

# Concord: A Secure Mobile Data Authorization Framework for Regulatory Compliance

*Gautam Singaraju and Brent Hoon Kang – University of North Carolina at Charlotte*

## ABSTRACT

With the increasing adoption of mobile computing devices that carry confidential data, organizations need to secure data in an ever-changing environment. Critical organizational data should be protected from a) a disgruntled user's access and b) a theft or loss of the mobile device. When such compromises do occur, future data access should be immediately revoked and the knowledge of the data that might have been exposed be identified. Such assessment enables an organization to demonstrate its adherence to mandated regulatory compliance.

We propose Concord: a framework that provides an organizational service that allows an organization to monitor data that has been accessed on its users' mobile devices. Concord distributes trust among multiple entities so as to enable data access following their successful interaction. Firstly, to enable data access, users of the mobile device require the organization's involvement to access the data on the mobile devices. Likewise, in the event of loss or theft of a mobile device, organizations can immediately discontinue further requests for data accesses to the previously-unread data on the mobile device. Secondly, a valid user's consent is required to access the data. Thus, should an intruder somehow receive organizational permission, the data on the mobile device is still inaccessible. Thirdly, upon identification of a compromise, Concord provides the organization with the detailed information about the data that has been exposed enabling them to initiate steps for regulatory compliance.

## Introduction

Today, organizations are faced with numerous cyber threats from internal as well as external sources. Hackers operating from external institutions constantly lurk in search of an IT vulnerability that would allow them to compromise non-public data. Meanwhile, insiders or disgruntled employees have access to confidential information [8]. Notably, the recent increase in the use of mobile devices [11] has increased organizations' risk of losing sensitive data as mobile devices are prone to theft or loss.

Incidents of data leak, theft, or misuse are not rare; regulation compliances [7, 13] have been imposed to guide an institution's data management and security policies. Regulatory compliances are designed to protect the privacy of non-public information. The compliance standards require institutions to monitor each user's data accesses. For example, data breach notification bill states that in case of data loss, institutions should inform the affected customers.

Protecting data is challenging, especially when the data is on mobile devices. We stipulate that a system that protects data should:

- a) Safeguard the confidentiality of the data;
- b) Enforce immediate access revocation; and
- c) Record the history of data accesses.

To address these requirements, we propose Concord, a data monitoring framework that assists in

complying with regulatory standards. Concord employs a "syndicated approach" where user's access to institution's data is dependent upon collective interaction of the entities that govern the data. The entities can only be partially trusted; while the complete trust can be placed only after their interaction.

Towards supporting such an interaction, Concord uses a 2-out-of-2 threshold cryptographic technique, which requires at least two entities to concur for data access. This paper uses the cryptographic technique called mediated RSA (mRSA) [6, 14]. Concord framework can be extended to use any 2-out-of-2 threshold cryptographic technique.

Using the Concord framework, a user of the mobile device needs to obtain the organization's consent to access data that resides on the mobile device. Consequently, upon detection of a compromise, the organization immediately discontinues further data access requests. An intruder would need to obtain both the organization's and the user's permission to access data. As Concord framework maintains the list of files accessed by users, it provides the organization with the knowledge of the exposed data.

Concord separates the data access history management from availability of data. As data availability can be improved by increasing the number of file servers, data access history should be maintained by another entity. Moreover the organization could securely maintain data access histories by securing a single entity.

The rest of the paper is organized as follows: the next section discusses the threat model addressed by Concord framework. Then related work in the area is overviewed after which the Concord framework and its design is shown. The next section describes the implementation issues of the Concord framework followed by an evaluation of the performance of the Concord framework and a conclusion.

### Threat Model

This section discusses the threat model faced by institutions. Organizations usually maintain network file servers whilst, users carry a subset of the data on their laptops. Due to the scattering of data across different storage devices, the organization faces potential risks due to mismanaged data servers or mobile devices. Consequently, the organizations would like to maintain a history of accesses of critical data for the fear of improper use or compromise of the data which further helps towards complying with regulatory standards. Based on the risks faced, we classify the threats into three categories:

1. Insecure Data Servers
2. Loss of Mobile Device with/without Disk Encryption
3. Disgruntled Users

#### Insecure Data Servers

An institution's data server is susceptible to compromise due to mismanagement, improper configuration or worse, a hacker. If the data is not encrypted, a hacker can access and modify critical data. Data servers, hence, cannot be trusted to store unencrypted data as they can be prone to compromise. As institutions usually maintain highly available data servers with backups, we assume service disruption is not an issue, while storing encrypted data is of utmost importance.

#### Loss of Mobile Device and Disk Encryption Keys

Mobile devices, such as laptops, carry a part of the organization's data. Loss of the mobile device could imply a data loss. For example, mandatory regulation compliance standard such as the Feinstein Bill dictates that upon loss of critical customer data, organizations must communicate about the loss to the affected customers. In addition to loss, the data on the mobile devices is susceptible to risks arising from unauthorized access either due to spy-ware, malware or from unauthorized users.

An organization would benefit from the knowledge of subset of data loss enabling them to alert a subset of users rather than all. Upon identification of an unauthorized access, the organization should be precisely aware of the data that could have been exposed to potential risks. To secure against threats, organizations encrypt data on the hard disk and secure the data key. If a hardware-based data key is compromised, the data on the mobile devices is prone to unauthorized access.

#### Disgruntled Users

About 60% of the reported attacks have been either from current or former employees [8]. Yet, organizations enable personal laptops with critical data which could be compromised. Upon detection of such users, data access from the mobile device should be immediately revoked to disable further accesses to the data stored on the laptop. Further the access history should be known to the organization.

#### Related Work

Based on the threat model discussed in Section 2, we discuss other currently available systems. We categorize the related work into three sections:

- Data protection for Mobile Devices,
- Data Protection Frameworks, and
- the cryptographic techniques available for such a framework.

#### Data Protection for Mobile Devices

Toward securing a mobile device when the user is not in the vicinity, Transient Authentication [3, 4] introduces a mechanism whereby the data and the memory of the mobile device is encrypted. Only users who carry a hardware token can access the mobile device. Transient Authentication aims to defend a system against unauthorized physical access. However, Transient Authentication does not consider the security threats arising from within the organization.

Another technique to secure the data is to encrypt it using a hardware-based encryption. For example, Seagate's hardware-based key is a technique that encrypts the disk. Though an encrypt-on-disk mechanism protects data on the mobile device when it is stolen, it cannot protect against loss of the key or a disgruntled user. In case both the key and the disk are lost, the organization would not be aware of the compromised data on the disk. Therefore, hardware key based solution does not consider compliance with regulatory standards.

