



# Software, IP security, and silicon acceleration, Part II

In last month's column a general security software architecture was described, some foundation terms and definitions were presented, and performance problem areas where security silicon companies are focusing development to help alleviate these bottlenecks was discussed. This column is the second part in the series where discussion will be focused more specifically on how one security silicon solution, Cavium Networks Nitrox product, accelerates and integrates with key management, IPsec, and Secure Sockets Layer (SSL) software components within the security software architecture.

## Quick security software architecture review

Figure 1 depicts the general security software architecture and identifies the potential performance bottleneck areas within the architecture. This was the ending point of last month's article. Added to the figure is a circle D that was not shown in last month's column. The D portion identifies performance problems associated with SSL communications.

A quick review of the bottlenecks shown in Figure 1 is described below. For information on the basic terms and definitions described here, refer to the September issue of *CompactPCI Systems, Software Corner* column or the online column at [www.compactpci-systems.com/columns/software\\_stack](http://www.compactpci-systems.com/columns/software_corner).

■ **Policy Management.** Typically, this is an *out-of-band* kind of activity involving the creation of policies and rules performed during initial provisioning of the box. Since this

is a relatively infrequent activity that occurs at the beginning of deployment, it is not time critical. It is therefore not necessary to look into more detail on performance acceleration options for provisioning of security policies. However, there is certainly a complexity issue. Thousands of rules and pass/drop/IPsec decisions can be complex to manage and maintain. While organizing these rules using graphical interfaces and rules languages is helpful, further control languages that help reduce provisioning are needed in the future (especially when combining with other features for a multi-function device).

- **Key exchange.** This is a key area of performance concern. A large number of connections that tend to be short-lived cause keys to be exchanged frequently. Since the key exchange involves multiple packets being passed between security gateways, this process can slow down performance and decrease capacity. The compute-intensive problem areas within key management are the public key math needed for either Diffie-Hellman or RSA operations, message hashing for SSL, and random number generation for Internet Key Exchange (IKE) nonces and IDs. It is this area where silicon acceleration can prove the most beneficial to increasing the number of key exchanges per second for a given solution.
- **IPsec processing.** There are two *sub-bottlenecks* within this block. The first is the processing of the security header itself. There are functions during this phase that analyze security headers, perform Security Association (SA) lookup, and

