Enterprise Security Solutions

Electronic Commerce Technical Brief

Cryptography Using Compaq *MultiPrime* Technology in a Parallel Processing Environment

By implementing a cryptosystem based on Compaq *MultiPrime* technology, users can experience increased performance over traditional RSA implementations while maintaining a prescribed level of security.



Contents

3	Introduction	8
4	The environment today	
5	Decryption and signature generation with the CRT	8
		8
5	CRT and two-prime RSA	
	cryptosystems	9
5	MultiPrime and the CRT	9
	implementation	
7	Motivation for using	
	<i>MultiPrime</i> in a parallel	11
	processing environment	
		14

8	Comparison of performance between <i>MultiPrime</i> technol- ogy and traditional RSA	
8	Description of the test system	
8	Description of the test software	
9	Discussion of results	
9	Performance improvement provided by Multiprime technology	
11	Security considerations of the MultiPrime cryptosystem	
14	Hardware implementation using MultiPrime technology	
15	Summary	

In a traditional RSA system, the public modulus of length L is generated by multiplying two prime numbers of approximately equal size. As L gets larger, the two processes involving the large private key exponent in this system become computationally intensive. These two operations—decryption and signature generation—have computational time that increases as the cube of the modulus, L³. The Chinese Remainder Theorem (CRT) is used to increase the performance of this traditional two-prime RSA system.

This paper advocates a CRT implementation using multiple primes with an equal number of exponentiators operating in parallel. Instead of a modulus of N = p * q, as in the traditional RSA system, the described *Compaq MultiPrime* technology–based cryptosystem uses a modulus where N is developed as three or more distinct prime numbers: $N = P_1 * P_2 * \dots P_{k-1} * P_k$, where k is an integer greater than 2. If p and q are 512-bit numbers, then N = p * q is 1024 bits. In a *MultiPrime* system, P_1, P_2, P_3 , and P_4 of 256 bits will also result in a 1024-bit modulus.

For a given modulus size, increasing the number of prime factors and using CRT with parallel exponentiators will increase performance. However, as the size of these prime factors decreases, the modulus can be factored by a small-factor algorithm—for example, the Elliptic Curve Method (ECM). On the other hand, increasing the modulus size through additional prime factors will increase the runtime of the Number Field Sieve (NFS). However, the maximum number of primes for a given modulus size is limited by the computational complexity tradeoffs between the NFS factoring method and the ECM. The *MultiPrime* cryptosystem described here offers significant performance advantages over the traditional RSA system along with the ability to maintain a prescribed level of security.

The environment today

There are several tradeoffs required with current RSA implementations, including an increasing need to identify ways to speed up the computation while retaining security.

> **For security reasons,** it is not unusual to find RSA systems proposed wherein the prime numbers p and q are on the order of 1024 bits long.¹ This makes the modulus (product of p and q) a number with a 2048-bit representation. Numbers of this size require enormous computer resources to perform the signature generation, encryption, and decryption processes because the time to do these operations increases as the cube of the modulus size. The encryption processing can be limited by using a small encryption exponent. The signature generation and decryption time can be speeded up using the CRT.

> There is a commercial need for longer and longer moduli due to incremental improvements in factoring techniques and ever-faster networks of computers being made available to break ciphertext. These large networks of computers will continue to be mobilized to attack the RSA system, resulting in the use of larger moduli.

This increasing need for security (larger moduli) in the RSA system is in conflict with the desire for higher and higher performance.² There is a requirement to identify ways to speed up the computation while retaining security.

This performance problem is also exacerbated as the volume of ciphertext messages requiring decryption or signature generation increases, such as can be found in e-commerce transactions using the Internet. For example, a financial institution might maintain an Internet site that could receive hundreds of enciphered and digitally signed transactions every second. The messages associated with each of these transactions must be processed and responded to in a timely manner. Securing these transactions with an RSA system using large primes to form the keys can impose severe limitations on the institution's ability to produce a timely response.

Several ideas have been introduced for increasing performance while retaining security. For example, Adi Shamir introduced "unbalanced RSA" for short messages.³

1. R. L. Rivest, A. Shamir, and L. Adelman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, 12(2):120–126, February 1978.

