Securing Wireless Data: System Architecture Challenges

Srivaths Ravi, Anand Raghunathan and Nachiketh Potlapally
Computer & Communications Research Labs
NEC USA, Princeton, NJ 08540
{sravi,anand,nachiketh}@nec-lab.com

ABSTRACT

Security is critical to a wide range of current and future wireless data applications and services. This paper highlights the challenges posed by the need for security during system architecture design for wireless handsets, and provides an overview of emerging techniques to address them. We focus on the computational requirements for securing wireless data transactions, revealing a gap between these requirements and the trends in processing capabilities of embedded processors used in wireless handsets. We also demonstrate that the use of security protocols causes significant degradation in battery life, a problem that will worsen due to the slow growth in battery capacities. These trends point to a *wireless security processing gap* that, unless addressed, will impede the deployment of secure high-speed wireless data and multi-media applications. We discuss approaches that are currently being pursued to bridge this gap, including low-complexity cryptographic algorithms, security enhancements to embedded processors, and advanced system architectures for wireless handsets that are enabled by new system-level design methodologies.

Categories and Subject Descriptors

C.2.0 [**Computer Systems Organization**]: Computer-Communication Networks- *General (Security and protection)*; C.2.1 [**Computer Systems Organization**]: Computer-Communication Networks- *Network Architecture and Design (Wireless Communication)*; E.3 [**Data**]: Data encryption; K.6.5 [**Computing Milieux**]: Management of Computing and Information Systems- *Security and Protection*; D.4.6 [**Software**]: Operating Systems- *Security and Protection*; C.5.3 [**Computer Systems Organization**]: Computer System Implementation- *Microcomputers (Portable devices)*; C.0 [**Computer Systems Organization**]: General- *System architectures*

General Terms

Security, Performance, Design, Algorithms

Keywords

Security, Security processing, Encryption, Decryption, Mobile Computing, Wireless Communications, Handset, Embedded system, Performance, DES, 3DES, AES, RSA, SSL, IPSec, WTLS, Design methodology, Platform, System architecture

1. SECURITY CONCERNS IN WIRELESS NETWORKS

The large scale adoption of wireless communication and mobile computing technologies has led to several applications that involve access to, and transmission of, sensitive information, making security a serious concern [1, 2, 3]. While security has been extensively addressed in the context of wired networks, the deployment of high-speed wireless data and multi-media communications ushers in new and greater challenges. The use of a public transmission medium implies that the physical signal is easily accessible to malicious entities. Furthermore, the mobility of wireless handsets, and the possibility of their being lost or stolen, present added security concerns. Perhaps most significantly, wireless clients (*e.g.*, smart phones, PDAs, networked sensors) are much more constrained in their processing capabilities and energy supplies (batteries), than their wired counterparts. Several security protocols and standards have been developed in the context of wired networks [4, 5], however, the above factors make wireless security an important field in its own right [6, 7, 8, 9].

The role of security mechanisms and protocols is to ensure privacy and integrity of data, and authenticity of the parties involved in a transaction. In addition, it is also desirable to provide functionality such as non-repudiation, preventing the use of handsets in denial-of-service attacks, filtering of viruses and malicious code, and in some cases, anonymous communication.

It is important to recognize that wireless security is an end-to-end requirement, and can be sub-divided into various security domains.

- *Appliance domain security* attempts to ensure that only authorized entities can use the appliance, and access or modify the data stored on it.
- *Network access domain security* ensures that only authorized devices can connect to a wireless network or service, and ensures data privacy and integrity over the wireless link.
- *Network domain security* addresses security of the infrastructure (voice and data) networks that support a wireless network. Infrastructure networks are typically wired, could include public networks, and could span networks owned by multiple carriers.
- *Application domain security* ensures that only safe and trusted applications can execute on the appliance, and that transactions between applications executing on the client and application servers across the Internet are secure.

Wireless security can be best addressed only if it is considered during the design of the network architecture, security protocols, and cryptographic algorithms, as well as the software (operating system, application software) and hardware architecture of the handset. In this paper, we focus on the requirements imposed by security protocols on system architectures for wireless handsets. We present an analysis of the computational requirements for securing wireless data transactions, revealing a gap between these requirements and the trends in processing capabilities of embedded processors used in wireless handsets. We also demonstrate that the use of security mechanisms significantly degrades battery life, and that the growth in battery capacities will lag far behind the energy requirements for secure wireless data communications. These trends result in a *wireless security processing gap*, which needs to be addressed through new system architectures and design methodologies.

The rest of this paper is organized as follows. Section 2 provides a brief overview of current and emerging security protocols that are relevant to wireless handsets. Section 3 analyzes the computational requirements for secure data transactions, and demonstrates the presence of a wireless security processing gap. Section 5 describes various approaches to bridging this gap, including low-complexity cryptographic algorithms and security protocols, security-specific enhancements to embedded processors, and new system architectures and design methodologies.