IBM tackles these security issues with their laptops using smart cards and biometric authentication [11]. IBM has developed a Trusted Platform Module that stores user passwords on a chip. Should a mobile device be stolen, Absolute Software Corporation's Computrace transfers data to a remote location and erases the hard disk. These mechanisms do not safeguard the data against malicious users.

#### Data Protection Frameworks

Plutus [9] provides strong security in an unsecured server (file server) setting. Plutus's design consideration introduces the concept of the data owner (reader and writer) who maintains the keys for the data. In this mechanism, the data creator is a trusted entity. Plutus uses a lazy revocation mechanism to revoke a users' data access, i.e., write access is revoked when there is an attempt to write. Plutus assumes that the data creators own the data. Plutus

assumes the mobile device is secure and does not protect against theft of data.

Network file servers, such as Encrypted NFS [12] use OpenSSH for secure communication. Network file servers provide centralized storage architecture that allows a single point of revocation for future data accesses. Network file servers require users to connect to the server to access the data.

### Threshold Cryptographic Algorithm

Threshold cryptographic algorithms have been used to provide efficient revocation [6, 10]. mRSA [6, 14] is a 2-out-of-2 threshold cryptographic algorithm that provides a single public key and two private keys. The mRSA threshold cryptographic algorithm allows a trust model with three entities: a) the client; b) the mediator; and c) the server. The server maintains the public key whereas the two private keys are distributed among the client and the mediator. This necessitates that both the client and the mediator to partake in a trusted relationship to decrypt the data.

We use the mRSA cryptographic mechanism in Concord's design to assist in its efforts to monitor the data access of the users. Additionally, the mRSA technique provides a mechanism for fast revocation of a mobile device.

We propose the Concord framework which places partial trust on all the entities (data server and mobile device). Concord provides a mechanism to monitor user activity by employing the mRSA cryptographic technique. The collective interaction of the entities ensures data protection without the need for any additional hardware. As the key distribution is managed by a single trusted infrastructure-level service, Concord enables the administrator to set security levels in accordance with the critical value of the data. We discuss the different security levels later.

### Concord Framework

The Concord framework addresses the threat model discussed earlier by distributing the trust among multiple entities. Concord framework mandates that only encrypted data be available to any entity and access to data can only be permitted after a successful interaction among them. Concord employs both encrypt-on-disk and encrypt-on-wire mechanisms.

As Concord assumes that a laptop transmits the data over the Internet, we place a constraint on the laptop should be connected to the Internet to be able to access the data. We transfer the metadata rather than transmitting huge files. We assume that if decrypted data is locally-copied, data is lost.

As organizations might have multiple file services, Concord minimizes the data access control management by segregating the data management and access control to a single entity rather than on multiple file servers.

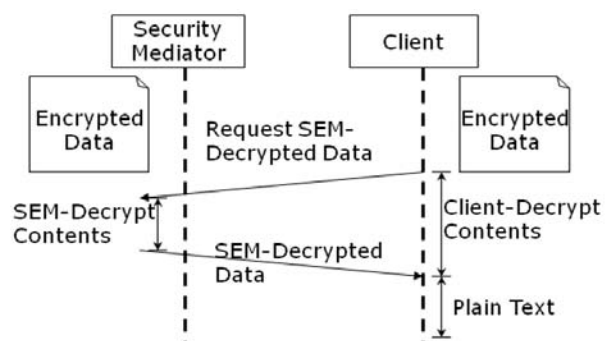
Concord assumes that the data creator need not be the data owner. The data access policies, therefore, need to be provided by the organization. The infrastructure-level monitors and revokes data accesses on a user's mobile device. As discussed earlier, along with the need to encrypt data for storage and transfer, the interaction with an intermediate entity mandates the use of a cryptographic mechanism. The cryptographic algorithm should:

1. Immediately revoke the client.
2. Mandate collaboration among entities for data access. (This ensures that compromise of any single entity does not compromise the data – by distributing trust among multiple entities.)

A 2-out-of-2 cryptographic mechanism supports such a requirement. The cryptographic mechanism requires two entities to collaborate to decrypt the contents. This provides an infrastructure-level service capable of maintaining access patterns of the users. We use the mediated RSA cryptographic algorithm [6, 14] a 2-out-of-2 cryptographic technique. In the following section, we discuss the mRSA protocol.

### mRSA Protocol

The Mediated RSA (mRSA) cryptographic protocol consists of a public key associated with two private keys. The private keys are distributed to one of the two entities: the Security Mediator (SEM) and the client. Any content that is encrypted by the server (holder of the public key) can be decrypted by combining decryptions by both the mediator and the client. Any single entity is unable to decrypt the content independently.



**Figure 1:** mRSA protocol with interaction between the mediator and the client to decrypt data (SEM – Security Mediator).

Figure 1 demonstrates the mRSA protocol. The encrypted contents are stored both at the client and the SEM. When the client requires access to the unencrypted contents, the client requests the SEM to decrypt the contents with its private key and starts decrypting the content with its own private key and is referred to as SEM-decrypted or Client-decrypted data. A single decrypt operation by Security Mediator does not provide plain text. The SEM-decrypted content is still

encrypted with RSA strength encryption. The mediator transmits the SEM-decrypted data to the client. The client combines the SEM-decrypted data with the Client-decrypted data to access the plain text. If the client is revoked, the mediator does not compute or transmit SEM-decrypted contents. Without the mediator's participation, the client cannot compute the plain text.

Employing the mRSA algorithm directly to mobile networks to allow them to share a huge size of data creates performance issues. For instance, if the encrypted data is about 1 GB, employing mRSA requires sizable bandwidth to transfer the data between the mediator and the client. To overcome this limitation, Concord framework uses the mRSA protocol to transfer secret key algorithm for cryptographically securing the data.

**Concord Components**

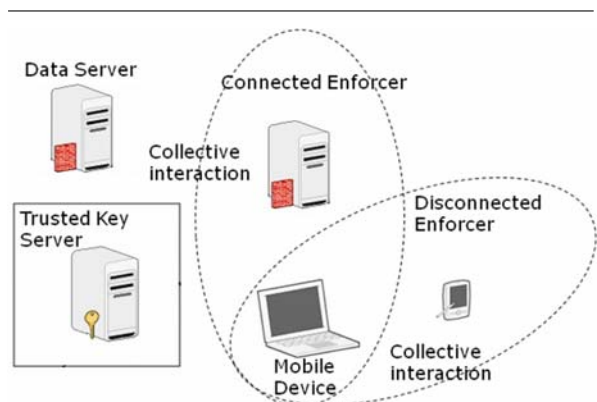
In this section, we describe the various components of the Concord Framework. Concord assumes that the infrastructure secures and maintains infrastructure-based entities.

