2015 IEEE 17th International Conference on High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), and 2015 IEEE 12th International Conf on Embedded Software and Systems (ICESS)

Distributed Enforcement of Sticky Policies with Flexible Trust

Jordan Brown and *Douglas M. Blough*
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0765

I. Abstract

In this paper, we describe an approach to distributed enforcement of sticky policies in heterogeneous hardware and software environments. These heterogeneous environments might have differing mechanisms for attesting to their security capabilities and data sources might specify different levels of trust for different data items. Such an environment requires highly flexible policy specification and enforcement mechanisms. We employ sticky policies that travel with data wherever it travels, and we separate them into two components, a hosting policy and a usage policy. Hosting policies are used to ensure that data are transferred only to entities that are provably capable of providing local enforcement and only further transferring data under the same policies. Usage policies confer access, viewing, and update capabilities on users based on their attributes. The approach is supported by attribute-based certificates and policies, which include what authorities are trusted to certify attributes. In addition to presenting a full description of the approach, we report on a prototype implementation that includes all of the aforementioned components and also makes use of a modified version of Excel we developed to track security labels as data move through spreadsheets that are being shared by multiple users across different systems.

II. Introduction

In the information age in which we live, organizations maintain large amounts of data, much of it sensitive. In many situations, there could be great value in sharing these data with individuals or partners outside of an organization's controlled hardware/software infrastructure. For example, in the domain of medicine, health professionals would like to safely access patient information on their laptops or even mobile devices, and controlled sharing of such data for both health care and research purposes has obvious scientific and social benefits. Clearly, inadvertent disclosure of health information could have severe consequences for patients with sensitive conditions. Thus, although the benefits of information sharing are great, the risks of data breaches if the information is not properly handled after it is shared are often greater. A similar analysis can be made concerning sensitive customer information held by commercial organizations, information maintained in databases by law enforcement organizations, etc.

Organizations that allow sensitive data to be disseminated need assurances that the systems receiving the data will control access so that users only see the information appropriate to their level of authorization, maintain sensitivity labels on information as data are propagated within the system, and only further disseminate the data to other systems if those systems meet the requirements of the data owner. Enabling concepts for this vision are *sticky policies*, wherein policies on data usage are propagated along with data as they move within and between systems, and *information flow control* (IFC), where sensitive information is tagged, tags propagate with data as they move around within a system, and access is restricted to authorized individuals based on the tags.

In this paper, we address the problem of controlled information dissemination in heterogeneous distributed environments where hosts might have very different hardware and software architectures and do not have necessarily have prearranged trust relationships. Data are disseminated together with policies that are specified by the data source and can be added to by intermediaries that insert new information. A policy is cryptographically tied to its corresponding data to prevent it from being undetectably altered. In our approach, policies have two distinct components. The *hosting policy* specifies the hardware and software requirements in order for some entity to host the data. The *usage policy* specifies the rights that users have on the included data types.

Our approach permits information sharing in a flexible way. This means that hosts do not all need to use the same local IFC mechanism nor have the same infrastructure. Some hosts might have hardware support for IFC, others might have a trusted IFC-aware operating system, and still others might have application-aware IFC as long as the application is certified by a trusted software company or trusted third-party auditor. This allows different combinations of trusted components to meet requirements for managing sensitive information. Given the diverse heterogeneous nature of systems in operation today, we believe this flexibility will remove a significant barrier to wide-spread deployment and use of technologies for controlled information dissemination. This approach is enabled by certified attributes that systems can present when they request sensitive information to be shared with them. The key point is that the data source decides what attributes are required for data hosting and use, as well as how those attributes can be certified, which enables flexible information sharing.

We have built, demonstrated, and evaluated a proof-of-concept prototype implementation of our approach. The prototype includes a management component to perform remote attestation and secure document exchange, an Excel add-in that performs local information management and flow control of data in spreadsheets, and a policy decision point to make hosting and access decisions based on data owners' policies.

978-1-4799-8937-9/15 $31.00 © 2015 IEEE
DOI 10.1109/HPCC-CSS-ICESS.2015.235

The add-in enables Excel users to tag cells or groups of cells with sensitivity labels, it controls access to tagged cells, and it propagates tags across cells as necessary.

The contributions of this paper can be summarized as follows. Our work explores a novel approach to robust, distributed enforcement of sticky policies across heterogeneous systems. Our solution provides data owners with a suite of options for maintaining flexible data privacy and integrity while minimizing the overhead for enforcing the desired policies and allowing separation between data hosting privileges and data usage rights. To facilitate the distinction between hosting and usage, our system provides the capability of verifiable redaction of data, which enables less sensitive data to be shared with verifiable links to policies while maintaining security requirements on highly sensitive data.

