Security issues for downloaded code in mobile phones

by D. Babb, C. Bishop and T. E. Dodgson

'Software defined radio' (SDR) is a technology that will appear in future generations of mobile phones, i.e. following the third-generation mobile phone technology that is currently being defined and developed. Early versions of 'pragmatic' SDR will allow the terminal to be reconfigured at any level of its protocol stack. Ultimately, the 'pure' SDR technology will allow a mobile phone or terminal to have its air interface software configured or reconfigured by other software (or software parameters) that have been downloaded to the terminal, e.g. over the air, or from a remote server via the Internet and one's personal computer (PC). A number of security issues arise with downloaded code that implements the air interface functions, and these may not be obvious simply from looking at the way PC software is updated on-line today. This paper starts with an outline of the code that allows a mobile phone to operate over a particular air interface. This sets the baseline for a discussion of the security issues surrounding the change of this code from one that is fixed and downloaded once only, to code that is reconfigurable during the life of a product.

1 Introduction

Today's mobile phones are small, hand-portable, battery-powered devices that can communicate using radio frequency signals on one or more of a variety of air interfaces. The Global System for Mobile communications (GSM), as specified by ETSI (European Telecommunications Standards Institute), is currently the most widely used digital, 'second-generation', air interface standard, though other systems are in use in some regions of the world.

A new, 'third' generation (3G) of mobile phone technologies, which can be grouped under the International Mobile Telecommunications 2000 (IMT 2000) family of standards and which include (amongst others) the Universal Mobile Telecommunications System (UMTS*) standard, are now being launched in some areas (though at the time of writing there are no commercial UMTS networks). These technologies offer higher data rates and hence a wider range of services can be supported by terminals. Some 3G systems bring with them a shift that involves the use of wideband code division multiple access (W-CDMA) technology.

Research on a future generation (also known as 4G, 'Beyond 3G', B3G and NG (Next Generation)) of mobile communication systems is in its early stages and products will probably start to appear on the market in around 10 years. Such systems will almost certainly employ software-defined radio (SDR) techniques in the mobile equipment. The air interface and communication method of a terminal employing SDR might be changed, for example, by reconfiguring both the software and the hardware, e.g. the firmware (FW), digital signal processor (DSP) and multiple RF sections utilising microelectromechanical switches (MEMS). Such reconfiguration might be achieved by downloading software.

The definition of SDR, i.e what it is, can vary depending on what school of thought is being consulted. The two extremes could be regarded as the 'pure' software radio, which is able to reconfigure its air interface software, and the 'pragmatic' software radio, in which any level of the protocol stack can be reconfigured. This paper encompasses both extremes, the air interface configuration concept, together with multiple protocol stack configuration. The pragmatic SDR concept can be taken to generate a road map towards the pure SDR, working down from the application, presentation and session layers, through transport and network layers to, ultimately, the data link and physical layers (the latter two layers comprising most of the processing pertinent to the

*The term UMTS is no longer in favour within 3GPP (3rd Generation Partnership Project), which is now developing the standard, as it is thought to be too Euro-centric (having been defined by ETSI). The preferred term is '3GPP System'. However, for the sake of clarity, and given that GSM/GPRS is also a '3GPP System', the term UMTS is used in this paper as the default third-generation system name.

Fig. 1 A typical GSM phone today, the Samsung T100

air interface itself).

This paper identifies software security issues associated with downloading code for reconfiguring a future generation of SDR mobile phones.

Mobile phone design and architecture issues related to the download of software code are discussed from a *generic* point of view only, as there are many different approaches around the world to the associated security issues. This paper does not claim to represent the views of any specific research programme, standards body or manufacturer, although there is a definite focus on the European approach. Because of space constraints, the discussion is confined to basic mobile phones, as combined personal-digital-assistant (PDA)/mobile phones add an extra dimension to the hardware and software architecture that needs separate and careful attention.

As this paper is about security, it is perhaps useful to state the meanings of the word 'security' as it is used here. In a general sense, 'security' can refer to:

- precautions that are taken to protect against theft, espionage, or other danger
- the state of being free from danger, damage or worry
- something given or pledged to guarantee payment.

From the mobile telecoms network point of view the 3rd Generation Partnership Project (3GPP), which is responsible for defining the UMTS standards, defines security[1] as:

> 'The ability to prevent fraud as well as the protection of information availability, integrity and confidentiality.'

However, the concept of downloadable applications, and the requirement for their secure interoperability across mobile phones from different manufacturers, has also led to the standardisation of a Mobile application Execution Environment, or MExE[2] (see later). This includes the idea that security mechanisms should also:

> 'prevent attack from unfriendly sources or transferred applications...'.

This introduces some of the elements of security as they will relate to SDR, in that mechanisms are required to ensure that downloaded software is fit for purpose, and that it must neither 'attack' (whether maliciously or otherwise) the equipment nor have a detrimental impact on the environment in which that equipment operates.

