TECHNOLOGY AND CODE
published: 04 February 2020
doi: 10.3389/fcomp.2020.00002

Strengthening Enforcement in a Comprehensive Architecture for Privacy Enforcement at Internet Websites

Carlisle Adams, Yu Dai, Catherine DesOrmeaux, Sean McAvoy, NamChi Nguyen and Francisco Trindade*

School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON, Canada

This paper extends previous work to strengthen the *enforcement* portion of a comprehensive architecture for enforcing privacy when a user needs to submit personal data to an Internet website in order to obtain goods or services. Our extension proposes to use a website's P3P privacy policy (derived in an automated way from its internal XACML access control policy) as a public key to encrypt the user's data using IBE (identity-based encryption) technology. The website will only acquire the corresponding private key to decrypt this data if a trusted 3rd-party auditor (acting as an IBE private key generator) has verified that the P3P policy is an accurate statement of the site's internal privacy practices. We discuss all the components of this model and describe our proof-of-concept implementation which demonstrates that such an architecture is feasible in real-world scenarios.

Keywords: user agent, privacy enforcement, XACML, XSLT, P3P, IBE

Edited by:
Frank Kargl,
University of Ulm, Germany

Reviewed by:
Yannick Chevalier,
Université Toulouse III Paul Sabatier, France
Silvio Ranise,
Fondazione Bruno Kessler, Italy

****Correspondence:***
Carlisle Adams
cadams@uottawa.ca

Specialty section:
This article was submitted to Computer and Network Security, a section of the journal Frontiers in Computer Science

Received: *14 August 2019*
Accepted: *14 January 2020*
Published: *04 February 2020*

Citation:
Adams C, Dai Y, DesOrmeaux C, McAvoy S, Nguyen N and Trindade F (2020) Strengthening Enforcement in a Comprehensive Architecture for Privacy Enforcement at Internet Websites. Front. Comput. Sci. 2:2. doi: 10.3389/fcomp.2020.00002

INTRODUCTION

The privacy of personal information has received increased attention, particularly in electronic environments, over the past 20 years, and especially in the past decade. Internet users are more aware today than ever before of the risks to their privacy if their personal data is leaked or otherwise transferred to unintended parties. Simultaneously, holders of personal data (companies, governments, hospitals, etc.) are under increasing pressure to safeguard this data and to be transparent about the protection practices that they employ. On the other hand, these holders typically see significant benefits to storing, using in various ways, and sharing as much personal information about their customers/clients as they can. There can therefore sometimes be a tension between what these holders know they should do, and what they would like to do, with personal information. This gives rise to the concept of "privacy enforcement": technology that can be used to ensure that a holder of data only does with the data what they have publicly stated (i.e., promised) that they will do.

In 2002, the World Wide Web Consortium approved a W3C Recommendation entitled *The Platform for Privacy Preferences 1.0 (P3P1.0) Specification* (Cranor, 2002a; P3P, 2002) The goal of P3P (see P3P, 2002, p. 1) was to enable websites to express their privacy practices in a standard format that can be retrieved automatically and interpreted easily by software user agents. These user agents would then be able to inform the human user about site practices and automate decision-making based on these practices when appropriate (so that users would not need to read the privacy policies at every site they visit). Despite interest in academic circles and elsewhere in P3P, this Recommendation was made Obsolete by W3C in 2018, primarily for two reasons (P3P, 2002; Status of This Document). First, there was insufficient uptake of this specification to justify its continued Recommendation status (e.g., fewer than 6% of the 10,000 most frequently visited websites supported P3P in 2018). The W3C did concede, however, that "new data protection regulations taking effect in 2018 may bring new interest in machine-processable mechanisms similar to P3P." In particular, the (General Data Protection Regulation, 2018) issued by the EU that became enforceable in 2018 places clear requirements on holders of personal data and has the power to impose severe fines on violators of this regulation. GDPR applies to not just EU-based websites, but also any website worldwide that may collect and hold personal data of EU citizens. Thus, it is indeed the case that technologies such as P3P that facilitate the required transparency of privacy practices at a website may see a resurgence of interest and uptake in the wake of GDPR and similar regulations.

The second reason given for making P3P obsolete in 2018 was that "no enforcement action followed when a site's policy expressed in P3P failed to reflect the site's actual privacy practices" (P3P, 2002). The work described in this paper makes an initial step toward addressing this deficiency, not by stipulating after-the-fact fines or punishments, but rather by proposing a technical mechanism that ensures near-real-time compliance with the site's stated privacy policy. We describe an extension for, and a proof-of-concept implementation of, an architecture for privacy enforcement proposed in Adams and Barbieri (2006). Our extension specifically targets a missing enforcement scenario in the original architecture.

RELATED WORK

Much of the previous work in the area of privacy enforcement focuses on legislative efforts to protect privacy [see, for example (Global Privacy Enforcement Network, 2013; Federal Trade Commission, Privacy and Security Enforcement, 2019), and the OECD Report on the Cross-Border Enforcement of Privacy Laws (Organization for Economic Co-operation Development (OECD), 2006)].