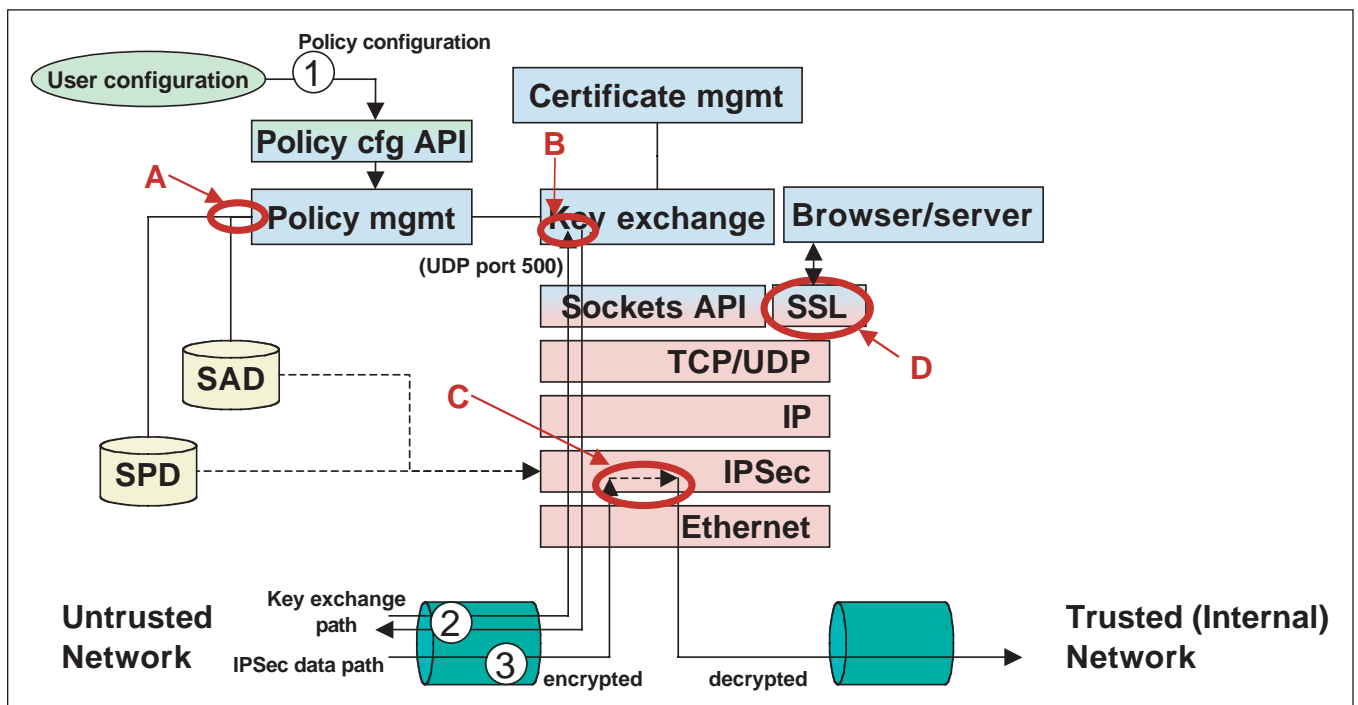


Figure 1

ensure IPsec packet replay attacks are not happening, etc. The other *sub-bottleneck* involves the actual encryption/decryption and hashing of the packet itself. As can be imagined, there are many algorithms that may be used and they involve multiple iterations/memory accesses, etc. As a result, this process can be extremely slow for software running on a general purpose CPU to execute.

- **SSL acceleration.** SSL record processing is another compute-intense area where security processing and encryption algorithms such as RC4 and hashing algorithms such as MD5 are used to provide security through the sockets layer. SSL acceleration was not mentioned last month, but there is significant momentum toward using SSL instead of IPsec to secure Virtual Private Networks (VPNs) due to the fact that SSL is *browser friendly* and does not require a separate client like VPNs do.

## Silicon acceleration overview

Cavium Networks ([www.cavium.com](http://www.cavium.com)) security silicon products provide an excellent illustration for security processing acceleration of software architecture at all of the bottlenecks identified. There are a number of security silicon products on the market today that provide *point* solutions specifically targeted at one or two of the bottlenecks described above. If one were interested in addressing all these issues with silicon assist, the traditional answer would be to add two to four chips on the circuit board. With the Cavium Networks Nitrox products, a single chip accelerates IPsec, SSL, and key management software.

The interesting usage scenario involving these bottlenecks is that they typically occur at different times. For instance, there may be a large burst of key exchanges as clients begin work or operation for the day. Key exchanges then become relatively infrequent. Once the large volume of key exchanges occurs, there are a large number of users generating IPsec traffic. Therefore, IPsec packet processing becomes intensive throughout the day.

One key area of differentiation for Cavium's Nitrox product lies in their *adaptive processing* technique. There are a number of compute engines within the Nitrox, each of which can provide IKE, SSL, and IPsec acceleration. The *adaptive* nature comes with the ability of the product to dynamically apply more compute resources to accelerate the function that's currently in the most demand. For example, in the environment description in Figure 1, one might find the Nitrox applying a large number of compute resources performing IKE acceleration during the VPN initialization burst. As more IPsec traffic flows through the device, the majority of compute resources would be shifted to perform IPsec acceleration throughout the day. This adaptive processing technique provides high integration and maximizes the usefulness of the part in a variety of security applications.

## Look-aside and in-line security silicon

Security processors come in two flavors, look-aside and in-line. In-line security processors sit functionally between the receive interface and the processor. In these cases, most of the IPsec processing information must be pre-programmed into the silicon's context memory, enabling it to perform the majority of the processing without the help of the CPU or network processor. This can be the fastest option, but at the cost of some flexibility.

Look-aside security processors sit *off to the side* of the CPU or network processor. The receive interface is connected directly to the processor. When the processor determines some portion of security processing is required, it passes the packet and some amount of control information to the security processor. The security processor then performs the tasks needed and passes the transformed packet back to the processor for transmission or further processing. Since the processor sits functionally between the security processor and the receive interface, it is less streamlined. However, because the processor gets first look at the packet, more complex operations can occur before and after security processing that can be important for many security applications.

There are stateful and stateless processing elements to IPsec processing. For example, anti-replay window and sequence numbering are stateful parts of the IPsec protocol. They change in a predetermined way with every packet. In-line security processor solutions must provide both stateful and stateless processing since they are directly in-line with the entire traffic flow through the system. Look-aside security silicon may only see a portion of the traffic and can only provide stateless processing. In these cases, the host processor is responsible for security association (SA) lookup, sequence number checking and generation, as well as anti-replay window and lifetime checking for the security association.

