

The latest cryptographic solutions from Linux on the System z platform

BY PETER SPERA



Can Linux* for the IBM* System z* platform meet the cryptographic needs of today's enterprise solutions? Simply put, yes. Cryptography in general can be a daunting topic with complexity and obfuscation at every turn. This article should lift the shroud a little, define and simplify some basic terminology and show how cryptography can be used in a System z enterprise solution.

The functionality of cryptography on Linux for the System z platform has been growing over the last few years. Software cryptography wasn't cutting it for the growing Web-server scenarios for Linux on the mainframe, known as Linux on S/390*, when hardware cryptography was introduced in the early days. It started a few years ago with simple SSL acceleration at a meager rate, roughly four times faster than software solutions of the time. In the short time since, the support has grown

ILLUSTRATION BY IMAGEZOO/IMAGES.COM



to not only include the SSL acceleration for clear-key applications at improved rates, but now extends to symmetric algorithms, digital signatures and secure key functions.

Handshakes and Keys

Those new to cryptography will start getting glassy-eyed without a brief explanation of a few terms. Currently one of the most computationally expensive crypto operations we encounter nearly every day is the SSL handshake. This handshake is what happens behind the scenes when we go to our favorite online store to buy the latest DVD. Once the handshake is complete, and both sides are trusted, that familiar lock is displayed in our Internet browser. The most common SSL handshake algorithm in use today is the asymmetric RSA algorithm (named after its creators, the esteemed cryptographers Rivest, Shamir and Adleman).

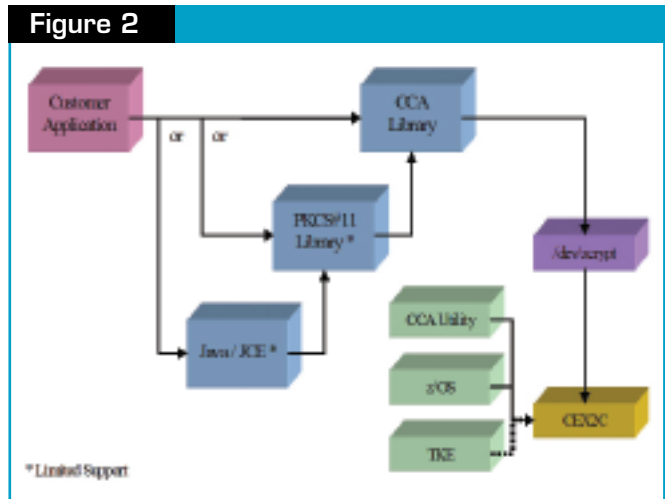
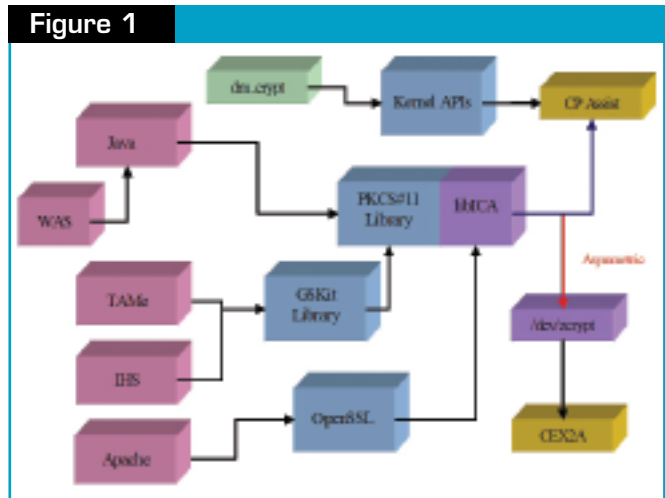
The term asymmetric describes the keys used in this type of cryptography. There is a public key and a private key, which are different; the keys are considered to be asymmetric. The private key is held secretly by the owner while the public key can be distributed widely without any concern for compromise. Public keys like RSA can be used to encrypt data for confidentiality, to sign data for integrity or to authenticate one user to another as is the case of the SSL handshake mentioned above.

Symmetric algorithms are quite different and use the same key for both encryption and decryption. These algorithms can be used to encrypt data at rest, on a hard drive or tape, or data in transit (e.g., the payload or data content of an SSL transaction). Symmetric keys, unlike public keys, shouldn't be shared unless all parties are trusted. Digital signature algorithms, such as the Secure Hash Algorithm (SHA), are also important cryptographic technologies found in the enterprise. Digital signatures are a common way to ensure the integrity of important data.

