Physically Restricted Authentication with Trusted Hardware

Michael Kirkpatrick
Department of Computer Science
Purdue University
mkirkpat@cs.purdue.edu

Elisa Bertino
CERIAS
Purdue University
bertino@cerias.purdue.edu

Abstract

Modern computer systems permit mobile users to access protected information from remote locations. In certain secure environments, it would be desirable to restrict this access to a particular computer or set of computers. Existing solutions of machine-level authentication are undesirable for two reasons. First, they do not allow fine-grained application layer access decisions. Second, they are vulnerable to insider attacks in which a trusted administrator acts maliciously.

In this work, we describe a novel approach using secure hardware that solves these problems. In our design, multiple administrators are required for installation of a system. After installation, the authentication privileges are physically linked to that machine, and no administrator can bypass these controls. We define an administrative model and detail the requirements for an authentication protocol to be compatible with our methodology. Our design presents some challenges for large-scale systems, in addition to the benefit of reduced maintenance.

1 Introduction

Modern information systems are designed to give users the ability to access data and perform tasks from a variety of settings and locations. In a typical environment, a user must enter a valid username and password to gain access to the system. Although this combination of policy and mechanism often provides a sufficient level of security, there are many cases in which more protection is needed. Specifically, in certain settings, it would be desirable to bind an authentication request to a single computer or set of computers, thus requiring the user's physical presence at an authorized location.

There are many applications of such a restricted authentication mechanism. In a corporate or government setting, users at multiple sites may need to access sensitive data stored on a remote server. Critical infrastructure systems may require strong assurances of provenance to guarantee that only trusted devices can generate data. The security of sensitive web-enabled transactions could also be strengthened by allowing a user to voluntarily restrict his own access to his personal computer.

In this work, we introduce a novel approach to multifactor authentication that uses the physical properties of the device itself. Our motivating example application is that of an organization, such as a corporation, that possesses a server storing data at multiple levels of sensitivity. Users are granted access to the most secure data only from particular secured workstations. Physical access to the workstations is restricted and monitored.

A naive approach to this problem is to authenticate the machine and to use end-to-end encryption. Examples of this approach would be the use of Challenge-Handshake Authentication Protocol

(CHAP), Transport Layer Security (TLS), or Internet Protocol Security (IPsec). We consider these cryptographic techniques to be a necessary part of our design, but they are not sufficient for the level of security that we require. Specifically, we identify two problems with this approach.

First, we require fine-grained authentication at the application layer of the protocol stack, not just the network or transport layers. For example, a user may use the same client software from multiple workstations to connect to the same server. However, only one of the workstations is secured for use with the protected data. From an unprotected workstation, the client should be granted access to *some* of the data, according to the security level. This fine-grained decision can only be made by the server application.

Second, we aim to ensure protection against insider threats. Government and academic researchers have shown that the insider threat problem is difficult and more common than many believe [16, 4, 25, 5]. A system administrator, who is a trusted insider, can abuse his privileges to reconfigure the network settings to grant access to an unprotected workstation with the intent of sending the data to an external recipient. An insider with sufficient knowledge and access could do irreversible damage before detection occurs. Our approach prevents any single insider from bypassing the security mechanism.

Our work is motivated in part by recent work in access control, such as the integration of spatial and temporal constraints into role-based access control [6, 14, 1], and the use of contextual information in pervasive systems [17]. These approaches consider the user's location or ambient conditions as part of the criteria for access control decisions. However, these approaches primarily focus on abstract models for access control and do not describe *how* the user proves his location. Our work accomplishes this task by binding authentication to machines in fixed locations.

Our design is enabled by advancements in binding software and data to specific hardware through the use of Physical Unclonable Functions (PUFs) [2, 23, 12, 11]. Given a challenge input C_i, a PUF uses slight variations that exist in hardware to create a unique response R_i. For example, PUFs based on Static Random Access Memory (SRAM) use the initial fluctuations in memory locations during power-up to create a unique binary result that is intrinsically different for each SRAM instance. These results are tamper-proof and unpredictable outside of the hardware.

While Trusted Platform Modules (TPM) [24] and secure coprocessors [7] can be use for a number of trusted computing tasks, our interest in PUFs is driven by the following factors. First, PUFs use characteristics of the hardware that cannot be modified, forged or reset, even with physical access to the device. Thus, PUFs provide a high level of integrity. Next, PUFs exist for a wide range of technologies for which TPM is inappropriate. For example, PUFs can be created for magnetic strip cards, and embedded devices built on FPGAs can use SRAM-based PUFs without modification of the hardware. Consequently, our approach can be applied to a variety of settings, ranging from remote access using a workstation to large-scale interconnected networks of embedded devices.