Figure 2 shows the different components of Concord. The trusted entities are:

1. Trusted Key Server
2. Connected Enforcer

And un-trusted entities include:

3. Disconnected Enforcer
4. Data Server and
5. Mobile Device



**Figure 2:** Concord Components – Data Server, Disconnected Enforcer (D-Enforcer) and Mobile Device are un-trusted where as the Key Server and the Connected Enforcer (C-Enforcer) are completely trusted. The collective interaction among one of the enforcers and the Mobile Device can decrypt the data.

**Trusted Key Server**

The Trusted Key Server generates cryptographic keys for other components and for the data. To provide higher security, Concord partitions the data into blocks referred to as Data Units. Each Data Unit is encrypted with a secret key algorithm using Data Unit

Keys. A Data Unit can be the entire disk volume, a directory or individual files. A flexible Data Unit allows multiple security levels depending upon data sensitivity. The advantage of the design allows a compromise be confined to a single unit. The Trusted Key Server stores the Data Unit Keys for other entities.

In addition to Data Unit Keys, the Trusted Key Server creates mRSA keys. When a new mobile device joins the organization, the mobile device creates an mRSA keyset. The Trusted Key Server stores the public key and passes the other keys to the enforcer and the client. The Trusted Key Server creates a secondary multiple mRSA key pair if a Disconnected Enforcer is involved.

**Connected Enforcer**

The Connected Enforcer (C-Enforcer) is a trusted infrastructure entity where the Data Unit access policies are enforced. The C-Enforcer is available over either wired or wireless networks. As shown in Figure 2, the mobile device can access the plain-text only after interacting with a C-Enforcer. If the mobile device is revoked, C-Enforcer does not provide the SEM-decrypted Data Unit Keys, disabling the mobile device's ability to view plain text.

C-enforcer has an additional benefit: it can maintain the list of Data Units accessed by a mobile device. Due to this benefit, the organization determines a user's data access patterns as and when they request Data Unit Keys. This determines the subset of data that has been accessed by the users of mobile devices. In the event of a security breach, user access history is critical for an organization to comply with regulatory standards. We assume that the C-Enforcer stores the logs in a tamper-proof storage.

C-Enforcer and Trusted Key Server can both be a part of the same entity or can be separated. We design the two as different entities to demonstrate the different functionality.

**Disconnected Enforcer**

To support mobility, Concord supports a Disconnected Enforcer (D-Enforcer) that functions as an enforcer in the absence of the C-Enforcer. The D-Enforcer, however, is an un-trusted entity that caches only a part of the Data Unit Keys to enables local reads and writes. The data on the D-enforcer is not monitored as rigorously as C-enforcer as: a) it does not maintain data request logs; and b) any key revealed to it is considered to have been viewed by the users.

Storing the SEM-decrypted content along with encrypted data may lead to data and key compromise. Thus, D-enforcer (e.g., a PDA) provides a second layer of security when data is stored on a mobile system. We assume that a mobile device is a laptop and the D-Enforcer usually is a PDA. By mandating the use of a D-Enforcer, the risk of data loss is distributed among multiple entities as the loss of both is less likely.

### Data Server

The Data Server stores the encrypted data and can function with any file system. Concord assumes that the Data server is not a trusted entity and that it can be replicated to provide high availability. By separating data access from data governance, organizations can provide highly replicated service while minimizing the overhead of access control. Storing encrypted Data Units on an infrastructure's data servers provides a two-fold advantage:

- the Data Server compromise does not imply data compromise; and
- the data is available to the users upon the loss of the Mobile Device.

### Mobile Device

A mobile device, such as a laptop, maintains the data in an encrypted format to secure data from physical loss. The encrypted Data Unit Keys are stored on the Mobile device as well as on an Enforcer. The Data Unit Keys can be transmitted securely using the mRSA protocol. The Data Unit Keys should be securely handled on the Mobile Device and deleted when the need for unencrypted data no longer exists.

### Concord Protocol

Concord provides a mechanism to download and store encrypted data on the Mobile Device. In a connected mode, the Mobile Device interacts with the C-Enforcer; in the absence of the C-Enforcer, the D-Enforcer doubles up as an enforcer. Both the Enforcers have different mRSA key pairs as the C-enforcer's keys are organizational as compared to the D-Enforcer.

As explained in the above sections, each Data Unit is encrypted with a Data Unit Key generated using a symmetric key algorithm. The use of the symmetric key algorithm avoids transferring huge data using the mRSA algorithm. This subsection discusses the design of Concord, that is, mRSA key setup, reader and writer architecture using the C-Enforcer and the D-Enforcer.

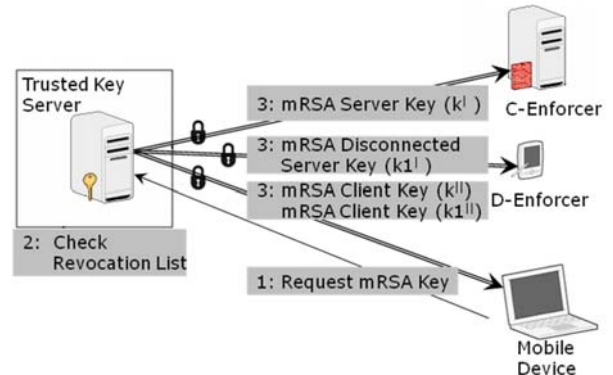
### Key Setup

In the Concord framework, key setup is required in two cases: first, when a new Mobile Device joins the Concord framework; second, when new Data Unit Keys are transferred to the Mobile Device. As mentioned in previous sections, the Concord framework uses mRSA, 2-out-of-2 threshold cryptography.

#### mRSA Key Setup

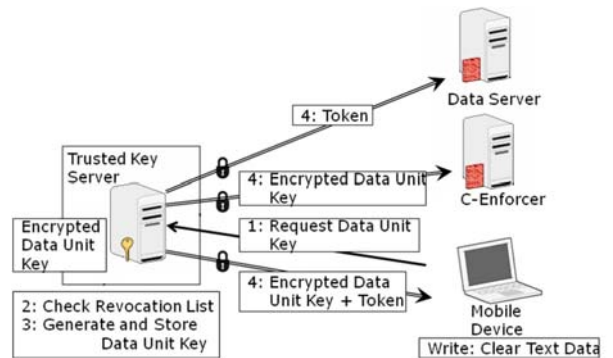
Figure 3 shows the process of setting up the mRSA keys for a Mobile Device. When a new Mobile Device joins the Concord Framework, the Mobile Device requests the Trusted Key Server for mRSA keys to be generated. The Trusted Key Server performs a validation check to ensure that the client has not been previously revoked by checking a revocation list. If the Mobile Device has been revoked, the mRSA keys are not generated. On the other hand, if the Mobile Device has not been revoked, mRSA keys are

generated for the client. If the C-Enforcer is involved, the mRSA keys are transmitted to the Mobile Device and the C-Enforcer. If the D-Enforcer is required in addition to the C-Enforcer for a Mobile Device, the mRSA keys are communicated securely to the C-Enforcer, the D-Enforcer and the Mobile Device. The creation of an mRSA key can occur only after an mRSA key is obtained for the C-Enforcer.



