2005-03-02

# Thor: The Hybrid Online Repository

Timothy W. Van Der Horst
*Brigham Young University - Provo*

# THOR:
# THE HYBRID ONLINE REPOSITORY

by

Timothy W. van der Horst

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Brigham Young University

February 2005

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Timothy W. van der Horst

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

| | |
|---|---|
| Date | Kent E. Seamons, Chair |
| Date | Charles D. Knutson |
| Date | Dan Ventura |

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Timothy W. van der Horst in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

Kent E. Seamons
Chair, Graduate Committee

Accepted for the Department

David W. Embley
Graduate Coordinator

Accepted for the College

G. Rex Bryce
Associate Dean, College of Physical and Mathematical Sciences

ABSTRACT

THOR:
THE HYBRID ONLINE REPOSITORY

Timothy W. van der Horst

Department of Computer Science

Master of Science

Digital credentials enable users to perform secure interactions by proving either their identity or that they posses certain attributes. Special care is taken to protect these credentials and their associated private keys during transaction time. However, protection of these items outside of the transaction is often delegated to a *secure credential repository*. A mobile environment creates significant challenges for secure repositories. We examine these challenges with respect to existing repository practices and produce a set of requirements that a repository must meet in order to cope with the harshness of a mobile environment. We also present the design and implementation of Thor (The hybrid online repository), a system that fulfills these requirements. Thor leverages preexisting local and remote repositories and enhances their usability and security through centralized management, credential context subsets, and credential identifier obfuscation.

# ACKNOWLEDGMENTS

I would like to thank my wife Laura for her inspiring dedication to her studies and her ever-present support. I also thank my son Nathan for making sure that his dad took a break from writing his thesis even once and a while. Thanks go to Dr. Kent Seamons for being a very supportive advisor and employer. Finally, special thanks goes to my parents for helping me focus on the important things of life.

This research was supported by funding from the National Science Foundation under grant no. CCR-0325951 and prime cooperative agreement no. IIS-0331707, and The Regents of the University of California.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1 — Introduction

Protocols that make use of *digital credentials* employ safeguards that protect the credentials during the life of the transaction. For the most part these protocols do not, however, concern themselves with the protection of these credentials outside the context of the transaction. This responsibility is delegated to a *secure credential repository*. Many different types of repositories have been created to protect this sensitive information. A mobile environment, however, invalidates many of the fundamental assumptions for storing, accessing and protecting sensitive information in these repositories and introduces several new potential insecurities.

This research proposes a set of requirements for secure credential repositories in a mobile environment and presents the design and implementation of a repository that satisfies all of these requirements.

## 1.1 Motivating Scenario

Sid is a member of senior management. As part of his duties, Sid is required to make frequent business trips. Sid uses a variety of wireless protocols (e.g., 802.11, Bluetooth, and IrDA) in a variety of places to connect to the Internet and perform transactions. Digital credentials help Sid authenticate as a valid subscriber to pay-per-use networks and as an employee to the internal corporate network.

Sid, however, is not always connected to a communications infrastructure. Despite this lack of connectivity, e.g., during a flight, Sid continues to work diligently. He uses digital signatures to authorize work orders, purchase approvals, and to sign his business related email. These documents are transmitted when Sid reconnects to a network.

Sid also has some free time on his trips so he likes to use digital credentials of a

personal nature while on trips. He uses his credit card credential to purchase tickets to historic landmarks and events. Sid also uses other digital credentials for these non-business related electronic transactions.

Digital credentials are an essential part of Sid's life while on the road as well as when he is at home. As the number of credentials Sid possesses increases, so does the need for organization. Although effective organization provides convenience when selecting the credential to use for a specific purpose, keeping them safe is far more important. If Sid's credentials are ever stolen, a malicious person could impersonate Sid in the electronic realm.

This research answers the following two questions raised by this scenario:

1. Where can Sid store his credentials so that they are accessible while he is mobile?

2. How are they kept safe?

## 1.2 Physical Credentials

Before these questions can be answered, the term credential must be more fully defined. Physical credentials have been around for a long time. Indeed, they have become commonplace in our society. These credentials are very useful and enable many day-to-day transactions. A driver's license serves as both an assertion that one has been certified to drive and of one's identity. Credit cards assert that a person is trusted to pay for purchased goods and services within a specified period of time. An employee ID asserts that one is a member of a company. Credentials can also contain sensitive information like a credit card number, social security number, or age. In some cases even one's identity could be considered sensitive.

Credentials have two major components. The first element is the actual physical credential which asserts attributes about the possessor. The second is a verifier or authenticator through which the possessor may prove that this credential really be-

longs to him. In the context of a driver's license, a face that matches the one on the driver's license would be such a verifier. In the case of a credit card, a signature that matches the one on the back of the card would be another such verifier.

There are several problems that exist with the physical credentials that are in use today. Two of the biggest problems are that they can be forged and that their verification process is easily compromised. The problem with these verifiers is that they are available to anyone possessing the credential. A signature for a credit card can be copied from the back of the card, and plastic surgery, make-up, or mere coincidence could produce a matching face for a driver's license. These verifiers are the cause of many of the insecurities of physical credentials in the physical world, and pose an even greater problem in the digital world.

In the digital world, e-commerce and other such online transactions are becoming very popular. The need for credentials to assert attributes about both parties in a transaction is very important. Physical credentials do not perform very well in this environment as their imperfections are magnified when they are introduced to the electronic realm.

## 1.3 Digital Credentials

Digital credentials, the electronic version of their physical counterparts, play a pivotal role in the electronic realm. They are a useful component for establishing secure communication links, such as TLS [12], and allow users to prove their identity through online authentication mechanisms. In addition to establishing identity, digital credentials can also assert attributes possessed by its owner. These types of credentials are particularly valuable in protocols that establish trust between strangers in open systems, such as trust negotiation [22, 11, 23]. In plain and simple terms, these credentials enable users to perform secure interactions with their PKI peers.

Digital credentials are the electronic equivalent of their physical counterparts.

Digital credentials make use of public key cryptography to provide benefits that their physical counterparts do not posses. A digital credential is composed of two parts: a digital certificate and a private key. A digital certificate cannot be forged; a valid digital certificate can only be created by a valid issuer. The private key is the verifier of this credential. This verifier is much more effective than the ones found on physical credentials. The private key never has to be revealed to anyone, but one can prove that one has possession of it through the use of a digital signature. Digital credentials have an added insecurity since people are not as well versed in the steps that are needed to protect them as they are with physical credentials.

## 1.4 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 gives an in-depth look at a mobile environment and the requirements for a secure repository in that environment. Chapter 3 gives an overview of existing repositories and how they measure up to the requirements of Chapter 2. Chapter 4 defines a hybrid repository and explains how it satisfies the requirements of Chapter 2. Chapters 5-7 present the design and implementation of Thor, a hybrid repository. Chapter 8 enumerates future work for this research and Chapter 9 gives the conclusions that follow from this work.

# Chapter 2 — Requirements for Secure Repositories in a Mobile Environment

This research is concerned with keeping credentials safe in a mobile environment. Secure credential repositories are an ideal candidate for this task due to their effectiveness in the non-mobile environment. However, before these existing repositories can be examined in the context of a mobile environment it is necessary to first examine a mobile environment and how its underlying assumptions differ from those of a non-mobile environment.

In this chapter a mobile environment is presented and its assumptions are elucidated and compared to those of a non-mobile environment. As mentioned in Chapter 1, a mobile environment changes many of the assumptions upon which repositories are built. To compensate for these differences in their underlying foundations this chapter presents a set of requirements for a secure repository in a mobile environment. This chapter also delineates the metrics which will be used to determine whether these requirements are met.

## 2.1 A Mobile Environment

Cell phones, PDAs, laptops, and many other mobile devices are rapidly permeating the landscape of personal and business computing. Many of these devices make use of wireless communication protocols to stay connected to a communications infrastructure, even while on the move. Though ability to use wireless communications is a valuable trait of many of these devices, even without it they are very beneficial. These devices automate repetitive tasks as well as store, search, manage, and manipulate a variety of information on behalf of their owners. This information ranges from simple items like an address book or database to more complicated information like

pictures and audio or video files. People also use portable devices to electronically purchase items and perform other electronic transactions.

Although the "coolness" of these gadgets plays an important role in the purchase and use of these devices by individual consumers, they also appeal to many businesses because they have the potential to increase the productivity of their employees. This impact is particularly relevant to employees who frequently travel. In the ideal situation an employee can be just as productive while away on travel as he is in his office. The wireless communications abilities available to these devices enhance these benefits. When connected, the traveling business person can update the information and instructions needed for the trip.

Due to the transient nature of mobile devices, they often operate in a myriad of domains outside of their trusted sphere. As such, there is a need for the mobile user to receive assurances that strangers can be trusted, as well as to prove that he can be trusted by those strangers. This need becomes particularly compelling due to the widespread use of these devices and their role of representing and acting on behalf of their owners in an electronic context. Digital credentials are a powerful tool in this regard. Once their safety can be assured, these credentials greatly assist the mobile user.

In order to be effective, a repository must operate within the constraints of a mobile environment. Although a mobile environment shares many similarities with its static cousin, there are still significant differences. These differences fall into the following four categories:

- Physical security
- Computational constraints
- Connectivity

- Usability

In the following sections, the requirements for a secure credential repository in a mobile environment are specified. For each requirement that is established one or more metrics will be given to judge their satisfaction. The metrics are phrased in the form of a question. An affirmative response to the question posed by the metric indicates compliance with the requirement.

## 2.2 Physical Security of the Mobile Device

Mobile devices are generally smaller than non-mobile ones. This difference in size promotes portability and is part of the allure of these devices. It is also one of the greatest motivating factors in the design of these devices. Trade-offs are made between battery life, computational power, weight and form factor (see Section 2.3). For the most part non-mobile devices do not have the same concerns. They can rely on a constant source of energy, and smaller is not necessary better.

The size of mobile devices also increases their risk for physical security problems. For instance, mobile devices can be very easily misplaced or stolen. A misplaced device in an office or home is more of a nuisance than a security risk. However, if the device is misplaced or left behind in a public place or while traveling this becomes much more of a problem. Often times, a device that was believed to have been misplaced may have indeed been stolen. Mobile devices are very susceptible to theft because of their small size. Another physical threat to mobile devices is the fact that they can be easily broken. Whether they are dropped, thrown, or smashed, the destruction of the device affects the availability of the information contained therein.

In other words, mobile devices are prone to theft, accidental loss, and destruction. Each of these factors leads to what this research will henceforth refer to as a *loss of the mobile device*. A loss of the mobile device suggests that either the entire device or its contents has been lost, stolen, or broken beyond repair. Concern for

this situation illustrates the first requirement for a secure credential repository in a mobile environment:

**R1** The loss of the mobile device must not equate to a loss of any of a user's credentials.

A simple metric is used to determine whether this requirement is met:

**M1** In the absence of the mobile device, can the contents of the repository be restored without reissuing all the credentials?

There are various methods that a repository can use to satisfy this metric and fulfill this requirement. One of the simplest methods is to create an off-device backup. Ideally, this backup resides in a different physical location than the user. This helps to ensure that the backup is not subject to the same threat that affected the mobile device. Although an off-device backup is useful for the accidental loss or destruction of the device, the restoration and continued use of the the restored credential should adhere to the following cautionary note.

**Cautionary Note**

An off-device backup should only be restored if the user can be assured that the credentials cannot be recovered from the mobile device in question. For example, the accidental destruction of the device or the contents of its memory is a good candidate for the use of the backup. However, the theft or loss of the device in a public area is a poor candidate because there exists the possibility that the contents of the repository may be recovered by a malicious party. If recovered, an impersonation of the user by the malicious party may be possible. In this case the user's credentials and keys should be revoked and new ones issued. This, of course, applies only to the credentials that were actually contained in the repository on the device. The user's credentials

that were not stored on the device in question should not be affected by the loss of the device. A password change for the repository should occur in this case.

