1

Why a Secure Platform?

by Craig Heath

1.1 User Expectations of Mobile Phone Security

Mobile phones are perceived somewhat differently from desktop PCs and laptops, although, from a technical point of view, a smartphone is actually a general-purpose computing device. Users of mobile phones have rather different expectations than PC users regarding their device’s security and reliability.

A mobile phone is essentially a personal item. It is not typically shared with other family members, and the fact that it is carried around in your pocket leads to a higher degree of trust in it as a reliable and secure repository for your personal data. It feels secure, just knowing it’s right there with you. However, contrary to this natural feeling of security, it may, in fact, be exposed to attacks via Bluetooth, Wi-Fi, GPRS or other network connections, even while it is sitting in your pocket.

To address this disparity of perception (without increasing the paranoia of mobile phone users to match their feelings regarding PCs, which would probably be damaging for the whole mobile phone industry), Symbian believes it should provide higher levels of mobile phone security to match customers’ existing expectations. Mobile phone users don’t expect, for example, to have to install anti-virus software on their phones after they buy them and, if the industry does its job properly, they won’t need to.

Mobile phone users also have clear expectations regarding the reliability of their phones which differ substantially from PC standards. They expect to be able to place and receive voice calls at any time; they don’t expect ‘blue screens’ (unrecoverable errors) and they don’t expect to have to reboot their phone at all, let alone daily.

The purpose of a mobile phone is, primarily, to be a delivery point for network services, so users really don’t care whether their calendar is stored on the phone or on a remote server, or indeed is synchronized between the two. One consequence of this ‘blurring’ between the device and the
network is that people are generally comfortable with device settings being managed remotely (typically by the mobile network operator). We expect that the controversy regarding the control of the device that has surrounded the work of the Trusted Computing Group (or TCPOA as it was formerly known), dubbed ‘treacherous computing’ in [Stallman 2002], is unlikely to be as significant an issue, if and when similar technology is applied to mobile phones.

Lastly, and certainly not least, users expect the behavior of their mobile phones to be predictable – they don’t want nasty surprises on their bill at the end of the month. This is demonstrated by the fact that people pay more for pay-as-you-go phones or for flat-rate services – they will pay a premium for predictability of billing.

1.2 What the Security Architecture Should Provide

Bearing in mind the security expectations of the mobile phone user, let’s consider what properties the phone needs in order to deliver those expectations.

1.2.1 Privacy

Privacy is a property that is preserved by systems that exercise a duty of confidentiality when handling all aspects of private information. Various kinds of private information may be held on a mobile phone, particularly contact details and calendar entries, but also such things as recording a call being placed and the parties involved in the call. The security architecture needs to ensure that private information is not disclosed to unauthorized parties. This is in order to prevent misuse, either directly, as in the recent case of the disclosure of the American celebrity Paris Hilton’s address book, or, for example, by using contact numbers held on the phone to spread malware to other devices.

1.2.2 Reliability

Reliability is the property of a system that ensures that the system does not perform in unexpected ways. This is often closely related to availability, which is the ability of a system to be ready to operate whenever it’s needed. The security architecture needs to contribute to this by protecting the integrity of critical system components and configuration settings, thus ensuring that they are not changed by unauthorized parties.

It is also worth mentioning, in passing, that apart from the specific security functionality that is discussed in detail later in this book, Symbian OS was designed to provide resilience in the face of errors occurring in application software. Interestingly, this design also helps in protecting
against security problems due to deliberate misuse of system services. Resilience may also be referred to as ‘survivability’.

1.2.3 Defensibility

In addition to the general properties of reliability, availability and resilience, further steps can be taken to help defend against attacks from malware, financial fraud (such as unauthorized use of premium-rate SMS numbers) and the use of the device to attack the network (for example, by sending malformed packets or floods of requests). The security architecture needs to subject such actions to specific controls, which can be used to prevent or limit damage to both customers and networks.

