Authentication in the Clouds: A Framework and its Application to Mobile Users

Richard Chow
PARC
rchow@parc.com

Markus Jakobsson
FatSkunk
markus@fatskunk.com

Ryusuke Masuoka
Fujitsu Labs of America
ryusuke.masuoka@us.fujitsu.com

Jesus Molina
Fujitsu Labs of America
jesus.molina@us.fujitsu.com

Yuan Niu
UC Davis
yuan.niu@gmail.com

Elaine Shi
PARC
eshi@parc.com

Zhexuan Song
Fujitsu Labs of America
zhexuan.song@us.fujitsu.com

ABSTRACT

Cloud computing is a natural fit for mobile security. Typical handsets have input constraints and practical computational and power limitations, which must be respected by mobile security technologies in order to be effective. We describe how cloud computing can address these issues. Our approach is based on a flexible framework for supporting authentication decisions we call *TrustCube* (to manage the authentication infrastructure) and on a behavioral authentication approach referred to as *implicit authentication* (to translate user behavior into authentication scores). The combination results in a new authentication paradigm for users of mobile technologies, one where an appropriate balance between usability and trust can be managed through flexible policies and dynamic tuning.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection - Authentication

General Terms

Algorithms, Security

Keywords

cloud computing, authentication, mobile computing

1. INTRODUCTION

As online access to services becomes ubiquitous and the cloud access model gains momentum, authentication is increasingly becoming a focal point for security professionals. With bank accounts, health records, corporate intellectual property and politically sensitive information being just a few clicks away – no matter where in the world you are – it is natural to worry about the identity of those wishing to gain access. At the same time, as penetration rates of cell phones approach (and oftentimes surpass) 100% in industrialized nations [5], more and more of these accesses are made from handsets. This introduces security vulnerabilities and complicates matters, given that handsets abide by entirely different usability, computational, and power limitations than traditional computers. For instance, it is often not practical to require security-related computation that notably limits battery life. Antivirus is the usual example of a computationally intensive security-related service [16], but we argue that usable and strong authentication on mobile devices have needs that are equally computationally intensive. Hence, one is led naturally to cloud computing, which is eminently suitable for addressing problems related to limited client resources, as it offloads computation from clients and offers dynamic provisioning of compute resources.

We describe an independent, policy-based cloud authentication platform using open standards, and supporting the integration of various authentication methods. We call this platform *TrustCube* [18]. TrustCube addresses the authentication problem in a simple, yet flexible manner – whether the client device is a mobile phone or not. Given the high elasticity of requests, the complex nature of authentication, and the need to consider varied sources of authentication data (e.g., device integrity reports, user credentials, etc.), there are distinct benefits associated with using policy-based authentication. Namely, doing so permits the use of fine-grained and user-specific security policies that can be instantly updated to address changing needs. Furthermore, TrustCube assumes a federated authentication framework, such as OpenID [6]. This implies a star-shaped authentication topology, for example, one where the authentication service is itself in the cloud and supports a set of other cloud services. Authentication services are particularly well-suited for a star-shaped topology, as it is a specialized service that is difficult to secure. A star-shaped topology also has privacy benefits, as only the center of the star needs to collect user-specific data. There are potentially heavy loads on the authentication service in a star-shaped topology due to its

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CCSW'10, October 8, 2010, Chicago, Illinois, USA.
Copyright 2010 ACM 978-1-4503-0089-6/10/10 ...$10.00.

central role in the process, but this can be well addressed through the elasticity of cloud computing, as multiple instances of the authentication service can be deployed within the cloud as needed.

Besides computational power and battery consumption, there are other important differences between the traditional computing framework and its mobile counterpart that affect bottom-line security. In particular, mobile devices are constrained in terms of text input and are more prone to theft than traditional computers. The input constraints make it difficult for users to input complex passwords (e.g., see [13]), often leading to the use of password managers, short passwords, and simple PINs. The higher risk for theft, in turn, makes these practices increasingly dangerous, as they threaten to put "open" devices in the hands of criminals.