An example usage of PUF technology is to protect a cryptographic key K. From a high-level perspective[1], K is stored by computing the XOR $W = K \oplus R_i$ for some challenge response R_i. The value W is then stored locally. To reconstruct the key at run-time, the challenge C_i is presented to the PUF, which recovers the key as $K = W \oplus R_i$. In the case of SRAM-based PUFs, this computation and the resultant cryptographic operations all occur on the processor, so K exists only in registers and is never present even in memory.

[1]PUFs involve noisy data, so the details of the key storage mechanism are actually more complex than presented here, as regeneration of the key requires the use of a fuzzy extractor algorithm. We refer the reader to [11] for a more detailed description of the process.

In this paper, we define an architecture and administrative model that combines these hardware-binding techniques with the goal of physically restricted authentication. Our approach requires k-of-n threshold cryptography [21] during installation of the secret key to the protected workstation. That is, no single administrator can install a new key unilaterally. Additionally, we define the required behavior for an authentication protocol that will work with our design, using the Feige-Fiat-Shamir [9] identification scheme as a basis. We provide a sketch of our security evaluation and describe our future work for this project.

2 Related Work

The literature of computer science contains a long history of identification schemes and authentication protocols [19, 20, 18, 9, 10]. Modern research in this area has become more focused on addressing issues of identity under specialized circumstances, such as internet banking [8], secure roaming with ID metasystems [15], digital identity in federation systems [3], and authentication for location-based services [13]. None of this work has addressed the issue of binding authentication to an individual computer.

The use of PUFs for cryptographic applications has been proposed in recent work [23, 12, 11]. These works have described the technical details of PUFs and how they can be used to protect cryptographic keys. However, these works do not focus on specific applications and do not examine the use of PUF against insider threats. Our work focuses on identifying a key problem of restricted authentication and showing how PUF can be used as a solution at the application layer.

3 Restricted Authentication

In this section, we start by describing the main goals underlying our system design, then describe an architecture and administrative model for performing the restricted multifactor authentication. We then detail the required behavior for an authentication protocol under our design.

3.1 Design Goals

In designing our protocol for restricted authentication, we set the following goals.

Prevent transfer of keys. As the goal of our system is to restrict authentication to a particular device, we must prevent the transfer of keys from one machine to another.

Mandatory enforcement. Legitimate users should not be able to circumvent the mechanism in an attempt to violate policy. Thus, there should be no way for users or administrators to bypass the system controls.

No administration. Once a machine is initially configured, we do not want to permit or require an administrator to be involved in future changes with regard to the authentication mechanism. That is, if a workstation is moved to a new location (with a new IP address), no administrator should be required to change any settings related to our protocols. Note that this does not preclude changes that occur as part of routine system maintenance. Rather, our goal is only to prevent any change to the installed key after the system initialization.

Resilience to attacks. A protocol used in our architecture must be resilient against known attacks on authentication systems. These attacks include replay and key leakage.

3.2 Architecture

Our architecture consists of the following principals:

Client C A secured workstation initiating the authentication session. Our vision assumes a number of such clients located at remote physical locations, although we generally refer to a single client for simplicity.

Server S The centralized server hosting the requested service. The server can store both protected and unprotected data.

Administrator A A system administrator. When multiple administrators are required, we refer to the group as A^*.

In our design, C is assumed to be more secure than other workstations in the organization. For instance, physical access to C is restricted and monitored, Mandatory Access Control (MAC) policies are enforced to prevent unauthorized configuration changes, and remote execution of code on C (e.g., access through SSH) is disabled. Furthermore, appropriate network layer technologies, such as TLS, SSL, or IPsec, are in place to secure communication to and from C.

We also assume C has a tamper-proof key storage mechanism, specifically a PUF. Recall that PUFs compute challenge-response pairs (C_i, R_i) that are unique to the hardware, and a key K can be stored by computing $W = K \oplus R_i$. Thus, K is not explicitly stored and exists only at run-time. In the case of SRAM PUFs, K exists only on the processor and is never even present in main memory. Transferring W to another machine is useless, as the PUF on that device cannot compute the same response R_i.

