PePPer: Fine-grained Personal Access Control via Peer Probing

Yael Amsterdamer[1] and Osnat Drien[2]

[1]Department of Computer Science, Bar-Ilan University, Israel

[1]first.last@biu.ac.il
[2]firstlast@gmail.com

Abstract

Users share data on multiple platforms where access control is managed according to the platform-specific policy. However, some data is conceptually owned by peers in a manner that is platform-independent. To enable peers to manage access control rights on such data we introduce PePPer, a tool for fine-grained, personal access control. This system, which runs on the client side, enables loading data items from different sources and annotating them with fine-grained access control requirements via provenance-style Boolean expressions. These expressions are evaluated to decide whether the client is allowed to share the data with a given peer, using a taxonomy that compactly captures data ownership and access control policies. If credentials depend on the unknown access policy of other peers, PePPer probes them to obtain relevant access permissions. For that, we employ efficient algorithms that minimizes the expected number of probes. Our algorithm adapt techniques from stochastic Boolean evaluation to our setting by accounting for multiple peers, policies, expressions and the access control taxonomy. Throughout this process, PePPer further presents to the client a continually updated partial view of the data currently known to be safely shareable. We demonstrate PePPer for the sharing of calendar entries among peers, using as example real-life calendars of politicians and public figures.

1 Introduction

The exchange of data items between peers is an extremely common operation in different platforms and contexts, including social networks, cloud-based file sharing and messaging applications. Access control (AC) permissions are typically managed in an application-specific manner, while data may flow from one application to another. Lacking a generic solution, the "conceptual" owners of a data item may be unable to enforce AC policies across the multiple applications.

As a simple example, Alice may forward a calendar meeting between herself and Bob to Carol, thereby revealing private information of Bob, and possibly of other peers such as invited guests, authors of attached documents, etc. The calendar item may also propagate through various media to other peers, further complicating access control and calling for a generic, application-independent solution.

To this end, we introduce PePPer (for Peer-Probing Personal access control), a client-side prototype system to manage data sharing in presence of AC constraints. Our AC model associates with each shared data item a Boolean combination of symbolic variables that we term concepts. Such concepts refer to, for instance, user accounts or devices through which an item can be accessed, or user-owned contents within the item such as usernames, locations, tags and so on. Their combination in a Boolean expression captures the intuition that access may be granted based on alternative sources of the data item, or based on joint permission by co-owners of the data item/contents. The former corresponds to disjunction and the latter to conjunction.

Deriving concepts and Boolean expressions over them for AC credentials is natural for many application domains. For relational/XML data propagated through queries, Boolean expression construction is in line with the semiring provenance model of [6, 8]. We use the intuition of alternative/joint access permissions to derive expressions according to application-specific semantics. For instance, in calendar events and similar applications, concepts can be extracted from field labels and values of the item such as participants and locations, as well as from specific textual content such as email addresses and phone numbers mentioned in the event description or note. Then, AC expressions should reflect *alternative access paths* to the data such the via invited participants, together with all the permissions *jointly required to view the contents*. When items are nested (e.g., a document attachment or reference to another calendar meeting), its Boolean expression is recursively constructed and joint in conjunction to the expression of the nesting item.

Continuing our running example, to see a calendar meeting organized by Bob in his office with Alice as an attending guest one may need permissions to the specific event via Alice or Bob, as well as to all the concepts in the event contents. The corresponding AC expression may have the form:

$$\begin{aligned}(\texttt{AliceEvent123} \vee \texttt{BobEvent123}) \wedge \texttt{BobOrganizer} \\ \wedge \texttt{BobAttend} \wedge \texttt{AliceAttend} \wedge \texttt{BobOffice}\end{aligned} \tag{1}$$

To decide whether Alice can share this event with Carol, one has to evaluate its AC expression by assigning true/false values to its variables, according to Alice's and Bob's AC policy with respect to Carol. However, since this is computed at the client side, Alice may not (fully) know the AC policy of Bob, and in this case has to ask him about the relevant permissions.