1.2.4 Unobtrusiveness

As far as a mobile phone user is concerned, any security architecture and components should be as invisible as possible. As previously noted, mobile phone users are less paranoid about, and thus more confident in, the security of their devices than are PC users. This is generally a Good Thing [Sellar and Yeatman 1930]. We know that the majority of mobile phone users would prefer not to be bothered with decisions and information about security, that they would rather that, like a car or TV, the phone ‘just works’. The security architecture, therefore, needs to handle as much as possible without explicit user interaction.

1.2.5 Openness

Symbian OS is just one of the components (albeit quite an important one!) that goes in to making a mobile phone. Symbian is part of a larger ecosystem (the value chain is discussed in more detail in Section 1.4) and our success is interdependent with the success of other suppliers – providers of software and hardware components that integrate with our OS. Reducing the openness, by ‘locking down’ the OS and preventing third parties from implementing software to run on phones, might provide a good defense, but it could at the same time destroy much of the value of Symbian OS as a platform to both third parties and customers. The security architecture needs to be open, so that suppliers are able to add their own components, and compelling third-party applications are available.

1.2.6 Trustworthiness

Trustworthiness is an elusive concept, but in any discussion of trust, the first question to ask is: Who is being trusted by whom to do what? In the case of Symbian OS, very briefly, the device manufacturer is being trusted
by the user and by network service providers to provide a device that preserves the first three properties discussed above: privacy, reliability and defensibility. Specific threats to these properties are covered later, but first let’s consider how the security architecture contributes, in general, to the trustworthiness of the device.

For the past 10 to 15 years, a lot of effort in the computer security field has been devoted to securing network boundaries, on the principle that if you stop bad things happening at your network border, then you won’t need to worry about the level of security being applied to individual devices within your network. However, today there are trends towards ‘boundaryless information flow’ (as promoted by The Open Group [Holmes 2002]) and ‘deperimeterization’ (a similar concept, espoused by the Jericho Forum [Simmonds 2004]). The consequence of these trends is that placing controls at the network boundary is becoming ineffective – and a perfect example of this problem is the mobile phone. Today’s smartphone may have several different network connections active at any time (Wi-Fi, Bluetooth, GPRS, etc.). Some of those connections may be inside an enterprise’s network boundary, and other connections outside, both at the same time; and the device itself is likely to be holding commercially confidential information. The device itself, therefore, must be trusted to protect the information on it, and act securely with the networks it connects to, according to a well-defined security policy.

We believe that this reliance on the security properties of an individual computing device represents something of a comeback for the concept of ‘trusted computer systems’ in the sense of the ‘orange book’ [United States Department of Defense 1985]. There is a substantial body of knowledge about the design of trusted computer systems and Symbian has taken advantage of it (see Chapter 2).

### 1.3 Challenges and Threats to Mobile Phone Security

Having considered user expectations of security and, at a high level, what the security architecture of a mobile phone should provide, we must ask ourselves: What specific challenges are posed by the nature and environment of the mobile phone, and what threats does the security architecture need to counter?

#### 1.3.1 The Scale of the Problem

Let’s first consider the potential scale of the problem – there are a lot of mobile phones out there. As we can see from Figure 1.1, the total number of mobile phone subscribers at the end of 2004 was roughly 1.7 billion. (For comparison, the number of PCs in use at that time was
estimated at 820 million [Computer Industry Almanac 2005]. As a first
approximation, we’ll assume that each of these subscriptions represents
one mobile phone. Figure 1.1 categorizes phones by the voice network
technology used, but we need to consider categories of mobile phone
based on their support for add-on software in order to consider security
architecture requirements.

Table 1.1 Categories of Mobile Phone Platforms

<table>
<thead>
<tr>
<th>Open OS (Native Execution)</th>
<th>Symbian OS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Windows Mobile</td>
</tr>
<tr>
<td></td>
<td>Palm OS</td>
</tr>
<tr>
<td>Layered Execution Environment</td>
<td>Java (including µITRON, Linux)</td>
</tr>
<tr>
<td></td>
<td>BREW</td>
</tr>
<tr>
<td>Closed Platform</td>
<td>Proprietary real-time OS</td>
</tr>
</tbody>
</table>