With respect to architectural approaches to technical/automated privacy enforcement, related literature includes Henze et al. (2014) and Adams and Barbieri (2006). However, the work of Henze et al. proposes an architecture for enforcing user privacy in a network of IoT devices (e.g., a body-area-network of health monitoring devices) connected to cloud-based services. In this paper, our focus is on protecting user privacy during interactions with web-based services (e.g., online shopping transactions). Other related research focuses less on a complete architecture and more on specific components that could conceivably be used in such an architecture; recent examples include Kučera et al. (2017), Canovas Izquierdo and Salas (2018), and Tripp and Rubin (2014), while previous work includes EPAL (APPEL, 2000; Ashley et al., 2002, 2003), Hippocratic databases (Agrawal et al., 2002), and sticky policies (Beres et al., 2003). These individual component solutions are important, but do not address the need for an integrated overall architecture that enforces privacy in browser-server interactions.

The work described in this paper builds on the architecture proposed in Adams and Barbieri (2006) because that model is most closely related to the environment in which we are interested (i.e., privacy enforcement at Internet websites) and requires only a single extension feature to address a missing component in its threat model. Our work proposes this additional feature and describes a proof-of-concept implementation of the full architecture.

METHOD: PROPOSED EXTENDED ARCHITECTURE

Consider the scenario in which a user visits a website and, in order to obtain the desired goods or services, the user needs to submit personal information as requested by the website. How can the user feel confident that the website will "do the right things" with the user's data? ("Doing the right things" may include not sharing it with 3rd parties, using it for only the required purpose, storing it for no longer than a specified amount of time, and so on.) A simple mechanism to facilitate this trust is the advertised privacy policy of the website, which states what data will be collected and how it will be used and shared. Such policies, typically written in a natural language such as English or French, are commonplace (and, in many jurisdictions, are mandated by law). However, the mere existence of a privacy policy on a website gives no guarantee about the actual behavior of the organization associated with that website. The organization may engage in any kind of behavior with this data "behind closed doors," and the user might never find out about it. Furthermore, it is well-known that many users do not read privacy policies (certainly not in any detail) because these policies are long, complex, and written in difficult legal terminology that is all but incomprehensible to the average person.

A proposed solution for the unreadable (and therefore unread) privacy policies was P3P. With this technology, a software user agent would read the privacy policy for the user, compare that policy with the user's privacy preferences (as captured in a file stored on the user's machine), and alert the user only if a mismatch was detected. In this way, the user did not need to read any privacy policies, but the privacy policy of every single visited website (in fact, every web page of every website) would be carefully read by the user agent. In order to make this work, standardized syntax and semantics for privacy policies was needed, which is precisely what the P3P Recommendation specified.

A proposed solution for the organization engaging in behavior contrary to its posted privacy policy was to have external 3rd-party auditors who would periodically examine the actions of the company and compare those actions with the privacy policy. The auditor (e.g., TRUSTe Privacy Seal Program, 2019) would issue a so-called "privacy seal" if the organization lived up to its privacy promises; the seal would be displayed on the organization's website and users could have some confidence that the organization was doing what it claimed.

The above solutions are important steps, but there is a significant limitation. For an auditor to confirm that an organization is complying with their privacy policy, the auditor would need to examine every single dataflow and data storage point throughout the organization, which for a large enterprise can be prohibitively complex and time consuming. Thus, such audits are often incomplete and, even when they are complete, are done infrequently.

The APEX architecture (Adams and Barbieri, 2006) addressed this limitation by proposing an explicit tie between an organization's P3P policy and its internal access control infrastructure. In particular, recognizing that access to any data (including personal data) is governed by access control policies, and that access control policies can be written in XML, it proposed eXtensible Stylesheet Language Transformation (XSLT) technology (XSLT, 2017) to derive a P3P policy from an eXtensible Access Control Markup Language (XACML) policy (XSLT, 2017) in a fully automated way. Given that XACML defines all data access and use across the organization, it follows that a P3P policy automatically derived from an XACML policy must necessarily reflect the organization's actual behavior with the personal data that it stores. Furthermore, the auditor's task is now reduced to examining the relevant XACML policy or policies, the XSLT engine (to confirm that the P3P policy was actually derived from these access control policies), and the relatively small number of Policy Decision Points (PDPs) and Policy Enforcement Points (PEPs) in the infrastructure that fulfill the intentions of the XACML policies. This set of tasks is much more tractable than examining all possible information flows and storage locations in a very large organization. Thus, the auditors are confirming that the XACML policy/policies are actually being used by the organization, and that the P3P policy/policies are actually derived from the XACML policy/policies (i.e., the auditor's task is to ensure that the organization's privacy promises faithfully reflect its internal practices). Of course, it is always possible that an organization will change their XACML policy without changing their P3P policy as soon as the audit has been completed. Without continuous auditing (which is impractical for many reasons), the best way to protect against this is to have auditors conduct random "surprise audits" (a practice similar to unannounced health and safety inspections at restaurants). If this is done with some frequency, and if there are substantial consequences for failing an audit, then organizations will be less likely to take this risk.

One concern not addressed by the original APEX architecture is a malicious organization that pretends to have gone through an audit when in fact it has not. A "privacy seal" is essentially an icon that is displayed on a website; it would be trivial for a malicious organization to copy this icon from somewhere else and paste it on their own site. This organization's customers would think that an audit has been performed and therefore that the organization complies with its stated privacy policy, but in reality the audit was never performed and the organization might be doing anything with the data it acquires. Of course the user can contact the auditor to find out whether (and when) an audit was completed on organization "X," but few (if any) users will do this for every website they visit.

