♦ Secure Base Stations

Peter Bosch, Alec Brusilovsky, Rae McLellan, Sape Mullender, and Paul Polakos

With the introduction of the third generation (3G) Universal Mobile Telecommunications System (UMTS) base station router (BSR) and fourth generation (4G) base stations, such as the 3rd Generation Partnership Project (3GPP) Long Term Evolution (LTE) Evolved Node B (eNB), it has become important to secure base stations from break-in attempts by adversaries. While previous generation base stations could be considered simple voice and Internet Protocol (IP) packet transceivers, newer generation cellular base stations need to perform more of the user- and signaling functions for the cellular radio access network. If adversaries can physically break into newer base stations, they can perform a range of undesirable operations such as snooping on conversations, carrying out denial-of-service attacks on the serving area, changing the software base of the base stations, stealing authentication and encryption keys, and disrupting legitimate cellular operations. The cell-site vault is a secure processing environment designed to resist such tampering and to protect the sensitive functions associated with cellular processing. It provides an execution environment where ciphering functions, key management, and associated functions can execute without leaking sensitive information. In this paper, we present the basic principles of the cell-site vault and present an overview of the types of functions that need to be protected in future base stations for cellular networks. We address the importance of providing a trust hierarchy within the cell-site vault, we present why the vault needs to be used to establish secure and authenticated communication channels—in fact, why the vault needs to be used for most external communications—and we present why it is important to execute functions such as data re-encryption inside the vault. A femtocell or home base station is particularly vulnerable to attacks since these base stations are physically accessible by adversaries. In this paper, we focus in particular on a cell-site vault design for a femto-class base station, including its standardization efforts, as it is challenging to include both secure and non-secure processing inside a single "system-on-a-chip." © 2009 Alcatel-Lucent.

Bell Labs Technical Journal 13(4), 227–244 (2009) © 2009 Alcatel-Lucent. Published by Wiley Periodicals, Inc.
Published online in Wiley InterScience (www.interscience.wiley.com) • DOI: 10.1002/bltj.20346

Panel 1. Abbreviations, Acronyms, and Terms

3G—Third generation	MME—Mobility management entity
3GPP—3rd Generation Partnership Project	MSC—Mobile switching center
4G—Fourth generation	NAS—Non-access stratum
AES—Advanced Encryption Standard	ns—Nanosecond
AKA—Authentication and key agreement	PCB—Printed circuit board
ASIC—Application-specific integrated circuit	PDCP—Packet Data Convergence Protocol
BSR—Base station router	PDU—Packet data units
CK—Ciphering key	RAM—Random access memory
DDR2—Double data rate, version 2	RLC—Radio Link Control
DRAM—Dynamic random access memory	RNC—Radio network controller
ECC—Error checking and correction	ROM—Read only memory
eNB—Evolved Node B	SA3—System Architecture group 3
EPS—Evolved packet system	SAE—System architecture evolution
ESP—Encapsulating Security Protocol	SDRAM—Synchronized dynamic random access memory
GPRS—General packet radio service	SGSN—Serving GPRS support node
H(e)NB—Home evolved Node B	SoC—System-on-a-chip
HSPA—High speed packet access	SRAM—Static random access memory
IK—Integrity key	TCB—Trusted computing base
I/O—Input/output	UICC—Universal integrated circuit card
IP—Internet Protocol	UMTS—Universal Mobile Telecommunications System
IPsec—Internet Protocol security	USIM—UMTS Subscriber Identity Module
LTE—Long Term Evolution	
MAC—Message authentication code	

Introduction

Security is an important consideration in cellular networks. Traditionally, most of the network elements that deal with security in a cellular network have been housed in the operator's core to prevent adversaries from breaking into them. Such break-ins might be attempts to eavesdrop on conversations, disrupt signaling, endanger subscribers' privacy, or engage in fraudulent activities, i.e., to circumvent billing. Base stations, even though they are usually placed in locked enclosures, are, nevertheless, not considered secure enough to assume that intrusions into them are impossible.

In Universal Mobile Telecommunications System (UMTS), a base station is considered to be just a transceiver that passes all data, which is usually encrypted, to a central radio network controller (RNC), without decrypting or interpreting the data. The RNC, which is located on the network operator's premises, is a physically secure network element where encrypted telephone conversations and other user plane traffic can be decrypted. Other components, such as a serving GPRS support node (SGSN) and mobile switching center (MSC), also located at secured premises, perform the signaling associated with setting up and tearing down calls. The RNC, SGSN, and MSC are housed in the operator's core, partly in order to reduce the risk of a break-in, although there is some evidence even this arrangement might not provide sufficient security [30].

The UMTS base station router (BSR) [19, 21] performs all UMTS-specific protocol processing in the base station. While this integration leads to serious performance improvements, as reported separately, it also implies that all functions that are traditionally kept inside a secure cellular core infrastructure are now integrated inside the seemingly insecure base station. Obviously, the BSR approach presents the risk of weakening the security architecture of the UMTS system. To make matters worse, the emerging femto base stations are, most likely, no longer under the direct control of the wireless operator. From a security perspective, the femto BSR represents the worst of all scenarios: security-sensitive functionality placed directly in the hands of potential adversaries where it cannot be supervised by the network operator, coupled with the incentive to reduce the complexity (and thus cost) of the device as much as possible.

As was confirmed by the 3rd Generation Partnership Project (3GPP*), the deployment of femto base stations built according to the principles of a BSR would run the risk of break-ins if no special measures are taken. Our challenge is to mitigate the risks by designing an architecture that allows the macro BSR and, more importantly, the femto BSR to be secured in spite of its vulnerable situation.

While the security risk is made visible for the femto BSR, we argue that as soon as operators place any form of base station in the hands of potential adversaries, they run the risk of attacks from these base stations. So, even though the security architecture presented in this paper is especially suited for the femto BSR, regular femto base stations benefit from our security solution as well. We argue that all communication to the operator requires a secure tunnel, and the only method to secure such a tunnel is by using the same techniques as the ones we use to secure the femto BSR.

The security issue first became apparent with the introduction of the UMTS BSR. However, later 3GPP deployment options, such as Evolved High Speed Packet Access (HSPA) [13] and 3GPP's System Architecture Evolution (SAE) Long Term Evolution (LTE) [14], require similar security solutions for base stations. In both 3GPP systems, the user plane processing executes completely inside the base station, including the processing that needs to remain secure. In fact, the security solution that was presented by the 3GPP's System Architecture Group 3 (SA3) to mitigate the security issues for the UMTS BSR is now also applicable to the other 3GPP deployment options [7].

The security solution in this paper is presented with securing the base station in mind. The solution is equally pertinent to a variety of different application domains as well, most notably consumer electronics.

UMTS Security Background

Before we describe how to protect BSRs from attacks, we first present the background necessary to understand the threats in more detail.

Currently, in UMTS, the security hierarchy is built around its authentication and key agreement (AKA) procedure [17]. The application—UMTS Subscriber Identity Module (USIM)—residing on the mobile universal integrated circuit card (UICC) in the mobile terminal contains a secret key which is shared with the home operator. The home operator stores the secret key in its authentication center. When the mobile terminal signs onto a UMTS network, a key exchange (the UMTS-AKA algorithm) takes place. UMTS AKA allows the mobile terminal to verify that it is communicating with the appropriate operator, and, similarly, the operator verifies that it is communicating with one of its own UICC (USIM) cards. The essence of this mutual authentication algorithm is based on each side proving to the other that it has the appropriate key by encrypting something with it.

At the end of the UMTS-AKA procedure, the network operator and the UICC card in the mobile terminal agree on two temporary keys: the ciphering key (CK) and the integrity key (IK). The UICC card transfers its keys to the mobile terminal.

The CK is used to encrypt data packets that are exchanged between the layer-2 protocol stacks in the network and the mobile terminal. The IK is used to verify that encrypted packets indeed were sent by one of the key holders—by way of a message authentication code (MAC). It is clearly important that the CK and IK are kept secret to avoid eavesdropping and call hijacking.

In BSR-based base stations, the network operator ensures the CK and IK are available to the base station because it is the BSR application that must process signaling traffic and transfer voice and data packets between the wireless link and the "backhaul" Internet Protocol (IP) network. The wireless link and the backhaul link, unfortunately, use different encryption standards, so all data must be decrypted and re-encrypted by the BSR applications. The BSR applications, therefore, need to have the CK and IK (as well as keys for encrypting backhaul data) and they also "see" the unencrypted voice or packet data.

