Digital Rights Management in a 3G Mobile Phone and Beyond

Thomas S. Messerges
Motorola Labs
1301 E. Algonquin Road
Schaumburg, IL 60196
+1 (847) 576-5827
Tom.Messerges@motorola.com

Ezzat A. Dabbish
Motorola Labs
1301 E. Algonquin Road
Schaumburg, IL 60196
+1 (847) 576-5377
Ezzy.Dabbish@motorola.com

ABSTRACT
In this paper we examine how copyright protection of digital items can be securely managed in a 3G mobile phone and other devices. First, the basic concepts, strategies, and requirements for digital rights management are reviewed. Next, a framework for protecting digital content in the embedded environment of a mobile phone is proposed and the elements in this system are defined. The means to enforce security in this system are described and a novel "Family Domain" approach to content management is introduced. Our new approach uses key sharing to help alleviate bad user experiences that are associated with some rights management systems. Examples outlining the enrollment of devices and the acquisition, rendering, and superdistribution of content are shown. Our proposed system is not only applicable to a mobile phone system, but may also be extended to other embedded systems, such as personal digital assistants, set-top boxes, or personal computers.

Categories and Subject Descriptors
D.4.6 [**Operating Systems**]: Security and Protection – *Access controls*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection – *Unauthorized access.*

General Terms: Design, Security.

Keywords: Digital rights management, cryptography, security, embedded system, mobile phone, digital content, copyright protection, MPEG-21, key management, open mobile alliance.

1. INTRODUCTION
The mobile phone industry is on the verge of moving into its third generation of products. The first generation offered analog communication capabilities, the second generation featured digital radio technologies, and now the third generation is poised to embrace high-speed data and multimedia capabilities [27]. As these new, multimedia-capable phones emerge, business opportunities for the sale of valuable digital content, such as music, books, videos, ringtones, and games, are attracting much interest. These business opportunities are very lucrative to cellular operators, who are anxious to recoup their huge investments for radio spectrum rights and expensive new infrastructure equipment [12]. Although predictions for revenues from digital content vary widely, one recent study predicts that by 2006 the total revenues for digital music will be about US $5.6b [19]. The problem lurking behind these potential business opportunities, however, is that digital items can be perfectly copied and shared at virtually no cost. In order to make these business opportunities succeed, copyright protection using Digital Rights Management (DRM) technology will be an essential component in future mobile phones. However, people's lives are not centered on the use of their mobile phone. Consumers will more likely accept a DRM system that does not restrict digital content use to a single class of devices. Consumers wish to have access to their content on any of their electronic devices – home entertainment centers, car stereos, personal computers, etc. Therefore, a means to enable content to be seamlessly shared beyond 3G mobile phones and amongst multiple classes of devices is also needed.

In general, DRM technology encompasses a broad array of systems and processes. Content providers need to define and organize rights, content distributors need to package content and sell rights to consumers, payment brokers need to reconcile billing, and client devices (e.g., mobile phones) need to render content while enforcing the rights. In this paper, we concentrate on the portion of DRM that enforces the rights. We assume that content providers and distributors wish to couple usage rules to a digital item. They also want to ensure that mobile devices receiving valuable digital content follow the associated usage rules in a secure manner (e.g., see [8]). This means that usage rules and content will be protected with cryptographic techniques. These techniques will ensure the authenticity, integrity, and confidentiality of the content and the usage rules. We will examine these security concepts in the context of the embedded system of a mobile phone and then show how to extend content sharing to a family domain of devices.

1.1 Background
Digital communication data rates on current mobile phones are between 9.6 and 19.2 Kbps, depending on service [9]. This is far too slow for the convenient transfer of most digital items. However, the data rate for 3G mobile phones is expected to reach 144 Kbps, 384 Kbps, or 2 Mbps, depending on the mode of operation (vehicular, pedestrian, or fixed location, respectively) [27]. With these data rates, a complete MP3 song (about 4 MB of data) can be downloaded to a phone in 16 to 222 seconds, depending on the mode of operation. Digital items can also be transferred between peers using messaging services, such as Multimedia Messaging

DRM'03, October 27, 2003, Washington, DC, USA.

Service (MMS) [2], or streamed. Newer mobile phones will also be equipped with personal area networking capabilities, such as Bluetooth, which can manage data rates up to 723 Kbps [5]. Thus, peer-to-peer sharing of digital items over short-range networks will also be possible. At the same time, the number of mobile phones with Internet connectivity (e.g., i-mode [10] or WAP [44]) is growing so rapidly that soon there will be more mobile phones than desktop computers connected to the Internet.

Since the technology is now in place for convenient access to digital content in the wireless world, there is fear that losses from piracy will mount. Currently, more than one out of every three software applications is pirated. This translates into $12 billion lost due to software piracy in 1999 [7]. Also, Napster [43] showed the world how easy it is for people to share their MP3 music files. The small size of compressed music files and the availability of higher-bandwidth networks have made music particularly vulnerable to illicit copying [22]. Estimated losses due to piracy in the music industry vary widely. In 2001, predictions of losses ranged from US $2.3b [17] to US $4.5b [16]. Figures published by the Recording Industry Association of America (RIAA), show that shipments of CDs to retail outlets have dropped from US $1.08b in 2000 to US $968.58m in 2001 [29]. According to this same article, the RIAA blames online piracy for this decrease, stating, "23% of surveyed music consumers say they are not buying more music because they are downloading or copying their music for free." To top it off, there are also reports that the US $80b television industry is also beginning to be "Napsterized" [14]. Some TV shows are now available on the Internet.

Now that music, television shows, software, and even movies have become so vulnerable to piracy, what can be done? Lawsuits were successful at ending illicit music sharing on Napster [43]. However, other file swapping tools such as Morpheus [39] and Gnutella [37], which claim to be decentralized, may be harder to shut down. DRM offers a possible technical solution. However, diverging approaches and proprietary solutions are causing confusion in the marketplace and are slowing down widespread adoption. To help clear up confusion, some groups are beginning to look at standardizing DRM technology. The Open Mobile Alliance (OMA) is developing DRM standards for the mobile phone industry [40].

New mobile phones are capable of downloading Java games, they can play digital music, and they can show videos. If these new phones are also capable of DRM, then a much richer suite of digital items can be available and both consumers and content providers would benefit.

1.2 Paper Organization

This paper examines how copyright protection of digital items can be securely managed on a 3G mobile-phone platform and beyond. In section 2, the basic concepts, strategies, and requirements for DRM on a mobile phone are reviewed. Next, in section 3, an overall framework for protecting digital content in an embedded environment is proposed and the essential data, hardware, and software elements are defined. Security issues such as what makes up a license, where is integrity and authenticity important, how rights are enforced, and what protects the digital content are briefly examined in section 4.

Sometimes, rights management schemes lead to bad user experiences. So, in section 5, we introduce a novel "Family Domain" approach to content management. This new approach uses key sharing to help alleviate bad user experiences that have been associated with some rights management systems. Then, in section 6, we give examples that outline "Family Domain" enrollment, and acquisition, rendering, and superdistribution of content. In section 7, we conclude this paper with a vision of how our system might be useful for devices other than mobile phones and a look at future research opportunities.

