eliminating the collateral damage of P2P applications. However, this won't address the vulnerability of files held on laptops or homeworkers' PCs used beyond the organisation's boundaries. For the time being, your best defence in these circumstances is education about the dangers. But we all know how effective that is.

References

1. Kaspersky Lab issues 2010 cyberthreat forecast, Kaspersky, December 15 2009 <http://www.kaspersky.com/news?id=207575980>
2. Navy Releases New Information On Presidential Security Leak, WPXI Pittsburgh, February 28 2009 <http://www.wpxi.com/news/18818589/detail.html>
3. Justice Breyer Is Among Victims in Data Breach Caused by File Sharing, Washington Post, July 9 2008 <http://www.washingtonpost.com/wp-dyn/content/article/2008/07/08/AR2008070802997.html>
4. David Dittrich and Sven Dietrich. 'P2P as botnet command and control: a deeper insight'. In Proceedings of the 3rd International Conference On Malicious and Unwanted Software (Malware 2008). IEEE Computer Society, October 2008. <http://staff.washington.edu/dittrich/misc/malware08-dd-final.pdf>
5. Davis, Neville, Fernandez, Robert and McHugh, 'Structured Peer-to-Peer Overlay Networks: Ideal botnets command and control infrastructures?' 2008, <http://www.professeurs.polymtl.ca/jose.fernandez/BotnetC2-ESORICS-article-final.pdf>
6. Storm Worm DDoS Attack, SecureWorks, Feburary 8 2007 <http://www.secureworks.com/research/threats/view.html?threat=storm-worm>
7. Matthew Steggink and Igor Idziejcak, 'Detection of peer-to-peer botnets', University of Amsterdam, February 2008 <http://staff.science.uva.nl/~delaat/sne-2007-2008/p22/report.pdf>

Securing IP networks

Sindhu Xirasagar, communication processor software product manager and Masoud Mojtahed, applications architect, LSI Corporation

In part one of this two-part series on securing IP networks, the issues and technologies involved with securing these networks were addressed. We discussed how the evolution of packet-switched networks has driven the need for scalable security solutions. In this second and final installment, we'll address the necessary specifications for the silicon, software and hardware platforms that can enable applications with packet processing at up to 3Gbps and IPsec processing at up to 1.5Gbps for all applications.

Implementing secure networks: OEM perspectives

A viable security solution for emerging security requirements must provide fast and accurate IPsec processing, with minor incremental increase in cost and power. With increasing adoption of IPsec for user data traffic as well as control and management traffic, throughput requirements of over 1Gbps are becoming commonplace. Bulk crypto processors are not optimal for meeting these performance requirements due to unacceptable increases in associated cost and power. The ideal solution includes high-performance packet processors with embedded security engines that can process all networking functions and support inline IPsec processing without host intervention. Further, the embedded IPsec engine must have comprehensive support for the various encryption and authentication algorithms. Ideally, the silicon is also capable of providing some hardware acceleration for compute-intensive security key generation functions.

"Processors that support IPsec processing must be available as a family of devices that scale along a range of cost, performance and power to enable component re-use."

Wireless and enterprise markets require that OEMs offer a broad portfolio of products supporting a range of performance at different cost points. In today's cost-sensitive environment, managing the total cost of ownership (TCO) including development, maintenance, support and operational costs, is critical to business success. Component re-use across the product portfolio enables engineering and operational resources to be leveraged and minimises the need for expensive test equipment, thus significantly reducing TCO. Processors that support IPsec processing must be available as a family of devices that scale along a range of cost, performance and power to enable component re-use.

Time-to-market is a perennial challenge for OEMs. Today, silicon devices have become complex subsystems, and it is very time and resource intensive for OEMs to learn internal details of the device and program it at a register level. The silicon vendor must also provide device configuration and management software with well-abstracted, application programming interfaces (API). To support IPsec, the API should support configuring the embedded IPsec engine to run different algorithms, update Security Association (SA) and Security Policy (SP) information, and so on. OEMs can integrate this runtime software directly into their application and focus their own efforts on application development and system integration.

Software-related costs dominate system TCO. Software reuse is a key element of minimising software-related costs. Ideally,

a single-application software base should be used across the entire system portfolio. For this, the underlying software components supplied with the silicon must scale across the entire device family. Hence these software components must support all the devices in the processor family and have the same application interfaces (C-API). The API software must enable easy configuration and real-time management of all components of the processor including the embedded IPsec engine. The API should be consistent across the entire processor family.