**Figure 3:** mRSA Key Setup for a new Mobile Device.

When a Mobile Device requests for mRSA Key, the Trusted Key Server checks the revocation list. If the client is revoked, it does not receive the mRSA keys else the mRSA keys are transmitted to other entities.



**Figure 4:** Obtaining Data Unit Keys. The Mobile Device requests the encrypted Data Unit Keys for the reader architecture. In the writer architecture, the Trusted Key Server provides the Mobile Device with a new Data Unit Key.

#### Data Unit Keys

Figure 4 demonstrates the mechanism for a Mobile Device to obtain a new Data Unit Key. When a Mobile Device requests the Trusted Key Server for a new Data Unit Key, the Trusted Key Server performs validation checks and generates the keys. The Data Unit Key is stored on the Trusted Key Server and the encrypted Data Unit Key is transmitted to the C-Enforcer and Mobile Device. In addition, the Trusted Key Server creates a token which is transmitted to the Mobile Device and the Data Server. When the Mobile

device wishes to put the encrypted Data Unit on the Data Server, the Data Server verifies the token and stores it.

**Download Encrypted Data Units**

Concord can be configured for use with any underlying file system. As discussed in the above sections, the Data Server stores encrypted Data Units. We assume that the underlying file system provides basic access control allowing only authorized users to download the encrypted data. Once the user is authenticated, the encrypted data can be downloaded to a Mobile Device and stored for future access.

**Read and Write Enforcement Protocol Using the C-Enforcer**

When a user creates data, Concord provides a secure mechanism to upload the Data Unit to the Data Server. Assuming that the mRSA keys have been assigned, upon the request for a Data Unit Key, the Trusted Key Server communicates the encrypted Data Unit key to the C-Enforcer and the Mobile Device.

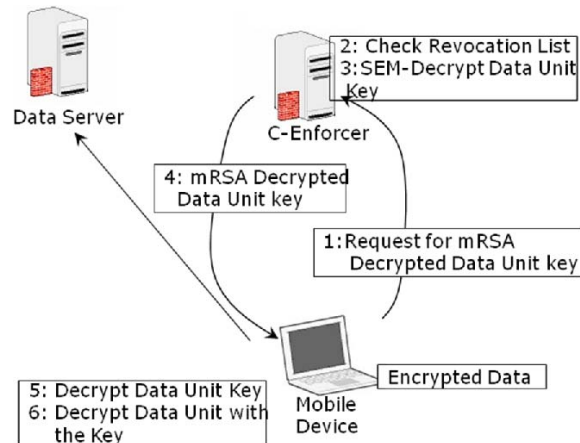
Figure 5 and Figure 6 shows the process of decrypting the Data Unit key through the interaction between the Mobile Device and the C-Enforcer. When the Mobile Device requests the decryption of the Data Unit key, the C-Enforcer checks the revocation list. Upon successful validation, the C-Enforcer decrypts the Data Unit Key using the mRSA private key it holds. The SEM-decrypted Data Unit Key is then transmitted to Mobile Device. The Mobile Device simultaneously decrypts the encrypted Data Unit key using its own private key. Finally, the Mobile Device combines the SEM-decrypted and Client-decrypted Data Unit Keys to retrieve the Data Unit Key. In the reader architecture (Figure 5), the client is able to read the content in the encrypted Data Unit using the Data Unit key obtained.

In the writer architecture shown in Figure 6, the Mobile Device requests for a Data Unit Key to encrypt the data. Once the data is encrypted, the user places the data onto the data server. If the Mobile Device requests to store the data as a part of a new Data Unit, it requests from the Trusted Key Server a single Data Unit Key that can be used to encrypt the clear data. A new Data Unit Key is created based on the protocol described earlier. When storing the Data Unit, the Data Server checks the randomly generated token. Token verification is used to indicate that a new Data Unit has been created and disallows creation of Data Units with the same name and location. Concord allows data writes only when the Mobile Device is in a connected mode. We explain the disconnected mode in the next section. In comparison, if the data needs to be added to an existing Data Unit, the Mobile Device retrieves the Data Unit Key, encrypts the Data Unit with the Data Unit Key, and stores it on the Data Server.

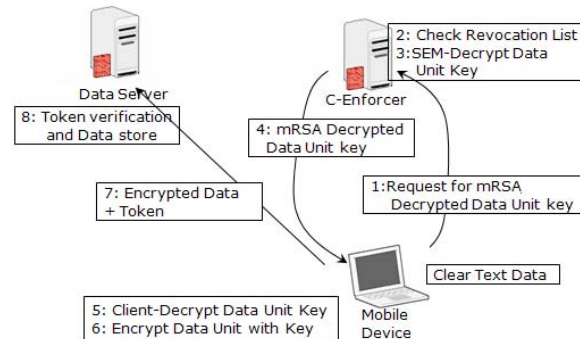
**Reader Architecture for D-Enforcer**

In a disconnected mode, the C-Enforcer delegates data enforcement to the D-Enforcer. In comparison to the C-Enforcer, the D-Enforcer maintains a

subset of Data Unit Keys. We believe that the subset of keys available to the D-Enforcer can be determined by an organizational policy regarding the number of Data Units to be shared based on the data sensitivity. Such a policy does not necessarily safe-guard against a hacker for which the keys are available on the D-Enforcer, it only identifies this data. We discuss key policy in the next section.



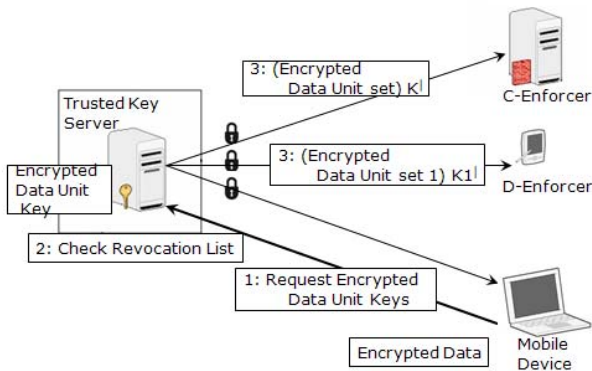
**Figure 5:** Decrypting the Data Key to accessing a Data Unit Key for reader architecture.