III. Background

A. Example Use Case

An example use case comes in the form of client interaction with a cloud service provider. Consider a sample interaction with the well known service provider Google. Users may have a large variety of data that are passed through this provider when accessing its diverse set of services. It is reasonable to expect that users have the ability to define access rights to their data within and beyond the provider's boundaries. For example, when using Google Wallet, the user could specify that the credit card information provided may only be handled by trusted billing companies and banks. Data generated on mobile devices could pre-specify data usage policies and limit the use of location data by Google Maps to only intended purposes. When sending a sensitive email, the user could specify that the contents should only be visible to the specified recipients. If properly enforced, this policy should prevent recipients from forwarding the message to non-recipients. Additionally, collaborative efforts on shared documents could be equipped with viewing and distribution policies in a distributed manner which allows for the contributors to decide the privacy levels appropriate for their own content. Our distributed policy enforcement system lays the foundation for each of these scenarios as well as many others.

B. Related Work

Trusted platform computing provides one mechanism for certifying attributes of remote systems. Implementations such as Nizza [14] and Flicker [17] deal with the trusted computing base, while others such as Terra [11] and CloudVisor [27] deal with issues at the virtualization level and trust in the cloud. Excalibur [24] uses remote attestation and trusted platform module (TPM) sealing mechanisms to ensure proper execution environments of Virtual Machines (VMs) as they migrate within a datacenter. Trusted virtual domains [9], [10] extend trusted platform technology using virtualization to provide isolated execution environments that span multiple physical platforms. In [12], the authors consider information flow control across virtual domains within a single enterprise infrastructure. This work includes two levels of usage policies, the top level of which specifies constraints across different workflows (virtual domains) and the lower level of which covers rights of different users within a single workflow. This work does not include our concept of a hosting policy, which is separate from the usage policy, because there is inherent trust in the platform provided by the common enterprise infrastructure.

Another supporting technology for our approach is property-based (or attributed-based) attestation. Prior work on property-based attestation spans various computing environments, ranging from mobile [5], [16] to client-server [1] to cloud [22], [26]. To our knowledge, Sadeghi and Stüble [23] were the first to suggest the use of trusted third parties to provide attribute attestation.

Our research looks to expand on sticky policy approaches considered to date. The work in [21] focuses on logging policy consent to facilitate auditing in the event of breaches. In [4], the authors focus on mechanisms for attaching policies to data. The work developing a Data Distribution Infrastructure (DDI) [15] works to ensure distributed policy enforcement by integrating services into local enforcement infrastructures but simply assumes remote trust capabilities are present. Various other contributions to the area rely on homogeneous components [25], expect policy breaches to be prosecuted via review audits [7], work under assumptions that leave trust establishment as out of scope [13] [20] [19], or simply acknowledge that trusted attestation services could strengthen the system without considering how to incorporate them [6]. Our work looks to facilitate verifiable attribute collection, which allows informed policy decisions without strict component knowledge prior to policy authoring. This empowers heterogeneous systems support through focus on system and component attribute attestation rather than restricting system composition or assuming trusted behaviors.

Finally, the work described herein builds on our earlier work with redactable signatures [2], [3], [8]. That work uses a Merkle hash tree structure to allow selective disclosure of attributes in certificates [2], [3] and provides strong cryptographic verification of selectable subsets of data in the context of health information exchange [8].

IV. Approach Overview

The objective of our approach is to ensure that sensitive data are never accessed in an insecure manner. Obviously, the definition of insecure is different for different people and might even differ for one person when considering different types of data. Through attribute verification, our approach ensures that sensitive data is treated with the security measures specified by the data source to maintain an expected level of privacy and integrity. We refer to the distributed software system that governs data security and authorizes data transfers as the *management service* and it is comprised of various heterogeneous management components. Figure 1 depicts management components residing at various levels on heterogeneous systems. This service facilitates distributed policy enforcement and is described in Section V-B.

The management components operate under the policies attached to the data, which are described in Section V-C. These policies outline the features that must be in place to securely host sensitive data and which types of software and users may access the data. Each type of sensitive data has its own hosting and usage rules, which allow for flexible trust specification.