Another consideration is how the processor passes control information to the security processor. The more control information that is passed over the bus between these two parts, the less the bandwidth utilization efficiency. In the specific instance of the look-aside version of the Nitrox, the host processor looks up the SA and generates a 32-byte command. This command includes a pointer to the area in the chip's context memory that locates the SA matching the current packet being sent to the Nitrox. This command is passed along with the packet to the security processor. The Nitrox device performs Encryption Security Payload (ESP), tunnel and transport modes, and IPv4 and IPv6 processing. Finally, the software on the host processor does anti-replay processing. Similarly on outbound, a packet and a command are given to the device. The security chip will perform the encryption and header encapsulation. Passing pointers to the context memory instead of passing the entire contents of the SA along with the packet minimizes the communication bandwidth overhead associated with passing control information between the processor and the security silicon.

The Nitrox can be interrupt driven or polled, depending on the CPU capabilities and the aggregate throughput requirements of the system. It is possible for the number of interrupts generated by the security silicon to exceed the handling capabilities of the processor. In these high-end applications, polling the security processor is recommended.

## Software architecture with security acceleration

With the security review and introduction to security silicon complete, it is now time to integrate in the security architecture model security silicon and software interfaces. Figure 2 shows the security software architecture with the API points integrated along with Cavium's in-line version of their security processor, the Nitrox II.

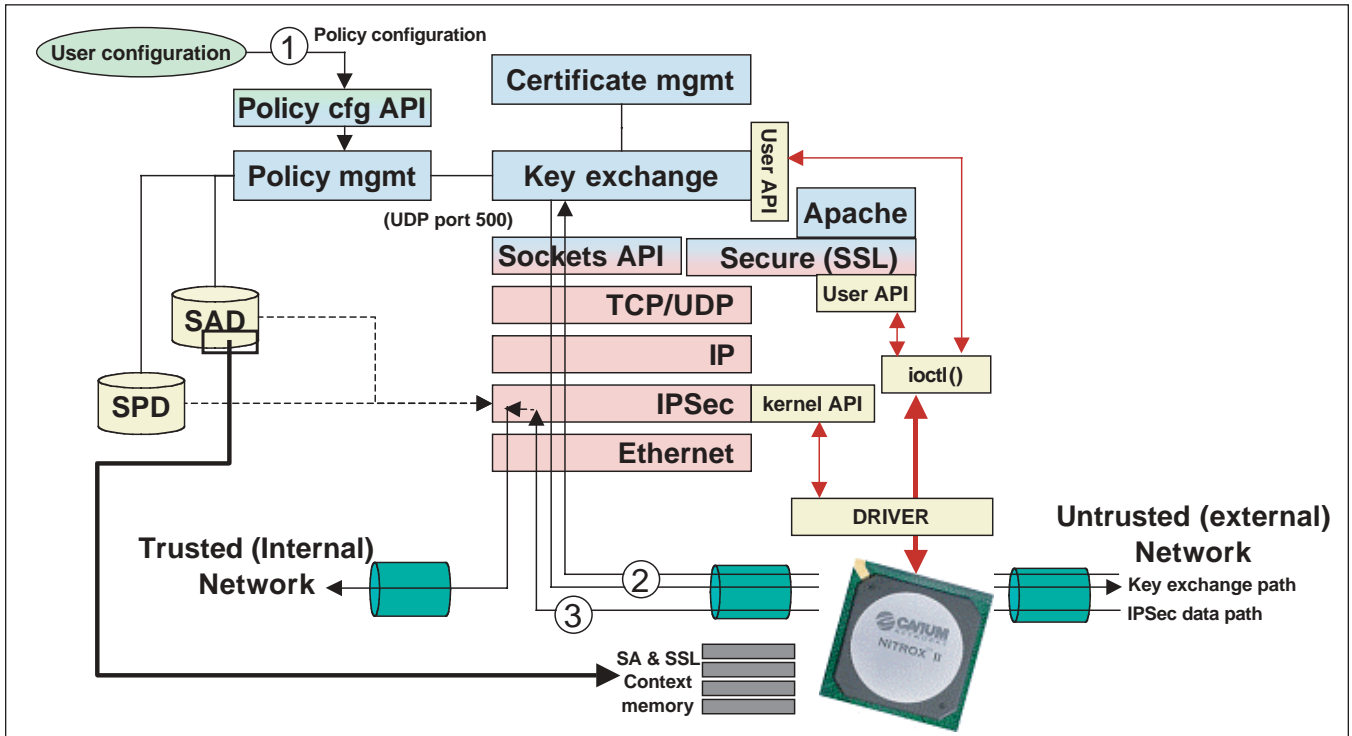


Figure 2

Cavium defines two main APIs for integration with security software stacks:

- A user API which provides IKE and SSL acceleration services.
- A kernel API which provides an IPsec exception data path for Nitrox II and a main IPsec data path for Nitrox 1.