We do not believe public awareness of the problem can turn matters around, but believe in the need for a new approach to authentication of users. Extending the traditional authentication paradigm beyond *"what you have – what you know – what you are"*, we suggest that *"what you do"* is a practical way to control access. We refer to this as *implicit authentication* [13, 12]. Implicit authentication allows us to identify users by their *habits*, as opposed to their belongings, memorized data, and biometrics. Products exist in this general area, e.g., [7, 3], but they have generally not been focused on mobile users and are based on data such as IP addresses and device profiles. We have built an implicit authentication system based on mobile data: calling patterns, SMS activity, website accesses, and location. The paradigm of authentication based on implicitly observed behavior helps us – among other things – to protect against unwanted access from stolen handsets.

We note that much of the data that is needed to make such mobile authentication decisions – such as calling patterns and crude location information – is automatically available to carriers. Other useful contextual data, such as calendar information, is hosted by a small set of cloud service providers. This also makes implicit authentication natural to use in the context of cloud-based authentication.

In this paper, we do not describe the implicit authentication algorithms and models explicitly, which is described in [17]. Instead, we concentrate here on a description of the top-level system and its application to the cloud; the intention is that any implicit authentication system can be plugged into our framework. However, in the Case Study of Section 4, we present some data from the particular implicit authentication system we have implemented, in order to give an indication of the effectiveness of the approach. In actual deployment, TrustCube would likely use other sorts of authentication modules in addition to implicit authentication, increasing the overall effectiveness of the system.

The use of implicit authentication implies a policy-based authentication framework. Implicit authentication is a statistical test whose output is not binary, and the thresholds and amount of uncertainty allowed depend on the particular application. Also, because false positives cannot be eliminated, implicit authentication works best with a well-defined fallback authentication technique. The use of implicit authentication also implies a star-shaped authentication topology, as there is a need to limit the scattering of the data, which is potentially very privacy-invasive. Thus, the use of implicit authentication naturally implies an authentication framework such as TrustCube, which is policy-based and designed for star-shaped authentication technologies.

Hence, we argue that the combination of TrustCube and implicit authentication is a natural solution to better control fraud risks while minimizing user frustration and handset computation and power consumption. In this paper, we describe how this can be done and report on a proof-of-concept implementation.

Outline. We begin by motivating the use of TrustCube and implicit authentication, supported by four different use cases and the adversarial behavior we consider (Section 2). We then describe the high-level architecture of our solution (Section 3). This is followed by an overview of a concrete implementation of our solution and the results of an experiment we used to assess its security (Section 4).

2. USE CASES AND ADVERSARIAL MODEL

There exist many situations in which our proposed authentication approach potentially improves security and usability. We will briefly describe four scenarios that highlight various use cases.

In the *first* scenario, we consider a consumer who performs a credit card transaction at a point-of-sale (PoS). The authorization request is redirected to the credit card clearing organization which inquires about the recent behavior of the consumer, as observed by her phone. If the device appears to be in use (i.e., the accelerometer sensor is activated), and its recent location traces are inconsistent with the PoS location, then the transaction may be rejected or require further corroboration, depending on policy. On the other hand, if everything is consistent with the legitimate user making the purchase, the clearing house may signal to the PoS that no signature is needed – resulting in a fast-track checkout. (Note that the consumer did not interact with her phone as part of the transaction, but the phone's recent history was used.)

In the *second* scenario, a bank customer uses his phone to check his bank account, using his regular password. If the recent history of the device is inconsistent with the user's habits, then this flags the login as a potentially high-risk event. The bank can use this information in its authentication policy to determine how to process the request, potentially involving another form of authentication. Thus, implicit authentication is not used as a replacement of regular authentication in this setting, but rather, as a *second factor*.

In the *third* scenario, a handheld medical device is protected by observing its use in the context of location, schedules, and more. This allows automated authentication of medical staff, protecting access to privacy-sensitive medical data in a context where stressful working conditions would otherwise make account sharing tempting.