2. DRM CONCEPTS AND STRATEGIES

In order to securely protect digital items, the content handling portion of mobile phone needs to possess the features of a *trusted system*, as defined by Stefik [42]. The phone will be relied upon to do certain things, such as not copy a digital item unless authorized to do so, not render content if it does not possess the rights, or pay a fee if that is what is required. In this paper, we start off by investigating how a mobile phone can be made into a trusted system and we outline the requirements for the various components that comprise this trusted system.

Figure 1 shows that, in the case of persistent content (i.e., content that is stored on the device and not streamed), the inputs to the trusted DRM system can be license and protected content files. The protected content file contains encrypted content and the license contains metadata and usage rules for the content. The content is encrypted to prevent it from being used by untrusted systems and the license is digitally signed to enable its integrity and authenticity to be verified. The license also contains a cryptographic hash of the content and a means to decrypt the content. For example, the key needed to decrypt the content would be encrypted with a particular device's public key and kept in the license. Only the device possessing the correct private key, which would not be known outside of the device, would be able to decrypt the content encryption key and gain access to the content. Finally, it is important that the signer of the license is an authority that is trusted or whose trust can be verified by the phone's DRM system.

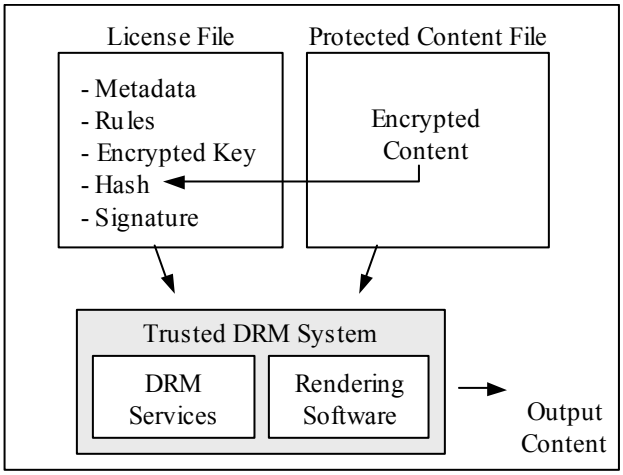

Figure 1. DRM is enforced using a protected content file and a license file. The license file will contain the usage rules, which are signed by a trusted authority and the protected content file will contain the encrypted content, which can be rendered only by devices possessing the corresponding license file.

When content is rendered, the trusted rendering software will present the protected content and corresponding license to the DRM services software. The DRM services will verify the signature of the

license, verify the hash of the content, decrypt the content, and then send the decrypted content back to the rendering software. The rendering software will then be able to play the music, show the video, or run the game, depending on the type of content. The presence of DRM will be transparent to users, except for cases when a user tries to render content without a proper license.

2.1 Existing DRM Systems

In 1999, the Secure Digital Music Initiative (SDMI), comprised of record companies, device manufacturers, and DRM technology vendors, completed a specification for portable devices [34]. This specification set forth the high-level requirements that a portable device should follow for handling digital content, but it was not detailed enough to provide a complete solution. The progress on SDMI was eventually stopped, but progress in other areas continues. For example, PressPlay, MusicNet, and Apple Computer's iTunes all offer on-line access to copy-protected digital music. Also, the US Congress has held hearings on digital content protection and legislation requiring copy-protection hardware has been proposed. However, it is still too early to see where this is heading.

Copyright protection technology, such as investigated by SDMI, does not provide a complete DRM solution so other issues also need to be considered. For example, rather than rendering the content, a device may also be used to stream, copy, or backup content to or from another device. Also, payment mechanisms for purchasing content and building a network of trust need to be considered. One system that works out some of these details is the *Keitaide-Music* system [38]. This system offers a secure system to deliver music to mobile phones and relies on a trusted memory card to protect the content.

The Motion Pictures Expert Group (MPEG) is considering the topic of DRM in the MPEG-21 standard. Contributors to the MPEG-21 standard are working to define system requirements and add DRM hooks. These hooks provide a standard framework for marking up protected content and describing how it should be handled. MPEG-21 will also include methods for expressing rights. Currently, MPEG-21 will use XrML [41] as its rights expression language, but further experiments are still in the works.

There are many companies that provide proprietary DRM technology (e.g., Microsoft, LockStream, InterTrust). Some of these companies also offer solutions for mobile phone systems. While these systems offer incompatible and different solutions, they must all rely on the foundation of a trusted system to ensure security. In this paper we try to pinpoint the essential architectural elements that are common amongst these solutions.

2.2 Open Mobile Alliance DRM

The Open Mobile Alliance, which works to define industry-wide specifications for applications that operate over wireless communication networks, has already released their version 1 DRM specification [40]. The scope of their specification is to "enable the controlled consumption of digital media objects by allowing content providers to express usage rights, e.g., the ability to preview DRM content, to prevent downloaded DRM content from being illegally forwarded (copied) to other users, and to enable superdistribution of DRM content". One goal is to devise a consumer-friendly DRM standard that maximizes interoperability, while minimizing complexity. The OMA specification gives requirements that affect the user experience, the user equipment (i.e., the phone), the usage rules, the content format, security, and privacy. In order to ensure a consumer-friendly solution, OMA advocates that content files can be distributed to other devices, but that licenses to use this content must be obtained from a server called the rights issuer. Later in this paper, we will show how our "Family Domain" approach can be used to enable content distribution to all devices owned by a consumer, without the need to acquire a new license for each transfer.

3. OUR DRM SYSTEM

Now that the basic requirements and background material have been reviewed, it is time to look at our proposed DRM system. The first step is to decide how to interface the DRM and security software with the phone's Operating System (OS) and applications. There are a number of possibilities. For example, in looking at a PC environment, Schneck [33] notes two approaches. His first suggestion is to replace the I/O elements of the OS with modules that contain access control mechanisms. These new modules would monitor all requests for I/O operations and would inform a user if a proper license for a digital item were not available. Schneck 's other approach, which does not require OS modifications, is to use a "hyperadvisor" that is situated between the OS and the hardware. When an application requests access to a protected file (protected files are identified with a special header), the hyperadvisor would invoke the DRM system and special software and hardware would complete the operation. Our view, shown in Figure 2, is slightly different than either of these approaches. Rather than replacing the I/O elements of the OS or adding a hyperadvisor, we propose that the OS be extended to support DRM functionality.

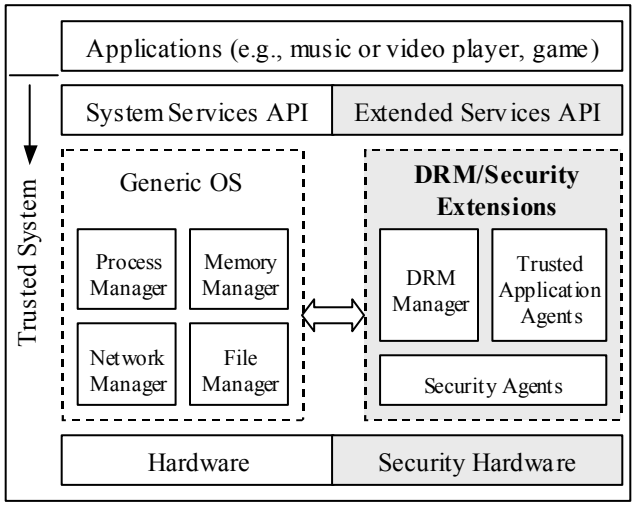

Figure 2. A generic operating system is extended with DRM and security capabilities by adding a DRM manager, trusted application agents, and security agents and hardware. The OS with extensions comprise a trusted system.

