privacyTracker: A Privacy-by-Design GDPR-Compliant Framework with Verifiable Data Traceability Controls

Harald Gjermundrød(✉), Ioanna Dionysiou, and Kyriakos Costa

Department of Computer Science, School of Sciences and Engineering,
University of Nicosia, Nicosia, Cyprus
{harald,dionysiou.i}@unic.ac.cy,
kyriakoskosta@gmail.com

Abstract. Breach or lack of online privacy has become almost a commonplace of today's digital age, mainly due to the inability of either enforcing privacy requirements or imposing strict sanctions against violations. The current state of affairs in data privacy is at a turning point for companies operating in EU state members as the enforcement of the General Data Protection Regulation (GDPR) empowers users with control over their personal data, including regulating its disclosure, withdrawing disclosure consent at any given time and tracking their data trail. Compliance with the GDPR is mandatory and it requires signifiant amendments and/or restructuring of data processing routines undertaken by enterprises. Currently, there is no framework to support the GDPR principles. This paper proposes privacyTracker, a GDPR-compliant framework that supports basic GDPR principles including data traceability and allowing a user to get a cryptographically verifiable snapshot of his/her data trail.

Keywords: User privacy · Data traceability · General Data Protection Regulation (GDPR)

1 Introduction

With the proliferation of digital technologies and the growing trend of digitizing all kinds of records (*e.g.* business, academic, medical, government) concerns over privacy issues are raised not only by organized groups but also by average users of technological solutions, who have a keen interest in the processing and handling procedures of personal data by organizations. According to the 2015 TRUSTe US Consumer Confidence Index [1], 92 % of the respondents worry about their privacy online, revealing as the top cause of concern the companies collecting and sharing personal information with other companies. Consumers want to be informed on how their personal data is used as well as be allowed to stop being contacted by third parties (30 %). Almost half of the respondents stated the need of clear procedures for removing personal information.

S. Casteleyn et al. (Eds.): ICWE 2016 Workshops, LNCS 9881, pp. 3–15, 2016.
DOI: 10.1007/978-3-319-46963-8_1

Privacy, as defined by Westin [2], is the *"claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others"*. Personal data protection is of utmost importance and must be safeguarded, especially online. Usually, online privacy is expressed as privacy policies posted on sites that outline what data is collected, why is collected and how it is used. However, more often than not doubt is cast on their effectiveness. Reasons include, among others, the complexity of the policies themselves that could create more confusion than clarification and the lack of awareness among users with regard to privacy matters. Furthermore, even though the privacy policies are available to the users, there could be a discrepancy between policy statements and their actual implementation. As a consequence, the user is at no position to *verify* that his privacy is properly handled by an organization.

Serious steps should be taken to offer guarantees for user data protection, especially in the light of the new European Council General Data Protection Regulation (GDPR) [3] that was approved in December 2015. Many businesses, most likely, will need to change their data processing practices to conform with the GDPR principles, which empower users not only with the *control* of their own personal data but also with practical *certainty* of their desired access controls. The control extends to include the *right to erasure*, where the user has the right to request erasure of personal data related to him/her under certain conditions. Technical measures must be in place to manage proper data collection and processing, including mapping legal requirements to policies, mapping policies to technical mechanisms, requiring explicit user consent for all collected personal data, updating user personal data to maintain its accuracy, disclosing personal data according to user control preferences, providing personal data traceability upon user request, certifying an enterprise as GDPR-compliant, and honoring the right to erasure, where the user has the right to request erasure of personal data related to him/her under certain conditions. The technical implementation of all GDPR requirements is not trivial, as it requires a complicated framework that maps the legal requirements into technical mechanisms and measures.

As of today, to the best of our knowledge, there is no such framework in place (data protection by design) that complies with the GDPR principles of data collection and processing. Furthermore, there is no compliance checking procedure to oversee the adherence to the regulation policies. Inspired by the GDPR, an ecosystem is proposed in this paper, that supports the collection, trade, and distribution of personal and other consumer data along the lines of the GDPR. At the same time, the ecosystem allows enterprises to create trusted relationships with their consumers based on transparency and verifiable proofs, when required, and remain relevant in the emergent sharing economy. To be more specific, the paper contributions are twofold: presenting the design principles of a GDPR-compliant framework that handles data processing by enterprises and discussing their practicality via the Implementation of privacyTracker, a privacy-by-design GDPR-compliant system.