3. A. Shamir, "RSA for Paranoids," Cryptobytes 1.3 Autumn.

^{2.} RSA Factoring Challenge, RSA Laboratories' announced results for RSA-155 challenge on August 26, 1999.

Decryption and signature generation with the CRT

With *MultiPrime* technology, traditional RSA operations such as the decryption and signature generation process can be optimized by using more than two prime factors in a parallel processing environment.

There are two major opportunities for increasing the performance of the RSA cryptosystem. The first is to choose the public key exponent, e, to be small (\geq 3). Choosing e to be the Fermat number 65,537 is common. The second is to optimize the decryption operation by casting the single-decryption operation modulo N into a system of two linear equations modulo p and q, respectively, where N = p * q. If L is the length of N in bits, then decryption and signature generation have computational time that increases as the cube of the modulus, L³. *MultiPrime* technology focuses on optimizing the decryption process using CRT with more than two prime factors in a parallel processing environment.

CRT and two-prime RSA cryptosystems

In traditional RSA cryptosystems, p and q are prime factors of N. Using the CRT allows decryption to be carried out as two fast exponentiation operations. Furthermore, special-purpose hardware allows these operations to be conducted in parallel. Simply looking at decryption as two exponentiation operations done modulo p and q on a single cryptographic engine allows a 4x increase in performance. If the processing system is designed for parallel processing, then an 8x increase in performance is possible.

The computations for fast decryption using the CRT are illustrated in figure 1.

MultiPrime and the CRT implementation

The CRT implementation is easily extended to *MultiPrime* technology. For example, if $N = P_1 * P_2 * P_3$, then an easy visualization of a CRT system with two parallel exponentiators would involve breaking it into two problems. That is, consider one problem modulo P_1 corresponding to the left branch of the illustration in figure 1.





An overview of the two-prime RSA encryption and decryption processes

If Bob wants to receive an encrypted message from Alice, he takes the following steps:

- He chooses two large random primes, p and q, which are kept secret.
- → The product N = p * q is computed.
- An encryption exponent, e, is chosen. N and e become the "public key" of the RSA cryptosystem. This public key is conveyed to Alice.
- Bob computes the secret decryption exponent, d, which satisfies the relation, e * d = 1 mod ((p-1) * (q-1)).
- D and N become the secret key, which is retained by Bob and is not distributed to Alice or any other possible user.
- When Alice wants to send an encrypted message to Bob, she obtains Bob's public key and converts her plaintext message, M, into a ciphertext message, C, which is computed as C = M^e (mod N).
- Alice sends the ciphertext to Bob, who decrypts it using his secret key in the equation, M = C^d (mod N).

In this system, the public exponent typically is chosen as small ($e \ge 3$). This means that the private exponent, d, which is the multiplicative inverse of e mod ((p-1) * (q-1)), may be on the order of size of the modulus N. The result is that Alice has a "small" computation to encrypt the message while Bob has to execute a computationally intensive formula to compute the plaintext message, M. Because Bob can choose to retain the prime factors p and q, he can choose to implement the CRT to increase the performance of this computation.

Alice is not affected if Bob chooses more than two primes to form N. If Bob chooses N = p * q * r * s (the product of four primes), then Alice's computations do not change. Alice is not aware that Bob's modulus consists of more than two primes or whether he chooses to use the CRT. The second problem can be done modulo $P_2 * P_3$ because CRT only requires that P_1 be relatively prime to $P_2 * P_3$. This second problem corresponds to the right-hand side of figure 1. Clearly, the right-hand side of this modified procedure in figure 1 in turn can be broken again into two problems. The result is a recursive relation that is satisfied by three recursions—the number of recursions is equal to the number of primes.

Decryption of the ciphertext, C, using the relationship $M = C^d \pmod{N}$ is used to develop the following decryption subtasks:

 $C_{1} = C \mod P_{1}$ $C_{2} = C \mod P_{2}$ $C_{3} = C \mod P_{3}$ $d_{1} = d \mod (P_{1}-1)$ $d_{2} = d \mod (P_{2}-1)$ $d_{3} = d \mod (P_{3}-1)$ $M_{1} = C_{1}^{d1} \mod P_{1}$ $M_{2} = C_{2}^{d2} \mod P_{2}$ $M_{3} = C_{3}^{d3} \mod P_{3}$

The results of each subtask (M1, M2, and M3) can be combined to produce the plaintext, M, by a number of techniques. A recursive form of the CRT reconstruction is preferred.