## 2.3 Computation Constraints of Mobile Devices

A key difference between mobile and non-mobile devices is their computational and power constraints. Even devices in the mobile arena can vary greatly in their capabilities. On one side of the spectrum are laptop-class devices with large batteries, powerful processors, and long-range wireless transmitters. These devices can make use of existing repository technologies with little or no modification in terms of computational restrictions.

Sensor network motes are on the other end of the mobile device capability spectrum. These devices have small batteries and minimal processor and transmitter power. These restrictions will not disappear in the near future. As technology improves, motes are more likely to maintain there computational resources and become smaller in size, rather than maintain their size and increase their computational power. Although sensor motes represent an extreme end of the capability spectrum, methods developed for sensor motes could lead to better protocols for devices in the middle of this spectrum.

One approach to dealing with the limited capabilities of mobile devices is to offload or share the computational load with a trusted remote agent or proxy. Base stations are a key element of a sensor network. They can serve as a data collection point as well as a management system for the entire network. The use of out-of-band resources makes more sophisticated repositories accessible to devices where they were previously unavailable.

The computational limitations that exist in mobile devices illustrate the second requirement:

**R2** The repository must function within the computational restrictions of the mobile

device.

Satisfaction of this requirement is based on a two-part metric:

**M2a** Can the repository be loaded and executed on the device?

**M2b** If it can be executed, does the response time of the repository exceed an acceptable threshold?

The ability to load and execute on a device is a good indicator that a repository may meet this requirement. However, being able to execute is only part of the story. If the repository requires extended periods of computation this will reduce battery life and response time of the device. Both of these factors contribute greatly to user frustration.

## 2.4 Connectivity of Mobile Devices

Due to the transient nature of devices in this environment, they experience various levels of connectivity at transaction time. Figure 2.1 illustrates three topologies that effectively categorize the connectivity of a mobile environment: *reliably-connected*, *unreliably-connected* (also called *intermittently-connected*), and *disconnected* access to a wired infrastructure.

*Reliably-connected* describes situations in which a user's device has reliable and adequate connection bandwidth to a wired infrastructure. The next, *intermittently-connected*, describes any situation in which a user's device has sporadic connections with sufficient bandwidth. The connectedness the device in this topology may not be in control of the user. The final categorization, *disconnected*, depicts situations in which the device has no access to a wired infrastructure.

Although there is a wide range in the types of connections available to the reliably-connect and intermittently-connected topologies the common thread that binds them

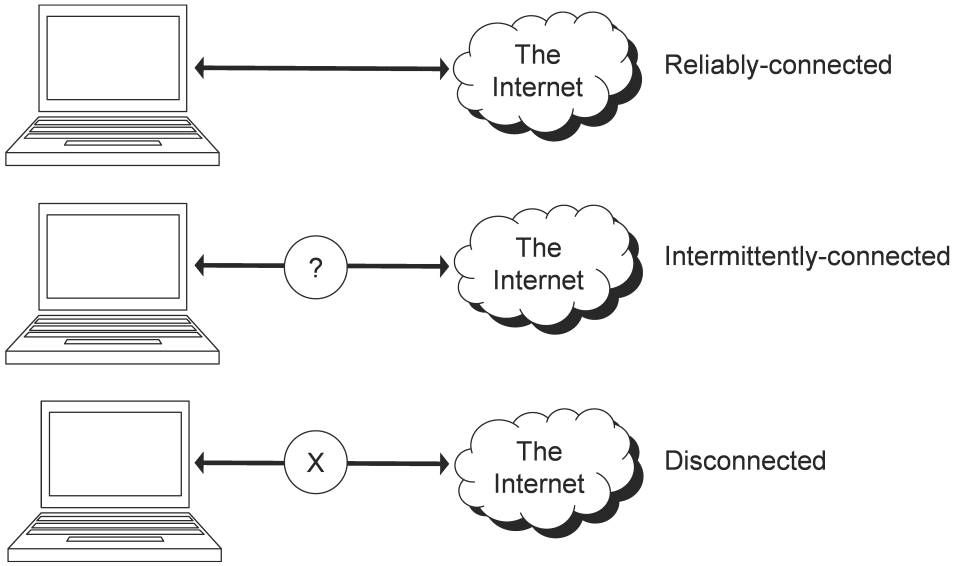


Figure 2.1: The three communication topologies of a mobile environment.

is that communication is possible. The bandwidth and latency of that connection will affect the operational time requirements for a repository that requires external communication to access credentials at transaction time, but it will still be able to provide credential availability. This is not the case in the disconnected topology where the credentials in such a repository are not available. In order to operate effectively in a mobile environment a user must be able to access his repository independent of the topology in which his device currently resides. This elucidates the third requirement:

**R3** The credentials in the repository must be available, regardless of the current topology of the mobile device.

The metric for determining whether a repository meets this requirement is simple:

**M3** Are the credentials in the repository available at transaction time without a connection to off-device resources?

If so, then the repository satisfies this requirement. This requirement is intentionally vague in stating whether or not it requires that ALL the credentials in the repository

be available, regardless of the current topology. A user or organization may wish to specify that certain credentials should never be available from a mobile device or while that device is disconnected. Also, due to storage limitations on the device it may not be possible for a user to have all his credentials stored in such a way that they are available in the disconnected state. For the purposes of this research, the availability of a user-defined subset of credentials, regardless of the current topology, is sufficient to satisfy this requirement.

## 2.5 Usability

Usability is a key feature to any system that requires user interaction. This is especially true for mobile devices since they usually rely on more ambiguous methods of input than the traditional keyboard and mouse, such as character and gesture recognition. The requirement that a repository be "usable", though essential to its acceptance, is subject to many different measurement metrics. Therefore, rather than require such a condition, this section identifies two requirements that would increase usability of the repository.

### 2.5.1 Centralized Management

It is possible for a typical mobile user to possess more than one mobile device and possibly several non-mobile devices. Even if a separate repository were maintained on each of these devices, it would be very helpful if all of these repositories were maintained by a centralized management tool. Credential management includes the ability to add, remove, and/or modify the credentials in the repository. This tool is particularly important as the number of credentials for a user increases. This tool can also assist the user by ensuring that all copies of a user's credentials are up-to-date, performing revocation checks on the user's credential chains, and reminding a user when credentials are about to expire. The automation of these features is very helpful for the average user because it automates important tasks that a user may not know

how to do, does not know that he should do, or does not want to be bothered with. Although there exist a myriad of features that this tool can have, the most important ability from a usability perspective is:

**R4** The repository must have an interface through which users may manage all their credentials. Changes made here can then be propagated to all of the users' participating devices.

The metric to verify the fulfillment of this requirement is:

**M4** Can the user manage all his credentials in a single location and have the repository propagate those changes to participating devices?

This requirement gives the user the ability to maintain all his credentials from a single location and have those effects propagated to all his devices. This usability requirement has close ties to Requirement **R1**, as the off-device backup that can satisfy this requirement is an ideal place to create and use such a tool.

### 2.5.2 Fine-Grained Synchronization

The previous requirement specifies that changes to a user's repository are to be propagated to participating devices. There exist many instances in which a user would not want the entire contents of his credential repository propagated to a particular mobile device. For example, the storage capacity of that device might be insufficient to contain the entire contents of a user's repository or a particular subset of credentials will never be used on that device by the user. Also, it is a common businesses practice to prohibit the storage of certain types of sensitive information on mobile devices. A third consideration is that the synchronization should take the minimal amount of communication and processing time on the part of the mobile device, in order to conserve the limited resources of the device.

The manner of repository replication and synchronization is important to devices in a mobile environment. A *repository-level* of synchronization, replacing the entire repository contents when only a portion of the repository has changed, is unacceptable. If a single credential is modified, those changes, and only those changes, should be propagated to the appropriate devices. This is called a *credential-level* synchronization.

The need for fine-grained synchronization necessitates the final requirement for a repository in a mobile environment:

**R5** The repository synchronization mechanism must include a granularity that permits synchronization at the credential-level.

The metric to verify the inclusion of credential-level synchronization is:

**M5** If a single credential is modified, is that change the only information needed to update the appropriate repositories?

## 2.6 Summary of Requirements

A summary of the requirements for a secure credential repository is shown in Table 2.1.

| Requirements | | Metrics | |
|---|---|---|---|
| **R1** | The loss of the mobile device must not equate to a loss of any of a user's credentials. | **M1** | In the absence of the mobile device, can the contents of the repository be restored without reissuing all the credentials? |
| **R2** | The repository must function within the computational restrictions of the mobile device. | **M2a** | Can the repository be loaded and executed on the device? |
| | | **M2b** | If it can be executed, does the response time of the repository exceed an acceptable threshold? |
| **R3** | The credentials in the repository must be available, regardless of the current topology of the mobile device. | **M3** | Are the credentials in the repository available at transaction time without a connection to off-device resources? |
| **R4** | The repository must have an interface through which users may manage all their credentials. Changes made here can then be propagated to all of the users' participating devices. | **M4** | Can the user manage all his credentials in a single location and have the repository propagate those changes to participating devices? |
| **R5** | The repository synchronization mechanism must include a granularity that permits synchronization at the credential-level. | **M5** | If a single credential is modified, is that change the only information needed to update the appropriate repositories? |

Table 2.1: Summary of the requirements for a secure credential repository in a mobile environment.

# Chapter 3 — Secure Repositories

A secure credential repository is a place where credentials are kept safe when not in use by a protocol. There exist a wide variety of secure credential repositories. Secure repositories vary greatly in their design, benefits, and shortcomings. Typically each application or protocol maintains its own private repository. In this chapter, a sampling of these repositories are presented. Repositories fall into two general categories: remote and local. Basney et al.[10] and Gupta et al.[15] provide good introductory overviews of existing secure repositories. This chapter analyzes the effectiveness of the two types of repositories in a mobile environment based on the requirements presented in Chapter 2.

## 3.1 Remote Repositories

The physical analogue of a remote repository is a safe deposit box with physical credentials stored inside. This box is stored either in a personal safe or in the safe of a trusted third party. In either case, its location tends to be static. In the case where the box is guarded by a trusted third party, the contents are either kept a secret from its guardian or permission is delegated to that party to access the contents on behalf of the user.

A electronic remote repository resides on a remote device or server and provides a central location for a user to store and manage his credentials. The remote repository is administered by a user on a machine of his choosing or he delegates that responsibility to a trusted third party to host it on his behalf. There are basically two different types of remote repositories. The difference between these two types resides in the amount of knowledge that the repository maintains regarding the contents of a user's credentials. The belief here is that the more a repository knows, the more it can help

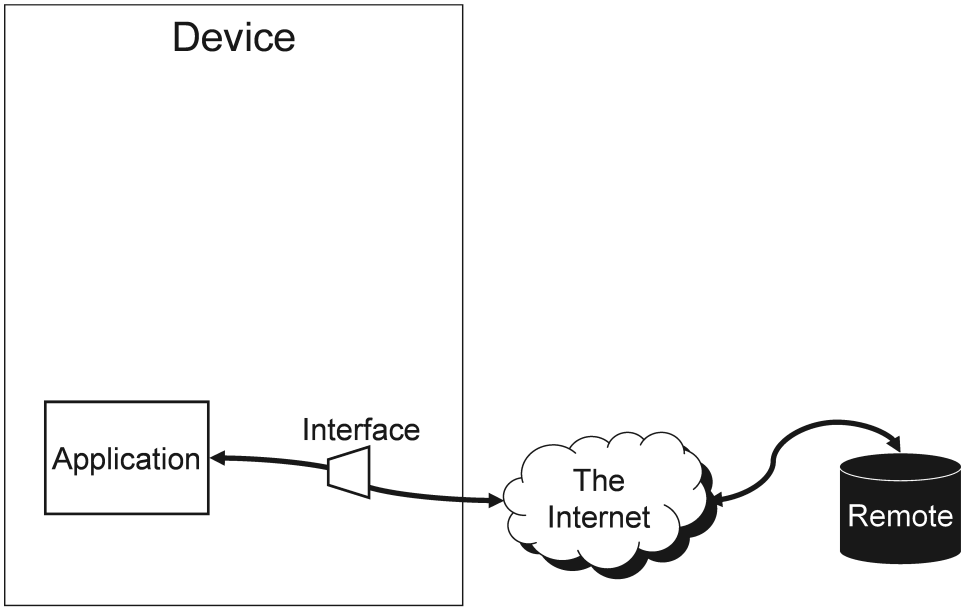


Figure 3.1: A typical remote repository. The repository is off-device and is accessible to an application via an interface and some communications infrastructure, e.g., the Internet.

the device, both in terms of security and computation. Simply put, if the server has knowledge of the sensitive information then it should be able to assist the client in the use of that information. This could be very valuable depending on the limitations of the mobile device. This method does raise concerns about insider attacks and claims of non-repudiation. On the other hand, if the repository has no knowledge of the contents of a user's credentials (they are encrypted so that the repository has to treat them as opaque objects), then the security of the credentials resides solely in the hands of the user. Access to this repository is granted based on an authenticator. This authenticator is a password, biometric, or some other pre-established relationship the device itself has with the repository.