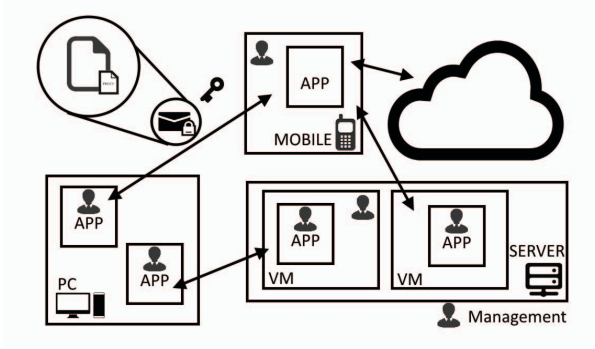

Fig. 1. High-Level Diagram of Approach

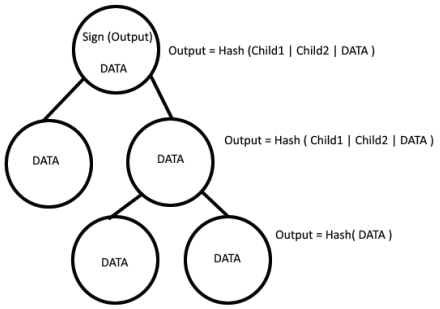

Fig. 2. Sample Hash Tree Structure

Once data are secured by a management component, it is the responsibility of that component to ensure that any new system or component requesting hosting or usage privileges meets the policy specifications. This requires attribute collection to facilitate a policy decision, which in turn requires secure communication and trusted verification of the attributes. Examples of this process are discussed in Section V-E.

Data transfer is represented by the encrypted policy-data package and accompanying key in Figure 1. The system is capable of transferring subsets of data to match the security on the receiving end of the transfer. The collected attributes of the remote management component are used by the policy to direct the redaction of data that cannot be included in the transaction to the new host. Further redactions may take place during user and application data access as defined in the usage policy attached to the data.

In a basic implementation of our approach, there is only one encryption key per data set. The management component on the sending end of a transfer first redacts all data that the receiving component is not authorized to receive (see Section V-A for redaction details), then encrypts the redacted data set using its encryption key, and then forwards the redacted data set and accompanying key to the receiving component. The receiving component thus obtains only the portion of the data for which it is authorized. It can then pass along a subset of that data to further components in keeping with the document's policy. A more complex implementation could make use of a multiple-key approach, wherein an entire data set with different portions encrypted with different keys is passed to any component but only the keys to the authorized portions are released to the component.

Note that, in the basic implementation, the process of redaction and dissemination could result in multiple versions of a data set if a component is allowed to transfer the data set multiple times to different recipients. We have also developed a mechanism for version tracking and control within this process, but due to space limitations, we omit its details. The redaction/dissemination process could also result in a component receiving a version of a data set in which some data that the component is authorized to receive have been redacted at an earlier point. The receiving component will be able to detect this situation and is free to request a more complete version of the data set from another source. The process of merging different data set versions into one is also handled by our version tracking and control mechanism.

V. Design

This section details the expected behavior and functionality of the components of our approach.

A. Data Format

To maintain data integrity in the presence of redactions, we store data using a hash tree structure. The exact layout of data items within the structure depends on the application and the desired granularity of the data items but the general concept can be described. The basic structure of the tree is shown in Figure 2. The output hash value of a node is calculated as follows: $H_N = Hash(H_{N_1}|H_{N_2}|...|H_{N_n}|Data_N)$, where H_N is the hash of a node N, H_{N_i} is the output hash of the i^{th} child of node N, $Data_N$ represents the data stored in node N, and $|$ represents data concatenation. If there is no data in a node, the output is calculated without the final element of the concatenation. Note that this differs from the classic Merkle hash tree structure [18], which stores data only at the leaves but not in internal nodes. Among other potential uses, our variation facilitates storage of metadata that covers subsets of data items by including metadata at the root of a subtree containing all of the appropriate data items.

When a data owner stores the data in such a hash tree structure, the owner signs the hash value at the root of the tree to enable the data to be verified later. Any subset of data in the hash tree can be redacted while still permitting the root signature to be verified. A redacted data item is simply replaced by its hash value in the tree. While this removes the data value, the remaining hashes up to the root hash can still be verified since the hash of the data is all that is used to compute hashes moving up the tree. Note that, in this way, data can only be removed (redacted) but cannot be modified. Modification of the data item would result in a different hash value, which when propagated up the tree, would result in an incorrect root hash that does not match the root signature. The final result is a signed redactable data structure, which we can use to transfer only allowed subsets of data to new systems/components, while still providing verifiability and integrity of the remaining data and associated policies.

To add new data into an existing hash tree, we create a hash tree for the new data and include the prior hash tree as

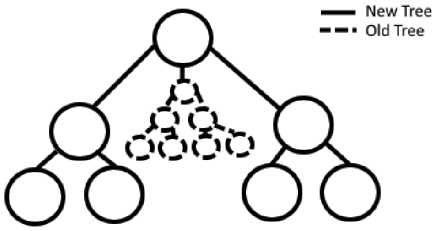

Fig. 3. Adding a New Sub-tree to a Hash Tree Structure

a child of the root of the newly created tree. This operation is depicted in Figure 3. The output hash of the new tree is signed by the author of the new data to provide overall data integrity. The signature of the prior hash tree is maintained to allow stand-alone verification of the data in that tree if a recipient is only concerned with that data and not interested in the newly added data. This way of combining new and old data results in non-binary tree structures, which also differs from the classical Merkle hash tree.