The following is a recursive CRT algorithm for computing the plaintext message, $M = C^d \pmod{N}$ where N = p[1] * p[2] * ... p[m]:

Let q[1] = 1; q[i] = q[i-1] * p[i-1], for i = 2, 3, ..., m

Then, $u[i] = q[i]^{-1} \mod p[i]$, for i = 2, 3, ..., m $x[i] = d \mod (p[i]-1)$, for i = 1, 2, ..., m $g[i] = C \mod (p[i]-1)$, for i = 1, 2, ..., m $a[i] = g[i]^{x[i]} \mod p[i]$, for i = 1, 2, ..., m y[1] = a[1]; $y[i] = y[i-1] + q[i] * ((a[i] - y[i-1]) * u[i] \mod p[i])$, for i = 2, 3, ..., mM = y[m]

Motivation for using *MultiPrime* in a parallel processing environment

The efficiency in cryptographic processing using *MultiPrime* technology is enhanced further by implementing CRT techniques and by using parallelism to evenly distribute the load across multiple exponentiators simultaneously.

> **To compare** *MultiPrime* system performance, a baseline system must first be established. This is an RSA two-prime system in which CRT is used and exponentiation is accomplished on a single processor or exponentiator. The time to encrypt or decrypt increases as the cube of the modulus size, L³, where L is the length of the modulus. Thus, the performance of the twoprime case on a single processor without CRT is proportional to L³. If CRT is used on a single cryptographic processor, then the time is proportional to $(L/2)^3 + (L/2)^3 = L^3/4$.

> First consider *MultiPrime* executing on a single exponentiator. If k primes are used and CRT is performed on the single exponentiator, then the time is proportional to L^3/k^2 . Therefore, the speedup of this *MultiPrime* system relative to the baseline system (normal two-prime RSA system using CRT) is on the order of $k^2/4$. If three primes are used (k = 3), then the speedup of the system is about 9/4, or 2.25 times.

If four primes are used, the increased performance is about 4x. This is a theoretical limit and ignores overhead including the CRT reconstruction.

Now consider *MultiPrime* technology executing on multiple exponentiators. The number of exponentiators is equal to the number of prime factors so all the CRT operations are accomplished in parallel. The time to accomplish k exponentiations simultaneously on k exponentiators is L^3/k^3 . The theoretical speedup of a *MultiPrime* system with k exponentiators versus the baseline system using CRT on a single exponentiator is the ratio of $L^3/4$ to L^3/k^3 or $k^3/4$. For k = 3, the performance improvement will be a factor of 7. The performance enhancement is 16 for a *MultiPrime* system with four primes (k = 4).

Consider the following example of a 2048-bit modulus:

- → N1 is the product of two 1024-bit primes.
- → N2 is the product of four 512-bit primes.
- → N3 is the product of sixteen 128-bit primes.

The theoretical performance speedup as defined above for a *MultiPrime* system over a two-prime system is k³/4 so that N2 will run 16 times faster than N1. Similarly, N3 will run 1024 times faster than N1.

However, the security characteristics of these three systems are different because N3 may be susceptible to attack using "small factor" algorithms or techniques such as the ECM. The security of this example is discussed in the section "Security considerations of the *MultiPrime* cryptosystem."

Comparison of performance between *MultiPrime* technology and traditional RSA

This section discusses the actual performance data running traditional RSA and *MultiPrime* technology. Included are descriptions of the test system and test software as well as a discussion of the results using a Compaq *ProLiant* 6000 server and a Compaq AXL200 PCI Accelerator Card.