The extension we propose in this paper uses Identity-Based Encryption (IBE; Boneh and Franklin, 2003) to mitigate this concern. IBE is a cryptographic technology that takes an arbitrary string (Shamir's original proposal Shamir, 1984 was to use an e-mail address) and maps it to a public key that can be used for encryption purposes. It has been recognized that the arbitrary string can be a policy, giving rise to the concept of Policy-Based Encryption (PBE). Therefore, we suggest that the organization's P3P policy should be used as the string that maps to a public key. In such a scenario, the 3rd-party auditor (or even a threshold number of several auditors using a cryptographic secret-sharing scheme) can play the role of Private Key Generator (PKG), giving the organization the corresponding private key only if the audit confirms that the organization fully complies with its P3P policy.

The components of the architecture, then, work as follows. The organization uses XSLT to derive a P3P policy from its relevant XACML policies, and it posts the P3P policy on its website. A user visits the site and the user agent automatically downloads the P3P policy and compares it with the user's privacy preferences. If there is no mismatch, the user's browser displays the requested web page. If the user needs to submit personal information to the website (e.g., mailing address, credit card number, etc.), for example by filling in a web form, the user agent will encrypt this data using a public encryption key derived from the P3P policy (as defined by IBE/PBE technology) and send it to the website. If this organization has received a successful audit, it will have the corresponding private key to decrypt and obtain the user's data; otherwise, it will be unable to do anything with the ciphertext it receives from the user agent.

This extended APEX architecture provides privacy enforcement by ensuring that websites behave internally in compliance with their advertised privacy policy. This can help to give users confidence to share their personal information with websites when it is required in order to obtain the goods and services they desire. **Figure 1** shows a conceptual view of this extended architecture.

RESULTS

This section describes the results of our proof-of-concept implementation of the extended APEX architecture presented in section method: proposed extended architecture.

XACML

XACML defines standardized syntax and semantics for writing access control policies, along with the necessary decision request & response messages that allow a PDP to determine whether an access request should be granted or denied. Simple examples of XACML policies are presented in section results of XACML (2017). We also created our own very simple XACML policy which was used to produce a valid P3P policy; please see **Appendix A** for details.

Note that a recent paper by Jiang and Bouabdallah (2017) proposed JACPoL as an alternative to XACML that is based

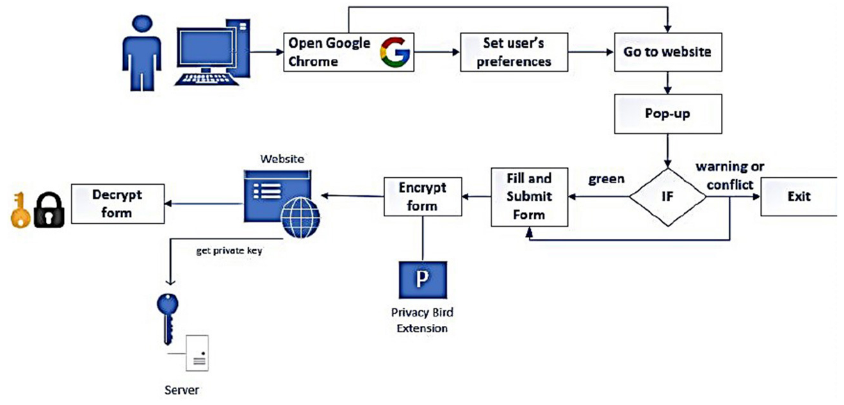


FIGURE 1 | Extended APEX architecture.

on JavaScript Object Notation (JSON) instead of XML and is therefore simpler and more efficient, while retaining descriptive power and human-readability. Although we explored JACPoL to some extent, for our proof-of-concept implementation we used XACML due to the availability of XSLT and the fact that both XACML and P3P are written in XML.

P3P

A P3P policy states the privacy practices of the website in a standardized format so that it can be read by a software user agent. It includes information about what data is collected, how long it is stored, who it may be shared with, and who a user should contact in the case of any disputes. A simple example of a P3P policy is presented as Example 3.2 in P3P (2002).

For our implementation, we created several simple P3P policies representing different privacy practices. This allowed us to easily test different scenarios (e.g., the P3P policy perfectly matched the user's preferences, or it did not match for any of several defined reasons). We also derived a very simple P3P policy in an automated way from our XACML policy; please see **Appendix B** for details.

XSLT

XSLT is a mechanism that can be used to transform an XML document into another XML document in a fully automated way. An XSLT Stylesheet is used to specify the transformation rules to be applied to the source document in order to transform it into the resulting document (note that the stylesheet itself is also written as an XML document). An XSLT Processor then inputs the source document and the stylesheet and outputs the resulting document. Finally, an XSLT Formatter can be used to pretty-print the resulting document for display purposes, if desired.

Several XSLT Processors are available. We reviewed a number of them for use in our implementation; these are described in the following subsections.

Browser-Based Translation

The most basic method we determined was to create an HTML document that calls the XACML document which has the XSLT document referenced. This method, while simple, has a key flaw that prevented us from using it: to run an HTML file we need a browser to act as the processor. The problem is that the common browsers (Google, IE, Firefox, etc.) only support XSLT 1.0, whereas XSLT is currently on version 3.0. As such this method of using a browser to act as the processor was insufficient due to the lack of many basic features that are a part of XSLT 3.0. For example, the ability to perform a "for-each" search was added in XSLT 2.0 and is thus missing in XSLT 1.0. Consequently, we discarded this method as useful for any modern XSL transformations.