It would be unrealistic to assume C is completely trusted, as malicious or vulnerable code is always a possibility. As such, we aim to minimize any assumption of trust in the software on C. As in [11], we assume the hardware contains a module that restricts PUF access to trusted applications. In our approach, our protocols require only that the installation and authentication be trusted. Given the design of these protocols, we find this assumption to be a reasonable compromise. As a result of our assumptions regarding C, we can make the following statements:

- If C initiates an authentication session to S, the program is guaranteed to be executing on C and not from a remote workstation using C as an intermediate resource.
- C enforces MAC policies that prevent information flow to insecure channels, such as removable storage devices. That is, we do not consider the challenge of securing the application after authentication occurs.
- Only the installation and authentication routines have access to execute the PUF, and they are trusted to behave according to the protocol design.

Finally, we must address the configuration of S itself. S is a centralized server that is storing both sensitive and unprotected data. The goal of our work is to provide controlled remote access to the protected data. As such, we assume that S is trusted to store keys securely. That is, if an attacker can steal a key K from S, he can steal the sensitive data directly without having to bypass our security mechanism.

Note, though, that the attacker can be an administrator with valid access to S. One threat in relation to S that we do consider is if the administrator is trying to steal a copy of the list of workstation secret keys. As we will show in Section 4, this attack will fail.

3.3 Administrative Model

One of the keys to the success of our approach is that administrative actions must be tightly controlled. Specifically, no single administrator should be able to install a key for a machine by himself. Then, once the key has been installed, no user should be able to extract and store the key.

To guarantee separation of duty for administrators, we use k-of-n threshold cryptography, such as Shamir's secret sharing. In these schemes, a secret is split into n parts, k of which can be used to reconstruct the secret. When a new secured workstation is set up, S generates a new secret key K that is specific to that new machine, storing a copy of K locally on S for the authentication process. S splits the key into n parts, and each administrator retrieves his share accordingly.

While we do not make specific requirements regarding the values k and n, it is important to prevent reuse of the secret shares. For example, if $k < n/2$, then there are at least 2 distinct subsets of the secret shares that can be used to reconstruct the secret key x twice. Similarly, k administrators should not be able to enter their secret share on two different workstations. As such, C must inform S when k shares have been entered, and S must then invalidate the shares to prevent reuse. The next section describes the installation process in more detail.

3.4 Protocols

We define the following protocols for installing cryptographic keys and authenticating the machine.

3.4.1 Installation

When a new machine is to be installed, S generates a symmetric key K and a number $N = pq$, where p and q are large primes. K is split into $K_1, \ldots, K_n$, and distributed to the n administrators A^*. One approach to secure key distribution is described in [22]. As such, we do not address this topic in detail and simply assume the distribution is handled securly. At least k of A^* enter their shares K_i and their identifiers ID_i to C. For the sake of simplicity, we will refer to the k shares entered as $K_1, \ldots, K_k$, although these may not necessarily correspond to the first k shares of the n. C creates and stores a random C_0 to present as a challenge to the PUF, which responds with R_0. C then stores $W = K \oplus R_0$ for future communication with S.

In addition, C evaluates the PUF using $C_0 \oplus ID_i$ as a challenge for each i, yielding R_i. For our authentication scheme, we need $gcd(R_i, N) = 1$, but that may not necessarily be the case. As a result, C must create bit strings b_i such that $X_i = R_i \oplus b_i$ and $gcd(X_i, N) = 1$. These bit strings b_i are then stored locally on C. Assuming C has a hardware identifier ID_{hw}, the following protocol describes the installation process.

1. $[A^* \rightarrow C] : K_i,\ ID_i,\ N,\ k$[2]
2. $[C \rightarrow S] : e_K(ID_1, \ldots, ID_k, H(K_1, \ldots, K_k), ID_{hw}, X_1^2, \ldots, X_k^2 \pmod N)$
3. $[S \rightarrow C] : e_K(accept, ID_{hw}, T)$, or *reject*

[2]Note that allowing the administrators to state the value of k does not lead to a vulnerability. If one or more lies about the value of k, then the reconstructed key K would be incorrect and the installation would fail. Thus, incorporating the distribution of k in the protocol is just a matter of convenience, as k does not have to be stored anywhere.

S decrypts the message in step 2 and uses the list of ID_i values to reconstruct the hash $H(K_1, \ldots, K_k)$. If the hash does not match, S destroys K and rejects the installation. Otherwise, it responds with an encrypted accept message with a timestamp T[3].