A major part of setting the scope is to detail the relevant points of the current status of code in mobile phones, whether downloaded or otherwise, so before looking to the future it is useful to look at the current situation. Many of the reasons for the current status on downloaded code for mobiles will have to be examined in order to understand what is needed to facilitate the download of code for reconfiguration in an SDR environment.

2 Today's second-generation digital mobile phone

Fig. 1 shows a typical GSM product, the T100 from Samsung. Not much downloadable code can be used on phones such as this. Most are limited to using downloaded ring tones and screen graphics. The reasons for this lie in the history of the mobile phone and the rules and regulations that have been required to get these products to market.

Regulatory issues

There are three main types of rules:

- Regulatory: For a mobile phone to carry the European Community's CE mark it must conform to the radio (basic radio frequency (RF) parameters, interference, etc.) and safety regulations in force throughout the European Union (EU). Compliance is determined by parameters of the software driving the hardware as well as by the performance of the hardware itself.
- Type approval*: The phone must comply with the full rules of the protocol stack and function correctly in the RF domain. Again, this depends on the software executing correctly and on a very tight relationship between software and hardware.
- Operator approval: This ensures that the phone functions in the desired manner when connected to a particular operator's network. Network configurations are very complex and the network operator has many options to choose from (for example, very few networks rely solely on equipment from a single supplier). Operators typically have to balance the trade-offs that occur between capacity (number of users on a system), services offered (which is related to the associated bit rate), coverage, user priority, system grade of service etc.

*Formal type approval itself no longer exists, as it has been replaced under the Radio and Telecommunications Terminal Equipment (R&TTE) directive by a regime based on a manufacturer's declaration of conformance. However, the term is still in widespread use to describe tested compliance to the requirements of the core specifications.

This has led to a basic design philosophy according to which software is downloaded once only—during production—and then made secure. Some software upgrades are available at service points, but in general the latter tend to replicate the facilities available during production.

Some security issues

It may come as a surprise to learn that almost every user feature in today's digital mobile phone has an impact in terms of security, and a large part of this is driven by software execution in the phone. Table 1 lists some major user features of today's mobile phones and their associated security considerations.

In addition, we expect a phone to operate at all times; indeed there are regulatory obligations that require the ability to make emergency calls at any time (providing the battery has enough power and there is network coverage at the location of the mobile phone). This must be reflected in the design of both the hardware and the software.

Basic architecture

In this subsection we discuss the basic architecture of a mobile phone, as this will serve as a reference in subsequent sections on downloaded code security issues in future phones.

Fig. 2 provides an overview of a typical mobile phone software architecture. The hardware is fixed and driven by the software (a point of discussion when looking at SDR technology), so that the performance and characteristics of the hardware and the product are defined by software. Today's assumption is that the software, and in particular that software related to the terminal's air interface capabilities, is supplied at the point of sale and doesn't change. This implies that there is no download of software code after purchase for the purpose

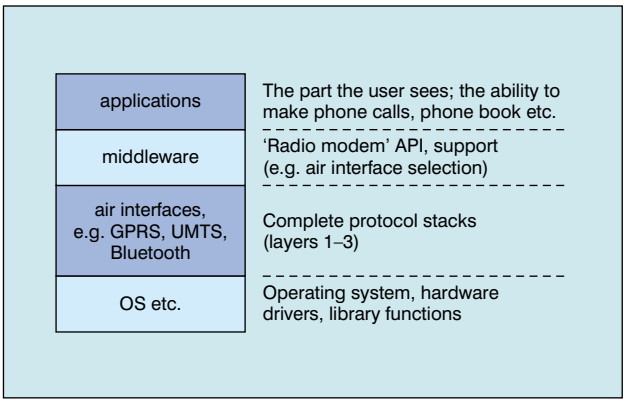

Fig. 2 Typical present-day mobile phone software architecture

of configuring software below the application layer (as mentioned above, it is already possible to download simple applications such as ring tones and screen graphics). The hardware has safety features (e.g. power supply control and RF power control) to make sure that the product behaves within preset limits.

The software may well reside in several components within the phone, as there will be at least two processors executing various parts of the code:

- a main microprocessor for executing radio modem software and user applications
- a digital signal processor (DSP) for executing baseband functions such as speech coding, channel coding and digital modulation.

This raises additional issues when designing the software security architecture if it is assumed that security threats can occur at run time and can affect any part of the terminal.