In our approach, only applications that access DRM-protected content need to be aware of the new DRM extensions. For example, when an application tries to access a particular file, a header may indicate that this file is protected. In which case, the application can use the DRM extensions to open the file and render the data. The application will access these extended system services through an Application Programming Interface (API) that is augmented to provide additional DRM-related services. Figure 2 shows that these DRM extensions include a DRM manager, security hardware, and a suite of trusted application and security agents. These extensions

will run in the same "privileged mode" as the OS and will have access to system data and resources. Applications, such as a music or video player's Graphical User Interfaces (GUIs), will run in "user mode" and have limited access to system data and resources. At the GUI level, these applications will not need to worry about DRM content other than to report on the status of the license. Lower-level components of the applications will invoke DRM extensions to process protected content, but as the content is decrypted and unprotected it will move under control of privileged-mode extensions. Some details of the blocks that implement these privileged-mode extensions are now examined.

3.1 DRM Manager

The DRM manager is responsible for the core DRM functions. It works with security agents to authenticate licenses and content, parse and enforce usage rules, access a secure DRM database, and provide decrypted content to a trusted application agent.

3.1.1 Authenticate Licenses and Content

Before an application can use protected digital content, it needs to use the DRM manager to verify the integrity and authenticity of the corresponding license file. This check will typically require that the cryptographic hash of the license file be computed and that a digital signature be verified. The DRM manager will need to parse the license and perform the appropriate checks.

The license may be cryptographically linked to the content via a hash value. In this case, a hash of the content will need to be computed and verified. The purpose of this hash is to bind the content to the license, thus it helps prevent a license for one digital item from being used for another digital item. Sometimes, in a low-end embedded system such as a mobile phone, the digital item may be too large to calculate its entire hash value at the time the license is checked. In this case, we propose that the hash value for the content could be computed in a piecemeal fashion. Small portions would be hashed and the hash results of each portion would collectively form a "hash table". Only the hash table (or a hash of the hash table) needs to be stored in the license file and initially verified (e.g., via the signature of the license file). The hash values in the hash table could be verified incrementally as each portion of the content is rendered. Thus, if licenses and content are packaged to include a hash table, hash verification can be distributed, making the processing task more suitable for an embedded processor (e.g., in a mobile phone).

In some situations, the signature of the license may need to be verified using a public key whose trustworthiness also needs to be checked. In this case, the DRM manager will need to verify an additional certificate or a chain of certificates that are contained in the license. The DRM manager will enlist the aid of security agents when performing this and all other cryptographic operations. The result of all these checks will determine whether the license and content files originate from valid sources and whether they have been modified. Upon completion of all these operations, the DRM manager will indicate to the application the results.

3.1.2 Enforce Rights

If the license and content have been successfully checked, an application can ask the DRM manager to perform an action on the content. For example, an application may ask the DRM manager to "play" the song, "display" the video, or "copy" the picture. Actions can be associated with three fundamental types of rights: render rights, transport rights, and derivative work rights [31]. The DRM manager should be capable of checking all of these rights and preparing for the appropriate action. Sometimes the license will stipulate an additional event for performing an action, such as a payment needs to be made or a play count needs to be decremented. The DRM manager will need to use a secure database to track these events.

Rights to an action are typically assigned to a specific entity (i.e., device). So, to enforce the usage rules, a DRM manager needs to have access to its device's credentials. A key/certificate manager, which is one of the security agents, is responsible for handling these credentials (e.g., keys, certificates, IDs). Applications that initiate requests for action will invoke the help of the key/certificate manager to obtain the appropriate credentials. These credentials, or a link to them, will be forwarded to the DRM manager.

Once the DRM manager has obtained the proper credentials, checked the rights, and approved a particular action, the digital item can now be decrypted and routed to the appropriate application agent.

3.1.3 Decrypt Content

Since a top-level application is not part of the trusted OS layer, it will not normally be allowed direct access to the decrypted content (one exception might be when the license indicates that it is fine to release the decrypted content). In our system, the DRM manager routes the decrypted content directly to a trusted application agent that is relied upon to perform the desired action (e.g., decode and play the song or video). The top-level application can control the decryption process and the operation of the action via system calls, but it will not have direct access to the decrypted content. Instead, the top-level application will act as the user interface and controller, while low-level trusted agents actually handle the data and are responsible for rendering the content.

3.1.4 DRM Manager Example

Figure 3 summarizes the three responsibilities of the DRM manager and shows the authentication, rules enforcement, and decryption steps that an application needs to perform to render content.

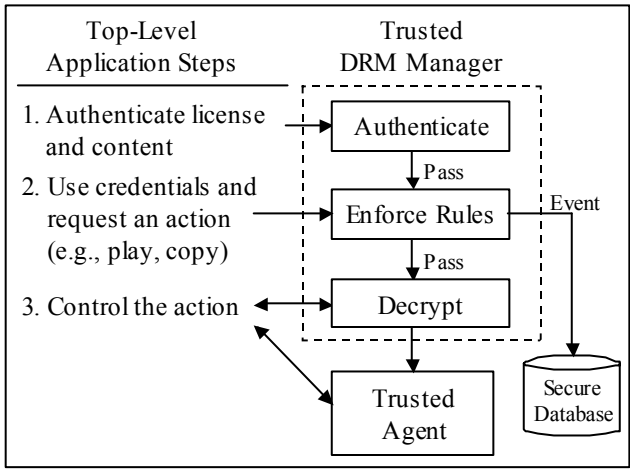

Figure 3. A top-level application uses the DRM manager to authenticate the license and content, and request and control an action (e.g., play, copy, display). The DRM manager ensures that the rules are enforced and possibly updates a secure database with an event, such as decrementing a count or logging a payment.

3.2 Trusted Application Agents

Like the DRM manager, the trusted application agents (as seen in Figure 2) are part of the extended OS. These agents support the ability of applications to access and manipulate decrypted content. They are referred to as "trusted" because they are part of the privileged OS layer. As with all the trusted software, we assume that some means is used to verify their integrity and authenticity and guard against hacker's attempts to make modifications.

The application agents can be organized according to the type of action they perform. Thus, there are rendering agents, transport agents, and derivative work agents.

3.2.1 Rendering Agents

Trusted rendering agents provide applications the ability to render DRM-protected content (e.g., a music player, a picture viewer, a video player, a book reader, a ringtone generator, an application loader). These agents provide the low-level drivers that convert the digital data into a format that can be consumed by a user. For example, a music player agent would take MP3 data and play it on a phone's headphones, a video player would take an MPEG4 stream and display it on the phone's screen, and an application loader would load and invoke a DRM-protected application, such as a Java game. The common feature amongst all of these agents is that they are trusted to properly handle decrypted digital content.

In order to allow for a richer user experience, the operation of some rendering agents, such as a music player, must be tightly coupled to the top-level application. The top-level application will provide the Graphical User Interface (GUI) to the rendering agents while the agents will merely provide low-level decoding operations and device-driver services. As such, the API between the agent and the top-level application needs to be cleverly designed to enable flexibility.