B. Management Service

The management service provides enforcement of both the hosting and usage policies through a set of distributed management components. Enforcement includes the identification and attestation of remote components to determine suitability for passing along hosting privileges on subsets of data which is described in Section V-E. Management components can take many forms and provide a variety of enforcement mechanisms. Our approach does not require that every management component provide every enforcement mechanism, but rather that components are able to reliably present their enforcement capabilities during attestation so that decisions may be made regarding access and hosting suitability. In the event that a requesting host is unable to provide the security features required to host an entire data set, the subset of data which the requester is qualified to host is transferred. This is handled by redaction prior to transmission. Redaction also takes place, based on the usage policy, when a user or application accesses data so that they are only presented with the data that they are authorized to access.

Sample enforcement mechanisms include secure key storage and management, legacy application sandboxing, IFC integration, and next hop attestation. Services like secure key storage may come in many flavors and use combinations of hardware and software approaches that would need to be presented when determining enforcement capabilities. Legacy application sandboxing may serve to open older unmanaged applications to data access by retrofitting them with required policy enforcement services. IFC integration may provide data owners with assurances that data derived from sensitive data also maintain the same level of sensitivity and policy expectations. Next hop attestation ensures that any recipient of sensitive information has been found to exhibit the expected attributes of the data owner before sharing the sensitive data. These mechanisms are neither exhaustive nor mandatory within the system, but they provide examples of the types of characteristics one might expect for sensitive data management.

Enforcement within a single system may either be consolidated or distributed. For example, Microsoft could produce an operating system that offers its services to ensure data privacy is maintained and enforces policy wishes on all data accesses on the device. Alternatively, or in addition to, a version of Open Office Calc may be developed that runs its own enforcement irrespective of the operating system. There are benefits to both approaches as centralized stand-alone enforcement can focus on hosting requirements while embedded application enforcement can bring application-specific knowledge to usage policy enforcement.

C. Policies

Any implementation that is widely used will need well-defined definitions of attributes to ensure each security request listed in a policy is equally represented across as many systems as possible. However, this work does not attempt to list or standardize attributes that maintain data privacy. This section looks to elaborate on some of the general aspects of both hosting and usage policies.

Existing works on sticky policies consider various methods for attaching policies to data in verifiable ways. However, by incorporating the policy into the hash tree structure of a data file, new methods for policy linkage are viable. A first method treats the policy as a separate but linked sub-tree similar to Figure 3. This allows for the policy to be verified and also remain redactable independently of the data for instances in which data need to be publicly released. Alternatively, the policy may be added as meta-data to the root of the sub-tree, which requires the policy to be released any time the data are released. Our approach allows inclusion of policies for newly added data that are linked into the tree but does not currently allow policies for existing data to be updated. Policy updates are a topic for future research.

To facilitate secure attribute definition and collection, policies may specify suitable certifiers of various attributes. For example a data owner may only trust operating systems certified by Microsoft to handle legacy application sandboxing, researchers employed by Georgia Institute of Technology, or TPM devices certified by Atmel. Additionally, there may be many different combinations of attributes trusted by a data owner that can be specified within the policy to denote expected levels of security. Each data set should have a policy specifying hosting attributes as well as a data usage policy.

Usage policies specify standard read/write privileges on data but also must include sensitivity label privileges such as adding and removing labels. One additional capability we have added in our approach is data viewing. With sensitive data and under certain workflows, it may be useful to give users the right to operate on the data without having access to view the data on the screen. Therefore, we propose a distinction between data access and data viewing capabilities, and we have implemented these as separate capabilities in our Excel add-in described in Section VI-B.

D. Certificates

Our approach emphasizes attribute-based policies as an efficient way of handling a variety of run-time scenarios for which exact details are not known at policy definition time.

However, we do believe that identity-based policies will still be meaningful in some cases. Standard identity certificates are widely deployed and we do not need to detail them here except to state that they serve as a linkage between a public key and the entity's identity, as verified by the certificate authority.

Attribute-based certificates maintain the ability to link a name with a public key but they also provide a way to dynamically disclose attributes in a verifiable, efficient manner. By treating attributes as data granules relating to the public key we may form a hash tree structure that contains the various characteristics attributed to a given entity. This structure allows control over the disclosure of attributes limiting the information that must be shared about oneself with an unknown remote system to establish trust [2].

E. Facilitating Trust

The work provided here primarily operates on the assumption that in some form or another, one device is able to detect characteristics of a remote system in a trusted manner. A common component that provides this functionality is a Trusted Platform Module (TPM), but other components such as Trusted Execution Environments (TEE) on mobile devices also provide similar functionality. These components are able to identify themselves and provide verifiable information regarding other components that are currently operating on the system.

