Policy-Carrying Data: A Privacy Abstraction for Attaching Terms of Service to Mobile Data

Stefan Saroiu, Alec Wolman, Sharad Agarwal
Microsoft Research

Abstract: *Despite decades of work on privacy-protecting systems, mobile user privacy remains at the mercy of cloud service providers. This paper proposes a different approach – let users attach Terms of Service (ToS) to their data before uploading it to the cloud. We propose an abstraction, called policy-carrying data (PCD), that lets users specify and attach ToS to their data. PCD guarantees that cloud providers claim they are compliant with the ToS policy before they are able to access the data. To offer this guarantee, PCD relies on attribute-based encryption. We present PCD's semantics, its properties, and describe how PCD can be added to JSON or REST. Our hope is that PCD opens a different research path – designing privacy abstractions that provide legal ammunition for mobile users against misuse of their data.*

1. INTRODUCTION

Despite its importance, protecting the privacy of mobile users' data stored in the cloud remains an elusive goal. The current landscape is very one-sided: cloud providers maintain control over how users' data is gathered, stored, managed, and used. On the other side, users are given only two choices. One is to abandon using the cloud and all apps' functionality requiring access to private data (e.g., GPS locations, users' profiles, etc...). The other choice is to lose control of their data and simply trust the cloud providers to treat private data sensibly.

This lopsided situation is further exacerbated by the business model of many cloud providers, where they partner with third-party ad networks to generate revenue. Both cloud providers and their partner ad networks have been shown to aggressively mine private data in ways that erode the privacy of most mobile users [16, 20, 12]. Even worse, there is evidence that cloud providers are forced by their local governments to further violate customers' privacy in the name of national security [14].

In response to this "privacy crisis", systems and tools have been developed to offer strong privacy protection for users' cloud data, such as information flow control [31, 7, 11], secure and trusted operating systems [29, 6, 18, 17], secure hypervisors [28, 23, 26, 32], and novel anonymization and encryption schemes [30, 8, 9, 22, 21, 25]. Despite their high degree of technical sophistication, these tools have yet to empower users with control over their cloud data.

This paper considers an alternative to building systems with strong cloud privacy protections. We advocate a much simpler approach – let users attach *terms of service* (ToS) to their data before it is uploaded to the cloud. Such a solution is similar to how valuable data is treated in other cases. For example, websites routinely publish their ToS dictating how users must treat the websites' data, programmers attach licenses to their code before publishing it online, applications ask users to click on a EULA before installation, and DVD producers force viewers to watch a short "do not copy this DVD" preview before any content is viewed. While these solutions cannot actually prevent data from misuse, they are *legally binding*: data owners can take legal action when a violation is detected. This provides a much-needed re-leveling of the playing field because it gives users additional leverage against cloud providers.

We believe that attaching terms of service to data is a problem of technical nature that the research community can help solve. We propose an abstraction, called *policy-carrying data (PCD)*, that helps to implement such privacy mechanisms. PCD binds a user's data to a policy that specifies the conditions under which data can be used, and offers the following guarantee: the cloud provider must explicitly opt-in to the user-specified policy before it can even access the data. As a result, if policy violations are discovered, the cloud provider cannot claim a lack of knowledge of user's desired policy associated with the data.

To offer this guarantee, PCD relies on encryption. While encryption is typically used to protect data confidentiality, PCD uses encryption in a different way: to force the decrypting party (i.e., the cloud) to claim it is compliant with the policy attached to the data. This is done using ciphertext-based attribute-based encryption (CP-ABE), a form of encryption that can be loosely thought as "encrypting data with a policy". To decrypt successfully, cloud providers must construct a list of attributes compliant with the policy specification; if the attributes are not compliant, decryption fails. An additional benefit of CP-ABE is that it allows policies to be specified in a human-readable form, such as XML.