While vendor-provided software certainly saves OEMs time and resources, product differentiation is also important. OEMs must be able to add features that may not be provided by the silicon vendor. A software architecture with stacked software components (layers) supporting successively higher levels of API abstraction is very useful for OEMs who want to selectively add proprietary features and deliver differentiated products.

Packet-switched network processing involves processing of several layer 1 and layer 2 protocols, in addition to IPsec processing for flows secured with IPsec. At a minimum, Ethernet and IP headers must be processed before IPsec encryption is applied and prior to the packet being sent to the next node in the network. Similarly, on the ingress side, IPsec decryption precedes other packet processing functions. This is followed by optional reapplication of the same or different IPsec encryption algorithm, before forwarding the packet to the next node in the network. For rapid software development, the silicon vendor must also make available a comprehensive tool suite that runs on standard PC platforms. The tools must provide integrated simulation support for end-to-end processing of IP flows including IPsec. Section 3.3 provides an overview of a comprehensive tool suite.

The silicon vendor must also provide a hardware platform incorporating the associated silicon device. This enables OEMs to start system integration before their own hardware becomes available, and shaves several months off product development intervals.

Characteristics of the ideal security processing platform

The ideal security processing platform must include a scalable family of devices. To cost-effectively address many of today's access and enterprise applications, a single chip must be capable of complete processing of up to 3Gbps of IPv4 and IPv6 packet traffic, and up to 1.5Gbps of IPsec traffic. Such a device must support end-to-end packet processing, including segmentation and reassembly, classification, policing, buffer management, traffic shaping and scheduling, statistics collection and reporting, and in-line IPsec processing with an embedded security engine.

The embedded security engine must support hardware acceleration of a comprehensive suite of protocols, encryption and authentication algorithms, as well as related support functions to rapidly and easily implement applications like IPsec and Secure Real-time Transport Protocol (SRTP). Key features of the embedded security engine include the following:

Security Protocols:

- IPsec per RFC 2401, 4301 and SRTP per RFC 3711
- ESP and AH protocol support per RFC 4302, 4303, 2406, 2402 in Tunnel and Transport Modes

Encryption Algorithms:

- DES and 3DES (CBC Modes)
- AES 128, 192, 256 (CBC and CTR Modes)
- AES-GCM 128, 192, 256 (Galois/Counter Mode)
- AES-CM (SRTP)

Authentication Algorithms:

- Message-Digest algorithm 5 (MD5) with HMAC
- Secure Hash algorithms (SHA-1, SHA-224SHA-256) with HMAC
- GHASH
- AES-XCBC-MAC
- AES-GCM (Galois/Counter Mode) (GHASH)

Key Support:

- Public key accelerator to offload host CPU from mathematically intensive functions required for RSA, Diffie-Hellman, ECC and DSA
- ANSI X9.17 compliant Pseudorandom Number Generator for generating keys and Initialization Vectors (IVs), and other tasks; Random IV generation for DES, Triple-DES and AES on a per-packet basis
- True Random Number Generator should be a nondeterministic random number generator used for generation of keys, cookies and nonces

"To cost-effectively address many of today's access and enterprise applications, a single chip must be capable of complete processing of up to 3Gbps of IPv4 and IPv6 packet traffic, and up to 1.5Gbps of IPsec traffic"

Security platform runtime software: layered architecture

As noted in Section 2, production-ready, runtime software is necessary to enable application developers to rapidly develop IPsec-enabled systems. The following software architecture diagram shows the major components and their interactions.

Ideally, communication processors should support all PDU-processing data-plane software written in high-level, purpose-designed languages that are compact and efficient. The data-plane software executes on the data-path engines such as the classifier, traffic manager and in-line IPsec engine. Control-plane and management-plane software execute on an embedded CPU or on an external CPU.

The runtime software architecture should provide access to chip functionality and have entry points at several abstraction levels via C-API. Ideally, runtime software should include API software at two levels of abstraction:

- Object-level API software to configure all the engines in the processor including the embedded IPsec engine