An alternative approach to trusting the hardware and software components directly may come in the form of domain trust. Given that a software component may validate a domain through secure DNS services, it may follow that some data owners may willingly share data with those domains under the expectation that they have the security measures in place to support an expected level of data privacy and policy adherence. This might well be the case in the cloud service provider use case discussed earlier, wherein a user might trust the cloud provider to maintain the data securely and enforce her policies on data sharing without performing a full remote attestation. Our approach is not intended to restrict data transactions but rather to allow policy enforcement on data flows by identifying remote entities to some level and thereby ensure accepted data access.

The point to be emphasized is that our approach leaves the decision up to the data source as to what capabilities a remote system must have and how those capabilities are certified. This is done via the hosting policy for a data set. While the above examples are common scenarios we envision, many other possibilities exist and can all be accommodated within our policy-based approach.

VI. Prototype Implementation

In this section, we describe a proof of concept prototype, which demonstrates a use case in which collaboration is done via the sharing of data in spreadsheets.

A. Implementation Overview

For this prototype, we modified Microsoft Excel using the add-in functionality provided to support application level policy enforcement in the form of IFC. Remote trust is established using the TPM 2.0 simulator provided with the TSS.NET library for Windows. We emphasize again that TPM functionality is only one possible method for providing certifiable verification of a remote system. We employed it in our prototype because it was the most direct route to the required functionality.

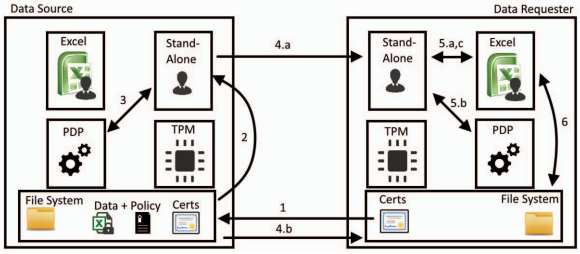

Fig. 4. Opening a file on a remote system

The majority of the development was done using C# with a Java-based XACML decision point deployed as a local Web service. When needed, libraries and classes were developed in C# and Java to facilitate communication. Hash trees were used to store data to allow for redactions to take place. To manage data storage on the system, a stand alone management component was implemented that uses the TPM simulator to store an application key. This key is used to encrypt a list of document keys. Figure 4 demonstrates the encryption key transfer process for our system.

The setup in Figure 4 shows an existing document-policy pair located on a source machine. The requesting system sends credentials and public keys for verification (Step 1). This process may happen via offline communication. These credentials are taken by the stand-alone management component (Step 2) and verified before they are passed to the decision point (Step 3). Once the attributes have been confirmed to satisfy the policy, the source's stand-alone component uses the certified public key to send the encryption key to the remote management component (Step 4.a). The encrypted file is also sent to the remote host (Step 4.b). Once the file and key are received by the new host, the Excel application requests access from the host's management component (Step 5.a). This triggers a local attestation and check with the policy decision point (5.b). Finally, the key is passed to the Excel application (Step 5.c) and the file is accessed (Step 6). Not shown in this figure, the Excel application then runs a policy check against the user to determine the subset of data that she is allowed to access.

B. Excel Add-In

Our prototype uses a Microsoft Excel add-in to demonstrate application level enforcement of policies. Once users have logged in, they may create data within the application, group cells into data granules, and add sensitivity labels for IFC using the interface shown in Figure 5. In the background, the application manages user identities when generating saved files, sensitive label propagation when referencing data in formulas, and policy enforcement based on user attributes.

The provided flow control add-in does not support the native save and load features provided within Excel. Instead,

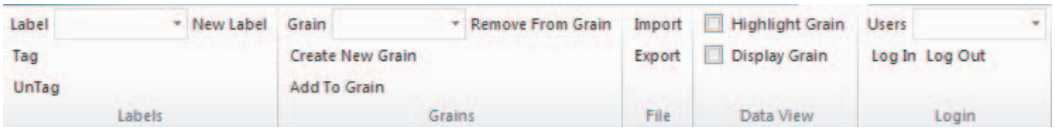

Fig. 5. Control ribbon for IFC functionality in Excel

import and export options were provided to allow for data restructuring and the inclusion of tags in the saved files. This functionality focuses on cell contents and labels rather than formatting and other various features Excel provides. Policy enforcement manages read, write, and view capabilities per user. These capabilities allow users to load data even if some data may not be presentable to them on the screen. Hidden cells are used to accomplish this.

One issue that arises from data redaction is that the placement of empty spaces might leak information. As such, it would be beneficial to have the ability to specify the importance or lack thereof of cell placement. This may include the ability to say row order is not important but data need to remain consistent across a row. Capabilities such as this are a topic for future research.