2. BACKGROUND

Wireless data communications can be secured by employing security protocols that are added to various layers of the protocol stack, or within the application itself. Security protocols utilize cryptographic algorithms (asymmetric or public-key ciphers, symmetric or private-key ciphers, hashing functions, *etc.*) as building blocks in a suitable manner to achieve the

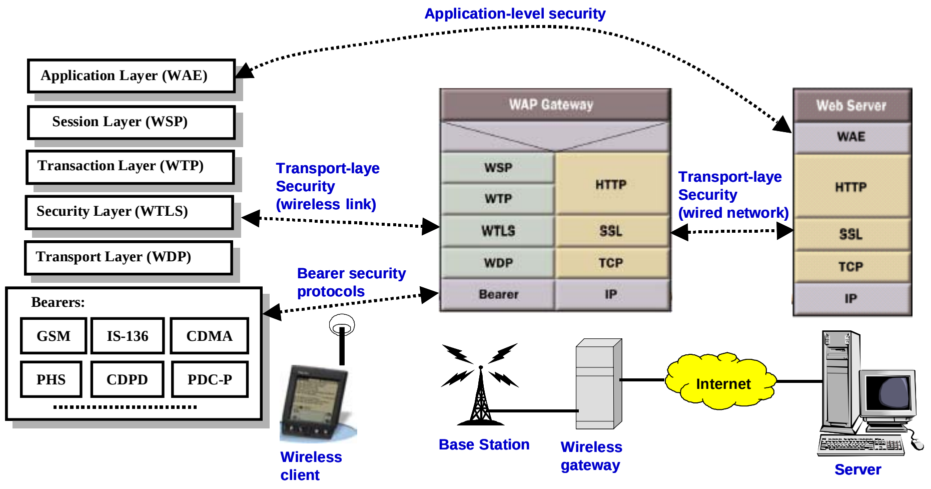


Figure 1: Security protocols in a wireless data network

desired objectives (peer authentication, privacy, data integrity, *etc.*). In the wired Internet, the most popular approach is to use security protocols at the network or IP layer (IPSec), and at the transport or TCP layer (TLS/SSL) [4, 5].

In the wireless world, however, the range of security protocols is much broader. Different security protocols have been developed and employed in cellular technologies such as CDPD [10] and GSM [11, 12], wireless local area network (WLAN) technologies such as IEEE 802.11 [13], and wireless personal area network technologies such as Bluetooth [14]. Many of these protocols address only network access domain security, *i.e.*, securing the link between a wireless client and the access point, base station, or gateway. Several studies have shown that the level of security provided by most of the above security protocols is insufficient, and that they can be easily broken or compromised by serious hackers [15, 16, 17, 18, 19, 20, 21]. While some of these drawbacks are being addressed in newer wireless standards such as 3GPP [22, 23] and 802.11 enhancements [13], it is generally accepted that they need to be complemented through the use of security mechanisms at higher protocol layers.

With the push to bring wired Internet data and applications to wireless handsets, and to enhance the wireless data experience, conventional Internet protocols are being increasingly used in wireless networks, by overlaying them on top of the underlying "bearer" technologies. This is leading to an increased adoption of widely accepted Internet security protocols to secure wireless data as well.

To illustrate how various security protocols fit into the context of a wireless handset, we consider a wireless network that uses the Wireless Application Protocol (WAP) [24]. Figure 1 shows a network architecture in which a wireless client communicates with a web server across the Internet, through a base station and a wireless gateway. The WAP standard defines protocols for the wireless link, which can be overlaid on top of existing wireless bearer technologies, such as GSM, CDPD, CDMA, *etc.* The WAP gateway translates traffic to/from the wireless handset (which uses the WAP protocol stack), to conventional Internet protocols (HTTP/TCP/IP), thereby facilitating inter-working with existing Internet servers. The architecture of Figure 1 allows for the use of security schemes at multiple layers of the protocol stack.

- Security protocols provided in the bearer technologies (such as CDPD, GSM, CDMA, *etc.*) may be used to provide network access domain security, including user authentication to the serving network, as well as a basic level of confidentiality and integrity over the wireless link. Note that, these security protocols may be employed for both voice and data, and independent of the nature of the data or application. However, as mentioned earlier, security protocols used in bearer technologies are mostly considered to be insufficient for data requiring high levels of security. Moreover, these techniques do not address the problem of maintaining end-to-end security across the wired infrastructure network.
- The WAP protocol stack includes a transport-layer security protocol, called WTLS, which provides higher layer protocols and applications with a secure transport service interface and secure connection management functions. WTLS bears similarities to the Internet security standard TLS/SSL, while including additional features such as datagram support, optimized handshake, and dynamic key refresh.
- Finally, specific applications may decide to directly employ security mechanisms instead of, or in addition to, the aforementioned options (through an application-level security protocol such as SET [4], or to provide additional functionality, such as non-repudiation, that is not provided in the transport-layer security protocol).

A well known concern with the WAP security architecture is the existence of a "security gap" at the wireless gateway, which arises since the translation between different transport-layer security protocols causes data to exist in decrypted form. This problem can be somewhat alleviated by maintaining the WAP gateway within a secure network domain (*e.g.*, behind the same firewall as the web server) [7]. Alternatively, the use of an end-to-end security protocol between the wireless handset and wired server eliminates this problem. For example, NTT DoCoMo's iMode service uses SSL to secure end-to-end connections [25], and the recently released WAP 2.0 specification includes a new mode that uses standard Internet protocols (HTTP/TLS/TCP/IP) between the wireless client and a server across the Internet [24].

3. WHAT MAKES WIRELESS SECURITY PROCESSING CHALLENGING?

In this section, we focus on the significant system design challenges to implementing security protocols (and the cryptographic algorithms they employ) on wireless handsets. Wireless data requires at least equal, and often a higher level of, security compared to wired data transmission. In addition, the need to maximize inter-operability with existing Internet applications, while providing end-to-end security, requires wireless clients to execute the same security protocols as servers across the wired Internet. However, wireless clients differ significantly from wired clients (such as desktop PCs) in their computational ability (processing power), and energy supply (battery). Furthermore, due to their small size and portability, wireless handsets must be designed considering the increased risk of being lost or stolen. Due to the above factors, wireless handset designers must face the following challenges:

- **Security processing gap:** It is well known that security protocols significantly increase computational requirements at the network clients and servers [6, 26]. These security processing requirements stretch the limited resources of wireless clients, significantly impairing user experience due to poor connection latencies and data rates, and shorter battery life. For example, a PalmIIIx handset requires 3.4 minutes to perform 512-bit RSA key generation, 7 seconds to perform digital signature generation, and can perform (single) DES encryption at only 13 kbps, even if the CPU is completely dedicated to security processing [27]. The computational requirements for security processing, and the disparity between these requirements and the computational capabilities of the embedded processors used in wireless handsets, are further explored in Section 4.
- **Battery gap:** Another critical bottleneck to security processing on wireless handsets is battery capacity, whose growth (5-8% per year) is far slower than the growth in security processing requirements or processor performance. Even without the energy overhead of security processing, the increase in energy requirements of wireless handsets is already out-pacing the growth in battery capacities, leading to a

battery gap [28]. It is only to be expected that the addition of security will further widen the battery gap. However, the magnitude by which energy requirements increase due to the addition of security is alarming. For example, based on the data presented in [29], a Sensoria WINS wireless sensor network node requires $21.5mJ$ ($14.3mJ$) to transmit (receive) a 1024-bit message, while encrypting the same message using RSA imposes an additional energy overhead of $42mJ$.

- **Flexibility:** The need for efficiency in security processing has to be considered together with, and traded off against, the need for flexibility. Each security protocol standard typically specifies a wide range of cryptographic algorithms that the network servers and clients need to execute in order to facilitate inter-operability [4, 5]. Further, a wireless handset is often required to execute multiple distinct security protocol standards in order to support (i) security processing in different layers of the network protocol stack (*e.g.* a wireless LAN enabled PDA that needs to connect to a virtual private network, and support secure web browsing may need to execute WEP, IPSec, and SSL), or (ii) inter-working among different networks (*e.g.*, an appliance that needs to work in both 3G cellular and wireless LAN environments). Finally, it is desirable, and often necessary, to allow for easy upgrade to future security protocols and evolving standards. Flexibility is obtained through the use of programmable processors and software implementations, while efficiency is typically obtained through custom hardware implementations. Marrying flexibility and efficiency is a significant challenge that requires the use of well designed HW/SW system architectures and system-level design methodologies.

- **Tamper-proof implementation:** It is critical to ensure that the security of a wireless handset is minimally compromised even if it physically falls into the hands of a malicious entity (*e.g.*, if it is lost or stolen), or if malicious software executes on it. While simple techniques such as password or PIN protection provide a basic level of user authentication, advanced biometric identification techniques are of increasing interest [30]. Biometric identification techniques introduce new system design challenges, such as integration of biosensor peripherals (*e.g.*, for fingerprint acquisition) into the system, secure storage for the valid templates, and efficient implementations of matching techniques. The use of "sandbox" execution environments, together with the use of digital certificates to verify the authenticity of downloaded programs, can minimize the possibility of malicious software executing on the handset [9]. However, the highest level of security requires hardware and firmware schemes such as isolation of relevant regions of system memory and peripherals, encryption of data on the system bus, *etc.* A variety of advanced techniques, such as linear and differential cryptanalysis [5], fault analysis [31], timing attacks [32], and power analysis [33], have been developed to break even the most complex ciphers. Resistance to some of these techniques can be built in only through careful circuit and HW/SW architecture design. While we do not discuss these issues further in this paper, it bears mentioning that several tamper-proof design techniques have been developed in other contexts, such as the design smart cards [34], and will be increasingly used by wireless handset architects and designers.

4. WIRELESS SECURITY PROCESSING GAP

In this section, we analyze the computational requirements of some popular cryptographic algorithms used in security protocols, and demonstrate that they impose a significant burden relative to the capabilities of embedded processors used in wireless handsets. Most security protocols employ at least three basic categories of cryptographic algorithms - symmetric ciphers, message authentication or hash functions, and public key ciphers. Public key algorithms (*e.g.*, RSA, DSA, Diffie-Hellman key exchange, ECC, *etc.*) are typically used for authentication and key exchange, while symmetric algorithms (*e.g.*, DES, 3DES, IDEA, RC4, AES, *etc.*) are used to ensure confidentiality, and message authentication algorithms (*e.g.*, MD2, MD5, SHA, *etc.*) are used to implement data integrity.

Figure 2 plots the security processing requirements in MIPS (millions of instructions per second) for the popular symmetric encryption algorithms, 3DES and AES, and integrity algorithms, MD5 and SHA, for different data rates. The data rates typical of current and emerging cellular (128 kbps - 2 Mbps) and wireless LAN (2 Mbps - 60 Mbps) technologies are indicated in the figure. For example, the processing requirements for 3DES, AES, SHA and MD5 at 10 Mbps are 535.9, 206.3, 115.4 and 33.1 MIPS, respectively. Note that security protocols require combined usage of a symmetric key encryption algorithm and a message authentication algorithm during the bulk data transfer phase. The curve labeled `composite` in Figure 2 plots the total processing requirement for using 3DES for encryption/decryption and SHA for message authentication in a security protocol. This curve indicates the workload that a security protocol can impose on the handset processor in order to support a given data rate. For example, to support data rates of 2 Mbps and 10 Mbps, the security processing requirements are 130.3 MIPS and 651.3 MIPS, respectively. In comparison, a state-of-the-art handset processor, such as Intel's StrongARM processor SA-1110, is capable of delivering around 150 MIPS at 133 MHz and 235 MIPS at 206 MHz [35]. The above data indicates a clear disparity between security processing requirements and available processor capabilities, even when assuming that the handset processor is fully dedicated to security processing. In reality, the handset processor also needs to execute the operating system, network protocols, and application software, which by themselves represent a significant processing workload.