While PCD does not guarantee that cloud providers do not abuse private data, it provides one form of a customer-dictated "EULA" listing the conditions under which data can be used. An example policy indicates that a GPS reading can only be used once and then must be deleted, or that it cannot be shared with any third-party; another example indicates that photos can only be stored on servers located inside the US. Cloud providers could choose to violate these policies. However, when policy violations are caught, they are likely to be much costlier to cloud providers because the violations were done knowingly and deliberately. We believe that this mechanism increases the likelihood that cloud providers will treat

HotMobile'15, February 12–13, 2015, Santa Fe, New Mexico, USA.

http://dx.doi.org/10.1145/2699343.2699357.

Policy	Specification
1	data_retention_limit = one time *and* service_name = Bing maps *and* anonymization_scheme = k-anonymity
2	data_retention_limit = one day *and* is_BitLocker_present = true *and* share_with_3rd_party = false
3	(data_location = EU *and* share_with_3rd_party = true) *or* (data_location = US and share_with_3rd_party = false)

Table 1: Examples of more sophisticated policies.

Data	Precision	Use	Threat Model	Location	Retention
meta-data	precise	actual service	national govnmt.	US	one-time
contents	k-anonymity	improve service	foreign govnmt.	Canada	one day
password	l-diversity	targeted ads	HDD theft	EU	one year
location	t-closeness	3rd-party	memory attacks	Asia	until accnt. is closed
payment	diff. privacy			Caymans	forever

Table 2: Taxonomy of attributes for PCD.

their mobile users' data according to their wishes. Cloud providers may choose to offer degraded service to those users that specify overly onerous ToS.

2. TOS-BASED POLICIES

2.1 Brief Background on ToS

Terms of Service (ToS) are a set of rules which one must agree to abide by in order to use a service. Websites often define ToS to state their users' rights and responsibilities as well as the website's limitations of liability. ToS can be subject to change; whenever the ToS change, websites must once again seek the consent from their users [13].

ToS are often long and complicated. It is widely believed that Web users often accept the websites' ToS without understanding them or even bothering to read them. ToS;DR [2] is a recent project that aims at reviewing all websites' ToS policies and rating them according to a color-coding scheme. With their tools, whenever a user encounters a new ToS, the user can immediately evaluate the ToS's restrictiveness by its color shade: solid green meaning "best ToS" to solid red meaning "this ToS raises very serious concerns".

Courts have overwhelmingly sided with enforcing online ToS [19]. There are a few exceptions when courts ruled against enforcing the ToS. The courts aimed to protect consumers from certain clauses they considered unreasonable, such as *onerous arbitration* clauses (i.e., in case of a dispute, the user agrees to settle it only through the arbitration mechanism described by the ToS) and *forum selection* clauses (i.e., any litigation resulting from the contract will be initiated in a specific jurisdiction or court). These clauses are not considered unreasonable when the plaintiff is a business because courts presume that businesses are "sophisticated economic entities" and know what they are doing when accessing another company's website.

2.2 Examples of Policies

Many ToS-based policies are possible. Some users would prefer policies that are simple and easy to understand. Example of simple policies that appeal to many users are:

1. When uploading a credit card number, a user may attach a policy that restricts its use to a single transaction. With such a policy, the cloud provider can use the credit card for a one-time charge, but is not allowed to store the number for future transactions.

2. When uploading a personal photo to a social networking site, a user may attach a policy forbidding any attempt to interpret the photo's content beyond simply rendering the photo, such as performing face recognition or object detection on the photo.

3. When uploading personal health data to the cloud (e.g., a mobile app that measures the user's pulse, or any form of health care records), a user may attach a policy that forbids sharing with a third-party, and requires data to always be stored in encrypted form.

4. When using a mobile payment application, a user may attach a policy that forbids any form of customer profiling on the data.

Privacy-savvy users may define more sophisticated policies on their data that go beyond these simple examples, and describe features related to the software and hardware they require from cloud providers. For example, policies can use the following types of features:

1. Features related to data privacy. Examples are data retention limits, the degree of data shareability (e.g., whether it can be shared with 3rd parties), or the type of anonymization scheme required (e.g., hashing, k-anonymity, differential privacy).

