User-Centric Handling of Identity Agent Compromise

Daisuke Mashima, Mustaque Ahamad, and Swagath Kannan

Georgia Institute of Technology, Atlanta GA 30332, USA

Abstract. Digital identity credentials are a key enabler for important online services, but widespread theft and misuse of such credentials poses serious risks for users. We believe that an identity management system (IdMS) that empowers users to become aware of how and when their identity credentials are used is critical for the success of such online services. Furthermore, rapid revocation and recovery of potentially compromised credentials is desirable. By following a user-centric identity-usage monitoring concept, we propose a way to enhance a user-centric IdMS by introducing an online monitoring agent and an inexpensive storage token that allow users to flexibly choose transactions to be monitored and thereby to balance security, privacy and usability. In addition, by utilizing a threshold signature scheme, our system enables users to revoke and recover credentials without communicating with identity providers. Our contributions include a system architecture, associated protocols and an actual implementation of an IdMS that achieves these goals.

1 Introduction

Digital identity credentials, such as passwords, tokens, certificates, and keys, are used to ensure that only authorized users are able to access online services. Because of sensitive and valuable information managed by such services, they have become targets of a variety of online attacks. For example, online financial services must use stronger credentials for authentication to avoid fraud. Because of the serious nature of threats and widespread theft and misuse of identity credentials, there is considerable interest in the area of identity management, which addresses secure use of such identity credentials. User-centric identity management, which allows users to flexibly choose what identity information is released to other entities, offers better control over the use of identity credentials. For instance, users can choose an identity provider that they believe is the most appropriate for each transaction. However, such user-centricity requires that disclosure of identity information needs to be under user control and also expects users to assume more responsibility over their identity usage owing to the absence of a centralized authority [1]. This would be possible only when users have a certain level of awareness and control of how and when their identity credentials are utilized.

To satisfy the user-centricity requirement, several currently proposed user-centric IdMSs rely on agent software, which we call an identity agent, that carries

M. Backes and P. Ning (Eds.): ESORICS 2009, LNCS 5789, pp. 19–36, 2009.

out a number of tasks related to management of identity credentials on behalf of the user. Identity agents can be deployed on users' devices or on networked entities. These agents assist users and thereby help reduce the burden imposed on them by an IdMS. For example, Windows CardSpace [2] utilizes client-side software to help users manage meta-data related to identity credentials as well as a certain type of authentication credentials used with online identity providers. Another example is GUIDE-ME (Georgia tech User-centric IDEntity Management Environment) [3][4] that utilizes local identity agents installed on users' devices to control network-resident identity agents that store and manage identity credentials originally issued by identity providers. While an identity agent running on a readily accessible device can potentially offer increased user awareness and flexible control, the nature of a local identity agent on a mobile device will make it an attractive target of theft. In addition, since such devices sometimes are managed by non-expert users, attacks by means of malware are also a concern. The compromise of such agents could allow adversaries to access stored identity credentials and result in possible disclosure of sensitive information, including breach of authentication and authorization in a system where access to services must only be provided to legitimate users. Clearly, we must deal with the problem of misuse of such identity agents.

We explore an approach to address these issues by focusing on an IdMS where relying parties (RPs), upon receiving an identity credential, require knowledge of the user's private key as a proof of credential ownership. In other words, this ownership proof and identity credential issued by an identity provider together work as a credential, following the concept of joint authority discussed in [5]. The user's private key tied to her identity credential is generally stored on the user's device hosting an identity agent. In such an architecture, identity misuse by adversaries can succeed only when a legitimate identity owner's private key is compromised. We believe this assumption is reasonable since RPs are motivated to reliably verify a requester's identity to provide services only to legitimate users. In addition, the number of IdMSs that satisfy this assumption is growing, including the proof key mechanism in Windows CardSpace [6], Credentica's U-Prove [7], and Georgia Tech's GUIDE-ME.

Under this assumption, in this paper, we propose a solution to empower users to have enhanced awareness over their online identity use by introducing a user-centric identity-usage monitoring system [8] and enable users to balance security, privacy, and usability solely based on their own needs. Our approach includes the optional use of an inexpensive storage token, such as a USB drive, to provide additional control. The main insight is that either we have enhanced security from the user provided storage token, or a transaction that is completed on a user's behalf will be monitored by a monitoring agent chosen and trusted by a user. Furthermore, our proposed architecture does allow a user's private key to be stored in an off-line safe place, and thereby the risk of compromise of the user's private key is reduced. Revocation of potentially compromised identity agents or credentials and their recovery can be done more easily and in a timely fashion, compared to the traditional way that involves certification authorities

and identity providers. We also present an actual implementation and associated protocols and evaluate user-centricity and security against possible threats (e.g., how various threats are addressed by our scheme). We believe that our approach leads to an IdMS architecture that better achieves the goal of the "User Control and Consent" law presented in [9].

The paper is organized as follows. In Section 2, we present an overview of the GUIDE-ME system and identify potential security threats to it. In Section 3, we describe the basic idea of our approach to mitigate the effects of a compromise in a simplified setting. The prototype implementation of our system in the context of the GUIDE-ME architecture is discussed in Section 4, which is then evaluated in Section 5. We will finally discuss related work in Section 6 and conclude the paper in Section 7 with future work.