If that wasn't confusing enough for the newcomers to cryptography, there's also the notion of clear-key and secure-key cryptography. Clear key indicates that the key exists somewhere in the software stack in the clear. This is typically the mode of operation for most consumer retail Web sites. Secure-key cryptography, used by many financial or banking applications to indicate that the key can never be found in a readable form outside the actual cryptographic hardware, is discussed more on page 38.

The Hardware

In the beginning—at least as far as Linux on the System z platform or the older S/390 is concerned—there was the PCI Cryptographic Coprocessor (PCICC) cryptography card. This coprocessor card had the potential to perform many complex cryptographic functions. Linux, however, was just entering



the scene and robust cryptography wasn't an integral component or requirement at that point, so it was used only for RSA acceleration. This first offering got things rolling with roughly 200 handshakes per second. Looking back, it seemed only moments later when the PCI Cryptographic Accelerator (PCICA) card became available, adding five times the speed to the SSL support available to Linux applications on the mainframe.

After that, things really got rolling with the PCI-X Cryptographic Coprocessor (PCIXCC) card and now the Crypto Express2 feature. The Crypto Express2 feature is made up of two physical cards, each of which can be configured as either a crypto coprocessor or a crypto accelerator, adding a great deal of flexibility and configurability to an enterprise solution. A Crypto Express2 feature with only one card is now available with the System z9* Business Class (BC) for those workloads with fewer crypto demands. Each new generation of crypto hardware highlights IBM's commitment to security and the mainframe. In a Linux for System z environment, the Crypto Express2 feature, configured with each card as an accelerator, has reached speeds of

approximately 6,000 SSL handshakes per second per feature.

The entry of the z990 mainframe marked the end of the Cryptographic Coprocessor Facility (CCF), which architecturally predated Linux. The CCF was therefore unusable to Linux on the mainframe. The z990 and z890 brought an expanded architecture that included user space, clear key, instructions for the symmetric Data Encryption Standard (DES) and Triple DES (TDES) algorithms and the SHA-1 digital signature algorithm. Following suit the System z9 Enterprise Class (EC) and BC extended this support to include the symmetric Advanced Encryption Standard (AES) 128-bit algorithm, SHA-256 (which is part of the SHA-2 standard) and an instruction for pseudo-random-number generator.

Clear Sailing With System z

Now that the cryptographic hardware available for Linux on the System z platform (represented by the yellow boxes in Figure 1, page 36) is better understood, the software solutions that depend on the hardware can be explored. Asymmetric cryptography, symmetric cryptography or digital-signature cryptography are all available to applications requiring clear-key solutions. Crypto hardware alone isn't enough; software components are required to complete the solution, making the

crypto functions available to applications in a useable form. Cryptography libraries have been developed or extended to bring the cryptography acceleration found in the hardware to the protected business applications.

The cryptographic solutions on Linux for the System z platform can sound quite daunting or complex, so it's best to refer often to Figure 1 for clarity. If application development is attempted it's most important that only the supported APIs and libraries be used by applications. The appropriate APIs are seen in the blue boxes in the center of the diagram. While the underlying support or the device driver (represented by the purple boxes) could be utilized by applications, they can change at any time and should be avoided.

The approved APIs for access to crypto on Linux for the System z platform are OpenSSL, PKCS#11 (implemented as OpenCryptoki on Linux) and GSKit. They are all available to user-space applications. OpenSSL and PKCS#11 are provided to support well-established open standards. In addition, PKCS#11 also enables Java* applications to take advantage of the cryptographic support on Linux for the System z platform. Additionally, the GSKit libraries utilize System z hardware cryptography to provide support to IBM applications. These IBM applications ship GSKit as necessary. The device driver,

The core of the secure-key solution on Linux for the System z platform is a combination of the Crypto Express2 Coprocessor, the device driver and the Common Cryptographic Architecture libraries.

along with PKCS#11 and OpenSSL support can be found as part of both the Novell SUSE and Red Hat distributions available for Linux on the System z platform.

In addition to the aforementioned user-space APIs, kernel cryptography APIs were added to Linux as part of the 2.6 kernel to provide similar cryptography functions to features or applications running in the kernel space. On Linux for the System z platform these kernel APIs were extended to make use of the integrated crypto CP assist instructions that are now part of System z990 and 890 and System z9 EC and BC. A cryptographic file system, such as eCryptfs, and dm_crypt (pictured by the green box in Figure 1) are examples of kernel functions that were ported to Linux on System z. Each of these ports was completed with no modifications needed to utilize cryptography on the System z platform. They seamlessly took advantage of the copy (CP) assist instructions via the built-in kernel crypto APIs.