The first category in Table 1.1 includes mobile phones that are based
on an open operating system that allows execution of native binaries
from third parties. This is probably the most attractive target for attackers
because more public information is available regarding interfaces to these
phones’ system software and there are publicly available development kits
for these devices which people can download and use to write software.
These software development kits allow third parties to develop code that
executes ‘natively’, in the same environment as the system software – this
allows powerful and efficient add-on applications to be developed, but it
does, at the same time, introduce a number of security concerns.
The second category in Table 1.1 includes phones that support layered execution environments (that is, an execution environment that does not give third-party software direct access to native code). This category consists primarily of Java-capable phones using a variety of underlying platforms, including Nokia Series 40, in Japan µITRON (pronounced micro-eye-tron) and also, perhaps unexpectedly, Linux. The Linux phones that have appeared so far do not allow the execution of third-party native code; after-market applications for them are typically written in Java. We have also chosen to consider Qualcomm’s BREW (Binary Run-time Environment for Wireless) platform in this category. It is hard to place in that, although it is an execution environment and not an operating system as such, it does allow third-party native code, but only under certain controlled circumstances. Typically there are no public development kits available for phones in this category, but nevertheless security exploits have been reported – in October 2004 there was an exploit [Gowdiak 2004] demonstrated on a Nokia Series 40 phone which used a security vulnerability in the Java virtual machine to enable execution of arbitrary native code.

The third category in Table 1.1 includes all the rest – closed-platform mobile phones that don’t allow third-party applications to be run on them. It is hard to create malware for this category, as for the second category, because the development tools are not available. Development tools do of course exist, but they are typically only used within the device manufacturers’ facilities. Even without public availability of system interface information and development kits, such phones can still be vulnerable to denial of service attacks – for example, malformed SMS messages have been known to cause security problems for closed-platform phones.

Figure 1.2  Open OS Mobile Phone Shipments up to December 2004

Let’s consider in more detail the first category, open OS mobile phones.
Taking the numbers of units shipped up to the end of 2004, we see from Figure 1.2 that there were approximately 25 million mobile phones running Symbian OS. Although Microsoft does not release separate figures for Windows Mobile (sales are reported for all Windows CE devices, which includes PDAs and embedded systems), we estimate that there were somewhere between 4 and 5 million mobile phones shipped running Windows Mobile by the end of 2004. At the same time PalmSource’s reported figures indicate that approximately 2 million mobile phones had been shipped running Palm OS.

The implication of this analysis is that, of the likely targets for mobile phone attackers, Symbian OS is the most attractive since it has the biggest target installed base and offers free public availability of interface documentation and development kits. Because of this, Symbian needed to take a lead in establishing a secure platform, while preserving the advantages of an open OS. Making it hard for third parties to write add-on applications for Symbian OS-based phones, although it might increase security, would be an unacceptable price to pay because it would limit access to markets.

1.3.2 Constraints on the Security Architecture

Mobile phones present some interestingly different challenges from typical desktop or server computer systems when implementing an effective security architecture.

Mobile phones are by their nature low-powered devices – low power in the sense of the speed or ‘horsepower’ of the processor and in the sense of minimizing the drain on the battery (to maximize talk and standby time). Because of this, there are not a lot of spare processor cycles available for any security checks that would introduce a run-time overhead. Even running such checks in otherwise idle processor time risks significantly decreasing the battery life (mobile phone software typically goes quiescent when not actively being used, which allows the processor to enter a power-saving mode).

Mobile phones also, typically, have limited storage, which discourages security solutions that might involve storing large databases of potentially relevant information such as virus signatures or certificate revocation lists. Although mobile phones with more storage are becoming available, our solution must be usable on lower-end devices, and the more storage that is free for user data, the better!

The possibilities for dialog with the user are also, typically, limited on a mobile phone – the screen is small and text entry may be slow and difficult by comparison with a PC monitor and keyboard. User prompts, therefore, have to be short and to the point and the security cannot depend on significant amounts of text entry such as pass-phrases.