2. Software-related features. Examples are: encrypted file-system (e.g., BitLocker), verified OSes (e.g., seL4 [18], IronClad [17]), or specific versions of software.

3. Physical/Hardware-related features. Examples are: geographic locations of data-centers, or the presence specialized hardware such as a Trusted Platform Module (TPM) or a Hardware Security Module (HSM).

Table 1 shows a few examples of more sophisticated policies. Each policy is a set of constraints linked by conjunction or disjunction operators. Each constraint tests a condition over an attribute which can be a string or a number. The condition can be an equality (e.g., attribute=value) or an inequality in case of numbers that span a *finite set*. Examples of finite sets are the set of past released version numbers for a piece of software, or the set of days of the week. Unfortunately, the cryptography underlying our abstraction cannot support infinite or uncountable sets. The set of natural numbers is countably infinite, and the set of real numbers is uncountable.

2.3 Taxonomy for Constructing Policies

Our vision is that a common set of policies will become de-facto standards that users will pick from when requesting protection for their data. This is similar to how Creative Commons (a non-profit organization) has defined a set of simple licenses for sharing content, or how ToS;DR has defined a color-coded scheme for interpreting EULAs.

Table 2 presents one taxonomy example as a starting point. Broadly, users may apply the same policies to most of their meta-data, such as call logs and web access logs, and a separate set of policies to data contents, such as photographs and chat messages. We expect users to treat a few data items differently – location, credit card numbers, and passwords. Each data item may have multiple policies that allow different uses depending on the granularity of the data.

3. MODEL AND SEMANTICS

The policy-carrying data (PCD) abstraction allows a mobile user's data to be bound to a user-defined policy. PCD offers two primitives: *encapsulate* and *descapsulate*. Encapsulate is performed by a user and takes as input the privacy-sensitive data and a policy, and outputs ciphertext. The reverse operation, decapsulate, is done by the cloud provider and takes as input the ciphertext. By construction, the decryption keys correspond to the set of policies the cloud provider claims it adheres to. Decapsulate decrypts properly *if and only if* the cloud provider claims it is compliant with the policy specified at encapsulation time, during encryption.

3.1 Model

With our abstractions, each cloud provider has a configuration, which is a set of human-readable attributes from a taxonomy of attributes (a strawman example was presented in Table 2). This configuration is published to a trusted provider, such as a certificate authority (CA). This is a one-time step; the CA generates a set of credentials based on these configurations and passes them to the cloud provider. PCD guarantees that the credentials can decapsulate only data whose policy is met by the configuration of the cloud provider.

The explicit step of publishing the configuration offers assurance because the cloud states to a third-party (i.e., the CA) a set of privacy measures. For example, the cloud can state that it never stores credit card information, it does not perform face recognition, or that it uses highly secure software and hardware to handle users' data.

However, PCD also allows for the cloud provider to act as a CA. This offers a weaker form of assurance because the cloud does not have to reveal its configuration to a third-party. Section 4 will provide a more in-depth discussion of the separation between the cloud provider and the CA.

3.2 PCD-based ToS

PCD lets users specify their own terms of service. The same way how websites require their users to click "Agree" on the ToS, PCD require websites to interpret their user's policy to access and use their data. We believe this requirement is sufficient to render the policy attached to the data as legally binding. When violations occur, customers (whether users or other businesses) can take the website to court.

While customers have the option of constructing their own policies for the PCD abstraction, it also possible for a third-party (similar to ToS;DR) to pre-construct a fixed set of policies and label them, perhaps using a color-coding scheme. A green policy could represent a case when the customer uses a non-restrictive policy (e.g., the website can use the GPS location for showing ads), where as a red policy a very restrictive policy (e.g., the website must discard the GPS location as soon as the Web request is answered).

