4] of the API is at https://smashhit.ari-mobility.eu/swagger/.

The OpenAPI specification of the API of the IdM is at https://smashhit.ari-mobility.eu/swagger-idm/.

[1] https://fiware-idm.readthedocs.io/en/latest/
[2] https://oauth.net
[3] https://jwt.io/
[4] https://swagger.io/specification/

Future considerations

While the selected Keyrock Identity Manager provides the required functionality to support the User Management component, also other alternative OAuth2[5] compliant Identity Manager implementations that could bring additional features to improve the user experience with regard to account management, authorization and identification have been studied.

In this sense, Keycloak[6] has been identified as a robust alternative Identity Manager to Fiware's Keyrock. This open-source Identity Manager, sponsored by Red Hat, is widely used in the community due to its advantages compared to other free options.

While Keyrock has limited or even no support for the following features that would be needed, Keycloak on the other hand does support all of them either natively or through custom configuration of the service:

- Two-factor authentication
- Additional account and password management features, such as temporary credentials or one-time password
- Strong identification integration of BankId[7] (supported through custom Identity Providers)
- Additional token-related security mechanisms, such as asymmetric key signing for JWT and automated key rotations
- Further control of the token lifecycle
- Support for social login (i.e., Facebook and Google) and user federation out of the box
- Additional UI customization options allowing for a better branding support

[5] https://oauth.net
[6] https://www.keycloak.org/
[7] https://www.bankid.com/en/

2 Consent Certification Component with Automatic Contracting Tool Integration

The Consent Certification component as shown in Figure 3 handles the life cycle of the consent certificates in the smashHit framework. The functionality includes consent creation, notification to the parties involved, consent revocation, consent validation, consent management, and accountability.

The component also handles different contracting functionalities such as creating and deleting contracts, performing contracts compliance, creating, and deleting contract specific obligations and retrieving contractors. It also handles i.e. the contract signatures and the contract terms.

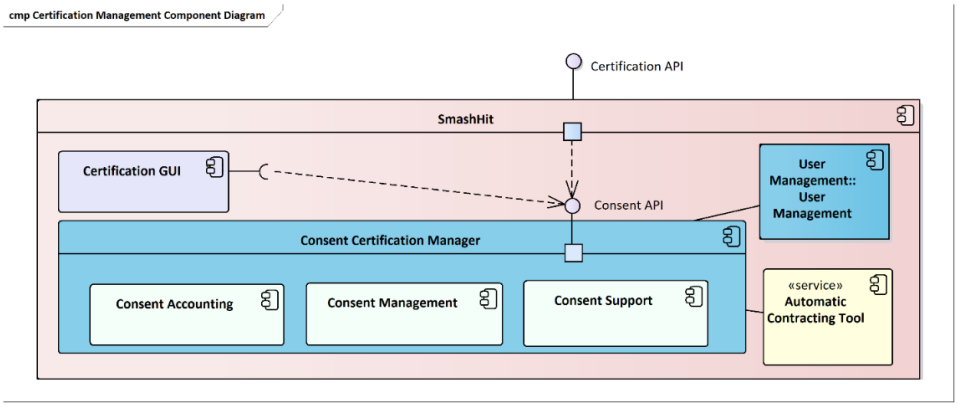


Figure 3: Consent Certification component

To provide the consent certificate functionality, the component will mainly use the services of the Automatic Contracting Tool (ACT).

The ACT supports the automatic generation of consent documents and contracts in compliance with GDPR by utilising semantic technology, namely knowledge graphs (KGs). The use of KGs, for example, provides benefits such as interoperability. In simple terms, KGs enable a common understanding. KGs therefore provide the needed uniformity in the consent and the contract representation.

List of the ***consent*** functionalities which are integrated with the ACT:

- Consent certificate creation
- Consent certificate revocation
- Consent certificate change
- Consent certificate validation
- Consent certificate query

List of the ***contract*** functionalities which are integrated with the ACT:

- Semi-automatic contract creation and annotation in the legal knowledge graph (KG).
- Integration of Consent Certification component with contracts for consent granting and revoking
- Automatic GDPR-compliant contract document generation
- Contract modelling, specifically terms & conditions and obligations with KG
- B2C and B2B contracts modelling and implementations
- Breach of contracts GDPR compliance verification
- Traceability of contracts within the KG via relationships between concepts

The entire functionality of the component is exposed via a REST API to interact with applications, internal smashHit components and the smashHit Web Application. The API is a RESTful web service over the HTTPS protocol that uses JSON data serialization formats. The OpenAPI specification[8] of the API is at https://smashhit.ari-mobility.eu/swagger/.

The OpenAPI specification of the API of Automatic Contracting Tool Consents is at https://act-tool.sti2.at/swagger-ui.

The OpenAPI specification of the API of Automatic Contracting Tool Contracts is at https://act-tool.contract.sti2.at/swagger-ui/.

Optimisation activities

Layered encryption was identified as one of the most significant bottlenecks in the earlier prototype. The increase in the number of layers results in a higher level of security and a slower response time. This is a well-established trade-off. Consequently, optimisation was performed to reduce processing and response times in accordance with the requirements of the Business Cases. To achieve this, the layers of layered encryption are reduced to a single layer without compromising security. Currently, the same RSA (Rivest–Shamir–Adleman)[9] encryption is employed as previously. In addition, additional optimisations were conducted. For example, the interaction with the Security module was optimised, for instance, by implementing concurrent asynchronous requests. An overall improvement of approximately 2.5 seconds has been made.

Future considerations

Since the free version of GraphDB[10] impose limitations on the number of concurrent queries, one way to improve performance would be to take a premium (or licensed) version of GraphDB into use.

[8] https://swagger.io/specification/
[9] https://en.wikipedia.org/wiki/RSA_(cryptosystem)
[10] https://graphdb.ontotext.com/

3 Data Traceability Component

The Data Traceability component as shown in Figure 4 allows the tracking of the data use and exchange among organizations and applications to provide transparency for users. By utilizing the Data Traceability component, the user will be able to understand when data was sent and when it was received together with additional information related to the transfer, thus giving the user more confidence to engage in data sharing.

The Data Traceability has two major components: Traceability Manager and Traceability Module. On the server side of the data traceability functionality, the Traceability Manager component co-ordinates the workflow using the Data Use Traceability service.

The Traceability Module is integrated as a client in an organization's system. After the initial registration, the organization can use the interface provided by the Traceability Manager to send information regarding data transactions directly to the Manager.

The Traceability Manager is in charge of processing requests coming from different Traceability Modules and creating the logs. The users can see the generated logs containing the transaction information on the Web interface of the smashHit platform.