**Figure 6:** Decrypting the Data Key to encrypt data for writer architecture.

Figure 7 shows the Mobile Device requesting a subset of Data Unit Keys from the Trusted Key Server. The Trusted Key Server performs the validation checks and transfers the complete set of Data Unit Keys to the Mobile Device and the C-Enforcer. In contrast to the C-Enforcer, the D-Enforcer and the Mobile Device receive a subset of the Data Unit Keys. The Mobile Device maintains two sets of encrypted Data Unit Keys: one encrypted with the mRSA key of the C-Enforcer and the other with that of the D-Enforcer.

Upon request from the Mobile Device for a Data Unit key, the D-Enforcer decrypts the key and transmits the SEM-decrypted key to the Mobile Device. The Mobile Device uses the SEM-decrypted key to compute the complete Data Unit Key. For regulatory compliance, all the data whose Data Units keys were given to D-Enforcer are assumed as accessed.



**Figure 7:** The Trusted Key Server distributes the Data Unit Keys to the Enforcers. The C-Enforcer maintains the complete set of Data Unit Keys whereas the D-Enforcer maintains smaller subset of Data Unit Keys.

**Security in Concord**

In this section, we discuss the security provided by Concord. We primarily discuss: (a) revocation of a Mobile Device; (b) granularity of the Data Unit and (c) D-Enforcer Data Unit Key subset policy. Table 1 discusses how Concord secures data in case of different compromises. We assume that the data on the mobile device is decrypted in a secure location. Concord cannot secure data that has been decrypted and copied onto devices that it cannot monitor. Upon loss of the Mobile Device a part of the keys along with the data would be available to the attackers. However, the data is encrypted and the keys are still encrypted. To decrypt the key, the second private key from an Enforcement Point is required. If the attacker has the D-Enforcer, a part of the keys could be decrypted and the other part of data is provably secure. The other option would be attack the C-Enforcer, which would be monitored. Therefore, without an Enforcer, the data is secure.

**Revocation in Concord**

Concord supports: a) revocation of read access to previously unread Data Units and b) revocation of the write access to all Data Units. If a Mobile Device needs to be revoked following a compromise, administrators can perform the revocation at the Trusted Key Server. The Trusted Key Server securely transfers the revocation list to the C-Enforcer immediately upon revocation. Further data access to the unread data on

the affected Mobile Device is not possible as the C-Enforcer will not provide the Data Unit Keys to the Mobile Device.

In the disconnected mode, the D-Enforcer stores a subset of the Data Unit Keys. A key distribution policy, enforced at the Trusted Key Server, determines the number of keys that can be stored at the D-Enforcer for Mobile Devices. The D-Enforcer stores a smaller subset of keys that can be used to decrypt the Data Units even after revocation of the Mobile Device. However, as the users cannot access the other Data Unit Keys, the compromise is contained. Future versions of the Data Units are encrypted with new Data Unit Keys, to protect them from future read access.

When D-Enforcer needs new keys, the old set of keys that was cached should be cleared and a new set can be provided. In such a case, the old set of Data Units is assumed to have been exposed to the user.

**Granularity of the Data Unit**

Partitioning the Data into multiple Data Units determines the desired security level when employing the Concord framework. When a potential compromise occurs or the data needs to re-encrypted following a user revocation, data partitioning allows re-encryption of a smaller subset of the data, instead of encrypting, say, the entire volume. The granularity of the Data Units can be configured by the organization and will dictate the level of security. The three available granularities for Data Units are:

- File-granular;
- Directory-granular; and
- Volume-granular.

File-granular configuration creates a Data Unit key for each file. This is the highest level of security that requires a large number of keys. For example, our experiment shows that for 77,900 files, about 3 MB is required to store the keys. We believe that this is acceptable as both the C-Enforcer and the Trusted Key Server are dedicated systems used to store keys for multiple Mobile Devices.

Directory-granular configuration provides security by encrypting the files in a particular folder; however, the sub-folders are encrypted using another Data Unit key. This granularity provides lesser security compared to the File-granular configuration, as multiple files share the same key. For the above example, we note that there were about 7,251 folders taking up about 0.3 MB.

Threat	How Concord helps
1. Insecure Data Server	Data on the Data Server is secure - as it is encrypted
2. Loss of Mobile Device	Data on Mobile Device is secure - as it is encrypted
3. Loss of D-Enforcer	Data Keys is secure - as it is encrypted
4. Loss of both Mobile Device and D-Enforcer	List of Compromised Data is available. Rest of the data is provably secure.
5. Disgruntled Users	List of Data accessed is available

**Table 1:** How Concord is able to secure the data for different compromises.

The volume granularity configuration involves use of a single key for the entire volume, providing the least security available with Concord. Such a system is similar to encrypt-the-disk system.

Each level of security comes with the cost of encryption and decryption. For example, if an organization wants to maintain a high level of security, it would imply that the organization needs to employ File-granularity, which would result in high costs in terms of storage and encryption and decryption operations. On the contrary, if the organization would like to maintain minimal security, the volume-granular can be used, which requires a minimal amount of time to decrypt the data. We suspect that a reasonable balance between security and cost would be to employ directory-granularity for the data-units.

#### ***D-Enforcer Data Unit Key Subset Policy***

As discussed in previous sections, the number of encrypted Data Unit Keys cached by the D-Enforcer is determined by organizational policy. Upon successful transmission of these keys to the D-Enforcer, the Trusted Key Server transmits the list of keys revealed to the C-Enforcer for regulatory compliance. For all the Data Units for which the encrypted Data Unit Keys are revealed to D-Enforcer, the user is assumed to have accessed the data for regulatory compliance. Therefore, a policy to restrict the amount of files that users can access in a disconnected mode is required to provide highly secure systems. For example, if a disgruntled user is known to have checked out a huge number of Data Unit Keys over time, it would imply that a huge amount of data might have been compromised. We suggest that such a policy should be designed on a need-to-know basis.

### **Concord Implementation**

In this section we discuss the implementation details of each of the components of the Concord framework.

#### **Trusted Key Server Implementation**

The Trusted Key Server requires administration to set the levels of security. For instance, apart from revocation, Trusted Key Server allows Data Unit key generation and distribution. All key-requests mandate Trusted Key Server to push keys to C-Enforcer or D-Enforcer. The revocation list at the Trusted Key Server should be kept updated and updated at C-Enforcer to discontinue further key requests.