We thus consider the *probing* of relevant peers to discover unknown AC permissions, namely sending questions about peer-owned concepts that are either answered automatically by the AC policy recorded on the peer side, or manually by the peer. To preserve the privacy of peers as much as possible, and – in the case probes are answered manually – reduce the amount of effort incurred to the peer in answering them, *our goal would be minimizing the amount of probes sent to peers.*

Clearly, not all concepts need to be probed, but their choice may be unknown in advance: e.g. if Alice grants Carol access to the event ($\texttt{AliceEvent123}=1$), there is no need to ask about `BobEvent123`; or, given a negative answer regarding `BobOrganizer`, there is no need to ask about `BobAttend` nor `BobOffice`. The truth values of concepts may also be dependent: e.g., if we know that Bob does not allow Carol to know his participation status in events (`BobParicipantStatus`=0), in particular she has no permissions to `BobAttend`.

For that, we model unknown permissions as random variables, for which dependencies are captured by a concept taxonomy. We then solve the computational problem of incrementally deciding which concepts to probe, to evaluate an AC Boolean expression. The target function to minimize may be set to either the expected overall number of probes (`INC-EVAL`), or the maximal expected number of probes per peer (`INC-BALANCE-EVAL`).

This formulation allows us to build upon results from the area of stochastic Boolean evaluation [5], adapting and extending them. In particular, we obtain that both variants of the problem correspond to NP-hard decision problems, but we present efficient approximation algorithms. Another novel component of our solution is the computation of prior probabilities for the variables based on the taxonomy, which we address via the connection to interpolation over partial orders [2].

Our models and algorithms are implemented in PePPer, that gets as input a concept taxonomy, concept ownerships (i.e. which peer should be contacted for probing each concept) and possibly known access policies. Data items can be loaded into PePPer via plugins that connect to an external application. The client can define a selection query over these items and attempt to share its results with a given peer; this will execute the algorithm of peer probing. During this process, the UI of PePPer allows the client to see a *safe view* of the items, consisting only of fields already approved to be shown.

While our solution is generic, our proposed demonstration of PePPer will use the calendar example illustrated above, based on publicly available real-life calendars of politicians and office holders. We will interactively engage the audience in selecting examples, and show the different facets of PePPer through the views of both the client and probed peers.

We stress that our focus is on the logical aspect of the problem, namely, the computation of access control permissions, rather than on the design of e.g. cryptographic means of enforcing it, which would be complementary to this work. PePPer is designed to enable clients to verify and respect their own and their peer's access policies (*collaborative* access control, as termed by [9]).

Related Work Fine-grained models of access control have been studied in different contexts, including social networks (e.g., [1, 3]), distributed systems (e.g., [9]) and databases (e.g., [6, 8]). The work of [9] resembles our setting in the management of access on the client side. However, they assume that full access control policies are available and do not consider probing. With respect to these works, our novelty is in modeling the access control problem as a variant of stochastic Boolean evaluation. This latter problem has been extensively studied in multiple contexts such as active learning and system testing (e.g., [4, 5, 7, 10]). Algorithmically, `INC-EVAL` is similar to the problem studied in this literature but the use of concept taxonomies is novel to our knowledge. The existence of multiple peers gives rise to `INC-BALANCE-EVAL` whose solution is also novel to our knowledge.

2 Technical Background

2.1 Access Control Model

Let $\mathcal{C}$ be a domain of concepts. An *access control taxonomy* $\mathcal{T}$ is a partial order $\langle \mathcal{C}, \prec \rangle$, where for $c, c' \in \mathcal{C}$, $c \prec c'$ denotes that an access permission to c implies access permission to c'. E.g., `AliceCalendar`$\prec$`AliceEvent123`. Let $\mathcal{P}$ be a domain of peers. An *ownership function* owns : $\mathcal{C} \to \mathcal{P} \cup \{\bot, \top\}$ maps each concept to its owner peer or to $\bot, \top \notin \mathcal{P}$ if no owner exists or the owner is unknown, respectively. We impose that if $c \prec c'$ and $\text{owns}(c) = p \in \mathcal{P}$ then $\text{owns}(c') = p$. Note that we assign a single owner for each concept, and capture mutual ownership via access control expressions.