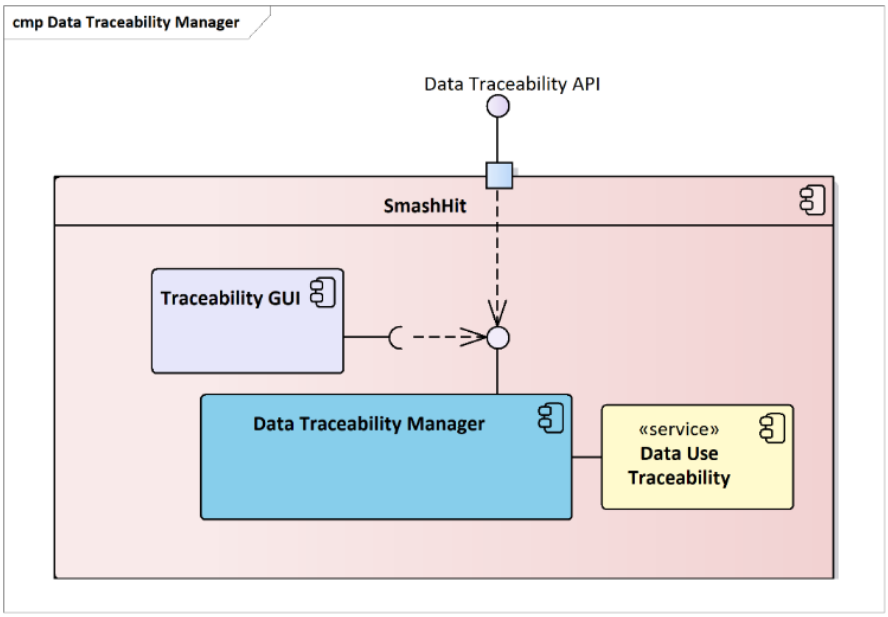


Figure 4: Data Traceability Manager component

The Data Use Traceability service will mainly provide end-to-end data use traceability using metadata for indicating data transfer and by logging each data transaction.

The functionalities which are provided by the Data Use Traceability service include:

- Register data
- Notify data transfer
- Verify received data
- Get data trace

The entire functionality of the component is exposed via a REST API to interact with applications (mainly via the local Traceability Module), internal smashHit components and the smashHit Web Application. The API is a RESTful web service over the HTTPS protocol that uses JSON data serialization formats.

Further, we tackle the risk of data leakage by providing re-identification functionality within the Traceability Module. The standalone re-identification functionality allows the organizations to easily and quickly re-identify data locally in their systems using watermarking or fingerprinting techniques. As vehicle trajectories inherently contain private data and can easily be modified to obscure the data origin, we use a novel watermarking technique for watermarking GPS trajectories called W-Trace.[11]

Figure 5 presents an example describing the watermarking process of W-Trace for a GPS trajectory. First the trajectory is partitioned into sub-trajectories and the watermark is inserted in each sub-trajectory. Further, all watermarked sub-trajectories are unified to obtain the final watermarked trajectory. In this way, if the data is later on modified by a third party, it will be possible to verify the presence of the watermark and understand that there was a data leakage.

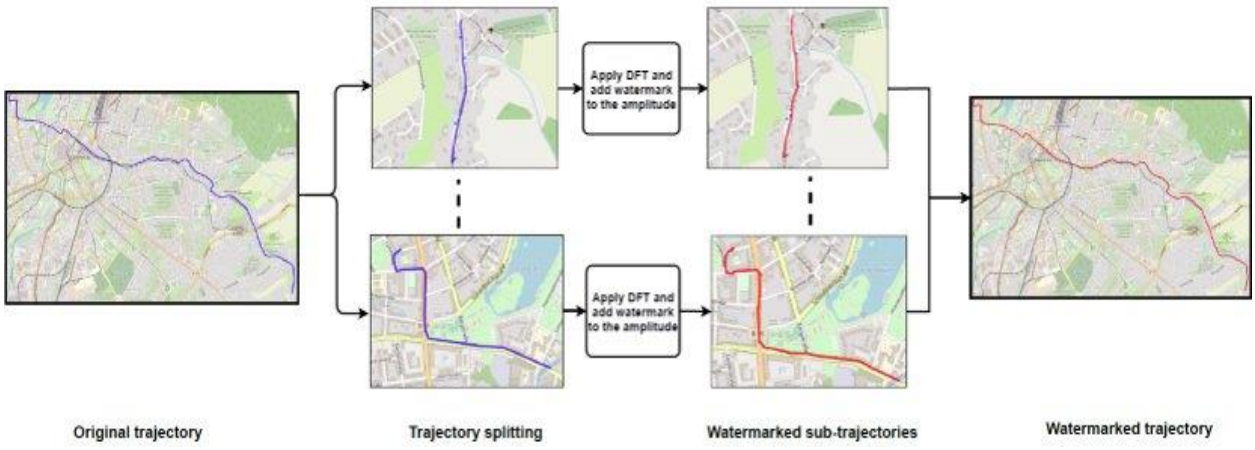


Figure 5: Watermarking process for a GPS trajectory[12]

Optimisation activities

The Traceability component is now fully integrated into smashHit and, specifically the Data Traceability Manager, in charge of processing requests coming from different Traceability Modules and creating the logs. The integration is made by connecting the central smashHit endpoint for traceability (https://smashhit.ari-mobility.eu/api/traceability) to the deployed endpoint (https://smashhit.l3s.uni-hannover.de/) to enable the forwarding of incoming requests. Note that the endpoint can be accessed by using a suitable API tool like Postman[13] with compliance to the OpenAPI specification.

The communication between the Traceability Module and the Data Traceability Manger was improved by adding an endpoint to check the presence of an actor using the Traceability component. Via the designed procedure, it is now ensured that the deployed Data Traceability Manager is informed through the provided smashHit central API about operations happening between partners, and it creates the logs. The generated logs can be shown to the user through the smashHit Web Application.

[11] Dadwal R., Funke T., Nüsken M., Demidova E. (2022). "W-Trace: Robust and Effective Watermarking for GPS Trajectories". In Proceedings of the 30th International Conference on Advances in Geographic Information Systems. ACM

[12] Map data: ©OpenStreetMap contributors, ODbL

[13] https://www.postman.com/

For the re-identification task, the part of verifying the authenticity of leaked data based on a neural-network-based framework was implemented. Activities to optimise the standalone re-identification component have been carried out to allow the companies to easily and quickly re-identify the data locally in their system using watermarking or fingerprinting techniques. It is now also possible to easily check if a given data was leaked.