Description of the test system

All tests were performed on a *ProLiant* 6000 server with the following installed hardware:

- → 650 megabytes of RAM
- Four 200-megahertz Intel[®] Pentium[®] processors
- → 512 kilobytes of cache
- Microsoft[®] Windows[®] 2000 operating system

All test values were the average of 10 different computations each performed for 10 iterations, for a total of 100 averaged times. All starting data used in the tests was retained in order to repeat the exact same test at a later date using either different hardware or different test software. An AXL200 PCI Accelerator Card was the hardware used for the exponentiation acceleration. Since the maximum modulus size for this card is 1024 bits, the traditional RSA computation in the hardware test was limited to using a 2048-bit modulus (with CRT).

Description of the test software

The software application used to perform the *MultiPrime* proof-of-concept and to measure its performance was written in C using Microsoft Visual C++[®] development system, version 5.0, and compiled as a console application. Care was taken to ensure that the multi-threading aspects of the Windows operating system interfered as little as possible; for example, during tests, no data was scrolled to the console window or written to disk.

All key generation, prime searching, and biginteger computations used the RSA BSAFE 3.0 CMP math routines. The BSAFE prime.c file was altered to accommodate the *MultiPrime* technology techniques. The computational algorithm was to use the *MultiPrime* iterative recombination formula and iterate for two steps only when performing the RSA computation, and iterating it for three or more steps when performing the *MultiPrime* technology–based computation.

All timing data for the tests, as well as initial values, was stored in internal arrays within the program until it finished; then all of the data was written to a file. The timing results were formatted so that they were easy to input into Microsoft Excel for graphing.

Discussion of results

Figure 2 displays the composite results of many experiments driving the system. The figure shows the relative response time for four different cases:

- Traditional RSA (using two primes) in software
- MultiPrime technology in software using 256- and 512-bit primes, respectively
- Traditional RSA (using two primes) in hardware
- MultiPrime technology in hardware using 256- and 512-bit primes, respectively

All curves are plotted against the modulus length in bits. In the case of *MultiPrime* technology, the number of primes is also given for the 256- and 512-bit prime cases. The reference for the response time $(10^\circ = 1)$ is taken as the case for the traditional RSA in software with a modulus size of 512 bits.

The software test results were obtained by using a single host processor using multiple CRT passes depending upon the number of primes used. The hardware test results were obtained by using the AXL200 PCI Accelerator Card in conjunction with the host platform. The AXL200 PCI Accelerator Card with its multiple exponentiator chips uses parallel CRT for both traditional RSA and *MultiPrime* computations. When *MultiPrime* technology is employed, each exponentiator services a unique prime in parallel.

Performance improvement provided by *MultiPrime* technology

Figure 2 also demonstrates that *MultiPrime* technology is viable for both hardware and software. The figure shows that as you progress from the traditional RSA in software through *MultiPrime* technology in software, through traditional RSA in hardware, and, finally, through MultiPrime technology in hardware, the response curves continue to flatten out, in general, and become more horizontal. This indicates that the exponentiation time is being reduced consistent with the theoretical model discussed in the previous section, "Motivation for using MultiPrime in a parallel processing environment." The top curve (RSA in software) is very close to the theoretical L³ dependence. The curve for 512-bit MultiPrime technology in software (second from the top) is very close to the L³/K² relationship. For example, the speedup at a modulus of 2048 bits between RSA and 512-bit *MultiPrime* technology is $k^2/4 = 4$ which is very close to the measured values for the 2048-bit modulus case. Note also that the response time decreases as the value of the primes decreases.





The contrast in performance between RSA and MultiPrime technology in hardware is not as pronounced as in software, but nevertheless is still significant. The difference is due to the fact that RSA in hardware is already accelerated and each hardware expo chip has its own specific characteristics that tend to differ from the theoretical results. The performance curve of the VMS311 custom application-specific integrated circuit (ASIC) chip tends to be more linear than that of software. Notwithstanding, the performance improvement is still achieved, especially using larger moduli. Another characteristic of the MultiPrime curves is that they increase somewhat as a function of the modulus and as a function of the number of primes. Theoretically, these MultiPrime curves should be flat (horizontal) because for a constant modulus size, n number of primes are being executed in n number of expo chips in parallel. What is occurring is that as the number of primes becomes large, the recursive CRT algorithm (as described in the subsection, "MultiPrime technology and the CRT implementation") becomes a noticeable fraction of the overall *MultiPrime* calculation (overhead) and hence the slope of the MultiPrime curves increase. This is noticeable in both the hardware and software computations for large numbers of primes.