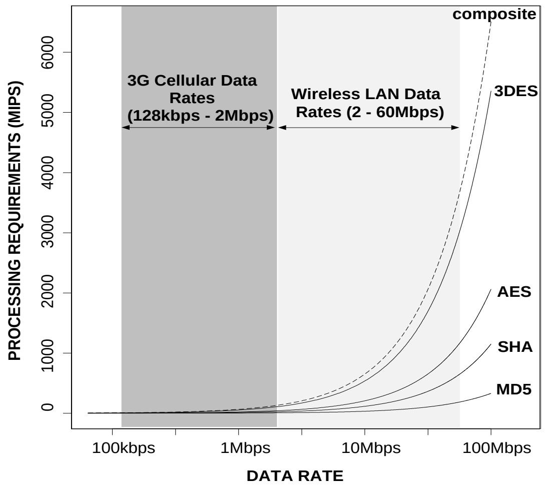


Figure 2: Processing requirements of cryptographic algorithms at different data rates

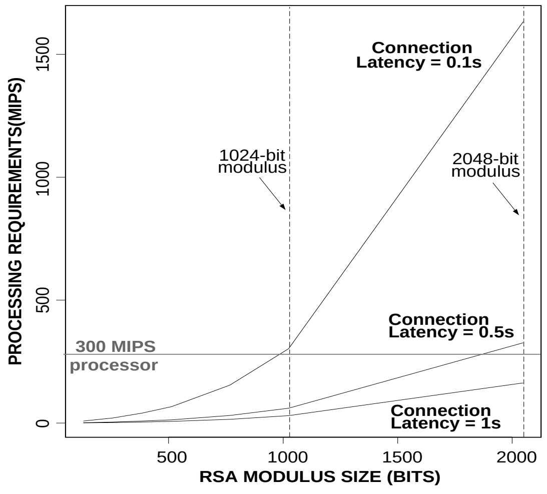


Figure 3: Processing requirements of RSA-based SSL handshake for different connection latencies and modulus sizes

While calculating the MIPS requirements for a wireless client, we should also factor in the processing burden due to authentication and key exchange services provided by a security protocol. For example, connection-oriented protocols such as SSL use public-key algorithms during connection setup for this purpose. The security processing performed during connection setup adds directly to connection latency. Figure 3 shows the processing requirements for RSA-based SSL handshake, for varying modulus sizes and three different connection latencies ($0.1sec$, $0.5sec$, and $1sec$). For any given connection latency, the MIPS required rises with increasing levels of security, *i.e.*, increasing modulus/key sizes. The figure also indicates clearly the combinations of security levels and connection latencies that can be supported by a given handset processor. For example, a 300 MIPS handset processor can support 1024-bit RSA-based authentication at connection latencies of $0.5sec$ and $1sec$, but not at $0.1sec$. Note that the above analysis assumes that the processor is completely dedicated to security processing; other processing requirements will further add to the connection latency.

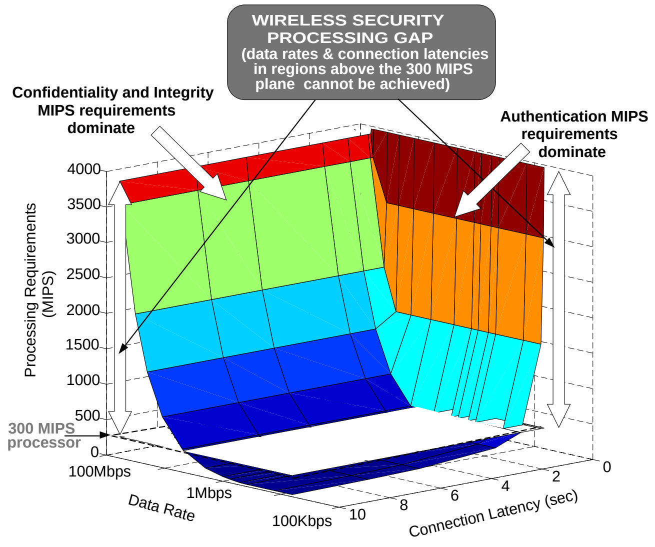


Figure 4: The wireless security processing gap

Figure 4 combines the data from Figures 2 and 3, and captures the effective workload on a processor for various combinations of connection latencies and data rates. Any given tuple (*cl*,*dr*,*mips*) in this 3-dimensional space specifies the workload (*mips*) involved in performing RSA-based authentication with a latency of *cl* as well as 3DES-based confidentiality and SHA-based integrity computations at data rates of *dr*. In general, the profile shows that for low values of *cl*, processing MIPS due to RSA-based authentication dominate, while for high values of *dr*, 3DES and SHA related processing dominate. Any embedded processor is represented in the 3-d space by a plane corresponding to its MIPS specification. For example, a 300 MIPS processor is represented by the white plane shown in the figure. Regions in which the processing requirements surface lies below the processor capability plane capture security processing workloads that can be met by the processor, while regions above the plane correspond to the collection of data rates and connection latencies that can not be met with the given processing capability.

The data presented in this section clearly demonstrates the presence of a *wireless security processing gap*, due to a mismatch between wireless security processing requirements and the processing capabilities of embedded processors used in wireless handsets. While embedded processor performance can be expected to increase due to improvements in fabrication technologies and innovations in processor architecture, the increase in data rates (due to advances in wireless communication technologies), and the use of stronger cryptographic algorithms (to stay beyond the extending reach of malicious entities) threaten to further widen the wireless security processing gap.

5. BRIDGING THE WIRELESS SECURITY PROCESSING GAP

The wireless security processing gap defined in Section 4 is simply a mismatch between the computational workload demanded by security protocols and the computational horsepower supplied by the processor in the handset. Several attempts have been made to lower this gap either by making the wireless security protocols and their constituent cryptographic algorithms lightweight, or by enhancing the security processing capabilities of the handset processor. Sections 5.1 and 5.2 examine these developments. Finally, Section 5.3 describes a novel security processing platform that combines innovations in design methodologies and system architecture to realize flexible and efficient wireless security processing.