To put it simply, you cannot afford to waste energy on anything.
1.3.3 The Challenge of Connectivity

As the technology of mobile phone networks develops and with the take-up of wireless LANs and other wireless short-link connection methods, mobile phones are increasingly being provided with many, potentially simultaneously active, data connections. Such connections include 3G/UMTS data (with speeds approaching that of typical domestic fixed broadband), 2.5G data (EDGE and GPRS), Wi-Fi (802.11b), Bluetooth and infrared. As mentioned previously, this multiplicity of connections is an example of the increasing ineffectiveness of controls at network boundaries. It is entirely possible that a modern phone could be communicating via Bluetooth to a company laptop on the company’s internal network, while being directly connected to the internal network via a Wi-Fi LAN access point, and also accessing the Internet via the 3G phone network.

Each of those connections is a potential point of attack. In this example, Internet Protocol (IP) traffic is flowing over all the connections, so the mobile phone could potentially act as a gateway between the networks, bypassing firewalls and access controls that may be implemented only on the company’s internal network. The security of the phone and the data on it must not depend on any particular connection that an attack may arrive on.

It’s also worth noting here that it is in the nature of wireless connectivity to be unpredictable. Coverage can be lost altogether, or the actual physical network access point that you’re connected to can change almost transparently when you’re roaming – either going to another network cell or actually roaming to a different provider. The security architecture must not assume continuous connectivity. This is important when doing things such as revocation checks on digital signatures. It would be very annoying, for example, not to be able to install and play a game you had previously downloaded because you happen to be on the London Underground (which reportedly will not have mobile network coverage on the underground platforms until 2008) and, therefore, can’t check a signature online with a Certificate Authority.

1.3.4 Malware and Device Perimeter Security

One of the most well-publicized security threats to open mobile phones is malware: trojans (malicious programs which masquerade as benign ones), worms (malicious programs which send copies of themselves to other devices) and viruses (malicious code which attaches itself to legitimate files and is carried along with them).

Malware targeting Symbian OS started to appear in June 2004 and over the following year we saw increasing numbers of new strains (13 in all). The first was Cabir, a worm that spreads via Bluetooth. It was stated by its authors to be a ‘proof of concept’, not intended to cause harm to mobile
phone users, but to alert people to deficiencies in mobile phone security (Symbian’s platform security project was well advanced by this point, so Symbian was well aware of the need for improved security without this reminder!) The worm’s authors also released the source code, which resulted in many minor variants of the code being implemented and released ‘into the wild’. One variant has been classified as a new strain (Mabir) as it includes the ability to spread via MMS messages as well as via Bluetooth.

We have also seen trojans such as Skulls, which appear to be harmless programs, however, when they are installed they corrupt configuration settings and cause legitimate applications to stop working. There has been, as of the time of writing, one virus, Lasco, which infects the installation files of legitimate applications.

One thing to note is that, thus far, none of this malware has been able to bypass the Symbian OS software install security controls. For the Cabir worm to infect a mobile phone, the user of the targeted phone has to explicitly acknowledge the three security-related dialogs shown in Figure 1.3.

![Figure 1.3](image)

**Figure 1.3** Device Perimeter Security Prompts

The user first has to acknowledge that they want to receive this Bluetooth message; then, because the installation file is not signed by a trusted authority the user is warned that the source of the software cannot be verified; lastly, the user is presented with the name of the application (or of the malware, in this case) and has to confirm that they do want to install it. We call this the ‘perimeter security’ for the actual device itself – in order to get executable content on to the device you have to go through this install phase.

Perimeter security has been a feature of Symbian OS since the early versions, but it is not the complete answer to malware. Even though the user is required to make a deliberate, conscious step to allow malware to install itself on their phone, it is still found to some extent in the wild. Ed Felten of Princeton University is well known for saying, ‘Given a choice between dancing pigs and security, users will pick dancing pigs every time’ [McGraw and Felten 1999], and indeed why shouldn’t they? Users are entitled to expect their mobile phones to help them defend
against malicious attacks, and most people have no desire to become security experts.