### 3.1.1 Virtual Smart Cards

Sandhu et al. [21] defines two types of remote repositories: virtual soft tokens and virtual smart cards. In the virtual smart card paradigm, the repository is always

involved in the use of the private key. This is accomplished by one of two methods. First, the repository has access to the entire private key and can perform signatures on behalf of the client, just like a physical smart card. Second, the repository uses the 3-key RSA algorithm to accomplish a joint signature with the client. Not all keys, however, can be converted to the 3-key format and this limits the migration of existing certificates to this system. NSD Security's Practical PKI [3] is an example of a virtual smart card that uses the 3-key algorithm.

### 3.1.2 Virtual Soft Tokens

Virtual soft tokens are a network-based storage solution of sensitive credentials. In this system the server is oblivious to the actual information that is being stored. This approach does not have the added benefit of server assistance with the use of the information, but it does greatly simplify the requirements of the server and greatly reduces the risk of a malicious server or insider gaining access to a user's sensitive credentials. An online server stores the credentials and are only retrievable by an authorized and authenticated client. This type of repository may store encrypted items such that a user must provide a decryption key for each item, or it may choose to release the item in an unencrypted form to the user. MyProxy [19] is an example of the latter, while Securely Available Credentials (SACRED) [9, 16, 13], Versign Roaming Service[7], and beTRUSTed's UniCERT Roaming Profile [1] are examples of the former. Versign and beTRUSTed both use applets that emulate a smart card to retrieve and use credentials and keys. SACRED retrieves the encrypted credentials and requires user interaction to decrypt them. A user is, therefore, completely responsible for the security of the unencrypted data because he is the only one with access to it.

Ideally, the credentials retrieved from a remote repository are cleared from the device when the transaction that required them has completed, thus preventing undue

exposure of the sensitive credentials and keys while on the device. To enhance the security of the information in the system, the password that is used to decrypt the credentials should be different than the one that is used to authenticate the client to the repository. This will ensure that the client does not even tempt the server to access the sensitive data.

### 3.1.3 Advantages

A remote repository has several benefits. First, it is always available to the client (provided the network where the server resides is accessible to the client). A user is not required to transport his repository with him and it provides a centralized location to manage credential updates. The remote repository can also be hosted by a third party, trusted to have a secure system and regular backups of a user's information. Problems can arise when the repository is in the possession of a third party because a malicious insider could erase a user's data. This should be covered by a legal obligation between the client and the server. Backups by the server help to alleviate this danger. One of the greatest benefits of using a remote repository in conjunction with a mobile device is that if the mobile device is lost, or stolen, the repository remains safe on the remote server.

### 3.1.4 Disadvantages

There are several disadvantages which plague remote repositories. They must be available at transaction time and thus create a dependence on a third party in order to complete the transaction. If the online repository is not accessible from where the mobile device is located, it is useless. An online repository creates an additional communication overhead: each time a transaction requires credentials, the mobile device must interact with it. An online remote repository also creates a highly attractive target for attack.

### 3.1.5 Does a Remote Repository Meet the Requirements of a Mobile Environment?

In terms of the first three requirements for a secure credential repository in a mobile environment, remote repositories fare quite well. By their nature, this type of repository provides an off-device backup that is accessible even if the entire mobile device is destroyed, thus satisfying Requirement **R1**. Some remote repositories, though not all, also have the capacity to offload or share computational tasks with the mobile device. This is definitely a good indicator for the satisfaction of Requirement **R2**. Also, when dealing with a remote repository it is important to note that only the client side software must satisfy requirement **R2**.

In terms of usability, remote repositories score very well. By design, all of a user's mobile devices access the same repository. That repository serves as the central management tool. Because there is a single repository there is no need for the synchronization between other repositories, other than for backup purposes. Remote repositories satisfy both usability requirements, **R4** and **R5**.

Unfortunately, remote repositories require a connection to an online component at transaction time. This is a violation of Requirement **R3**.

A pure remote repository solution, though able to provide several advantageous features, is insufficient to meet the needs of a repository in a mobile environment.

Table 3.1 shows a summary of how a remote repository measures up to the requirements for a secure credential repository in a mobile environment.

## 3.2 Local Repositories

The most common type of repository is the local repository. This repository is collocated on a user's device with the rest of his applications. The physical counterpart of a local repository is the wallet. The physical credentials in the wallet are safe while the wallet is physically secure. A wallet can also be stolen, lost, or destroyed.

| Requirements | | Metrics | | |
|---|---|---|---|---|
| **R1** | The loss of the mobile device must not equate to a loss of any of a user's credentials. | **M1** | In the absence of the mobile device, can the contents of the repository be restored without reissuing all the credentials? | YES |
| **R2** | The repository must function within the computational restrictions of the mobile device. | **M2a** | Can the repository be loaded and executed on the device? | YES |
| | | **M2b** | If it can be executed, does the response time of the repository exceed an acceptable threshold? | YES |
| **R3** | The credentials in the repository must be available, regardless of the current topology of the mobile device. | **M3** | Are the credentials in the repository available at transaction time without a connection to off-device resources? | NO |
| **R4** | The repository must have an interface through which users may manage all their credentials. Changes made here can then be propagated to all of the users' participating devices. | **M4** | Can the user manage all his credentials in a single location and have the repository propagate those changes to participating devices? | YES |
| **R5** | The repository synchronization mechanism must include a granularity that permits synchronization at the credential-level. | **M5** | If a single credential is modified, is that change the only information needed to update the appropriate repositories? | YES |

Table 3.1: Summary the fulfillment of the requirements for a secure credential repository a mobile environment by a remote repository.

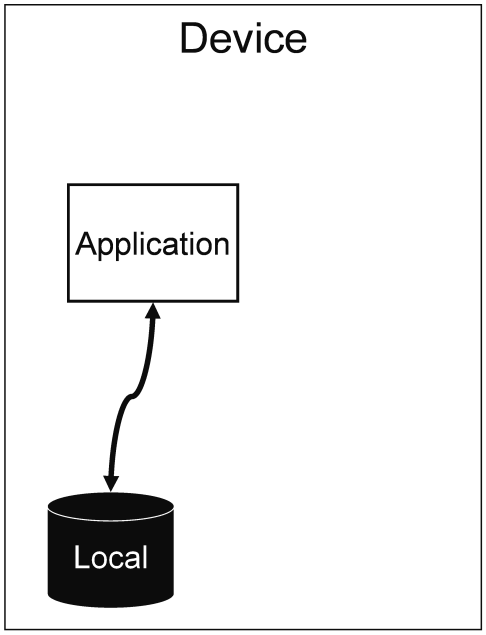


Figure 3.2: A typical local repository. The repository is on-device and is accessible directly by the application.

However, unlike a wallet the local repository can be "locked" so even if it is lost or stolen its contents will not be easily accessed.

A local repository is usually some sort of database, file, or set of files that is encrypted in order to restrict access to an authorized user. Access to this repository is granted based on an authenticator. This authenticator is a password, biometric, etc. A local repository may store encrypted items such that a user must provide a decryption key for each item, or it may release the item in an unencrypted form to the user. The latter requires that a user only have one password for complete access, while the former requires a user to supply a password for the repository and for each item stored therein. As the number of items stored in the repository increases so does the number of passwords that a user must remember. This leads to several problems (see Section 6.2). There are many standards in place for local repositories such as PKCS#8, a standard format for storing a private key, and PKCS#12 and #15 which

are used to store more generic forms of sensitive information. The Java KeyStore, an encrypted database, capable of storing a variety of items, is also representative of a local repository.

### 3.2.1 Secure Modules

Another type of local repository stores the sensitive data in an encrypted form on a secure module that is attached to a user's device. An example of this is Sony's Memory Stick. Through the use of MagicGate [8], a Memory Stick can store its contents in an encrypted form and release them only to a user that successfully authenticates himself. Another secure module that is used with mobile devices is a cryptographic smart card. Smart cards have several advantages over other local repositories. Like other repositories, access to the card is protected by a password or biometric. The private keys never leave the card since all necessary processing can be done within the card. Unfortunately, the space available on these cards is very limited. There is normally about 32KB of space for both an application and its data.

### 3.2.2 Advantages

Local repositories have several advantages that make them very well suited for the protection of sensitive electronic information. First and foremost is the fact that they reside locally on a user's machine. The credential repository, therefore, remains in the physical protection of the user. The locality of the repository also lends itself well to matters of latency. The repository is on the user's device and therefore access should be quite quick compared to a remote solution that requires the overhead of the establishing and communicating over an encrypted channel.

### 3.2.3 Disadvantages

Several disadvantages impact the effectiveness of local repositories. First and foremost the local repository must be present with a user to be of any use. Also, the loss of the device is equivalent to the loss of the repository and, without an off-device

backup, the contents of the repository must be reacquired from their respective issuers. Loss of the local repository to a malicious entity also makes the repository subject to an off-line brute-force attack. This attack may not yield anything within the lifetime of a user if a sufficiently strong password is selected, however if a weak password was chosen by a user, then such an attack could yield very fruitful results. Enforcement of a strong password policy is essential to the security of a user's information.

Another problem is synchronization. If a user has several devices, he has to replicate his sensitive information on every device. When a credential expires, is revoked, or for any reason needs to be updated or removed, the changes must be propagated to every one of a user's devices. This could be a costly and time-consuming process, due to the lack of centralized management. Also, the security of a local repository can vary from device to device.

### 3.2.4 Does a Local Repository Meet the Requirements of a Mobile Environment?

In terms of computational resources, local repositories tend to be simple, yet secure. The use of an attached secure module also assists the mobile device with repository related computation. Local repositories easily satisfy Requirement **R2**.

Since a repository of this type either resides on the device itself or in an attached module, all necessary communication is accomplished independently of the current communications topology of the device. This feature provides guaranteed satisfaction of **R3**.

Unfortunately, because the repository contents reside entirely on the local device, the loss of the device equates to the loss of all of a user's credentials. This is a violation of Requirement **R1**. This violation can have exceptions. For example, it could be argued that an off-device backup could be created of the local repository, thus fulfilling this requirement. Though this is possible, it is the exception, rather

than the rule, that such a backup would be part of the normal operation of a local repository. (This idea will be explored in greater detail below when Requirements **R4** and **R5** are discussed.)