Future considerations

In addition, similar functionalities of the Data Traceability component in production and business case environment will be verified in later tests, where data collected via the City Feedback App will be transferred from Infotripla's system to Forum Virium Helsinki.

4 Context Manager Component with Context Sensitivity Solution Integration

The Context Manager component as shown in Figure 6 provides information to the Context Sensitivity Solution (CSS), with the aim to obtain contextual information from the data sources available in smashHit. The data sources of smashHit for the CSS are primarily data evolving from the user preferences of data subjects. The CSS provides a list of context configuration options concerning the general data use, e.g., types of applications that may use his data for the data subject to choose from, and e.g., time scheduling types of restrictions which the data subject can set to his data use.

The main functionalities of the CSS are data ingestion, context monitoring, and context extraction and provision. It makes the extracted context available to other smashHit components, i.e., the smashHit platform and the Security & Privacy (S&P) component, obtained by applying ontology-based reasoning rules and extraction techniques.

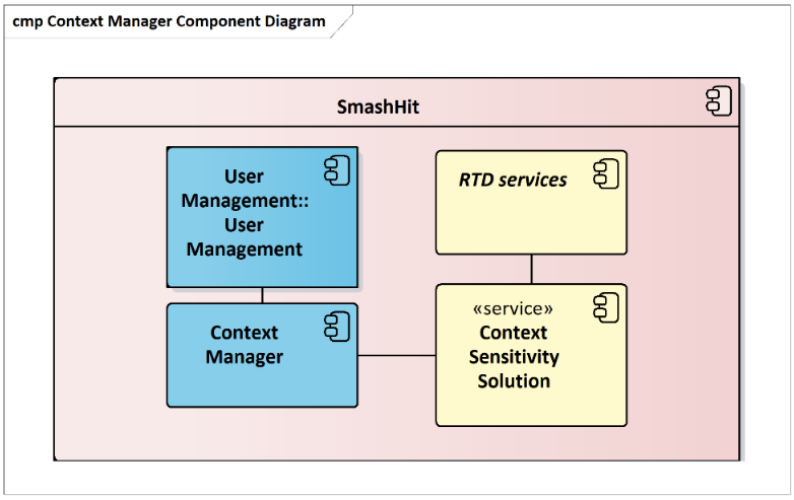


Figure 6: Context Manager component

Following the overall architecture of the CSS, as shown in Figure 7, the following functionalities have been integrated with the CSS:

- ***Data ingestion and context monitoring***: context configuration based on the user preferences
- ***Context extraction and provision***: user preferences context change extraction, provision of consent conflicts and continuous update of the specified policy context variables of the smashHit Privacy and Security Mechanisms (i.e., NGAC Tool)

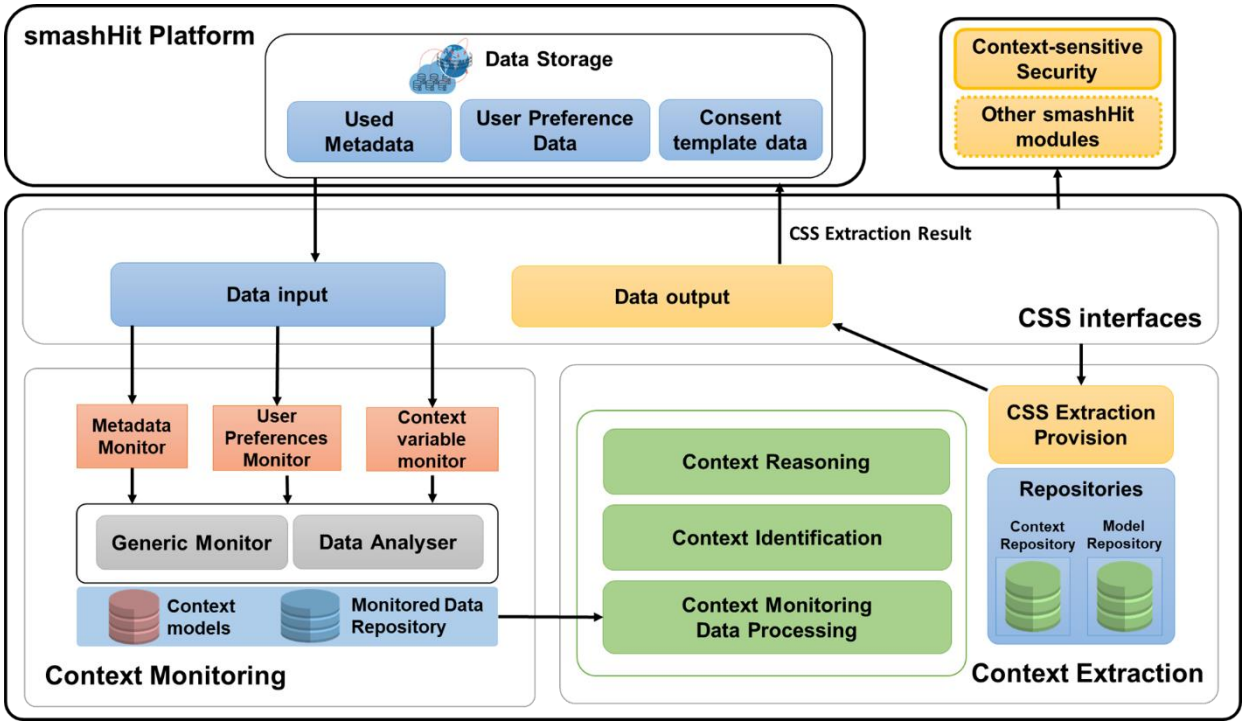


Figure 7: CSS general architecture overview

The entire functionality of the component is exposed via a REST API to interact with applications and the smashHit Web Application. The API is a RESTful web service over the HTTPS protocol that uses JSON data serialization formats. The OpenAPI specification[14] of the API is at https://smashhit.ari-mobility.eu/swagger/.

The OpenAPI specification of the CSS API to interact with internal smashHit components is at https://app.swaggerhub.com/apis-docs/smashhit-atb/smashHit-Context-Sensitivity-Solution-OAS3.0/1.0.0#/.

Optimisation activities

Overall, for the optimised solution, we have aimed to validate the configuration and use of the user preferences. Based on the feedback from the testing activities of the Business Cases, an overall performance optimisation has been done as described in the following. Table 1 shows the technical optimisation key performance indicators (KPIs) and its measurements for the optimised CSS solution, in comparison with the measurements of the Full Prototype (FP) before optimisation.