Visual Studio 2019

A component of Microsoft Visual Studio (2019) is the XML editor. Of interest in our case is that a module of the XML editor is the XSLT tool, which allows the running of XSLT documents on XML documents. It also has inbuilt debugging features such as breakpoints and step-by-step processes. This combination of features within the XML editor was very desirable as it would allow work to be done in a single place. However, Visual Studio presents a single problem that made it unusable for our situation: cost.

The basic XML editor is a part of all versions of Visual Studio, including the free version. However, the extra functionality, including everything related to running XSLT, requires that a purchased version of Visual Studio be used. As such we did not use Visual Studio for its XSLT processing features. For other research groups trying to determine which processor to use, if an all-in-one package of editor and executor is desired, Visual Studio is probably the best fit.

FreeFormatter.com

FreeFormatter (2018) is an online website dedicated to providing free formatters, validators, and other tools online. Of interest to our case is the XSL Transformer page which takes an XML and XSL file as input, and outputs the resultant XML file. The translations performed by FreeFormatter.com are seemingly entirely correct; however, it is not a full system and merely prints the resultant XML document on the webpage. As such, it is not a system that can be easily used to retrieve the resultant P3P document.

However, its ease of use made it useful as a validator in our early work to confirm that we were performing the XSL translations correctly, or to help correct any other problem. In practice, this is probably where it is best suited (i.e., as a secondary checker of an XSL translation program, or to check smaller cases to make sure you are creating correct XSLT documents).

Oxygen XML Editor

Oxygen XML Editor (2020) is a standalone XML editor which has XSLT support. This means that, like Visual Studio, all activities related to the construction of the XSLT document can be done within the editor. Although it is a fully capable package to act as a processor, it does not offer any significant extra features over the Visual Studio XML editor. Furthermore, it also requires a purchase to use, and so we did not use it, nor do we see any reason to use it over the larger package of Visual Studio.

XSLTPROC

xsltproc (2019), which stands for "XSLT Processor," is a command line tool to run libraries which act as an XSLT processor. It possesses all the features that the (XML editor, XSLT processor) combinations have, along with a few extra functions. These extra features include timing functions and other debugging features to allow precise optimization. However, these extra features are beyond what was required for our project and were irrelevant to our selection process.

xsltproc is capable of being used on all operating systems: there are libraries for each operating system, and the libraries are available for free. We decided to use xsltproc as our XSLT processor of choice for this project.

Since xsltproc is just an XSLT processor, and not also an XML editor like Visual Studio and Oxygen XML Editor, we were required to use a basic text editor to write out XACML and XSLT documents. Unlike XSLT processors, text editors are not complex and any editor that can save files as ".xsl" and ".xml" can be used. In our case, Notepad++ was used.

Translating to P3P

When creating the XSLT document, we found that some P3P elements had an equivalent (or, at least, an analogous) XACML element from which the relevant data could be retrieved. Other P3P elements had no XACML equivalent (e.g., <ENTITY/> data, such as company name and physical address). For such elements we had to hard-code in the XSLT Stylesheet the data to be populated in the resultant P3P document. Although this hard-coding would need to be done for every given company, it only needs to happen once, and then will likely remain consistent for many years (even if the XACML policy of the company is modified), and so the amount of effort is minimized (in the sense of being amortized over time). Note that this hard-coding process is necessarily a human-centric process; it is not an automated activity. That is, the creation of these portions of the XSLT document requires human input. However, once the XSLT document is complete, it can be applied to any XACML policy in the company to produce a valid P3P policy in an automated fashion. The XSLT document can be used repeatedly, as XACML policies change and grow over time, to produce a valid P3P policy that corresponds identically to the XACML policy currently in place.

A decision regarding the data for populating the resultant document (i.e., the actual value for hard-coding or, alternatively, where it can be found in a source XACML document) was needed for every required P3P element: <POLICY/>, <ENTITY/>, <ACCESS/>, <DISPUTES-GROUP/>, <STATEMENT/>, <PURPOSE/>, <RECIPIENT/>, <RETENTION/>, <CONSEQUENCE/>, and <DATA-GROUP/>. The full collection of these decisions comprised the final XSLT Stylesheet document. In **Appendix C**, we show a relatively extensive example of an XSLT document that can create a valid P3P policy from a valid XACML policy (this example XSLT document was created by students D. Li and H. Yan in separate work). *Due to space constraints, only a portion of the XSLT document is shown in this paper. The full XSLT document is available at http://www.site.uottawa.ca/\protect\T1\textdollar%7Bsim%20%7D\protect\T1\textdollarcadams/papers/XSLTDoc(ExtendedAPEX-AppendixC).pdf*.

Note that the concept of privacy in a general sense can be fairly broad, including notions such as anonymity and pseudonymity, unlinkability of data or actions, unwanted intrusions into a "personal space," and so on. Many of these notions are unrelated (or very distantly related) to traditional access control, and would consequently be difficult or impossible to derive in an automated way from a typical access control policy (such as an XACML policy). In the context of our architecture, however, such privacy notions are not included in a standards-compliant P3P policy and so this potential "semantic gap" between traditional access control and broader notions of privacy does not arise. Thus, automated translation from an XACML policy to a P3P policy is entirely possible (once a few elements have been hard-coded into the XSLT document, as discussed above).

