Physically Restricted Authentication with Trusted Hardware

Michael Kirkpatrick  
Department of Computer Science  
Purdue University  
West Lafayette, IN  
mkirkpat@cs.purdue.edu

Elisa Bertino  
Department of Computer Science  
Purdue University  
West Lafayette, IN  
bertino@cs.purdue.edu

ABSTRACT
Modern computer systems permit 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.

Categories and Subject Descriptors

General Terms
Security

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 a corporation that processes sensitive data as part of a government contract. This data is stored along with less sensitive data on a central server that we assume to be protected. Users are granted access to the most secure data only from particular secured workstations at remote locations. All access to these workstations is tightly controlled and monitored. In order to ensure that users at only these workstations are granted access, the server application must be able to identify the machine making the request as a trusted client.

A naïve 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 [19, 4, 30, 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, 17, 1], and the use of contextual information in pervasive systems [20]. 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 thus extends previous research on access control by addressing the challenge of enforcing these models.

Our design is enabled by advancements in binding software and data to specific hardware through the use of Physical Unclonable Functions (PUFs) [13, 21, 2, 27, 15, 14]. 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. Furthermore, the inherent limitations of manufacturing physical devices prevent the intentional creation of two devices that demonstrate identical PUF behavior.

Other secure hardware technologies, such as secure co-processors [9] or Trusted Platform Modules (TPM) [29] can be used for a number of trusted computing tasks. In fact, the AEGIS secure processor [28] incorporates a silicon PUF. However, our interest in PUFs is driven by a number of factors. First, our focus is on the general problem of physically restricted authentication. That is, the client requesting authentication may be a device for which these other technologies are inappropriate. For example, our general approach to physical authentication can be applied to embedded devices, FPGA-based applications, or RFID systems. Thus, our scheme can be deployed in a variety of settings, ranging from remote access using a workstation to large-scale interconnected networks.

The other factor driving our interest in PUFs is the high level of integrity they provide. That is, PUFs use characteristics of the hardware that cannot be modified, forged or reset, even with physical access to the device. Additionally, PUFs have been shown to be resilient to a wide variety of environmental conditions, such as changes in temperature [8]. Thus, the use of PUFs creates a robust defense against attacks on the authentication process.

An example usage of PUF technology is to protect a cryptographic key $K$. From a high-level perspective, $K$ 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.

In this paper, we define an architecture and administrative model that describes the application of these hardware-binding techniques with the goal of physically restricted authentication. Our approach requires $k$-of-$n$ threshold cryptography [25] 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 [11] 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 [23, 24, 22, 11, 12]. Modern research in this area has become more focused on addressing issues of identity under specialized circumstances, such as internet banking [10], secure roaming with ID metasystems [18], digital identity in federation systems [3], and authentication for location-based services [16]. 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 [27, 15, 14]. These works have described the technical details of PUFs and how they can be used to protect cryptographic keys. Additional works [8, 7] have focused on the problem of identifying RFID devices. However, these works do not explore the application of PUFs to recent access control models, and do not examine the use of PUFs as a defense against insider threats. In contrast, our work focuses on identifying the problem of physically restricted authentication. We then show how PUFs can be used as a solution for this problem, restricting remote access at the application layer.

## 3. PHYSICALLY RESTRICTED AUTHENTICATION

In this section, we define the problem of physically restricted authentication and identify the design goals that must be satisfied in a solution. We then describe an architectural and administrative model for addressing this problem. We then detail the required behavior for an authentication protocol under our design.

### 3.1 Problem Statement

We define the problem of physically restricted authentication to be the binding of an authentication request to a set of known hardware devices. A solution to this problem requires satisfying the following constraints.

**Succe$$\text{ss for correctness.** The physical restriction mechanism must never cause the authentication to fail when correct credentials (i.e., username and password) are presented.**

**Unforgeability.** An attacker should not be able to provide a false proof that satisfies the restricted authentication with greater than negligible probability.