5.1 Low Complexity Security Protocols and Cryptographic Algorithms

The design of efficient security protocols for wireless handsets remains an active area of research. To maintain inter-operability of the wireless handset with peers across the wired network, many of the existing wired security protocols such as SSL need to be supported on the wireless handset. While many of these security protocols are considered unwieldy for small-footprint devices, it has been shown in [36] that by carefully selecting and implementing only a subset of a protocol's many features, it is possible to reduce the security processing workload on a wireless client. The security processing workload of a security protocol can also be reduced by selecting optimal SW implementations for its constituent cryptographic algorithms [37]. Security protocols can also also be made to adapt their encryption policies based on the content of the data being encrypted. Video encryption algorithms such as [38, 39] focus on protecting the more important parts of a video stream, thereby reducing the total amount of data encrypted.

Another way of reducing the workload of a security protocol is to use lightweight cryptographic algorithms for various security functions. For example, protocols such as SSL, WTLS *etc.* generally use RSA-based public-key cryptography for authentication. The security of the basic RSA algorithm is derived from the NP-hard problem, integer factorization [5]. RSA can thus provide high security if the modulus is a large integer (1024 or 2048 bits) whose factoring becomes extremely difficult. However, this means that the basic computation for decrypting data (modular exponentiation) must be performed using a large key, making it computationally expensive.

Security protocols can now take advantage of alternative public-key cryptosystems that provide high levels of security while demanding less computing and memory resources. Elliptic Curve Cryptosystems (ECC) [40] are based on the observation that there are no fast algorithms for computing discrete logarithms of points on elliptic curves. For a given level of security, the key sizes required in ECC, as well as the computed signatures, are smaller than RSA, making encryption more efficient. This feature makes ECC highly suitable for small-footprint handheld devices.

The lattice based cryptosystem NTRU [41] is another example of a public-key cryptosystem that is increasingly being used in wireless security software toolkits. Though NTRU uses private keys whose lengths are comparable to the ones used in RSA and other conventional algorithms, the basic operations in NTRU work only on portions of the key (7 or 8 bits) at a time. Consequently, manipulation of data with full keys is not needed, leading to simpler computations that can be performed at reasonable speeds on even 8-bit processors.

For bulk data encryption/decryption, the security limitations of the traditional DES, and the high computational requirement of 3DES have led to the development of the AES encryption standard [42]. In the 3G cellular standards, confidentiality and integrity are maintained using a low complexity block cipher Kasumi [22, 23].

5.2 Embedded Processors with Enhanced Security Processing Capabilities

There have been several attempts to improve the security processing capabilities of general purpose processors. Since most microprocessors today are word-oriented, researchers have targeted accelerating bit-level arithmetic operations such as the permutations performed in DES/3DES. Multimedia instruction set architecture (ISA) extensions such as those in PA-RISC's Max-2 [43] or IA-64 [44] already incorporate instructions for permutations of 8-bit or larger sub-words. For arbitrary bit-level permutations, only recently have efficient instructions been proposed [45]. Instruction set extensions have also been proposed for other operations such as substitutions, rotates and modular arithmetic present in different private-key algorithms [46].

Many such extensions have already been applied to embedded processors used in the wireless handset domain. For example, the SmartMIPS [47] cryptographic enhancements extend the basic 32-bit MIPS ISA to speed up secure data processing. Similar features are also found in the ARM SecureCore family [48]. The security processing capabilities of SecureCore processors can also be further extended by adding custom-designed cryptographic processing units through a co-processor interface. This is useful for delivering efficient performance on new and proprietary cryptographic algorithms without having to re-design the basic processor core.

The importance of adding cryptographic accelerators to the basic processor core is being increasingly recognized by processor developers for wireless handsets. TI's OMAP platform for 2.5G and 3G handsets [49] features an ARM processor and a TI DSP core for enhancing the performance of wireless data and multimedia applications. TI also offers a wireless security library that includes both hardware and software implementations of several cryptographic functions. These cryptographic accelerators can be integrated with the basic OMAP platform to improve the performance of security protocols such as SSL, IPSec and WTLS [50].

5.3 MOSES: A Mobile Security Processing Platform

MOSES (MObile SEcurity processing System) is a programmable security processor platform being developed at NEC to enable secure data and multi-media communications in next-generation wireless handsets. The objective is to address the wireless security processing gap indicated in Section 4, while allowing for easy programmability in order to support a wide range of current and future security protocol standards. The system architecture, shown in Figure 5 consists of layered, optimized software libraries that implement the cryptographic algorithms, and a state-of-the-art configurable and extensible processor that is customized for efficient security processing.

The MOSES project employs a novel system-level design methodology to build the HW/SW platform shown in Figure 5. The MOSES design methodology combines state-of-the-art commercial design tools with several novel domain-specific methodologies that are indispensable in deriving an optimized system architecture.

We next present a brief overview of various aspects of the MOSES project, including hardware/software architectures and design methodologies, and conclude with a performance analysis of the platform when used for accelerating a complete end-to-end secure wireless transaction. Further details are available in [37, 51, 52].