3.4.2 Authentication

Assuming the installation was performed and S was able to confirm the shares used, we define the following authentication protocol, which is based on the Feige-Fiat-Shamir identification scheme [9].

1. $[C \rightarrow S] : ID_{user}, ID_{hw}, x \equiv +/- r^2 \pmod N$
2. $[S \rightarrow C] : e_K(ID^*)$
3. $[C \rightarrow S] : e_K(y \equiv r \cdot \Pi(R_i \oplus b_i)^{e_i} \pmod N, p, K_s)$

Here, r is a random value and ID^* indicates a random subset of $ID_1, \ldots, ID_k$. The exponents $e_i = 1$ if $ID_i \in ID^*$, and $e_i = 0$ otherwise. For each ID_i, C presents the challenge $C_0 \oplus ID_i$ to the PUF, which returns R_i. This response is then XORed with the correction bits b_i to ensure $gcd(R_i \oplus b_i, N) = 1$, ensuring our protocol works identically to that of Feige-Fiat-Shamir. S accepts the proof if $y^2 \equiv +/- x \cdot \Pi X_i^{2e_i} \pmod N$, and the user's password p is correct. K_s is a session key derived from using r as a challenge input to the PUF.

3.5 Scalability

Our architecture and protocols were designed with the goal of restricting device authentication using trusted hardware techniques. The use of secret sharing for key distribution and PUF evaluation for authentication accomplish these goals. However, providing a robust level of security while at the same time achieving scalability entails addressing many challenges.

The choice of k requires consideration. If k is large, there may not be enough administrators available at any given time for an installation, which may be prohibitive. Reducing k would solve this problem, but S would have fewer combinations of ID^* to present as a challenge, which would weaken the protocol. A possible compromise would be to have a smaller k while having C generate additional challenge-response pairs using random identifiers ID_i. An additional challenge for large-scale systems is the storage of k or more pairs of $i(ID_i, X_i)$ for each machine. If many devices have been installed, allocating enough storage may not be trivial. Addressing this challenge requires further consideration and detailed analysis of storage requirements.

Although our design presents challenges for scalability, it also presents benefits. Specifically, the administrative cost resulting from our protocols is paid entirely in the installation. Once the keys have been installed and the device registered, administrators do not need to perform additional work when devices are moved. This advantage is a direct result of binding the authentication to the hardware, rather than to transient properties, such as an IP address. As a result, our design allows devices to be redeployed as necessary without explicit reconfiguration. For applications where devices are repeatedly used for short periods of time then moved to new locations, this approach could enhance the scalability of the system. Assessing the actual administration costs would require to develop a detailed cost model that also includes costs of user activities, as in models found in information system management.

[3]The intent of the encryption in this response is to serve as a digital signature. If a public key infrastructure exists, such could be used instead of symmetric key encryption.

4 Security Analysis

In examining the security guarantees of our system, we focus on replay and collusion attacks, as other classes of attacks are not applicable. Specifically, our assumption that end-to-end encryption protects network traffic prevents eavesdropping and modification attacks. Typing attacks are only applicable to implementations of protocols, not the protocol definitions themselves.

The only attack in which we distinguish between an administrator and a regular user is in attacking the secret sharing scheme. In any other potential attack scenario, having administrator privileges are of no benefit, so we do not make the distinction. In our analysis, we show that our design is resilient under very few assumptions regarding an attack profile.

Malicious administrator. As one of the primary motivations for our work is to defend against an insider threat, we examine that scenario first. Specifically, we consider the case where an administrator (or group of administrators) wishes to bypass the security mechanisms of our design to grant illicit access to an untrusted workstation. Note that, in some circumstances, the attacker may not care if he gets caught. Thus, if he succeeds in getting access for even a moment, we consider his attack successful.

The goal for a malicious administrator, then, is to grant access to an untrusted workstation. Note that the k-of-n secret sharing assumption is that k must be large enough that a colluding group of that size is unlikely to happen. If there are at least k colluding administrators, their attack *will* be successful, because there is no mechanism to distinguish between an attack and a legitimate installation. Thus, if k is sufficiently large, this scenario cannot occur.

If there are $j < k$ colluders, attempting to guess the missing shares can succeed with negligible probability. Using Shamir's scheme to distribute the key K, the shares are constructed as $(x, f(x))$ where x is randomly chosen between 0 and $q - 1$ (for a large q) and f is a polynomial of degree $k - 1$. Assuming all the x inputs are chosen separately, guessing the $k - j$ shares occurs with a probability that is approximately $1/q^{k-j}$. That is, such an attempt is certain to fail in practice.