2 GUIDE-ME Overview and Security Threats

In this section, we briefly describe the high-level architecture of the GUIDE-ME system [3][4][10] as an example of a user-centric IdMS that provides a context for the techniques explored in this paper. In this system, identity agents store and manage users' identity credentials and corresponding private keys and disclose the credentials based on policies defined in advance by users. In the GUIDE-ME architecture, there are two types of identity agents. Locally-installed agents (local IdA) run on devices that are with users (e.g., smart phones and laptop PCs), and remote agents (remote IdA) reside in the network. The decision to partition the identity agent functionality between local and remote entities offers a number of benefits that are explained in [4]. The architecture also includes relying parties (RP), which are service providers. The architecture of GUIDE-ME and communications among entities are illustrated in Fig. 1.

In GUIDE-ME, an identity credential is a claim about a set of attribute values for a user and also includes some way to verify the claim. Credentials are defined in a novel way so that users can only disclose the minimal information that is required to complete a transaction. Such minimal-disclosure credentials are realized by using a Merkle Hash Tree (MHT) based implementation [10]. When verifying a credential, in addition to verifying the signature made by an

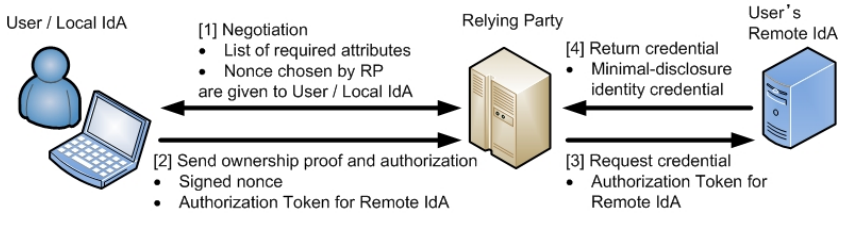

Fig. 1. Overview of GUIDE-ME Architecture

identity provider, a RP verifies a requester's credential ownership through the requester's signature on a nonce chosen by the RP (*RP Nonce*).

As introduced earlier, GUIDE-ME utilizes two types of identity agents, a local IdA and remote IdA. A local IdA on a user device stores a user's private key and meta-data which allows it to refer to identity credentials stored on a remote IdA. A local IdA also manages and checks user's identity-related policies about the disclosure of identity attributes. A remote IdA is run by a party that naturally holds certain identity credentials for a user, such as an employer or another entity that is trusted by the user. It stores users' long-term identity credentials issued by identity providers. Its primary responsibility is to manage these identity credentials and to create minimal-disclosure credentials based on authorizations from the user's local IdA.

A transaction in GUIDE-ME starts with a request from a user to a RP. The RP specifies which identity attributes it requires to provide a service (although trust negotiation may be involved, we skip it as it is out of the scope of our paper). A RP Nonce is also given to the user during the negotiation. At the user device, the local IdA creates an "Authorization Token" (AT) that tells the remote IdA to disclose specified identity attributes to the RP that is named by the user. More specifically, based on the meta-data it holds, the local IdA includes in the AT a list of identity attributes to be released, and signs it with the user's private key so that the remote IdA can verify the authenticity of the token. The local IdA sends a message including the AT and the RP Nonce to the RP. This message is signed with the user's private key so that the RP can verify the signature on the RP Nonce. The RP then forwards the AT to the user's remote IdA, requesting the user's identity credential. The remote IdA, only when the signature on the AT is valid, creates a minimal-disclosure credential and sends it to the RP. The RP finally verifies the provided credential and the user's signature on the RP Nonce and processes the request when this is successful.

In GUIDE-ME like architectures, one possible threat is the compromise of a local IdA. For instance, if a user's device hosting a local IdA is physically stolen, the adversary can use it in arbitrary transactions in order to misuse the legitimate user's identity. Although authentication may be supported by a device that runs a local IdA, security schemes based on PINs or passwords can be easily compromised. Furthermore, an infected device may allow adversaries to steal the user's private key and other data, which could lead to misuse of credentials.

Once a local IdA is compromised, the user does not have a simple and effective way to revoke its capability to interact with remote IdAs and RPs to complete identity-related transactions. Because a local IdA has access to the user's private key, the user must contact the issuing certification authority and identity provider to ask for revocation of the corresponding public key and identity credential. This process usually takes time, so the window of vulnerability might be long enough to allow the adversary to abuse the identity credential. Furthermore, in case the local IdA is compromised and the user does not recognize the problem, the situation would be even worse.

3 Approach to Handle Identity Agent Compromise

One major problem that user-centric identity management systems based on identity agents suffer from is that compromise of an identity agent allows an adversary to arbitrarily misuse identity credentials of the victim. The adversary can provide valid user signatures to complete transactions that seem to come from the legitimate user. To avoid this, it is possible to store the private key on a remote IdA, which is often better managed than user devices, and have it provide a signature for ownership verification. We could also hold the key in an external media. However, the possibility of compromise of a remote IdA or theft of an external media cannot be completely ruled out. Thus, to effectively mitigate such threats, it is necessary to eliminate the single point of attack that could give an adversary the full control of stolen identity credentials. In other words, under our assumption, keeping user's private key in an off-line safe place as long as possible is a better option. Another issue is how to deal with possibly compromised identity agents. To disable compromised agents, the victim's private key must be revoked. However, propagation of revocation information to relying parties could take a long time because such a process depends on a certification authority (CA) and identity providers. So, it is desirable that a user can revoke it without involving such entities which are not under user control. Furthermore, an IdMS should help legitimate users recognize problems when agents are compromised. To achieve this goal, we need to introduce monitoring functionality which can log identity usage and implement a scheme to detect potential identity misuse.