The remainder of this paper is as follows. Section 2 gives an overview of personal data protection in terms of policies and legislation. Section 3 introduces privacyTracker, a novel framework compliant to GDPR principles and Sect. 4 presents a privacyTracker prototype. Section 5 concludes the paper.

2 Personal Data Protection Overview

The common approach, followed by organizations and companies, to user data privacy is the use of privacy policies. These are usually posted on the organization's main site or are presented to the user, who in turn has to give consent before allowed to proceed with a transaction. There is a plethora of research efforts on privacy policies mostly focusing on (1) formalizing privacy policies that could be analyzed for illegal disclosure and potential conflicts, (2) investigating the effectiveness of privacy policies, (3) privacy policy compliance frameworks and (4) provenance of data [4–8].

The absence of privacy policies or their failure to comply to data protection directives and legislations often lead in violation of user privacy. Additionally, the uncontrolled sharing of information and their aggregation from various sources pose non-negligent threats to user privacy as it yields in constructing user profiles without the user's consent. The examples below demonstrate that indeed privacy policies are no silver bullet in safeguarding one's privacy:

- **Absence of privacy policies:** a recent example comes from an audit of the websites of the 2016 US presidential candidates, conducted by the Electronic Privacy Information Center (EPIC), that found out 4 sites had no stated privacy policy at all [9] and several others did not state their data disclosure practices.
- **Violation of Privacy Regulations:** On February 2015, a report that has been commissioned by the Belgian Data Protection Authority found that Facebook is acting in violation of European law [10]. According to the report, *users are offered no choice whatsoever with regard to the sharing of location data.*
- **Potential Violation of Privacy Regulations:** Security firm AVG can sell search and browser history data to advertisers in order to "make money" from its free antivirus software, a change to its privacy policy has confirmed. The updated policy explained that AVG was allowed to collect "non-personal data", which could then be sold to third parties. The new privacy policy came into effect on 15 October 2015, but AVG explained that the ability to collect search history data had also been included in previous privacy policies, albeit with different wording.

Even in the case where privacy policies are enforced and accurately translated into actual implementation statements that do not compromise the stated privacy, still the user is not aware of his/her personal and other data distribution. There is no practical mechanism that permits the active participation of users in carrying out a formal inquiry on the *whereabouts* of their personal data collected by organizations. This is a serious flaw in the current data privacy frameworks.

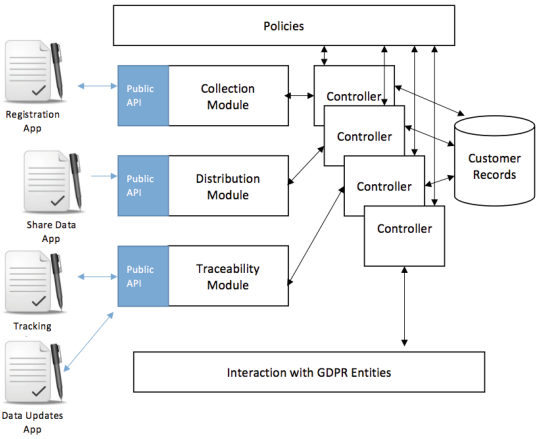

Fig. 1. privacyTracker framework

The current state of lack of accountability when it comes to preserving personal data privacy is about to change as the European Commission General Data Protection Regulation (GDPR), put forward in 2012, attempts to reform the data protection rights across the European Union. An agreement of the proposed regulation was reached on December 2015 and, once it receives formal adoption by the EU parliament and council, its rules will be in effect after 2 years. The GDPR will replace the existing legal framework Directive 95/46/EC and it aims to strengthen citizens' rights to data privacy by giving them control over their personal data.

Any framework that adheres to the GDPR principles must, at a bare minimum, satisfy those data processing requirements (Articles 5(1a), 5(1d), 6(1a), 6(1c), 7(1), 7(3), 12(1), 12(2), 14(1a), 14(1ac), 14a(2g), 15, 16(1), 17(1), 17(2a), 17a(1), 18(2), 19(2)) where the enterprise is obligated to provide undisputed evidence on the handling and sharing of consumer data. This involves addressing the following issues regarding the data in question:

1. be able to accurately set the data collection time and the identity of the collector
2. be able to provide a list of all entities that posses a copy of the original data
3. be able to determine modifications on the data, if any
4. be able to determine the data accuracy and validity, with mechanisms on how to address inaccuracy and invalid data
5. be able to configure the data lifetime, with controls to allow data owners to request data to be erased (right to be forgotten)

Currently, it is nontrivial to get answers to any of the inquiries stated above (except perhaps the first one). Reasons include, among others, the lack of technical solutions, inadequate mandatory legal frameworks that support privacy regarding citizen data and in some cases, lack of interest from the citizen himself on privacy matters. The presented research effort addresses the first obstacle, that of insufficient technical approaches.