The curve for traditional RSA in hardware stops at the 2048-bit modulus point. This is due to the fact that the hardware chips have a limit of 1024bit operation and in traditional two-prime CRT cannot go beyond two 1024-bit primes providing a modulus of 2048 bits. Only *MultiPrime* technology provides the extensibility to proceed to higher moduli without having to change the hardware technology.

So far, this paper has presented the effects on exponentiation performance with respect to the size and number of primes without regard to the consideration of security, specifically to the factoring strength of the multiple primes. The next section considers the security limits of using *MultiPrime* technology.

Security considerations of the *MultiPrime* cryptosystem

Are the primes large enough in the *MultiPrime* system to attain the same security level provided by the traditional RSA two-prime system? This section compares ECM and NFS factoring techniques to identify a minimal, maximal, and optimal number of multiple primes to use.

> **Instead of a** modulus of N = p * q as in the traditional RSA system, the *MultiPrime* system provides a modulus where N is developed as three or more distinct prime numbers, N = P₁ * P₂ *P_{k-1} * P_k, where k is an integer greater than 2. If p and q are 512-bit numbers, then N = p * q is 1024 bits. In a *MultiPrime* system, P₁, P₂, P₃, and P₄ of 256 bits also results in a 1024-bit modulus.

> The number of primes, k, in the *MultiPrime* system could be expanded but at some point one would ask: Are the primes sufficiently large to attain the security one would expect of a 1024bit modulus using two primes? How do these additional but smaller primes affect the security of this system? What is the tradeoff being made between the enhanced performance and security in this system? This section answers these questions and establishes a definitive process for selecting the number of primes that attain a prescribed level of security.

> The Number Field Sieve (NFS) is the best-known factoring method and is unaffected by the number of primes in the target. The execution effort for NFS to factor a number N can be based on the time complexity function for NFS.^{4,5} The execution effort expressed in MIPS-Years is shown below.

No. of MIPS-Years =

$$1.5 * 10^{-5} \exp(1.923 \sqrt[3]{\log N} (\log \log N)^2)$$

The runtime of the NFS algorithm depends on the size of the product $N = P_1 * P_2 * ... P_{k-1} * P_k$ and is independent of the size of the individual primes. It is important to recognize that an NFS algorithm does not run faster if the primes become small.

Before proceeding, consider the previously discussed example of a 2048-bit modulus:

- ➤ N1 is the product of two 1024-bit primes.
- ▶ N2 is the product of four 512-bit primes.
- N3 is the product of sixteen 128-bit primes.

The time complexity of the NFS is the same for N1, N2, and N3. However, the impact of "smallfactor algorithms" such as the Elliptic Curve Method (ECM) must be considered. This class of algorithms has a runtime that depends on the size of the prime factors: The larger the primes, the greater the time complexity of this class of algorithms. In general, the largest factor found with ECM has been a 157-bit factor (47 digits). The 2048-bit modulus, N3, is the product of too many small primes and is susceptible to factoring by the ECM. The modulus N2 consisting of four 512-bit primes is more secure than N3. Some security criteria must be specified before the maximum number of primes can be established.

^{4.} R. Schroppel, private communication, August 1999.

^{5.} S. A. Vanstone, A. J. Menezes, and P. C. van Oerecht, *Handbook of Applied Cryptography*, CRC Press, New York, 1997.

The time for ECM to find a prime P as a factor of a modulus with D digits is shown below. ⁶

No. of MIPS-Years =

$$3 * 10^{-15} D^2 \exp(\sqrt{2 \log P \log \log P})$$

Figures 3, 4, and 5 show ECM and NFS work factors for 1024-, 2048-, and 4096-bit moduli.

The curves in figures 3 and 4 show a crossover point in which the computational work factor of the NFS and ECM factoring methods are equal. Suppose you choose this crossover point to establish the maximum number of primes to be used for a specific modulus size. This would be a conservative criterion with the result that the optimal number of primes is the maximum number such that the runtime of the ECM is equal to or greater than the NFS runtime. For details, see Table 1.