3.3 Trade-off: Policy Restrictiveness vs. Level of Service

The question raised now is: "Why would a user choose any policy other than the most restrictive one so that the user's privacy is maximized?" While the user is free to select any policy, the service provider may use policies to determine what quality of service to offer their users. Depending on the service, the provider may have a number of quality knobs that can be adjusted:

- auto policy discount
- location service accuracy
- search query accuracy
- freshness of web service data
- personalization features
- speech recognition accuracy
- online storage size

For example, if a user restricts their auto insurance company from retaining GPS traces gathered from their phone, then the insurance company may not give the user certain policy discounts. This enables a trade-off between the user's need for privacy and the insurance company's need for risk mitigation. Similarly, an online file storage service may offer less free storage capacity to a user who restricts their data to be stored only in certain datacenter locations. This allows a trade-off between the user's need for protection (e.g, from certain governments) and the storage provider's need for limiting cost in expensive datacenter locations.

3.4 Bootstrapping PCD

One obstacle to deploying PCD is requiring cloud providers to implement our abstraction. One possibility to help bootstrap our system is via a proxy. This proxy could interpret the policies attached to the users' data and decide which cloud provider is best suited with meeting this policy. Effectively, the proxy's role is to construct a set of attributes and values that correspond to each cloud provider. While such a proxy could help speed the adoption of PCD, its existence also raises privacy risks because it would be exposed to all its users' data.

4. CRYPTOGRAPHIC SUPPORT

PCD relies on Ciphertext Policy Attribute-Based Encryption (CP-ABE), a type of public-key encryption in which data is encrypted using a policy, and the decryption keys is dependent on a set of attributes. Decryption is possible only if the set of attributes satisfies the policy. PCD relies on a certificate authority (CA). The CA generates a master private key and a master public key. Encryption is done using the master public key and a user-specified policy. A cloud provider must present a set of attributes to the CA. The CA uses its master secret key to return the cloud provider's decryption key embedding the corresponding attributes. The decryption key can be used successfully against all ciphertexts whose policies are satisfied by the cloud provider's set of attributes.

These are the steps to use CP-ABE in the content of policy-carrying data:

1. CA generates public and private master keys, ($\text{MasterK}_{\text{pub}}$ and $\text{MasterK}_{\text{priv}}$)
2. Cloud provider presents the set of attributes to the CA ($\text{Attrib}_{\text{provider}}$). CA uses private master key $\text{MasterK}_{\text{priv}}$ to generate a decryption key embedding these attributes ($K(\text{Attrib}_{\text{provider}})$).
3. User encrypts data D with policy P and CA's public master key $\text{MasterK}_{\text{pub}}$, producing ciphertext C. User uploads C to provider.
4. Provider decrypts C using its decryption key $K(\text{Attrib}_{\text{provider}})$. If decryption is successful, provider's attributes match the user-specified policy P.

The separation between the CA and the cloud provider (i.e., the principal attempting to decrypt) is crucial to the security of CP-ABE because the CA must guard the secrecy of the secret master key. However, in our case, CP-ABE is used only to guarantee that the cloud provider interprets the user's privacy policy. This guarantee can be met even when the cloud provider also acts as the CA. To decrypt, the cloud provider must use the master secret key and a set of attributes to generate a decryption key. This step is where

Figure 1: CP-ABE: on the left, the CA is separate from the cloud provider; on the right, the CA and the cloud provider are the same. (icons by Freepik [1]/CC BY 3.0.)

the PCD guarantee is met: the cloud provider must use a set of attributes meeting the policy specification to generate a decryption key capable of decrypting the ciphertext.

Figure 1 illustrates the two cases: on the left the CA is a separate entity than the cloud provider, on the right the cloud provider and the CA is the same entity. The case on the left is important to ensure the security of CP-ABE. On the right, ABE is insecure because the cloud provider is the CA, and thus possesses the master private key.

Both cases provide the PCD guarantee – the cloud provider must generate a decryption key whose attributes match the policy, thus having to interpret the policy. However, there is an additional important distinction. On the left, the cloud provider states its configuration to a third-party, a step that can be independently verified. For example, an external audit could verify the configurations claimed by a cloud provider; such an audit could be useful to settle disputes. On the right, the cloud provider takes no external, independently verifiable step to reveal its privacy configurations to a third-party.