3 privacyTracker - A GDPR-Compliant Framework

This section presents the design and implementation details of *privacyTracker*, a privacy-by-design framework that addresses the GDPR data processing requirements. This work follows similar ides to how [11] addressed the involvement of the citizens in an eGovernment setting. Figure 1 depicts the main modules of *privacyTracker*. Details on the main 3 modules are given below (Collection, Distribution, Traceability), along with information on the auxiliary data structure, the Customer Record, which is the core building block of *privacyTracker*. Any framework compliant with the GDPR principles must be policy-driven, thus configurable. This explains the presence of the Policy module that governs the data collection, distribution, and management procedures. Furthermore, provision for interactions with other GDPR entities such as supervisory authorities, data protection officer and the European data protection board could be integrated in the framework.

3.1 Customer Record

The main auxiliary data structure of privacyTracker is the *Customer Record*, a multi-linked list of records that keeps user data encoded in the XML data formate, conforming to the definition of the XML Schema Definition Language (XSD). The advantage of using the open standard self-describing data formate is its portability, thus ease of integration with other applications. The *Customer Record* fields are organized in two sections, the mandatory metadata section and the optional section. The metadata section is comprised of record identification fields, data tractability fields as well as cryptographic controls to ensure data integrity, authenticity, and nonrepudiation. The optional section consists of user public data, user private data that user consent was given for disclosure, data provided by the enterprise itself, to just name a few optional fields. The *Customer Record* metadata fields are defined as follows:

Record Identification

- URI (Unique Resource Identifier) - string concatenation of company name, user email address and auto-generated random identifier. This value is unique within the entire framework, but changes whenever the record is distributed to another entity. Thus, a user may be associated with several URIs.
- User Email Address - could be replaced by a digital signature in the future.
- Genesis Time - timestamp of the initial creation of the record. This value is immutable throughout the framework.
- Creation Time - timestamp of the creation of the record locally. This value is mutable as each company, upon receiving a record, creates a new one locally.
- Expiration Time - record data is considered outdated after this time.

Data Tractability

- Backward-to-Root Reference - A backward reference (link) to the originator entity of the record.

– Backward Reference - A backward reference (link) to the entity that the record was obtained from.
– Forward References - A list of forward references (links) to all entities that this record was disclosed to from its present location.

Cryptographic Controls

– Original Record - A copy of the received signed record.
– Signature - Hash code of the complete record (excluding the original record) signed with the current entity's signing key.

Figure 2 illustrates a record shared among 4 companies, forming a 3-level tree. The root of the tree is *Company A* that created the original record. *Company A* directly shares it with *Company B*, which in turn discloses the record to *Company C* and *Company D*. The bidirectional solid lines between companies represent the forward and backward references while the directed stippled lines represent the backward reference to the root of the tree.

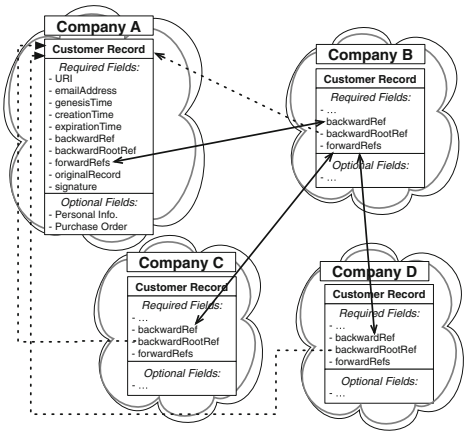

Fig. 2. Customer record tree

Using the example of Fig. 2, the *Customer Record* as it is stored by *Company B* is shown in Listing 1.1. There is a backward root reference to *Company A*, which was the originator of the record as well as a backward reference to the same entity as it is the one that provided the record. Additionally, as *Company B* forwarded the record to both *Company C* and *Company D*, the latter two entities are included in the forward reference list. For brevity reasons, the parent record field is not shown as this is an exact copy of the record stored by *Company A*.