An *AC policy* over $\mathcal{T}$ is a Boolean function ac : $\mathcal{C} \times \mathcal{P} \to \{1, 0\}$ where intuitively $\text{ac}(c, p) \mapsto 1$ iff p has access permissions to c. We require for all $p \in \mathcal{P}$ and $c, c' \in \mathcal{C}$: (i) if $\text{owns}(c) = \bot$ or $\text{owns}(c) = p$ then by default $\text{ac}(c, p) = 1$; (ii) if $\text{owns}(c) = \top$ then $\text{ac}(c, p) = 0$; and (iii) if $c \prec c'$ then $\text{ac}(c, p) \implies \text{ac}(c', p)$.[1]

We then use the above representation to control the sharing of data items between peers. Following [6, 8] we define the access control semiring as $(\mathrm{B}[\mathcal{C}], \vee, \wedge, 0, 1)$, namely positive Boolean expressions using the concepts in $\mathcal{C}$ as variables. We use elements of this semiring (termed AC expressions) to annotate data items. Intuitively, in disjunctive normal form, every conjunction in an AC expression represents an *alternative* access path to a data item, and such an access path is valid iff all of its concepts are *jointly* accessible.

2.2 Probing Peers for Access Policy

We consider a scenario where a peer p wishes to share some data items with peer p', but ac is either unknown or partially known. For each concept c, an unknown value of $\text{ac}(c, p')$ can be discovered by a probe to its owner $\text{owns}(c) \neq \top, \bot$. More generally, given $\mathcal{T}$, an ownership function owns, a multiset of access control expressions $\mathbf{E}$, a peer $p \in \mathcal{P}$ that needs to access $\mathbf{E}$ and possibly some mappings of the ac function with respect to p, we define `INC-EVAL` as the problem of incrementally selecting the next batch of variables $C \subseteq \mathcal{C}$ to evaluate (by probes to $\text{owns}(c)$ for every $c \in C$) that minimizes the *expected number overall probes*, according to some prior probability distribution on the probe answers.

We further consider a variant of `INC-EVAL` that aims to minimize the number of probes *per peer*, in order to split the revealed information between peers more evenly, from privacy and effort (in the case of manual probe answering) considerations. Denote by $C_{p'}$ the subset of probes directed to peer p' in C, formally $\{c \in C \mid p' = \text{owns}(c)\}$. We define `INC-BALANCE-EVAL` similarly to `INC-EVAL`, but where the target goal is minimizing the *maximal expected number of probes of any peer*, i.e. minimizing $\max_{p' \in \mathcal{P}} |C_{p'}|$.

The problem `INC-EVAL` may be phrased in terms of (monotone) Boolean evaluation [5]: given a monotone Boolean formula along with costs and probability distributions for its variables, choose which variable to observe next to minimize the expected evaluation cost. Via this connection, we obtain that `INC-EVAL` and thereby also `INC-BALANCE-EVAL` (`INC-EVAL` can be reduced to

[1] For simplicity, we define here AC policies per peer; but our approach extends in a straightforward manner to peer groups.