We have not yet seen an explosive spread of mobile phone malware, certainly nothing on the scale of worms such as Code Red and Sasser in the PC world, and we have some reason to believe that we never will. The need for user confirmation before software installation significantly limits the rate of spread, and, referring back to the figures on the number of mobile phones in use, at the end of 2004 something less than 2% of all mobile phones were running Symbian OS, so, if you picked a random target, it probably wouldn’t be susceptible. Nevertheless, the percentage of phones running Symbian OS will increase in future, and there is a risk of flaws in the device perimeter security which could be exploited, so Symbian’s platform security architecture is designed to minimize risk from malware even after it has succeeding in installing onto the mobile phone.

1.4 How Symbian OS Platform Security Fits into the Value Chain

Having considered the high-level goals of our platform security architecture and reviewed the major challenges facing us, it is worth spending a little time considering how developers actually deliver something into the hands of the mobile phone user. There are several kinds of organization that contribute to the development of a mobile phone product, creating a ‘value chain’, and, as we have previously noted, there is significantly more to the user experience than the product delivered in the box by the device manufacturer – the phone is an end point for network services and a repository for after-market content. Symbian, as an OS vendor, has a crucial part to play in providing the building blocks to construct a secure solution, but it can’t solve all the problems on its own.

All of the links in the value chain shown in Figure 1.4 have a part to play.

The OS vendor, their technology partners and the device manufacturer need to work together on improving platform security in the construction of a device. The OS and the device hardware need to be tightly integrated with technology from specialist security suppliers, such as anti-virus and firewall vendors, to make these components work effectively together and minimize the risk of things being overlooked.

After-market software vendors, content providers and content distributors – although not contributing to the mobile phone as it is sold to the user – still have an important part to play in building the user’s trust in the security of the platform. In particular, the use of digital signatures helps to demonstrate that the supplier of the software, or other content, is willing to stand behind the quality and security of what they are delivering. There are several signing programmes for mobile phone software now
in place (we will discuss the Symbian Signed program in more detail in Chapter 9). The more content providers and software vendors that take advantage of these to promote trustworthy channels for distributing content to phones, the more unusual it will be for the user to get security warning dialogs appearing on their phones, with the desired result that mobile phone users will pay much more attention to such dialogs when they do appear.

Enterprise IT administrators and network operators also need to be contributing to mobile phone security; they can provide infrastructure for managing security on the devices, particularly for application lifecycle management: provisioning, updates, revocation, patching and so on.

Lastly, end users also have a part to play. Whilst the industry should resist the temptation to assign responsibility to the end user as a ‘cop out’, it does need to educate mobile phone users to take reasonable precautions in their own interest. They should be very careful when they receive software they weren’t expecting (it is instructive to note that the advice regarding the original Trojan horse, ‘beware of Greeks bearing gifts’ [Virgil 19BCE], still applies 2000 years later to software trojan horses – be suspicious of unexpected deliveries). Users should also recognize that regular backups are prudent in anticipation of physical harm to their phone. It seems that there have been more mobile phones dropped down toilets [Miller 2005] than have been disabled by malware!
1.4.1 Security as a Holistic Property

We can also look at the shared responsibilities for providing security in another, more theoretical, way (see Figure 1.5).

![Security Building Blocks](image)

The ultimate aim of a secure system is to give rise to trust and confidence on the part of the system’s owners and users that clear security policies are being enforced. A common mistake is to look at one or more of the foundation blocks in Figure 1.5, and to assume that, if one of those security features is implemented, the system’s security is, therefore, improved. This will only be the case if that feature is integrated into the system as a whole in such a way that it supports one or more of the security functions above it; which in turn gives rise to a system security quality above that; which contributes either to the enforcement of defined rights and responsibilities or the assurance that such enforcement is effective; which then, finally, promotes trust and confidence in the system. Put briefly, security is a holistic property; it is more than the sum of its parts (or, if done badly, rather less than the sum of its parts).