Table 1: Impact of user features on phone security

Phone feature	Expected security/security impact
Making voice and data calls	Nobody should be able to listen to a call by 'eavesdropping' or recording the radio signals in any way. This security is mainly achieved through software execution. The software execution is secure as it cannot be changed in the phone—it is a one-time only secure load.
TFT LCD (thin film transistor driven liquid crystal display) screen with 4096 colours	The security impact is indirect: any downloaded code in the form of pictures and wallpapers must not interfere with the operation of the phone, which must work and display correct information.
16 polyphonic ring tones	The security impact is indirect: any downloaded code in the form of new ring tones or tunes must not interfere with the operation of the phone. There is also a minor safety issue: the phone must not cause physical side effects, such as harm to the ears due to high volume.
Voice dial, voice command	This feature provides (sometimes unwanted) security in that only one person—the main user—can use it on any one phone, as this facility has to be 'trained'.
Voice memo	This generally uses the same technology as voice dial. The security considerations relate to how and where the data is to be stored. If the memos are stored on an external memory card, such as Smart Media or Compact Flash, then the user may want to protect this data in some way when it is removed from the phone.
Personal organiser	The user data must be secure: it should not be possible for an unauthorised entity to read it out of the phone in any way, e.g. via the air interface.
Games	Games tend to be processor and memory intensive. They must not use so much memory and processor power that they prevent the phone from maintaining its connection to the network or from receiving calls.

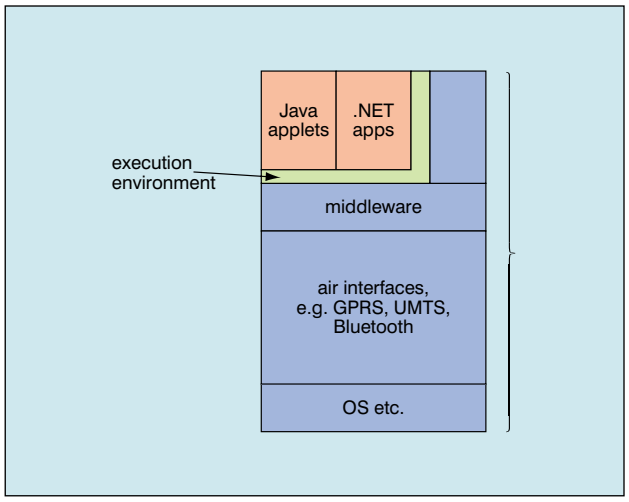

Fig. 3 Location of the MExE in the software architecture of a mobile phone

Additionally if a Bluetooth device or a wireless local-area network (WLAN) device (for example one that conforms to the IEEE 802.11b standard) is included in the mobile equipment (and such equipment is beginning to appear on the market), this will probably introduce additional hardware and memory support, although from the overview software architecture point of view Fig. 2 remains valid.

The physical memory for the phone may or may not be shared between the various processors, and may be a mixture of flash memory, static random-access memory (SRAM) and read-only memory (ROM). Also it may or may not have hardware memory protection, which is another detail to look at in the overall security analysis.

Another point, which from a customer's viewpoint is very important and should therefore be considered when looking at mobile phone security architectures, is that many mobile phones are expected to run for up to a week without charging! This is achieved by fine-tuning the software and hardware architectures so that the phone operates in a highly power efficient manner, and it is also dependent on the air interface design. Complex security algorithms tend to use a lot of processor power and memory, if only for a short time, and this may have a significant influence on the life of the phone's battery if we select algorithms that are complex and run many times. One solution that reduces overall complexity is to partition the memory into several protected areas using a memory protection unit, so that each stratum runs in a single address space only. Unfortunately there is only a limited amount of memory to achieve this, and memory is expensive.

This all sounds like bad news, however solutions have been found that make current, digital, mobile phones very secure, very reliable and not too 'power hungry' (in comparison, for example, with the average home PC, whose associated Internet protocols were not really designed with privacy in mind).

The modern digital mobile phone has three basic security architectures. These are independent but all need to be present and correct for the phone to operate correctly. The three basic architectures are:

- *software security:* This is achieved by virtue of the fact that the software cannot be changed by the user (except in the case of simple downloaded applications). Today software security is tightly linked to hardware security as this also does not change.
- *subscriber identity security:* This is typically taken care of by the subscriber identity module (SIM) card and the system itself together with the associated signalling protocols. A valid SIM card (together with valid terminal equipment identification) will allow a user access to the corresponding system. The SIM card itself can be protected through the use of a valid personal identification number (PIN) which, when activated, will require that it is entered every time the phone is switched on. The phone itself may have a user blocking code (independent of the SIM PIN) that would also, when activated, require input to operate the terminal.
- *air interface security:* This ensures that someone who is deliberately searching for a radio signal cannot understand the radio signals that are sent and received by the phone. Moreover, if someone does try to listen with malicious intent, they cannot easily gather useful information, such as the subscriber identity. This is achieved by having a fairly complex air interface in which the digital data are encrypted. This is standard in GSM, where a robust time-varying encryption procedure, makes 'listening in' impossible.