4.1 Why CP-ABE?

A legitimate question is why does PCD need to rely on a relatively uncommon form of encryption (CP-ABE)? Why can't PCD use a more common cryptographic scheme, like RSA? To answer these questions, we start by first describing a possible implementation of PCD using RSA and then list its drawbacks.

Mobile users can take their polices and the data to upload and XOR them together. This XOR-ed blob (i.e., policy $\oplus$ data) could be then encrypted with the cloud provider's public key. The user uploads both the ciphertext and the policy. The cloud provider must use its RSA private key to obtain the XOR-ed blob, and then XOR it with the policy to decrypt the data.

$$\text{Encryption:RSA\_pubkey}(\text{policy} \oplus \text{data}) \quad (1)$$

$$\text{Decryption:RSA\_privkey}(\text{ciphertext}) \oplus \text{policy} \quad (2)$$

CP-ABE offers two advantages over traditional encryption. First, CP-ABE offers policies with multiple clauses linked by conjunctive and disjunctive operators. Traditional encryption does not extend naturally to support policies with multiple "and" and "or" operators. Second, although the cloud provider needs the user's policy to be able to decrypt, its use of the policy is mundane. The cloud provider uses the policy as one of the inputs to the decryption function without having to interpret it. It's similar to how the cloud provider uses other RSA decryption parameters, such as the type of the padding scheme or the length of the key. In contrast, with ABE, the cloud provider itself must generate the set of policies it adheres to. Only after these policies are generated, the cloud provider can obtain the decryption keys. With ABE, the cloud provider has no choice but to claim it adheres to the right policy.

5. POSSIBLE IMPLEMENTATIONS

While there is no single standard on how web services are implemented today, we believe that most Web frameworks are amenable to adding the PCD abstraction to users' data. This subsection presents preliminary designs to adding PCD to two popular Web standards: JSON [5] and REST [24].

JavaScript Object Notation (JSON). JSON is an open standard that uses human-readable text to transmit data objects in a key-value pair format. For example, a GPS location in JSON could be:

```
{
  ``firstName'': ``Barack'',
  ``lastName'': ``Obama'',
  ``latitude'': ``38.8951N'',
  ``longitude'': ``77.0367W''
}
```

Since PCD is human-readable, incorporating PCD into a JSON protocol is trivial. For example, adding policy 1 from Table 1 to this example would become:

```
{
  ``firstName'': ``Barack'',
  ``lastName'': ``Obama'',
  ``lat'': 0x53c5b77d34713801e61bd5a5b00a4aea,
  ``long'': 0xf38f927640da51fdacdb93243317b0de,
  ``PCD'': ``data_retention_limit = one time
         AND service_name = Bing Maps
         AND anonymization_scheme = k-anonymity''
}
```

Representational State Transfer (REST). In REST, data objects are identified using URIs, such as http://example.com/GPS. The common way to upload a value using REST is to issue an HTTP PUT request to the URI. For example:

```
PUT /GPS/coordinates?firstName=Barack&\
  lastName=Obama&\
  lat=0x53c5b77d34713801e61bd5a5b00a4aea&\
  long=0xf38f927640da51fdacdb93243317b0de \
  HTTP/1.1
Host: www.example.com
X-PCD: data_retention_limit = one time
       AND service_name = Bing Maps
       AND anonymization_scheme = k-anonymity
```

Other Formats. For Web services that do not follow JSON or REST, the PCD policy can be transmitted through a Web cookie. The server must read the cookie and interpret the policy before decrypting the passed-in data objects.