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

**No administration.** With the exception of the original installation process, the authentication mechanism must not require or permit modification, even by system administrators². For example, if a machine is moved to a new location (with a new IP address), no administrator should be required to change any settings to permit authentication from the new location.

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, but the protected data must only be provided to the secured workstations.

**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 in the form of a PUF. Recall that PUFs compute challenge-response pairs (Cᵢ, Rᵢ) that are unique to the hardware, and a key K can be stored by computing W = K ⊕ Rᵢ. 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 Rᵢ.

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 [14], 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:

- 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, we assume he can steal the sensitive data directly without having to bypass our security mechanism.

Another assumption is that the attacker can be an administrator with valid access to S. Consequently, we consider two specific threats in relation to S. First, the administrator could attempt to steal a copy of the list of the workstation authentication tokens. Second, the administrator could try to authorize an untrusted machine unilaterally. As we will show in Section 4, both attacks will fail.

3.3 Administrative Model

One crucial goal of our design is that administrators must be prevented from acting unilaterally. Specifically, no single administrator should be able to configure a workstation for authentication by himself. 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 machine, and stores K locally on S for use in the installation. S splits this 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 K twice. In practice, it will be difficult to have a large number of administrators involved during an installation. Hence, a practical approach may be to have n = 3 and k = 2.

A similar challenge is that 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 K₁, . . . , Kₙ, and distributed to the n administrators A∗. One approach to secure key distribution is described in [26]. As such, we do not address this topic in detail and simply assume the distribution is handled securely.

²Note that our goal of “no administration” only relates to the authentication mechanism. That is, it does not preclude changes that occur as part of routine system maintenance.
In addition to the key $K_i$, $S$ generates a random challenge $C_i$ for each administrator\(^3\). For installation, at least $k$ of $A^*$ enter their shares $K_i$ and $C_i$ into $C$. We denote the $k$ shares entered as $K_1, \ldots, K_k$ and $C_1, \ldots, C_k$, although these may not necessarily correspond to the first $k$ shares of $n$.

In addition, for each $C_i$, $C$ evaluates the PUF, 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 $ID_{hw}$, and $e_K$ denotes symmetric key encryption under the key $K$, the following protocol describes the installation process.

1. $[A^* \rightarrow C] : K_i, C_i, N, k$
2. $[C \rightarrow S] : e_K(C_1, \ldots, C_k, H(K_1, \ldots, K_k), ID_{hw}, X_1^2, \ldots, X_k^2 \pmod{N})$
3. $[S \rightarrow C] : e_K(\text{accept or reject}, ID_{hw}, T)$

$S$ decrypts the message in step 2 and uses the list of $C_i$ values to reconstruct the cryptographic 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 $T$. Once the response has been sent from $S$, $S$ destroys all $C_i$, storing only $X_1^2, \ldots, X_k^2 \pmod{N}$ in order, along with the associated $ID_{hw}$.

### 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 [11].

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

In this protocol, $r$ is a random integer and $I^*$ indicates a set of random integers between 1 and $k$. That is, for each $i \in I^*$, $C$ executes the PUF with the challenge $C_i$, 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 $y^2 \equiv +/ - x \cdot \prod_{i \in I^*} X_i^2 \pmod{N}$, and the user’s password $p$ is correct.

### 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 it increases the vulnerability of the scheme to collusion among administrators. An additional challenge in regard to $k$ for large-scale systems is the storage of all $X_i^2$ for each machine. If many devices have been installed, allocating enough storage may not be trivial. Addressing these challenges requires further consideration and detailed analysis of the storage requirements and deployment scenario.

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 developing a detailed cost model that also includes costs of user activities, as found in models for information system management.

## 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.

Our threat model assumes the attacker has administrator privileges on both $S$ and $C$. In regard to $S$, the adversary is able to see all data stored, including the secret values $X_i^2$. However, we do not consider attacks involving the corruption of the data stored on $S$. This assumption is made because an attacker capable of this corruption could simply corrupt or steal the sensitive data. Rather, for attacks on $S$, we focus on an attempts to model a machine’s PUF to permit an untrusted machine to fake the authentication protocol successfully.