Now compare this security to that provided when logging on to the Internet or an Internet POP3 mail server from a home PC. Although the details can vary depending on the Internet service provider and software packages being used, it is likely that at least your user name and sometimes your password will be sent as plain text and, once logged in, the data will not normally be encrypted. At this point you get into a major argument from the security point of view as to how easy it is to listen to a mobile phone or somebody's home phone, asymmetric digital subscriber line (ADSL) or cable connection, but that debate is not for this paper. Suffice it to say that once the radio system has gone digital, with a few selected security mechanisms such as encryption on the air interface, then it really is very secure; but no matter how complicated you make the security, someone who really wants to, will find a way around it.

This section has outlined security considerations for existing second-generation mobile phones, showing that downloadable code is limited to simple applications and cannot be used to configure software below the applications layer. However, as phones and the applications that can be downloaded to run on them become more complex, an additional level of security is considered necessary to continue to ensure the integrity both of the applications and, as a consequence, of the phones on which they are running. Indeed, it is the integration of a (very secure) mobile phone technology with a (comparatively insecure) Internet technology that raises the question of how to make the resulting, integrated, system itself secure.

3 The protected application environment: Mobile Execution Environment (MExE)

The Mobile Execution Environment (MExE) is an open specification, first created by ETSI and now maintained and developed by 3GPP, that specifies a standardised environment for executing applications in enhanced second-generation, e.g. General Packet Radio Service (GPRS) and third-generation (e.g. UMTS) mobile terminals[3]. It has already been implemented on some of the current GPRS phones and certain aspects of it, i.e. at least those relating to download and security, will probably be a standard feature in dual-mode GPRS/UMTS phones. The aim of MExE is to allow software to be downloaded to mobile phones, with a specific aim of downloading applications (as opposed to software for reconfiguration of software/firmware below the application layer) and providing a standardised interface between the application and the phone (or mobile telephony) air interface. The execution environment is shown in Fig. 3 as a wrapper between the Java applets and .NET applications, and the rest of the phone. In particular, the aim here is to provide standard interfaces to phone features, so that downloaded applications can be run in a secure environment and in a standardised manner regardless of the phone on which they are being run.

Note that this is not the only software architecture approach that can be adopted. Moving the applications to a separate processor and operating system, such as Psion's EPOC operating system, can have the same effect as putting a wrapper between the applications and the phone air interfaces. Another solution is Qualcomm's BREW™ (Binary Runtime Environment for Wireless), which has a slightly different architecture but gives the same overall result[4].

Classmarks are used in the current MExE specification to categorise mobile devices based on their performance. Today three classmark types have been specified:

- Classmark 1 supports Wireless Application Protocol (WAP), used by all WAP browsers
- Classmark 2 supports PersonalJava (a subset of the Java 2 Platform, Standard Edition), used for Java MIDlets* with no interface to phone functions
- Classmark 3 supports J2ME* Call Connected Limited Device Configuration (CLDC) and Mobile Information Device Profile (MIDP), which allows Java MIDlets to interface to the phone air interface for limited phone functions.

A classmark 4 is planned, which will be for Microsoft's .NET environment[5] and downloaded programmes written in C#. Though only classmark 1 products are currently in use, between the time of writing and publication of this paper classmark 2 and 3 products should start to be available.

Even in these environments the downloaded software will have various levels of trust, from untrusted (source unknown) to trusted by either the mobile phone manufacturer or the operator. The support framework for defining and testing trust is still being designed and discussed at the time of writing this article, but one key point will be independent testing of both the execution environment within the mobile phone and how the applications get 'trusted' status.

There are a number of important points to note about MExE as currently specified:

- A capability to make emergency calls is always present.
- The execution environment tends to augment the functions on a mobile phone (e.g. by addition of a WAP browser or games) rather than replace older phone software.
- Applications running in the execution environment tend to have lower priority than voice or data calls using the air interface.

There is no reconfiguration of the air interfaces. However, with the march towards SDR, there has been some discussion about future MExE classmarks encompassing the air interface part of the software architecture. Current thoughts will be to manage the configurability in such a way that there is no chance of the air interface becoming 'contaminated' through the various threats mentioned earlier. The architecture would strive to make a clear demarcation/interface between the air interface software and application software.

4 Fixed multiple radio access technologies: third-generation phones

In the coming years the digital mobile phone will have several new important technology features, the major one being the ability to work with multiple fixed air interfaces (reconfigurable air interfaces will be discussed in Section 5, on SDR), which will enable 'seamless' roaming, for example between UMTS and GSM/GPRS. Indeed, some of today's GPRS phones are already equipped with a second air interface in the form of Bluetooth. Another feature will be the increased speed of the data link that is available via the air interface. If you have GPRS today then speeds of up to around 48 kbit/s are available, depending on the network and a number of other variables. Speeds higher than this are deemed possible and can be obtained from the appropriate system specification. With the arrival of UMTS, data rates of up to 384 kbit/s should be available in urban areas, with an upgrade route to much higher speeds under certain conditions over the next few years (e.g. using High Speed Downlink Packet Access—HSDPA). These improved air interface data rates enable improved services, for example fast Web browsing and quick download of medium-sized files.

