

Downloadable Content – Security and Robustness

INTRODUCTION

The ability to download applications is accompanied by significant security risks. Among the most prominent risks introduced when downloading applications are those resulting from the user's inability to assess the integrity and authenticity of the code being downloaded. Put more simply, the user cannot be sure who originally created the code or whether the code being downloaded is exactly the same as that created by the originator. Additionally, in the world of mobile devices, robustness is also a key requirement when it comes to downloadable content. Once an application has been validated as secure, the downloading itself must be robust in the sense that new applications should not be allowed to negatively affect system software. Despite the best intentions of the application programmer, software may contain bugs that could potentially alter the proper functioning of a mobile device. Therefore, to ensure device stability and reliability, protection mechanisms should be implemented.

INFORMATION SECURITY

When software applications are provided on a CD or some other physical media, security problems are much less notable than when dealing with downloadable code. The CD packaging (logos, captions, etc.) to all intents and purposes certifies the originality of the product. The efforts that would be required to fake a commercial product and distribute an alternative version are very expensive, far beyond the means of hackers trying to force innocent occasional users to install backdoor¹ programs. Similarly, the read-only nature of a CD and the sealing of software provided on other physical media prevent casual adversaries from making changes in distributed code, for example by infecting the code with a virus or a Trojan-horse².

11 "Backdoor" is a term for programs that, once installed, allow an adversary to connect to the device on which they are installed, usually over a network, for the purpose of controlling the device, downloading content from it, or using its resources for malicious activities.

21 "Trojan-Horse" is a term describing a modified program that to the unsuspecting user looks and feels as if it is performing one operation while in fact it is performing another, usually malicious, operation on the user's device. An example is a program pretending to be a user's e-mail client, but which asks the user for his/her e-mail password and sends this password to an adversary.

In contrast, when applications are downloaded, they must travel over public network nodes, where they can be easily modified in a way that cannot be detected by the recipient. Though non-cryptographic mechanisms, such as CRC (Cyclic Redundancy Check), do exist and can detect modifications to content caused by communication errors, such mechanisms cannot prevent attacks by active adversaries who modify the transferred CRC values to match the modified content, thus avoiding detection.

The risks involved in downloadable applications (often called "mobile code") are not new. Long before the real-time operating system OSE offered the ability to download applications into resource-constrained environments, mobile code was already available in the PC environment in the form of Java applets and ActiveX controls as well as ordinary executable files of applications that users could download over a network and install on their machines. The risks were recognized and a variety of solutions were devised, such as anti-virus programs, anti-Trojan programs, Java sandboxing and code signing.

Enea Embedded Technology – the OSE provider – and Discretix are together offering a robust code-signing architecture designed to ensure the originality and legality of downloaded applications. With this technology, users downloading an application can be sure that the content they are downloading originated with the legal entity as claimed and that the code was not modified in transit by any other entity. This system also ensures that the authentic downloaded content will be safely installed in the device with proper protection mechanisms.

