**Secure Mobile Database Applications**

Mobile devices are gradually becoming commonplace. The computational and networking power of mobile devices is constantly increasing and new technologies are integrated into them to support new functionalities and services. On the other hand, the field of databases and more generally data management is also expanded with new services and applications. Several modern database management systems support small-footprint databases that can be executed on mobile devices and admit disconnected computing and synchronization with a central database. We call an application that comprises a server with a central database and several autonomous mobile clients with replicated parts of the database a mobile database application.

One of the most important issues of modern computing systems is the provision of sufficient security and privacy guarantees for user data. In the field of databases and database management systems, security is a well-studied subject. However, in the case of a mobile database application, there are additional security challenges due to the distributed nature of the application and the hardware constraints of mobile devices. Achieving a sufficient level of security for such a platform is an important problem that has to be addressed. For example, data privacy and confidentiality are identified as critical open issues and research directions in mobile databases.

In this work, we consider mobile database applications and focus on the security issues that arise in this context. For this aim, we present a case study of a secure mobile database application. In particular, we design, develop and test an electronic announcement board. A database server is used for the central storage of all application data, while small-footprint relational databases are used on mobile clients. We identify a set of security issues and show how to handle these issues on the prototype mobile application.

The rest of the paper is organized in the following way. The mobile database application is described in Section 2. Security techniques are presented in Section 3. The implementation and the test platform are described in Section 4. Section 5 presents possible attacks and how they are faced by the application and Section 6 contains a final discussion.

**1 The Mobile Database Application**

We consider the following mobile database application (MDA): An electronic announcement board where authorized users can publish and/or read announcements. There are two types of users of the announcement board, author users and read-only users. The rights of a user are determined by its type: An authoring user has the right to create new announcements and to modify or delete announcements authored by himself. A read-only user has the right to read all announcements. The announcements are centrally stored in a database server and the users, author users, and read-only users can use mobile devices to perform their application-related operations remotely. The core of the application is built on mobile database technology. As shown in Figure 1, the application uses the client-server model. From the user’s point of view, there are two main application components: An authoring tool for authoring announcements and a viewer to access all announcements. Moreover, if the announcements are intended for public access, then read-only access can also be provided through a web interface.

In this work, we define a mobile database as a small-footprint database that is installed on a mobile device. Most commonly the local database is a replica of a part of a central database that is installed on a server computer.

Major database management system (DBMS) vendors like Oracle, IBM, and Microsoft, are providing mobile extensions for their database servers. We have chosen *a Pocket PC with Windows Mobile 5.0 and SQL Server 2005 Mobile Edition* as the computing platform for our mobile application. However, corresponding technologies of other vendors could also be used. See for example Table 1.

**Table 1.** Indicative combinations of mobile platforms and mobile DBMSs

|  |  |  |
| --- | --- | --- |
| **Mobile** | **Operating System** | **Mobile DBMS** |
| **Device**Pocket PC | Symbian | Oracle Mobile |
| PocketPC | Windows | MS SQL Server Mobile |
| Palm, Pocket | Symbian, Windows | EditionIBM DB2 Everyplace |
| PCPalm, Pocket | MobileSymbian, Windows | Sybase SQL Anywhere |

PCMobile

**1.2 Motivation**

In a mobile database application, a part or a replica of the database is locally installed on the mobile device. This is a significant difference compared to a conventional client-server application where all data is centrally stored in a database server. The approach with a mobile database provides the necessary autonomy for the mobile device to work independently from the central database. The client application can work with the mobile database asynchronously and needs to connect to the central database only when it is necessary to synchronize. This approach has several advantages compared to a conventional approach where the clients do not use local storage:

— Flexibility and Reliability: Asynchronous operation makes the application more flexible and tolerant to network failures.

— Efficiency: Except for the synchronization steps, for all other operations, the client has immediate access to the data since it is locally stored on the mobile device.

— Enhanced security: Disconnected computing reduces the total time that the mobile device is exposed to potential attacks over the network.

— Energy efficiency: The mobile device has to operate its network system, hardware, and software, only during the synchronization operations.

— Reduced fees for network usage: This holds in the case where the usage of the communication link is charged. If the network link up-time is charged then the benefits are obvious. However, even if only the network traffic is charged, the decentralized approach of a mobile database can still reduce network fees. In this case, the cost decrease is achieved by reducing the traffic volume between the mobile device and the

server.