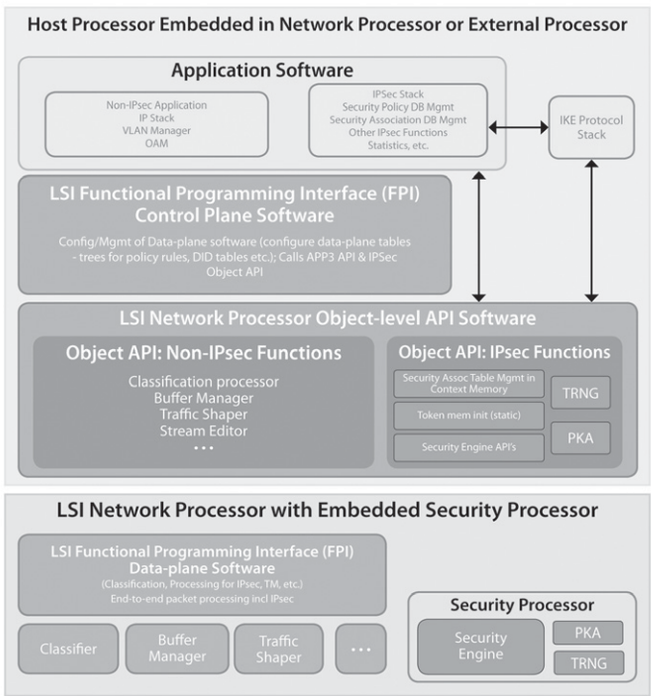


Figure 1: LSI Security Solution Layered Software Architecture.

- Functional API software that implements data-path and control-plane software for complete end-to-end flow processing including packet classification, IPsec encryption/decryption, filtering, traffic management and packet modification

The Object-level API software provides access to all the functions in the communication processor, including all functions in the embedded security engine. Security API functions include the following: add, remove and update Security Associations; update SPD and SAD tables used by the data-path code; and create, delete and modify data-path flows.

The API framework enables the Internet Key Exchange (IKE) functionality to provide the necessary parameters for proper functioning of IPsec as specified in their respective RFCs, irrespective of the IKE stack and the underlying IPsec hardware. To achieve this, the APIs and the associated parameters exposed to the end user must be generic in accordance to the IKE and IPsec RFCs.

Application-level Functional Programming Interface (FPI) software packages include data-plane software and control-plane software with API at a functional level, hence the term Functional Programming Interface Software. An FPI package is typically application specific, and implements comprehensive features to support end-to-end processing for all the types of flows that must be supported in the application.

Typical packet processing functionality includes classification, IPsec protocol processing, application of encryption and decryption algorithms, selection and application of traffic management algorithms, policing and modification of packets for forwarding. The data-plane software implementing these functions runs on the acceleration engines. The control-plane component of the Functional API software runs on the embedded or external host processor and enables dynamic configurability of the data-plane behaviour at the application level. The FPI software packages present APIs at the functional level of abstraction, hiding all device level details from the application programmer.

Availability of FPI software optimised for the target silicon from the silicon vendor or third parties is important since it significantly cuts down the amount of effort in software development for OEMs and system integrators, thereby reducing the total cost of ownership.

OEMs integrate their application software with Functional API software and Object-level API software provided by silicon vendors or third parties. As discussed earlier, re-use of application software over a portfolio of systems targeting a wide range of features and performance is highly desirable from the perspective of total cost of ownership. To enable such software scalability, it is desirable that the Functional API and Object-level API, including those supporting security functions, are consistent across different devices in the communications processor family.

Software Development Tool Suite

An important component of a programmable processor solution is a comprehensive suite of tools for rapid software development, debugging and optimisation. The tool suite should support development, debugging, simulation, testing and optimisation of applications without the use of actual hardware.

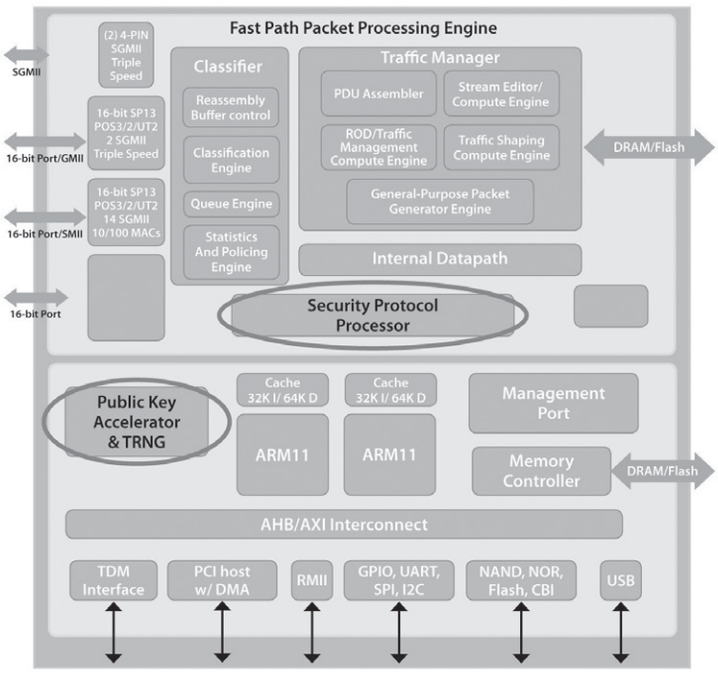
A user-friendly, unified-tools environment that has all functions accessible via a graphical user interface (GUI) and is easy to navigate is required. In addition to efficient compilers, advanced debuggers for all data-path software running on the processor must be included. Accurate, fast, end-to-end simulation of traffic flows is also required, and simulation metrics must be comprehensive and detailed so that software developers can quickly find pinch points and optimise the code.