Table 1: Optimisation KPIs and their measurements for CSS

KPI	Validation measure	Measurement FP	Measurement Optimised
Performance	Every API call should last less than 1s	~ 30ms in average	~ 24ms in average
	Execution times: Context monitoring and data ingestion		~ 20% faster query execution and response times
	Execution times: Context extraction		~ 15% faster extraction algorithm(s)

[14] https://swagger.io/specification/

Data integrity and flawless functionality	Error-free execution		as implemented in FP
	Exceptions are sent as responses		as implemented in FP
	Underlying data remains of integrity		as implemented in FP
Stability and accessibility	Uptime: 99% of the requests succeed	Server uptime ~ 99%	Server uptime ~ 99,3%
Information security and secure access	Authorization token	n.a.	Authentication token for context provision REST APIs has been introduced.

Context monitoring and data ingestion

The context monitoring repository and its underlying database implementation has been code-optimised in order to achieve faster query execution and response times.

Figure 8 and Figure 9 show execution times for the context monitoring and data ingestion component before and after the optimisation.

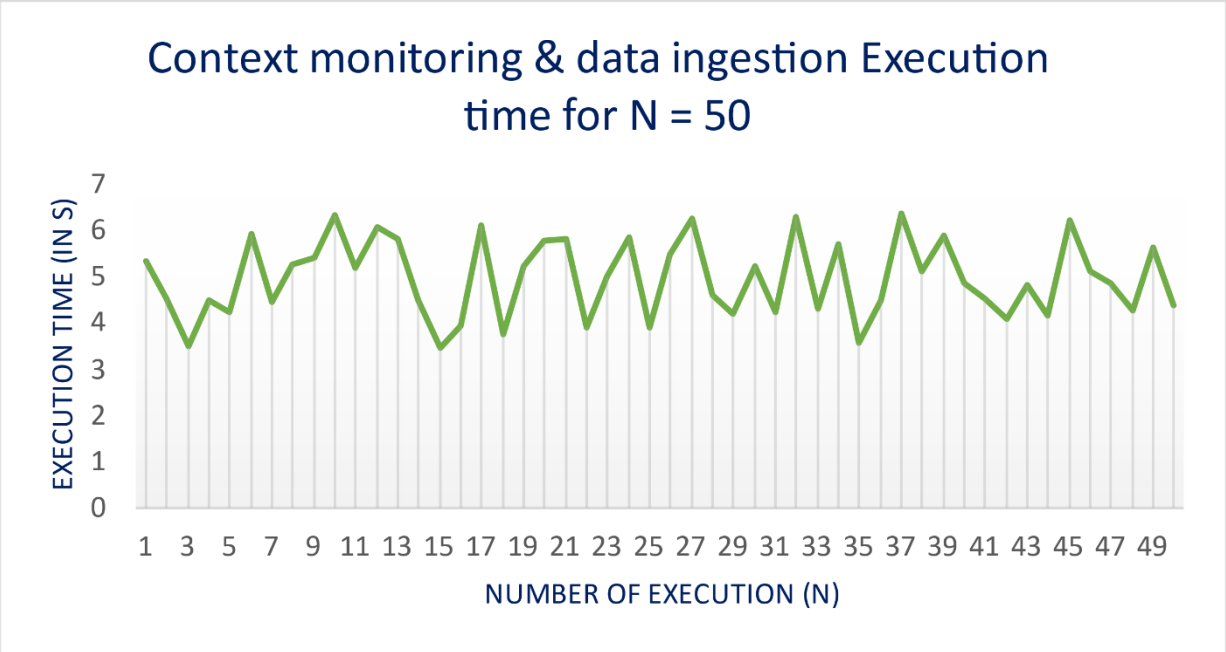


Figure 8: Execution times for the context monitoring and data ingestion for n = 50 executions, non-optimised

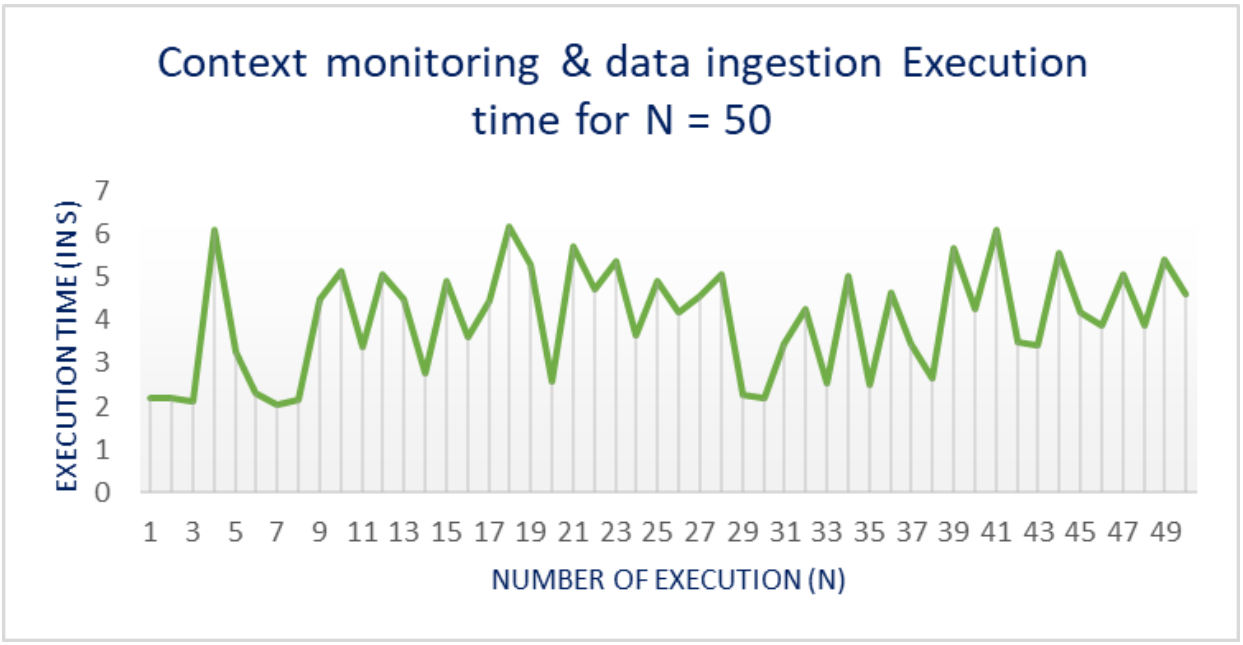


Figure 9: Execution times for the context monitoring and data ingestion for n = 50 executions, optimised

Context extraction and provision

The context extraction algorithms for calculating a data subject's conflicts with regard to Purpose and Sector preferences have been code-optimised in order to achieve faster calculation times. As a result, the average response time for the extraction result has been improved by about 15%, also for high-load requests.