In our system, the execution of a DRM-protected software application is also categorized as a rendering operation. An agent, referred to as the application loader, is responsible for enforcing usage rules prior to executing a previously installed application. This loader agent makes sure that the rights and privileges assigned to an application are enforced while the application is running. In the case of Java application, the loader agent may need to configure the MIDP-NG [18] privileges, or set the MExE [1] domain to manufacturer, operator, third-party, or untrusted.

3.2.2 Transport Agents

Transport agents provide services that move content from one location to another (e.g., email attachments, messaging services, streaming, copying, loaning, device synchronization, or superdistribution). When transferring protected content, the DRM manager is first used to ensure that the usage rules are enforced. Next, a transport agent is invoked to start the transfer. The transfer might involve the establishment of a Secure Authenticated Channel (SAC) with the receiving device. In this case, the transport agent would also enlist a security agent to complete a protocol, such as Transport Layer Security (TLS) or Wireless TLS (WTLS).

Like the application agents, transport agents may need to handle decrypted content. Thus, transport agents also need to be trusted. As an example, consider that some newer mobile phones may have built-in digital music players where the audio signal is transmitted to headphones using a Bluetooth connection. In this situation, a trusted transport agent would be relied upon to establish a secure Bluetooth link between the headphones and the phone. The transport agent would receive the decoded and decrypted audio signal from a trusted MP3 decoder application agent. Then, the trusted transport agent would encrypt the audio signal and route it to the Bluetooth hardware, which would finally transmit the encrypted data to the headphones. The transport agents on the headphones and in the phone need to be trusted by content providers to properly handle decrypted audio data.

3.2.3 Derivative Work Agents

Derivative work agents are used to extract and transform protected content into a different form. For example, a copy of a digital item might have different rights than the original. When duplicating a digital item, a derivative work agent is invoked to make sure the copy's license is updated appropriately. Alternatively, this agent might contact a server to obtain a new license for the copy.

Another example of using a derivative work agent is in the installation of DRM-protected software or data. To ensure fast execution, installed software and data is decrypted; however, this makes it vulnerable to copying. In order to ensure continued enforcement of rights, a derivative work agent is used to place the decrypted data into an access-controlled file system. A security agent maintains the access-controlled files and allows only certain trusted agents access privileges.

Examples of data and applications that might be installed by a mobile phone include: software patches, games, or ringtones. Once an application or data is installed, other trusted agents can use it. A ringtone generator can play installed ringtones and a Java application loader can load Java games for playing. It may also be necessary to retain the original license for installed data or applications. An application or data may need to be uninstalled or a user may want to manipulate the data in some way.

3.2.4 Trusted Application Agent Example

Figure 4 shows a summary of how trusted application agents are used to install and play a new ringtone. The first step is to use a derivative work agent to decrypt and load the new ringtone. The user wants to hear the ringtone over his headphones. Thus, the next step is to use a secure link agent for establishing communications between the headphones and the phone. Lastly, a trusted ringtone player agent is used to access and play the ringtone.

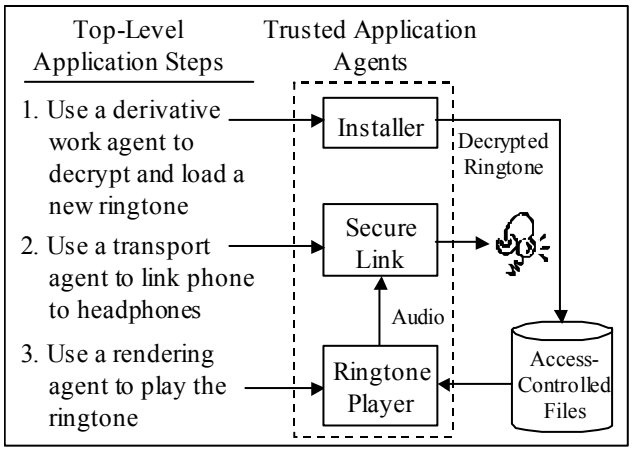

Figure 4. A top-level application uses trusted application agents to install a new ringtone. The ringtone is played over headphones by using other trusted agents to establish a secure link and play the ringtone.

3.3 Security Agents

The security agents handle the security-related functions that are commonly needed in all DRM systems. These functions include: secure memory and file management, cryptographic operations, and key management. The security agents may also work closely with available security hardware. Very often, embedded hardware can greatly enhance the security of a DRM system.

3.3.1 Memory and File Management

A DRM system needs to ensure that access to memory and files can be controlled. For example, we want to make sure that an installed ringtone cannot be accessed by just any application. It should only be available to the ringtone player, which is trusted not to make copies of the data.

There are at least three security functions related to memory and file management. These functions include the maintenance and operation of 1.) an access-controlled file system, 2.) a secure memory system, and 3.) a memory separation system.

The *access-controlled file system* is important for a number of reasons. One function it provides is the storage of digital content that is no longer encrypted. Our previous case of a Java game provides a good example of why such access control is needed. It would be very inefficient to always keep a DRM-protected application, such as the Java game, in an encrypted state. Therefore, there is a need to decrypt an application, but still store it securely. This is where the access-controlled file system can be used. A protected application can be decrypted and safely kept in files that only trusted agents can access.

Another use for the access-controlled file system is to store a secure database. This database can be used to track all events that the DRM system needs to log. For instance, usage rules may state that a particular song can only be played once. Thus, when a rendering agent plays this song, it will log a "play" event into the database. Similarly, encrypted private keys and data may be stored in this database. A security agent would be responsible for ensuring the secrecy and integrity of the secure database entries. Only trusted agents will be allowed to access this data.

If not already available, an existing file system could be augmented to include access-control functionality. Some requirements for the access-controlled file system include:

- Files are assigned ownership attributes that specify which trusted agents can access the files.
- Tampering of the ownership attributes can be detected.
- Files are optionally encrypted.
- Files that are not encrypted must be physically located within the phone.

One already implemented example of an access-controlled file system is Sony's memory stick system [4]. In this device, the responsibility for enforcing access control is placed on a special hardware module that resides on the memory stick. Smartcards, and, in the case of some mobile phones, Wireless Identity Modules (WIM), may also provide an access-controlled file system. In our system, we believe that a trusted security agent can work with the phone's onboard memory to maintain access control. However, memory separation between tasks needs to be maintained.

To guarantee task separation there is a need for a hardware-supported *memory separation system*. We want to ensure that when a trusted operation is running, untrusted operations cannot eavesdrop on the memory being used. In our system, a memory separation manager is responsible for maintaining the separation of tasks. When a task is run, this memory manager can configure a hardware monitor to define which memory is available to the task. In this way, we can ensure that tasks stay within their assigned memory areas and that they do not maliciously interfere with trusted operations.

Lastly, in any DRM system there is critical data that should never be allowed to leak out of the system. A *secure memory* system protects this data. For example, if a phone's private keys were to leak out, a hacker might be able to compromise the security and extract decrypted content. For high-security systems, physically probing the bus lines or pins of hardware components inside the phone is one avenue of attack that needs to be blocked. This can be accomplished using a secure memory that resides on the same IC as the processor. Unlike the secure database, the secure memory would only temporarily hold data, such as decrypted keys that are being used for a DRM operation. The volatile secure memory is linked to tamper detection circuitry. If suspicious events, such as attempts to enter debug mode, are detected, this memory is immediately cleared.