**1.3 Architecture**

The architecture of the mobile database application (MDA) is shown in Figure 2. The application uses the client-server model1. The server side of the application has three main components: A central database, a server agent, and a web server. The central database provides the central storage place for all announcements. The server agent connects the central database with the web server. The web server provides the end-point of the communication link that is used to transfer data between the mobile and the central database. The client

side has also three main components: The client application, the client agent, and the mobile database. The client agent is responsible for the communication between the mobile database and the central database and between the client application and the mobile database. The client application is a mobile application with a graphical user interface (GUI) that provides the necessary interface to the users for using the application. The mobile database is a local small-footprint database on the mobile device which replicates an appropriate part of the central database.

The mobile database application has to use a communication link between the client and the server. The only requirement for the communication link is that it must support the secure hypertext transfer protocol (HTTPS). There are currently several different options for providing the communication link. The most important are;

Wireless Networks, Bluetooth, GPRS, and 3G.

At both endpoints of the communication link are agents of the mobile database management system. We tested our application with a wireless network connection and with a Bluetooth connection.

Android operating system is a stack of software components which is roughly divided into five sections and four main layers as shown below in the architecture diagram.



**Linux kernel**

At the bottom of the layers is Linux - Linux 3.6 with approximately 115 patches. This provides a level of abstraction between the device hardware and it contains all the essential hardware drivers like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.

**Libraries**

On top of Linux kernel there is a set of libraries including open-source Web browser engine WebKit, well known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.

**Android Libraries**

This category encompasses those Java-based libraries that are specific to Android development. Examples of libraries in this category include the application framework libraries in addition to those that facilitate user interface building, graphics drawing and database access. A summary of some key core Android libraries available to the Android developer is as follows −

* **android.app** − Provides access to the application model and is the cornerstone of all Android applications.
* **android.content** − Facilitates content access, publishing and messaging between applications and application components.
* **android.database** − Used to access data published by content providers and includes SQLite database management classes.
* **android.opengl** − A Java interface to the OpenGL ES 3D graphics rendering API.
* **android.os** − Provides applications with access to standard operating system services including messages, system services and inter-process communication.
* **android.text** − Used to render and manipulate text on a device display.
* **android.view** − The fundamental building blocks of application user interfaces.
* **android.widget** − A rich collection of pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons etc.
* **android.webkit** − A set of classes intended to allow web-browsing capabilities to be built into applications.

Having covered the Java-based core libraries in the Android runtime, it is now time to turn our attention to the C/C++ based libraries contained in this layer of the Android software stack.

**Android Runtime**

This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called **Dalvik Virtual Machine** which is a kind of Java Virtual Machine specially designed and optimized for Android.

The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine.

The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

**Application Framework**

The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications.

The Android framework includes the following key services −

* **Activity Manager** − Controls all aspects of the application lifecycle and activity stack.
* **Content Providers** − Allows applications to publish and share data with other applications.
* **Resource Manager** − Provides access to non-code embedded resources such as strings, color settings and user interface layouts.
* **Notifications Manager** − Allows applications to display alerts and notifications to the user.
* **View System** − An extensible set of views used to create application user interfaces.

**Applications**

You will find all the Android application at the top layer. You will write your application to be installed on this layer only. Examples of such applications are Contacts Books, Browser, Games etc.

**2 Security Issues and Techniques**

In this Section, we describe the security-related techniques that are applied in the mobile database application.

**2.1 Secure network connection**

The mobile database and the central database have to be synchronized at specific times. The synchronization is implemented in the system software of the mobile database and is performed over the HTTP protocol. Using HTTP has the significant advantage of using a widely available protocol and possibly the disadvantage that its performance may be lower than a proprietary protocol for the database synchronization operation. We have selected the secure HTTP protocol (HTTPS) to perform the necessary synchronization operations between the mobile and the central database. More precisely we use HTTPS with server and client authentication. This choice assures:

— Confidentiality of the data that is transferred.

— Authentication of the server computer.

— Authentication of the client's computer. Even though client authentication worked on the mobile platform we did not manage to apply it within the synchronization process of the mobile database. We believe that this is due to a shortage of the current system software and that will be overcome in the forthcoming versions.