Figure 10 and Figure 11 show execution times for the context extraction process before and after the optimisation.

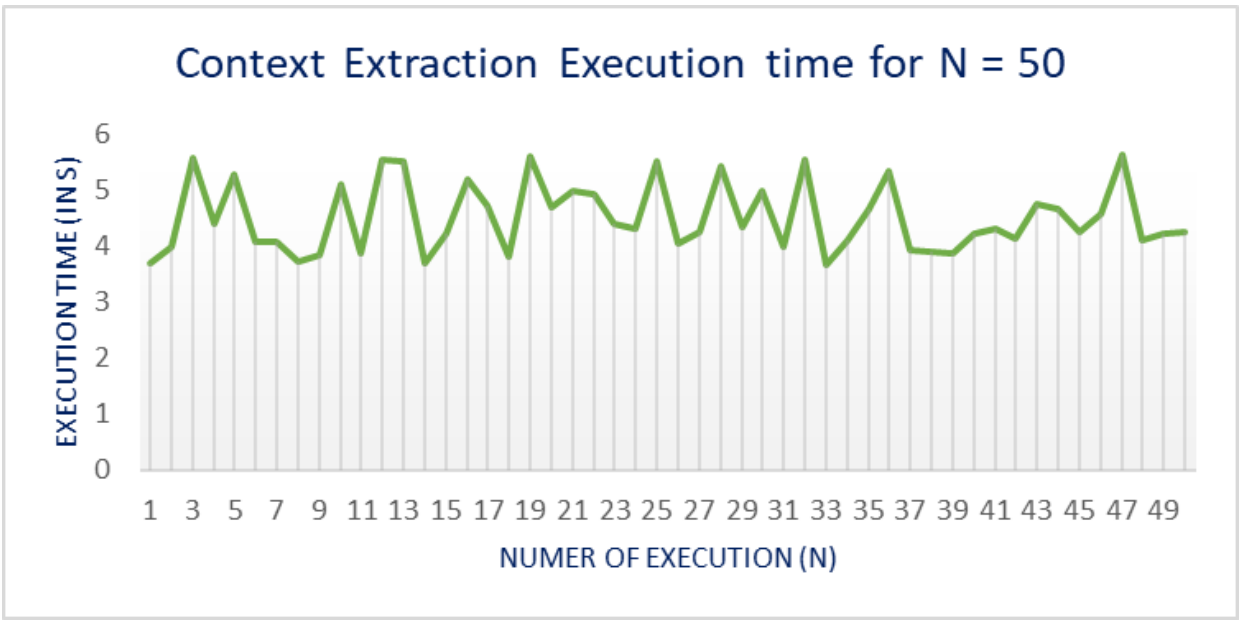


Figure 10: Execution times for the context extraction for n = 50 executions, non-optimised

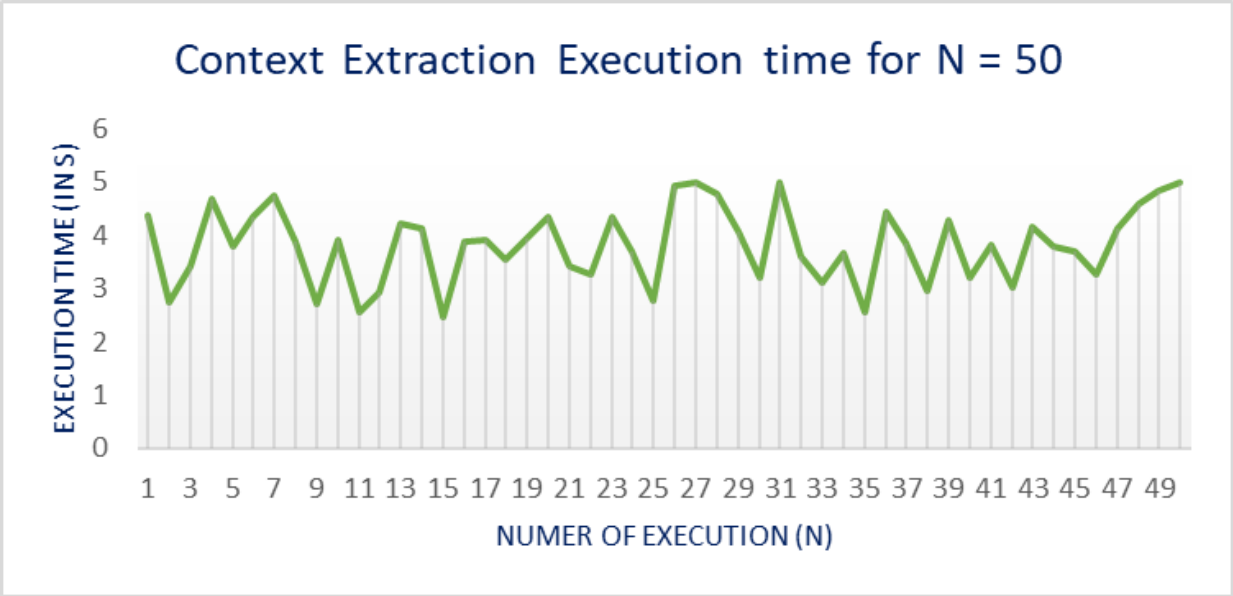


Figure 11: Execution times for the context extraction for n = 50 executions, optimised

Interfaces

The internal implementation of the Data ingestion and context provision REST APIs has been code-optimised in order to achieve faster server response times. As a result, the average response time of an API has been reduced significantly by about 20%, from 30ms in FP to average of 24ms in the optimised solution. At the same time, the overall performance with high-load requests has been improved.

Figure 12 and Figure 13 show the average response times of the context provision API before and after the optimisation.