Based on these observations, we propose a scheme using threshold signatures [13][14], which enable us to split a user's private key into several key shares. Each key share is used to make a partial signature, also called a signature share. If the number of signature shares equals at least a pre-defined threshold, they can be combined into a signature that can be verified with the user's public key. For example, under a 2-3 threshold signature scheme, any two signature shares

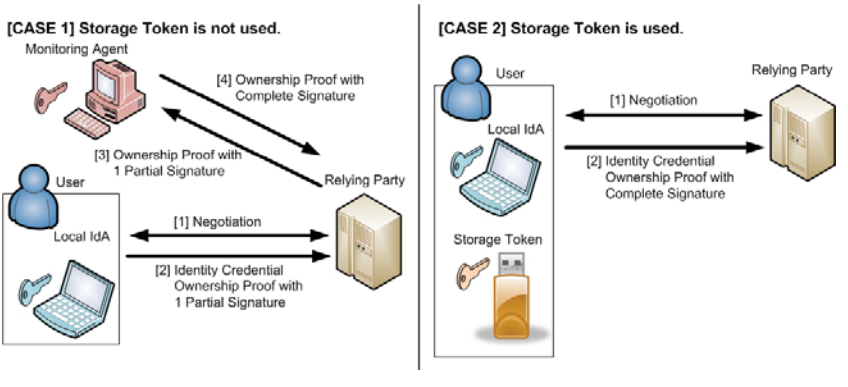

Fig. 2. Basic Idea of Our Approach Using 2-3 Threshold Signature Scheme

out of three are enough to generate a complete signature, but any single share is not sufficient to convince other parties.

Fig. 2 illustrates the basic idea of our approach in a simplified setting involving only a local IdA under 2-3 threshold signature scheme. In this setting, for the sake of simplicity, we also suppose that the user's identity credential is stored on the device where the local IdA runs. We deploy one key share on the user's device and another in a storage token, which can actually be an inexpensive USB drive or removable media. The third key share is stored at the online entity called a monitoring agent. The monitoring agent is run on a trusted third party chosen by a user or could be run on a user's private home server. Here, we use 2-3 threshold signature scheme, but the number of total key shares and threshold value can vary depending on the underlying system architecture and user needs. For instance, in an architecture utilizing both a local IdA and remote IdA, 3-4 threshold signature scheme is reasonable when an additional key share is assigned to the remote IdA. This case will be discussed later in Section 4.

As shown in Fig. 2, if the storage token is not provided by the user (CASE 1 in Fig. 2), the local IdA can create only one signature share and can send it with the identity credential. In this case, the relying party can not verify the validity of the user signature, and is then required to contact the user's monitoring agent. The monitoring agent can make another signature share and combine them into a complete signature so that the RP can verify it with the user's public key. On the other hand, if a user inserts the storage token, which contains another key share (CASE 2 in Fig. 2), the local IdA can generate two partial signatures locally which are sufficient for generating a complete signature. Then, the RP can verify the combined signature without contacting the monitoring agent.

We briefly discuss the benefits of this approach. First, since a local IdA, storage token, or monitoring agent has only one key share, none of them is a single point of attack because a complete user signature can not be forged with just one share. More importantly, revocation can be done without involving a CA or identity provider by renewing key shares when compromise of one entity is suspected. Furthermore, since the monitoring agent can be used in place of the storage token, the user can use a service even when the storage token is not available at the time of request. This property also offers another benefit which allows the user to balance usability and privacy. Using a storage token allows users to bypass the monitoring feature, but otherwise monitoring is enforced. In other words, the identity-usage monitoring feature can be flexibly turned on or off by a user. We believe that such a user-controllable monitoring mechanism minimizes user's privacy concern, which is an issue in traditional fraud detection mechanisms [8]. On the other hand, if usability is more important, a user does not have to always carry and use the storage token.

We chose to deploy a monitoring agent on a trusted third party, but there are other alternatives. It could be located with a local IdA. If a monitoring agent is running on a user's device, its functionality would be totally disabled once the device is compromised or stolen. This is a serious security concern. It is also not a good idea to place a monitoring agent with a remote IdA, even if it exists, because

of the same reason. By deploying a monitoring agent on a trusted third party, we are able to prevent misuse of identity credentials even when identity agents are compromised. It may be argued that requiring RPs to contact a monitoring agent would require changes to the RPs and may impose additional performance overhead. However, we think that our choice is justified by the observation that it ensures accurate reporting of identity usage information to a monitoring agent when the user so desires even in case identity agents are compromised. If such usage information is provided by a local IdA, because of potential compromise of it, a monitoring agent does not have an effective way to verify the accuracy of the information. On the other hand, RPs are motivated to provide correct information to avoid being manipulated by malicious users.

4 Prototype Implementation

Based on the approach discussed in Section 3, we now present a concrete design and implementation of a prototype that extends the GUIDE-ME architecture with a monitoring agent and a storage token. Our prototype is implemented in Java (J2SE), and we use Shoup's threshold signature scheme [14][15]. We demonstrated the viability of our idea by implementing and evaluating the prototype described in this section.

We also conducted response time measurement and confirmed that the additional processing overhead due to threshold signatures and additional communication is in acceptable range. For example, in our experimental setting where a separate PC is used for each entity and a user device is connected via a cable TV Internet service (13 hops away from a RP), response time measured at a user device increased on average by about 0.5 second in case a storage token was used and by 0.8 second when a monitoring agent was involved, compared to the original GUIDE-ME system. Based on the criteria explored in recent research [16], this increase in response time is tolerable for users.