6. ADDITIONAL USES

In recent years, sophisticated privacy tools have been developed to control the amount of information disclosed to a website or a third-party. However, the parameterization of these tools is often under-specified or entirely under the control of the website. An additional use of PCD, beyond just specifying the desired algorithm, is to offer users a way to initialize the parameters (or the configuration) of a website's privacy tools. This section lists a set of privacy tools, their configuration parameters, and how PCD can let users define their input values.

k-Anonymity. k-Anonymity [30] maps sensitive data to a set of identifiers in such a way that they are indistinguishable among k individuals. The value of k is crucial to the privacy of this scheme – a higher value of k offers stronger privacy. With PCD, different users could choose different value of k for their data.

l-Diversity. l-Diversity [22] is an extension of k-Anonymity that aggregates sensitive data into a set of equivalence classes, such that sensitive data has diverse values within each class. l-Diversity is stronger than k-Anonymity because it reduces the likelihood of reversing the anonymization in case the sensitive data has a homogeneous distribution, or when the attacker has additional background knowledge about the data. As before, users could define the l-diversity metric the website must apply to the data classes.

t-Closeness. t-Closeness [21] partitions the sensitive data into equivalence classes in such a way that the distance between the overall distribution of sensitive values and the distribution in each class must be bounded by t. There are several metrics that measure distance between distributions, and, with PCD, users can decide on the metric.

Differential privacy (DP). DP [8, 9] provides an intuitive formalization of privacy. Given a dataset and a query, DP measures how much information is revealed by answering the query. Information is revealed when an attacker who knows the query answer is more likely to guess the existence of a data item in the dataset. Any query answered on the dataset leaks some information, however certain queries leak more information than others. The amount of privacy loss is controlled by injecting noise in the query answer.

DP frameworks offer two knobs. First, a noise knob controls how much "noise" data to inject in the query answer if set to high, the query answer has low privacy loss, but it is also more inaccurate, and vice-versa. Noise is generated dynamically for each query answer; if the same query is repeated, the answer changes from one run to another based on the random noise. The second parameter is the privacy budget of the entire dataset. The privacy lost by each query answer is deducted from this privacy budget. Once it reaches zero, the system refuses to answer any additional queries on this particular dataset. While we do not expect average Web users to be able to parameterize a DP framework, PCD still allows users to select DP configurations pre-defined by third-parties.

7. BRIEF PERFORMANCE EVALUATION

An area of concern is the performance of CP-ABE. To investigate the suitability of CP-ABE as an abstraction mechanism, we perform the following brief performance evaluation. We encrypt a 1KB data item using different policies with increasing levels of complexity. We vary the number of attributes in the policy from one to ten and measure the performance of encrypt and decrypt of CP-ABE.

Our setup uses an NVIDIA Jetson Tegra K1 platform equipped with 2 GB of RAM and an 4-Plus-1 quad-core ARM Cortex A15 CPU running at a maximum clock speed of 2.3 GHz. We use a publicly-available implementation of CP-ABE found at `http://hms.isi.jhu.edu/acsc/cpabe/`, and we repeat each experiment 100 times and report the average. We checked that all our experiments have low variance.

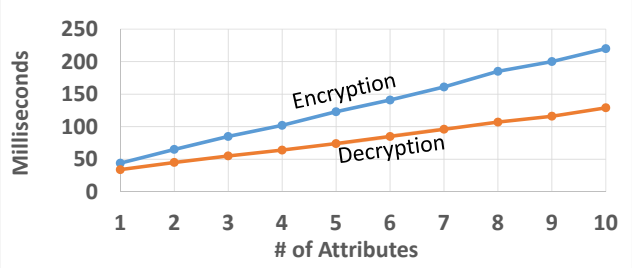

Figure 2: Performance of CP-ABE encryption and decryption.

Figure 2 illustrates the performance of encrypt and decrypt as a function of policy complexity. This Figure shows two findings. First, the overhead of CP-ABE is not high; even with complex policies, CP-ABEs' performance is measured in tens of milliseconds. Second, decryption is quite fast (it is less expensive than encryption, a finding consistent with previously reported evaluations of attribute-based encryption). Fast decryption indicates that cloud providers need not worry about the performance overhead due to PCD.