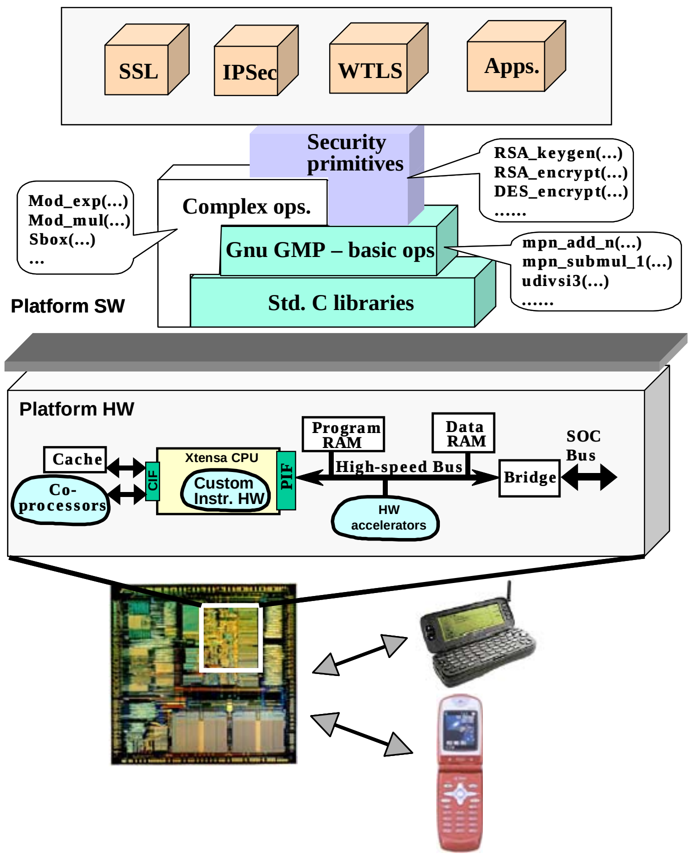


Figure 5: Overview of the MOSES security processing platform

5.3.1 SW Architecture

The software architecture for MOSES was designed using a layered philosophy, much like the layering used in the design of network protocols. At the top level, the SW architecture provides a generic interface (API) using which security protocols and applications can be ported to the platform. This API consists of security primitives such as key generation, encryption, or decryption of a block of data using a specific public- or private-key cryptographic algorithm (*e.g.* RSA, ECC, DES, 3DES, AES, *etc.*). The security primitives are implemented on top of a layer of complex mathematical operations such as modular exponentiation, prime number generation, Miller-Rabin primality testing *etc.*. These complex operations are in turn decomposed into basic mathematical operations, including bit-level operations (typically used in private-key algorithms) and multi-precision operations on large integers (typically used in public-key algorithms).

The use of such a layered SW approach has several advantages. The design of each SW layer can proceed concurrently, leading to drastic reductions in design times. Custom instructions can be developed for the basic operations layer without waiting for the SW implementation of higher layers to become available. The most important advantage, however, lies in the ability of the layered SW architecture to enable the use of design methodologies for co-designing the hardware and software constituents of MOSES (see discussion on design methodologies below).

5.3.2 HW Platform Architecture

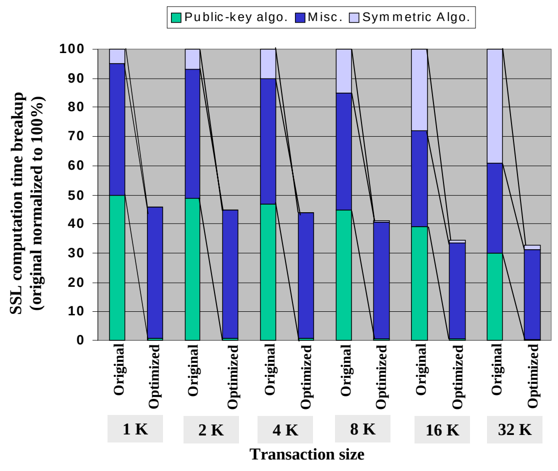
The hardware platform is based on the Xtensa configurable and extensible processor from Tensilica, Inc. [53]. Security processing enhancements to the basic hardware configuration are made in three different ways. For small granularity computations identified during design space exploration of a cryptographic algorithm, the instruction set of the processor is extended through the addition of custom instructions that speed up their operation. The added instructions are executed by custom hardware, which is tightly integrated into the processor execution pipeline. However, based on the specified area and performance constraints for the processor core, more coarse-grained cryptographic functions may be mapped to custom hardware outside the basic processor core. The custom hardware is added in some cases as a co-processor that interfaces to the Xtensa's single-cycle cache interface. In other cases, where high-performance communication with the processor core is not required, the cryptographic functions are implemented as peripheral hardware connected to the processor bus.

5.3.3 System Design Methodologies

The system design methodology for the MOSES platform included state-of-the-art commercial tools for tasks such as cross-compilation, profiling, instruction set simulation (ISS), and automatic generation of RTL descriptions of the extended processor. However, no well-developed tools/methodologies exist commercially for system design tasks such as exploration of algorithm and architectural design spaces, fast performance estimation, *etc.* Absence of such a system design tool suite can lead to unacceptable design times and inefficient design solutions. The system design methodologies used to design MOSES attempt to fill such missing links.

For example, in designing the optimized SW architecture for MOSES, it becomes necessary to find the best performing RSA algorithm for the SW platform in MOSES from nearly 500 RSA algorithm candidates [51]. Using just an instruction set simulator (ISS) for evaluation takes over a month of CPU time. However, by exploiting the layered SW architecture, performance macro-models can be derived for the different primitives in the basic operation layer, which can then be instantiated in the source code of the different RSA algorithm candidates. Native compilation and execution of the instrumented code can be used to estimate the performance of each algorithm. Finding the best performing algorithm using such a macro-modeling based performance estimation methodology takes only 4 hours and 40 minutes [51].

The overall MOSES system-level design flow includes novel techniques for algorithmic exploration and tuning (based on automatic performance characterization and macro-modeling of the software libraries), and architectural refinement based on selection of instruction extensions to accelerate performance critical, computation intensive operations. Details of these techniques are available in [52].