The user API is the point of integration for the IKE and SSL layers. In order for security processing to be performed, the SA along with other context information needs to be stored in the SA and SSL context memory. Context related primitives provided with the Nitrox include:

- **Write\_IPsec\_SA.** This writes a security association record into the context memory of the security silicon. Notice that the Security Association Database (SAD) in the security software architecture needs to be enhanced slightly to store a pointer to this context memory where the SA is located once this primitive is invoked following a key exchange. Typically the SAD will contain the SA, but in this case, only a pointer to the SA is required to be stored in the SA. It is this pointer to the context memory that will be used to process packets matching the security association.
- **Write\_context, Read\_context.** These primitives allow the host processor to write and read the rest of the context information into the context memory of the security silicon.
- **Erase\_context.** As SA lifetimes or lifebytes are exceeded, this primitive provides a way to remove context from the security silicon.

There are also a set of primitives relating to key exchange services. These include:

- **RANDOM.** This provides random number generation for IKE nonces and IDs.
- **ME, ME\_LARGE.** These provide modular exponentiation processing services for the IKE layer.

The IPsec service primitives are:

- **Process\_inbound\_packet**
- **Process\_outbound\_packet**

The detailed function prototypes behind these IPsec packet service primitives are:

```
extern Uint32 Csp1ProcessInboundPacket (
    void* in,
    void* out,
    Uint64 ctx,
    int rlen,
    Request_cb cb,
    void *data);
```

```
extern Uint32 Csp1ProcessOutboundPacket (
    void* in,
    void* out,
    Uint64 ctx,
    int rlen,
    Request_cb cb,
    void *data);
```

These two primitives provide the services to the IPsec layer in kernel mode for IPsec header processing and encryption/decryption services provided by the security silicon.

These function calls have been integrated into IPsec stacks and key exchange software such as FreeSwan and the SSH security software products.

## SSL API

The SSL also has APIs associated with it to perform RC4 and MD5 processing. Below are three RC4 service APIs to provide an example for the kind of information that passes between the SSL layer in the software architecture and a security processor chip like the Nitrox.

```
UInt32 Csp1RsaServerFullRc4(
    OperationMode
    operation_mode,
    UInt64 context_handle,
    UInt64 *key_handle,
    HashType hash_type,
    SslVersion ssl_version,
    Rc4Type rc4_type,
    MasterSecretReturn master_secret_ret,
    UInt16 modlength,
    UInt8 *encrypt_premaster_secret,
    UInt8 *client_random,
    UInt8 *server_random,
    UInt16 handshake_length,
    UInt8 *handshake,
    UInt8 *client_finished_message,
    UInt8 *server_finished_message,
    UInt8 *encrypt_master_secret,
    UInt32 *request_id);
```

```
UInt32 Csp1EncryptRecordRc4(
    OperationMode operation_mode,
    UInt64 context_handle,
    HashType hash_type,
    SslVersion ssl_version,
    SslPartyType ssl_party,
    MessageType message_type,
    UInt16 message_length,
    UInt8 *message,
    UInt8 *record,
    UInt32 *request_id);
```

```
UInt32 Csp1DecryptRecordRc4(
    OperationMode operation_mode,
    UInt64 context_handle,
    HashType hash_type,
    SslVersion ssl_version,
    SslPartyType ssl_party,
    MessageType message_type,
    UInt16 record_length,
    UInt8 *record,
    UInt8 *message,
    UInt32 *request_id);
```

Specifically related to the Nitrox, these APIs have been integrated into IPsec stacks, SSL implementations, and key exchange software such as OpenSSL and the RSA Security SSL products. There is also a software development kit that comes with the Nitrox that provides the example applications, these APIs, a Linux driver, and underlying ioctl() calls that the APIs use to communicate with the driver.

## Conclusion

In this month's article, security software architecture from last month's article was reviewed, an example security processor from Cavium Networks was discussed, and a description of the integration required to integrate security silicon to accelerate IKE, Ipsec, and SSL processing. As available bandwidth moves from hundreds of megabits into gigabit and beyond, security silicon acceleration and their easy and effective integration into the security software environment will play an increasingly critical role in the development of network equipment now and into the future.

Many thanks goes out to Rajiv Khemani and Randy Devol of Cavium Networks for responding to inquiries with excellent information and insight into the latest technology in security silicon acceleration products, techniques, and trends that are covered in this column.

*Nitrox* is a trademark of Cavium Networks. Third party trademarks and brands are the property of their respective holders.

For further information, contact Curt at:

E-mail: [software@compactpci-systems.com](mailto:software@compactpci-systems.com)