Attack Model

When a BSR boots up, the groundwork is laid for the BSR to provide secure and dependable service.

The BSR must verify the authenticity of the network service provider and be able to authenticate itself to it. It must—through the way in which it is constructed—convince the network service provider that it is a genuine, tamper-resistant BSR running the correct software. And, when the BSR starts providing service, it must do so correctly and securely, while protecting the privacy of the calls flowing through it.

The biggest challenge may be to convince the network service provider (operator) that the BSR is the genuine thing, with appropriate software running on it. We assume that some of these BSRs are in people's homes (femto BSR), and that at least a few of the people in physical possession of such BSRs will attempt to foil their security.

It is important that we make assumptions about the kinds of attacks the BSR must be protected against. These assumptions can range from simple attacks on the BSR via the radio channel or the backhaul network (e.g., Ethernet), to very sophisticated (and expensive) attacks involving drilling into the chips, examining the chip internal state under an electron microscope, and other such intrusions.

We demand that BSRs are constructed in such a way that a sophisticated attack on one of them, which reveals all the secrets on that BSR, does not yield the means to break into others without physically damaging those as well. The profits from such an attack, therefore, are limited to what service can be stolen and what conversations can be eavesdropped from the compromised BSR only. If the attack is expensive enough, it is not worth pursuing.

BSRs in people's homes are only allowed to serve residents, and it is therefore up to homeowners to protect home BSRs against physical attacks. BSRs serving the general public and enterprises need additional physical protection, in the form of locked cabinets, and other security measures to resist physical tampering. Our assumption, therefore, is that a very sophisticated and almost certainly destructive attack on a BSR yields a financial reward that is a small fraction of the cost of the attack, and is thus not an appealing one.

Our concern is to protect against non-destructive attacks: pure software attacks, attempts to read flash memory, or attempts to use logic analyzers to monitor the input/output (I/O) from the chips on board (e.g., traffic between the processor and memory).

Our defense against attacks on the BSR software is:

1. To protect the master keys used for loading the operating system, verifying the authenticity of the network service provider and authenticating the BSR to it, and for establishing session keys in such a way that they are only accessible during the bootstrap phase.
2. To concentrate all operations that require the manipulation of secure data (i.e., keys and unencrypted private data) in a very small "trusted computing base" (TCB) [33], which can only be entered in a controlled manner (described in the section titled "Secure Bootstrap").
3. To never allow keys or unencrypted private data to leave the central processing chip. This includes disallowing keys or unencrypted data to be stored in off-chip random access memory (RAM).

Cell Site Vault

To make sure that no adversary can obtain access to the keys inside the BSR, we designed a cost-effective secure processing environment for the base station to mitigate the risk of such break-ins. Since we focused on the femto base station, our solution had to be a low-cost one. The low-cost constraint implied that no special hardware could be used to protect the secrets. We have termed our solution the "BSR cell-site vault" [5, 6].

The vault is a small TCB in which all of the security-related functions that the BSR must conduct are concentrated. The vault is protected from the rest of the system through mechanisms that are similar to those used to protect an operating-system kernel from user-level program attacks: the vault can only be entered through a well-defined interface that is carefully controlled by the vault itself, just like an operating system kernel protects itself from attack by user programs by restricting kernel entry to system calls and traps only.

Figure 1 shows the environment in which the cell-site vault operates. A base station is shown with a secure (gray) portion and an unsecured (dark brown) portion. The secure portion executes the software that manages credentials that must be kept

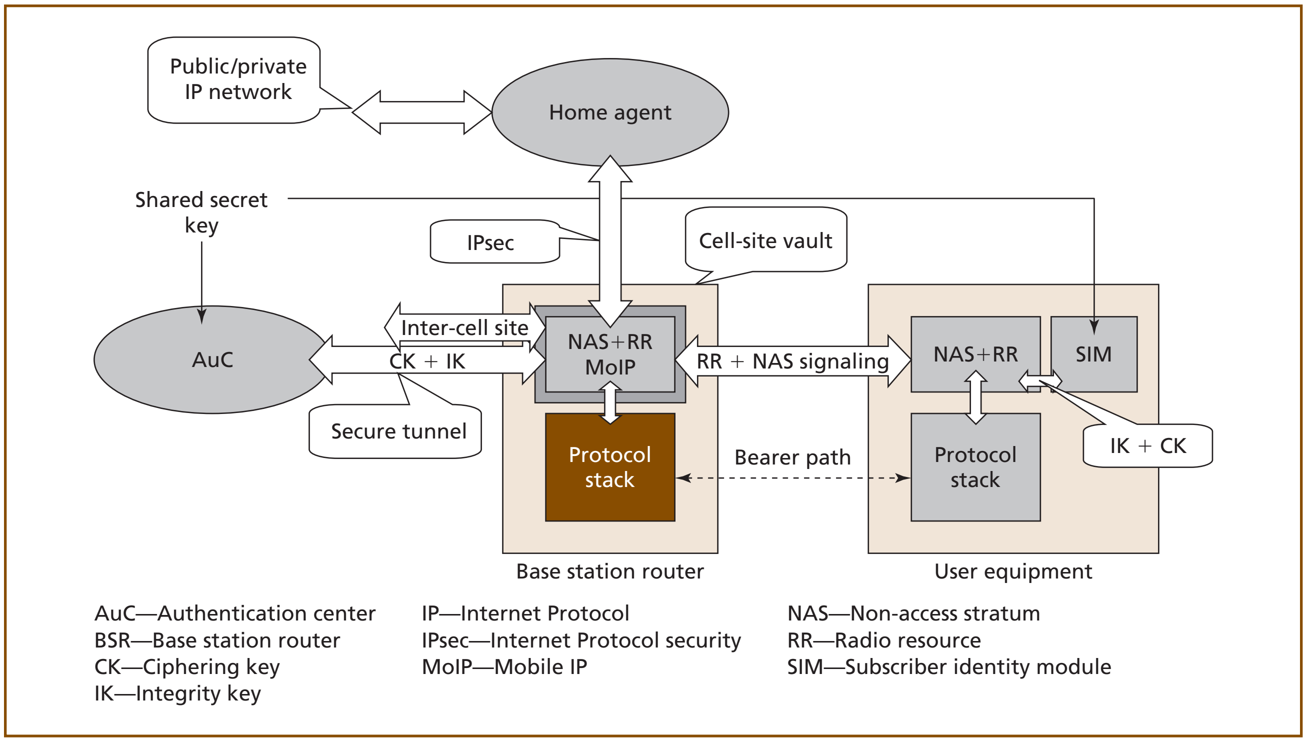


Figure 1.
A BSR base station in its environment.

secret, while the unsecured portion does everything else (such as regular protocol processing). As shown, all secure communication to external sources goes through the cell-site vault (white arrows): there are encrypted tunnels to the authentication center, to the layer-3 mobility anchor, and to the terminal. The keys are stored—and encryption/decryption is carried out—in the cell-site vault.

Figure 2 shows the secure processing environment. The security boundary is inside the "system-on-a-chip" (SoC). Typically, an SoC is implemented by an application-specific integrated circuit (ASIC). We assume it is hard to break into the SoC and to obtain the secrets within it. The mandatory components that are stored inside the cell-site vault are: a private master key, a secure boot function, a secure hypervisor [31], secure peripherals, and secure memory [20]. With these five components, we can build a processing environment for the BSR base station that can run security software so that no adversary can access the security credentials that are kept in the SoC.

To minimize cost, we chose a single processor inside the SoC that supports a secure hypervisor. The goal is to execute security software in the secure state of the processor, and execute regular base station software in the regular state of the processor. **Figure 3** shows the various states of the processor: user and kernel space, secure user and kernel space. Hence, UMTS security applications execute in the secure space, while the regular base station applications execute in the regular processor space. In addition, the interface between the two components goes through the regular operating system to a run-time system that executes inside the secure portion of the processor. This run-time system verifies all communication that goes through the interface before the requests are dispatched to secure space.