3.3.2 Cryptographic Operations

The security agents also provide access to symmetric and public-key cryptographic functions. Protected content is encrypted using a symmetric-key algorithm, such as AES [3], and the binding between content and licenses is done with a hash algorithm, such as SHA-1 [35]. Public-key operations, such as RSA [30] or ECC [20], are used for content key decryption, signature verification, signature generation, and for certain secure networking protocols (e.g., TLS or WTLS).

Recall that prior to rendering a digital item, the signature of a license needs to be checked and the content decryption key, which is in the license, needs to be decrypted. These operations, plus the hash of the content, may need to be completed in a short amount of time. Figure 5 gives some typical execution times for processing DRM-protected content using software implementations of RSA, ECC, SHA-1, and AES on a 16 MHz ARM7 microprocessor. A significant point of Figure 5 is that ECC is much better suited (20 time faster) than RSA for decrypting content keys. For software implementations, the optimal performance is achieved if ECC is used to decrypt the content key, but RSA is used to verify the signature of a license. When hardware accelerators perform these cryptographic operations, the difference between RSA and ECC may be less of an issue.

Operation	Time
Hash of a license (5KByte)	SHA1: 3 ms
Verify license signature	RSA(1): 100 ms ECC(2): 150 ms
Decrypt content key	RSA(1): 1,800 ms ECC(2): 90 ms
Decrypt content (2 Kbyte)	AES(3): 1.6 ms

(1) 1024-bit RSA with CRT (2) WTLS Curve 3 (3) 128-bit key

Figure 5. Typical execution times for processing DRM-protected content using software implementations of RSA, ECC, SHA-1, and AES on a 16 MHz ARM7 microprocessor. The above data shows that if hardware is not available, ECC is much better suited for wrapping content keys.

3.3.3 Key/certificate Manager

The Key/Certificate Manager (KCM) is a software module responsible for securely handling a database of the phone's credentials, which include private keys, public keys, certificates, and identification numbers. Private keys need to be kept secret, certificates need to be verified, and the links between public and private keys need to be maintained. This software provides these and other services to the rest of the DRM system.

As an example, the phone will contain a root certificate, upon which all trust can be verified. The verification of a signature may require the traversal of a certificate chain, which ends at this root certificate. It is expected that certificate chains be kept small (perhaps two or less) for mobile systems. The KCM is responsible for parsing and verifying the appropriate certificates.

As another example, the KCM needs to control the use of the phone's private keys. These keys should be usable only by OS components that are trusted. Also, ideally these keys will be decrypted only into the secure memory, thus they can be easily cleared if tampering is detected.

When the DRM system is configured, new keys or certificates may need to be installed. The KCM also handles this function. For example, when a device is configured to work with a particular DRM system, a new private key and public-key certificate may need to be installed. There are also cases when a key or certificate needs to be deleted. For example, when a subscription service expires, the corresponding private key and certificate need to be removed. Also, when a certificate is revoked or updated, an old certificate may need to be removed.

3.4 DRM Credentials

A DRM system needs to maintain keys and certificates that can be used to gain access to protected content and also establish trust with other entities. Unlike the PC environment, a mobile phone can provide more reliable information regarding the credentials of a device and its user. For example, in GSM phones, the International Mobile Equipment Identification (IMEI) number identifies the device and is now required to be unchangeable. Also, the International Mobile Subscriber Identity (IMSI) number provides information that is bound to a subscriber's service account [13]. Hartung and Ramme point out that if information, such as the IMEI or the IMSI (which are present in the network control layer of a mobile system), is made available to the application layer, then a more secure DRM system can result [15].

We now describe six elements, shown in Figure 6, that comprise the credentials of our DRM system: a Serial Number (SN), a Model Number (MN), a unit private key (*KuPri*), a unit certificate (*UnitCert*), a DRM private key (*KdPri*), and a DRM certificate (*DRMCert*). Trusted security agents (e.g., the KCM) are used to manage these elements and ensure that the private key remains secret. In addition, the DRM system will also need a root key (or keys) that is used to check the authenticity and integrity of the credentials of other devices, servers, or licenses.

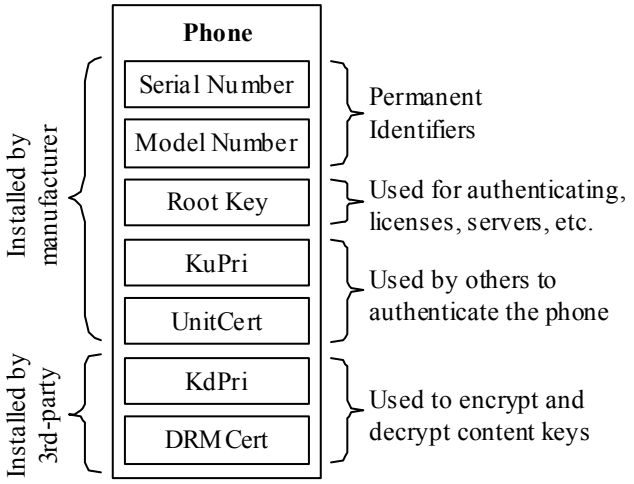

Figure 6. The phone's credentials consist of permanent identifiers, a root key, private/public unit keys, and private/public DRM keys. The unit keys are used to authenticate the phone and the DRM keys are used to assign content to a particular phone.

3.4.1 Serial and Model Numbers

The SN, which could be the same as the IMEI, is an unchangeable number that unambiguously identifies the phone. This number is useful for binding content to a phone. For example, a license might stipulate that only a device with a certain SN has rights to render a digital item. In order to enforce this license, the DRM manager would need to make sure its SN matches the SN in the license.

The MN is a number that unambiguously identifies the hardware and software version of a phone. It can be used by the DRM infrastructure, such as content providers, to indicate the phone's capabilities. As different phones with different DRM and content rendering capabilities are developed, it will be important for the content providers to know how to package the digital content for particular phones. The MN can also indicate security capabilities, such as whether the phone has hardware, rather than software, support for security. Some content providers may wish to sell valuable content to only phone models that offer hardware-backed security assurance.

3.4.2 Private Keys and Certificates

The phone's *KuPri* is the phone's unique private key and the *UnitCert* is a certificate that certifies the corresponding public

key (*KuPub*). We propose that *KuPri* and *UnitCert* should be used for establishing secure-authenticated channels to a phone, such as needed when a phone receives streamed content, is sent DRM-protected content, or is given new DRM keys. A phone's *KuPri* and *UnitCert* would form the foundation of trust for a DRM-enabled phone and would need to be installed in a secure manner, such as by the phone's manufacturer.

The phone's *KdPri* is also a unique private key and *DRMCert* is also a certificate that certifies the corresponding public key (*KdPub*). Unlike *KuPri* and *UnitCert*, the DRM key and certificate are meant to be used exclusively when assigning content to a device. For example, a digital item might be encrypted with a content key *CEK*. The *KdPub* key can be used to encrypt *CEK*. Thus, only the device possessing the corresponding *KdPri* would be able to decrypt *CEK* and, in turn, decrypt the digital item. In this manner, content can be assigned to a specific device.