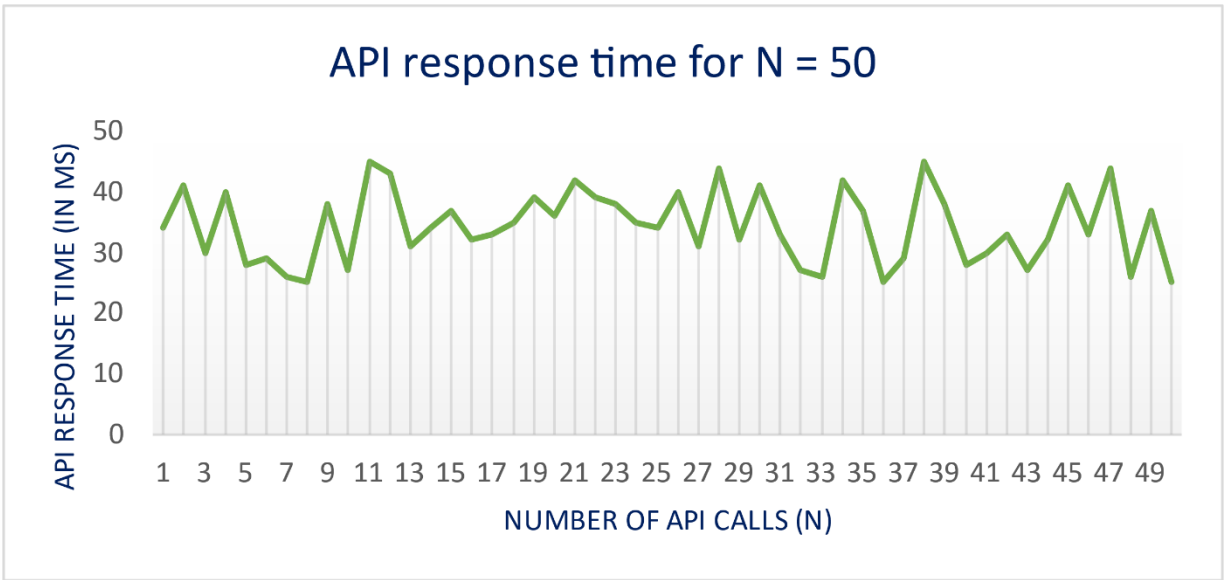


Figure 12: Average response times of the context APIs for n = 50 API calls, non-optimised

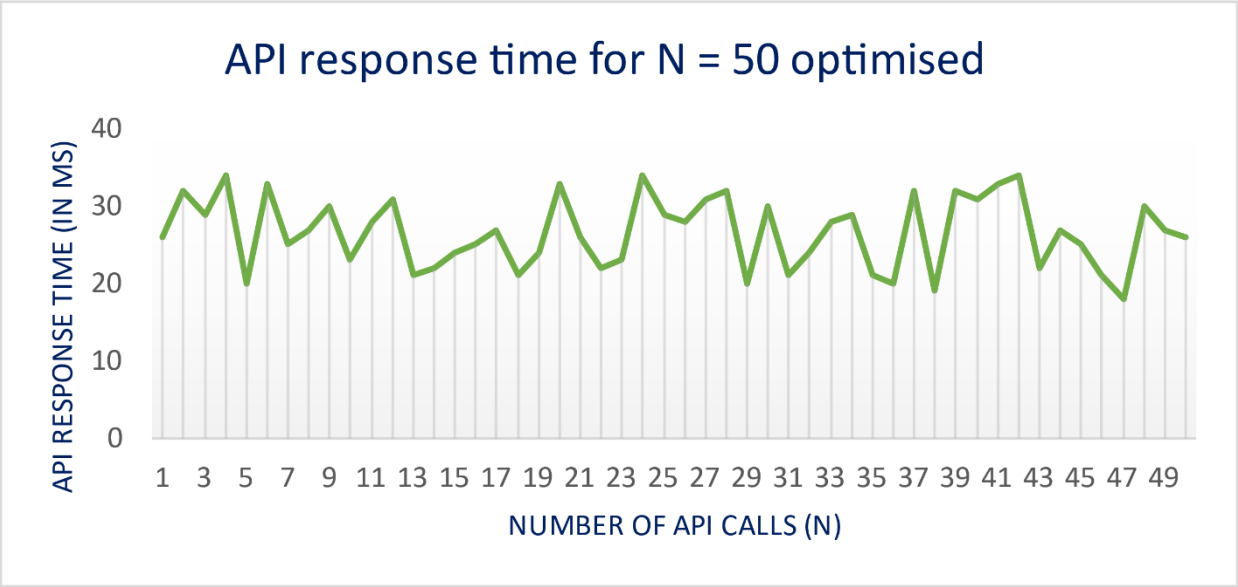


Figure 13: Average response times of the context APIs for n = 50 API calls, optimised

Ontological reasoning optimisation

For our context extraction component, we are using the smashHit Core Ontology[15] as a semantic basis for all extraction operations, together with one of the most used reasoning techniques in the semantic space, which is commonly known as ***Inference Reasoning***. Inference on the Semantic Web can be characterized by discovering new relationships. On the Semantic Web, data is modelled as a set of (named) relationships between resources. "Inference" means that automatic procedures can generate new relationships based on the data and based on some additional information in the form of a vocabulary, e.g., a set of rules.[16] As an example, consider the following two statements:

1. Drivers are defined as persons that drive cars (complete definition)
2. We also know that drivers are adults (partial definition)

From these two statements, we can consider all drivers also to be adult persons (e.g., grown-ups). This simple example can easily be mapped to one of our extraction scenarios, dealing with the categorisation of *Purposes for Data Processing.* From the following abstract of the Purpose hierarchy within the smashHit Core Ontology, we can consider *Research And Development* being a subclass of *Purpose.* At the same time there exist three subclasses of *Research And Development,* i.e. *Academic Research, Commercial Research* and *Non-Commercial Research.*

[15] For more info on the ontology, see our technical essay: https://smashhit.eu/wp-content/uploads/2022/10/TECHNICAL-ESSAY-smashHit_semantic_model_v05.pdf

[16] See also https://www.w3.org/standards/semanticweb/inference

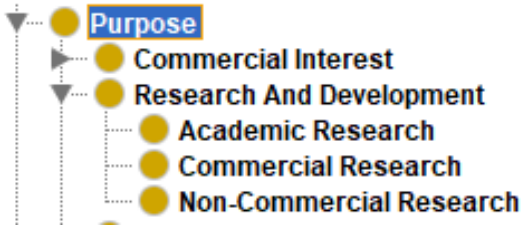


Figure 14: Abstract from the Purpose hierarchy

We can also follow, that *Research And Development* is the parent category for all these three subcategories, which comes in handy when detecting potential consent conflicts with the user preferences within the smashHit system. For the implementation, we are using the *Apache Jena Framework*[17], which allows us to model such class-subclass relationships in a semantically valid way. On top, it provides a quite enhanced toolkit of automated reasoning techniques, which can then be applied to the model. Finally, this implementation choice led to improved execution times of the overall context extraction process.