Secure Bootstrap

The essential element for bootstrapping a BSR (or any processing system with similar security requirements) is at least one key, within the SoC that needs to be used to securely bootstrap the security hierarchy. If the key is stored or generated off-processor, there is no way it can be provided to the processor without being observable by an attacker—one of the central assumptions is that an attacker can use a logic analyzer to

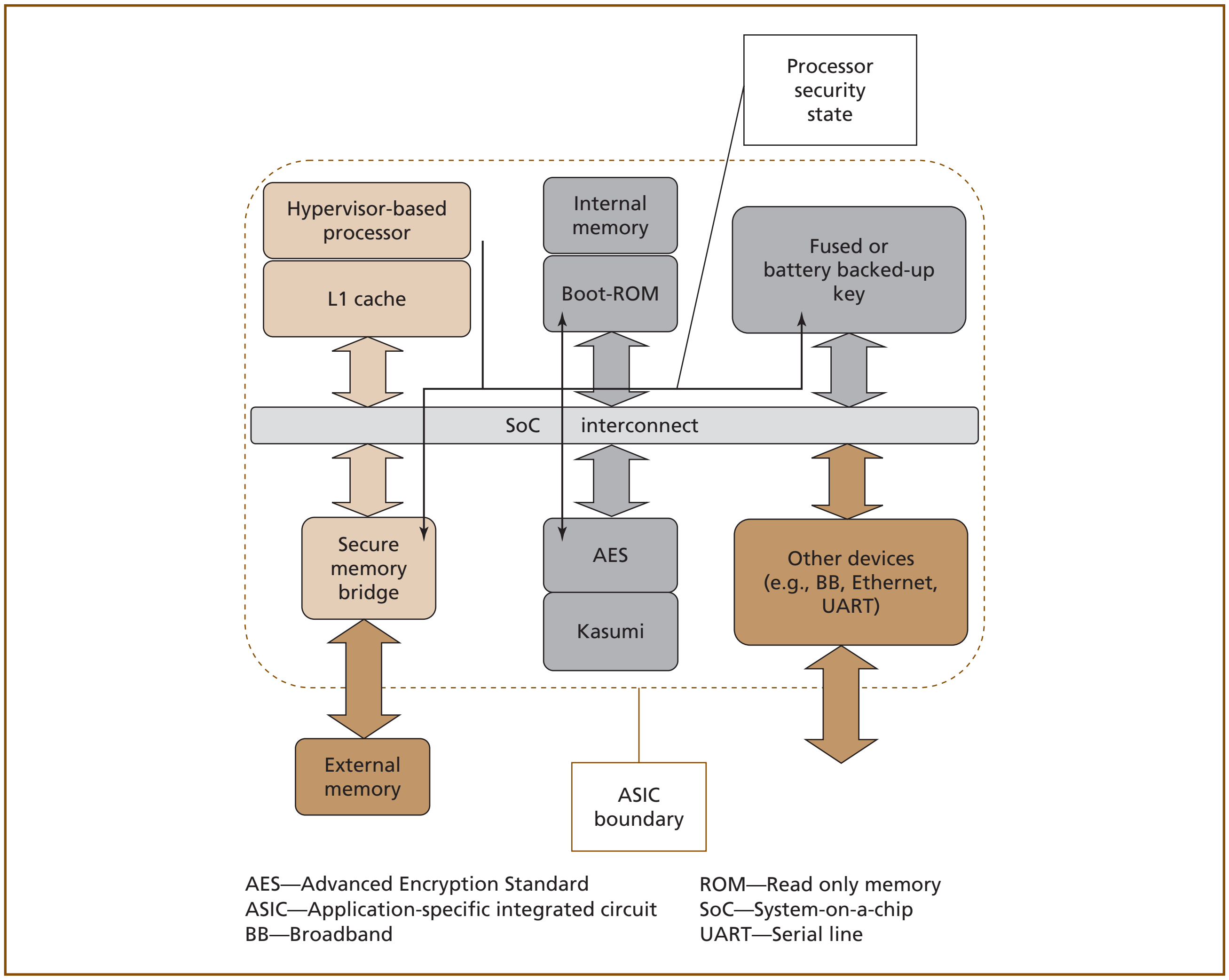


Figure 2.
A cell-site vault in a system-on-a-chip.

observe all I/O for each of the chips on the printed circuit board (PCB). Any security solution, therefore, that uses a UICC card to protect the BSR (as the only container of a unique secret key) cannot work, given the threat assumptions.

The master key or master keys must perform two essential functions:

1. Verify the system software loaded into the processor before executing any of its instructions, and
2. Ensure mutual authentication between the service provider and BSR base station, and the establishment of a session key.

Both essential functions can be performed with one master key or two. The software verification key, if public-key encryption is used, does not have to be secret, but the matching key with which the software is encrypted or signed does have to be secret. The second key must be secret. It can be the private key in a public-key pair, or a symmetric-encryption key shared with the network service provider.

Note that although the software verification key may be public and does not then have to be secret, it must be stored on-chip. If it is off-chip, it is liable to be modified by an attacker when it is loaded onto the chip; the system may then verify code as legal and proper when it is not.

The mutual authentication key can be a symmetric encryption key shared with the network service

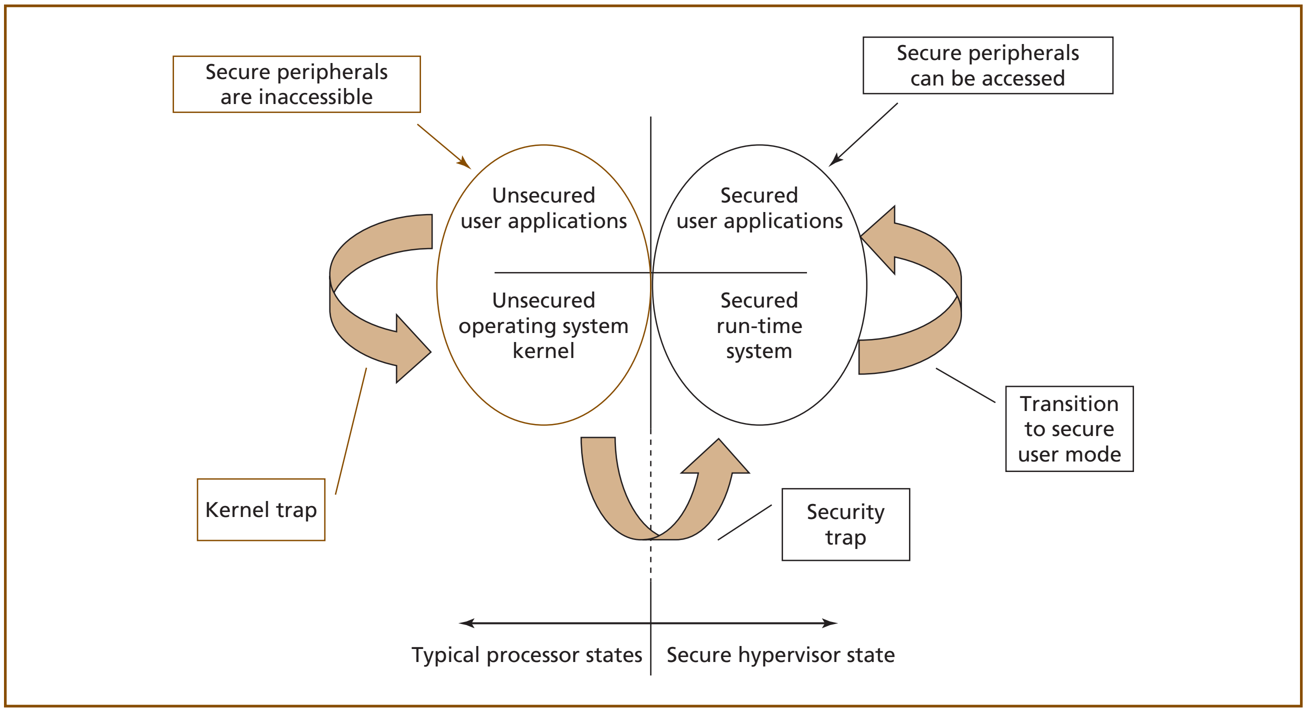

Secure peripherals are inaccessible
Secure peripherals can be accessed
Unsecured user applications
Secured user applications
Unsecured operating system kernel
Secured run-time system
Transition to secure user mode
Kernel trap
Security trap
Typical processor states
Secure hypervisor state

Figure 3.
Secured and unsecured processing on the same processor.

provider, or it can be a secret key that the chip uses to authenticate to the network plus a (not necessarily secret) public key that the chip uses to verify the identity of the network service provider.