User Agent

Privacy Bird was an extension for Internet Explorer (versions 5.01, 5.5, 6.0) that was developed and released in the early 2000's by AT&T Corp. (Cranor, 2002b; Privacy Bird, 2002). This extension examined the P3P policy of a visited website and compared it with the user's privacy preferences. If there was no conflict between the policy and the preferences, an icon in the upper-right corner of the browser window would be a bird singing happily, whereas any conflict would instead show an icon of an upset bird cawing like an angry crow. Privacy Bird was not maintained after IE 6.0 and so it does not run on current browsers. For our proof-of-concept implementation, we needed to create our own browser extension that replicated and augmented the functionality of Privacy Bird (our implementation can be found at https://github.com/FranciscoAT/priv_bird and is publicly available for review and further development). We decided to build an extension for Chrome, rather than IE, because according to w3schools.com, the most used web browser is Chrome: in 2019, there are over 79% Chrome users while only 4% of users use IE/Edge (w3schools.com, 2019b).

We created a sample website which can be populated with one of several possible P3P policies that we designed for our experiments. When the user opens Google Chrome and visits our website, the badge icon will change color according to the following conditions:

- Red: the P3P policy conflicts with the user's preferences in at least one area;
- Yellow: possible conflicts have been found between the policy and the preferences; the user is encouraged to consult the human-readable policy for further details;
- Green: the P3P policy is valid and is aligned with the user's preferences.

In the cases where the icon is Red or Yellow, the user can click on the icon to display a pop-up listing the conflicts (or possible conflicts) found. At this point, the user can decide to remain on the site and continue with filling out the form, or leave the site immediately (in order to protect his/her privacy).

Our sample website was a simple page requesting the user's name, credit card number, address, e-mail, and phone number. The user's privacy preferences were created and edited using a tool on the user's machine with the interface shown in **Figure 2**.

Various P3P policies were created to comply or conflict with the current saved preferences file so that the functionality of our Chrome extension could be tested thoroughly.

One challenge in getting our extension to work efficiently was to resolve the mismatch between the stored data files (i.e., the user preferences and the P3P policy) and the extension background script file (which was written in JavaScript). For the user preferences, we had the option of using HTML web storage (which stores data as strings for either a single session or indefinitely), or using Chrome.storage API [which stores data as objects and is "optimized to meet the specific storage needs of (Chrome) extensions" (Chrome storage, 2019)]. We chose Chrome.storage API for our implementation primarily because of its object storage capability.

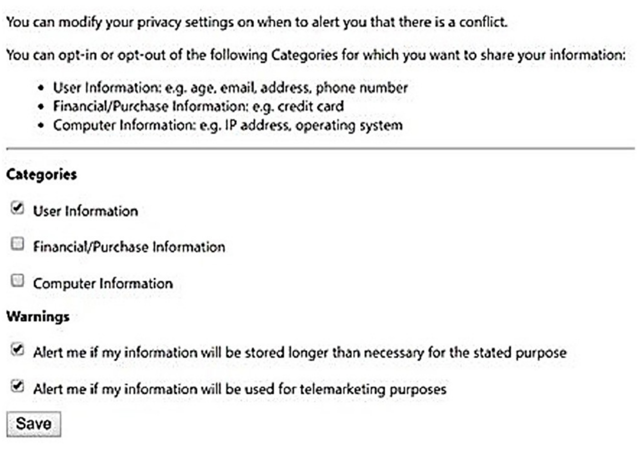
For the P3P policy, the extension downloads it from the website in XML format, but this is difficult to parse (i.e., it is difficult to retrieve necessary elements and attributes) using JavaScript. A preferable syntax is JSON because "JSON is quicker to read and write and uses arrays" (w3schools.com, 2019a), whereas using an XML format would require us to "use an XML DOM to loop through the [policy] to extract values and store in variables" (w3schools.com, 2019a). We found a simple-to-use **xmlToJSON** function on Kushagra Gour's GitHub that was last updated in 2019 (Gour, 2019) and used this for our implementation. The function **getLocalChromeValues** returns the user's preferences, and the function **compareStatement** determines whether the preferences match the JSON-encoded P3P policy. Specifically, within **compareStatement**, the following checks are made.

- Retention: the extension tries to find information where the retention is equal to "legal-requirement" or "indefinitely." This means that the user's data may be stored for longer than the specific stated purpose of the company.
- Purpose: the extension tries to find whether the company collects the user's data for telemarketing purposes.
- Categories: the extension tries to find whether the company collects user, financial, or computer data.

If conflicts are found, the badge icon is set to the appropriate color and an explanation can be shown in a pop-up window. The above checks would of course be expanded significantly in a real product, but for our proof-of-concept implementation this was sufficient to demonstrate the required functionality.

IBE

As discussed in section method: proposed extended architecture above, we wish to mitigate the risk of a website that posts a P3P policy but does not behave internally in accordance with that policy. Inspection by an external auditor such as a Privacy Seal organization is an important component of this, but we don't want the user to simply put all his/her trust in an icon on the web page. Policy-Based Encryption (PBE) appears to be a useful technology to ensure that internal company behavior


You can modify your privacy settings on when to alert you that there is a conflict.