4.1 System Architecture

An overview of the enhanced GUIDE-ME architecture is shown in Fig. 3. A user's master private key is stored in some off-line safe storage and does not appear in the diagram. We now use the 3-4 threshold signature scheme. Four key shares are generated and are distributed to the storage token, local IdA, remote IdA, and monitoring agent.

Although we focus on a setting in which each user has one local IdA, remote IdA, and monitoring agent, a user can have multiple agents of each type in our architecture, which is desirable in terms of system availability. When multiple agents are used, all agents of the same type are assigned the same key share. For example, when a user has multiple devices, all local IdAs have the same key share, and the total number of distinct key shares is always four. By doing so, even if more than one local IdAs belonging to a user are compromised, an adversary obtains only one key share. Thus, the system will not allow him to generate a valid signature to establish the ownership of an identity credential.

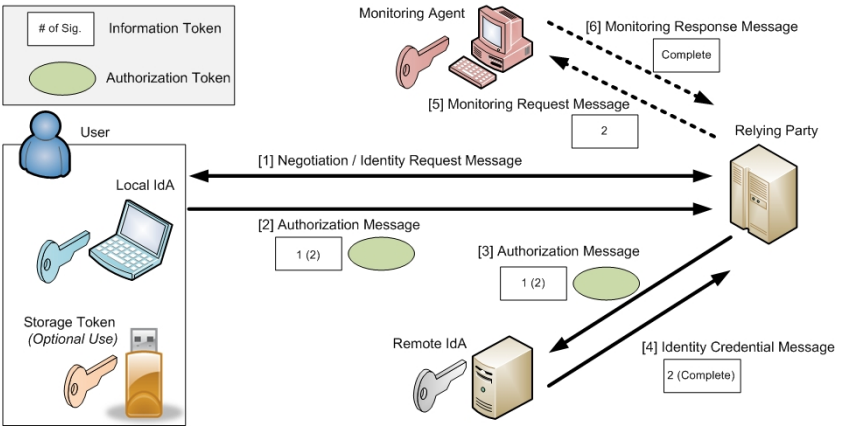

Fig. 3. Overview of Prototype Implementation

4.2 Implementation Details

We implement each entity (a local and remote IdA, monitoring agent, and RP) by a process and describe the messages exchanged among these processes. In addition, the white boxes in Fig. 3 represent "Information Token " (IT), which is described next, and the numbers in the boxes represent the numbers of partial signatures made on the corresponding tokens. "Complete" means a complete signature made from three or more partial signatures. The numbers in parentheses represent the partial signature counts when the storage token's key share is used to bypass monitoring. Although the GUIDE-ME architecture itself provides richer features, such as policy enforcement, we focus on ones related to compromised identity agent handling.

We use two key data structures that contain the necessary information which is carried by messages exchanged between the various entities. We use the term "token" to refer to them as well, but they should not be confused with the storage token that was introduced earlier. The first one, an "Authorization Token" (AT), is very similar to the one used in the basic GUIDE-ME system described in Section 2. An AT allows a user to specify which identity attributes she is willing to disclose to a RP for a certain transaction. The only difference is that an AT is signed with a local IdA's key share instead of a user's private key. The purpose of this signature is to convince a remote IdA that the AT is actually issued by the legitimate user's local IdA. Since a partial signature can be verified with the corresponding verification key just like the relationship of a private key and public key [14], the remote IdA can still verify the authenticity of the AT. We also introduce an "Information Token." The primary purpose of an IT is the verification of ownership based on the user's signature on RP Nonce. An IT may also include information about a monitoring agent (e.g., its location) when the user intends a transaction to be monitored.

Table 1. Protocol Message Description

Message Name	From	To	Description
Identity Request Message	RP	User, Local IdA	Sent at the end of the initial negotiation phase. Signed by a RP. **Contents:** List of identity claims to be released, RP's public key certificate, and RP Nonce
Authorization Message	Local IdA	Remote IdA	Sent via a RP. Signed with a local IdA's key share. **Contents:** AT and IT with one or two partial signatures
Identity Credential Message	Remote IdA	RP	Convey an identity credential. **Contents:** Minimal-disclosure identity credential and IT with two partial signatures or a complete signature
Monitoring Request Message	RP	MoA	Sent only when a user allows a transaction to be monitored, i.e. a storage token is not used. **Contents:** IT with two partial signatures
Monitoring Response Message	MoA	RP	Only sent as a response to a *Monitoring Request Message*. **Contents:** IT with a complete signature

Messages exchanged by the entities are summarized in Table 1. In the table, MoA stands for a monitoring agent. We discuss the processing of these messages by each entity next.

Local IdA. A local IdA, running on a user's device, waits for an *Identity Request Message*, which arrives when the user initiates a transaction with a RP. First, the local IdA verifies the RP's signature on the message to verify its integrity and authenticity. The identity of the RP must be carefully verified by making sure that its certificate is valid and issued by a trustworthy CA and by additionally using SSL/TLS server authentication etc. It then parses the message to obtain a RP Nonce and information about required identity attributes. Based on requested identity attributes and policies defined by the user, the local IdA allocates and initializes the AT and IT. After that, the local IdA makes partial signature on them. AT is partially signed by using local IdA's key share. For IT, when only one key share is available, the local IdA makes one partial signature on it. If two key shares, including one from the storage token, are available, the local IdA makes two partial signatures so that the RP has no reason to contact the monitoring agent. Finally, the local IdA sends an *Authorization Message* to the RP, which then forwards it to the user's remote IdA.