Listing 1.1. Partial Customer Record Document

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<custRecord rec:URI="www.CompB.com:JohnDoe:20151025_120500" xmlns:rec = "http://
    www.unic.ac.cy/customerRecord">
 <rec:emailAddress>johnDoe@mail.com</rec:emailAddress>
 <rec:genesisTime>201510151205</rec:genesisTime>
 <rec:creationTime>201510251205</rec:creationTime>
 <rec:expirationTime>201810151205</rec:expirationTime>
 <rec:bwRef>www.CompA.com:JohnDoe:20151015_120500</rec:bwRef>
 <rec:bwRootRef>www.CompA.com:JohnDoe:20151015_120500</rec:bwRootRef>
 <rec:fwRefs>
   <custRecordList:fwRef>www.CompC.com:JohnDoe:20151028_120500</custRecordList:
       fwRef>
   <custRecordList:fwRef>www.CompD.com:JohnDoe:20151029_121520</custRecordList:
       fwRef>
 </rec:fwRefs>
 <rec:parentRecord>...</rec:parentRecord>
 <rec:signature>uWta23rEsAEw56Sefgs34...</rec:signature>
  ...
</custRecord>
```

The structure and controls embedded in *Customer Record* allows for utilization of standard generic tree operations for tree traversal and construction of data trails. Furthermore, record removal as well as update operations are possible via the forward references kept in the record. Needless to say, in a real deployment, deeper and broader trees would be constructed per customer record.

3.2 Collection Module

The *Collection* module is the data collection point of *privacyTracker*. Customizable registration applications interact with this module via its public API. There is no automated way to examine whether or not the collected data is lawful and adhering to legal state/country processing laws. Thus, for maintainability purposes, low coupling is strived between the registration application and the Collection module. That implies user consent is obtained via the customized registration application and the data communicated to the Collection module is flagged as disclosed or non-disclosed, based on the user preferences. Each new registration results in the creation of a new customer record. Any optional fields that are outcomes of further data processing or user-company transactions are assessed for legality by the controller module. Similarly to the data collection legality issue, it is beyond the scope of this research effort to automate the legality of data processing. However, the provision of the placeholder could accommodate a future automated routine as a plugin.

3.3 Distribution Module

The Distribution module manages requests to share customer data, either in coarse-grained manner or fine-grained manner. Similar to the previous module, transfer data requests are submitted via a custom application that interfaces with the module API. The requestor could form customized queries on preferred data transfers or use predefined queries. The receiving entity evaluates the request,

which leads to 3 possible course of actions: reject, accept as received or partially accept by filtering out records and/or record fields that are not to be disclosed. The latter option gives control to the owner of the data records to decide their further disclosure, even when the data owners gave consent for its disclosure.

As a record gets distributed and handled by many entities, undisputed verifiable guarantees must be provided regarding the record integrity. Any record modifications should be attributed to the entity that made the changes. This is achieved via cryptographic techniques, and to be more specific by digitally signing the hash of the customer record. A company could potentially modify a record in order to incorporate additional data and/or change existing ones and share the new version with others rather than forwarding the version it obtained. Prior to distribution, the original record is embedded in the new record as one of the metadata cryptographic control fields and the hash of the new record is generated, signed, and inserted as the second metadata cryptographic control field (that was signed by the company that disclosed the record). The embedded cryptographic controls provide for nonrepudiation as a user would be able to gather all available versions of his/her record (via the traversal algorithm described later on) and a company could not deny the existence of record versions originated from it. Note that companies receiving a record from the same source must possess the same original record, regardless of any further changes that they may do on the record.

3.4 Traceability Module

A core element of any proposed GDPR-compliant framework is the ability to trace data from its original source to various destinations. Data traceability requires the collaboration of all enterprises and has two components: tracking and tracing. Tracking is the capability to record the path of data as it gets shared with other companies other than the source company that collected the data. Tracing is the capability to identify the origin of data and needless to say tracing will only be successful with properly implemented tracking. Data traceability is the building block to support a variety of GDPR requirements, including the right to erasure and providing the original source of the data.

The proposed framework supports data traceability by utilizing two references of the customer record metadata. When the organization (source) is about to share the record with another organization (target), the source company places a Forward Reference in the record metadata that points to the location that the target company will use to store the record. Similarly, the target organization upon record transfer, inserts a Backward Reference into the metadata of the new record that it creates locally, which points back to the record of the source company. This process is repeated whenever the record is shared. As a result, an *implicit* tree is created (see Fig. 2), with the root node being the originator of the data.

In addition to the forward and backward references there is also a Backward-to-Root Reference in all the records. The reason for maintaining the backward-to-root reference is for recovery reasons in case there should be a link breakage

somewhere along the record trail. Link breakage is interpreted as company unavailability or unreachability during contact attempts. A variety of reasons could cause this situation, including out-of-business and legal issues. Using the backward-to-root reference, the unavailable link is located and the repair mechanism is initiated. With the backup backward-to-root reference then this breakage could be located and a repair could be initiated.

It is important to note that whereas a user has the legal right to traverse the record tree, from root to the branches, companies should only be allowed to traverse one level up or one level down the tree (parent node or child nodes) to preserve user privacy. This is a default setting in the *privacyTracker* and access controls are in place to implement this restriction (it could be lifted if deemed necessary).

Below, details are given on constructing the data trail for a specific user, repairing unreachable link references, and addressing the right-to-erasure; all operations are mapped into generic tree operations.

Construction of Data Trail. The construction of a data trail is a standard generic tree traversal problem. Algorithm 1 depicts the steps to traverse the *Customer Record* implicit tree in bottom-up approach, starting from any tree node (i.e. any company that holds the record) towards the root of the tree (i.e. the original creator of the record). The end result is a path from any node to the root.

Algorithm 1. Traverse Customer Record Algorithm