The other possible attack would be for the administrator to steal the list of secret values from S, with the goal of enabling a new machine. This attack will fail, though, because the responses R_i are never explicitly revealed to S. Under the assumption that performing a modular square root is difficult, as used in the Feige-Fiat-Shamir scheme, an attacker cannot extract $(R_i \oplus b_i)$ from $(R_i \oplus b_i)^2 \pmod{N}$. Thus, stealing the list of challenge responses is of no avail to such an attacker.

Attacks from *C*. Assume an attacker has a valid username and password, and is attempting to launch an attack from C. The goals of such an attack could be to connect to S through C (e.g., via SSH) to bypass the physical restriction, to connect from another machine posing as C, or to steal the secret key for installation on different hardware. Recall that our architecture stipulated that only trusted applications can execute the PUF. Assuming the authentication module has been designed to prevent attempts to launch it from such a service, the attempt will fail. Furthermore, if the attacker tries to modify the authentication code, the application will no longer be trusted to execute the PUF. Hence, the attacker cannot bypass the physical restriction.

The second goal, to connect from a machine posing as C will also fail, even if he has the key K. S maintains a mapping of the secret values and keys, so K is bound to the values X_i^2. Again, using the modular square root assumption, the attacker cannot retrieve X_i from X_i^2, so he cannot complete the protocol without having access to the PUF responses R_i for that machine.

The last possibility is that the attacker would attempt to steal the key K and the values $X_1, \ldots, X_k$. As we have described previously, only trusted applications have access to execute the PUF, and the values $X_1, \ldots, X_k$ exist only at run-time on the processor. Thus, the attacker cannot

reconstruct these values from any stored data.

Policy-violating user. This scenario assumes that a legitimate user wishes to bypass the restricted authentication policy, but without the presumption of malice. Rather, the user attempts to violate the policy as a matter of convenience, to gain access from a workstation of his choice. However, as before, the user must be able to bypass both the secure hardware and the network protections. Even if the user is capable of these attacks, the user would be accepting too great a risk in doing so. Specifically, the user's activity is likely to be detected by an auditing mechanism or an intrusion detection system. Thus, the risk of this user getting caught is high. As we assumed this user was not malicious, we find that the challenges and the risks involved provide a strong disincentive to such a user. Consequently, we do not believe this to be a realistic threat.

Replay attacks. Here, we assume that an attacker has a transcript of a successful authentication session. That is, he can see both the random commitment $x \equiv +/-r^2 \pmod{N}$, $e_K(ID^*)$, and the encrypted response. Without knowledge of K, the attacker must simply replay the encrypted response in its original form. However, as r and the subset ID^* are different at each session, squaring the response y will not yield the correct result. Thus, our approach is resilient to such attacks.

5 Conclusion and Future Work

While authentication systems have long been studied, our work is unique in addressing the issue of restricting authorization to a pre-defined set of computers at the application layer. We have defined an architecture and an administrative model to enforce the policy restriction. Under our design, no single administrator can act unilaterally to enable access from an additional workstation. Rather, installation of a new workstation uses k-of-n threshold cryptography to require multiple administrators act in coordination. This administrative model reduces the threat of an insider attack on protected data. We have also detailed the architectural requirements and devised an authentication protocol for our scheme. We have provided a sketch of the security guarantees of our approach.

Our work, which has focused on the architectural assumptions and protocols necessary for using PUFs to defend against insider threats, has many open research directions. Our future work involves implementing such a design on hardware enabled with PUFs, such as a Field-Programmable Gate Array (FPGA), to evaluate the performance overhead involved in our scheme. We also plan to develop a comprehensive cost-benefit analysis to assess the "administration scalability" of our approach. Additionally, we have provided rudimentary analyses of the security guarantees of our system. In future work, we plan to offer a more formal analysis and prove the resilience of our design against certain types of attacks.

References

[1] Aich, S., Sural, S., and Majumdar, A. K. Starbac: Spatiotemporal role based access control. In *OTM Conferences* (2007).

[2] Atallah, M. J., Bryant, E. D., Korb, J. T., and Rice, J. R. Binding software to specific native hardware in a vm environment: The puf challenge and opportunity. In *VMSEC '08* (2008), ACM.