It could also be argued that the use of a secure module should be considered an off-device backup. Although this is conceptually valid, in practice the secure module is usually located in close proximity to the mobile device, if not always attached. Although the repository is conceptually "off the device" it will most likely suffer the same fate as the device itself. Even if nothing bad were to happen the device, the secure module has the same chances of being lost, stolen, or broken as does the mobile device.

In terms of the usability requirements, local repositories do not fare very well, since they are generally self-contained units. For the most part, they do not even have the ability to communicate with other repositories. Synchronization is usually performed in local repositories by performing full copies of its contents. Local repositories offer no centralized management nor fine-grained synchronization and therefore do not satisfy Requirements **R4** and **R5**.

A pure local repository solution, though able to provide several advantageous features, is insufficient to meet the needs of a repository in a mobile environment.

Table 3.2 shows a summary of how a local repository measures up to the requirements for a secure credential repository in a mobile environment.

## 3.3 Summary of Existing Repositories

Both types of current repository solutions have been explored and compared to the requirements for a secure credential repository. Neither of these two types adequately meet the requirements that have been established in Chapter 2. Figure 3.3 shows how these two types of repositories cover the requirement space.

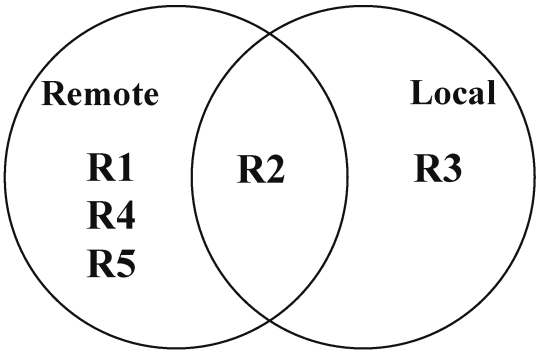


Figure 3.3: The requirement space contains five different requirements. The remote repository covers all of the requirements except **R3**. The local repository covers Requirements **R2** and **R3**.

| Requirements | | Metrics | | |
|---|---|---|---|---|
| **R1** | The loss of the mobile device must not equate to a loss of any of a user's credentials. | **M1** | In the absence of the mobile device, can the contents of the repository be restored without reissuing all the credentials? | NO |
| **R2** | The repository must function within the computational restrictions of the mobile device. | **M2a** | Can the repository be loaded and executed on the device? | YES |
| | | **M2b** | If it can be executed, does the response time of the repository exceed an acceptable threshold? | YES |
| **R3** | The credentials in the repository must be available, regardless of the current topology of the mobile device. | **M3** | Are the credentials in the repository available at transaction time without a connection to off-device resources? | NO |
| **R4** | The repository must have an interface through which users may manage all their credentials. Changes made here can then be propagated to all of the users' participating devices. | **M4** | Can the user manage all his credentials in a single location and have the repository propagate those changes to participating devices? | NO |
| **R5** | The repository synchronization mechanism must include a granularity that permits synchronization at the credential-level. | **M5** | If a single credential is modified, is that change the only information needed to update the appropriate repositories? | NO |

Table 3.2: Summary the fulfillment of the requirements for a secure credential repository a mobile environment by a local repository.

# Chapter 4 — Hybrid Repository

Both local and remote repositories have their benefits and drawbacks. Local repositories have very little communication overhead and do not require access to an online server at the time of the transaction. However, they also require that a user bring the repository with him, and the propagation of updates in this model can become complicated. Remote repositories, on the other hand, always have up-to-date credentials and allow a user to access those credentials from any device. Since the device contains no sensitive credentials, when the device is lost, nothing but the device is lost. Unfortunately, the communication overheard, accessibility, and availability issues can limit the effectiveness of online repositories in a mobile environment.

Due to these limitations, neither a pure local nor remote repository meets all of the usage requirements of a mobile environment. This suggests the creation of a new type of repository, one better suited to the demands of this environment. This new type of repository must allow a user to maintain a safe copy of his sensitive information in a secure remote location, while giving him the option to create a local cached copy for situations in which the mobile device will be in a disconnected state.

This research proposes the creation of a new type of repository called the *hybrid repository*. A hybrid repository is a combination of the two existing repository types (see Figure 4.1). This union leads to the elimination of many of the drawbacks inherent in these two types of repositories (see Figure 4.2). The hybrid repository acts as a strict remote repository, a local repository (a full copy of credentials still resides in the remote repository), or a mix of the two. This configurable capability gives the user the flexibility to control the availability of his credentials. The user can also choose different local repositories depending on the restrictions of his devices.

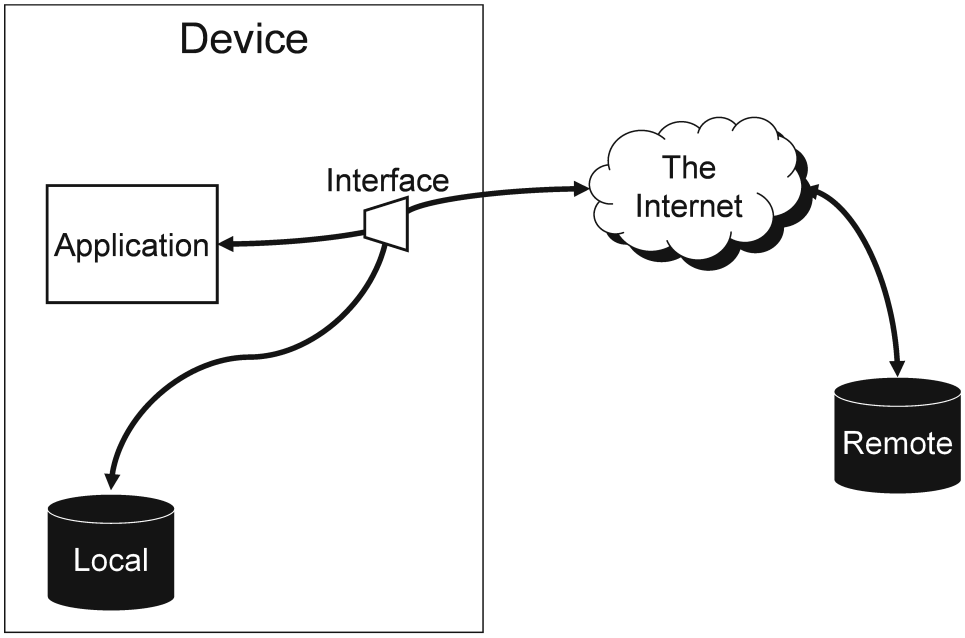


Figure 4.1: A typical hybrid repository. The repository has on-device and off-device components and is accessible by an application through an interface.

Simply stated, a hybrid repository gives the mobile user the power and flexibility to protect and use his credentials while in a mobile environment.

The physical analogies presented in the previous chapter still remain valid. A hybrid repository combines the concepts of the wallet and safe deposit box. The user stores the "original" credentials in the safe deposit box. In the case where the safe deposit box will be inaccessible for a time, or simply for convenience, the user creates a "certified copy" and carries it in his wallet. The "original" credentials always remains in the safe deposit box as a backup in case the wallet is ever lost, stolen, or destroyed.

## 4.1 Existing Repositories that Resemble a Hybrid Repository

The Entrust TruePass architecture [2] integrates many ideas of the hybrid repository. TruePass is designed to give "strong authentication, digital signatures and end-to-end encryption to the Web Portal." It allows a user to store his Digital ID as a roaming profile, in the Windows digital ID store, on a smart card, or as an encrypted

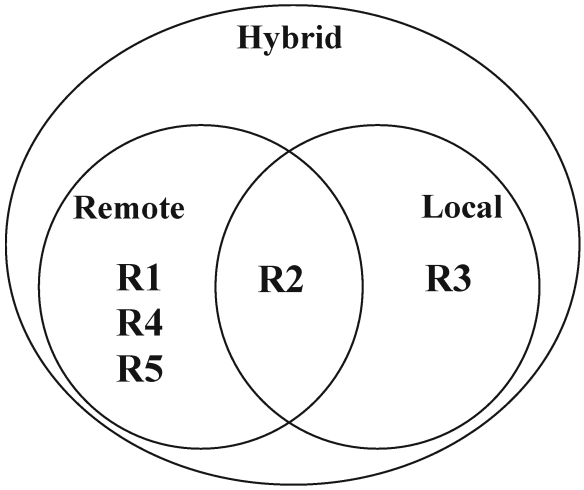


Figure 4.2: The requirement space contains five different requirements. The remote repository cover all of the requirements except **R3**. The local repository covers Requirements **R2** and **R3**. The hybrid repository covers all of the requirements.

file on a user's hard drive. This allows a user great flexibility to choose the manner in which his credentials are available, but restricts a user's options in other areas. Although the Entrust TruePass system has many attributes of a hybrid repository, it also has several aspects that may make it unappealing to mobile users. The main aspect that should dissuade the average mobile user from adopting this system is its blurring between the concepts of repository and protocol. The Entrust repository is only accessible by Entrust Applications which, in turn, only communicate with Entrust server-side modules. The user cannot make these credential available to the protocol of his choosing. This is an essential feature to a general-purpose repository because it is an unreasonable assumption to believe that every party a user wishes to communicate with is Entrust-enabled. This repository is thus unsuitable for general use as a secure credential repository in a mobile environment.

RSA Keon Web PassPort [6] also integrates the many concepts from the hybrid repository. It stores credentials on a remote server and uses a browser applet to emulate a smart card. This applet is accessible by any user application via PKCS#11

or the Microsoft Cryptographic API. Although this emulated smart card is a type of local repository, it is never written to persistent local storage. The ability to create and use persistent local storage is an important attribute of the hybrid repository because it allows accessibility in the disconnected topology (see Requirement **R3**).

## 4.2 Does a Hybrid Repository Meet the Requirements of a Mobile Environment?

A hybrid repository provides an off-device backup that is accessible even if the entire mobile device is destroyed, thus satisfying Requirement **R1**. The specific implementation of this repository greatly affects the fulfillment of Requirement **R2**, but the fact that current repository technologies function well given computational constraints of a mobile environment is a good indicator that the hybrid repository will also perform similarly.

In terms of Requirement **R3**, the remote aspect of the hybrid repository is not much help. However, the local aspect permits all necessary communication to be accomplished independently of the current communications topology of the device, thus providing satisfaction of this requirement.

In terms of usability, hybrid repositories have the potential to score very well. Through the remote aspect, a location is provided to serve as the central management point and facilitate the creation of a central management tool. The creation of such a tool, however, is implementation specific. The satisfaction of Requirements **R4** and **R5** is therefore not a guarantee for the hybrid repository, but the resources needed in order to implement the functionality are available.

Table 4.1 provides a summary of how a hybrid repository measures up to the requirements for a secure credential repository in a mobile environment.

| Requirements | | Metrics | | |
|---|---|---|---|---|
| **R1** | The loss of the mobile device must not equate to a loss of any of a user's credentials. | **M1** | In the absence of the mobile device, can the contents of the repository be restored without reissuing all the credentials? | YES |
| **R2** | The repository must function within the computational restrictions of the mobile device. | **M2a** | Can the repository be loaded and executed on the device? | YES |
| | | **M2b** | If it can be executed, does the response time of the repository exceed an acceptable threshold? | YES |
| **R3** | The credentials in the repository must be available, regardless of the current topology of the mobile device. | **M3** | Are the credentials in the repository available at transaction time without a connection to off-device resources? | YES |
| **R4** | The repository must have an interface through which users may manage all their credentials. Changes made here can then be propagated to all of the users' participating devices. | **M4** | Can the user manage all his credentials in a single location and have the repository propagate those changes to participating devices? | YES[1] |
| **R5** | The repository synchronization mechanism must include a granularity that permits synchronization at the credential-level. | **M5** | If a single credential is modified, is that change the only information needed to update the appropriate repositories? | YES[1] |

[1]The creation of a centralized management tool is not guaranteed by the hybrid repository, but the resources needed in order to implement the functionality are available.