Let’s work through a specific example: we want the mobile phone user to have confidence that their contacts data will not be disclosed to unauthorized parties. We might decide that a good way to do this would be to encrypt the contacts database. However, this in itself doesn’t achieve anything because, if the attacker can get the encryption key, then they can still get the data. The encryption key needs to be protected by an access control mechanism that restricts access to it; which, in turn, must be part of an authorization mechanism that decides which entities should be allowed that access. The authorization mechanism needs to enforce defined rights and responsibilities. For example, a network server could have the right to access contact data for synchronization on the understanding that it has the responsibility for protecting that data when it is stored off the phone. Finally, the security policy being followed by
the network server should be clearly expressed to the users, so that they have confidence in using the service.

To provide security in this holistic way, all of the components of the system (and for mobile phones this includes after-market software and network services) need to be cooperating.

1.4.2 Security Incident Detection and Response

Much of this chapter has focused on technology and behavior that is intended to prevent or mitigate security problems. As Bruce Schneier eloquently argues in [Schneier 2000], prevention is only the first part of the story; we also need to look at detection and response because prevention is never 100% effective.

The first person to notice that something is going wrong is likely to be the mobile phone user and they will probably refer the problem to the mobile network operator first, as the problem may appear as unexpected behavior of the network services, or, perhaps, a billing anomaly. From this point we need to ensure that the report gets to the correct party as quickly as possible. Security incident-handling processes are well established for server systems: US-CERT was founded in 1988 at the Carnegie Mellon Software Engineering Institute, and does a fine job of coordinating reports of security incidents and the response from various vendors in the server value chain. At the time of writing, the scale of security incidents for mobile phones is much smaller, but the industry would probably benefit in future from a similar body, providing independent coordination of security issues for mobile phones.

There are two aspects to responding to a security problem. The first is repairing the defect (assuming there is one) which led to the problem. This could, potentially, be a very large-scale operation; there could easily be 10 million phones in use that are running a particular system build which needs a software patch. Over-the-air patching of the system image on the phone may well be the most effective way to do this, but we need to make sure that this process itself is secure. There is a danger in introducing such functionality, in that it might actually create more problems than it solves by opening up a way to bypass the security controls implemented on the mobile phone and replace critical parts of the operating system.

The second aspect of response is recovery. Once the cause of the security incident has been addressed, how will the user’s data and services be returned to operation? An important part of this is the ability to recover from configuration problems, which could also arise from non-malicious hardware or software failures, or ‘user error’. In order to address this, we recommend that mobile phones should be capable of a ‘hard reset’, or being booted in a ‘safe mode’. In the case where user
data has been lost, there should be an easy way to recover it; again this is not just an issue for security problems – many mobile phones are lost and many are physically broken. As we’ve already suggested, the typical mobile phone user won’t care where their data is actually stored – on the phone, on a server, or mirrored between the two – as long as they can get a replacement phone and still have their old data on it. Over-the-air synchronization or backup of the user data from the phone, without user intervention, may well be the best way of achieving this, and may be an additional service.

1.5 How Application Developers Benefit from the Security Architecture

There is a risk that the security features we are discussing in this book could be perceived as an inconvenience rather than as a benefit, particularly from the perspective of an application developer, who will have more things to worry about as a consequence of them. As we have discussed, the primary beneficiary of these security improvements is intended to be the mobile phone user, but it is important to highlight several benefits that they bring specifically to the application developer.

Firstly, by increasing the users’ trust and confidence in open OS mobile phones the entire value chain should benefit. Users will be more willing to install after-market software when they have some assurance that they are protected from the worst effects of malware. They will be inclined to buy more after-market software and will appreciate the benefits of open OS mobile phones more. Thus they will buy more of these phones in preference to closed-platform phones or those with layered execution environments, creating a bigger market for after-market software – a ‘virtuous circle’.

Secondly, the ‘least privilege’ aspects of the platform security architecture (see Section 2.4) mean that any unintentional defects in add-on applications (such as a buffer overflow) are less likely to result in serious security vulnerabilities, thus protecting the reputation of the application developer.

Thirdly, the platform security architecture means that security services can be provided in an open way, avoiding ‘security by obscurity’. Symbian hopes that this will promote ‘best of breed’ security services to be provided by the platform and by specialist security vendors to add-on applications, thus avoiding duplication of effort and allowing developers to concentrate on their own areas of expertise.