You can opt-in or opt-out of the following Categories for which you want to share your information:

- User Information: e.g. age, email, address, phone number
- Financial/Purchase Information: e.g. credit card
- Computer Information: e.g. IP address, operating system

Categories

[✓] User Information

[ ] Financial/Purchase Information

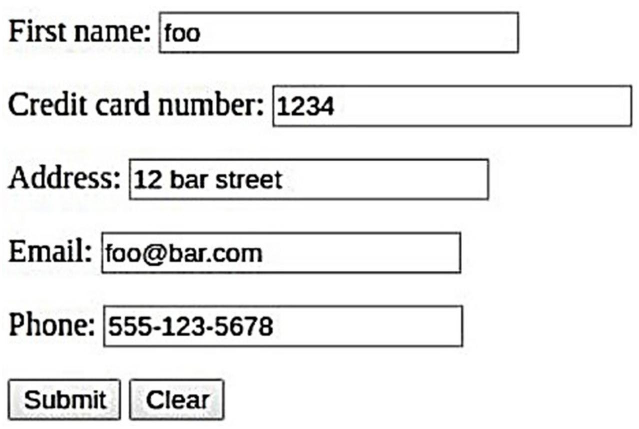
[ ] Computer Information

Warnings

[✓] Alert me if my information will be stored longer than necessary for the stated purpose

[✓] Alert me if my information will be used for telemarketing purposes

FIGURE 2 | User privacy preference editing tool.


First name: foo

Credit card number: 1234

Address: 12 bar street

Email: foo@bar.com

Phone: 555-123-5678

FIGURE 3 | Original form data.

conforms to its advertised policy, as attested to by an audit: if the P3P maps to a unique encryption key, then user data can be encrypted using that key prior to being transmitted to the website, and the site will only be able to decrypt the data if the company has successfully passed the audit. In this scheme, the Privacy Seal organization acts as the IBE/PBE Third Party Authority (TPA), with its own Master key pair (public and private); the TPA generates all system parameters for the cryptographic algorithms and performs the function of Private Key Generator (PKG) so that it can supply the company with the private key that corresponds to its P3P policy if the audit is successful. Note that it may be useful to ensure that the posted P3P policy cannot be accidentally or maliciously changed by any party (since this would prevent the legitimate organization from decrypting user data that is sent to it). One simple way to ensure this is for the organization to digitally sign the generated P3P policy before it is posted. Using W3C's "XML Signature Syntax and Processing" specification (XMLSig, 2013; which includes a specified XML canonicalization transform guaranteeing that semantically identical XML documents produce identical bit strings) will prevent any P3P modifications from being accepted by any party.

On the client side, our extension first checks whether the website has a P3P policy. If so, we use the IBE library Node Package Manager "ibejs" version (ibejs, 2016) to encrypt the client data before it is transmitted to the site. Because of the way Chrome security policies work, we are unable to simply change the form data before the outgoing message is sent. Rather, we need to make a new form in JavaScript, fill it in appropriately, and send that form to the same POST location. Note, however, that the form it creates is hidden so that the user does not see anything unusual (i.e., to the user, it appears that the form transmission occurs normally).

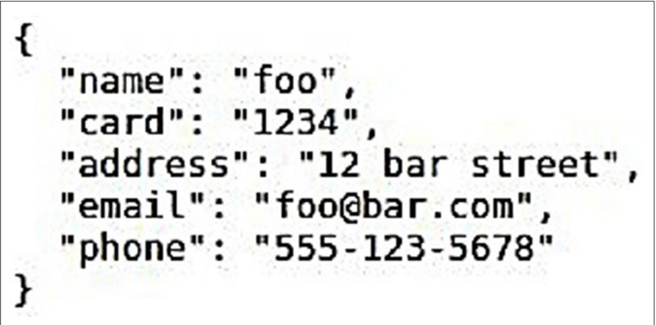
On the server side, we created a separate Remote Site that plays the role of the external auditor. This remote site is the only entity that knows the Master private key, and it uses this to generate a company private key when given a P3P policy as the public key (using IBE cryptographic operations). The company makes a call to Remote Site to obtain the company private key; the key is returned to simulate a successful audit, or is not returned to simulate an unsuccessful audit. If the company acquires the company private key, the user's data can be decrypted. Note that if the website has no P3P policy, our implementation proceeds without privacy enforcement (i.e., the user goes to the website, fills in the form data, and submits it directly to the website).

Figures 3–5 show the original form data (filled in by the user at the browser), the ciphertext that is transmitted to the server, and the successfully decrypted plaintext at the server end, respectively.

Our proof-of-concept implementation is a relatively straightforward use of the underlying IBE toolkit, but it is not difficult to envision more sophisticated approaches that, for example, encrypt only the sensitive information in the web form (leaving non-sensitive data in plaintext), or use a hybrid approach in which the P3P policy is only used to encrypt a randomly-generated symmetric key and the symmetric key is used to encrypt the actual user data.

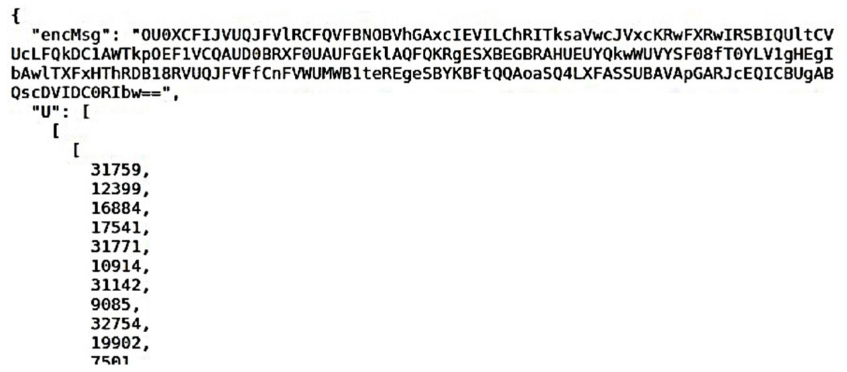
SUMMARY AND FUTURE WORK

Given society's growing dependence on digital technology, as well as increasingly strengthened legal requirements around the privacy of personal information, there is a need for



```
{
  "name": "foo",
  "card": "1234",
  "address": "12 bar street",
  "email": "foo@bar.com",
  "phone": "555-123-5678"
}
```