```
1: function TRAVERSE(CustRecordURL url, EmailAdr adr)
2:     CustRecordURI parentURI ← null
3:     CustRecord parentRecord ← null
4:     CustRecordURI currentURI ← GETCUSTRECURI(url, adr)
5:     CustRecord currentRecord ← GETCUSTREC(currentURI, adr)

   ▷ Loop backward until reach root
6:     while (currentRecord != null) do
7:         SHOWRECORD(record)
8:         parentURI ← GETPARENTURI(currentRecord)
9:         parentRecord ← GETCUSTREC(parentURI, adr)

   ▷ Test for broken link
10:        if (parentRecord = null and parentURI != null) then
11:            REPAIRTREE(currentRecord, adr)
12:        else
   ▷ Check for tampering with record
13:            if (VERIFYREC(currentRecord, parentRecord) = false) then
14:                REPORTVIOLATION(currentRecord, parentRecord)
15:            end if
16:        end if
17:        currentRecord ← parentRecord
18:    end while
19:    SHOWSUMMARY(void)
20: end function
```

Suppose a customer receives an unsolicited request from *Company D*. Traversing the path from *Company D* to the root, the customer could discover who originally collected the data and how the original record was propagated from company to company to end up in *Company D*. Along this path one should be

able to determine who disclosed the record unlawfully. The algorithm requires two input variables: the *url* of the company that sent the solicitation and the user's email address. The company gets a customer record request and returns the customer record URI which the user can use for the request to return the whole customer record (see lines 4–5). The backward tracing starts as a repetition process (see lines 6–18). The parent record is first obtained. In case the parent record is *null*, but the parentURI is not *null*, then a breakage in the tree has taken place. In this scenario, the tree repair algorithm is initiated (details below). If there is no breakage in the tree, then a validation check is done (see line 13) to test the integrity of the record contents compared to the parent record contents. If such a modification took place, a violation is reported to the user. It is outside the scope of the framework, for the time being, to investigate how violations are addressed. The last line in the repetition process (see line 17) is used to move one level up in the tree towards the root.

A user has the right to obtain from an organization all the recipients to whom his/her data have been disclosed. A similar algorithm could be used to search the tree top-down (using breadth or depth first search) in the opposite direction. Suppose that the user desires to view all recipients of his/her data starting from a specific company. In this case, a forward searching algorithm will be used (not included here) with the end result being a tree.

Recovering from Unavailable Link References. The repair algorithm works like a standard *remove node* from a double linked list. Suppose that the parent node of the current node is unavailable, thus the link references must be updated so as the current node will have backward reference to its grandparent node. This entails using the backward-to-root reference to perform a forward search to locate the grandparent of the current node and readjust the link references. The assumption is that no other nodes in the tree are unavailable. In the unlikely scenario where 2 nodes on the data trail are unavailable, two different approaches could be deployed to reestablish connectivity in the tree, with different tradeoffs.

Right-to-Erasure. The right-to-erasure requires erasure of user data from all its recipients. With the current data structure, this is easily implemented by constructing a tree for the user data starting from the root to all its leaves, and proceed with deleting all versions for the particular user along all tree paths.