[3] Bhargav-Spantzel, A., Squicciarini, A. C., and Bertino, E. Establishing and protecting digital identity in federation systems. In *Proceedings of the 2005 ACM Workshop on Digital Identity Management* (2005), ACM, pp. 11–19.

[4] Chinchani, R., Iyer, A., Ngo, H. Q., and Upadhyaya, S. Towards a theory of insider threat assessment. In *International Conference on Dependable Systems and Networks (DSN '05)* (2005).

[5] CSO Magazine and CERT and United States Secret Service. 2004 e-crime watch survey: Summary of findings. http://www.cert.org/archive/pdf/2004eCrimeWatchSummary.pdf, 2004.

[6] Damiani, M. L., Bertino, E., Catania, B., and Perlasca, P. Geo-rbac: A spatially aware rbac. *ACM Transactions on Information Systems and Security* (2006).

[7] Dyer, J. G., Lindemann, M., Perez, R., Sailer, R., van Doorn, L., Smith, S. W., and Weingart, S. Building the ibm 4758 secure coprocessor. *Computer 34*, 10 (2001), 57–66.

[8] Federal Financial Institutions Examination Council. *Authentication in an Internet Banking Environment*, October 2005.

[9] Feige, U., Fiat, A., and Shamir, A. Zero knowledge proofs of identity. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing* (1987), pp. 210–217.

[10] Fiat, A., and Shamir, A. How to prove yourself: Practical solutions to identification and signature problems. In *Proceedings on Advances in Cryptology (CRYPTO '86)* (1987), Springer-Verlag, pp. 186–194.

[11] Guajardo, J., Kumar, S. S., Schrijen, G.-J., and Tuyls, P. Fpga intrinsic pufs and their use for ip protection. In *Proceedings of the 9th Cryptographic Hardware and Embedded Systems Workshop (CHES)* (2007), pp. 63–80.

[12] Guajardo, J., Kumar, S. S., Schrijen, G.-J., and Tuyls, P. Physical unclonable functions and public-key crypto for fpga ip protection. In *International Conference on Field Programmable Logic and Applications* (2007), pp. 189–195.

[13] Han, K., and Kim, K. Enhancing privacy and authentication for location based service using trusted authority. In *2nd Joint Workshop on Information Security* (2007), Information and Communication System Security.

[14] Hansen, F., and Oleschuk, V. Srbac: A spatial role-based access control model for mobile systems. In *Proceedings of the 8th Nordic Workshop on Secure IT Systems (NORDSEC '03)* (October 2003), pp. 129–141.

[15] Hoang, L. N., Laitinen, P., and Asokan, N. Secure roaming with identity metasystems. In *IDtrust '08* (March 2008), ACM.

[16] INFOSEC Research Council (IRC). Hard problem list, 2005.

[17] Kulkarni, D., and Tripathi, A. Context-aware role-based access control in pervasive computing systems. In *Proceedings of the 14th Symposium on Access Control Models and Technologies (SACMAT)* (2008).

[18] Miller, S. P., Neuman, B. C., Schiller, J. I., and Saltzer, J. H. Kerberos authentication and authorization system. In *Project Athena Technical Plan* (1987).

[19] Needham, R. M., and Schroeder, M. D. Using encryption for authentication in large networks of computers. *Communications of the ACM 21*, 12 (December 1978), 993–999.

[20] Otway, D. J., and Rees, O. Efficient and timely mutual authentication. *ACM SIGOPS Operating Systems Review 21* (January 1987), 8–10.

[21] Shamir, A. How to share a secret. *Communications of the ACM* (1979).

[22] Shang, N., Paci, F., Nabeel, M., and Bertino, E. A privacy-preserving approach to policy-based content dissemination. Tech. Rep. 2009-14, Purdue University Center for Education and Research in Information Assurance and Security (CERIAS), 2009.

[23] Suh, G. E., and Devadas, S. Physcal unclonable functions for device authentication and secret key generation. In *Proceedings of the 44th IEEE Design Automation Conference (DAC)* (2007), IEEE Press, pp. 9–14.

[24] Trusted Computing Group. Trusted Platform Module Main Specification. http://www.trustedcomputinggroup.org/, October 2003.

[25] United States Secret Service and CERT Coordination Center. Insider threat study: Illicit cyber activity in the banking and finance sector. http://www.secretservice.gov/ntac/its_report_040820.pdf, August 2004.