The secret key embodies the fundamental identity of the BSR. Under no circumstances should it become public. Protecting it requires two things:

1. Making sure that the TCB that uses it is as small and secure as possible, and
2. Using it as little as possible so cryptanalysis has as little material to break it as possible.

One popular and perfectly adequate method to store the keys in a processing chip is to use "fuse memory." This is on-chip memory that, when the chip is manufactured, contains all zeroes (or ones). When the chip is inaugurated by the service provider, the fuses can be individually flipped, essentially by burning the fuses. One special fuse allows the burning to be carried out and when the key has been entered, it too is burned, making further modifications impossible. There are other techniques, too. For example, using a tiny amount of battery backed-up on-chip RAM to contain the key; when the system senses an attack, or when the chip is disconnected from the battery, the keys are automatically erased.

We require two additional features for secure bootstrap. One is a small amount of on-chip read only memory (ROM) that contains the primary bootstrap code. The other is a switch that can be thrown by software to cut off access to the on-chip key memory. The switch only returns to the access-allowed position by a hard reset of the chip, and this will also initialize the program counter to start executing the bootstrap code in ROM.

The idea for such a switch is not new. Butler Lampson mentions this switch in his article on authentication in [27]. The switch must be considered essential if it can be assumed possible that an attacker might compromise the operating system—a sensible assumption.

With these ingredients, secure bootstrap proceeds in the following steps:

1. The processor is reset (either by powering it on or by pressing the reset button); the processor is brought to a known state in which any previous settings for memory management, interrupts, peripheral device programming, and code running is rendered innocuous. The switch providing access to key memory is set to allow access and the processor starts executing the program stored in on-chip secure ROM.
2. The primary loader program in ROM loads a secondary boot loader program from off-chip flash memory into on-chip RAM. Most modern processors allow the on-chip cache to be used as RAM during the boot phase. Systems-on-a-chip often have on-chip static random access memory (SRAM) that is used for other purposes during normal operation. Any of these methods suffice, provided the memory is on-chip so that it can neither be monitored nor modified by an adversary. The loader must check that loaded code does not overwrite any of the loader's state as this would allow an attack. It must also calculate a secure hash (also known as a digest or checksum) of the loaded code and verify it using a digitally signed copy of the hash that must accompany the secondary boot loader in the flash. If the signed hash does not match, the primary boot loader halts the processor: the flash has been damaged or tampered with.
3. When the secondary boot code has been successfully loaded and verified, it can be run (we repeat here that it must be loaded into and run from on-chip memory). Whereas the primary boot loader in ROM may be only chip-specific (the only off-chip access is to flash memory which can be configured to be in a well-known location), the secondary boot loader is system-specific. It can use main memory, and devices such as the Ethernet. The first task of the secondary boot loader is to perform authentication to the network service provider. So, it typically configures the Ethernet, obtains an IP address (if not built in), and establishes two-way communication with the network service provider.
4. Using the on-chip authentication keys, the secondary boot loader performs authentication and, assuming it succeeds, establishes a session key that the BSR can use until the next reboot. The secondary boot loader then throws the switch, denying any future access to the master keys until the next hard reset of the processor.
5. The secondary boot loader may now load the operating system—from flash, from disk, or downloaded from the network service provider. The operating system must be signed and verified, just like the secondary boot loader, but possibly with a different key which may have been supplied as part of the authentication process with the network service provider.
6. Finally, the operating system is allowed to run and it is provided with access to the session key, so it can communicate with the network service provider.

When the operating system is securely activated, the BSR applications can be started in the secure and un-secure portions of the processor.

Secure Memory

As described earlier, the vault uses a secure hypervisor [28, 31] or ARM Trustzone* [18] processing environment to partition the application space into an area that is protected from adversaries and an environment for executing "regular" non-secret base station functions. To protect the secure area, it is also important to protect the memory in which the secure applications operate: the code that is executed, the keys on which that code operates and the data buffers that are re-encrypted all need to be secret. We foresee two solutions:

1. Include memory space within the vault, i.e., within the SoC. Note, this appears to be an expensive proposition.
2. Use external off-chip memory, but use memory encryption when the processor is operating in secure mode. This section describes memory encryption in detail.

The ARM Trustzone architecture broadcasts the execution security level throughout the system (i.e., the SoC) as a modifier on the address bus. All load/store operations of the processor to peripherals or memory carries with it the processor's current security level. We use these security state bits to drive an encryption engine within the memory controller to encrypt or decrypt regions of memory that need to remain private. This idea is not new and has been published before [20]. Our novelty here is the application to secure a low-cost femto base station.

In what follows, we have connected an encryption engine in series with the memory controller. Whenever the memory controller needs to write data to off-chip memory, it encrypts and "integrity protects" that data, and when data is read back, it decrypts and "integrity checks" that data. In case a decryption fails, the memory controller flags the operation as failed and signals the processor of the event. The processor may halt or reset on such events. The memory controller's encryption engine itself is part of the cell-site vault: it is part of the SoC that also includes the main processor and cell-site vault key material. We assume it is not cost-effective for an intruder to break into the SoC, and thus also into the memory controller.

Cryptographically secure messages are usually signed by a message authentication code, which is a cryptographic hash [29] of the message. Suppose a 256-bit message is encrypted, resulting in a 256-bit scrambled version of the message. The 256-bit space of possible clear text messages is mapped by the encrypting function to a 256-bit space of encrypted messages. The mapping is one-to-one. Every 256-bit encrypted message maps (via decryption) to a possible 256-bit clear text message. So if an adversary replaces an encrypted message with a different one, there's no way to detect tampering. The decryption system would simply convert that illicit 256-bit encrypted data (via decryption) to an incorrect clear text message and the system would accept it as valid (but incorrect) data. Including a MAC in the encryption system provides a means to detect such tampering. Not only must the adversary change the encrypted data, but they must also create a corresponding MAC that is the cryptographic hash of the new data. MACs are necessary to detect tampering. The overhead of including a MAC in each encrypted memory transaction is a necessary overhead in a secure memory system.

Most modern processors have integrated instruction and data caches that constrain the types of memory transactions a memory system sees. Instead of the memory traffic produced by a processor's bare load/store instructions, the memory system sees cache miss and cache flush traffic. Such cache systems transfer one cache line at a time instead of individual words or bytes. These larger blocks, typically 32-bytes, can be more efficiently encrypted and decrypted. A good encrypting memory system should be most efficient in these cache line sized transactions, but also support single byte transactions as well.

Another point to realize in processor/memory performance is that processors need not wait for memory write operations to complete and can execute ahead, as long as the data has been transferred to the memory system and is queued for writing sometime in the future. The data need not actually reside within the memory before the processor proceeds in executing subsequent instructions. Just the promise that it will be written eventually is all that is necessary. In contrast, processors often cannot usually perform any useful operations while they wait for data to return from a memory read operation. They are stuck waiting, and memory read latency directly impacts overall system performance. This asymmetry in read/write latency requirements is of paramount concern in designing an efficient encrypted memory system.

Today's dynamic random access memory (DRAM) technology is inherently serial in nature. They are no longer just random access memories, but also provide access latency advantages for nearby sequential locations. In fact, double data rate, version 2 (DDR2) synchronized dynamic random access memory (SDRAM) chips do not transfer single words [26], but rather bursts of two or four sequential words starting from an initial address. Any efficient memory system must take advantage of the improved access times provided by such multi-word transfers.

Dynamic memory circuits operate as memory elements by "remembering" small electrical charges on tiny capacitors. Whether that tiny charge is detectable or not determines if the memory location contains a one or a zero. Moreover, the "dynamic" aspect of DRAM comes from the need to refresh the charges periodically, otherwise the circuit forgets and the DRAM becomes unreliable. Unfortunately, random events in the environment such as radioactive decay or even cosmic rays, can upset that charge storage and disrupt the memory function completely. These events are called single event upsets [24], because they are unpredictable and statistical in nature and usually only affect a single memory location.

While single event upsets may be rare, they unfortunately do occur. So any encrypted memory system that depends on MAC matching for tamper detection must also be resilient under single event upsets. The technology to correct single bit errors involves storing redundant memory bits for each word in memory and is called "error checking and correction" (ECC) [23]. For a 32-bit wide memory system, ECC adds an additional seven bits. Each 32-bit memory word becomes a 39-bit (or 40-bit, with one bit unused) wide memory system to provide the extra seven ECC bits. Well-known parity circuitry generates the seven extra ECC bits on memory writes, and the extra seven ECC bits are used on memory reads to check and correct single bit errors in the 32-bit word. An encrypted memory system must incorporate ECC in order to distinguish random bit errors from malicious attacks.