4 A privacyTracker Prototype

A prototype was built along the principles of *privacyTracker* as a proof-of-concept regarding the feasibility of the proposed approach. The prototype is a web-application consisting of three modules, built on top of a WAMP (Windows, Apache, MySQL, PHP) server. Additional technologies used are JavaScript, CSS, XML, HTML 5, MD5 hashing algorithm, and OpenSSL. The experimental setting consisted of 6 companies.

Collection Module: The collection module, depicted in Fig. 3, allows user registration. There are 3 ways that user data could be communicated to the *privacyTracker*. First, directly using the prototype's registration form. In this case, data validation is supported (e.g. address format in different countries) via regular expressions, followed by insertion into the backend MySQL database. Second, having customized registration modules using the provided API to populate the database. Third, through the distribution module (presented next), where traded data is merged with the local company data. It could be the case that multiple entries exist for a single user. The database consists of 3 tables and is normalized to support this. PHP scripts generate the tables in the database, hence there is no need of manual management of the database.

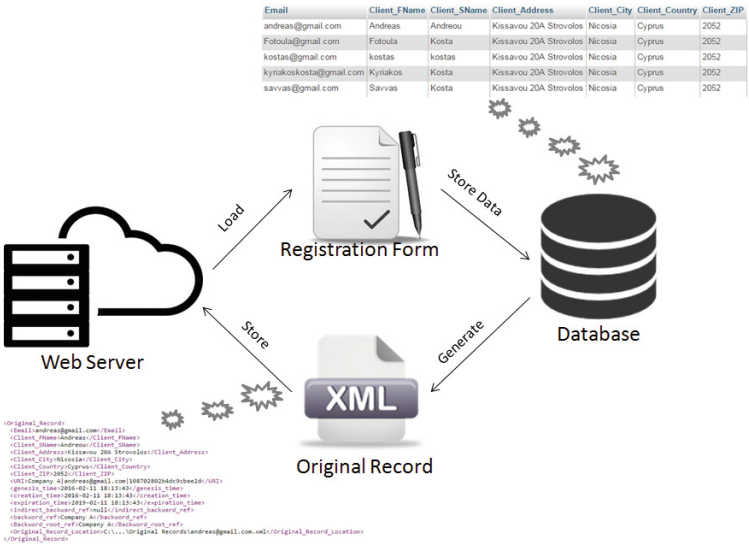

Fig. 3. Registration module

Distribution Module: The distribution module is responsible for the sharing/selling/trading customer information and it is divided into 3 submodules. The first submodule accepts requests for data transfers, which are translated into SQL queries. The prototype supports a web view where the user manually specifies the information to be traded and the receiver entity. The selection of data to be shared is illustrated in Fig. 4. The second submodule encodes the result of the SQL query into an XML document, digitally signed by the current enterprise. The signed document is transferred to the receiving organization using an SSL channel. Once the document is received, the sending company proceeds with updating the forward references of the successfully transmitted records. The last submodule is executed by the receiving company that, upon verification of the XML signed document, converts it to SQL statements that populate the recipient database with the new data. In addition, the backward reference is created to point to the sending company. The received XML document is also saved into

the permanent log directory. In the case that the receiving company already has information about a user (identified by the email address), the user-specific records are merged. In the unlikely scenario that the exact same record already existed, the company keeps its own original copy. This could happen if a lattice is created; for example *company A* sells a record to *company B* and *company C*; then *company D* buys the same record from both *company B* and *company C*.

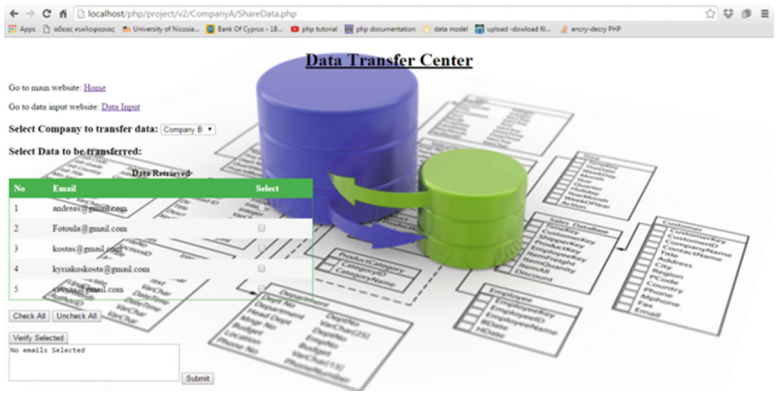

Fig. 4. Distribution module

Traceability Module: A web form was created for each of the six companies that accommodates end-users' requests to query on stored information related to them. The end-user provides the email address that serves as the *authentication* token. It is in the future plans to enhance the authentication process with one-time passwords (emailed to the user) to prove authenticity. Once authenticated the user request gets converted into an SQL query that returns all the information collected for this specific user. The resulting records from the query are encoded in XML and digitally signed. From the returned XML document, the end-user can use the forward and/or backward references to build a trace tree. It is envisioned that user apps will be created to automatically build the complete trace tree from any starting point. The *privacyTracker* framework provides the appropriate APIs and hooks for the development of such apps.