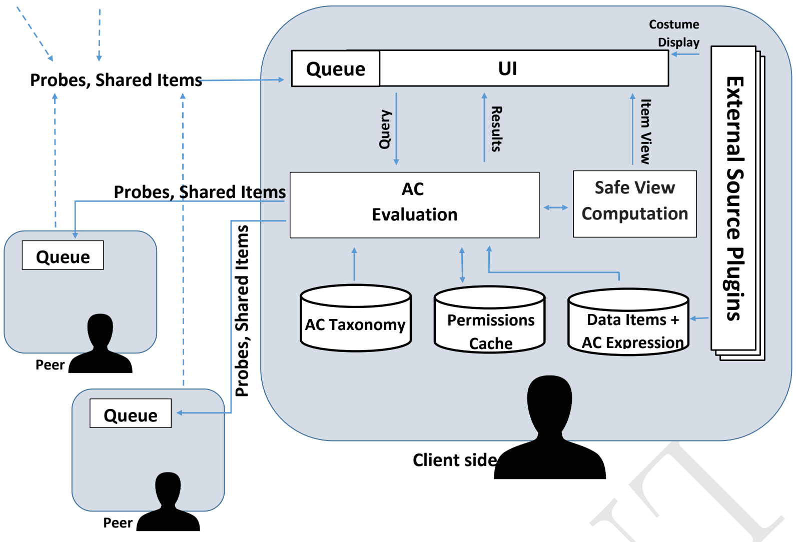


Figure 1: Architecture of PePPer.

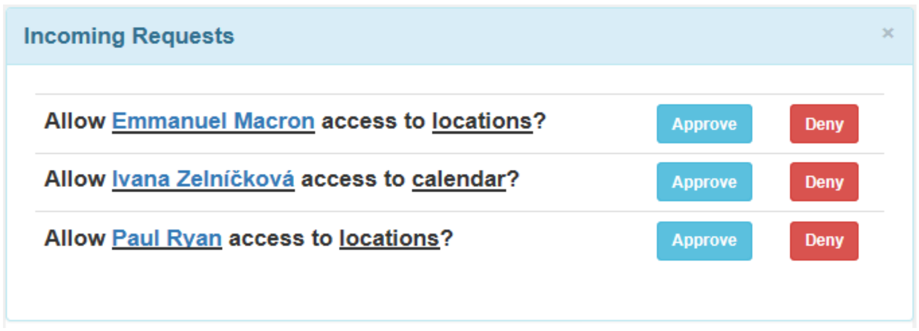
`INC-BALANCE-EVAL` by selecting one peer to be the owner of all concepts) correspond to NP-hard decision problems [4]. Previous work has studied optimal solutions for restricted cases, heuristics and approximations (see, e.g., [10] for a survey).

In our work we adapt an approximate solution by [5], as we next briefly outline. For each Boolean formula $e_i \in \mathbf{E}$ we define two *utility* functions $g_0^i, g_1^i : \{1, 0, *\}^{|\mathcal{C}|} \to \mathbb{R}^+$, where each entry represents either a value assignment to a variable or no assignment ($*$). $g_0^i(\vec{c})$ and $g_1^i(\vec{c})$ are respectively the number of terms (conjunctions) set to 0 in the DNF form of e_i, and the number of clauses (disjunctions) set to 1 in the CNF form of e_i by the partial assignment $\vec{c}$. Denote by m_i and l_i the number of terms and clauses in e_i respectively. The utility function $g^i(\vec{c}) = m_i l_i - (m_i - g_0^i(\vec{c}))(l_i - g_1^i(\vec{c}))$ reaches its maximum, $m_i l_i$, when $g_0^i(\vec{c}) = m_i$ or $g_1^i(\vec{c}) = l_i$, i.e., e_i is evaluated. A greedy algorithm is then used, to repeatedly select the unknown variable c_j whose probe maximizes the expected value of g^i. Some properties of g^i (monotonicity, submodularity) guarantee that the expected number of questions is within a factor of $\ln(m_i l_i) + 1$ from the optimum.