A gap in our current implementation is lack of IFC support for cut and paste operations. Excel provides the capability to disallow pasting and moving data, which prevents tagged data from being moved away from the tags. However, in the current implementation, we cannot prevent data from being copied to the clipboard and pasted elsewhere. This is a similar leak potential to writing data down or taking a screen-shot from information that appears on the screen, but a full featured solution would support tagged copy-paste operations or prevent movement of sensitive data to the clipboard.

VII. Evaluation

Our evaluation covers expected overheads of the use case depicted in Figure 4. All experiments in this section were conducted on an Intel Core i7 CPU with a 3.5 GHz clock frequency and 32 GB of RAM.

A. Evaluation of Policy-Based Data Transfer

The first step in this process is gathering certificates and verifying attributes. If we are relying on TPM functionality for attribute attestation, this requires transfer of public identity keys and certificates, PCR event logs, PCR event certificates, binding key public information, TPM provided certificate of the binding key, and the binding key policy. Using this information, we are able to reconstruct the PCR values to link attributes to the system as well as the binding key. The process varies primarily on the number of steps required to reconstruct PCR values. TPMs provide the capability to combine hashes from multiple software components in each PCR using a hash chain. Each software component included in a PCR value requires one step in PCR reconstruction.

To demonstrate a basic range of overheads we ran tests of reconstruction and validation for fifteen registers across a range of one to ten steps per register. At the upper end, this corresponds to verification of 10 software components in each PCR for a total of 150 verified software components on the system being attested, which is clearly an extreme case. Figure 6 shows an estimate of the overheads seen in the verification process, which are quite low, in the range of 50–250 milliseconds of wall clock time. Since the wall clock time represents an upper bound on the computation time, actual overheads are expected to be even lower than this in practice. This demonstrates the expected overheads for the processing between steps 2 and 3, and 4 and 5 in Figure 4.

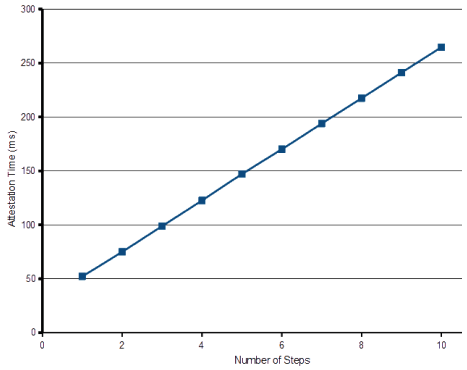

Fig. 6. Attestation Overheads

After the certificates have been validated, the attributes and request must be passed to the policy decision engine. Using the attached policy and attributes found in the verified certificates, the engine is able to decide the suitability of the remote host for hosting the requested data. To better represent the overhead of a deployed system, we evaluated the decision engine as a stand alone process rather than as a Web service. The Web service was used to ease prototype development but is not the expected deployment method in practice.

The process for this test involved requesting three forms of access for a given user for each resource. The reason for this is that, for each resource, a user may have viewable access, which means they may view it on the screen; they may have write access, which allows them to edit and write new data with a given tag; and they may also have read privileges which allow them to use the data in computational processes so long as the data is not shared back with them. The tests, detailed in Figure 7, measured the complete time for decisions on all resources for a user with no access rights to prevent early decisions on any particular case. In the usage policy, each resource access mode was allowed for ten different specific users.

Figure 7 shows a time of just over twenty seconds to make full decisions on 1000 resources (labels). This includes decisions on the computer readable, screen readable, and writable permissions on each label. It should be noted that

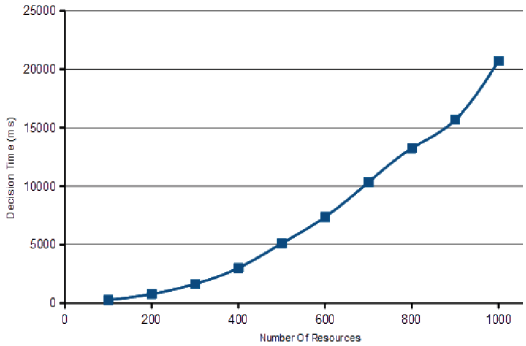

Fig. 7. Full Policy Decision Time for a File vs. the Number of Labels in the File