**2.2 Encrypted local database**

The local database on the mobile device is encrypted and each time the user opens the mobile database, he has to enter his password. In case the mobile device is stolen or violated by an intruder, the data that is stored on the local database is not readable. The encryption algorithm is part of SQL Server Mobile Edition and unfortunately, we were not able to find documentation for the specific algorithm. We assume that the vendor does not simply rely on obscurity and that the encryption is based on one of the established symmetric key en

cryption algorithms. If the built-in encryption algorithm of the mobile database is considered insufficient, it is of course possible to implement this feature within the client application. The synchronization of the small footprint database that is installed on the mobile device with the central database is performed with database replication technology. For this purpose, there is an appropriate publication on the database server. A publication is the meta-data package of information about which data is replicated. The mobile database uses the publication of the database server for the synchronization operation. To connect to the publication an appropriate user account on the database server has to be used. This means that the application user has to be authenticated at the database server.

**2.4 Authentication at the web server**

As already noted, the communication between the mobile database and the central database is performed over HTTP. On the server side, the communication link is handled by a web server. Hence, it is possible to take advantage of standard web server authentication and require the user to authenticate at the web interface level. This requirement is very important since it protects the mobile database agent that is executed on the server side within the web server. Without web server authentication every network user would be able to contact the server-side agent by simply using the appropriate URL.

**2.5 Server-side mobile agent account**

Both endpoints of the communication link are handled by mobile database agents. During a synchronization process, the agent operations on the server side can either be executed by the default agent account of the server's operating system or in the context of a dedicated account of the server's operating system. We use a dedicated operating system account for the execution of the agent service. The account has been granted the minimum permissions that are necessary for its role. This decision satisfies the common security rule of granting minimum sufficient permissions.

**2.6 Separate user accounts for the authoring and the read-only application**

In case a user has to use the application both as an author of announcements and as a reader of all announcements we can either assign two accounts to the user, an authoring account and a read-only account or grant both functionalities to a unique user account. Even though the security of the application would not be lowered by using a unique account, we preferred to use two separate, dedicated accounts. This approach reflects more naturally the structure of the application. Application provided security

For authoring operations, each user has access only to his data. A set of database triggers implemented in the database server, check that the data manipulation operations of the user are valid. This check prevents all users from accidental or malicious modifications of data for which they have no authorization. More precisely, an author

— can create new announcements that are signed with his name,

— can delete or update announcements that are signed with his name, and

— has no access to announcements created/signed by other users.

The above functionality resembles in a loose sense the virtual private database technology (VPD) of Oracle [9].

**2.7 The read-only client**

The read-only part of the MDA is implemented as a separate client application. The read-only client provides access to viewing all announcements. We apply certain techniques to assure the security of the central database:

— The publisher of the database server that is used for the synchronization of the read-only application is defined to be read-only. Consequently, it is not possible to apply any modification to the central database from the read-only application.

— Read-only clients have no access to the main table of the central database. Instead, the read-only clients read the announcements from a replicated instance of the main table. A set of database triggers implemented in the database server keeps the replicated table always updated. In case an accidental or malicious modification of the data in the replicated table would occur, it would not affect the main table of the application.

**2.8 Communication between the servers**

The announcements are also available over HTTP as a web page. A dynamic web page with an aspx code gives a list of the announcements. The web server must have access to the database to read the data. For this reason, we have to deal with a common security issue in database-driven websites: Choosing the appropriate database account that the web server is using to access the database. We created a specific account in the database that has only one permission: To perform a selection on the replicated announcements table. This decision applies the principle of granting the minimum sufficient permissions.

**2.9 Client-side data encryption**

We also tested a common but very important feature, that of encrypting the user data in the database. Even though this feature is not directly relevant to the announcements application, we consider it very important for secure mobile database applications and more generally for secure database applications. The user gives a password to the client application and all his critical data is encrypted on the client side before it is permanently stored in the database. This encryption guarantees the confidentiality of the data against any database user including the local database administrators. The approach is very simple: The client application applies a symmetric key encryption algorithm, for example, AES, and stores the encrypted data in the database. When the user reads the data, he provides his password and the data is decrypted. We verified this approach and it works transparently as soon as the user has given his password. A shortage of the current mobile platform was that some library functions, like for example the function ”PasswordDeriveBytes”, were not provided by .NET Compact Framework v2.0. We overcame this problem by providing a hand-coded implementation of the required function that was absent.

**3 Implementation**

**3.1 Development**

The development platform for the MDA was Visual Studio 2005 with cross-compilation for Windows Mobile 5.0 and the .NET Compact Framework v2.0. The application is implemented in C#.

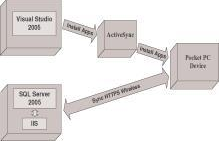
**3.2 Testing**

The mobile database application (MDA) has first been tested on a Windows Mobile Emulator and then on a real Pocket PC with Windows Mobile 5.0.