In the *fourth* scenario, an enterprise employee uses his phone to access an internal corporate portal. Depending on the sensitivity of the portal, the authentication policy may take various combinations of data, such as passwords, hardware tokens like SecurID [8], regular (HTTP) cookies, cache cookies [14], device configuration data [9], and contextual data such as previous accesses by the same employee. The policy may also require evaluation of the system integrity of the phone, requiring data ranging from the versions of software that are installed to a Trusted Computing-style attestation from a root of trust [10]. Implicit authentication may

not be applicable or available. Here, we consider authentication to a service provider that uses TrustCube to guide access decisions. Any of this data could potentially be forged by an attacker. At the same time, any piece of data may be incorrect or missing from a *legitimate* transaction. The use of a flexible authentication engine with risk-informed weights allows a balance to be struck between security and accessibility. It also benefits from knowledge of past authentication attempts; therefore, if the authentication server is used by multiple service providers, each one of these benefit from the improved precision offered by one unified authentication server.

In all of these settings, the infrastructure for deriving an authentication score for a given user may be independent of the context where the authentication decision is needed. For instance, the user and his service providers might use a cloud authentication service, supported by TrustCube or a similar authentication framework, where the authentication service would poll data sources at frequent intervals (or even other authentication services). It would use the obtained information to build and update user models, and to make policy-driven authentication decisions. The underlying policies can be chosen differently for different use cases, for different service providers, and for different users, allowing for a fine-grained tuning of the policies to varying authentication needs.

Adversary and Our Goals. In the context of these use cases, we consider an adversary who wishes to obtain access to a resource based on *theft* – of a client device, of an auxiliary item (e.g., a credit card), or of user credentials (e.g., a password). In detail, we consider these cases:

- Theft of client device. The adversary gains physical access to the client device – e.g., handset – and wishes to use this in a way that benefits him. He may wish to place phone calls, perform payments from accounts associated with the device, access personal data, or simply become the new user of the client device.
- Theft of auxiliary item. The adversary attempts to use a resource associated with a given client device, without necessarily having control of this device. As examples, the adversary wish to perform a credit card transaction using a stolen credit card or gain access to a building using a employee identity card.
- Theft of user credentials. The adversary has stolen user credentials corresponding to a given client device – e.g., using phishing, malware or shoulder surfing – and wishes to gain access to a resource associated with these credentials and with the client device.

In all of these theft-based scenarios, the goal of the authentication infrastructure is to detect the inconsistency that arises from the adversary's access requests, and in the context of the information that TrustCube obtains from the data aggregators and the client device in question.

3. HIGH-LEVEL ARCHITECTURE

3.1 Authentication Flows

We consider an architecture with the following types of participants: *client devices*, *data aggregators*, an *authentication engine*, and *authentication consumers*. The client devices generate observable context and actions as part of their regular use. The data aggregators collect data on context and actions from client devices, and from auxiliary sources (such as schedules provided by third parties). The authentication engine obtains data from data aggregators, and may request data directly from client devices. It makes authentication decisions based on collected data and authentication policies. Authentication consumers provide policies to the authentication engine based on end user access requests (e.g., a webpage access request or a payment request). Finally, the authentication consumer responds to a client's request based on the authentication result it receives. Figure 1 demonstrates the relationship between participants.

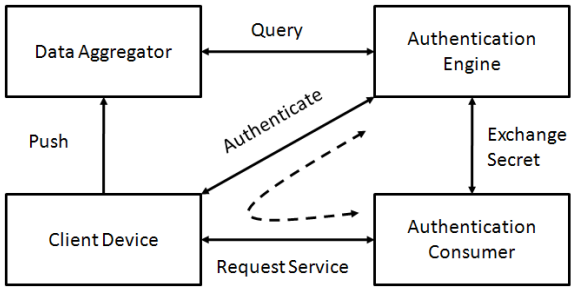

Figure 1: Participants in the general architecture and the relationships among them. The Client Device pushes context and activity data to a Data Aggregator; the Authentication Engine queries the Data Aggregator for individual device reports; the Client Device requests a service from the Authentication Consumer; the Client Device authenticates itself through the Authentication Engine; and the Authentication Engine exchanges a secret with Authentication Consumer during authentication (in order to later verify authentication results). Arrows indicate the direction of data flow, and a dashed line indicates an indirect connection (e.g., using browser redirection).