The components of an encrypted memory system are shown in the block diagram of **Figure 4**. A dual-ported write buffer is positioned between the bus interface module and the encryption system. The write buffer serves as a queue of pending write operations to the encrypted memory system and allows bus write-transactions to complete early, before data is actually written into off-chip memory. The write buffer also supports sub-encryption-block sized data transactions. Since the encryption process requires entire 256-bit encryption blocks, bus transactions of single bytes and words are captured in the write buffer and then "backfilled" by reading the remaining bytes from off-chip encrypted memory. Only when the entire 256-bit write buffer entry is valid can it be encrypted and retired to memory. Bus read-transactions "hit" the write buffer return immediately, while those that "miss" initiate an encrypted memory transaction and only complete after the memory access latency and decryption overhead. The write buffer increases

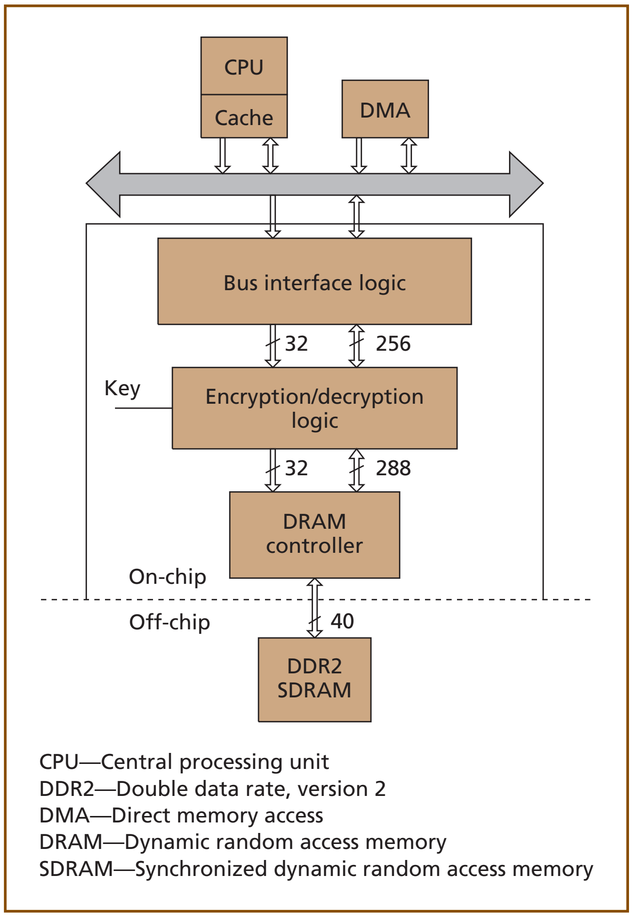

CPU
Cache
DMA
Bus interface logic
32
256
Key
Encryption/decryption logic
32
288
DRAM controller
On-chip
Off-chip
40
DDR2 SDRAM
CPU—Central processing unit
DDR2—Double data rate, version 2
DMA—Direct memory access
DRAM—Dynamic random access memory
SDRAM—Synchronized dynamic random access memory

Figure 4.
Encrypted memory block diagram.

bus write performance, decouples the bus interface logic from state machines that control the encryption/decryption process, and simplifies the design effort. With this architecture, changing the bus interface would have minimal impact on the rest of the encrypted memory system.

The logical flow of the encryption/decryption process is shown in **Figure 5**. The encryption algorithm is Rijndael [22], the same chosen for the Advanced Encryption Standard (AES). It is not exactly AES, because AES is defined to operate only on fixed 128-bit blocks. Larger encryption tasks using AES are composed of chaining multiple 128-bit encryption operations together in a serial manner, which would unreasonably extend the latency of the encrypted memory system. But Rijndael is extensible to operate on larger block sizes and still retain its cryptographic properties. There are two properties that make Rijndael

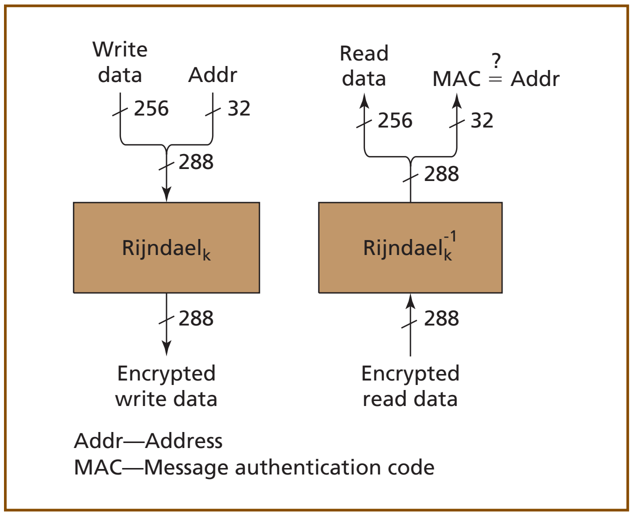


Figure 5.
Encryption/decryption data flow.

a good cryptographic function, as detailed in **Panel 2**. Instead of the 128-bit blocks used in AES, Rijndael is used to directly encrypt a 288-bit block composed of 256-bits of data and 32-bits of address. The resultant 288-bits of encrypted output effectively encodes both the data as well as a 32-bit message authentication check value. Upon decryption, using an inverse Rijndael function, if the address appears in the proper 32-bit locations then the authentication of the message has been effectively checked and the 256-bits of decrypted data is assumed to be correct. Using Rijndael to produce a MAC in this way, spread across the entire 288-bit encrypted output, is more effective than having a fixed 32-bit MAC field, because an adversary can no longer attack the encryption by exhaustively searching all four billion possible MAC values.

As mentioned earlier, adding MAC bits to each encryption block is an essential overhead of an encrypted memory system. Earlier encrypted memory systems [32] have chosen to collect all the MAC fields in a separate arena and recursively generate MAC fields for them as well. This "tree" of MAC values provides a cryptographic hash of the entire memory space. While this is no doubt more cryptographically secure, the overhead of updating the tree of MACs for each memory transaction has significant performance impact. A 128 MB memory system, with MACs generated for each 256-bits of storage, would require nine MAC updates for each memory transaction. While these updates can be cached and retained on-chip, they do represent significant chip resources and time.

Instead of building a tree of MAC values, we use a single flat MAC for each 256-bit encryption block. Each 8 words (256-bits) of address space is then expanded to address 9 words (288-bits) of external memory as shown in **Figure 6**. This expansion can be readily implemented by a single 32-bit adder applied to the address. It does, however, mean that only eight-ninths of the physical memory is available for encrypted memory use. The remaining one ninth, though spread out in the memory space, effectively holds the MACs. By placing the MACs inline with the encrypted data as a ninth word, the system takes advantage of memory's inherently faster sequential access and provides the entire 288-bit encryption block and MAC with minimal latency.

In order to protect from replay (where an adversary replaces the current encrypted memory contents with a previously stored value) and *k*-wise attacks (where the adversary takes advantage of seeing the

Panel 2. Rijndael cryptographic properties

1. Rijndael is good one-way function. With Rijndael it is straightforward to produce an encrypted result given a key and input data. But it is difficult to invert that process and determine the key from the encrypted data and unencrypted data. i.e., $R(k,d) = e$, is easy to compute while $R^{-1}(d,e) = k$ is very difficult to compute and therefore it's hard to guess the key.
2. Rijndael is a good pseudo-random function, i.e., there is a good probability that every bit of the output is dependent on each bit of the input. Rijndael spreads the effects of changing a single bit of input to all bits of the output. (And why error checking and correction [ECC] is mandatory. A single bit error in the 256-bits of encrypted data would propagate to many bit errors in the decrypted data.)