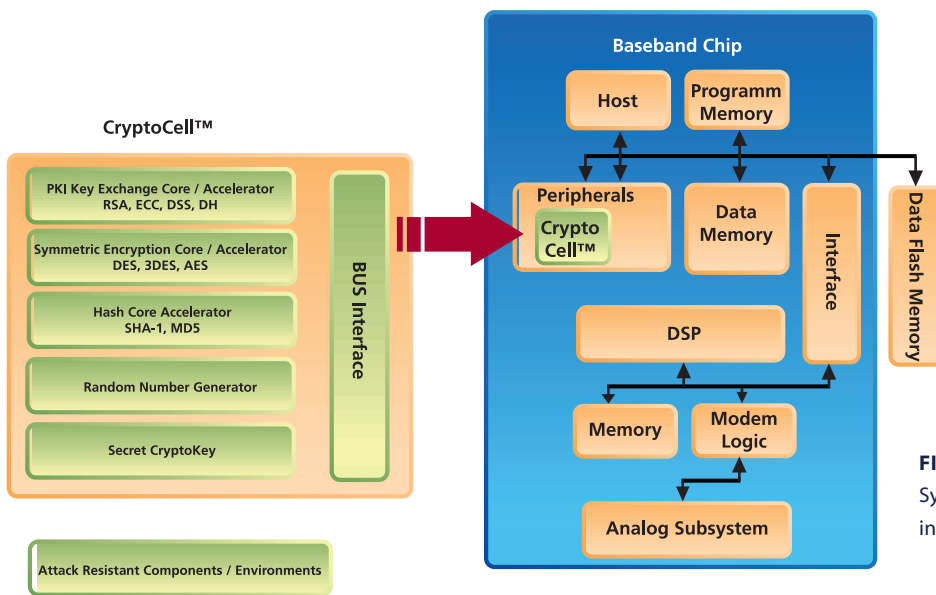


FIGURE 1:
System Diagram - CryptoCell integrated into the digital baseband

The system is based upon digital signature technology and makes use of perceivably robust signature schemes and secure "time-proven" algorithms, similar to those used by commercial and legal digital signature applications. The robust attack-resistant implementation of the digital-signature verification module ensures that key components of the module will be extremely difficult to circumvent, even by someone with complete access to the operating system running on the device. Each piece of code is digitally signed using its author's private key. This signature is verified by the device upon arrival of the code and before installation. In order for a variety of authors to publish content, certificates containing their respective public keys may be attached to downloaded applications.

The root public key used to validate those certificates is stored in an attack-resistant storage space on the device, where its integrity is assured. This digital signature verification mechanism is implemented through a combination of software (to handle the application download) and hardware (to allow for secure storage on the device and for provision of tamper-proof cryptographic primitives).

ROBUST DOWNLOADING

Once a downloaded application has been acknowledged as authentic and its origin validated as approved, the application must still be handled with a certain degree of precaution. A dynamic system is far more vulnerable than a static system, where software is never changed. Even if a particular downloaded application is "bug free", testing or predicting how the complete system will behave once the new application is downloaded and starts to execute presents a significant challenge. This is especially true for realtime systems, including mobile devices. Race conditions, deadlock situations, starvation and other familiar programming challenges are well known to the experienced programmer of real-time embedded systems. So regardless of the high quality of a downloaded application and regardless of its minimal number of bugs, the way in which it is stored and executed in the target device should still be examined.

In effect, two basic categories of applications are usually downloaded: system software and user applications. System software refers to applications that the end-user does not see or use directly, for instance a networking protocol or a graphics routine. User applications include software that the end-user interfaces directly, for example a game or a calendar. System software is usually native software, implemented in C or assembly language, while user applications can be implemented in a variety of languages, including C/C++, Java, Visual Basic and others. These applications may also be executed in a sandbox environment like a virtual machine, where they are completely protected from the rest of the system. (A virtual machine is itself an example of system software.)

The first category, system software, is always downloaded directly by the operating system, while the second category, user applications, may be downloaded by the operating system or by the executing environment, i.e. the virtual machine. Whenever the operating system directly downloads system software or user applications, it needs to provide mechanisms for partitioning, isolation, supervision and protection in order to safeguard already installed programs from bugs that may be found in the newly downloaded software. A downloaded application that behaves in a faulty or defective manner should never be allowed to affect the rest of the system.

The OSE realtime operating system features built-in capabilities that are particularly suited for dynamically downloaded content. Flexible design and communication models ensure a software architecture that is robust even when programs are installed or removed during runtime. Fault isolation is improved by the OSE's built-in error detection. A robust and secure memory model provides all applications with their own memory pool that is separated from other memory pools. OSE has advanced memory management unit (MMU) support, and if an MMU is present in the device, the memory pools should also be protected from each other. If a downloaded application is removed from the device, OSE automatically reclaims all the system resources it has allocated. All the memory used by the application is also cleaned up and returned to the system without any memory fragmentation, ready to be reused completely.

SUMMARY

The digital signature model developed by Discretix is seamlessly integrated with the robust OSE realtime operating system. The joint solution provided by Enea Embedded Technology and Discretix assures that the user can enjoy the versatility and endless possibilities presented by downloadable applications without ever having to worry about the genuine security risks normally involved when mobile devices are dynamically updated.

CONTACT INFORMATION

EUROPEAN HEADQUARTERS

ENEА EMBEDDED TECHNOLOGY
 NYTORPSVÄGEN 5B
 SE-183 23 TÄBY, SWEDEN
 PHONE: +46 (0)8 507 140 00
 FAX: +46 (0)8 507 140 40
 EMAIL: INFO@ENEА.SE
 WWW.ENEА.COM

USA HEADQUARTERS

ENEА EMBEDDED TECHNOLOGY
 12760 HIGH BLUFF DRIVE
 SAN DIEGO, CA 92130
 PHONE: +1 (858) 720-9958
 FAX: +1 (858) 720-0150
 EMAIL: INFO@ENEА.COM
 WWW.ENEА.COM

Some of the product names used herein have been included for identification purposes only and may be trademarks of their respective companies. OSE is a registered trademark of Enea Embedded Technology . ©2003 Enea.

WHITE PAPER WP-23 032004