1. MDA on a Windows Mobile emulator. To execute MDA on an emulator, we used Visual Studio to start the emulator and install the application on the emulator. The emulator works as a real Windows Mobile Pocket PC. The network connection for the emulator is provided by ActiveSync. We tested all operations of the mobile application and they worked well.

2. MDA on a Pocket PC with Windows Mobile 5.0 (Figure 3). In this case, we use Visual Studio and ActiveSync to install the MDA on the mobile device. The mobile application can use any of the available communication options for synchronization. For example Wireless Networks, Bluetooth, GPRS, and 3G. The only requirement is that the communication link must support the HTTPS protocol. We tested the application with a Wireless Network connection and a Bluetooth connection. All operations and security features of the mobile application worked well.

The application code can be found at the URL:



"http://utopia.duth.gr/~pefraimi/projects/SecMobDB”.

**Fig. 3.** The mobile application on a Pocket PC with Windows Mobile 5.0

**4 Resistance in Attacks**

The overall security of the MDA is achieved by ensuring:

— Security for the mobile device

— Security for the central computer

— Security for the communication link

— Security for application-specific issues

We examine the tolerance of the mobile database application against a comprehensive set of threats/attacks. We can distinguish the following types of threats:

— Threats from any user with access to the communication link

— Threats from a read-only user of the application

— Threats from an author user of the application

We consider a set of specific threats/attacks for the MDA and discuss how each threat is faced by our approach:

— Attack on the communication link: Eavesdropping of network traffic of the application or a fake client or server node. The security of the communication link is assured with the usage of the HTTPS protocol.

• Eavesdropping for example with a sniffer: In HTTPS all traffic is encrypted and hence, the confidentiality of the packet contents is protected.

• Fake client or server node: Using both the client and the server authentication features of HTTPS (features that are provided by the Secure Sockets Layer - SSL) assures the legitimacy of both the client and the server nodes. As already noted, in the current version of the mobile database software, the client authentication of HTTPS did not work properly within the synchronization process.

— Attack against the mobile device: The encryption of the mobile database ensures the confidentiality of the local data in case the mobile device is stolen or attacked.

• Stolen device: The local database that is installed on the mobile device is encrypted and hence, if the device is stolen, the application data is not readable. We note again, that the encryption is a feature of SQL Server Mobile and that we were not able to find any documentation about the encryption algorithm that is used.

• Network attack: The mobile database admits the client application to work while the mobile device is disconnected. The mobile device has to enable its network connection only during the synchronization operation of the mobile database. However, even during the short period that the portable device uses its network connection, it can become the target of malicious software. In Windows Mobile, there is currently no built-in firewall but there are third-party products that can cover this shortage. In any case, the data that is stored in the mobile database is encrypted and cannot be read.

—Attack against the server: The server computer where the server part of the application is executed must permit network access, in particular incoming connection requests, to its web server. Hence the server computer can become the target of attacks against the web server. We apply common security techniques to protect the server. Discussing these techniques is beyond the scope of this paper. There are numerous sources for the security of computing systems offering web services.

—Attack against the MDA: An important threat for any multi-user application comes from the registered users of the application.

• Attack from a read-only user: A read-only user can read all announcements. If a read-only user attempts to modify the contents of the database he will not succeed. First, the GUI of the client application does not provide this feature. This prevents unintentional attempts to modify data. Now, if a user intentionally uses some proprietary software or a low-level database utility to modify the application data, he will still fail because the publication at the database server is read-only. Finally, even if any data would be modified (in some way that we did not predict), the change would concern not the real

database table, but a replicated table, that is used for the read-only services.

• Attack from an author user: An authoring user has more permissions than a read-only user. We consider what will happen if an authored user attempts to perform operations for which he is not authorized. In this case, too, the GUI prevents unintentional user attempts to perform illegal operations. In the case that the user intentionally attempts to modify the data of other users by using some proprietary software or a low

level database utility, a set of triggers in the database server (see Section 3.7) prohibits the unauthorized operations.

**5 Discussion**

Developing a secure mobile database application is an important task. Our experience with developing and testing the application is satisfactory in several aspects. The efforts to implement the mobile database application were reasonable, it works reliably and it is efficient and user-friendly. For the security of a mobile database application, our case study showed that there are sufficient tools and techniques available to provide a security level comparable to the security level of conventional platforms. The few shortages that we faced are most likely technical issues that should be overcome in the forthcoming versions of the system software of the mobile platforms. Finally, an important issue is the lack of appropriate documentation for certain encryption algorithms that are used within the system software of mobile platforms.