Remote IdA. Upon receiving an *Authorization Message* forwarded by a RP, a remote IdA first verifies partial signatures on both tokens to see if they are actually generated by the legitimate user's local IdA. After successful verification, it makes a partial signature on the IT. If the received IT already has two partial signatures, the remote IdA then combines three partial signatures, including its

own, into one complete signature. Otherwise, it just adds its own partial signature to the IT. Remote IdA's primary task is to create a minimal-disclosure identity credential [10] based on the meta-data about credentials specified in the AT. Finally, it sends an *Identity Credential Message* to the RP.

Monitoring Agent. On receiving a *Monitoring Request Message* from a RP, a monitoring agent makes its own partial signature on the IT in the message, which should already have two partial signatures, and then combines three partial signatures into one complete signature. Finally, it returns a *Monitoring Response Message* to the RP. A monitoring agent could block a transaction or raise an alarm in a real-time manner when identity misuse is suspected. Although an anomaly detection feature can be implemented, such functionality will be explored in our future work. Currently, a monitoring agent just logs the identity-usage information, such as the timestamp and the RP's identity. In addition, based on the user specified configuration, it sends the summary of usage log to the user periodically via a different and independent channel, e.g., SMS.

Relying Party (RP). A RP first receives a request for a transaction from a user. On receiving this request, it prepares a list of required identity attributes based on its policies, sends an *Identity Request Message* to the user's local IdA, and waits for an *Authorization Message*. When this message is received, the RP forwards the message to the remote IdA specified by the user, which will then respond with an *Identity Credential Message*. Upon receiving it, the RP checks the signature on the IT, and if the IT is accompanied by a complete signature, the RP verifies it by using the user's public key. Then, the RP verifies the identity provider's signature on the credential. Only when both signatures are valid, the RP accepts the identity credentials. If the IT in the *Identity Credential Message* does not have a complete signature, the RP contacts the monitoring agent specified by the user by sending a *Monitoring Request Message*. This makes the monitoring agent aware of the transaction. The information about the monitoring agent is not included when the user does not want the transaction to be monitored, and in this case, the RP has no reason to contact the monitoring agent. In response, a *Monitoring Response Message* is sent by the monitoring agent. If the IT in this message has a complete signature, the RP verifies it by using the user's public key to see whether it should accept the user's identity credential or not.

4.3 Revocation and Recovery

A user initiates a revocation process when she suspects that her device is lost or an identity agent is compromised or the monitoring agent informs her of suspicious transactions. The user can use her private key with a key share generator tool implemented by us to renew key shares. The tool distributes generated key shares to each entity. Because key shares must be protected, they are transferred via a secure and authenticated channel using the user's private key and the receiver's public key. Verification keys also need to be regenerated at the same time

and distributed to the user's remote IdA and monitoring agent. We assume that each user has at least one trustworthy computer to execute the key share generator on it so that these re-generation and re-distribution operations are securely performed. Once key shares are updated, an identity agent under the control of an adversary can no longer create a valid partial signature because its key share is outdated. This revocation process can be completed without involving the certification authority, which helps in shortening the window of vulnerability. Users can also run the key share generator periodically in a proactive manner, which is highly recommended to further improve security. In addition, recovery of compromised or disabled entities can be done by starting a new instance of the entities and re-distributing newly-generated key shares to them.

Our approach also offers a variety of options in the event that a service becomes unavailable. In case a user loses her storage token, she is still able to continue using services as described in Section 3. Because the monitoring agent must be involved, such transactions will be always monitored, which is desirable when one of the key shares has been lost. When a local IdA becomes unavailable for some reason, for example because of a hardware problem, the user can quickly create a new instance of a local IdA and continue using the service by using a key share available from her storage token in place of the local IdA's key share. This would be possible when the local IdA code can be downloaded from a trusted server and run on a new device. In this scenario, a user does not have to renew all key shares by using her private key, which is stored off-line and may not be readily accessible. The local IdA effectively uses the storage token key share until new shares are generated and distributed. Again, all transactions initiated by the user in this situation will be monitored by the monitoring agent. In this way, the monitoring agent in the architecture offers the user flexibility to monitor transactions under her control and provides necessary redundancy to complete operations when user's local IdA is unoperational or her storage token is lost.

In a more extreme scenario where the storage token and the user device are both stolen and the remote IdA or the monitoring agent is compromised as well, the user would have to revoke her private key itself by contacting the certification authority and the corresponding identity credentials by contacting identity providers. However, the likelihood of such a scenario is much smaller than a case in which only one entity is compromised.

5 Evaluation

5.1 User-Centricity

In this section, we analyze our approach in terms of properties of user-centricity for federated identity management systems proposed in [17]. Since some properties are already met by the original GUIDE-ME system, we focus on the additional properties that our approach can provide.

One major contribution of our work is the integration of an identity-usage monitoring feature in a user-centric way [8]. A monitoring agent running on

a trusted third party can log identity-usage information on behalf of the user whenever it is involved in the execution of a transaction. If a user decides that a transaction be monitored, i.e. storage token's key share is not used, the participating RP must contact the user's monitoring agent to successfully complete a transaction. In addition, the monitoring feature can be flexibly controlled by users, so it is expected to minimize users' privacy concerns. For instance, it is possible that for a transaction which could leak sensitive information (e.g., a certain prescription may indicate a medical condition), the user may decide that the monitoring agent must not be involved in the transaction. Notification feature is also implemented by a monitoring agent as mentioned in Section 4.