One way to improve the secure usage of a device's private keys, *KdPri* and *KuPri*, is to bind the use of these keys to the unchangeable serial number of the phone. For example, this serial number can be included in the certificates, *UnitCert* and *DRMCert*. The phone would securely boot, validate its trusted software, and then use the trusted software to confirm that the serial number in the certificates matches its own serial number. Next, it would verify that the public key in the certificate is properly paired with the private key. This scheme helps prevent problems if an attacker is able to load a private key from one phone into another. The reason for the improved security is that private keys and certificates are loaded via secure software means, but that the serial number would be based on a hardware mechanism that is harder to undermine.

3.4.3 Short-Lived Certificates

In 1995, Macq and Quisquater described a system where a trusted authority grants "entitlements" to enforce access to a digital TV system [23]. They recommend that, for more robust security, an entitlement should be valid for only a limited time. In our system, the certificates act as the entitlement. A device's certificates can have expiration dates that need to be authenticated prior to allowing access to critical DRM operations. To revoke a device's access, no action is required, since without the active renewal of these certificates, the device would eventually stop working on its own. This situation is preferable to active revocation, because of the possibility that a revocation message to a device could potentially be blocked. Of course, a trusted source for time and date information is also needed if short-lived certificates are used.

4. SECURITY ISSUES

Now that our system has been fully described, we can start to examine how the components work together to provide a secure DRM solution. In this section we briefly examine some security issues such as what makes up a license, where is integrity and authenticity important, how rights are enforced, and what protects the digital content.

4.1 License

Rights are assigned to a digital item using a license, which is an unambiguous, machine-readable document that describes how a piece of content may be used. There are many possible license formats (e.g., XrML [41] or ODRL [36]), but there are only four essential items that really need to be in a license. These items include, a value that links the license to the digital item, the rights allowed for that digital item, a means to decrypt the digital item, and a signature of the license.

4.2 Integrity and Authenticity

There are many places where integrity and authenticity are important. The DRM manager needs to ensure the authenticity and integrity of the license; application agents need to ensure the authenticity and integrity of other trusted devices (e.g., the Bluetooth headphones); and a phone needs to prove its authenticity to other devices and the DRM infrastructure (e.g., content providers).

We assume that authenticity and integrity can be established either through a Public-Key Infrastructure (PKI) or a shared secret. For example, a phone verifies the signature of a license using a root key that is securely embedded into its hardware. However, Bluetooth headphones and a phone might share a secret key that is preprogrammed into each device or securely established. Strong and publicly scrutinized cryptographic algorithms, such as RSA [30] or ECC [20] for signatures, AES [3] for symmetric encryption, and SHA-1 [35] for hashing, will also help to ensure that the DRM system remain secure. The phone will need to verify signatures (public-key encryption) and decrypt content keys (public-key decryption).

Content providers need to trust that the DRM system in a phone will keep all private or shared symmetric keys secret – even from the phone's owner. If a private key is not secret, then there is a potential rogue phone problem. That is, someone could place an authentic private key into a rogue phone. Content providers would not be able to distinguish this rogue phone from a real phone, and might inadvertently sell it content. This is a problem because a rogue phone does not necessarily enforce any usage rules for content.

4.3 Rights Enforcement

It is the responsibility of the DRM manager to enforce the usage rules. The DRM manager will need to parse the license file and recognize and process the different rights expressions. If the DRM manager finds a conflicting expression or one that it cannot understand, then it must fail in a safe manner. One way to try and fool the DRM manager might be to supply older or newer versions of licenses. Thus, the DRM manager needs to be able to recognize the version of the license file. It should be designed to be backwards compatible, so that old licenses can be properly interpreted.

4.4 Content Protection

Content is protected with encryption up until the time it is rendered or installed into the access-controlled file system. Encrypted content can be streamed to a phone from a remote server or it can be stored locally in a memory device. In either case, the content will be decrypted and routed to the appropriate rendering hardware by trusted agents. These agents are trusted not to leak or copy the decrypted content.

Before a trusted agent can start decrypting content, it needs to obtain the decryption key *CEK*. If the content is stored locally, a version of *CEK* encrypted with *KdPub* will be in the license. The trusted agent will need to use its *KdPri* to decrypt *CEK*. If the content is streamed, *CEK* will be the session key that is negotiated with the server during the establishment of a SAC. This SAC is established by using the phone's *UnitCert* and *KuPri*.

4.5 Privacy Issues

There are two privacy issues that need to be considered in a DRM system:

- User information used to create a content license must not be disclosed without the explicit consent of the end user.
- The user's identity must not be disclosed to a content provider and/or to other parties without the explicit consent of the end user.

Ideally, a DRM system should not put at risk a user's private information. For example, in designing a DRM system, a user's privacy might be enhanced if transaction is tied to a person's device rather than their identity. Feigenbaum et al. [11] and Kravitz et al. [21] take in-depth looks at DRM privacy issues.

5. FAMILY DOMAIN

A number of proposed DRM schemes, such SDMI, PressPlay and MusicNet, have received some poor reviews from the public [24],[26]. In general, consumers are resistant to DRM systems and need to be assured that their rights will also be protected. One privilege that consumers wish to protect is the right to use their content on any of their devices – not locked to one particular device. Some proposed DRM systems require a public-key infrastructure and a centralized locker approach to give users access to content anytime, anywhere (e.g., [32]). However, approaches like these may not be suitable for devices such as mobile phones or other multimedia equipment that, unlike personal computers, may not have permanent networking capabilities.

In order to preserve a consumer's right to move content to all of his devices, we propose a new concept of "Family Domain" content management. In our scheme, the consumer decides which devices belong to his domain (e.g., all devices he and his family own) and a trusted server, which we refer to as a Domain Authority (DA), installs a common DRM private key in each of these devices. In effect, the DRM private key becomes a domain private key that enables access to all the content in a domain. A secure perimeter is established. Devices inside the domain have full access to the content and devices outside the domain do not. Our scheme is also suitable for devices that have limited or no networking capabilities. A device needs to only register with a DA once and this be registration can be performed via a proxy device that acts as a temporary gateway to the network.

Some DRM schemes, such as SDMI, restrict access to content based upon a check-in/check-out policy in which security restrictions are encountered every time content is loaded into or out of a device. In our domain-based system, users contend with security only when a new device is added to or removed from a domain. Thus, we believe that a domain-based DRM system will gain wider consumer acceptance.

5.1 Device Configuration

In our "Family Domain" system, portable devices are assigned to a particular domain by registering with the DA. When a device registers into a domain, we say the device has "joined" the domain. When a device no longer wants to be part of a domain, it can "leave" the domain by canceling its registration. The DA enforces registration policies, such as limiting the number of devices in a domain and limiting the number of times a device can join and leave a domain. The DA can also detect potential fraud by tracking which devices are joining and leaving the domains. Excessive activity may indicate that a device is trying to abuse the system. Such devices can then be prohibited from further registration activities.

The ability to add or removed devices from one's domain can be controlled using passwords. Users could potentially share family domain passwords and add non-family members to a domain. However, if this concerns content providers, it might be possible to tie domains to service accounts or to tie entry into a domain to a user's private information or an ability to spend money. As an example, when we investigated the procedure for registering a Liquid Audio player, a user was offered two options: a full or a fast-track passport. A fast-track passport enables DRM-protected music to play on a specific PC while a full passport allows music to be shared on any PC that has a copy of the full passport. To limit the sharing of full passports, which could lead to abuse, Liquid Audio requires a credit card number be used when obtaining the full passport. The passport is associated with the credit card number, so users will be unlikely to share their passports. To make this system more acceptable to the users, this linking of access control to a payment mechanism might be offered as a convenience, such as a way to enable one-click shopping.