The pink boxes on the diagram's left side show the diversity of the applications that utilize hardware cryptography acceleration on Linux for the System z platform. To date, the highest exploitation of SSL acceleration comes from Apache*, but as more applications exploit the hardware, and as described above, more algorithms are implemented, this is beginning to shift. Technologies such as WebSphere* Application Server (WAS) with Java applications and IBM HTTP Server (IHS) now support the cryptographic hardware infrastructure on Linux for the System z platform, driving diverse workloads.

Random-number generation (RNG) and pseudo-random-number generation (PRNG) play an important role in many cryptographic solutions. The quality and availability of a steady stream of random numbers can adversely affect the security or acceleration benefits, respectively, realized in an end-to-end encryption solution. The PRNG CP assist instruction is being integrated into the Linux for System z kernel and will be accessible via `/dev/prandom`. This new device can be used to back `/dev/random` or `/dev/urandom`, as appropriate, based on an installation's security policy. Implementing the new device will provide installations the necessary flexibility to take advantage of hardware PRNG support from existing applications without the need to recode or recompile their critical applications.

Safe and Secure on the System z Platform

The latest extension to the cryptography support for Linux on the System z platform is the capability to use the secure-key functions that are part of the cryptography hardware. These new functions will be available starting with System z9 EC and BC with the Crypto Express2 card configured as a coprocessor. This support targets distributed applications that can be consolidated on the System z platform, providing an alternative to off-platform cryptographic solutions that can't offer the end-to-end security or consolidation benefits of a System z solution.

The core of the secure-key solution on Linux for the System z platform is a combination of the Crypto Express2 Coprocessor, the device driver and the Common Cryptographic Architecture (CCA) libraries. The cryptography card is represented by the yellow box shown in Figure 2 (page 36), while the purple box represents the device driver. The base software library needed to access the secure cryptography functionality in the hardware is CCA, pictured in one of the blue boxes in Figure 2. This new CCA support is similar to the support already available on the System i* platform, Linux for the System x* platform and AIX*. The CCA library will provide full support for the secure-key cryptographic functions. Early customer requirements indicated the need to extend the CCA support to include PKCS#11 and Java/JCE APIs for secure key as well. In response to these requirements, limited support has been developed for PKCS#11 and Java/JCE, pictured in the blue boxes of Figure 2, to enable key generation, data encryption and data decryption for the DES, TDES and RSA algorithms. Client applications, pictured by the pink box, can then have access to these secure-key functions.


The availability of secure-key solutions brings with it the requirement for card and key management. It's necessary to configure master keys for both symmetric and asymmetric functions in the hardware. This can be done via the Trusted Key Entry (TKE) utility, a new Linux CCA utility, or by configuring the hardware via `z/OS*` and then re-assigning the configured crypto card to the Linux image. Any of these solutions will produce the same result and provides the clients the diversity to choose the solution that best fits their existing key management security policies. In addition to key

management for master keys, it's necessary to consider the key store that will be required by the application or solution. Again, Linux for the System z platform offers the flexibility to store keys securely using solutions that meet the enterprise's existing security policies. The secure-key solution supports CCA, PKCS#11 or Java JCE as viable key stores.

The introduction of the secure-key support has brought an opportunity to redesign and rename the device driver needed to access the cryptographic PCI card. The device drive previously known as /dev/z90crypt is now known as /dev/zcrypt.

Stepping Beyond Obfuscation

The SSL acceleration and asymmetric support provided by Linux on the System z platform have been widely used by Web servers and Web applications supporting industries, such as retail, banking and finance and ISPs. These industries have benefited from the high volume of secure communications and transactions that can flow through Internet-facing applications, such as Web and Java applications, running on Linux for the System z platform. A strong competitor, Linux on the System z platform has

grown to become a standard, protecting back-end z/OS applications and data. The flexibility of the APIs available to access the hardware cryptography has permitted many applications to build on this support, allowing them to mature into critical business elements required by many industries. With the new extension into secure-key technology and the diversity of these APIs as well, new applications have begun to develop in this exciting environment. The banking and finance industries will have the tools they need to centralize a new generation of distributed applications, implementing secure technologies like PIN verification, single sign-on and service-oriented architecture. The cryptographic support available to Linux on System z applications has grown past the obfuscation found in some distributed solutions, offering the diverse algorithms and access needed by mission-critical business applications protecting the enterprise from end to end. 



Peter Spera is a senior software engineer with IBM. He focuses on security for Linux on the System z platform, but he's also involved in other areas, such as system integrity and vulnerability reporting. Peter can be reached at spera@us.ibm.com.