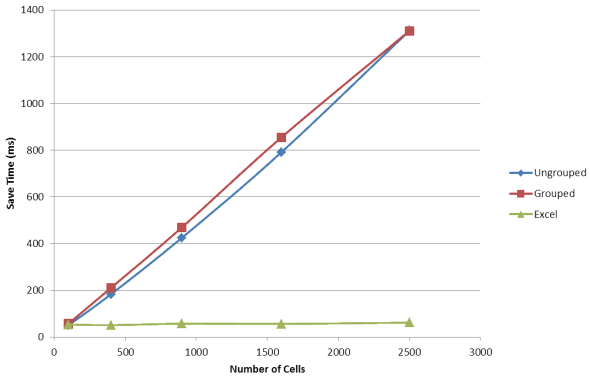

Fig. 8. Save Overheads using Excel Add-in

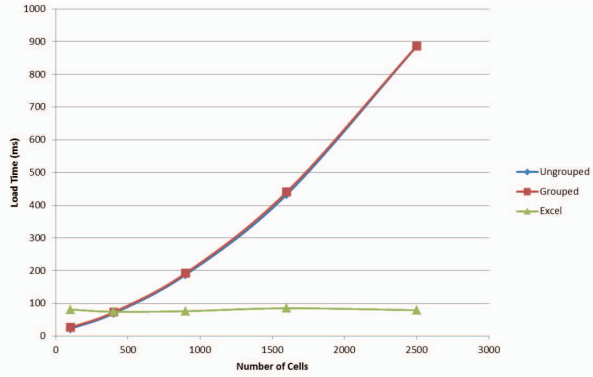

Fig. 9. Load Overheads using Excel Add-in

this data point represents a very extreme case. In this situation, each label is assumed to have its own rules in the policy and the three access modes mean that there are 3000 rules in the policy. Also, as mentioned earlier, there is no early decision meaning that all 3000 rules are evaluated. Policies for most of the use cases we envision should have at most a few hundred rules. The data point for 100 resources in Figure 7 represents a rule size of 300 and, in this more representative case, the policy evaluation time was only 261 milliseconds, which is quite reasonable. It should also be noted that this evaluation time is only incurred once, at file opening time, to correctly issue access permissions on data and labels. These overheads are applicable to the steps 3 and 5.b in Figure 4.

B. Evaluation of Application-Level Overheads

Finally we analyze the overheads seen in our Excel add-in in the process of save and load functionality seen in step 6 in Figure 4. To support data redaction and sensitivity labels, our add-in required proprietary save and load functions. Our implementation has an internal representation of the label mapping that attempts to form large rectangles of similarly labeled cells. To account for this, we compare save and load functionality across a spreadsheet with no same-label neighbors (ungrouped) and one with all cells of the same label (grouped). The data in Figure 8 show that similar overheads are seen between these two cases. This implies that the majority of the overhead comes from the common process of extracting the cell information from Excel, storing the information within the hash tree for saving, and generating the final XML structure. We currently see times of approximately 1 second to perform a save with 2000 cells in a workbook.

The overhead for loads, while also being significant, is lower than the save overhead (less than 1 second for up to 2500 cells). The overhead is again mostly due to the XML processing, extraction of data from the stored hash tree, and repopulation of the data within Excel and our background tag map. These results may be seen in Figure 9.

For a more complete picture of the overhead of our implementation, a few items should be mentioned. First, our experiments did not include file encryption and decryption time. However, if the standard method of encrypting the file using symmetric key encryption and encrypting the symmetric key using public key encryption is performed, the encryption/decryption times for the file sizes in our experiments and an average current CPU would be less than one millisecond and would, therefore, not impact the results. Second, we did not attempt to address CPU overheads of run-time operations within the modified Excel such as creating labels, tagging data, etc. During extensive testing of the modified Excel, these overheads were imperceptible to us.

While the reported save and load overheads are admittedly high, our Excel implementation was designed primarily for functionality and not optimized for performance. We expect the overheads would be substantially reduced if the IFC functionality was integrated directly into the worksheet application instead of being implemented as an add-in. Furthermore, we used basic XML library functions to parse the XML input on a load and to generate the XML output on a save. Since XML processing is notoriously slow, we are currently investigating optimizations, such as serialization, that could serve to reduce these overheads to reasonable levels.

VIII. Conclusion

We have presented a distributed enforcement approach for sticky policies that permits data to be disseminated safely across heterogeneous software and hardware components that provide differing levels of security and without pre-existing trust relationships. In addition to supplemental changes to

the current implementation to address known limitations, our future work will explore potential solutions to data subset distribution. Our aim in this exploration will be to address possible solutions for data re-integration when a transaction occurs from a lower security component to one with a greater level of protection. Methods for managing verifiable updates to policies on existing data will also be considered in future work.

IX. Acknowledgments

This work was supported in part by the National Science Foundation under Grant IIP-1230740.

References

[1] C. Ardagna and et al. A privacy-aware access control system. *Journal of Computer Security*, 16(4):369–397, 2008.

[2] D. Bauer, D.M. Blough, and D. Cash. Minimal information disclosure with efficiently verifiable credentials. In *Proceedings of the 4th Workshop on Digital Identity Management*, pages 15–24, 2008.