Now, to solve `INC-EVAL` we need to simultaneously evaluate all the formulas in $\mathbf{E}$, which may be done by defining $g(\vec{c}) = \sum_{e_i \in E} g^i(\vec{c})$. g is a function that reaches its maximum, $\sum_{e_i \in \mathbf{E}} m_i l_i$, when all the expressions are evaluated. By similar analysis to that of g^i, we get that the solution is a $\ln(\sum_{e_i \in \mathbf{E}} m_i l_i) + 1$ approximation of the optimum.

We next consider an approximate solution to `INC-BALANCE-EVAL`. For that, we keep track of the number of probes per peer $p \in \mathcal{P}$ (denoted by $\text{probes}(p)$). At each step, let $\mathcal{C}^*$ be the set of variables still unknown, and let $\mathcal{P}^* = \{p \in \mathcal{P} \mid \exists c \in \mathcal{C}^*,\ p = \text{owns}(c)\}$ denote their owners. When selecting the next probe, we consider only variables in $\{c \in \mathcal{C}^* \mid \text{probes}(\text{owns}(c)) = \min_{p' \in \mathcal{P}^*} \text{probes}(p')\}$, i.e., that are owned by least-probed peers.

In these algorithms, we implicitly account for the AC taxonomy $\mathcal{T}$: for every


Incoming Requests

Allow Emmanuel Macron access to locations?

Allow Ivana Zelníčková access to calendar?

Allow Paul Ryan access to locations?

Figure 2: Example pop-up showing user probes.

$c \prec c'$, the number of eliminated terms/clauses is affected by implications of the form $\mathrm{ac}(c, p) \implies \mathrm{ac}(c', p)$, yet the approximation bound holds.

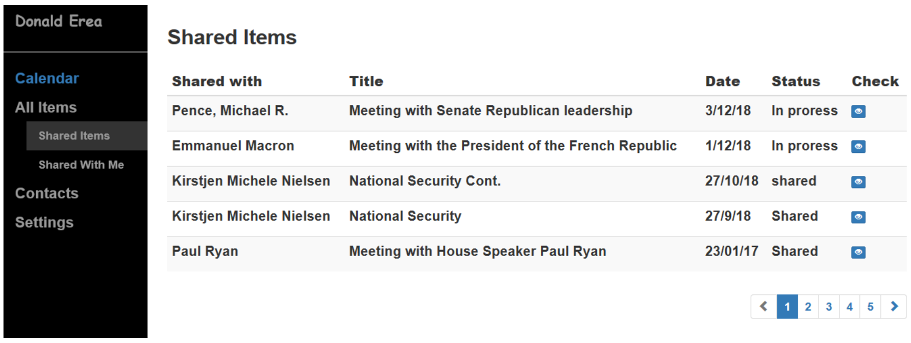
To further accelerate peer probing we perform a lookahead into the following steps. Let c_0 be the variable selected to be probed next at some point of the process. We first assume that $c = 0$ and then $c = 1$, and denote by c_1^0 and c_1^1 the corresponding selected variables for the following step. If $c_1^0 = c_1^1$ then we probe its owner without waiting for the answer to c_0, and proceed in this manner. This allows us to execute probes in parallel for different peers as well as display a batch of related questions to a single peer.

Deriving probabilities Our probabilistic model assumes a prior probability $\Pr[c = 1]$ for every $c \in \mathcal{C}$. Clearly, given sufficient statistics on past probes of c one can compute an estimation of this probability function. In the absence of such statistics, we compute prior probabilities that account for dependencies defined by the AC taxonomy using the techniques of [2] to interpolate values under partial order constraints. Intuitively, the interpolation scheme accounts for the partial order by assigning a higher value to variables that precede more of the other variables (and vice versa).

2.3 Safe Views

So far, we have considered items as atomic units. In practice, they may be composite and each individual field of a data item may also be annotated by an AC expression. In this case, we can independently evaluate access permissions to each field, and compute at each point of the peer probing process a *safe view* that includes only accessible field of each item. E.g., Alice may decide to share with Carol a partial view of the calendar meeting excluding its location, if Carol has permissions to every other field.