To support these features the software, as well as the hardware processing power, will grow dramatically. The software architecture in Fig. 2 is still valid, but the security aspects grow since one must consider all the possible download services that might be obtainable with a high-speed interface. Some of these features have already been

*A reduced version of Java called Java 2 Microedition (J2ME) has been developed for mobile information devices. Applications conforming to this standard are called 'MIDlets'.

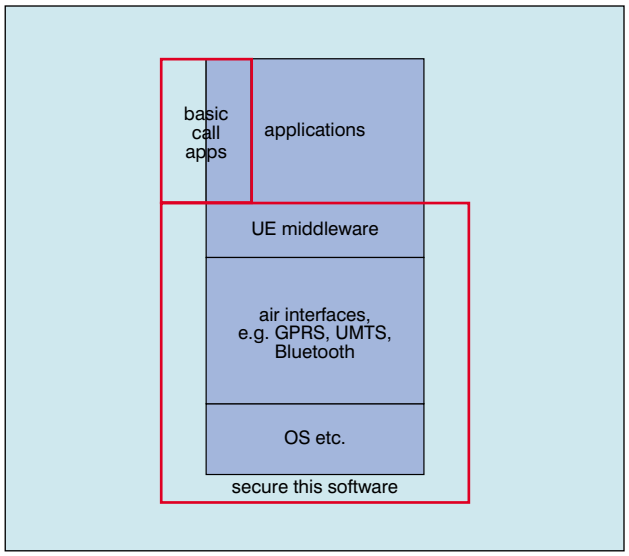

Fig. 4 Basic areas (in red boxes) of UMTS phone software architecture that must be secured

considered, such as that on the Mobile Execution Environment (MExE), which will have enhanced capabilities once there is a high-speed air interface. To be secured then is the area shown inside the red boxes in Fig. 4, namely the core air interfaces and the basic applications for making and receiving calls.

With UMTS the subscriber identity module becomes a USIM (Universal SIM)—the basic technology is the same but there are some enhanced features. The security of the UMTS air interface is enhanced over that of GSM (though it should be noted that the GSM air interface is already extremely secure).

User data memory cards

These are not tied to any particular air interface. However, as they can carry quite a lot of data and as UMTS provides higher data rate services than does GSM/GPRS, it is quite likely that 3G phones will be developed to take, and be supplied with, standardised memory cards such as those already in widespread use for digital cameras, MP3 players, etc. Smart Media, Compact Flash and the Sony Memory Stick are widely used examples of such cards, and today can typically store up to 128 Mbyte of data, although 256 Mbyte versions are becoming available. This is a considerable amount of data—the total memory inside a UMTS phone will probably be in the region of 20 to 40 Mbyte, depending on the configuration and functionality. The memory cards in themselves do not present a threat as long as they are treated as pure data devices. At the low level they are usually configured to act like floppy disks.

There are however some security aspects to be considered of the data that is read from and written to these devices. Firstly, data that is read from the card should only go to pre-assigned or controlled areas in the phone. If the data is phone-specific data, such as phone books or screen pictures, that is fine; but if the data is something that runs some kind of executable code on the phone, such as a Java applet, then there is a real security threat: does the code contain a virus, for example? The approach on PCs today is to virus check the external storage media before copying files to the PC; the same approach should probably be used in the case of user data memory cards for phones.

Digital rights management (DRM)[6] is also an issue with such memory cards, and the same will apply to ring tones with the advent of the increased polyphonic capabilities that are appearing on phones. Security is required when transferring any copyright data, such as music, via any part of the mobile phone, as opposed to just playing the media. The problem here is that in a UMTS phone (or even in an earlier generation phone that has a WLAN interface), multimedia data such as music can be received, played, and then sent on to many people very quickly. If the data contains marketing information, that is probably the desired effect, but if it contains copyright material that has been purchased on a limited licence, such as music for personal listening, wide distribution is not a desired result.

5 Software-defined radio: future generation phones[7,8]

As mobile communication systems evolve, there will be a significant change in the architecture of mobile phones. It is expected that future generation mobile phones will not just include all the required air interfaces, but will be able to reconfigure the air interface software and, to a more limited extent, the associated hardware. Fig. 5 presents an architecture roadmap in which the end-to-end timescale is possibly between 10 and 15 years.

The first step in evolution will be the addition of more air-interface variants (centre of Fig. 5), probably using a technology similar to that to be used in 3G multimode phones. Further into the future (right of Fig. 5), there will be a significant change in architecture from the fixed hardware and software architecture described in earlier sections to one in which the architecture can be reconfigured.

In Fig. 5 the areas where and the extent to which download of software code is permitted have been highlighted. This roadmap is a view of what might

A scenario

Consider the following simple scenario:

You are on your way from your office to a meeting and using a (W-CDMA) mobile phone in an urban area in 'always on' mode to connect to your mail server, collect e-mails as they appear and automatically download them to your laptop. The meeting is in a medium-sized town, and when you arrive and settle in to the meeting, the network notices that you have stopped moving, but are still receiving calls on your e-mail. Your accounting preferences allow you to roam to other types of network.