Another property our scheme contributes to is revocability. The GUIDE-ME architecture uses long-term identity credentials that are stored on user's identity agents. In our modified architecture, as long as the number of compromised key shares is less than the threshold, the user can revoke the compromised key shares by updating the entities with new key shares without involving the identity providers or the certification authority. Each of the key shares can be viewed as a partial privilege to use the identity credentials, and identity misuse happens only when multiple key shares are compromised under our assumption. In our architecture, such privileges of compromised entities can be revoked in a timely manner by the user.

Finally, we discuss usability, which is also one of the components of user-centricity. Our proposed solution relies on a storage token, and similar tokens are used in multi-factor authentication schemes, such as [18]. It is argued that such tokens negatively impact usability because a user may not have a token with her when she needs to access services. Thus, mandatory use of such tokens could have undesirable impact on usability. We believe that our approach offers a reasonable middle ground. If the user does not mind the monitoring agent to be aware of all the transactions initiated by her, the storage token is not required at all and the monitoring agent can serve as a network resident software token. In this case, the user's experience is exactly the same as when the storage token is not required to use a service. The important point is that there is a trade-off between usability and privacy, and users themselves can flexibly balance these based on their preferences.

5.2 Threat Analysis

We present a systematic analysis of the threats against the various entities in our architecture and how they are mitigated by the solutions we discussed. Although we primarily considered the compromise of user devices and local IdAs, we also explore the security impact when the other entities are compromised.

Compromise of User Device and Local IdA. A user device hosting a local IdA could be compromised or physically stolen by an adversary. In such a case, the adversary can have access to the key share stored on the device. By exploiting the information on the device, the adversary can try to mount various attacks. The most serious threat is that the adversary can impersonate users and

misuse their identity credentials. However, without the possession of storage token's key share, the adversary can not complete the transaction without being monitored by the monitoring agent. Even if an adversary succeeds in mounting such attacks, the monitoring agent includes functionality to report identity usage information to the user periodically, which helps the legitimate user become aware of the attack. Once the user recognizes the impersonation attack, she can immediately initiate the revocation process to disable the compromised local IdA. Thus, the compromise of the key share stored on a local IdA alone is not a critical risk.

A user device could be compromised without being detected by the user. An adversary could compromise the local IdA code or the underlying OS by means of malware or spyware. The most critical consequence of such attacks is the compromise of the storage token's key share, which could be secretly copied upon its usage, along with the local IdA's key share. Once the adversary obtains both key shares, no protection would work effectively. Although users could rely on security tools, such as anti-virus or personal firewall software, we can not completely eliminate the risk of the device being compromised. Hardware support to detect tampering [19] should be helpful, but TPM is not always available. However, even in this case, our system offers the user to make a choice based on the degree of her trust in her own device. Specifically, if the user wants to completely avoid this risk, she should never use the storage token with this device. Then, compromise of a user's device will not allow the adversary to obtain the two key shares even when the OS is compromised. Furthermore, in this case, all transactions will be monitored, which allows the user to counter this threat by giving up some privacy. If the user can partially trust her device, she can choose to use her storage token when necessary and to update all key shares periodically in a proactive fashion to minimize the risk. In this way, our scheme offers trade-off between security, usability, and privacy, and a user is able to balance these based on her own risk threshold.

Theft of Storage Token. A storage token used in our system holds one key share. Because a storage token can be lost or stolen, it is important to make sure it is not a weak point in terms of security of the system. An adversary could download local IdA code, assuming it is easily available online for the sake of convenience of legitimate users, and use it with a stolen storage token. However, in this case, the monitoring agent needs to be involved in the transaction. This will allow the user to detect the misuse. If storage token key share is tampered with or corrupted instead of being stolen, a user should be able to recognize the problem from error messages saying that construction of a complete signature failed. As a fall back, even in this case, the user can use services by involving the monitoring agent, as discussed in Section 4.3. As can be seen, a storage token used in our approach requires minimal resources and security features. Although additional security functionality, such as password protection or device-level authentication, could be used, it is not mandatory. In this sense, a storage token can be just a USB drive or removable media.

Attacks Against Monitoring Agent. The other component added by us to the architecture is a monitoring agent, and it holds a key share as well as a database that stores a log of identity-usage information. If a monitoring agent is simply disabled by an adversary, the user can notice the problem because a transaction involving the monitoring agent should return an error. In addition, if the user does not receive usage summary reports, which are supposed to be sent periodically, she can realize that something is wrong with the monitoring agent. In such cases, she can contact the trusted party that is running the monitoring agent to address this problem. A more sophisticated attack would replace the monitoring agent code by one that does not record the information about transactions that are initiated by a malicious party impersonating the legitimate user. In this case, a user has no way to become aware of the attack. Therefore, trusted parties running monitoring agents must be responsible for detecting such compromise by checking integrity of the monitoring agent code periodically, and a user should carefully choose a trustworthy party to run her monitoring agent. The compromise of a key share stored at a monitoring agent is less serious because of the 3-4 threshold signature scheme. The compromise of the database that stores accumulated identity-usage information would cause privacy concerns. Although a design of detailed mechanism is part of our future work, the data should be stored in privacy-preserving manner. The encryption of the usage database is also possible to counter this threat.