8. RELATED WORK

Our PCD abstraction is inspired by Excalibur [27], which offers policy-sealed data, another abstraction for building trusted cloud services. Like PCD, Excalibur uses CP-ABE to encrypt customer data and bind it to a customer-chosen policy. However, unlike PCD, Excalibur is primarily a security mechanism. Excalibur combines a hypervisor, verified security protocols, and TPM-based attestation to ensure that customer-encrypted data can only be decrypted on cloud servers whose software and hardware configuration is compatible with the customer-specified policy. As a result, Excalibur imposes a high barrier on the cloud-service infrastructure. In contrast, our goal with PCD is just to ensure that the cloud provider explicitly opts-in to the customer policy – no heavy-weight enforcement mechanisms are necessary.

Prior work [15] has proposed middleware for anonymizing user location data along spatial or temporal dimensions. TaintDroid [10] takes an altogether different approach – it traces the flow of private user data through mobile app code to identify when it leaves a mobile device. Similarly, PMP [3] detects when apps use private data. It uses crowdsourcing to determine whether an app should have access to that data, but does not address privacy once the data leaves the app. Other work [20] has studied the economics of mobile app advertising and presents a framework for dynamically obfuscating user data to achieve a revenue target. All this work is complementary – PCD would enable users of these techniques to specify conditions on what levels of anonymity and privacy they desire once data has left the device and reached the cloud.

Privacy legal scholars have previously argued for a *contractual* approach to online privacy [4]. Their argument stems from the lack of consensus among people about *how* important privacy is. Current legal efforts to protect privacy are not sensitive to the individual levels of privacy desired by an individual. A law offers too little privacy for some, and too much for others. Instead, a contractual solution is preferable, where individuals can enter separate contracts that dictate their privacy needs. PCD offers a straightforward mechanism for implementing a contractual approach to privacy.

9. CONCLUSIONS

This paper proposes policy-carrying data (PCD), a privacy abstraction for mobile services. With PCD, users construct a policy expressing how their private data must be treated by the cloud. PCD guarantees that cloud providers claim to be compliant with the specified policy before getting access to the data. This paper describes how attribute-based encryption can be used to offer the PCD abstraction. It also provides a strawman PCD design and taxonomy, and a preliminary performance evaluation. We hope that PCD can offer an alternative approach to offering cloud services with strong privacy guarantees.

10. REFERENCES

[1] Freepik. https://www.freepik.com, 2014.

[2] Terms of Service Didn't Read. https://tosdr.org/, 2014.

[3] Y. Agarwal and M. Hall. ProtectMyPrivacy: Detecting and Mitigating Privacy Leaks on iOS Devices Using Crowdsourcing. In *ACM MobiSys*, 2013.

[4] S. Bibas. A Contractual Approach to Data Privacy. Faculty Scholarship. Paper 1016, 1994.

[5] T. Bray. RFC 7159: The JavaScript Object Notation (JSON) Data Interchange Format. http://www.rfc-editor.org/info/rfc7159, 2014.

[6] X. Chen, T. Garfinkel, E. C. Lewis, P. Subrahmanyam, C. A. Waldspurger, D. Boneh, J. Dwoskin, and D. R. Ports. Overshadow: A virtualization-based approach to retrofitting protection in commodity operating systems. In *ASPLOS*, 2008.

[7] S. Chong, J. Liu, and A. C. Myers. Sif: Enforcing Confidentiality and Integrity in Web Applications. In *USENIX Security Conference*, 2007.

[8] C. Dwork. Differential Privacy. In *ICALP*, 2006.

[9] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In *IACR Theory of Cryptography Conference*, 2006.

[10] W. Enck, P. Gilbert, B. gon Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones. In *USENIX OSDI*, 2010.

[11] D. B. Giffin, A. Levy, D. Stefan, D. Terei, D. Mazières, J. C. Mitchell, and A. Russo. Hails: Protecting Data Privacy in Untrusted Web Applications. In *USENIX OSDI*, 2012.