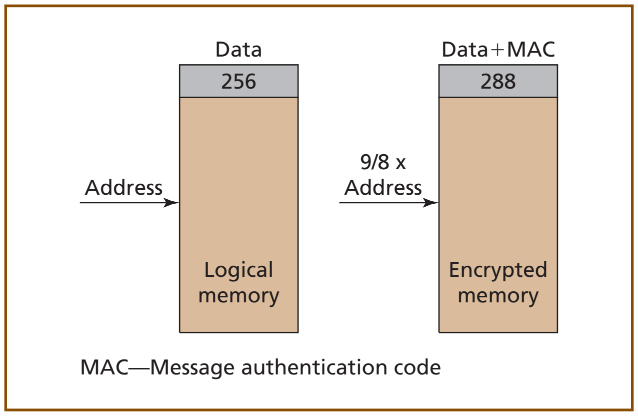


Figure 6.
Expansion of encrypted memory space to include MAC.

same key and address being used for encrypting *k* different data values) the encrypted memory system has the ability to mutate the encryption key. This is similar to normal cryptographic communications where a key goes stale after *k*-uses and must be discarded in favor of a new key.

The memory space is partitioned in two regions and the system can use two different cryptographic keys, one for each region. By maintaining an address register that defines the boundary between those regions, a simple address comparison can be performed to identify which region the memory transaction refers to, and which key to use. Then, to change from one key to another, a mutation process can be run to grow one region at the expense of the other (incrementing the boundary register) and the data can be read, decrypted with the first key, and encrypted and written back to DRAM with the new key. Once the boundary register reaches the end of memory, the original key is no longer in use and can be safely discarded. A new key is created and the process repeats with the boundary register pointing back to the beginning of memory. In this way, keys can be regularly changed and prevented from going stale, thus maintaining the integrity of the cryptographic memory system.

There is no doubt an encrypted memory system has an impact on performance. However, since the write buffer allows writes to proceed offline, there is little impact expected on write performance unless the application is write-bound. But for read transactions, the decryption latency is serially added to the memory-read latency. In the prototype system based on a Xilinx* Virtex-4* device, the average memory latency and bus interface overhead observed was about 180 nanoseconds (ns) with DDR2, while the 288-bit Rijndael decryption process takes 70 ns, or a 30 percent latency overhead. This is not too onerous a price to pay for secure off-chip storage, especially as the processor performance is shielded with two levels of on-chip caching prior to the encrypted DRAM system. We continue to investigate more efficient methods for encrypted off-chip memory.

A UMTS Application on the Cell-Site Vault

This paper is about securing a femto BSR with the UMTS application in mind. We have described various components to secure the processor and to securely boot the processor, as well as the execution model of the processor and a method to secure the memory. Here we combine the various components and describe how to partition the UMTS BSR application such that secure elements of the BSR UMTS application operate from the unsecured components.

In this section, we assume a vault can be securely booted and that applications can be started in the secure and insecure domain. Additionally, we assume communication paths can be set up between the unsecured applications and secure applications.

Figure 7 presents a possible break-up of the BSR application on the cell-site vault. The figure shows the following components that need to execute in the processing environment:

1. *Node B application.* The Node B application is an application that executes in the unsecured domain and accepts encrypted packet data units (PDUs) from the higher-level user protocol stack and sends those PDUs into the layer-1 hardware. Alternatively, it takes packets that have been received by layer-1 and hands those packets to the higher level user protocol stack. An HSDPA scheduler would execute in this space. None of this needs to run in the cell-site vault.
2. *Layer-2 user plane protocol processing.* Some parts of the MAC, the complete Radio Link Control (RLC) and Packet Data Convergence Protocol (PDCP) layer [1–3] are part of the user plane protocol processing. These parts need to run in the secure

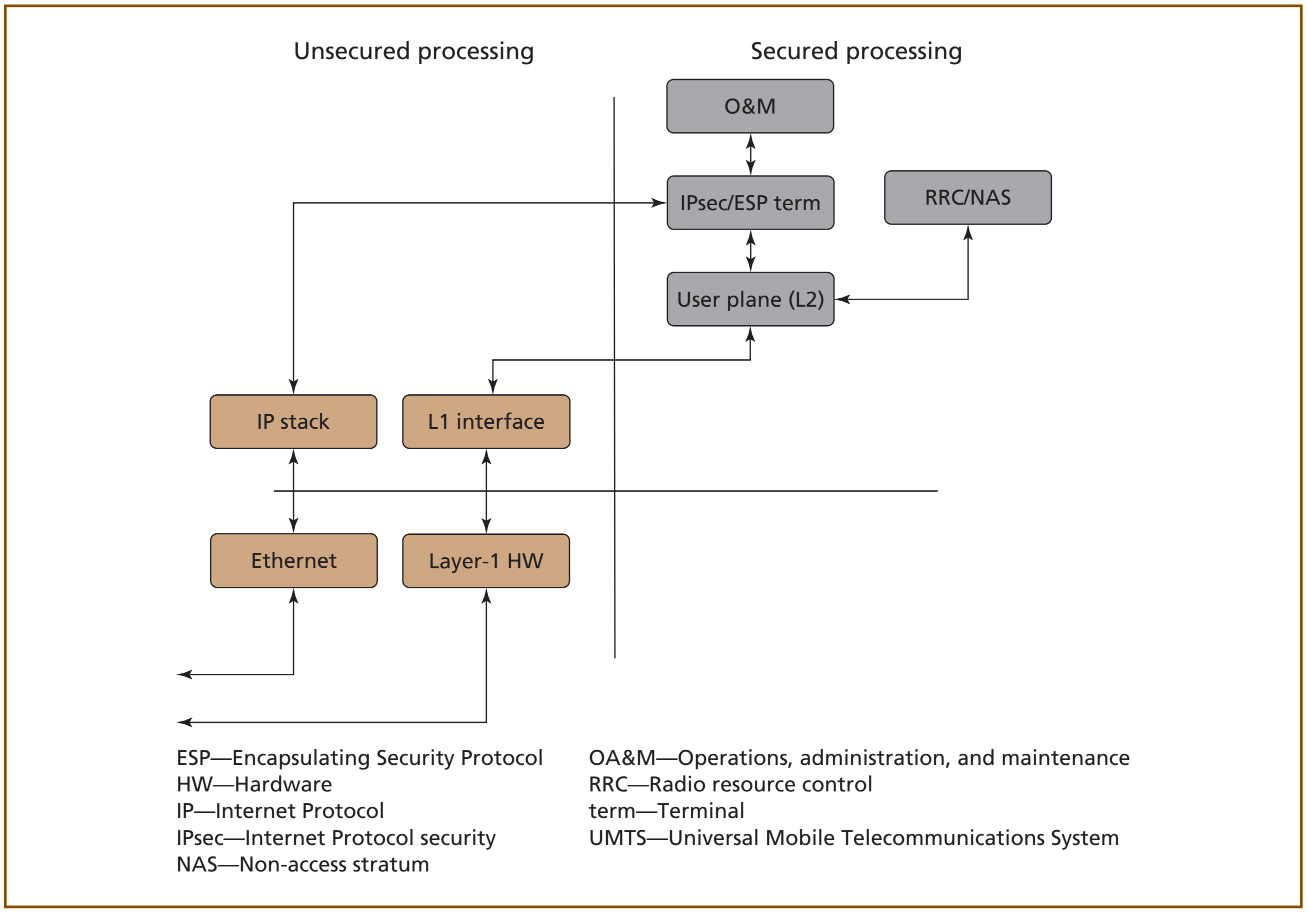

Unsecured processing
Secured processing
O&M
IPsec/ESP term
RRC/NAS
User plane (L2)
IP stack
L1 interface
Ethernet
Layer-1 HW
ESP—Encapsulating Security Protocol
HW—Hardware
IP—Internet Protocol
IPsec—Internet Protocol security
NAS—Non-access stratum
OA&M—Operations, administration, and maintenance
RRC—Radio resource control
term—Terminal
UMTS—Universal Mobile Telecommunications System

Figure 7.
A software architecture for a UMTS application.

domain since Kasumi ciphering functions [12] are part of the RLC and MAC layers, and the ciphering functions and RLC/MAC are heavily intertwined. The ciphering functions encrypt the unencrypted RLC/MAC PDUs that travel towards the mobile terminal and decrypt the PDUs that arrive from the mobile terminal. Since the PDCP layer performs header compression, it needs to be able to "see" the unencrypted IP packets and thus needs to operate in the secure domain. The per-user ciphering key CK and the associated encryption state needs to be kept in the vault to prevent adversaries from obtaining access to those keys and thus prevent them from eavesdropping or call hijacking.