In addition to data stored at a monitoring agent, an attacker has access to the contents of an "Information Token." Thus, a compromised monitoring agent would allow an adversary to access this token's contents. Since the token only contains a partial ownership proof that is valid only for a specific transaction, RP Nonce, location information of monitoring agent, and so on, which are not confidential, disclosure of the contents does not jeopardize the system. The other type of concern related to an "Information Token" is that an adversary can replay a fully-signed token in another transaction. However, this will not work as long as a RP checks its nonce in the token which is unique for each session. If an adversary controlling the monitoring agent tries to modify the nonce, a combined signature is no longer valid because the monitoring agent's partial signature is made on data different from what is partially signed by the local IdA and remote IdA.

Compromise of Remote IdA. An adversary could target a remote IdA's key share. Although he could gain access to a user's identity credentials, this alone will not allow him to misuse the credentials, by virtue of 3-4 threshold signature scheme and joint authority [5]. Thus, a remote IdA is not a single point of attack either. Although it does not directly result in identity misuse, protection of the information included in credentials stored at a remote IdA should be ensured. This is outside the scope of this paper and will be explored in our future work.

A compromised remote IdA could allow an adversary to capture information that is sent to it by other entities. For example, an "Information Token" is included in an *Authorization Message* in Fig. 3. The adversary can obtain information included in the token. However, because it contains non-sensitive

information, it does not jeopardize the system's security. Regarding an "Authorization Token," our extensions do not add any new vulnerabilities beyond what must be addressed by the underlying GUIDE-ME architecture.

Compromise of Multiple Identity Agents. As shown earlier, the compromise of any single entity is handled by our system. Although it is less likely to happen, we do consider a case in which a user's local IdA and remote IdA are compromised at the same time. Our system can provide some mitigation of the risk even in this situation. Because we are using the 3-4 threshold signature scheme, two partial signatures are not enough to convince a RP. Thus, the monitoring agent will be contacted which will help users learn of the compromise. This is actually our primary motivation for placing a monitoring agent at a separate and trusted site.

In case a user owns multiple user devices to run local IdAs, even if the adversary succeeds in taking control of more than one local IdAs, his attempt to misuse identity credentials will not be successful. As noted in Section 4.1, the same type of identity agents are assigned the same key share. Thus, in this example, the adversary can only obtain a single key share, which is not sufficient to create a complete signature. The same holds when multiple instances of remote IdA and monitoring agent are deployed.

Malicious Relying Party. We assume that a non-malicious RP exactly follows the protocol described in Section 4. Although the security of RP is outside the control of users, we discuss the impact that a compromised or malicious RP could have on the system.

Adversaries could mount phishing attacks by spoofing a RP site. In this case, anomaly should be detected when a user initially negotiates with the RP. A user or her agent, such as a web browser or local IdA, can do it by verifying the RP's certificate and signature made by the RP. Furthermore, even if it failed for some reason, for example when a malicious RP somehow owns a valid certificate that establishes plausible credibility, a monitoring agent also can detect anomaly based on the identity, such as IP address, of a RP sending a *Monitoring Request Message* in case the user intends her transactions to be monitored.

A malicious RP might replay tokens or credentials to another (non-malicious) RP. In this case, as long as the non-malicious RP checks the nonce and a remote IdA checks the partial signature on the "Authorization Token," the malicious RP cannot impersonate legitimate users. This is because, in the protocol, a RP chooses one nonce for each transaction and requires a user to include it in tokens. Finally, it is possible that a malicious RP omits contacting a monitoring agent though it is required to do. In this case, the log kept by the monitoring agent, which is sent to a user periodically, will not include certain transactions even though the user intended them to be monitored. In this case, the user will find out that the RP is not faithfully following the protocol because of the missing transaction records.

6 Related Work

A number of federated and user-centric IdMSs have been proposed recently. Liberty Alliance's identity federation framework [20] involves three entities, identity providers, service providers, and users. The basic protocol goes as follows. When a user wants to use some service provided by a service provider, the user contacts the service provider first. Then, if the user is not authenticated by any identity providers trusted by the service provider, it redirects the user to an identity provider chosen by the user. The user authenticates herself to the identity provider and is redirected back to the service provider with an identity credential after the successful authentication process. OpenID [11] is a lightweight identity management system and has goals similar to those of Liberty Alliance. Although it was originally designed to deal with relatively simple cases, its functionality has been expanded by OpenID Attribute Exchange specification, which enables OpenID providers to transport users' profile data [21]. CardSpace [2] is a user-centric identity metasystem designed based on The Laws of Identity [9]. It provides a consistent user interface that enables users to select an appropriate identity provider for each context simply by selecting a "card." In terms of the architecture and protocol, both OpenID and CardSpace have similarity to Liberty Alliance's. Although these are getting more widely deployed, none of them implement identity-usage monitoring, which our approach offers. Thus, if an authentication credential gets compromised, there is no effective way for a user to become aware of and exercise control over identity usage. Exploring ways to integrate our approach in other IdMSs is part of our future work.

In public key setting, threshold cryptography primarily aims to share the knowledge or privilege of a private key among a number of members in order to prevent abuse of the private key as well as to make the signature made by it more reliable [22]. In addition to this objective, it can help eliminate a single point of attack, which is the case with a normal private key. Moreover, because participation of all members is not usually required, threshold cryptography can also be effective for the sake of higher system availability. Practical application of threshold cryptography is explored in online distributed certification authority area [23][24]. The authors utilize threshold cryptography scheme primarily to attain a higher level of system availability, fault-tolerance, and security. In our approach, in addition to the benefits mentioned above, threshold signature scheme is utilized to allow users to balance usability, security, and privacy demands in user-centric identity management context.