"It is desirable that the Functional API and Object-level API, including those supporting security functions, are consistent across different devices in the communications processor family"

The tool suite must also include comprehensive traffic generators and analysers with support for typical network traffic distributions for all protocols including IPsec. In some systems, traffic flows with proprietary protocols are supported in conjunction with flows of standard protocols such as IP or IPsec. The traffic generators must support simple mechanisms for users to define such proprietary protocols.

High-Level Feature	Standards Document
IPsec	RFC 2401 - Security Architecture for the Internet Protocol RFC 4301 - Security Architecture for the Internet Protocol
ESP	RFC 2406 - IP Encapsulating Security Payload (ESP) RFC 4303 - IP Encapsulating Security Payload (ESP)
AH	RFC 2402 - IP Authentication Header RFC 4302 - IP Authentication Header
Attacks on Cryptographic Hashes	RFC 4270 - Attaches on Cryptographic Hashes in Internet Protocols Draft-hoffman-ide-ipsec-hash-use-02.txt – Use of Hash Algorithms in IKE and IPsec
Algorithm Usage Requirements	RFC 4305 - Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) RFC 4307 - Cryptographic Algorithms for Use in the Internet Key Exchange Version (IKEv2) RFC 4308 - Cryptographic Suites for IPsec
Ciphers	RFC 2405 - The ESP DES-CBC Cipher Algorithm With Explicit IV RFC 2410 - The Null Encryption Algorithm and Its Use With IPsec RFC 2451 - The ESP CBC-Mode Cipher Algorithms (3DES) RFC 3602 - The AES CBC-Cipher Algorithm and Its Use With IPsec RFC 3686 - Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP) NIST, FIPS PUB 197, Advanced Encryption Standard (AES), Nov 2001 NISA, Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation, Dec 2001 RFC 4309 - Using Advanced Encryption Standard (AES) CCM Mode With IPsec Encapsulating Security Payload (ESP), Dec 2005 RFC 4106 - The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP), Jun 2005 RFC 4543 - The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH, May 2006 NIST, The Galois/Counter Mode of Operation (GCM), May 2005 http://csrc.nist.gov/CryptoToolkit/Modes/pro posedmodes/gcm/gcm-revised-spec.pdf NIST FIPS 186-2, Digital Signature Standard, Jan 2000 http://csrc.nist.gov/publication/fips/fips186-2/fips186-2-change1.pdf ANSI X9.31 - RSA Digital Signature NIST FIPS 186-2, Digital Signature Standard, Jan 2000 http://csrc.nist.gov/publication/fips/fips186-2/fips186-2-change1.pdf ANSI X9.31 - RSA Digital Signature ANSI X9.62 - ECDSA, Elliptic Curve Digital Signature Algorithm
Authentic Algorithms	RFC 2104 - HMAC: Keyed Hashing for Message Authentication, Informational February 1997 RFC 2403 - The use of HMAC-MD5-96 within ASP and AH, Nov 1998 RFC 2404 - The use of HMAC-SHA-1-96 within ASP and AH, Nov 1998 RFC 3566 - The AES-XCBC-MAC-96 Algorithm and its Use With IPsec, Sep 2003 NIST, FIPS-180-2 Secure Hash Standard (SHA-1, SHA-2, SHA-384, SHA-512), Aug 2002 RFC4231 - Identifiers and Test Vectors for HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384 and HMAC-SHA-512, Dec 2005 RFC4494 - The AES-CMAC-96 Algorithm and Its Use With IPsec, Jun 2006
IKE and IKEv2	RFC 2407 - The Internet IP Security Domain of Interpretation for ISAKMP, Nov 1998 RFC 2408 - Internet Security Association and Key Management Protocol (ISAKMP), Nov 1998 RFC 2409 - The Internet Key Exchange (IKE), Nov 1998 RFC 4109 - Algorithms for Internet Key Exchange version 1 (IKEv1), May 2005 RFC 4306 - Internet Key Exchange (IKEv2) Protocol, Dec 2005 RFC 4434 - The AES-XCBC-PRF-128 Algorithm for the Internet Key Exchange Protocol (IKE), Feb 2006
NAT	RFC 3022 - Traditional IP Network Address Translator (Traditional NAT), Jan 2001 RFC 3715 - IPsec-Network Address Translation (NAT) Compatibility Requirements, Mar 2004 RFC 3948 - VDP Encapsulation of IPsec ESP Packets, Jan 2005
IP Protocol Numbers	www.iana.org/assignments/protocol-numbers
RTP and SRTP	RFC 3550 - RTP: A Transport Protocol for Real-Time Applications, Jul 2003 RFC 3711 - The Secure Real-time Transport Protocol (SRTP), Mar 2004 RFC 3551 - RTP Profile for Audio and Video Conferences with Minimal Control, Jul 2003 RFC 4568 - Session Description Protocol (SDP) Security Descriptions for Media Streams, Jul 2006
Diffie-Hellman	RFC 2539 - Storage of Diffie-Hellman Keys in the Domain Name System (DNS), Mar 1999 RFC 2631 - Diffie-Hellman Key Agreement Method, Jun 1999 RFC 3526 - More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE), May 2003 RFC 5144 (Jan 2008), 4754, 4753 and 4492
IP	RFC 1191 - Path MTU Discovery, Nov 1990