Future considerations

For future work, we will further perform research in the extending possible extraction scenarios by monitoring not only the user preferences, but also the actual data from the cyber physical products to the extent possible. This could also allow the definition of additional context variables to be used within the smashHit security policy, which could benefit from a more fine-granular control of the Privacy and Security mechanisms and finally lead to enhanced contextual awareness of the data subjects for data sharing (for this also refer to the User Guide - Data Owner).

[17] https://jena.apache.org/

5 Privacy and Security Manager Component with Privacy and Security Tool (NGAC) Integration

The Privacy and Security Manager component as shown in Figure 15 provides a dynamic database of policies, conditions, and event-responses to enforce privacy and security of the smashHit framework. The Privacy and Security Manager provides privacy and security functionality by means of the Privacy and Security Tool (TOG-NGAC[18]).

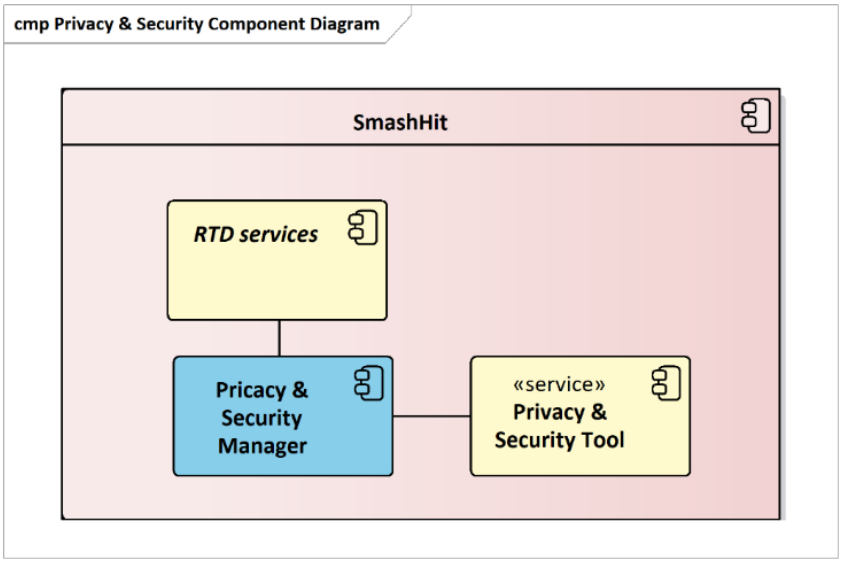


Figure 15: Privacy and Security Manager component

The functionalities which are provided by the Privacy and Security Tool (NGAC[19]) include:

- Access control policy administration.
- Privacy abstraction policy administration.
- Policy query.
- Security auditing.
- Event-Response policy.

During integration, a clear improvement was conceived and a privacy abstraction level that closely followed the GDPR-inspired concepts and vocabulary used in the smashHit project was implemented as the Declarative Policy Language for Privacy (DPLP) interface. This abstraction is built on the foundation of the access control policy administration API but automatically maps privacy constructs to the primitive policy elements of the lower-level API.

An audit API provides common security auditing functionality for the recording of security-relevant events that occur anywhere in the smashHit framework.

While investigating extant privacy policy specification languages we have found that ideas gleaned there to be valuable in defining and implementing the privacy policy supporting features

[18] The Open Group implementation of NGAC

[19] InterNational Committee for Information for Information Technology Standards - Cyber security technical committee, “1. American National Standard for Information Technology – Next Generation Access Control (NGAC),” *ANSI, INCITS 565-2020,* April 2020

in our access control language. Features needed for the present use cases have been implemented in the current DPLP extensions. Future implementation of other related features will further expand the breadth of use cases that can be addressed.

The entire functionality of the component is exposed via a REST API to interact with applications and the smashHit Web Application. The API is a RESTful web service over the HTTPS protocol that uses JSON data serialization formats. The OpenAPI specification[20] of the API is at https://smashhit.ari-mobility.eu/swagger/.

The OpenAPI specification of the TOG-NGAC[21] API is at https://app.swaggerhub.com/apis/tog-rtd/ngac-apis/1.1.1#/.

Optimisation activities

While investigating privacy policy specification languages, we have found that this is still an evolving field, unlike access control policy specification which is relatively mature. We investigated and experimented with privacy policy languages from the research literature and have found that ideas gleaned there to be valuable in defining and implementing the privacy policy supporting features in an access control language. Some of these needed for the present use cases have now been implemented in the current DPLP extensions. Since the completion of the full prototype, during final integration, improvements have been made in error detection and reporting in the DPLP privacy abstraction APIs. Also, a new policy query API (pqapi/policy_sat) has been added to directly compare a DC/DP privacy policy to a DS privacy preference to determine whether the privacy policy satisfies the privacy preference and if not to report the points of non-satisfaction. The policies to be compared may have been previously loaded into the server or may be provided as immediate parameters of the call. This interface could be used by end-user facing UIs to detect points of incompatibility of policies during an initial encounter between a DC/DP and a DS.

Future considerations

Future implementation of other features will further expand the breadth of use cases that can be addressed by the Privacy and Security Tool. An example of such a feature is the incorporation of obligations in the access control and privacy policies so that a “grant” response to a privacy policy query would include a list of the obligations that are associated with the grant. The addition of prohibitions would permit an explicit denial of a permission in contrast with a denial being the result of an absence of any enabling permission. Another useful feature would be a procedure to automatically import changes to the Security and Privacy Tool corresponding to changes that have been made to the smashHit Core Ontology.

[20] https://swagger.io/specification/
[21] The Open Group implementation of NGAC

6 smashHit Web Application

The smashHit Web Application provides a graphical user interface to the smashHit framework. The application allows the organization administrators to register and manage organizations, configure applications, and consent templates. Also, the application allows data subjects to see and manage all their consents granted to applications. Data subjects can also see data traceability information and manage user preferences via the application.

The user and developer guides present the different functionalities available to the three main actors of the smashHit web application: platform administrators[22], organisations acting as data providers or processors[23], and data subjects[24]. The purpose of these guides is to guide the reader step-by-step through the deployment and use of the smashHit platform. The functions are illustrated with screenshots and examples, so we recommend that the reader consult this supplementary material if he/she wants to learn more about the smashHit Web Application.

The smashHit Web Application is implemented using Angular[25] framework and uses data services exposed by the smashHit framework to generate the dynamic web pages and uses the Identity Manager to login users into the application. The OpenAPI specification[26] of the used API is at https://smashhit.ari-mobility.eu/swagger/. The smashHit Web Application is available at https://smashhit.ari-mobility.eu. On interoperability with other actors in the sector, the smashHit consortium has already started contact with organizations such as MyData[27] group and BDVA[28] with plans to work and define a strategy for the platform to be interoperable.