MacKenzie and Reiter [25] address the problem of securing password-protected private keys stored on a user device and revocation of such keys in case the device is compromised. They employ a network resident server and split the functionality between the user device and the online server to achieve these goals. Although their goals are similar to ours and they ensure security when a device, the online server, or the password is compromised, their scheme requires the server to be always available for public key operations. In contrast, our architecture allows users to have an option to use services even when a monitoring agent is not

available. In addition, implementing a monitoring feature, which helps users quickly notice problems, is another advantage of our system.

7 Conclusions

In this paper, by focusing on a user-centric identity management architecture involving identity agents, we presented a way to enable users to exercise more robust and flexible control over online identity usage by utilizing a low-cost storage token and an online monitoring agent. In our approach, a user can revoke potentially compromised identity agents and credentials without involving certification authorities or identity providers. In addition, our scheme ensures that user's identity usage is monitored by her monitoring agent unless the user explicitly acts to avoid it. Users also are able to determine when the storage token is used, and thereby they can balance usability, security, and privacy based only on their own needs and preferences. We also developed a concrete prototype of the proposed approach and evaluated it in terms of user-centricity and mitigation against threats. Our threat analysis showed how the theft or compromise of each entity in the system can be reasonably handled.

Our future work includes the enhancement of monitoring agent's functionality, such as designing anomaly detection algorithms based on identity-usage information and protection of accumulated usage logs. Exploring a way to protect identity attributes included in credentials from adversaries by means of cryptography is another area. We will also explore how to integrate our approach into other identity management architectures. Finally, although we based our work on identity management systems in this paper, we believe our approach is more general and can be integrated even into other types of systems. Thus, we will further explore such possibilities.

Acknowledgment

This research was supported in part by the National Science Foundation (under Grant CNS-CT-0716252) and the Institute for Information Infrastructure Protection. This material is based in part upon work supported by the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001, under the auspices of the Institute for Information Infrastructure Protection (I3P) research program. The I3P is managed by Dartmouth College. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of any of the sponsors.

References

1. Hansen, M., Berlich, P., Camenisch, J., Clauß, S., Pfitzmann, A., Waidner, M.: Privacy-Enhancing Identity Management. Information Security Technical Report (ISTR) 9(1) (2004)

2. Chappell, D., et al.: Introducing Windows CardSpace, http://msdn.microsoft.com/en-us/library/aa480189.aspx
3. Bauer, D., et al.: Video demonstration of Credential-Holding Remote Identity Agent (2007), http://users.ece.gatech.edu/gte810u/RIDA_Video
4. Ahamad, M., et al.: GUIDE-ME: Georgia Tech User Centric Identity Management Environment. In: Digital Identity Systems Workshop, New York (2007)
5. Lampson, B., et al.: Authentication in Distributed Systems: Theory and Practice. ACM Transactions on Computer Systems 10(4) (1992)
6. Microsoft and Ping Identity, A Guide to Integrating with InfoCard v1.0, http://download.microsoft.com/download/6/c/3/6c3c2ba2-e5f0-4fe3-be7f-c5dcb86af6de/infocard-guide-beta2-published.pdf
7. U-Prove Technology, http://www.credentica.com/u-prove_sdk.html
8. Mashima, D., Ahamad, M.: Towards a User-Centric Identity-Usage Monitoring System, In: Proc. of ICIMP 2008 (2008)
9. Cameron, K.: The Laws of Identity (2004), http://www.identityblog.com/
10. Bauer, D., Blough, D., Cash, D.: Minimal Information Disclosure with Efficiently Verifiable Credentials, In: Proc. of the Workshop on Digital Identity Management (2008)
11. Recordon, D., Reed, D.: OpenID 2.0: A Platform for User-Centric Identity Management. In: Proceedings of the 2nd ACM workshop on DIM (2006)
12. Shibboleth, http://shibboleth.internet2.edu
13. Desmedt, Y.G., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 307–315. Springer, Heidelberg (1990)
14. Shoup, V.: Practical threshold signatures. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 207–220. Springer, Heidelberg (2000)
15. Java Threshold Signature Package, http://sourceforge.net/projects/threshsig/
16. Akamai Technologies. Retail Web Site Performance (2006), http://www.akamai.com/4seconds
17. Bhargav-Spantzel, A., et al.: User Centricity: A Taxonomy and Open Issues. Journal of Computer Security (2007)
18. RSA SecureID, http://www.rsa.com/
19. Jaeger, T., et al.: PRIMA: Policy Reduced Integrity Measurement Architecture. In: The 11th ACM Symp. on Access Controll Models and Technologies (2006)
20. Liberty Alliance Project. Liberty Alliance ID-FF 1.2 Specifications, http://www.projectliberty.org/
21. Hardt, D., et al.: OpenID Attribute Exchange 1.0 - Final, http://openid.net/specs/openid-attribute-exchange-1_0.html
22. Desmedt, Y.: Some Recent Research Aspects of Threshold Cryptography. LNCS (1997)
23. Zhou, L., et al.: COCA: A secure distributed on-line certification authority. ACM Transaction on Computer Systems (2002)
24. Yi, S., et al.: MOCA: Mobile Certificate Authority for Wireless Ad Hoc Networks. In: The 2nd Annual PKI Research Workshop Pre-Proceedings (2003)
25. MacKenzie, P., Reiter, M.K.: Networked cryptographic devices resilient to capture. In: Proc. of IEEE Symposium on Security and Privacy (2001)