Table 4.1: Summary the fulfillment of the requirements for a secure credential repository a mobile environment by a hybrid repository.

# Chapter 5 — Thor

The creation of a hybrid repository presents several interesting challenges. This research presents a method of dealing with these challenges by leverage existing repositories. Many effective local and remote repositories already exist, although no single one satisfies all five of the requirements for a secure repository in a mobile environment. It makes sense to leverage the strengths and benefits of existing systems by creating a way to combine different existing repositories into a single virtual repository. Such a system would allow a user to configure the underlying local and remote repositories used based on his personal situations and needs. A user also has the option to switch the underlying repositories without having to change the way that he interacts with the virtual repository. The prototype system is called Thor (The hybrid online repository). The following sections present the additional goals of this prototype, a repository interface for combining existing repositories, and the system design for Thor. Chapter 6 delineates Thor's additional management enhancements and security features and Chapter 7 explores the reference implementation of Thor.

## 5.1 Goals

Three goals govern the design and implementation of Thor:

**G1** All five requirements for a secure credential repository in a mobile environment must be satisfied.

**G2** There must be no modifications required to the existing repository implementations.

**G3** Where possible, increase usability and security of existing repositories without modifying them.

The motivation of the first goal is obvious and part of the overall goal of this research. The second goal promotes the use of existing systems, which already perform well in a non-mobile environment, eases the transition from purely local and remote repositories and facilitates the adoption of this system. The third goal not only serves to improve the safety of the credentials stored in the repository, but is also an attractive feature to promote this system's adoption and to inject new ideas into the credential repository field.

## 5.2 Repository Interface

Thor specifies an interface through which preexisting local and remote repositories can be combined. Thor uses this interface to control and interact with each repository. Thor, the local repository, and any other user applications reside on the local device, while the remote repository resides at an off-device location accessible via a wired infrastructure. Ideally, an application, authorized by the user, accesses the credentials by interfacing directly with Thor or by using the local repository directly.

This interface exploits the fact that every repository provides the same basic functionally: the ability to store and retrieve credentials. Although the actual names and parameters of these operations vary from repository to repository, all repositories expose an API that allows a user to accomplish the following three operations:

1. Put an encrypted credential in the repository with a unique identifier.

2. Get an encrypted credential from the repository based on a unique identifier.

3. Delete an encrypted credential in the repository based on a unique identifier.

In order to allow repositories to seamlessly interoperate within Thor, it is essential that an interface be created that standardizes these three operations. Figure 5.1 presents the Java-style repository interface that is specified by Thor. In order for a

```
public interface Repository {

   public void put(byte[] item, String identifier) throws RepositoryException;

   public byte[] get(String identifier) throws RepositoryException;

   public void delete(String identifier) throws RepositoryException;

}
```

Figure 5.1: This is the Java-style interface defined by Thor. This interface exploits that fact that all repositories have three basic functions: get, put, and delete.

new repository to be integrated with Thor, a wrapper that maps the methods and parameters of the Thor interface to the methods and parameters of that repository's existing interface must be created. By forcing the creation of a wrapper that implements Thor's repository interface, a level of abstraction is produced that permits Thor to use a repository independent of its actual implementation. This also allows a user to easily switch the underlying repositories used by Thor without having to change Thor itself.

Each repository remains usable without the Thor client interface module (through its traditional interface) unless Thor's password manager was used to remember repository authentication passwords (see Section 7.5). In this case the appropriate credential decryption keys must be retrieved from Thor.

Several other operations are required in order to setup and clean-up Thor's interaction with the underlying repositories. The following operations are handled during the instantiation and finalization of the interface:

1. Establish a connection to the repository.

2. Authenticate a user to the repository, and vice-versa (specify root certificates that are trusted for this repository).

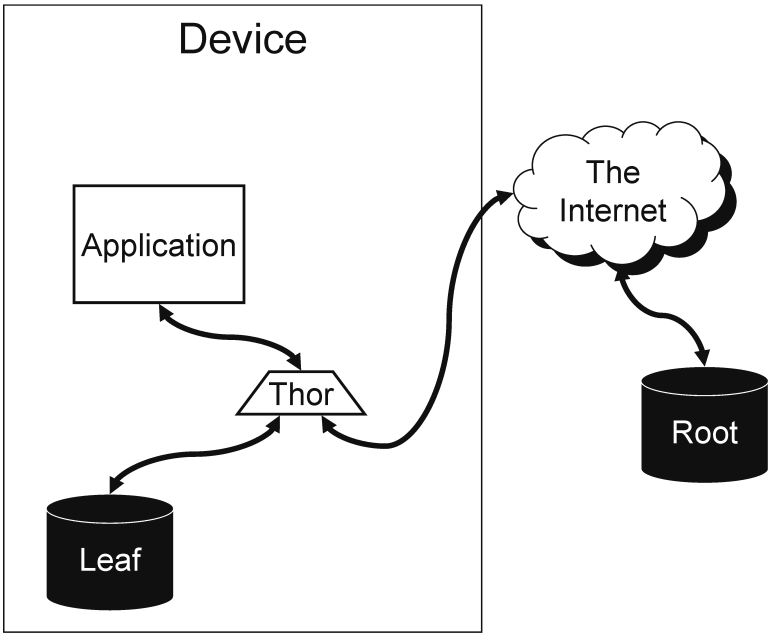


Figure 5.2: Thor's design is simple. Rather than create the new local and remote components required by a hybrid repository, Thor leverages existing repositories as its local and remote elements. The remote element is called the root repository node. The local element is called the leaf repository node.

3. Disconnect from the repository

## 5.3 Design

The overall design of Thor is intentionally simple (see Figure 5.2). As described in Section 5.2 Thor uses the repository interface to combine the repositories. The following sections describe how these repositories are organized and combined.

### 5.3.1 Repository Tree

Thor organizes its repositories into a tree structure (see Figure 5.3). The top node in the tree is called the *root repository node*. There can be only one root repository node. The other nodes in the tree structure are called *leaf repository nodes*. There is no limit on the number of leave nodes.

The root repository node provides a location for a centralized management point (see Section 5.3.3) and an off-device backup. The types of repositories that can fulfill

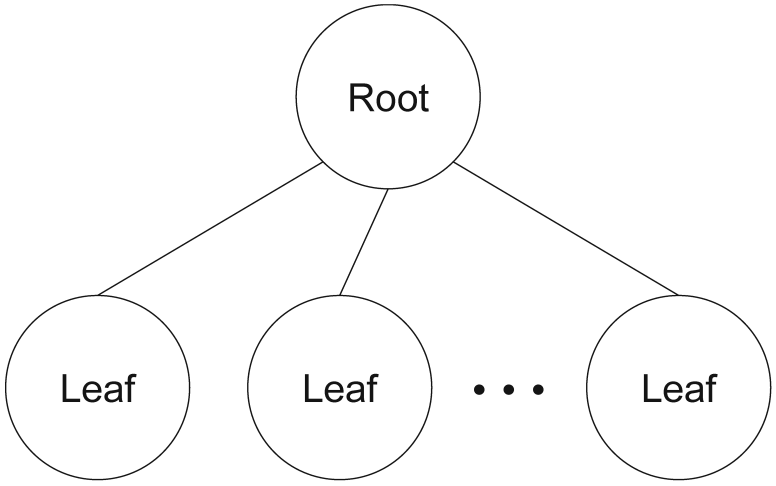


Figure 5.3: The repository tree structure used by Thor. The top node in the tree is the root repository node. Its children are called leaf repository nodes. There can be any number of leaf nodes, even on the same device, but only one root node.

this role are explored in the next section. Possible future extensions to this node include the addition of a backup root node, or a multi-node root repository that would operate in a similar fashion to RAID.

Although the use of an online remote repository creates an attractive target for attackers, it also provides a centralized and unified location to prevent and deal with those attacks.

The leaf repository node always has an associated root repository node. The addition of new credentials to the repository via a leaf node, or changes made to existing credentials, must pass through the root node to be made available to other leaves. Constraints and recommendations for leaf node repositories are given in the next section. Although there is no inter-leaf communication or synchronization ability, possible future extensions could include this ability on a limited basis.

The root repository node provides an off-device backup suitable for the satisfaction of Requirement **R1**. The presence of a leaf repository node provides a local copy of the user's credentials that are accessible without any need to communicate with the root

repository node. Although the user may choose a only a subset of all his credentials to reside in the leaf node, the guaranteed availability of that subset is sufficient for the satisfaction of Requirement **R3**.

### 5.3.2 Repository Selection Constraints

The second requirement for a secure credential repository gives the only overarching constraint on the selection of the repositories that are integrated in any particular Thor system. Requirement **R2** dictates that the repository must function within the computational constraints of a mobile environment.

Although every repository provides the same basic functionally, as described in Section 5.2, that does not mean that every repository is a suitable candidate for the root repository node. In order to effectively perform the function of the root repository node, a repository must satisfy the following constraints. First, only remote repositories can be assigned the role of a root repository node. Second, a complete copy of a user's credentials must be able to be stored in the repository. This allows a complete remote backup to be made and is also utilized as the centralized synchronization point. Third, the entire credential must be retrievable from the server. This allows a user to populate his local repository with the necessary credentials and keys for operation in the disconnected topology. Because of these last two constraints, the virtual soft token style of remote repositories is better suited than the virtual smart card style. Finally, the remote repository should treat the credentials as opaque encrypted objects. This will increase the strength of non-repudiation claims because only the client will be able to access his sensitive credentials in an unencrypted form.

Since a leaf node resides on the mobile device it makes the most sense conceptually that a leaf node be a local repository. However, a remote repository that is hosted on the local device can also produce the desired effect. Based on this idea, a leaf repository will not have any additional constraints. This research recommends that

1. The repository be a remote repository.
2. The repository must be able to store a complete copy of a user's credentials.
3. The repository must permit retrieval of the entire credential.
4. The remote repository should treat credentials as opaque objects.

Figure 5.4: Summary of the constraints of the root repository node.

a local repository be used, however the final choice lies with the preferences of the user.

Although the tree structure used by Thor is an ideal arrangement for the integration of a centralized management utility, further elucidation of the centralized management provided by Thor is needed before it can be determined whether this repository system satisfies Requirements **R4** and **R5**.

### 5.3.3 Centralized Management Utility

Requirement **R4** specifies that the repository must provide a single location or interface through which a user's credentials are managed and any changes are propagated to the participating mobile devices. The root repository node provides the excellent foundation for such a task. This node already provides an interface to manage the credentials — all that remains is to modify the root and leaf repository nodes so they can communicate. Unfortunately, this method for the creation of a centralized management utility violates Goal **G2**, which dictates that no changes should be made to existing repository implementations.

Thor's Central Management Utility provides an innovative solution to Requirement **R4** without violating Goal **G2**. This utility is a client-side software agent. This utility can connect to any of the user's participating repository on the user's behalf. This connection uses the existing repository interface provided by the specific repository and thus does not require modification to any existing repositories.

The current version of the management utility connects to the root repository node, the leaf repository node collocated on the same device, or both at the same time. This utility provides a uniform interface to all the user's repositories.

When the connection to the desired repository is made, the management utility retrieves its meta-data that is stored there. This meta-data is stored as a "credential" in the repository and contains a list of credential identifiers and modification dates (see Section 5.3.4 for more detailed information). This list is used to display the repository contents as well as limited credential information: issuer subject, expiration date, etc.

When connected to the root repository node, the user is permitted to manage and access the credentials stored there. Changes made while connected to this node are recorded so leaf nodes can be informed of the modifications when they connect. These alterations are recorded in the utility's meta-data.

When connected to a leaf repository node, the user is also permitted to manage and access the credentials stored there. Changes made while connected to this node are also recorded in the utility's meta-data. However, as stated in Section 5.3.1, if a credential is to be made available to other leaf repository nodes, then it must first be uploaded to the root repository node.