Future considerations

Potential new features not initially foreseen in the requirements that could improve the user experience in the web applications, as well as new functionalities that could bring added value to the smashHit framework, have been analysed. The following is a list of features that have identified improvements:

- ***Identity delegation***: Since the current user interface is meant for data subjects directly, one open research topic would be how to handle consents involving minors, where a tutor would need to intercede, or in general, attorneys or legal representatives.
- ***Branding tools support***: Some of the feedback from the Business Cases pointed out some confusion or even lack of trust from data subjects when they received notifications regarding their consent and account from “smashHit” as a sender, instead of the organization that they signed up through. In order to improve user trust, it should be considered to implement some kind of branding tools so that notifications from smashHit to data subjects can be more easily recognized as associated with the application using smashHit itself.
- ***Localization***: Following what has been stated in the previous point, it should be considered to translate and localize the smashHit Web Application to different languages.
- ***Consent portability***: In order to improve interoperability with other existing and future consent management solutions, further efforts should be made on the portability of consents to facilitate the import or export of user data to similar systems.

[22] https://smashhit.eu/wp-content/uploads/2022/10/USER-GUIDE-platform_administrator_v2.pdf
[23] https://smashhit.eu/wp-content/uploads/2022/10/USER_DEVELOPER-GUIDE-data_provider_processor_v2.pdf
[24] https://smashhit.eu/wp-content/uploads/2022/10/USER-GUIDE-data_owner_v2.pdf
[25] https://angular.io/
[26] https://swagger.io/specification/
[27] https://www.mydata.org
[28] Big Data Value Association (https://www.bdva.eu)

7 Glossary

ACT: Automatic Contracting Tool (used by the Consent Certification component of smashHit)

API: Application Programming Interface

B2B: Business-to-Business

B2C: Business-to-Consumer

BC: Business Case of smashHit Consortium

Consent: As per Article 4(11) of the GDPR, 'consent' of the data subject means any freely given, specific, informed and unambiguous indication of the data subject's wishes by which he or she, by a statement or by a clear affirmative action, signifies agreement to the processing of personal data relating to him or her.

CSS: Context Sensitivity Solution (used by the Context Manager component of smashHit)

DPLP: Declarative Policy Language for Privacy

FP: Full Prototype of smashHit framework

GDPR: Abbreviation for 'General Data Protection Regulation', a legal norm on EU level adopted in 2016, which is directly applicable within its scope and lays down rules for the processing of personal data so as to protect natural persons' fundamental rights and freedoms, in particular their right to the protection of personal data.

GPS: Global Positioning System

HTTPS: Hypertext Transfer Protocol Secure

IdM: Identity Manager (used by the User Management component of smashHit)

JSON: JavaScript Object Notation

JWT: JSON Web Tokens

KG: Knowledge Graph

KPI: Key Performance Indicator

NGAC: Next Generation Access Control (utilized in the Privacy and Security Tool of smashHit)

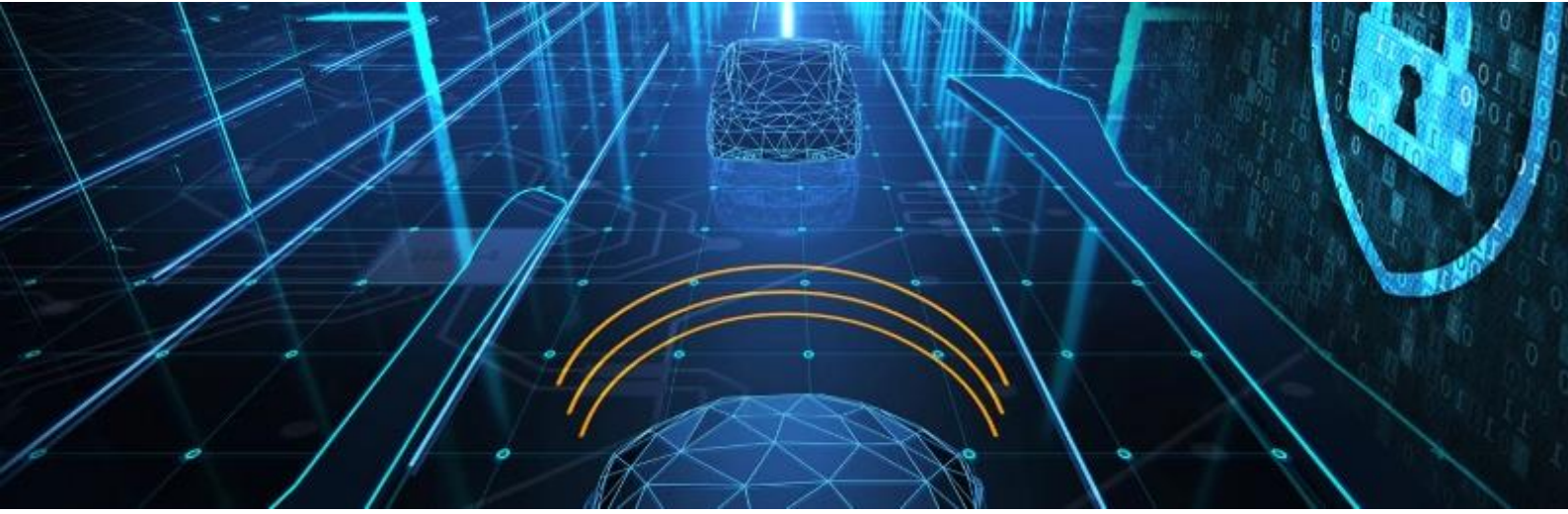
Personal data: Any information which are related to an identified or identifiable natural person. (GDPR Art.4 (1))

REST: Representational State Transfer

RTD: Research and Technological Development

TOG-NGAC: The Open Group implementation of NGAC (used in the Privacy and Security Tool of smashHit)

UI: User Interface



Our vision - Solving Consumer Consent & Data Security for Connected Car and Smart City

smashHit

Further information

More information about smashHit, recent blog posts, the publications created within the project and other material like white papers and guidelines for users and developers can be found on our project website:

https://smashhit.eu

Our consortium

ATB

universität innsbruck
Department of Computer Science

Leibniz Universität Hannover

LexisNexis RISK SOLUTIONS

THE Open GROUP

infoTripla

UNIVERSITÄT BONN

arctic
a brand of aebi schmidt

Atos

FORUM VIRIUM HELSINKI

Funded by the Horizon 2020 Framework Programme of the European Union