FIGURE 5 | Decrypted data at website.



```
{
  "encMsg": "OU0XCFIJVUQJFVlRCFQVFBNOBVhGAxcIEVILChRITksaVwcJVxcKRwFXRwIRSBIQUltCV
UcLFQkDC1AWTkpOEF1VCQAUD0BRXF0UAUFGEklAQFQKRgESXBEGBRAHUEUYQkwWUVYSF08fT0YLV1gHEgI
bAwlTXFxHThRDB18RVUQJFVFfCnFVWUMWB1teREgeSBYKBFtQQAoaSQ4LXFASSUBAVApGARJcEQICBUgAB
QscDVIDC0RIbw==",
  "U": [
    [
      [
        31759,
        12399,
        16884,
        17541,
        31771,
        10914,
        31142,
        9085,
        32754,
        19902,
        7501
```

FIGURE 4 | Ciphertext received by website after JSON parse (only a portion is shown in this figure).

techniques that will safeguard sensitive user data in many types of environments. This paper describes a comprehensive architecture for privacy enforcement at Internet websites, extending the earlier APEX model by adding IBE/PBE to the XACML, XSLT, P3P, Privacy Bird, and Privacy Seal components to mitigate the threat of malicious websites that deceive users into believing that an external audit has been successfully conducted. We furthermore do a full proof-of-concept implementation to show that this extended architecture is both usable and effective.

Directions for further research in this area focus primarily on completeness and efficiency. In particular, it would be useful to expand our XSLT Stylesheet so that it could input any XACML document appropriate for a typical Internet website and output a P3P document tailored for that website; our implementation experimented with only a few specific source and target documents. In addition, our user preferences page should be enhanced to support the remaining categories that the P3P specification defines, as well as to include more options in the Warnings section for the different types of alerts that a user may wish to see (i.e., beyond telemarketing and data retention); coupled with this, the **compareStatement** function would need to be modified to support all the possible categories. Finally, as mentioned in section results, various methods can be explored to increase the performance of the data encryption step, especially for cases in which the user is sending a large amount of data to the website.

The work described in this paper suggests that privacy enforcement at Internet websites is possible. With GDPR and similar legislation now in force in many places around the world, there is a greater need than ever before for Privacy Enhancing Technologies (PETs) of this kind.

DATA AVAILABILITY STATEMENT

The project described in this paper has two main components: an XACML to P3P translator using XSLT (platform-independent, written in XML, with a project link at https://github.com/TianyouDai/XACMLtoP3P); and a Chrome browser extension (platform-independent (but run on Windows), written in JavaScript, with a project link at https://github.com/FranciscoAT/priv_bird). There are no restrictions on the use of this project.

AUTHOR CONTRIBUTIONS

YD and SM did all the implementation of the XACML to P3P transformation using XSLT. CD, NN, and FT did all the implementation of the Chrome extension and the user preferences. CA proposed the project, supervised the research and implementation work, and wrote major portions of the final manuscript.

FUNDING

This work described in this paper was partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) through Discovery Grant #03649.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fcomp.2020.00002/full#supplementary-material

REFERENCES

Adams, C., and Barbieri, K. (2006). "Privacy enforcement in E-services environments," in *Privacy Protection for E-Services*, ed G. Yee (Hershey, PA: Idea Group Publishing), 172–202. doi: 10.4018/978-1-59140-914-4.ch007

Agrawal, R., Kiernan, J., Srikant, R., and Xu, Y. (2002). "Hippocratic Databases," in *Proceedings of the 28th VLDB Conference* (Hong Kong). doi: 10.1109/TITB.2009.2029695

APPEL (2000). "A P3P Preference Exchange Language (APPEL)," in *WWW Consortium, P3P Preference Interchange Language Working Group, W3C Working Draft*. Available online at: http://www.w3.org/TR/P3P-preferences (accessed January 24, 2020).

Ashley, P., Hada, S., Karjoth, G., Powers, C., and Schunter, M. (2003). *Enterprise Privacy Authorization Language (EPAL 1.2). W3C Member Submission*. Available online at: http://www.w3.org/Submission/2003/SUBM-EPAL-20031110/ (accessed January 24, 2020).

Ashley, P., Powers, C., and Schunter, M. (2002). "From privacy promises to privacy management: a new approach for enforcing privacy throughout an enterprise," in *Proceedings of the 2002 New Security Paradigms Workshop* (Virginia Beach, VA). doi: 10.1145/844102.844110

Beres, Y., Bramhall, P., Casassa Mont, M., Gittler, M., and Pearson, S. (2003). *Accountability and Enforceability of Enterprise Privacy Policies. HPL-2003-119.* Palo Alto, CA: Trusted Systems Laboratory (TSL), Hewlett-Packard Laboratories.

Boneh, D., and Franklin, M. (2003). Identity based encryption from the Weil pairing. *SIAM J. Comp.* 32, 586–615. doi: 10.1137/S0097539701398521

Canovas Izquierdo, J. L., and Salas, J. (2018). "A UML profile for privacy enforcement," in *International Workshop on Security for and by Model-Driven Engineering (SecureMDE)*. Available online at: https://modeling-languages.com/a-uml-profile-for-privacy-enforcement/ (accessed August 9, 2019). doi: 10.1007/978-3-030-04771-9_46

Chrome storage (2019). *Chrome.storage*. Available online at: https://developer.chrome.com/apps/storage (accessed April 17, 2019).