When the user is connected to both the root repository node and a leaf node, the Central Management Utility compares the information contained in the meta-data from each repository and notes the differences. Only when the utility is connected to both the root repository node and a leaf node can synchronization occur.

Thor uses a simple, yet effective method for synchronization. If the leaf repository has credentials that are not in the root repository the user is prompted to determine whether it should be uploaded. If a new copy of a credential exists in either the leaf or root node, then the old copy is updated. If the root repository has credentials that the leaf does not, by default no action is taken. Credentials from the root repository

are only downloaded to the leaf repository on direct command of the user. This preserves the ability of the user to store a subset of his credentials on the device.

Thor's synchronization provides a credential-level granularity in compliance with Requirement **R5**. When a credential is updated, only the information needed to update that credential is transmitted. In this case, it is necessary to transmit the entire credential. This new copy will overwrite the old copy.

In summary, Thor's Central Management Utility provides a single location for a user to manage all his credentials and propagate those changes to the user's participating devices. This fulfills the conditions dictated by Requirement **R4**. This utility accomplishes its tasks without any need to modify the existing repositories, thus adhering to Goal **G2**.

### 5.3.4 Meta-Data

Meta-data is stored in the repositories used by Thor and contains information vital to the task of the Central Management Utility. The meta-data is also a time/resource saving mechanism. A list of the entire repository contents, with valuable descriptive information, can be retrieved without having to actually look at any of the credentials in the repository. This section describes how the meta-data is structured and what it contains.

Thor's meta-data is an XML document. It is stored and retrieved from the repository just like an ordinary credential. In fact, the repository shouldn't be able to distinguish between the meta-data and other credentials. In the event that meta-data is larger than the credential size limit imposed by a repository it can be split into multiple files and later recombined.

The *basic* format of the meta-data XML document contains the following tags:

<**RepositoryMetaData**> This tag is the root element of the document. It has one and only one child, the credentials tag.

<**credentials**> This tag has zero or more credential tags as children.

<**credential**> This tag represents a single credential in the repository. It has two attributes: *realID* and *lastModified*. realID is the unique identifier for this credential. lastModified contains the timestamp of the last modification date of this credential. This tag has zero or more instances of the credentialInfo tag as its children.

<**credentialInfo**> This tag contains credential specific information. This tag has two attributes: *name* and *value*. name is the identifier for the information contained in contents of value. This tag has no children.

The credential identifier in <credential> is a user specified unique identifier. It is recommended that this identifier be something more descriptive than "cred1". Something like "Online banking authentication" is much better. This name is used not only for easy user selection and management, but also for synchronization purposes. The date contained in the lastModified attribute of <credential> should be accurate to the nearest minute and contain info about the time zone of this value.

The credentialInfo tags contain a set of attribute/value pairs. These values represent values of the credential that the management utility automatically collects when the credential is first uploaded as well as user-defined and entered values. Figure 5.5 is a list of common attributes in an X.509v3 credential. Figure 5.6 is a sample meta-data file in the basic format.

If no meta-data exists in the repository, there are two simple procedures to create it. The user can either enter all the information in by himself, or the management utility can auto-detect the contents of the repository. Both methods involve downloading, examining each credential, and the creation of the appropriate tags in the

1. Issuer
2. Subject
3. Valid from
4. Valid to
5. Serial Number

Figure 5.5: A list of common attributes in a X.509v3 credential. These values are automatically detected and stored in the meta-data when the credential is first uploaded.

XML meta-data document. The time required depends on the number of credentials in the repository.

This section has presented the basic format of the meta data. The enhancements of Chapter 6 require additions to the basic meta-data. These additions are elucidated in Section 6.4.

## 5.4 Does Thor Meet the Requirements of a Mobile Environment?

As a hybrid repository, Thor automatically has Requirements **R1**, **R2** and **R3** fulfilled. Thor's Central Management Utility provides satisfaction for Requirements **R4** and **R5**. Table 5.1 provides a summary of how Thor measures up to the requirements for a secure credential repository in a mobile environment.

```
<RepositoryMetaData>
  <credentials>
    <credential realID="LMC Employee" lastModified="2004-05-24T19:33:06-0700">
      <credentialInfo name="Issuer" value="Lucky Moose Consulting"/>
      <credentialInfo name="Subject" value="Doc Hopper"/>
      ...
    </credential>
    <credential realID="ACM" lastModified="2004-12-31T13:31:02-0700">
      <credentialInfo name="Issuer" value="ACM Membership Deptartment"/>
      ...
    </credential>
    <credential realID="IEEE" lastModified="2003-09-22T05:54:02-0700"/ >
    ...
  </credentials>
</RepositoryMetaData>
```

Figure 5.6: A sample meta-data file created by Thor's Central Management Utility. This is the basic format for this file (see Section 6.4 for an extended version).

| Requirements | | Metrics | | |
|---|---|---|---|---|
| **R1** | The loss of the mobile device must not equate to a loss of any of a user's credentials. | **M1** | In the absence of the mobile device, can the contents of the repository be restored without reissuing all the credentials? | YES |
| **R2** | The repository must function within the computational restrictions of the mobile device. | **M2a** | Can the repository be loaded and executed on the device? | YES |
| | | **M2b** | If it can be executed, does the response time of the repository exceed an acceptable threshold? | YES |
| **R3** | The credentials in the repository must be available, regardless of the current topology of the mobile device. | **M3** | Are the credentials in the repository available at transaction time without a connection to off-device resources? | YES |
| **R4** | The repository must have an interface through which users may manage all their credentials. Changes made here can then be propagated to all of the users' participating devices. | **M4** | Can the user manage all his credentials in a single location and have the repository propagate those changes to participating devices? | YES |
| **R5** | The repository synchronization mechanism must include a granularity that permits synchronization at the credential-level. | **M5** | If a single credential is modified, is that change the only information needed to update the appropriate repositories? | YES |

Table 5.1: Summary the fulfillment of the requirements for a secure credential repository a mobile environment by Thor.

# Chapter 6 — Security and Usability Enhancements

With an eye to fulfilling Goal **G3**, this chapter demonstrates how the repository abstraction and central management provided by Thor enables the integration of usability and security enhancements into the underlying repositories without their modification. The following sections delineate the scope and benefits of three such enhancements: 1) Organization of the credentials; 2) Password management; and 3) Credential identifier obfuscation. Each of these enhancements are integrated into Thor's Central Management Utility. The final section in this chapter presents the modifications needed to the basic meta-data structure presented in Section 5.3.4 to enable these enhancements.

## 6.1 Organization of Credentials

When working with multiple credentials, it is useful to have a method of organization for those credentials. The centralized management utility of Thor allows a user to create groups for the credentials in his repositories. These groups are subsets of the set of all of a user's credentials and they need not be mutually exclusive. The default group is called "ALL". This group contains all the credentials in the repository.

An example of a logical division for credentials groups is their usage context. For example, groups can be created for business, e-commerce, medical, military, educational, government, and/or personal use. Groups based on context allow a user to quickly specify the retrieval of credentials based on the transaction he is performing or will perform in the future. Groups can also be created based on the application that will be using them. It makes sense that an application for e-commerce should not require access to non-related credentials. The credentials are, of course, only as safe as the weakest application that has access to them. Limiting the credentials accessible

to an application limits the vulnerability to a user's most sensitive credentials.

The creation of groups is accomplished by a *virtual organization* of the credentials. In other words, the credential organization that is created by a user is not necessarily reflected in the actual physical storage of the credentials within the repositories. This organizational structure is stored persistently within Thor as part of the meta-data stored in each repository(see Section 6.4).

The creation of credential groups creates another area for synchronization. If a group is expanded or shrunk on the root repository node, the changes to both the group and the credentials can be automatically extended to the repository nodes when they connect.

## 6.2 Password Management

The grouping capabilities of the centralized management utility increases the usability of the existing repositories. Also, by dictating which applications have access to which credentials it increases the security of credentials while on the local device. These improvements do not, however, increase the security of the credentials that reside on the remote repository node. An effective way to increase the security of the credentials on the root repository node, as well as the leaf repository node, is to encrypt each credential with a strong, high-entropy password. This makes each credential as hard to break as any other. This is especially effective in thwarting an insider attack on the root repository node if it is hosted by a third party.

Unfortunately, it is also very hard for the average user to remember such a password. This problem is compounded when multiple such passwords are required to be memorized by a user. This creates a significant management problem that usually results in a user writing down his strong passwords in a non-secure place. In this case, it is highly beneficial to have a software agent to act on behalf of a user and remember the decryption passwords for each credential. A user then has to remember only one

password to authenticate himself to the software agent. The strength of this entire system resides in the strength of the password to authenticate to the software agent. Since there is only one password to remember, it could be enforced to be strong but still not be too difficult to remember.

The idea of protecting many passwords with a single password is not a new concept. Many systems, e.g., Password Safe [4], have been designed to accomplish this very idea. It is obvious, of course, that the security of the whole system lies in the strength of the single password. No matter how strong the other passwords are, they can all be accessed via the single master password. Given its sensitive contents, the password file is an ideal candidate for attack.

Several disadvantages exist with respect to the password management scheme as described above. If the password manager is lost, destroyed, or otherwise unavailable, having such difficult passwords for each credential will avail a user nothing as his data is all but destroyed as well; the data is encrypted and the key to unlock it is itself locked and may never be recovered. This raises concerns about its availability and legitimate accessibility.

The meta-data of the management utility is an ideal location to store the password management information. Since there is information stored about every credential in meta-data it is simple to add another piece of information that contains the encryption key for that credential. Because the meta-data is specific to the repository node in which it resides, the root repository node contains a collection of all the credential encryption keys, while the leaf nodes contain only the keys that pertain to the credentials stored on that node.

### 6.2.1 Password Recovery

The root repository node provides a "backup" for all the password manager information. However, what is to be done if a user's password is forgotten? Certainly

a user does not want give his password to a third party for safe keeping. This type of password backup scheme allows the third party to access his sensitive information, thus enabling the third party to impersonate the user. The password cannot be simply reset because it is needed to unlock the decryption keys stored in the encrypted meta-data. An encrypted backup copy of the original password must be stored. This solution raises two concerns: 1. Where is this backup copy stored? and 2. What key is used to encrypt it?

To answer the first concern, the backup password should not be exclusively stored locally because the loss of the device equates to a loss of the backup password. A copy of the backup password must therefore be stored off-device. The root repository node makes an ideal candidate for this storage. If the password to the root repository is also lost, it can be reset by whatever means the remote repository has established for dealing with forgotten passwords, thus enabling the retrieval of a copy of the encrypted backup password. In regards to the second concern, great care must be taken in the selection of the encryption key. If a password-derived key is used to encrypt the backup password, what is to be done if this password is also forgotten? This train of thought begins an endless loop and at some point a limiting threshold must be set.

Many online systems, in the event of a forgotten password, ask a user a question about his personal history. A correct answer re-enables a user's access to the system. Frykholm and Juels [14] propose an enhancement to this system that allows a high-entropy key to be derived from a certain percentage of correct responses to a sequence of questions. This derived key is an ideal candidate to encrypt the backup password because it provides a user the ability to recover a password without interaction with a third party. It also provides an increased probability of supplying the information required to successfully retrieve a password.

Although the integration of this multiple question password scheme would be very beneficial, its implementation is beyond the scope of this research. The current implementation of the management utility includes a single question password generation scheme. A future extension will integrate the scheme presented by Frykholm and Juels.