The Trusted Key Server serves the following requests:

- The mRSA key setup, including checking the revocation list and generation of unique mRSA key pairs for both the Enforcers and the Mobile Device.
- Generation of the Data Unit Keys using the Advanced Encryption Standard (AES) standard.

The Trusted Key Server stores the mRSA keys for each Mobile Device. The Trusted Key Server maintains the list of revoked Mobile Devices. Further, the Trusted Key Server maintains the list of the Data Units and their corresponding Data Unit Key for each Mobile Device.

#### ***C-Enforcer Implementation***

The C-Enforcer's communication with the Mobile Device is the primary mode of request for a Data Unit key that requires mRSA decryption. The C-Enforcer has been implemented in Java. The communication of the C-Enforcer with various components has been developed using the Java's socket library. The current Concord prototype allows only a single Data Unit Key to be decrypted per request.

The encrypted Data Unit Keys are stored on the Trusted Key Server as a single data structure (a Java HashMap). The data structure keeps a mapping between the hash of the Data Unit's storage path and the associated Data Unit Keys. We implemented this feature in Concord using SHA1 that generates a 20 byte output.

#### ***D-Enforcer Implementation***

We implemented the D-Enforcer using Java ME running on a PDA. The Java ME Virtual Machine (JVM) was CrEme 4.0 [5] compliant with J2ME Connected Device Configuration (CDC) 1.0 specification based on JDK 1.3.1. The D-Enforcer stores encrypted Data Unit Keys in a Java HashMap. Storing hash of the data as an identifier to associate the encrypted file keys reduces the in-memory data structure size on D-Enforcer.

For the implementation, D-Enforcer has two services on the active port in a) receiving and b) sending mode. The communication between the Trusted Key Server and the D-Enforcer is in the receiving mode; allowing the data to be pushed from the Key Server to the D-Enforcer as and when required. The communication between the D-Enforcer and the Mobile Device is in the sending mode; allowing the D-Enforcer to push the data to the Mobile Device. Such a mechanism has been designed to reduce the amount of services run on the D-Enforcer to reduce the cost of operations on the D-Enforcer.

#### ***Data Server Implementation***

The Data Server prototype has been implemented using Java. The functionality can be easily provided using any file system such as NFS or AFS. The Data Server interacts with the Mobile Device over two channels: the Control Channel and the Data Channel. The Control Channel serves the specific goal of controlling the data that needs to be transferred where as the Data Channel is used to transfer itself. The Control Channel sets up the server socket for the Data Channels to open multiple sockets for streaming. The socket implementation supports a large number of files that are streamed in parallel. The streaming



mechanism employs a thread pool to stream one file per thread.

### Mobile Device Implementation

The Mobile Device is a temporary store for encrypted Data Units. We implement the Mobile Device interface using Java. We designed a command line interface which allows the users to register, login, conduct mRSA key setup, request encrypted data, decrypt content and lastly, logout.

### Experiments and Results

This section discusses Concord's performance. The performance analysis involves (a) Performance comparison of the C-Enforcer versus that of the D-Enforcer; (b) Data Unit Key Generation time; (c) Data Unit Key Distribution time for both the C-Enforcer and the D-Enforcer.

#### Experiment Setup

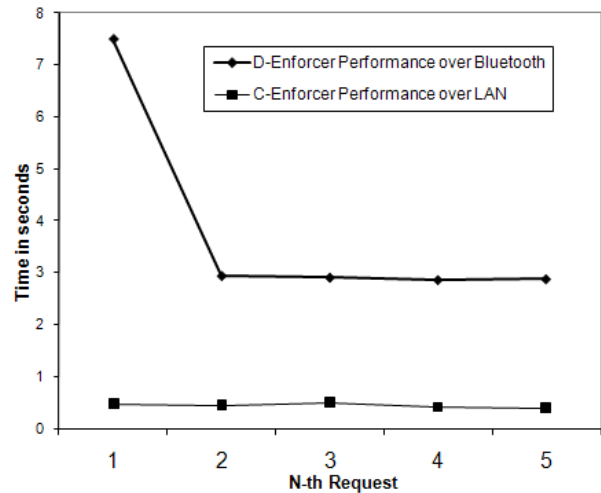
All of the experiments have been performed using the following devices and networks with specified configurations. The Trusted Key Server was running on Intel Pentium 4 CPU 3.00 GHz with 2 GB RAM running Microsoft Windows XP Professional version 2002 with service pack 2, whereas the C-Enforcer and Data Server were running on different machines with a similar configuration. The Mobile Device used for performance analysis was a Intel Pentium M 1700 MHz 1 GB RAM whereas the D-Enforcer was a Intel ARM HP iPAQ PocketPC h4300 PDA with 64 MB RAM running Windows CE.net Version 4.1. The Local Area Network bandwidth was 100 Mbps with a delay of about 0.1-0.2 milliseconds and Bluetooth link was 700 Kbps with a delay of about 60-70 milliseconds. The Java Virtual Machine (JVM) on the PDA is CrEme 4.0 [5] compliant with J2ME Connected Device Configuration (CDC) 1.0 specification and based on JDK 1.3.1. The CDC is the J2ME configuration that supports full Java implementation on PDA HP iPAQ PocketPC.

#### Performance Comparison Between C-Enforcer and D-Enforcer

As indicated above, the D-Enforcer communicates with the Mobile Device over a low-bandwidth Bluetooth communication link. On the other hand, a C-Enforcer is implemented over a high bandwidth connection and has additional computational resources.

Figure 8 shows the total decryption time for a single Data Unit Key. The total time includes the time taken to partially decrypt the Data Unit Key on the C-Enforcer and on the D-Enforcer; finally, the partially-decrypted data from the Enforcer point and the Mobile Device is used to completely decrypt the Data Unit Key. The performance of D-Enforcer is improved by reusing the socket that is used through the socket pooling. The D-Enforcer shows significant performance improvement when the socket is reused. The average one-time socket connection setup time was found to

be 4.39 seconds. Since the D-Enforcer is dedicated to serving a single Mobile Device, the communication socket can always remain open. That is, no socket connection timeout is set.



**Figure 8:** Performance analysis of D-Enforcer and C-Enforcer involves the measurement of the time taken to decrypt a Data Unit Key (48 bytes) against the nth decryption request. The average one-time socket connection setup time for Bluetooth was found to be 4.39 Seconds.