Note: Due to large speedups in the optimized case, the public-key and private-key components are not always visible in the above graph

Figure 6: Estimated speedups for SSL transactions

5.3.4 Performance

The MOSES platform facilitates the speed-up of security protocols such as SSL, IPSec, WTLS, *etc.* Figure 6 shows the estimated speedup from using MOSES as a handset processor that performs secure transactions using the OpenSSL [54] implementation of the SSL protocol. The SSL protocol includes a handshake phase, which first allows the server and client to authenticate each other, using public-key techniques such as RSA. Then, it allows the server to create symmetric keys, which are exchanged and used for rapid encryption and decryption of bulk data transferred during the session. The breakup of the computation workload for SSL processing between the private-key algorithm, public-key algorithm, and other miscellaneous computations, is indicated in Figure 6 for various session sizes. For small data transactions (where public-key algorithm computations in the SSL handshake dominate), the MOSES platform contributes to an overall transaction speedup of around 2.18X. In the case of large transactions, (where the private-key algorithm starts to dominate the overall computation) MOSES achieves an overall transaction speedup of 3.05X.

MOSES can also be used as a co-processor in a handheld device to accelerate security-specific computations. Functioning as a co-processor to an IPAQ 3870 PDA [55] playing a 10 MByte secure real-time video, MOSES facilitates a 9X reduction in connection setup latency and a 32X improvement in effective data rate. The significant speedups obtained through the MOSES platform indicate that new HW/SW architectures and system-level design methodologies can play an important role in bridging the wireless security processing gap.

6. CONCLUSIONS

Adequate security will be critical to enabling growth in a wide range of wireless applications and services. However, there are several challenges unique to wireless devices and their environment, which need to be addressed. We envision that, in addition to new security protocols optimized for the wireless environment, new system architectures and system design methodologies will be required to address many of these challenges, including the wireless security processing gap defined in this paper. Security considerations will become an integral part of system design for wireless handsets, rather than being addressed as an afterthought.

7. REFERENCES

[1] U. S. Department of Commerce, *The Emerging Digital Economy II*. `http://www.esa.doc.gov/508/esa/TheEmergingDigitalEconomyII.htm`, 1999.

[2] World Wide Web Consortium, *The World Wide Web Security FAQ*. `http://www.w3.org/Security/faq/www-security-faq.html`, 1998.

[3] *ePaynews - Mobile Commerce Statistics*. `http://www.epaynews.com/statistics/mcommstats.html`.

[4] W. Stallings, *Cryptography and Network Security: Principles and Practice*. Prentice Hall, 1998.

[5] B. Schneier, *Applied Cryptography: Protocols, Algorithms and Source Code in C*. John Wiley and Sons, 1996.

[6] S. K. Miller, "Facing the Challenges of Wireless Security," *IEEE Computer*, vol. 34, pp. 46–48, July 2001.

[7] P. Ashley, H. Hinton, and M. Vandenwauver, "Wired versus wireless ecurity - The Internet, WAP and iMode for e-commerce," in *Proc. 17th Annual Computer Security Applications Conf.*, Dec. 2001.

[8] *Wireless Security Basics*. Certicom (`http://www.certicom.com/about/pr/wireless_basics.html`).

[9] A. K. Ghosh and T. M. Swaminatha, "Software security and privacy risks in mobile e-commerce," *Communications of the ACM*, vol. 44, pp. 51–57, february 2001.

[10] *Cellular Digital Packet Data System Specification, Release 1.1*. CDPD Forum, Jan. 1995.

[11] *European Telecommunication Standard GSM 02.09*. Digital Cellular Telecommunications System (Phase 2+): Security Aspects.

[12] C. Brookson, "GSM security: A description of the reasons for security and the techniques," in *Proc. IEE Colloqium on Security and Cryptography Applications to Radio Systems*, pp. 2/1–2/4, June 1994.

[13] *IEEE 802.11 Wireless LAN Standards*. IEEE 802.11 Working Group (`http://grouper.ieee.org/groups/802/11/`).

[14] *Bluetooth security white paper*. Bluetooth SIG Security Expert Group (`http://www.bluetooth.com/`), Apr. 2002.

[15] Y. Frankel, A. Herzberg, P. A. Karger, H. Krawczyk, C. A. Kunzinger, and M. Yung, "Security issues in a CDPD wireless network," *IEEE Personal Communications*, vol. 2, pp. 16–27, August 1995.

[16] S. Patel, "Weaknesses of North American wireless authentication protocol," *IEEE Personal Communications*, vol. 4, pp. 40–44, june 1997.

[17] J. R. Walker, *Unsafe at any key size: An analysis of the WEP encapsulation*. IEEE document 802.11-00/362 (`http://grouper.ieee.org/groups/802/11/Documents/`), Oct. 2000.

[18] N. Borisov, I. Goldberg, and D. Wagner, "Intercepting mobile communications: The insecurity of 802.11," in *Proc. ACM Int. Conf. Mobile Computing and Networking*, pp. 180–189, July 2001.

[19] W. A. Arbaugh, *An inductive chosen plaintext attack against WEP/WEP2*. IEEE document 802.11-01/230 (`http://grouper.ieee.org/groups/802/11/Documents/`), May 2001.

[20] A. Mehrotra and L. S. Golding, "Mobility and security management in the GSM system and some proposed future improvements," *Proceedings of the IEEE*, vol. 86, pp. 1480–1497, July 1998.

[21] ISAAC group, U. C. Berkeley, *GSM cloning*. `http://www.isaac.cs.berkeley.edu/isaac/gsm.html`.

[22] *3GPP Draft Technical Specification 33.102*. 3G Security Architecture.