In the location of your meeting there is a public WLAN that is known by your service provider and, as their voice network

happen; it can be challenged, and perhaps we should challenge it more. One of the reasons for not showing software downloads in the air interface area in the close future is that today this air interface is treated with care. Until recently the radio regulators (effectively the government), a type approval agency and the radio network operators were all involved (as outlined in Section 2) for every single phone, to make sure that it performed in an exact, predetermined, repeatable and approved manner, in particular following the appropriate air interface standard. The reason for all this effort is that the air interface is a shared medium, unlike a computer network or home phone or cable connection, which effectively are not. The consequence of this is that the software and hardware that are responsible for implementing the complex air interfaces have to be designed so that they do not interfere with each other or stop others from using their system. The advantages of doing all this work are also clear: a mobile phone can 'roam' and be used in other countries; it can be used on networks with equipment supplied by different manufacturers; and it will not stop other people using the network.

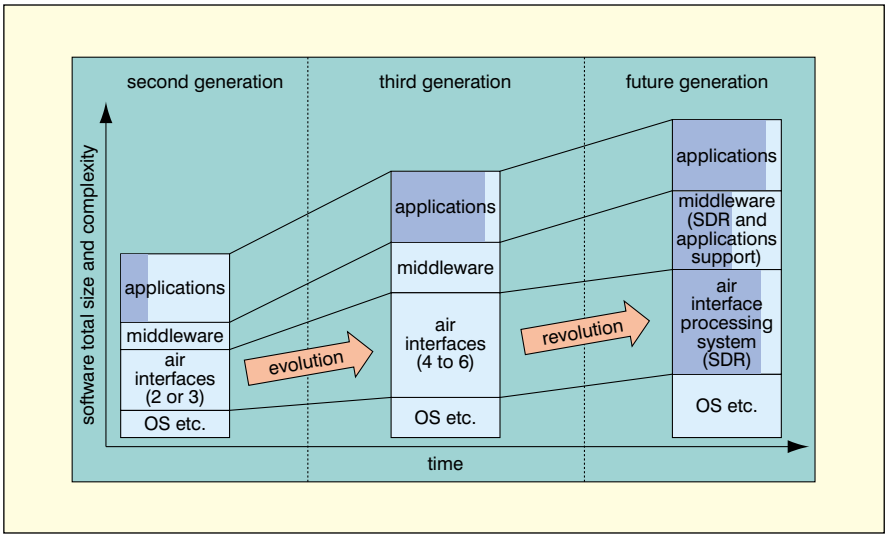

Fig. 5 Roadmap of mobile phone architecture. Areas where download of code is permitted are highlighted.

SDR adds to the roadmap those parts of the software architecture that can be downloaded (from any of the diverse sources listed later). In the more distant future (right of Fig. 5) not only will software download be targeted, but also dynamic reconfiguration of the hardware will be addressed, so that the result is that best suited to the user requirements. The consequences of moving to this type of system are enormous. The security checking procedures in the downloaded software, and in the supporting software system and environment, will have to answer the following questions:

- Have I got the software I thought I wanted—does it perform the relevant function?
- Is it authenticated, authorised and accounted (paid for)? This is commonly abbreviated as AAA[9].
- Does it work in my hardware configuration?
- Does it maintain its regulatory status? (Can I still use it and not interfere with the network or other users?)
- If it turns out not to pass 'security' checks when downloaded, can the phone still perform as it was without further downloads?
- While upgrading the software or reconfiguring the software or hardware, can I still make calls?
- How do I (if I want to) make sure before performing the download that the downloaded code will be both usable and acceptable?
- How do I check for a 'good' source?
- Does the new software work with my existing software?

In a similar situation on a PC, generally during any software upgrade there are periods when the PC is not usable, either while the machine is being restarted or while key parameters are being changed. Is this acceptable on mobile phones?

To answer these questions the software reconfiguration systems will have to run a complex database system that knows not only the limits of your hardware, but what you have paid for in terms of services and what your current software is. This raises the question of where this database should reside. It is probably too complex for the phone, which will however have to hold basic data. Using this database, the software reconfiguration system will have to produce new software that can be downloaded to a phone, and then reconfigure the phone's hardware to run in a totally new configuration.

A situation that could arise with SDR is that the service provider might decide to reconfigure the air interface on behalf of the user, in order to get better performance from the network (see the Panel 'A scenario for software defined radio'). How will this take place? If the phone is being used at the time it might be disruptive. If the phone was in an 'always on' data connection downloading e-mail

for SDR

is very busy, they decide to download some new software to your phone, which is of the SDR type. Your data service then switches the e-mail download over to the WLAN, since this is not only cheaper for you but also allows the W-CDMA network to be reused for other services.