Cranor, L. F. (2002a). *Web Privacy with P3P*. (Sebastopol, CA: O'Reilly & Associates, Inc).

Cranor, L. F. (2002b). *Privacy Bird User Study*. Available online at: https://www.w3.org/2002/p3p-ws/pp/privacybird.pdf (accessed April 17, 2019).

Federal Trade Commission, Privacy and Security Enforcement (2019). See Available online at: https://www.ftc.gov/news-events/media-resources/protecting-consumer-privacy/privacy-security-enforcement (accessed August 9, 2019).

FreeFormatter (2018). *Free Online Tools for Developers*. Available online at: https://freeformatter.com/ (accessed January 24, 2020).

General Data Protection Regulation (GDPR) (2018). See Available online at: https://gdpr-info.eu/ (accessed August 9, 2019).

Global Privacy Enforcement Network (2013). See Available online at: https://www.privacyenforcement.net/ (accessed August 9, 2019).

Gour, K. (2019). *Function to Convert XML to JSON*. Available online at: https://gist.github.com/chinchang/8106a82c56ad007e27b1 (accessed April 18, 2019).

Henze, M., Hermerschmidt, L., Kerpen, D., Häußling, R., Rumpe, B., and Wehrle, K. (2014). "User-driven Privacy Enforcement for Cloud-based Services in the Internet of Things," in *The 2nd International Conference on Future Internet of Things and Cloud (FiCloud-2014).* Available online at: https://arxiv.org/ftp/arxiv/papers/1412/1412.3325.pdf (accessed August 9, 2019). doi: 10.1109/FiCloud.2014.38

ibejs (2016). *ibejs—Identity Based Encryption JavaScript library*. Available online at: https://www.npmjs.com/package/ibejs (accessed April 19, 2019).

Jiang, H., and Bouabdallah, A. (2017). "JACPoL: A Simple but Expressive JSON-Based Access Control Policy Language," in *IFIP International Conference on Information Security Theory and Practice* (Springer LNCS), 56–72. Available online at https://link.springer.com/chapter/10.1007/978-3-319-93524-9_4 (accessed August 9, 2019). doi: 10.1007/978-3-319-93524-9_4

Kučera, M., Tsankov, P., Gehr, T., Guarnieri, M., and Vechev, M. (2017). "Synthesis of Probabilistic Privacy Enforcement," in *ACM Conference on Computer and Communications Security* (Dallas, TX). Available online at: https://acmccs.github.io/papers/p391-kuceraA.pdf (accessed August 9, 2019). doi: 10.1145/3133956.3134079

Organization for Economic Co-operation and Development (OECD) (2006). *Report on the Cross-Border Enforcement of Privacy Laws (2006).* Available online at: http://www.oecd.org/sti/ieconomy/37558845.pdf (accessed August 9, 2019).

Oxygen XML Editor (2020). *SyncRO Soft SRL*. Available online at: https://www.oxygenxml.com/ (accessed January 24, 2020).

P3P (2002). "The Platform for Privacy Preferences 1.0 (P3P1.0) Specification," in *WWW Consortium, W3C Recommendation*. Available online at: https://www.w3.org/TR/P3P/ (accessed January 24, 2020).

Privacy Bird (2002). *Privacy Bird*. Available online at: http://www.privacybird.org/ (accessed April 17, 2019).

Shamir, A. (1984). "Identity-based Cryptosystems and Signature Schemes," in *Proceedings of CRYPTO '84, LNCS 196* (Berlin: Springer-Verlag), 47–53. doi: 10.1007/3-540-39568-7_5

Tripp, O., and Rubin, J. (2014). "A Bayesian Approach to privacy Enforcement in Smartphones," in *23rd Usenix Security Symposium*. Available online at: https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-tripp.pdf (accessed August 9, 2019).

TRUSTe Privacy Seal Program (2019). See http://www.truste.com.

Visual Studio (2019). *Microsoft.* Berkeley, CA: ISI ResearchSoft.

w3schools.com (2019a). *JSON vs XML*. Available online at: https://www.w3schools.com/js/js_json_xml.asp (accessed April 18, 2019).

w3schools.com (2019b). *Browser Statistics*. Available online at: https://www.w3schools.com/browsers/ (accessed April 20, 2019).

XACML (2017). *eXtensible Access Control Markup Language (XACML) Version 3.0 Plus Errata 01*. Available online at: http://docs.oasis-open.org/xacml/3.0/errata01/os/xacml-3.0-core-spec-errata01-os-complete.pdf (accessed August 9, 2019).

XMLSig (2013). *XML Signature Syntax and Processing Version 1.1, W3C Recommendation.* Available online at: https://www.w3.org/TR/xmldsig-core/ (accessed December 5, 2019).

XSLT (2017). *XSL Transformations (XSLT) Version 3.0, W3C Recommendation*. Available online at: https://www.w3.org/TR/2017/REC-xslt-30-20170608/ (accessed August 9, 2019).

xsltproc — command line xslt processor (2019). *Xmlsoft*. Available online at: http://xmlsoft.org/XSLT/xsltproc.html (accessed March 05, 2019).

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.