## 6.3 Credential Identifier Obfuscation

The repository interface in Section 5.2 specifies that there must exist a unique identifier for each item in the repository that enables the client and server to uniquely identify a specific information package. Although the credentials that are uploaded are encrypted, this identifier could leak sensitive information when the mere existence of a credential is sensitive. For example, if a label says, "Online banking authentication credential" it may prove to attract attention and become the focus of an attacker. A label of nonsensical characters such as "JMVRYIKG/GGFBN25" reveals nothing about the contents of the information package. This is called *credential identifier obfuscation.*

The use of non-descriptive identifiers is particularly important in helping to prevent a malicious insider or someone who has broken into the repository from focusing his attack on an obviously valuable target. To be effective this obfuscation should be done on all credentials. Since these nonsensical names mean nothing, it is difficult for a user to remember the mapping of identifiers to credentials. A mapping of meaningful credential identifiers to the randomly generated obfuscated ones is easily stored with the repository meta-data. In the management utility this feature is completely automated and does not require any user intervention. Since the repository meta-data is a very attractive target, it is also included in this identifier obfuscation.

## 6.4 Modifications to Meta-Data

In order to implement these enhancements, several new tags must be specified for the meta-data XML document. The *extended* format of the meta-data XML document contains the following tags and tag modifications:

<**RepositoryMetaData**> This tag is extended to have two more children: <groups> and <moreMetaData>.

<**groups**> This new tag represents the collection context groups and has zero or more group tag as children.

<**group**> This tag represents a single context group of the repository. It has two attributes: *name* and *lastModified*. name is the unique identifier for this group. lastModified contains the timestamp of the last modification date of this group. This tag has zero or more instances of the member tag as its children. This tag can also have zero or more group tags as its children. This allows for the creation of subgroups.

<**member**> This tag represents a member of a context group. It has a single attribute: *realID*. The value of realID corresponds to the unique identifier for this credential and the realID value of its credential tag.

<**moreMetaData**> This tag is a pointer to an additional meta-data file. Absence of this tag means that there is not an additional meta-data file. This tag has two attributes: *name* and *key*. name is the file name for the additional meta-data. key is the encryption key for this additional file. The multiple meta-data files can be chained together with this tag.

<**credential**> This tag represents a single credential in the repository. It is modified in this format to have two new attributes, in addition to the two that have been

previously defined. These additional attributes are: *fakeID* and *key*. fakeID is the obfuscated identifier for this credential. key is the encryption key for this credential.

As discussed in Section 5.3.4, it may be necessary to split a single meta-data file into multiple pieces due to constraints imposed by the repository. Another motivation for splitting the file is that a single large meta-data file could be easily identified when surrounded by nothing but smaller credential files. Splitting a large meta-data file into smaller pieces and padding as necessary is very helpful in disguising the file from an attacker.

Figure 6.1 is a sample meta-data file in the extended format.

```
<RepositoryMetaData>
  <groups>
    <group name="LMC Creds" lastModified="2004-05-25T108:01:50-0700">
      <member realID="LMC Employee" />
      <member realID="LMC R&D VPN" />
    </group>
    ...
  </groups>
  <credentials>
    <credential realID="LMC Employee" lastModified="2004-05-24T19:33:06-0700"
                fakeID="2nmgbKhd5Y..." key="aFVTMOaDDf...">
      <credentialInfo name="Issuer" value="Lucky Moose Consulting"/>
      <credentialInfo name="Subject" value="Doc Hopper"/>
      ...
    </credential>
    <credential realID="ACM" lastModified="2004-12-31T13:31:02-0700"
                fakeID="hDCRDO2B05h..." key="PDWodMkKKg...">
      <credentialInfo name="Issuer" value="ACM Membership Deptartment"/>
      ...
    </credential>
  </credentials>
  <moreMetaData name="hDCRDO2+B05..." key="bPDWodMkKW..."/ >
</RepositoryMetaData>
```

Figure 6.1: A sample meta-data file created by Thor's Central Management Utility. This is the extended format for this file (see Section 5.3.4 for the version).

# Chapter 7 — Implementation

This chapter presents the proof-of-concept implementation of Thor. The following sections describe the design decisions made for the repository nodes, the target platforms, and the function of the Central Management Utility. The behavior of this prototype provides evidence that Thor does indeed satisfy its design and implementation goals.

## 7.1 Root Repository Node

This prototype implementation of Thor uses the SACRED Credential Server [9, 16, 13] as its root repository node. This repository was chosen because it has a simple, easy to use interface and is designed for application and device independence. SACRED also has the ability to use a variety of communication and authentication protocols.

The implementation of the SACRED Credential Server selected [5] is the first publicly available implementation. This version is written in Java and was developed through a collaborative effort between the Internet Security Research Lab[1] at Brigham Young University and the National Center for Supercomputing Applications[2] at the University of Illinois at Urbana-Champaign. This implementation of the SACRED Credential Server uses BEEP [20] as its communication mechanism. The SACRED profile in BEEP is tuned with TLS and uses SASL/DIGEST-MD5 as an authentication mechanism.

## 7.2 Leaf Repository Node

The Java KeyStore serves as the leaf repository node in this implementation. The KeyStore repository was designed to have a simple, easy to use API and meets the

[1]Participants: Kent Seamons, Tim van der Horst

[2]Participants: Jim Basney, Zheng Sun, Dong Xin

interface requirements specified in Section 5.2. It can also be accessed by a myriad of different applications. Specifically we used the JCEKS from the SunJCE provider, which is included in the Java 2 SDK v1.4.2. The Java KeyStore also has the added bonus of being able to use a different physical storage type (e.g., use a PKCS#12 repository) while maintaining the exact same interface. Due to this feature a user can choose a variety of providers and physical store types while maintaining an identical interface.

The Java KeyStore is also available for the Java 2 Micro Edition (J2ME). J2ME is specifically designed for use with mobile phones, PDAs, and embedded systems. Future extensions will make use of the J2ME.

## 7.3 Target Platforms

A minimum of two devices are needed for this prototype. The first device, a standard desktop machine, houses the SACRED Credential Server. This machine is a Pentium IV 3.0 GHz with 1 GB of RAM running Windows XP. Once the SACRED server is configured for standard use, neither the repository nor the machine requires any modifications to be integrated as the root repository node of Thor.

The second device needed for this prototype is a mobile device. There are many different types of mobile devices and, as explained in Section 2.3, they have a wide range of computational abilities. Due to these constraints it is unlikely that a single implementation of Thor will function properly on every one of these devices. For the purposes of this research the Thor prototype will be designed to run on a laptop-class device. Although this is on the high end of the computational spectrum of mobile devices, it is adequate to prove that the design and implementations goals of Thor are met. Future extensions will adapt the implementation so that it will work within the constraints of the target device.

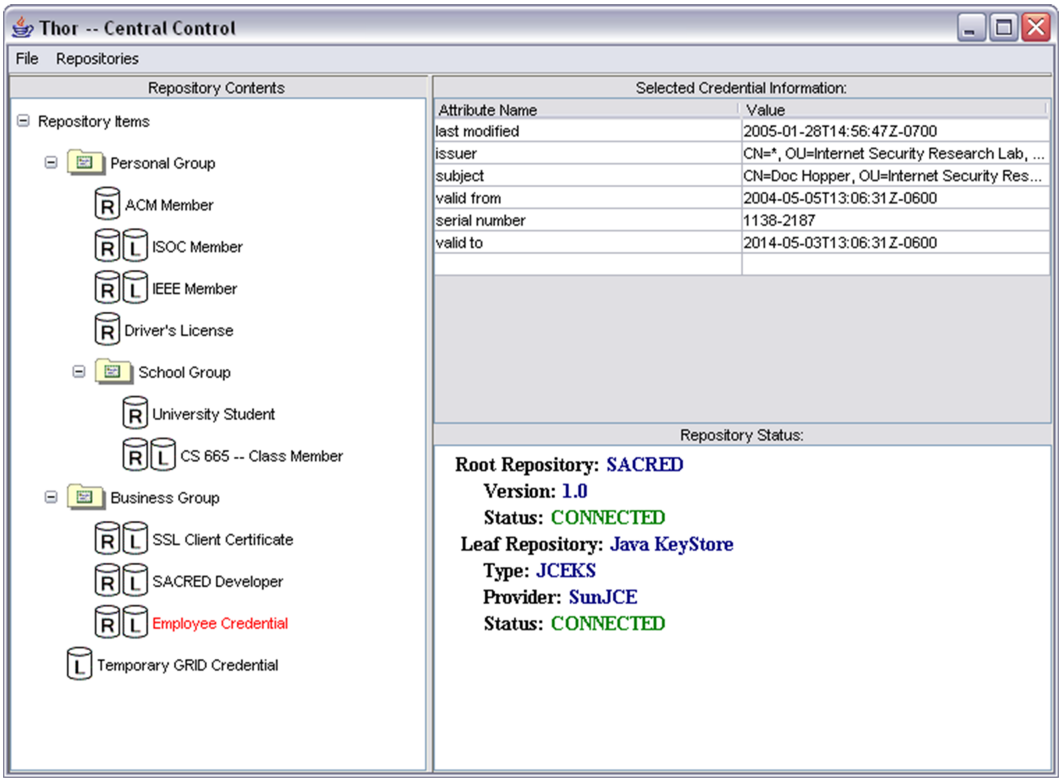


Figure 7.1: The Central Management Utility provides a location to manage and view the credentials in the repositories controlled by Thor.

## 7.4 Application Interface

In this version of the Thor prototype, applications do not directly interface with Thor. Instead, applications interface directly with their traditional repositories, as they did before they were integrated with Thor. Thor manages the contents of these repositories as it would any other leaf node. The creation of an interface to directly access the credentials that reside in Thor is discussed in Chapter 8 as future work.

## 7.5 Central Management Utility

Thor's Central Management Utility provides a graphical user interface to manage the contents of the repositories controlled by Thor. To access the utility, a user must first authenticate to the utility via a username and password. Once the user has authenticated the meta-data is retrieved from the repositories and displayed.

Thor's Central Management Utility contains three displays (see Figure 7.1). The *Repository Status* section displays the repository nodes with whom the utility is currently configured to make connections. This section also displays the specific repository type and its connection status. The utility in Figure 7.1 is connected to both a leaf node and the root node.

The second portion of the utility is the *Repository Contents*. This part of the utility displays the credentials that reside in the connected repositories. Each credential is listed by its unique credential identifier. To the left of each identifier there is either one or two cylinders. A cylinder with an "R" indicates that the credential resides in the root repository node. A cylinder with an "L" indicates that the credential resides in the leaf credential node. The presence of both cylinders means that the credential is in both the root and leaf nodes.

The management utility integrates the credential grouping capabilities described in Section 6.1. The currently active groups are also displayed in the Repository Contents section of the utility. In Figure 7.1, there are three groups: "Personal Group", "School Group", and "Business Group". Note how the "School Group" is a sub-group of the "Personal Group". The "Temporary GRID Credential" does not belong to a group. Groups are created and managed via the *Group Management* Utility (see Figure 7.2).

The third section of the utility displays additional information about the currently selected credential. The information displayed is taken from the credentialInfo tags of that credential in the meta-data. This section also provides the ability to modify or add information about a credential.