Formally, let e and $e_1, \ldots e_F$ denote the AC expressions of a data item I and its fields $f_1, \ldots, f_F$. A safe view of I exists iff $e(\vec{x}) = 1$, where $\vec{c} \in \{0, 1, *\}^{|\mathcal{C}|}$ is a partial assignment of truth values to e as in the previous section. The safe view includes I with the subset of its fields that are known to be accessible: $\{f_i \mid 1 \leq i \leq F \wedge f_i(\vec{c}) = 1\}$. A full item is accessible if and only if $e \wedge \bigwedge_{i=1}^{F} e_i$ evaluates to true.



Shared Items

Shared with	Title	Date	Status	Check
Pence, Michael R.	Meeting with Senate Republican leadership	3/12/18	In proress	👁
Emmanuel Macron	Meeting with the President of the French Republic	1/12/18	In proress	👁
Kirstjen Michele Nielsen	National Security Cont.	27/10/18	shared	👁
Kirstjen Michele Nielsen	National Security	27/9/18	Shared	👁
Paul Ryan	Meeting with House Speaker Paul Ryan	23/01/17	Shared	👁

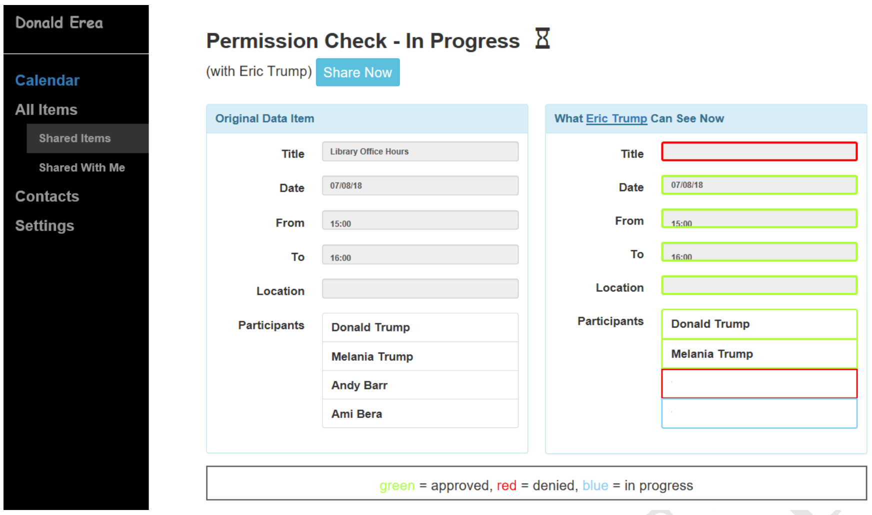
Figure 3: Example status of calendar events shared with peers.

3 System Overview

PePPer is implemented in node.js using express, Postgres for the data repository and javascript and jade for GUI. Its architecture is depicted in Figure 1. The data repository includes the data items owned by each client, where items and their relevant fields are annotated by AC expressions. The data is fetched via plugins to external sources.

We will demonstrate a plugin to Google Calendar (`https://www.google.com/calendar`), where AC expressions are defined via patterns: for instance, each event is annotated by a disjunction over its participants, as in the brackets of (1). The concepts annotating most fields are composed of the field value and label, e.g., `BobStatusAttending`. Each plugin can additionally construct and record an AC taxonomy of its computed concepts, along with known order constraints (e.g., `BobStatus`$\prec$`BobStatusAttending` since "attending" is a particular value of the "status" field) and known owners (e.g., `AliceCal` is owned by Alice's user). The taxonomy of all plugins is integrated into the main AC taxonomy of PePPer.