Table 1: Standards and specifications documents for implementation.



APP3300 and APP2200 communication processor device block diagram

Figure 2: Communication Processor with embedded IPsec engine: APP3300 and APP2200.

Acronym/Abbreviation	Definition
API	Application Programming Interface. A software layer interfacing directly to an application program providing all control and management functions.
SPP	Security Protocol Processor
IP	Internet Protocol
IKE	Internet Key Exchange
VPN	Virtual Private Network
AH	Authentication Header
ESP	Encapsulating Security Payload
IPsec	Internet Protocol Security
NPU	Network Processor Unit
SSL	Secure Socket Layer
TLS	Transport Layer Security
SSH	Secure Shell

Table 2: Definitions, acronyms and abbreviations.

The tool suite must support multi-chip, system-level configurations to enable system optimisation. Built-in automated regression test facilities should be available. Once optimisation is complete, the code is ready to be loaded on hardware. Utilities to minimise errors in moving from the software-based tools environment to hardware must be available. For example, a single API call supported by the runtime software should load the optimised binary image onto hardware.

Hardware Development Platform

All runtime software delivered by the silicon vendor must be validated using application-level test suites on a hardware platform similar to the LSI APP3300 hardware platform, shown in Figure 2. The hardware platform is a complete system development platform with flexible interface support, including multiple Gigabit Ethernet and Fast Ethernet interfaces, as well as other application-specific interfaces. Interfaces for plug-in modules enable platform extension for specific applications. Application demonstrations include end-to-end packet processing including IPsec, integrated IKE stack, and so on.

The evolution of packet-switched networks drove the need for scalable security solutions. IPsec is being widely adopted, especially in wireless infrastructure and enterprise applications In order to meet OEM requirements, silicon vendors must provide comprehensive silicon, software and hardware platforms to enable applications with packet processing at up to 3Gbps and inline IPsec processing at up to 1.5Gbps for all applications.

IPsec and IKE Industry Standards

The following industry standards comprise the complete set of specifications for IPsec and IKE. Most security standards are ratified as a standard by National Institute of Standards and Technology of the United States (NIST). The Internet Engineering Task Force (IETF) standards often reference these more detailed specifications without duplicating the contents, but rather specifying parameters within them.

About the Authors

Sindhu Xirasagar has 22 years in the technology industry and has held several positions in product management and marketing, strategic marketing and business development for a range of technologies including telecommunications solutions, semiconductors and client-server software. In her current role as product line manager for software and system solutions for LSI Corporation, she is responsible for integrated platforms for enterprise and communications infrastructure equipment, including wireline and wireless applications.

Masoud Mojtahed is an Applications Architect for LSI Corporation where he leads the software team to design and develop wireless Node B application software using the LSI Communications Processor families. Masoud has over 15 years of experience in the telecommunications industry and has worked in several positions involving global product support and hardware verification. He has also been a system engineer as well as a software engineer in both voice and data communication.