The authentication flow is as follows: Before authentication starts, the authentication consumer lists the access requests (e.g., a webpage access request or a payment request) that require authentication. For each request, the authentication consumer will register a policy with the authentication engine. The policy includes at least three parts: the access request, the information to be collected from client devices or data aggregator for this access request, and a rule to generate the authentication result. During normal operation, client devices periodically report to the data aggregator. This data will be used to track user behavior and support authentication requests.

The authentication flow starts when an access request is received by the authentication consumer. (This request may have been initiated by a client device that the system collects data from, or by another device, such as a credit card reader.) Upon receiving the request, the authentication consumer redirects the request to the authentication engine, along with request details. The authentication engine retrieves the policy for the access request, extracts the information that needs to be collected, and sends an inquiry to the client device and/or data aggregator. The client device and/or data aggregator receives the inquiry, generates a report, and sends it back to the authentication engine.

The authentication engine then applies the authentication rule in the policy and determines the authentication result (whether or not the client device is authenticated for the access request) and sends this back to the authentication consumer. Based on the authentication result, the authentication consumer will either provide the service (e.g., return the webpage content or accept the payment request) or reject the request.

3.2 Data Analysis and Processing

Both *pull* and *push* methods are adopted by client devices to provide data. The push path is from client devices to data aggregators. The main purpose is to constantly report the context and behavior of client devices. The push operation allows the client device to clear storage space by clearing its local cache after reporting. The pull path is a request from the authentication engine to client devices and data aggregators to send data back to the authentication engine.

Data aggregators might mine the data received to derive a data and behavioral model for the client device. For example, in the case of implicit authentication, the authentication result may depend on whether the current user is following her regular patterns.

In general, we envision the following framework for the machine learning algorithm used by a data aggregator, see Figure 2. From a user's past behavior, we first learn a *user model* which characterizes an individual's behavioral patterns. Given a user model and some recently observed behavior, we can compute the probability that the device is in the hands of the legitimate user. We use this probability as an *authentication score*. The score is used to make an authentication decision: typically, we can use a threshold to decide whether to accept or reject the user, and the threshold can vary for different applications, depending on how security sensitive the application is.

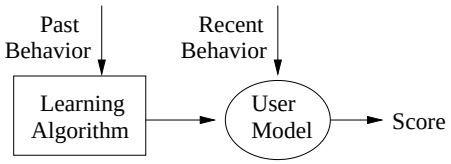

Figure 2: Learning a User Model. The learning algorithm uses past behavior to create a user model. The user model uses recent behavior to generate a score.

4. IMPLEMENTATION APPROACH

We implemented the authentication framework described above. We call it TrustCube or Trust^3 because, unlike traditional user-based authentication, TrustCube supports a wide range of policies, which may include reports on the user, the platform, and the environment of a client device (3 factors). The general architecture is given in Figure 3.

We developed our client side agent on Android [2], an open-source mobile operating system. One of the advantages of Android is the ability to run a background monitoring service, critical for the sort of data collection performed by implicit authentication. The client side agent collects two kinds of data.

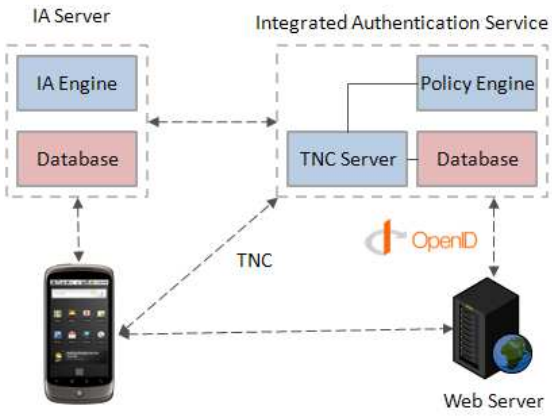

Figure 3: The TrustCube architecture, one implementation of the authentication framework of Figure 1. The authentication between the smart phone and integrated authentication server uses the TNC (Trusted Network Connect) protocol. Service requests are redirected to the integrated authentication service using the OpenID protocol.