The client of PePPer can view data items that were loaded by plugins or shared with her by peers, and in turn can ask to share these items with others. Upon a share request, the AC Evaluation module executes our algorithms (Section 2.2) by sending selected probes to peers. The probes are managed by peers' incoming requests queue, and appear as pop-ups at the bottom of the screen (Figure 2). The client can track the progress of sharing via the Shared Items screen (Figure 3). By clicking the button next to an item, the client can compare the original data item with a safe view thereof (see Section 2.3). E.g., in Figure 4 the event title and third participant were declined for sharing with the peer, while permissions to show the last participant are not determined. The safe view (on the right) is updated with each probe response. At each point, the client may halt the probing process early and share the item as-is (Share Now button). The Permissions Cache records the responses to probes from and to the client, and reuses these permissions in subsequent evaluations.



Permission Check - In Progress

(with Eric Trump)

Original Data Item

Title	Library Office Hours
Date	07/08/18
From	15:00
To	16:00
Location	
Participants	Donald Trump
	Melania Trump
	Andy Barr
	Ami Bera

What Eric Trump Can See Now

Title	🟥
Date	🟩 07/08/18
From	🟩 15:00
To	🟩 16:00
Location	🟩
Participants	🟩 Donald Trump
	🟩 Melania Trump
	🟥
	🟦

green = approved, red = denied, blue = in progress

Figure 4: Example original item and its safe view.

4 Demonstration

We will demonstrate the use of PePPer for the sharing of calendar events, using the previously described calendar plugin (Section 3). To exemplify the calendar contents we have collected publicly available calendars of politicians and office holders. We will begin the demonstration by explaining the system architecture and its interface, as well as the calendar data. Then, we will invite the participants to play the role of a politician of their choice, that will be the client. Via the UI of PePPer we will browse through some of the calendar events for this client and query them. For example, we will select all events that occur in a certain country in the upcoming year. We will inspect the AC expressions of events, explaining how they are derived by our system based on e.g. the event contents. Then, we will demonstrate a request to share these events with a particular peer – another public figure. In turn, this involves probing owners of the events using our algorithms, asking them to authorize sharing. We will show the resulting pop-up requests on the relevant peers' sides. We will repeat this process for the different variants of the problem, demonstrating how the sequences of probes are affected by each variant. We will further show the effect of probe answers on the computed "safe view" on the client side.

Finally, We will allow participants to look under the hood, through a dedicated administrator view. This will display the count of probes (per peer and overall) as well as a visual insight into the algorithm reasoning by highlighting relevant parts of the evaluated Boolean expression.

Acknowledgements

This work was funded in part by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Ministers Office, and by the Israel Science Foundation (grant No. 1157/16).

References

[1] T. Abdessalem and I. B. Dhia. A reachability-based access control model for online social networks. In *DBSocial*, 2011.

[2] A. Amarilli, Y. Amsterdamer, T. Milo, and P. Senellart. Top-k querying of unknown values under order constraints (extended version). *CoRR*, abs/1701.02634, 2017.

[3] B. Carminati, E. Ferrari, and A. Perego. Enforcing access control in Web-based social networks. *ACM Trans. Inf. Syst. Secur.*, 13(1), 2009.

[4] L. A. Cox, Q. Yuping, and W. Kuehner. Heuristic least-cost computation of discrete classification functions with uncertain argument values. *Annals of Operations research*, 21(1), 1989.

[5] A. Deshpande, L. Hellerstein, and D. Kletenik. Approximation algorithms for stochastic Boolean function evaluation and stochastic submodular set cover. In *SODA*, 2014.

[6] J. N. Foster, T. J. Green, and V. Tannen. Annotated XML: queries and provenance. In *PODS*, 2008.

[7] D. Golovin and A. Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *J. Artif. Intell. Res.*, 42, 2011.

[8] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *PODS*, 2007.

[9] V. Z. Moffitt, J. Stoyanovich, S. Abiteboul, and G. Miklau. Collaborative access control in WebdamLog. In *SIGMOD*, 2015.

[10] T. Ünlüyurt. Sequential testing of complex systems: a review. *Discrete Applied Mathematics*, 142(1-3), 2004.