#### Concord Setup Performance

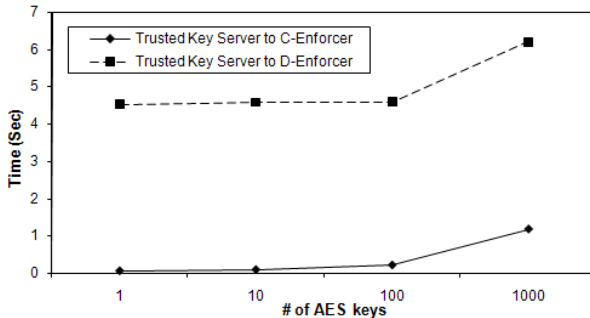
Concord setup requires time to distribute Data Unit Keys (AES keys) to the Mobile Device and the C-Enforcer as well as the D-Enforcer. This section shows (a) Trusted Key Server performance and (b) the performance of file key distribution to the Enforcers.

#### Trusted Key Server Performance

The time taken to generate Data Unit Keys is proportional to the number of keys that are being generated; however, as the number of keys increases, the time taken to generate the keys is relatively constant. Our experiments show that the time required for generation of a key decreases exponentially as the number of keys generated increases. When one key was generated, 0.46 Sec per key was required whereas 1.06 Sec per key was required when 1,000 keys were generated. We provide an optimization for Trusted Key Server performance by creating multiple keys. These keys are provided to the users upon request.

#### Performance of Data Unit Key Distribution to Enforcers

In Concord, Data Unit Keys are transmitted from the Trusted Key Server to the Enforcers and to the Mobile Device. The amount of time taken to send the Data Unit Keys from the Trusted Key Server to the Enforcers is shown in Figure 9. For the D-Enforcer, using Bluetooth, our experiments show that about 4.39 seconds are required to create a socket. The time taken to transmit about 1000 keys is about 6.5 seconds.



**Figure 9:** AES Key distribution time from Trusted Key Server to D-Enforcer and C-Enforcer.

The experiment reiterates the fact that the time taken to communicate the Data Unit Key is dependent on the bandwidth of the communication medium. The Bluetooth link is a very low bandwidth in comparison to that of LAN.

### Conclusion

Concord framework allows organizations to monitor the data accessed on a mobile device as an infrastructure-level service by distributing the trust among multiple entities. With an ability to provide irrefutable data access history, Concord data access framework supports regulatory compliance standards. The knowledge of the data accessed by the users on their Mobile Device enables the organization to demonstrate their adherence to mandated regulatory compliance such as HIPPA and the Feinstein Bill.

Concord provides a novel secure data access and monitoring framework with data being stored on the employee's mobile devices. Concord requires an enforcement server, either the connected C-Enforcer or the disconnected D-Enforcer, to be involved in decrypting the data on user's mobile device. With the help of the enforcement server, Concord enables the organization to effectively monitor data access and revoke unwanted clients. Concord identifies critical organizational data that has been accessed by a disgruntled user's access and theft or loss of the mobile device by handling data securely during storage and wire while in a cryptographic format. Concord employs a 2-out-of-2 cryptographic technique, called mRSA, which encrypts data while in storage or when being transferred on wire.

### Future Work

The Concord implementation discussed in this paper has been implemented as a prototype. During this prototype implementation, our measurements demonstrated that the high cost involved in encryption and decryption using mRSA. We are working on another threshold cryptographic solution that would be able to reduce the number of operations to increase the efficiency. We plan to perform an extensive I/O evaluation to evaluate the costs involved.

### Acknowledgments

This research is partly supported by TIAA CREF Biggs Faculty Fellowship. We also would like to thank Pratik Thanki for his help in setting up the experiment.

### Author Biographies

Gautam Singaraju is a doctoral candidate at University of North Carolina at Charlotte. As part of the research efforts, he focuses on secure IT infrastructure design; specifically on system administration issues. Some of his work is in: 1) securing email infrastructure, 2) infrastructure design for regulatory compliance – premise-aware data protection services, 3) intrusion detection systems, and 4) Virtualization-based secure infrastructure. Gautam can be reached at gsingara@uncc.edu.

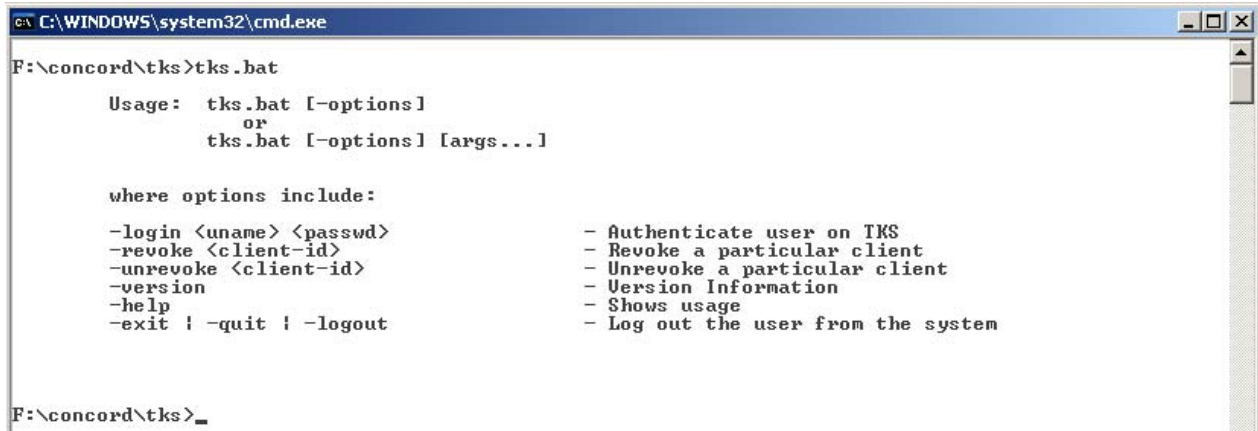
Brent Hoon Kang is currently an assistant professor at the College of Computing and Informatics at UNC Charlotte. He leads the Infrastructure Systems Research Lab at UNCC which explores the secure architecting of large-scale infrastructure systems. Through the lab, he has worked on (1) securing email infrastructure, (2) research on malware and botnet enumeration and (3) topics such as premise-aware data protection infrastructure, and IT infrastructure design for regulation compliance. Recently he has been working with a group of IA students in researching malware and bot infection behavior as part of the Global HoneyNet Research Alliance. As part of his efforts on Information Assurance (IA) education program, he has been developing the hands-on cyber exercise components that foster students' creativeness and problem solving skills for IT systems design and defense. He received his Ph.D. from the University of California at Berkeley. Hoon can be reached at bbkang@uncc.edu.