[3] D. Bauer, D.M. Blough, and A. Mohan. Redactable signatures on data with dependencies and their application to personal health records. In *Proc. Workshop on Privacy in Electronic Society*, pages 91–100, 2009.

[4] M. Beiter and et al. End-to-end policy based encryption techniques for multi-party data management. *Computer Standards & Interfaces*, 36(4):689–703, 2014.

[5] I. Bente and et al. Towards permission-based attestation for the Android platform. In *Trust and Trustworthy Computing*, pages 108–115. Springer, 2011.

[6] S. Betgé-Brezetz and et al. End-to-end privacy policy enforcement in cloud infrastructure. In *Proc. 2nd Int'l Conf. on Cloud Networking*, pages 25–32, 2013.

[7] E. Birrell and F. Schneider. Fine-grained user privacy from avenance tags. Technical Report http://hdl.handle.net/1813/36285, Cornell University Department of Computer Science, 2014.

[8] J. Brown and D.M. Blough. Verifiable and redactable medical documents. In *Proc. AMIA Annual Symposium*, pages 1148–1157, 2012.

[9] S. Cabuk and et al. Towards automated provisioning of secure virtualized networks. In *Proc. 14th Conf. on Computer and Communications Security*, pages 235–245, 2007.

[10] L. Catuogno and et al. Trusted virtual domains: Design, implementation, and lessons learned. In *Proc. Int'l Conf. on Trusted Systems*, pages 156–179, 2009.

[11] T. Garfinkel and et al. Terra: A virtual machine-based platform for trusted computing. *SIGOPS Operating Systems Review*, 37(5):193–206, 2003.

[12] Y. Gasmi and et al. Flexible and secure enterprise rights management based on trusted virtual domains. In *Proc. 3rd Workshop on Scalable Trusted Computing*, pages 71–80, 2008.

[13] M. Ghorbel and et al. Privacy data envelope: Concept and implementation. In *Proc. 9th Int'l Conference on Privacy, Security and Trust*, pages 55–62, 2011.

[14] H. Härtig and et al. The Nizza secure-system architecture. In *Proc. Int'l Conference on Collaborative Computing: Networking, Applications and Worksharing*, 2005.

[15] F. Kelbert and A. Pretschner. Data usage control enforcement in distributed systems. In *Proc. of the Conference on Data and Application Security and Privacy*, pages 71–82, 2013.

[16] K. Kostiainen, N. Asokan, and J.-E. Ekberg. Practical property-based attestation on mobile devices. In *Trust and Trustworthy Computing*, pages 78–92. Springer, 2011.

[17] J. McCune and et al. Flicker: An execution infrastructure for TCB minimization. *SIGOPS Operating Sys. Review*, 42(4):315–328, 2008.

[18] R. Merkle. A certified digital signature. In *Proceedings of Advances in Cryptology*, pages 218–238, 1989.

[19] M. Migliavacca and et al. Distributed middleware enforcement of event flow security policy. In *Proc. 11th Int'l Conference on Middleware*, pages 334–354, 2010.

[20] I. Papagiannis and P. Pietzuch. Cloudfilter: Practical control of sensitive data propagation to the cloud. In *Proc. of the Workshop on Cloud Computing Security*, pages 97–102, 2012.

[21] S. Pearson and M. Mont. Sticky policies: an approach for managing privacy across multiple parties. *IEEE Computer*, 44(9):60–68, 2011.

[22] A. Ruan and A. Martin. Repcloud: Achieving fine-grained cloud TCB attestation with reputation systems. In *Proc. 6th Workshop On Scalable Trusted Computing*, pages 3–14, 2011.

[23] A.-R. Sadeghi and C. Stüble. Property-based attestation for computing platforms: Caring about properties, not mechanisms. In *Proc. of the Workshop on New Security Paradigms*, pages 67–77, 2004.

[24] N. Santos and et al. Policy-sealed data: A new abstraction for building trusted cloud services. In *Proc. USENIX Security Symposium*, pages 175–188, 2012.

[25] X. Wang and et al. Protecting outsourced data privacy with lifelong policy carrying. In *Proc. Int'l Conferences on High Performance Comp. and Comm. & Embedded and Ubiquitous Comp.*, pages 896–905, 2013.

[26] S. Xin, Y. Zhao, and Y. Li. Property-based remote attestation oriented to cloud computing. In *Proc. 7th Int'l Conference on Computational Intelligence and Security*, pages 1028–1032, 2011.

[27] F. Zhang and et al. Cloudvisor: Retrofitting protection of virtual machines in multi-tenant cloud with nested virtualization. In *Proceedings of SOSP*, pages 203–216, 2011.