For $C$, we assume the machine restricts PUF execution to trusted applications. Thus, if an attacker tampers with the software executing the protocol on $C$ (e.g., to prevent the encryption), permission to execute the PUF is denied. As with $S$, our focus is on attempts to enable access from an untrusted machine. Consequently, we assume that an adversary can install software to eavesdrop on the protocol and make a transcript of the session.

Our initial analysis focuses on the attacks involving malicious administrators attempting to bypass the physical restriction. We also describe the protocol’s resilience against replay attacks, and consider the incentives for a legitimate user who wishes to bypass the restrictions as a matter of convenience. Note that in some cases, an attacker may not care if he gets caught. Therefore, if the attack can succeed
with non-negligible probability for even a single session, we consider the attack successful as a whole.

**Malicious administrator.** As one of the primary motivations for our work is to defend against an insider threat, we examine that scenario first. In this scenario, the primary goal is to bypass physical restriction mechanism to enable access from an untrusted workstation.

One possible attack to consider is a group of colluding administrators performing an installation on an untrusted machine. However, the intent of the k-of-n secret sharing is that k must be large enough that forming a colluding group of that size is unlikely to happen. If there are at least k colluding administrators, their attack will be successful. This result is due to the fact that there is no mechanism to distinguish between an attack and a legitimate installation.

If there are \( j < k \) colliders, the attackers must first construct the key \( K \) from their \( j \) shares. Using Shamir’s scheme to distribute \( K \), the shares are constructed as \((x, f(x))\), where \( x \) is randomly chosen. The strength of Shamir’s scheme means that \( f \) cannot be modeled with fewer than \( k \) shares. Thus, if we assume the probability of guessing a single pair \((x, f(x))\) is \(1/q\), then the probability of successfully reconstituting the key is \(1/q^{k-j}\), which is negligible in practice.

The other possible attack would be for the administrator to steal the list of secret values from \( S \), with the goal of modeling the PUF to imitate the behavior from an untrusted machine. First, note that quadratic residues have at least four possible modular square roots. That is, given \( X^2 \), an attacker can guess \( X \) with at best a probability of 1/4. Consequently, if \( S \) requests a set of 10 challenges, the probability of successfully guessing the responses \( X_i \) and completing the protocol is \(1/2^{20}\). Thus, even for small challenge sets, the probability of a successful attack is negligible.

Additionally, recall that our authentication protocol is based on the Feige-Fiat-Shamir identification scheme. This scheme is a zero-knowledge protocol, which means that each authentication session reveals no additional information that would help to model the secret values \( X_i \). Thus, an attacker who can observe the authentication protocol (either online or from a recorded transcript) cannot guess the value of all \( X_i \) with non-negligible probability. Furthermore, even if the attacker can see that stored bit strings \( h_i \), he has no information to accurately model the PUF behavior for the pairs \((C_i, R_i)\).

**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 \( x \equiv +/− r^2 \mod N \), \( I^* \), and the response. As described above, this protocol is zero-knowledge, so the response is of no avail to an attacker trying to determine the secret values \( X_i \). In addition, recall that \( I^* \) changes at each session. Thus, given \( k \) challenge-response pairs stored for \( C_i \), the probability of successfully reusing the \( r \) and \( y \) values is \(1/2^k\). As \( k \) increases, this probability of success becomes negligible. The system could be made even more resilient by prohibiting reuse of the same \( r^2 \), if necessary.

**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.

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 the use of PUFs as a defense 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. Another direction for future consideration is to develop a novel authentication protocol that is based exclusively on the properties of the PUF, rather than to adapt existing work (in this case, the Feige-Fiat-Shamir scheme) to this physical dimension. 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.

6. REFERENCES