### Bibliography

- [1] Bluetooth Wireless, "Bluetooth Wireless Technology," *Bluetooth SIG – The official website for the Bluetooth short range wireless connectivity standard*.
- [2] Bluetooth Protocol and Bluetooth PAN (Personal Area Network), *Bluetooth Special Interest Group (SIG)*.
- [3] Corner, M. D. and B. D. Noble, "Zero-Interaction Authentication," *Proceedings of MOBICOM*, Atlanta, GA, September, 2002.
- [4] Corner, M. D. and B. D. Noble, "Protecting Applications with Transient Authentication," *Proceedings First International Conference on Mobile Systems, Applications and Services*, 2003.
- [5] CrE-ME 4.0 (J2ME-CDC) Beta, Java Virtual Machines for Java Based Embedded Devices, *J2ME CDC 1.0 Specification*.
- [6] Ding, X. and G. Tsudik, "Simple Identity-Based Cryptography with Mediated RSA," *Proceedings of CT-RSA '03*, LNCS 2612, pp. 193-210, Springer, 2003.

- [7] Feinstein, Bill, "E-Loan and ING Direct Endorse Feinstein Identity Theft Legislation, First Two Major Companies to Endorse Feinstein Bill," June, 2005.
- [8] Kabay, M. E., "Insider Attacks Are a Thorny Problem, Insider Computer Crime is Difficult to Defend Against," *Network World*, August, 2003.
- [9] Kallahalla, M., E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus - Scalable Secure File Sharing on Untrusted Storage," *Proceedings Second USENIX Conference on File and Storage Technologies (FAST)*, USENIX, March, 2003.
- [10] Libert, B. and J. Quisquater, "Efficient Revocation and Threshold Pairing Based Cryptosystems," *Proceedings 22nd Annual Symposium on Principles of Distributed Computing PODC '03*, July, 2003.
- [11] Mitchell, R. L., "Decline of the Desktop," Computer World New Story, *Computer World*, September 26, 2005.
- [12] Strandboge, J., "Encrypted NFS with OpenSSH and Linux," *SysAdmin Journal for UNIX and Linux System Administrators*, March, 2002.
- [13] Standards for Privacy of Individually Identifiable Health Information, *Security Standards for the Protection of Electronic Protected Health Information*, April, 2003.
- [14] Vanrenen, G. and S. W. Smith, "Distributing Security-Mediated PKI - Public Key Infrastructure," *EuroPKI 2004*, Springer-Verlag, LNCS 3093, pp. 218-231, June, 2004.
- [15] Rivest, R., A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, Vol. 21, Num. 2, pp. 120-126, February, 1978.

## Appendix



```

C:\WINDOWS\system32\cmd.exe
F:\concord\tks>tks.bat

Usage:  tks.bat [-options]
        or
        tks.bat [-options] [args...]

where options include:

-login <uname> <passwd>          - Authenticate user on TKS
-revoke <client-id>              - Revoke a particular client
-unrevoke <client-id>            - Unrevoke a particular client
-version                          - Version Information
-help                              - Shows usage
-exit | -quit | -logout          - Log out the user from the system

F:\concord\tks>_

```

**Snapshot I:** The TKS interface for administrators allows the revocation of malicious client.



```

C:\WINDOWS\system32\cmd.exe - deploy-tks.bat
F:\concord\tks>deploy-tks.bat
TKS service started as: tks@152.15.98.50:2000

```

**Snapshot II:** The Command that starts the TKS server.

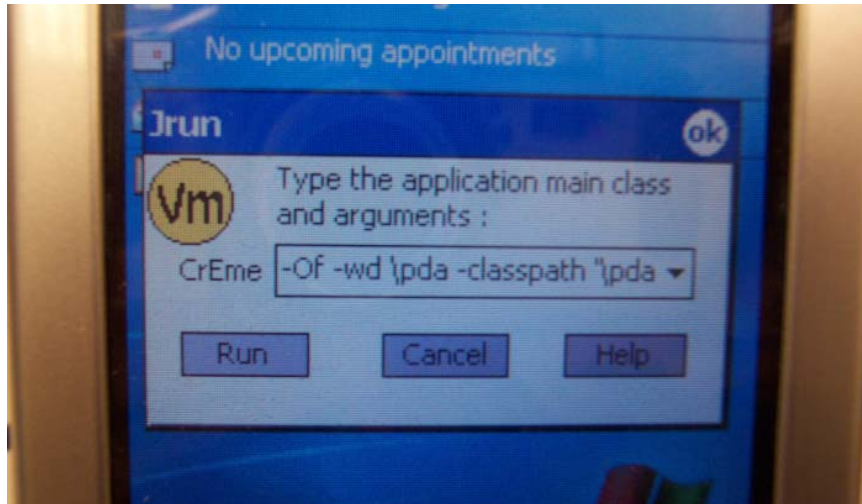


```

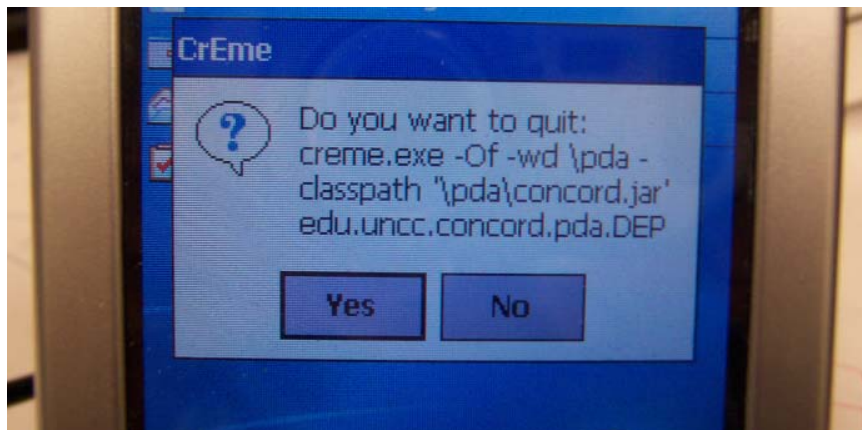
C:\WINDOWS\system32\cmd.exe - deploy-cep.bat
F:\concord\cep>deploy-cep.bat
CEP service started as: cep@152.15.97.38:2000

```

**Snapshot III:** Command that starts the CEP server.



**Snapshot IV:** Command that starts the DEP server on a mobile device using CrEme jvm. The command is: `CrEme.exe -Of -wd \pda -classpath '\pda\concord.jar' edu.uncc.concord.pda.DEP`.



**Snapshot V:** The dialog box message that gets displayed when the DEP server is running on a mobile device.