First, it collects a user's context and activities, and reports these to the data aggregator regularly (in our system, the data aggregator is the implicit authentication server, or IA server). In some deployments, this type of data would already be collected by the carrier, allowing us to avoid explicitly exporting it from client devices. The collected data consists of phone call and SMS history, browser history, network information, and location. The data is stored locally until it is successfully reported to the data aggregator. To protect privacy, all data collected (except GPS location) are hashed with a random key at the time of collection. This key is device-specific, generated and stored on the device, and never exported. The system cannot infer the actual data from the hashed data, nor can it test a piece of data to see if it agrees with the original data. In this way, the system balances security and privacy: unusual patterns can still be detected while the user keeps private the phone numbers called and the web sites visited.

Second, during authentication, it collects information about the phone and reports this data to the authentication engine (in our system, the authentication engine is the integrated authentication server). The information it collects is based on the policy provided by the authentication engine, and may include, for example, the applications that are running and installed and the firmware version.

The IA server exposes two web service interfaces: report and query. The report interface allows client side agents to report context and activity information routinely; the query interface allows other entities (e.g., the authentication engine) to get a score for a device which indicates how normal the behavior of the device is at the moment. The score depends on current data and a machine-learned model of the device's past behavior, based on a sliding window of data collected in the past. The window does not extend too far in the past, so that recent behavior is more important than behavior in the far past.

The integrated authentication service is the authentication engine. The service is developed in Java and deployed

as an Amazon EC2 [1] instance and encapsulated as an AMI (Amazon Machine Image). The service exposes two interfaces: a web-based user interface for authentication consumers to define and maintain policies and a web service interface to authenticate client devices. The authentication service scales effortlessly because it itself is a cloud service.

Policies can be easily uploaded, modified, and monitored using the integrated authentication service's user interface. Policies can also be based on the platform, enabling mobile and desktop clients to share the same authentication engine with different policies. The policy supports rules of the following form: (1) integrity check rules constructed using a set of conditions describing the platform and environment and OR and AND operators; (2) thresholds for the implicit authentication score; and (3) specification of an alternate the user authentication method, picked from a set of available authenticators such as passwords, PIN pads and biometrics, in case the IA score is below the threshold. The integrity check rule includes items such as the minimum OS version, acceptable network settings, and a white/black list of installed/running applications. For example, a policy might look like: (1) the device should run Android 2.1 update 1 or above AND the WiFi SSID should be "hospital" AND only default and hospital applications can be installed; (2) the minumum IA score to view medical data is 800; (3) if the IA score is below the minimum, the user must use a PIN pad to further authenticate.

Finally, the authentication consumer is a web server. We deployed a sample web server that hosts some sensitive medical data. In order to access the site, a user must authenticate using the integrated authentication service. To model a web-based service hosted in the cloud, we developed the web server in Google's AppEngine [4].

The protocol to redirect an access request between a web server and the integrated authentication service is OpenID, an international standard for federated authentication allowing one ID provider to serve multiple ID consumers. We observe that this protocol supports the separation of the authentication component from a web-based service and it fits our general architecture well. The OpenID protocol also includes methods for a web server and an integrated authentication service to exchange a secret before authentication using the Diffie-Hellman key exchange method. Later, the web server may use the shared secret to verify authentication results. Other standards for federated identity, such as SAML SSO, can be easily supported in the future.

Since information is transferred through open networks in TrustCube, we adopted the TNC (Trusted Network Connect) [11] protocol for trusted reporting. TNC, a TCG (Trusted Computing Group) protocol, addresses and attempts to provide network access control that meets security requirements through non-proprietary standards. The TNC client is implemented in the client side agent and the integrated authentication service acts as a TNC server.