5 Conclusion

To the best of our knowledge, there is no practical mechanism that determines accurately the disclosure of data collected by organizations. There are privacy policies that vaguely specify the handling and processing of data, however the consumer is not informed neither about the identity of the third-party entities that have access to his/her data nor the actual data that is accessible by them. This paper presented the *privacyTracker* framework, a novel approach that empowers consumers with appropriate controls to trace the disclosure of data as collected by companies and assess the integrity of this multi-handled

data. This is accomplished by constructing a tree-like data structure of all entities that received the digital record, while maintaining references that allow traversal of the tree from any node, both in top-down manner and bottom-up manner. A prototype was developed based on the *privacyTracker* principles as a proof-of-concept of the viability of the proposed principles.

Acknowledgment. The authors would like to thank the BeWiser consortium (funded under EU FP7, Grant No: 319907) for fruitful discussions on citizen security and privacy issues.

References

1. TRUSTe: 2015 truste us consumer confidence index (2015). https://www.truste.com/resources/privacy-research/us-consumer-confidence-index-2015/. Accessed 25 Sept 2015
2. Westin, A.: Privacy and Freedom. Atheneum, New York (1967)
3. Parliament, E.: Regulation of the European Parliament and of the Council on the Protection of Individuals with regard to the Processing of Personal Data and on the Free Movement of Such Data (General Data Protection Regulation). Technical report (2015)
4. Karjoth, G., Schunter, M., Waidner, M.: Platform for enterprise privacy practices: privacy-enabled management of customer data. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, pp. 69–84. Springer, Heidelberg (2003)
5. Kalloniatis, C., Mouratidis, H., Vassilis, M., Islam, S., Gritzalis, S., Kavakli, E.: Towards the design of secure and privacy-oriented information systems in the cloud: Identifying the major concepts. Comput. Stand. Interfaces **36**(4), 759–775 (2014). Security in Information Systems: Advances and new Challenges
6. Barth, A., Datta, A., Mitchell, J.C., Nissenbaum, H.: Privacy and contextual integrity: framework and applications. In: 2006 IEEE Symposium on Security and Privacy Security and Privacy, pp. 184–198 (2006)
7. Bertino, E., Ghinita, G., Kantarcioglu, M., Nguyen, D., Park, J., Sandhu, R., Sultana, S., Thuraisingham, B., Xu, S.: A roadmap for privacy-enhanced secure data provenance. J. Intell. Inf. Syst. **43**(3), 481–501 (2014)
8. Mont, M.C., Pearson, S., Bramhall, P.: Towards accountable management of identity and privacy: sticky policies and enforceable tracing services. In: 2003 Proceedings of 14th International Workshop on Database and Expert Systems Applications, pp. 377–382 (2003)
9. Epic: Electronic privacy information center survey: 74% of presidential candidate's websites fail on privacy. https://epic.org/2015/09/survey-74-of-presidential-cand.html. Accessed 25 Sept 2015 (2015)
10. Alsenoy, B.V., Verdoodt, V., Heyman, R., Ausloos, J.,Wauters, E.: From social media service to advertising network: a critical analysis of facebook's revised policies and terms. Technicalreport, Interdisciplinary Centre for Law and ICT/Centre for Intellectual Property Rights of KU Leuven and the department of Studies on Media of the Vrije Universiteit Brussel (2015)
11. Gjermundrød, H., Dionysiou, I.: A conceptual framework for configurable privacy-awareness in a citizen-centric egovernment. Electron. Gov. **11**(4), 258–282 (2015)