The possibilities for network management, roaming, getting the service you want on demand, etc. are extremely interesting, but only if the software security is good and makes it all work first time, every time.

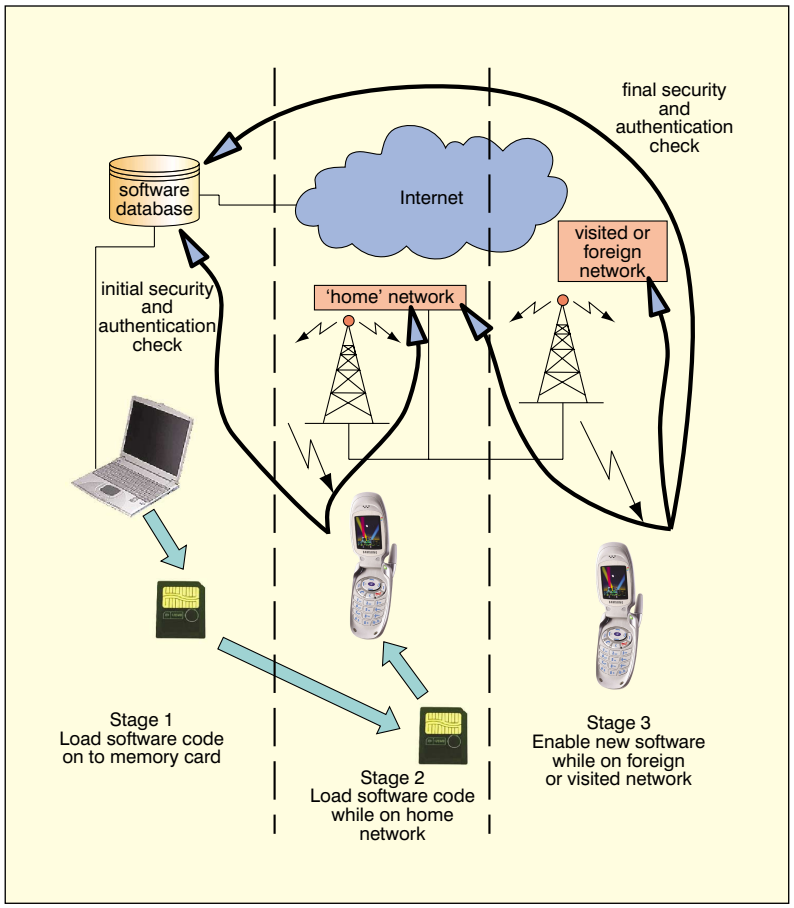

Fig. 6 An example software download procedure

or performing a Web page update, etc. the reconfiguration might not be noticed, provided it happened quickly and with minimal interruption to the data. Real-time voice or video calls may not be so forgiving; so the required software download would have to take place in the background and prepare the hardware so that some sort of 'fast' changeover could occur.

Ensuring that the software is suitable to run on the phone, so that the phone remains legal and continues to function, is a daunting task in the light of today's technology (when dealing with small, mobile and portable, battery-powered devices that people rely on for basic communications).

Sources of downloaded software

In looking at the security of downloaded code, and in particular at the impacts for SDR, one aspect that has to be addressed first is how the code gets into the phone. As with an ordinary PC, there are a number of sources for downloading code. In the future mobile phone environment these may be:

- over the terminal's primary air interface, e.g. the GSM/GPRS or UMTS air interface, perhaps utilising a protocol such as WAP (though it is appreciated that WAP has been designed to facilitate the transfer of Web-based content to limited-capability terminals, i.e. mobile phones). This is very similar to the way you would download software via the Internet to your home PC (using the Hyper Text Transfer Protocol, HTTP). The user would be expected to have been involved in the set-up of this transfer, probably by first browsing a Web site to find the required data.
- over the terminal's secondary air interface, perhaps using Bluetooth or WLAN technology. This is where things start to get interesting: if a phone has a Bluetooth interface today, then it has the ability to transfer files quickly from another Bluetooth-enabled device to, say, a laptop computer. At this point the 'downloaded' file route/path is not certain, but this could be important information to have if you are using the software to reconfigure your phone.
- from a memory device that has been inserted into or connected to the phone in some way, e.g. Smart Media, Compact Flash and Sony Memory Stick. Again, as this data could have been put on the card from anywhere the information on the data source becomes important. In the future this could be a miniature hard disk or other mass storage device.

A code downloading scenario

One possible method for downloading code to a mobile phone is described in the sequence below and illustrated in Fig. 6. This is just one of many permutations, and is probably not the preferred way, but it does show the complications that will have to be addressed in any overall security architecture, and the solutions that are eventually designed.