Case Study: Device Theft

We now examine the performance of our proposed system. First, we note that success rates depend very much on the type of adversarial behavior considered. For example, one may expect the detection rates to be much greater in a context where we wish to detect the theft of *credit cards* than one where we aim to detect the theft of a *handset*. This is so since credit cards – and credit card numbers – are typically stolen separately of the associated mobile devices. Therefore, there will be a very likely discrepancy between the location of the credit card (as observed by the point of sale or IP address) and the location of the handset. Location discrepancy is one of the indicators of credit card theft, along with indications that the registered handset is used normally – i.e., that it is not forgotten at home. In contrast, if we want to identify device theft, we need to rely on the observed behavior alone. We do not have the benefit of any simple rules, as we do for the detection of credit card theft.

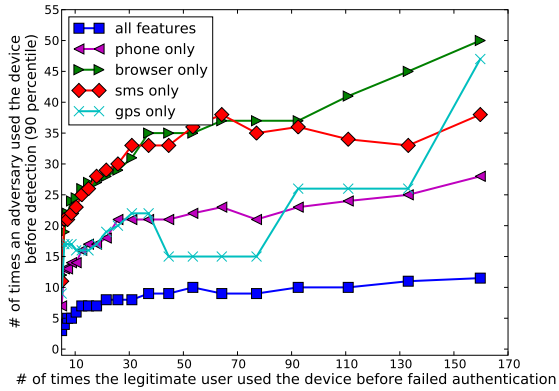

Figure 4: False positive and false negative rates for different features separately and combined. The x-axis shows the number of successful implicit authentications before a failed authentication for a legitimate user, while the y-axis shows the number of times an adversary can access resources before being locked out with 90% probability. For example, to interpret one point on the graph, a legitimate user can use the device around 150 times before a failed authentication, while an adversary will be locked out after using the device 10 or fewer times with 90% probability.

In the following, we take a brief look at our expected success rates in identifying the theft of handsets. We emphasize that the ability to detect this sort of theft has to be balanced against the risk of failing legitimate users – the usual balancing act between false positives and false negatives. Here, a false negative means that a legitimate user will fail the implicit authentication, and therefore, will have to authenticate in a traditional manner. We do not aim for perfect security, as that would mean a return to user inconvenience. Instead, we aim to lock out attackers as soon as possible, and keep them locked out.

We consider the results of an experiment we performed [15], in which we recorded the activities of a collection of users over a two-week period and trained our detector on this data. We then created adversarial attempts, and attempted to detect these. To model an adversarial attempt, we pasted in the activity trace of one user with the activity trace of another user, with care taken to avoid peculiarities such as sudden jumps in terms of location. By varying our acceptance thresholds, we obtain tradeoffs between the number of times between failed *legitimate* authentication attempts and failed *adversarial* authentication attempts. These are shown in Figure 4. We compare these rates to those obtained from

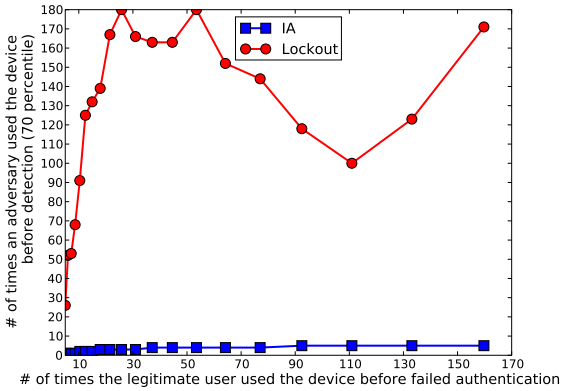

Figure 5: Tradeoffs between false positives and false negatives for a simple policy that locks a device after a set period of inactivity. The x-axis shows the number of successful implicit authentications before a failed authentication for a legitimate user, while the y-axis shows the number of times an adversary can access resources before being locked out with 70% probability.

a simple timed lock-out policy in Figure 5. We note that these are preliminary results obtained from an experiment with approximately 50 users, observed over a period of two weeks. We expect that the error rates will be better for large-scale deployments that collect behavioral data for extended periods of time.