Thor's Central Management Utility also implements the password management and identifier obfuscation described in Chapter 6. These features are completely transparent to the user. An additional feature that can be enabled is the management

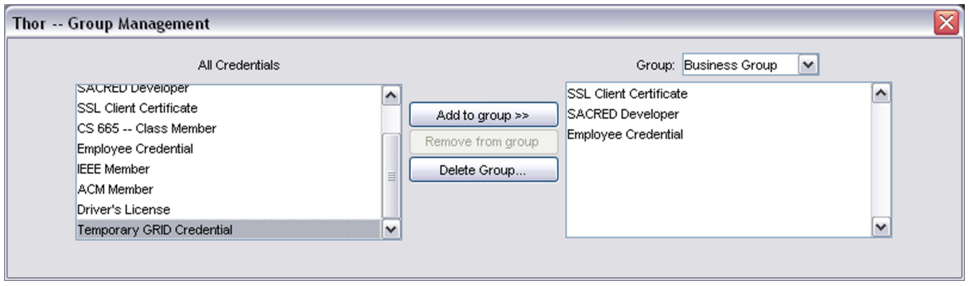


Figure 7.2: This interface allows the user to create and manage groups of credentials. A single credential can be a member of multiple groups. The list on the left contains all the user's credentials. When a group is selected, its credentials appear in the list on the right. Credentials can be added or removed from a group using the "Add" and "Remove" buttons between the lists.

$$\text{keyMaterial} = \text{CryptographicHash}(\text{password} || \text{<repositoryName>})$$

Figure 7.3: This is a simple password derivation scheme. The repository name is appended to the plaintext password used to authenticate to Thor. This new string is put through a cryptographic hash function, such as SHA-1, which creates a 20-byte result. This result is then used as the derived password.

of repository authentication passwords. The current prototype of the management tool uses a simple password derivation technique to create new passwords for the repositories. These new passwords are derived from the authentication password to Thor and never need to be stored as they can easily be generated when needed. Figure 7.3 illustrates the password derivation technique. Password recovery is a separate application and is described in the next section.

## 7.6 Password Recovery Utility

In the event that the password to the Central Management Utility is lost, a user can run Thor's Password Recovery Utility. This will allow the user to recover his forgotten password. The user will have to reset the password to the root repository node using existing procedures. As mentioned in Section 6.2 this is a simple question/answer recovery scheme. This utility produces a decryption key and a credential

**Goals**

| | | |
|---|---|---|
| **G1** | All five requirements for a secure credential repository in a mobile environment must be satisfied. | ✔ |
| **G2** | There must be no modifications required to the existing repository implementations. | ✔ |
| **G3** | Where possible, increase usability and security of existing repositories without modifying them. | ✔ |

Table 7.1: Summary of how well Thor meets its design and implementation goals specified in Section 5.1. A ✓ indicates fulfillment. A ✗ indicates an insufficiency

identifier using the answer to the question. The credential associated with this identifier is retrieved from the root repository node. This "credential" is actually the backup password file. The backup password file is created during the initial setup of Thor. This file is also padded to be the size of a credential. Once the password is recovered the meta-data contained in the repository is unlocked and all of the user's credentials are one again accessible. After the password is recovered it is recommended that the user choose a new password.

## 7.7 Does Thor Meet its Design and Implementation Goals?

There are three design and implementation goals for Thor (see Section 5.1). Section 5.4 shows that Thor meets Goal **G1**. Section 5.3.3 shows the fulfillment of Goal **G2**. Chapter 6 and Section 7.5 show that Thor satisfies Goal **G3**. When all is said and done, Thor meets all of its design and implementation goals. Table 7.1 provides a summary of how Thor measures up to its design and implementation goals.

# Chapter 8 — Future Work

Thor creates many possibilities for further improvements to repository management and security. Many of these improvements have already been discussed in previous chapters as future extensions. There still remains several of other repositories that have not yet been incorporated into our system. Integration of these repositories and their benefits is a definite priority. One of the most important features that has yet to be integrated with Thor is smart cards.

Smart cards provide a unique solution to the problem of protecting sensitive information while it is not in use by a protocol. This protection is both at the physical protection level as well as the electronic level. A smart card is tamper-resistant and can perform calculations on the data without removing it off the card. This ability comes with its costs. Smart cards are limited in the amount of information that can be stored on them. Usually this is a limit of 32K for both an application and the data to be stored. They are also limited in their computational power. There is an overhead required to access and use the card. This overhead needs to be taken into account if smart cards are going to be considered for integration into a secure system and if time constraints and processing power are issues. Smart cards have a limitation on the devices that they can operate with. In order for a device to use a smart card it must be equipped with a smart card reader. These readers can be either wired or wireless, but nonetheless they will require resources from the device in order to perform their tasks. Perhaps the greatest advantage of the adoption of smart cards is the ability to require two things of a user: Something a user has or possesses, the card, and something that a user knows, his password or PIN required to access the card. This is something that is very desirable when a system is protecting sensitive

credentials that, if revealed, could have highly negative repercussions for the person whose information was hijacked or abducted.

Another interesting topic with respect to smart cards is their integration into a system. The method of smart card integration remains an open question. Will the system charged with the authentication be aware of a smart card or not? If the system is aware of a smart card then the system must be designed so that it can recognize when the authentication information is not originating from the card. If the system is designed such that it is not aware of a smart card's involvement the system can be viewed as oblivious to this fact. A user therefore could have the means to have a smart card authenticate on his behalf or log in normally. The server never knows the difference. Indeed this raises the question as to whether the system should ever know by what means a user authenticated to the system. This is beneficial to a user because it means that there are multiple methods to authenticate to the same system based on the capabilities, or future capabilities of the user. The server, on the other hand, may be very interested in learning how a user authenticated because it allows the server more options to use in a more fine-grained access control model. For example, a password authentication is sufficient to download a credential, but to modify it requires smart card authentication.

Smart cards also provide a solution to a problem called *keyjacking* [18]. This problem refers to the fact that the contents of any process are actually under the control of any other process running with the user's identity. By keeping the credentials off the device and away from any malicious processes, the integration of smart cards into Thor greatly reduces this risk. A similar solution was applied to the MyProxy credential repository through the addition of an IBM 4758 cryptographic co-processor[17].

Currently, Thor is limited to credential retrieval and storage. The actual use

of the credentials is in the applications. The integration of a virtual smart card (the emulation of a smart card in software similar to RSA's Keon Web PassPort) so that other applications could access and use Thor like a physical smart card is very desirable. This also ensures that a user's private keys never leave the protection of the repository. This is particularly compelling when a user is accessing his information on a device that he does not own. The virtual smart card is pinned in memory and ensures that no residual credential information is left on that device. Though the virtual smart card is non-persistent, its contents can be stored in the local repository when communication to the remote repository is not available or desired. Building on this methodology, several virtual cards can be created based on the context groups described in Section 6.1, thus limiting the applications that have access to specific credentials. The use of the smart card interface also allows applications to access credentials through an existing standard, rather than forcing the adoption of a new interface.

Further refinement of the repository interface (see Section 5.2) is also in order. This fine-tuning will lead to the proposal of an API standard which will allow for the further interoperation of existing secure repositories. This interoperability increases the security of sensitive digital credentials, and most importantly, gives the user the power to select the combinations of systems that best meet his personal needs.

Many remote repositories have the ability to share computational loads with the mobile device. In order to take advantage of this feature the interface specified by Thor needs to be expanded to incorporate this functionality.

Another area for further investigation and research is to incorporate better methods of password recovery. This system could benefit from existing password recovery schemes that do not rely on a third party. It is important to not rely on a third party because a user should not have to give a third party access to his sensitive

information.

# Chapter 9 — Conclusions

A mobile environment contains many hazards to managing the sensitive information contained in digital credentials. This research examines these hazards and establishes a set of requirements for secure repositories in a mobile environment. These requirements take into account the physical harshness, the connectivity, and usability issues of this environment. It also shows that existing repositories, in context of these requirements, are inadequate.

A new type of repository, the hybrid repository, is defined. The hybrid repository combines the features of local and remote repositories to address the requirements of a mobile environment.

Thor, a prototype design and implementation of a hybrid repository, is presented and examined. Thor's repository interface allows a user to combine existing local and remote repositories into a single virtual repository. This hybrid repository, in addition to satisfying the requirements of a mobile environment, adds several usability and security features to existing repositories. These additional benefits are achieved without any modification to the repositories. This is accomplished by the storing of meta-data (which contains additional information about the credentials) in the repository.

Thor enables the creation of credential groups to assist the user in keeping his credentials organized. Thor also provides several features that are transparent to the user. Each item is stored in the repository with a high-entropy encryption key and with an obfuscated credential identifier. A reference implementation is presented to illustrate these improvements.

Thor empowers users by automating several features that enhance the protection

of their sensitive digital credentials and by giving them the flexibility to choose how and where their information is stored while in a mobile environment.

# References


[1] Betrusted UniCERT User Roaming. *Available at http://www.betrusted.com/products/unicert/advancedmodules/roaming.asp.*

[2] Entrust True Pass White Paper. *Available at http://www.entrust.com/truepass/index.htm.*

[3] NSD Security Solutions Whitepaper. *Available at http://www.nsdsecurity.com/downldsx0343/nsd_security_solutions_whitepaper.pdf.*

[4] Password Safe. *Available at http://www.schneier.com/passsafe.html.*

[5] A Reference Implementation of SACRED. *Available at http://sacred.sourceforge.net.*

[6] RSA Keon Web PassPort Technical Overview. *Available at http://www.rsasecurity.com/products/keon/whitepapers/passport/kwebp_wp_0702.pdf.*

[7] Verisign Roaming. *Available at http://verisign.com/products-services/security-services/pki/pki-security/wireless-roaming/index.html.*

[8] S. Araki. The Memory Stick. *IEEE Micro*, 20(4):40–46, July-August 2000.

[9] A. Arsenault and S. Farrell. Securely Available Credentials – Requirements. *IETF Informational RFC 3157*, August 2001.

[10] J. Basney, W. Yurcik, R. Bonilla, and A. Slagell. The Credential Wallet: A Classification of Credential Repositories Highlighting MyProxy. In *31st Research Conference on Communication, Information and Internet Policy*, Arlington, Virginia, September 2003.

[11] P. Bonatti and P. Samarati. Regulating Service Access and Information Release on the Web. In *Seventh ACM Conference on Computer and Communications Security*, Athens, Greece, November 2000.

[12] T. Dierks and C. Allen. The TLS Protocol Version 1.0. *IETF Standards Track RFC 2246*, January 1999.

[13] S. Farrell. Securely Available Credentials Protocol. *IETF Standards Track RFC 3767*, June 2004.

[14] N. Frykholm and A. Juels. Error-Tolerant Password Recovery. In *Eighth ACM Conference on Computer and Communications Security*, Philadelphia, PA, USA, November 2001.

[15] S. Gupta. Security Characteristics of Cryptographic Mobility Solutions. In *1st Annual PKI Research Workshop*, Gaithersburg, MD, April 2002.

[16] D. Gustafson, M. Just, and M. Nystrom. Securely Available Credentials (SACRED) – Credential Server Framework. *IETF Informational RFC 3760*, April 2004.

[17] M. Lorch, J. Basney, and D. Kafura. A Hardware-secured Credential Repository for Grid PKIs. In *IEEE/ACM Internation Symposium on Cluster Computing and the Grid*, Chicago, Illinois, April 2004.

[18] J. Marchesini, S.W. Smith, and M. Zhao. Keyjacking: The Surprising Insecurity of Client-Side SSL. Technical report tr2004-489,, Department of Computer Science, Dartmouth College, 2004.

[19] J. Novotny, S. Tueke, and V. Welch. An Online Credential Repository for the Grid: MyProxy. In *Tenth IEEE International Symposium on High Performance Distributed Computing*, San Francisco, CA, August 2001.

[20] M. Rose. The Blocks Extensible Exchange Protocol Core. *IETF Standards Track RFC 3080*, June 2004.

[21] R. Sandhu, M. Bellare, and R. Ganesan. Password-Enabled PKI: Virtual Smart Cards versus Virtual Soft Tokens. In *1st Annual PKI Research Workshop*, Gaithersburg, MD, April 2002.

[22] W. H. Winsborough, K. E. Seamons, and V. E. Jones. Automated Trust Negotiation. In *DARPA Information Survivability Conference and Exposition*, volume I, Hilton Head, SC, January 2000.

[23] M. Winslett, T. Yu, K. E. Seamons, A. Hess, J. Jacobson, R. Jarvis, B. Smith, and L. Yu. Negotiating Trust on the Web. *IEEE Internet Computing*, 6(6):30–37, November/December 2002.