[12] P. Gill, V. Erramilli, A. Chaintreau, B. Krishnamurthy, K. Papagiannaki, and P. Rodriguez. Follow the money: Understanding economics of online aggregation and advertising. In *IMC*, 2013.

[13] E. Goldman. How Zappos' User Agreement Failed In Court and Left Zappos Legally Naked. Forbes – http://www.forbes.com/sites/ericgoldman/2012/10/10/how-zappos-user-agreement-failed-in-court-and-left-zappos-legally-naked/, 2012.

[14] G. Greendwald and E. MacAskill. Boundless Informant: the NSA's secret tool to track global surveillance data. The Guardian – http://www.theguardian.com/world/2013/jun/08/nsa-boundless-informant-global-datamining, 2013.

[15] M. Gruteser and D. Grunwald. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In *ACM MobiSys*, 2003.

[16] S. Guha, B. Cheng, and P. Francis. Privad: Practical privacy in online advertising. In *USENIX NSDI*, 2011.

[17] C. Hawblitzel, J. Howell, J. Lorch, A. Narayan, B. Parno, D. Zhang, and B. Zill. Ironclad Apps: End-to-End Security via Automated Full-System Verification. In *USENIX OSDI*, 2014.

[18] G. Klein, K. Elphinstone, G. Heiser, J. Andronick, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, M. Norrish, R. Kolanski, T. Sewell, H. Tuch, and S. Winwood. seL4: Formal Verification of an OS Kernel. In *ACM SOSP*, 2009.

[19] M. A. Lemley. Terms of Use. *Minnesota Law Review*, 91, 2006.

[20] I. Leontiadis, C. Efstratiou, M. Picone, and C. Mascolo. Don't kill my ads! balancing privacy in an ad-supported mobile application market. In *HotMobile*, 2012.

[21] N. Li, T. Li, and S. Venkatasubramanian. t-Closeness: Privacy beyond k-anonymity and l-diversity. In *ICDE*, 2007.

[22] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-Diversity: Privacy Beyond k-Anonymity. In *ICDE*, 2007.

[23] J. M. McCune, Y. Li, N. Qu, Z. Zhou, A. Datta, V. Gligor, and A. Perrig. TrustVisor: Efficient TCB Reduction and Attestation. In *IEEE Symposium on Security and Privacy*, 2010.

[24] C. Pautasso, E. Wilde, and R. Alarcon. *REST: Advanced Research Topics and Practical Applications*. Springer, 2014.

[25] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan. CryptDB: Protecting Confidentiality with Encrypted Query Processing. In *ACM SOSP*, 2011.

[26] H. Raj, D. Robinson, T. Tariq, P. England, S. Saroiu, and A. Wolman. Credo: Trusted Computing for Guest VMs with a Commodity Hypervisor. Technical Report MSR-TR-2011-130, Microsoft Research, 2011.

[27] N. Santos, R. Rodrigues, K. Gummadi, and S. Saroiu. Policy-Sealed Data: A New Abstraction for Building Trusted Cloud Services. In *USENIX Security Conference*, 2012.

[28] A. Seshadri, M. Luk, N. Qu, and A. Perrig. SecVisor: A Tiny Hypervisor to Provide Lifetime Kernel Code Integrity for Commodity OSes. In *ACM SOSP*, 2007.

[29] A. Shieh, D. Williams, E. G. Sirer, and F. B. Schneider. Nexus: a new operating system for trustworthy computing. In *ACM SOSP*, 2005.

[30] L. Sweeney. k-Anonymity: A Model for Protecting Privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5), 2002.

[31] S. Zdancewic, L. Zheng, N. Nystrom, and A. C. Myers. Untrusted Hosts and Confidentiality: Secure Program Partitioning. In *ACM SOSP*, 2001.

[32] F. Zhang, J. Chen, H. Chen, and B. Zang. CloudVisor: Retrofitting Protection of Virtual Machines in Multi-tenant Cloud with Nested Virtualization. In *ACM SOSP*, 2011.