Previous studies have considered issues such as key management for multicast, where a group key is shared amongst a group of users who can leave or join the group. Efficient methods to manage such a scheme are available [28]. The intent of these schemes is to prevent leaving members from decrypting future content and prevent joining members from decrypting previous content. Some of these methods might also be applicable to managing family domain keys.

5.2 Family Domain Example

In the case of "Family Domain" content management, a consumer will contact a content provider and purchase a song for his domain. The purchase transaction protocol will be the same as for buying content locked to a single device. However, the DRM certificate would actually be a domain certificate, which contains the domain public key. Now, when the content provider encrypts the content key *CEK*, any device in the domain (i.e., all devices with the same *KdPri*) will have the same rights to the content. Any device in the domain that receives this content will be able to render it. The main difference is that the license will lock the content to a domain rather than a device.

6. EXAMPLE USE CASES

In this section we examine four important scenarios that show how the components in our DRM system might interact with each other. We look at examples of enrolling a device into a domain, buying new content, rendering content, and sending a friend some content.

6.1 Enrollment of Device into a Family Domain

When a consumer wants to create a domain of devices, the procedure can be very simple. For example, a device can be added to a domain by registering it with the DA using the following steps:

1. The consumer activates the domain enrollment application, which initiates contact with the DA.
2. The phone and DA establish a SAC and the device identifies itself to the DA.
3. The consumer indicates whether he wants to form a new domain or add the device to an existing domain.

4. The DA sends a new or an existing (in the case of joining an existing domain) *KdPri* and *DRMCert* to the phone.
5. The phone securely installs the *KdPri* into its access-controlled database.

To ensure that the domain concept is not abused, the DA will enforce the policy that only a limited number of devices are allowed in each domain. Also, the DA must authenticate the phone's *UnitCert* to guard against unauthorized phones from joining a domain.

6.2 Over-the-Air Content Acquisition

A user can shop for content using his phone's web browser. The phone will connect to a Content Provider (CP) and display the available items. The user will select items to purchase and the CP and phone will enter a purchase transaction protocol, whereby the purchased content is packaged and sent to the phone. The purchase transaction steps are as follows:

1. The web browser makes a system call to the trusted acquisition protocol.
2. To protect possible payment information and also bind payment information to the phone's identity, the *UnitCert* and *KuPri* are used to establish a SAC.
3. The phone uses the SAC to send the phone's *DRMCert* to the CP.
4. The CP uses *KdPub* in the *DRMCert* to create a license file that cryptographically binds the content and rights to the phone's *KdPri*.
5. The CP sends the license file and protected content file to the phone.

Once the content and license arrive at the phone, they can be stored in any sort of memory. There is no need to protect the content at this time, since it is encrypted with *CEK* and *CEK* is encrypted with the phone's *KdPub*. Only trusted software can access *KdPri* and decrypt *CEK*.

6.3 Content Rendering

A user can render DRM-protected content with an application that interacts with the trusted DRM extensions. For example, to play a song, the user will press the "play" button and the following background steps will occur:

1. The top-level application gives the name of the protected content and license files to the DRM manager, which uses the phone's root key to authenticate the license.
2. The top-level application requests a "PLAY" action from the DRM manager, which enforces the usage rules and, if necessary, logs an event in the secure database.
3. The DRM manager invokes a trusted security agent to decrypt the content key CEK using the KdPri.
4. The top-level application controls the rendering agents, which decrypt, process, and send the content to an output device.
5. If any of the above steps fail or the requested right is not allowed, the content is not rendered and the top-level application is notified.

6.4 Peer-to-Peer Superdistribution

The basic model for superdistribution, where content is passed along more than once, was invented in 1983 by Ryoichi Mori and is described in [25]. As was seen with Napster, peer-to-peer sharing has the potential to propagate content extremely rapidly. Thus, content providers, who are anxious to increase revenues, are very interested in allowing secure peer-to-peer superdistribution in a mobile phone environment.

In order to share a digital item, a user will use a top-level application to select which item to share and where to send it. The user will then press the "send" button. The two devices will connect and the content will be delivered. We assume that the content is protected (i.e., encrypted); therefore, the recipient device does not need to be authenticated.

After a recipient device obtains the content and license, its user can try to render the content by pressing the "play" button. Here are the background steps that occur when the recipient of the new content tries to play the content:

1. The top-level application gives the name of the protected content and license files to the DRM manager, which uses the phone's root key to authenticate the license.
2. The top-level application requests a "PLAY" action from the DRM manager. The manager determines that the recipient's credentials do not allow rendering. However, a short, decrypted sample may be included with the content.
3. The user is asked whether she wants to hear the short sample (if available) or purchase full rights to the content.
4. If the user wants to hear the sample, a trusted rendering agent is used to render the sample.
5. If the user wants to buy the full rights, a purchase transaction protocol is invoked and a new license is delivered over the air.

One of the nice features of superdistribution in a mobile phone system is that a free Bluetooth connection can be used for distributing the large content files, while a more costly network connection is used only for the smaller license files.

7. CLOSING REMARKS

Our DRM framework has been proposed for a mobile phone environment, but it is also applicable to other devices, such as a PDA, set-top box, automobile, or a PC. It would be particularly advantageous to extend our family domain concepts to these other devices because content could be more seamlessly shared amongst all devices owned by a consumer.

In one future scenario, a consumer, say Alice, will be listening to her car radio and hear a song she likes. She can just press a button on the radio to purchase this song. Behind the scenes, Alice's car radio and the rest of the devices in her domain will be able to connect to a service provider. This service provider will maintain Alice's content list and make it available to all of her family domain products. Later in the day, when Alice is jogging in the park, she may want to listen to her new song on a portable music player. She will simply scan her content list for this new song and add it to her playlist. At the same time Alice's husband, say Bob, might be at home. He can listen to DRM-protected songs on his home audio system. These songs could be stored locally or streamed from his cable company to his set-top box. If our family domain concept, plus our DRM framework, is

adopted into all of these devices, then scenarios like this will be possible and more potential business opportunities will result.

7.1 Future Research

There are still many areas of research and development that need to be completed before the DRM system described in this paper becomes a reality. For example, there are many use cases that need to be explored, the software blocks need to be more thoroughly described, secure mechanisms to extend the OS need to be developed, and hardware support to enable a trusted computing platform needs to be deployed.

The subject of DRM is still divisive and confused. No one is certain which standards will emerge or which methods will be accepted. In a recent report on potential business models for digital music distribution, Buhse [6] reports that it is still too early to determine which business models will succeed, so he recommends that companies prepare themselves for various scenarios. A DRM system that is flexible enough for different business models, yet still efficient for an embedded mobile phone system, is needed.

Our proposed approach offers a good path towards a secure and consumer-friendly DRM system. Our security framework and "Family Domain" DRM approach can benefit both content owners and consumers.

8. ACKNOWLEDGMENTS

The authors would like to thank Ron Buskey and members of Motorola Labs' Security and Privacy Technology Lab for their helpful comments and advice, including Larry Puhl for his work on Family Domain, and Dean Vogler and Yi Li for helping to develop prototype software.