[23] C. W. Blanchard, "Wireless security," *BT Technology Journal (*`http://www.bt.com/bttj/`*)*, vol. 19, pp. 67–75, July 2001.

[24] *Wireless Application Protocol 2.0 - Technical White Paper*. WAP Forum (`http://www.wapforum.org/`), Jan. 2002.

[25] S. Okazaki, A. Takeshita, and Y. L. Lin, "New trends in mobile phone security," in *Proc. RSA Conference (*`http://www.rsasecurity.com/conference/`*)*, Apr. 2001.

[26] G. Apostolopoulos, V. Peris, P. Pradhan, and D. Saha, "Securing electronic commerce: Reducing SSL overhead," in *IEEE Network*, pp. 8–16, July 2000.

[27] D. Boneh and N. Daswani, "Experimenting with electronic commerce on the PalmPilot," in *Proc. Financial Cryptography*, pp. 1–16, Feb. 1999.

[28] K. Lahiri, A. Raghunathan, and S. Dey, "Battery-driven system design: A new frontier in low power design," in *Proc. Joint Asia and South Pacific Design Automation Conf. / Int. Conf. VLSI Design*, pp. 261–267, Jan. 2002.

[29] D. W. Carman, P. S. Krus, and B. J. Matt, "Constraints and approaches for distributed sensor network security," Tech. Rep. #00-010, NAI Labs, Network Associates, Inc., Glenwood, MD, Sept. 2000.

[30] G. Lawton, "Biometrics: A new era in security," *IEEE Computer*, vol. 31, pp. 16–18, Aug. 1998.

[31] D. Boneh, R. DeMillo, and R. Lipton, "On the importance of checking cryptographic protocols for faults," *Springer-Verlag Lecture Notes in Computer Science (Proceedings of Eurocrypt'97)*, vol. 1233, pp. 37–51, 1997.

[32] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," *Springer-Verlag Lecture Notes in Computer Science*, vol. 1109, pp. 104–113, 1996.

[33] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," *Springer-Verlag Lecture Notes in Computer Science*, vol. 1666, pp. 388–397, 1999.

[34] O. Kommerling and M. G. Kuhn, "Design principles for tamper-resistant smartcard processors," in *Proc. USENIX Wkshp. on Smartcard Technology (Smartcard '99)*, pp. 9–20, May 1999.

[35] *Intel StrongARM SA-1110 Microprocessor Brief DataSheet*. `http://www.intel.com/design/strong/datashts/278241.htm`.

[36] V. Gupta and S. Gupta, "Experiments in Wireless Internet Security," in *Proc. Wireless Communications and Networking Conference*, pp. 860–864, Mar. 2002.

[37] N. Potlapally, S. Ravi, A. Raghunathan, and G. Lakshminarayana, "Optimizing Public-Key Encryption for Wireless Clients," in *Proc. IEEE Int. Conf. Communications*, pp. 1050–1056, May 2002.

[38] A. S. Tosun and W. Feng, "Lightweight Security Mechanisms for Wireless Video Transmission," in *Proc. Intl. Conf. on Information Technology: Coding and Computing*, pp. 157–161, Apr. 2001.

[39] J. Wen, M. Severa, W. Zheng, M. Luttrell, and W. Jin, "A Format-Compliant Configurable Encryption Framework for Acess Control of Multimedia," in *Proc. Intl. Wkshp. on Multimedia Signal Proc.*, pp. 435–440, Oct. 2001.

[40] N. Koblitz, *A Course in Number Theory and Cryptography*. Springer-Verlag, 1987.

[41] *NTRU Communications and Content Security*. http://www.ntru.com.

[42] *AES Algorithm (Rijndael) Information*. `http://csrc.nist.gov/encryption/aes/rijndael`.

[43] R. B. Lee, "Subword Parallelism with Max-2," *IEEE Micro*, vol. 16, pp. 51–59, Aug. 1996.

[44] Intel Corp., *Enhancing Security Performance through IA-64 Architecture*. http://developer.intel.com/design/security/rsa2000/itanium.pdf, 2000.

[45] R. B. Lee, Z. Shi, and X. Yang, "Efficient Permutations for Fast Software Cryptography," *IEEE Micro*, vol. 21, pp. 56–69, Dec. 2001.

[46] J. Burke, J. McDonald, and T. Austin, "Architectural Support for Fast Symmetric-Key Cryptography," in *Proc. Intl. Conf. ASPLOS*, pp. 178–189, Nov. 2000.

[47] *SmartMIPS*. http://www.mips.com.

[48] *ARM SecurCore*. http://www.arm.com.

[49] *OMAP Platform - Overview*. Texas Instruments Inc. (`http://www.ti.com/sc/omap`).

[50] *Reducing the Security Threats to 2.5G and 3G Wireless Applications*. Texas Instruments Inc. (`http://focus.ti.com/pdfs/vf/wireless/securitywhitepaper.pdf`).

[51] N. Potlapally, S. Ravi, A. Raghunathan, and G. Lakshminarayana, "Algorithm exploration for efficient public-key security processing on wireless handsets," in *Proc. Design, Automation, and Test in Europe (DATE) Designers Forum*, pp. 42–46, Mar. 2002.

[52] S. Ravi, A. Raghunathan, N. Potlapally, and M. Sankaradass, "System Design Methodologies for a Wireless Security Processing Platform," in *Proc. ACM/IEEE Design Automation Conf.*, pp. 777–782, June 2002.

[53] *Xtensa application specific microprocessor solutions - Overview handbook*. Tensilica Inc. (http://www.tensilica.com), 2001.

[54] *Open SSL Project*. `http://www.openssl.org`.

[55] *iPAQ 3870 PDA*. Compaq Corp. (`http://www.compaq.com/products/handhelds/`).