- Stage 1: A software upgrade is downloaded to a PC and the software copied to a memory card. The details don't matter for this example, but let's say it's some sort of performance upgrade to the air interface for use when roaming.
- Stage 2: While the phone is in use on the normal 'home' network, the memory card is inserted into the phone and the software possibly copied into the phone. Some sort of security check has to be made both with the original software database and with the home network. This check needs to show that the software is valid for the mobile phone and for the network it is currently being used on.
- Stage 3: Not until the user visits a foreign network is the new software actually executed, so the security and authentication should probably (this is a topic for research and discussion) be repeated on the visited network to make sure that the new software is secure for use. This may also mean accessing the home network to authenticate the mobile equipment, and accessing the original software database for details of the software characteristics.

All of this implies that downloading 'authorised' code may not be simple. It is more likely to involve complex combinations of security and configuration management,

and the scenarios presented here are ideal cases. What would happen if a user was downloading code while roaming and then couldn't find (for any reason) the original source server? Would the phone shut down? Customers would be unlikely to be happy about this; and the operators could lose revenue, so they would not be happy either. If the phone is to make its own decisions, how would this be implemented, and at what level should the phone's internal software security operate? How much intelligence would the phone need and how would this intelligence be implemented in a fail-safe way?

6 Conclusion

It would be hard to derive a 'solution to security' that could be deemed all-encompassing at the outset, however from the arguments and discussions in this paper sensible guidelines might be suggested, including the following:

(1) If possible, separate the application downloadable software area from the air interface area.
(2) Incorporate robust, less 'power hungry' air interface encryption techniques.
(3) Adopt a 'hand-shaking' man–machine interface (question and answer type procedure verifying that download content and source are trusted by the user).
(4) Networks should continue to adopt a policy of minimum transfer of non-encrypted personal information (for example, International Mobile Subscriber Identity and equivalents thereof should not be transmitted unnecessarily).
(5) Avoid the use of static ID where possible (for example, use dynamically assigned IP addresses for Internet sessions).
(6) Provide extremely secure minimum back-up (fixed) configuration software with good default security settings (to ensure that complete system lock-out cannot occur).
(7) Provide (updatable) virus capture software.
(8) Provide content encryption of stored data and all removable memory (this will also ensure integrity if mobile devices are lost/stolen).
(9) Use secure user authentication (PIN numbers/pass phrases etc.).
(10) Encourage security consciousness. Provide clear/simple guidelines for using terminals in a secure manner.

There is much to research in this area and some of the scenarios currently proposed look large and complex for mobile devices. Everybody relies on mobile phones, and their reliability cannot be compromised (simply for the sake of technical flexibility), especially if prices to the consumer are likely to increase. A balance will therefore have to be struck.

References

1 'Vocabulary for 3GPP specifications'. 3GPP TS 21.905 v3.3.0
2 'Mobile station application execution environment functional description. Stage 2'. 3GPP TS 23.057 v3.4.0
3 See the MExE Forum Web site at http://www.mexeforum.org and http://www.mobilemexe.com/whatis.asp?link=1
4 See http://www.qualcomm.com/brew/
5 See http://www.microsoft.com/net/
6 WALLER, A. O., *et al.*: 'Securing the delivery of digital content over the Internet', *Electron. Commun. Eng. J.*, October 2002, **14,** (5), pp.239–248
7 TUTTLEBEE, W. (Ed.): 'Software defined radio: enabling technologies' (Wiley, 2002)
8 WWRF: 'Book of visions 2001'. See http://www.wireless-world-research.org
9 HASAN, H., *et al.*: 'The design of an extended AAAC architecture'. IST Mobile & Wireless Telecommunications Summit 2002, 17th–19th June 2002, Thessaloniki, Greece, pp.36–40

Derek Babb is the Technology Manager, Advanced Technology Group at Samsung Electronics Research Institute, a division of Samsung Electronics (UK) Ltd. (see www.samsungelectronics.co.uk). He has over 20 years engineering and technology experience in telecommunication systems, mainly in mobile technology. In the past he has worked for the BBC, Multitone Electronics, Plessey Military Communications and Ericsson Mobile Platforms.

Address: Samsung Electronics Research Institute, Communications House, South Street, Staines, Middlesex, TW18 4QE, UK
E-mail: derek.babb@seri.co.uk

Craig Bishop is the Technology Manager for Systems and Standards at Samsung Electronics Research Institute, where he has worked in the areas of mobile terminal platform development and telecommunications system standardisation. He has also previously worked for the UK Radiocommunications Agency and the BBC.

E-mail: craig.bishop@seri.co.uk

Dr Terence E. Dodgson works in the Advanced Technology Group at Samsung Electronics Research Institute. He has worked in the areas of image processing, digital signal processing, telecommunications, satellite communications, RF radio and mobile phone technology in both a research and managerial capacity. He has also worked for GEC, British Aerospace, DERA, Motorola, Ericsson Mobile Platforms, and Wave Solutions of Birmingham University.

E-mail: tdodgson@seri.co.uk

First received 8th July and in revised form 4th September 2002