3. *Layer-3 user plane termination.* This part needs to run in the secure domain. The application terminates IPsec/ESP tunnels [25], decrypts downlink packets, and prepares those packets for the layer-2 user plane processing. In the uplink, this application receives packets from the layer-2 user plane in their unencrypted form and transmits the packets through the Internet Protocol security/Encapsulating Security Protocol (IPsec/ESP) tunnel to the operator. The keys associated with the IPsec/ESP tunnel need to be kept inside the vault to prevent adversaries from obtaining the session keys which are used to communicate with the operator.
4. *Layer-3 signaling plane.* This part needs to run in the secure domain since it needs to authenticate radio-resource control packets with the integrity key [4] and since it terminates the SGSN non-access stratum (NAS) signaling [11, 15]. It is important to protect the integrity key to prevent an adversary

from spoofing the radio resource messages, and it is important to protect the ability of NAS signaling to hide call details such as telephone numbers and IP addresses that are assigned to the mobile.

For LTE, 3GPP has determined that NAS signaling is performed by a central mobility management entity (MME) while the radio resource function and user plane encryption are handled at the base station [14]. By keeping the NAS handling in the core, the security issues associated with NAS handling are less stringent when compared to executing those functions inside the base station. However, given that the LTE base station performs ciphering and integrity protection functions and terminates a secure tunnel between the MME and the base station, we argue that the LTE base station also requires a cell-site vault.

We do not worry much about other security functions such as lawful intercept and charging. We argue that those functions should not be part of a BSR and really need to reside inside the operator core.

Security and 3GPP

In 2006, we set out to design the cell site vault with the components described earlier in this paper. The reason to design the cell-site vault was to convince 3GPP that we can secure a base station in a cost-effective manner and thus enable UMTS deployments with security functions executing in the base station (read: BSR). 3GPP System Architecture Group 3 is responsible for analyzing and proposing security solutions for the 3GPP group of cellular systems. SA3 has analyzed 3GPP UMTS solutions before, though the focus is currently very much on 3GPP LTE and UMTS femtos.

In 2006, SA3 established that RNC security functions can execute in the base stations [7, 8]. This decision enabled the deployments of combined RNC/Node Bs for the so-called "Evolved HSPA" profile. The SA3 decision was in part based on presentations of a conceptual cell-site vault. The cell-site vault as described in this paper presents an instantiation of the cell-site vault that was presented to SA3. Further, in [9] and [10], SA3 reiterated its earlier decision on "Evolved HSPA" and agreed that security functions for 3GPP LTE can also execute inside the LTE base station. In addition, SA3 presented arguments for securing the backhaul through a cell-site vault [14].

Currently 3GPP SA3 is addressing the security architecture for the home Node B (3G home base station) as well as for the home LTE base station (4G home base station), and we collectively call these network elements the home evolved Node Bs (H(e)NBs). Both solutions connect into an evolved packet system (EPS) operated by the cellular operator. Security is a critical aspect of H(e)NB, and 3GPP SA3 is currently standardizing the security architecture [16] for such network elements. Given the earlier decisions by 3GPP as described above, it is very likely that the study item for the H(e)NB closely follows the results for Evolved HSPA and the security architecture for 3GPP LTE.

Reflections

In this paper, we have highlighted the development of a cell-site vault: a secure processing environment that was originally designed for a UMTS base station router to address the security concerns raised by 3GPP SA3. This cell-site vault provides an environment which can be securely booted to set up secure connections between the BSR and the operator, to securely re-encrypt data between the operator's core and the wireless link and to execute the signaling functions for a UMTS application. The solution provides a single processor with multiple security domains where regular processing can be separated from the security-related processing. In addition, main memory is encrypted to protect against eavesdropping through memory probes.

We argue that the cell-site vault is strong enough to address most of the security concerns raised by 3GPP SA3. In fact, in part based on our presentations, 3GPP SA3 decided to facilitate the execution of security functions in the cell-site vault for Evolved HSPA, LTE and now for home base stations.

While the cell-site vault has been used for a base station application here, there are many more applications of the cell-site vault. The vault can be used to protect phones (to prevent the use of other operator's USIM cards), and it can be used in set-top boxes and game consoles. In fact, we argue that the vault can be used in any consumer electronics device that needs to protect information such as keys, software, or media.

Acknowledgements

A great number of people contributed to this work. We would like to acknowledge help from Juan Garay, Mark Kraml, Jim Seymour, Vladimir Kolesnikov, and Ganesh Sundaram.

***Trademarks**

3GPP is a trademark of the European Telecommunications Standards Institute.
Trustzone is a trademark of ARM Limited.
Virtex-4 and Xilinx are trademarks of Xilinx, Inc.

References

[1] 3rd Generation Partnership Project, "Medium Access Control (MAC) Protocol Specification," 3GPP TS 25.321, 2003, <http://www.3gpp.org/ftp/Specs/html-info/25321.htm>.

[2] 3rd Generation Partnership Project, "Radio Link Control (RLC) Protocol Specification (Release 5)," 3GPP TS 25.322, Rel. 5, 2003, <http://www.3gpp.org/ftp/Specs/html-info/25322.htm>.

[3] 3rd Generation Partnership Project, "Packet Data Convergence Protocol (PDCP) Specification," 3GPP TS 25.323, 2003, <http://www.3gpp.org/ftp/Specs/html-info/25323.htm>.

[4] 3rd Generation Partnership Project, "Radio Resource Control (RRC) Protocol Specification," 3GPP TS 25.331, 2003, <http://www.3gpp.org/ftp/Specs/html-info/25331.htm>.

[5] 3rd Generation Partnership Project, "SAE/LTE: On the Termination Point for Security Associations," 3GPP TSG SA WG3 Security Meeting SA3#44, S3-060436, Lucent Technologies, July 2006, <http://www.3gpp.org/FTP/tsg_sa/WG3_Security/TSGS3_44_Tallinn/Docs>.

[6] 3rd Generation Partnership Project, "Collocating the eNodeB and MME/UPE," 3GPP TSG SA WG3 Security Meeting SA3#44, S3-060437, Lucent Technologies, July 2006, <http://www.3gpp.org/FTP/tsg_sa/WG3_Security/TSGS3_44_Tallinn/Docs/>.

[7] 3rd Generation Partnership Project, "HSPA Evolution Option Security," 3GPP TSG SA WG3 Meeting #45, S3-060654, Nokia, Lucent Technologies, T-Mobile, Vodafone, Alcatel, China Mobile, Telecom Italia, Nov. 2006, <http://www.3gpp.org/FTP/tsg_sa/WG3_Security/TSGS3_45_Ashburn/Docs>.

[8] 3rd Generation Partnership Project, "LS on Security in HSPA Evolution, Release 7," 3GPP TSG SA WG3 Security Meeting #45, S3-060789, Nov. 2006, <http://www.3gpp.org/FTP/tsg_sa/WG3_Security/TSGS3_45_Ashburn/Output>.

[9] 3rd Generation Partnership Project, "LS on Potential Implementation of User Plane Encryption in LTE Base Station Site, Release 8," 3GPP TSG SA WG3 Security Meeting #46, S3-070153, Feb. 2007, <http://www.3gpp.org/FTP/tsg_sa/WG3_Security/TSGS3_46_Beijing/Docs>.

[10] 3rd Generation Partnership Project, "eNB Security Requirements," 3GPP TSG SA WG3 Security Meeting SA3#46b, S3-070256, Nokia, Siemens Networks, Mar. 2007, <http://www.3gpp.org/FTP/tsg_sa/WG3_Security/TSGS3_46b_LTESAE_adHoc_SophiaAntipolis/Docs>.

[11] 3rd Generation Partnership Project, "Mobile Radio Interface Signalling Layer 3, General Aspects (Release 7)," 3GPP TS 24.007, 2007, <http://www.3gpp.org/ftp/Specs/html-info/24007.htm>.

[12] 3rd Generation Partnership Project, "Specification of the 3GPP Confidentiality and Integrity Algorithms—Document 1: f8 and f9 Specification," 3GPP TS 35.201, 2007, <http://www.3gpp.org/ftp/Specs/html-info/35201.htm>.

[13] 3rd Generation Partnership Project, "High Speed Packet Access (HSPA) Evolution—Frequency Division Duplex (FDD) (Release 7)," 3GPP TR 25.999, v7.1.0, Mar. 2008, <http://www.3gpp.org/ftp/Specs/html-info/25999.htm>.