9. REFERENCES

[1] 3GPP TS 23.057, "3rd Generation Partnership Project; Technical Specification Group Terminals; Mobile Station Application Execution Environment (MExE); Functional description; Stage 2; (Release 4)".

[2] 3GPP TS 23.140, "Multimedia Messaging Service (MMS); Functional description; Stage 2".

[3] "Advanced Encryption Standard (AES)," FIPS PUB 197, Nov. 2001, Available at: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[4] S. Araki, "The Memory Stick," IEEE Micro, vol. 20, issue 4, July-Aug. 2000, pp. 40-46.

[5] P. Bhagwat, "Bluetooth: Technology for Short-Range Wireless Apps," *IEEE Internet Computing,* vol. 5, issue 3, May-June 2001, pp. 96-103.

[6] Willms Buhse, "Implications of Digital Rights Management for Online Music – A Business Perspective," *Proceedings of the ACM Workshop in Security and Privacy in Digital Rights Management*, associated with *ACM CCS '01*, Philadelphia, PA, November 2001.

[7] Business Software Alliance Report, Available: http://www.bsa.org/.

[8] S. M. Cherry, "Making Music Pay," *IEEE Spectrum*, vol. 38, issue 10, Oct. 2001, pp. 41-6.

[9] F. Dahlgren, "Future Mobile Phones – Complex Design Challenges from an Embedded Systems Perspective," Proceedings *of the Seventh IEEE International Conference on Engineering of Complex Computer Systems*, 2001, pp. 92-4.

[10] K. Enoki, "i-mode: The Mobile Internet Service of the 21st Century," *IEEE International Solid-State Circuits Conference (ISSCC)*, 2001, pp. 12-5.

[11] Joan Feigenbaum, Michael J. Freedman, Tomas Sander, and Adam Shostack, "Privacy Engineering for DRM Systems," *Proceedings of the ACM Workshop in Security and Privacy in Digital Rights Management*, associated with *ACM CCS '01*, Philadelphia, PA, November 2001.

[12] Xianjun Geng and A.B. Whinston, "Profiting from Value-Added Wireless Services," *Computer*, vol. 34, issue 8, Aug. 2001, pp. 87-9.

[13] GSM 02.09 (ETS 300 506), "Digital Cellular Telecommunications System (Phase 2); Security Aspects," Aug. 2000.

[14] Anita Hamilton, "The Pirates of Prime Time," *Time.com*, Feb. 16, 2002, Available: http://www.time.com/time/business/article/0,8599,203498,00.html.

[15] F. Hartung and F. Ramme, "Digital Rights Management and Watermarking of Multimedia Content for M-Commerce Applications," *IEEE Communications Magazine*, vol. 38, issue 11, Nov. 2000, pp. 78-84.

[16] "International Federation of the Phonographic Industry (IFPI) Music Piracy Report," June 2002, Available: http://www.ifpi.org/site-content/library/piracy2002.pdf.

[17] International Intellectual Property Alliance, "USTR 2002 'Special 301' Decisions and Estimated Trade Losses Due to Copyright Piracy," April 30, 2002, Available: http://www.iipa.com/pdf/2002_Apr30_USTRLOSSES.pdf.

[18] "JSR-000118 Mobile Information Device Profile Public Review Draft Specification 2.0," Available at: http://java.sun.com.

[19] Jupiter Media Metrix – Press Release, "Subscriptions Will Account For Almost Two-Thirds Of US Digital Music Sales In 2006," Jan. 15, 2002, Available: http://www.jmm.com/xp/jmm/press/2002/pr_011502.xml.

[20] N. Koblitz, "Elliptic Curve Cryptosystems," *Mathematics of Computation*, vol. 48, 1987, pp. 203-9.

[21] David W. Kravitz, Kim-Ee Yeoh, and Nicol So, "Secure Open Systems Protecting Privacy and Digital Services," *Proceedings of the ACM Workshop in Security and Privacy in Digital Rights Management*, associated with *ACM CCS '01*, Philadelphia, PA, November 2001.

[22] Calvin K. M. Lam and Bernard C. Y. Tan, "The Internet is Changing the Music Industry," *Communications of the ACM*, vol. 44, issue 8, August 2001, pp. 62-8.

[23] B.M. Macq and J.-J. Quisquater, "Cryptology for Digital TV Broadcasting," *Proceedings of the IEEE* , vol. 83, issue 6 , June 1995, pp. 944-57.

[24] Anna Wilde Mathews, Martin Peers and Nick Wingfield, "Music Industry Finally Online – But Listeners Stay Away in Droves," Wall Street Journal, May 7, 2002.

[25] Ryoichi Mori and Masaji Kawahara, "Superdistribution: The Concept and the Architecture," *The Transactions of the IEICE*, vol. E 73, no. 7, July 1990.

[26] Walter S Mossberg, "Sony's Digital Music Clip is Cool, but Treats Users Like Criminals," Wall Street Journal, March 2nd, 2000.

[27] M.W. Oliphant, "The Mobile Phone Meets the Internet," *IEEE Spectrum*, vol. 36, issue 8, Aug. 1999, pp. 20-8.

[28] Benny Pinkas, "Efficient State Updates for Key Management," *Proceedings of the ACM Workshop in Security and Privacy in Digital Rights Management*, associated with *ACM CCS '01*, Philadelphia, PA, November 2001.

[29] "Piracy Blamed for CD Sales Slump," *BBC News*, Feb. 26, 2002, Available: http://news.bbc.co.uk/hi/english/entertainment/new_media/newsid_1841000/1841768.stm

[30] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Crypto-systems," *Comm. ACM*, vol. 21, 1978, pp. 120-126.

[31] Bill Rosenblatt, Bill Trippe, and Stephen Mooney, "Digital Rights Management: Technology and Business," *M&T Books*, New York, New York, 2002, pp. 62.

[32] Thomas Sander, "Golden Times for Digital Rights Management?," Financial Cryptography : 5th International Conference, FC 2001, Grand Cayman, British West Indies, February 2001, pp. 64-74.

[33] P.B. Schneck, "Persistent Access Control to Prevent Piracy of Digital Information," *Proceedings of the IEEE*, vol. 87 issue 7, July 1999, pp. 1239-50.

[34] Secure Digital Music Initiative (SDMI), "SDMI Portable Device Specification," Part 1, ver. 1.0, 1999.

[35] "Secure Hash Standard (SHS)," FIPS PUB 180-1, April 1995, Available at: http://www.itl.nist.gov/fipspubs/fip180-1.htm.

[36] See http://odrl.net.

[37] See: http://www.gnutella.com/.

[38] See http://www.keitaide-music.org.

[39] See: http://www.musiccity.com/.

[40] See: http://www.openmobilealliance.org/.

[41] See http://www.xrml.org.

[42] Mark Stefik, "Letting Loose the Light: Igniting Commerce in Electronic Publication," in *Internet Dreams*, Mark Stefik ed., MIT Press, 1997, pp. 219-254.

[43] R Stern, "Napster: A Walking Copyright Infringement?" *IEEE Micro*, vol. 20 issue 6, Nov.-Dec. 2000, pp. 4-5, 95.

[44] Wireless Application Protocol, Available: http://www.wapforum.org/.