5. CONCLUSION

Cloud computing has brought new challenges and opportunities for authentication. There is increasing demand for usable authentication to access services and data for both enterprises and consumers. At the same time, the cloud provides abilities such as centralized analysis and monitoring, and potential for new and more accurate authentication techniques.

Simultaneously, there is another trend that is important to understand in the context of cloud computing and authentication: the shift in platforms from traditional PCs toward smart phones and other mobile platforms. Usage patterns of mobile platforms are very different from PCs, and are associated with richer behavioral data, for example location and call logs. On the other hand, user authentication can be more intrusive while on the go, making privacy more of a concern. When mobile users access the cloud, their behavioral data is starting to be used for applications such as advertising, using the cloud's data-aggregation ability. One of the themes of the paper is that this data can similarly be used for new kinds of authentication. We refer in particular to implicit authentication, which uses a user's past behavioral data to authenticate, and which is particularly well-suited for mobile devices.

We have described a concrete cloud authentication system that we have built based on these ideas. This system has at its core an authentication service – dubbed TrustCube – that *itself* resides in the cloud. Any cloud-based service can re-direct authentication to the authentication service via a federated authentication framework such as OpenID. Our system has the ability to accept various authentication methods in a policy-driven manner, from TCG-style device integrity measurements to passwords. The system is flexible enough to support newer, cloud-oriented authentication techniques. In particular, we have integrated the system with implicit authentication, and we have described several simple end-to-end use cases with our authentication framework and implicit authentication.

6. REFERENCES

[1] Amazon Elastic Compute Cloud (Amazon EC2). On the Web at `http://aws.amazon.com/ec2/`.

[2] Android. On the Web at `http://www.android.com/`.

[3] Entrust IdentityGuard. On the Web at `http://www.entrust.com/strong-authentication/identityguard/index.htm`.

[4] Google App Engine. On the Web at `http://code.google.com/appengine`.

[5] List of countries by number of mobile phones in use. On the Web at `http://en.wikipedia.org/wiki/List_of_countries_by_number_of_mobile_phones_in_use`.

[6] OpenID. On the Web at `http://openid.net`.

[7] RSA Adaptive Authentication. On the Web at `http://www.rsa.com/node.aspx?id=3018`.

[8] SecurID. On the Web at `http://en.wikipedia.org/wiki/SecurID`.

[9] The 41st Parameter. On the Web at `http://www.the41st.com/`.

[10] Trusted Computing Group. On the Web at `http://www.trustedcomputinggroup.org/`.

[11] Trusted Network Connect. On the Web at `http://www.trustedcomputinggroup.org/developers/trusted_network_connect/`.

[12] R. Greenstadt and J. Beal. Cognitive security for personal devices. In *The First ACM Workshop on AISec*, 2008.

[13] M. Jakobsson, E. Shi, P. Golle, and R. Chow. Implicit Authentication for Mobile Devices. In *HotSec '09: Proceedings of the 4th USENIX Workshop on Hot Topics in Security*, 2009.

[14] A. Juels, M. Jakobsson, and T. N. Jagatic. Cache cookies for browser authentication. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, 2006.

[15] Y. Niu, E. Shi, R. Chow, P. Golle, and M. Jakobsson. One experience collecting sensitive mobile data. In *USER Workshop of SOUPS*, 2010.

[16] J. Oberheide, E. Cooke, and F. Jahanian. CloudAV: N-Version Antivirus in the Network Cloud. In *Proceedings of the 17th USENIX Security Symposium (Security)*, 2008.

[17] E. Shi, Y. Niu, M. Jakobsson, and R. Chow. Implicit authentication through learning user behavior. In *Information Security Conference (ISC)*, 2010.

[18] Z. Song, J. Molina, S. Lee, H. Lee, S. Kotani, and R. Masuoka. Trustcube: An infrastructure that builds trust in client. In *Future of Trust in Computing, Proceedings of the First International Conference*, 2009.