[14] 3rd Generation Partnership Project, "General Packet Radio Service (GPRS) Enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) Access," 3GPP TS 23.401, Rel. 8, 2008, <http://www.3gpp.org/ftp/Specs/html-info/23401.htm>.

[15] 3rd Generation Partnership Project, "Mobile Radio Interface Layer 3 Specification, Core Network Protocols, Stage 3," 3GPP TS 24.008, 2008, <http://www.3gpp.org/ftp/Specs/html-info/24008.htm>.

[16] 3rd Generation Partnership Project, "Security of H(e)NB (Release 8)," 3GPP TR 33.820, <http://www.3gpp.org/ftp/Specs/html-info/33820.htm>.

[17] 3rd Generation Partnership Project, "3G Security—Security Architecture," 3GPP TS 33.102, <http://www.3gpp.org/ftp/Specs/html-info/33102.htm>.

[18] ARM, "TrustZone Technology Overview," <http://www.arm.com/products/esd/trustzone_home.html>.

[19] M. Bauer, P. Bosch, N. Khrais, L. G. Samuel, and P. Schefczik, "The UMTS Base Station Router," Bell Labs Tech. J., 11:4 (2007), 93–111.

[20] R. M. Best, "Microprocessor for Executing Enciphered Programs," U.S. Patent 4,168,396 (Sept. 18, 1979).

[21] P. Bosch, L. Samuel, S. Mullender, P. Polakos, and G. Rittenhouse, "Flat Cellular (UMTS) Networks," Proc. IEEE Wireless Commun. and Networking Conf. (WCNC '07) (Hong Kong, Ch., 2007), pp. 3861–3866.

[22] J. Daemen and V. Rijmen, "AES Proposal: Rijndael—The Rijndael Block Cipher," National Institute of Standards and Technology (NIST) Proposal, Mar. 9, 1999, <http://ftp.csci.csusb.edu/ykarant/courses/w2005/csci531/papers/Rijndael.pdf>.

[23] R. W. Hamming, "Error Detecting and Error Correcting Codes," Bell Syst. Tech. J., 29:2 (1950), 147–160.

[24] T. Karnik, P. Hazucha, and J. Patel, "Characterization of Soft Errors Caused by Single Event Upsets in CMOS Processes," IEEE Trans. Dependable and Secure Comput., 1:2 (2004), 128–143.

[25] S. Kent and R. Atkinson, "IP Encapsulating Security Payload (ESP)," IETF RFC 2406, Nov. 1998, <http://www.ietf.org/rfc/rfc2406.txt>.

[26] T. Kinsley, "DDR2: The New DRAM Standard," Micron Technologies White Paper, 2006, <http://download.micron.com/pdf/whitepapers/DDR2_theNewStandard.pdf>.

[27] B. W. Lampson, "Authentication in Distributed Systems," Distributed Systems (S. Mullender, ed.), Addison-Wesley, New York, Wokingham, Eng., Reading, MA, 1993.

[28] G. Neiger, A. Santoni, F. Leung, D. Rodgers, and R. Uhlig, "Intel Virtualization Technology: Hardware Support for Efficient Processor Virtualization," Intel Technol. J., 10:3 (2006), 167–177.

[29] B. Preneel, Analysis and Design of Cryptographic Hash Functions, Ph.D. Thesis, Katholieke Universiteit Leuven, 1993.

[30] V. Prevelakis and D. Spinellis, "The Athens Affair," IEEE Spectrum, 44:7 (2007), 26–33.

[31] R. Sailer, E. Valdez, T. Jaeger, R. Perez, L. van Doorn, J. L. Griffin, and S. Berger, "sHype: Secure Hypervisor Approach to Trusted Virtualized Systems," IBM Research, RC23511 (W0502-006), Feb. 2, 2005, <http://domino.research.ibm.com/library/cyberdig.nsf/index.html>.

[32] G. E. Suh, AEGIS: A Single-Chip Secure Processor, Ph.D. Thesis, Massachusetts Institute of Technology (MIT), Sept. 2005.

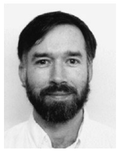
[33] United States, Department of Defense, "Department of Defense Trusted Computer System Evaluation Criteria," DoD 5200.28-STD, Dec. 1985, <http://nsi.org/ Library/Compsec/orangebo.txt>.

(Manuscript approved October 2008)

PETER BOSCH is a department head for the Services Infrastructure group at Bell Labs Antwerp, in Belgium. Prior to joining the Services Infrastructure group, he was a member of technical staff in the Bell Labs End to End Wireless Networking Research department in Murray Hill, New Jersey. After receiving his Ph.D. in computer science from the University of Twente in The Netherlands, he joined the Plan 9® operating system group at Bell Labs and later shifted his research focus to wireless systems research. He has worked on the initial prototypes for the BSR, co-developed the high-speed downlink packet access (HSDPA) demonstrator, integrated an enhanced mobility procedure for the BSR, and is now involved in resolving SAE/LTE BSR issues. His approach to building a well-integrated cellular system encompasses an end-to-end view of the distributed system—from radio frequency (RF) channels to IP-based applications—rather than partitioning the decision and development processes, which can lead to complicated systems that do not work well.

ALEC BRUSILOVSKY is a member of technical staff in the Secure Communications group of the Alcatel-Lucent Wireless Standards Development department in Naperville, Illinois. He is a lead delegate to the 3GPP SA3 (Security), with primary responsibilities in the security aspects of EPS and its interworking with HRPD as well as WiMAX. He has B.S. and M.S. degrees from the Telecommunications and Informatics University in Moscow, Russia. Prior to joining Alcatel-Lucent, Mr. Brusilovsky developed engineering and propagation design tools for US Cellular Corporation. He has also served as past chair of the IETF Working Group (IETF SPIRITS WG).

RAE MCLELLAN is a distinguished member of technical staff in the Bell Labs End to End Wireless Networking Research department in Murray Hill, New Jersey. He is an expert in computer architecture and VLSI systems. He created NuBus, the IEEE-1196 32-bit system bus standard used in the Apple Macintosh II; and architected and implemented Crisp, a 32-bit CMOS microprocessor, and a supporting system chipset for pen-based mobile computing. Recently, he created an integrated ASIC control system for a MEMS-based spatial light modulator. He also has designed telecommunication systems ranging from residential gateways in access networks to terabit optical routers. Mr. McLellan received his B.S. in electrical engineering, and M.S. in computer science, both from the Massachusetts Institute of Technology (MIT) in Cambridge. He has published 20 technical papers and holds eight U.S. patents.



SAPE MULLENDER is director of the Network Systems Lab at Bell Labs Antwerp, in Belgium. He has worked extensively in operating systems, multimedia systems, and, in recent years, wireless systems research. He was a principal designer of the Amoeba distributed system; he led the European Union's Pegasus project, which resulted in the design of the Nemesis multimedia operating system; and he made valuable contributions to work on the Plan 9® and Inferno® operating systems. He received a Ph.D. from the Vrije Universiteit in Amsterdam, The Netherlands, where he also was formerly a faculty member. He currently holds a chair part time in the Computer Science Department at the University of Twente. Dr. Mullender has published papers on file systems, high-performance remote procedure call (RPC) protocols, migratable object location in computer networks, and protection mechanisms, and he was involved in the organization of a series of advanced courses on distributed systems.

PAUL POLAKOS is director of the Bell Labs End to End Wireless Networking Research department in Murray Hill, New Jersey. His focus at Bell Labs is physics and wireless research. He has been instrumental in the definition and development of key technology initiatives for digital wireless systems, including intelligent antennas (IA) and multiple input-multiple output (MIMO) Bell Labs Layered Space-Time (BLAST), advanced base station and radio access network architectures, radio signal processing, enhancements to wireless networks for high data rates and high capacity, and dynamic network optimization. He holds B.S., M.S., and Ph.D. degrees in physics from Rensselear Polytechnic Institute in Troy, New York, and the University of Arizona in Tucson. Prior to joining Alcatel-Lucent, he was actively involved in elementary particle physics research at the U.S. Department of Energy's Fermilab and at the European Organization for Nuclear Research (CERN) and was on the staff of the Max-Planck Institute for Physics and Astrophysics in Munich. He is author or coauthor of more than 50 publications and holds numerous patents. ♦