Figure 3. ECM and NFS work factors for 1024-bit modulus consisting of 2 to 15 prime factors.



Figure 4. ECM and NFS work factors for 2048-bit modulus consisting of 2 to 15 prime factors.





Figure 5. ECM and NFS work factors for 4096-bit modulus consisting of 2 to 16 prime factors.

Table 1. Optimal number of prime factors for a specific modulus size (security criteria is that runtime of ECM = NFS)

Modulus size in bits	Number of primes	Theoretical performance speedup over a two-prime CRT system
512	2	2 *
1024	3	6.7
1536	3	6.7
2048	3	6.7
2560	3	6.7
3072	3	6.7
3584	3	6.7
4096	4	16
8192	5	31

† Note that the baseline system configuration uses CRT and always uses one exponentiator (even in the case of two primes). The MultiPrime system always uses as many exponentiators as prime factors so that two exponentiators are used in parallel in the two-prime case.

Hardware implementation using *MultiPrime* technology

When combined with *MultiPrime* technology, specialized hardware allows for longer moduli in order to keep pace with rapidly improving factoring techniques and advanced networks of computers that are capable of breaking ciphertext.

The AXL200 PCI Accelerator Card is

designed to be a specialized security processor that can be added into virtually any generalpurpose server. It is designed to off-load compute-intensive security processing to free the host CPU to perform application tasks. The hardware design (see figure 6) is based upon two custom ASIC chips. The VMS310 ASIC chip is a secure 32 RISC ARM processor core that works in concert with four high-performance VMS



Figure 6. The Compaq AXL200 PCI Accelerator Card parallel architecture.

exponentiators. Each VMS311 chip provides for six efficient onboard 1024-bit exponentiators. These exponentiators perform all parallel processing and CRT tasks. All traffic via the communications bus between the VMS310 chip and the VMS311 chip is encrypted with the Triple-DES algorithm. When an application needs to perform a compute-intensive RSA operation such as a decryption of an encrypted message back into plaintext, the task is sent to the PCI card. The hardware is designed to optimize parallelism for both traditional RSA and MultiPrime system implementations. The RSA operation is simultaneously deconstructed and threaded evenly among the 24 onboard exponentiators. The task is then reconstructed using CRT techniques for a significant performance improvement over single-threaded RSA exponentiators.

Summary

MultiPrime technology adds a new dimension to the traditional securityversus-performance paradigm. It provides significant performance advantages while maintaining a prescribed level of security.

> **This paper advocates** the use of hardware parallelism, CRT, and an optimal number of primes as the basis for a new patented technology from Compaq known as *MultiPrime*. This new technology is designed to provide *increased performance while maintaining a prescribed level of security* with none of the tradeoffs that are required with current RSA implementations.

In effect, *MultiPrime* technology allows users to trade off the modulus size and number of primes to increase performance and maintain the prescribed security level. To contrast performance and security characteristics, RSA decryption processes that use two prime factors were used as a basis for comparison. This system typically uses a small exponent for encryption and a large private exponent for decryption. *MultiPrime* technology was applied in hardware as well as software to the decryption task to illustrate how it does not affect the decryption operation, yet delivers dramatic performance improvements.

A significant contrast was provided using parallelism, CRT, and multiple exponentiators processing a *MultiPrime* distributed task. An example of a 2048-bit modulus being reduced into eight 256-bit processes was provided to illustrate the speedup of a *MultiPrime* system. In this case, the performance increase was 40 times faster than with a system that computes the product of two 1024-bit primes. There are some security considerations in implementing this new technology. One key consideration is whether the primes are sufficiently large to achieve the security that one would attain using a modulus made up of two primes. To address this issue, a method was introduced that compares ECM and NFS factoring techniques to identify a minimal, maximal, and optimal number of multiple primes to use.

The paper also described a hardware-based system designed to take advantage of this new technology. This system is based on using highperformance exponentiators to perform *MultiPrime* tasks in parallel, thus speeding RSA operations. The specialized hardware in combination with *MultiPrime* technology also allows for longer moduli in order to keep pace with rapidly improving factoring techniques and advanced networks of computers that are capable of breaking ciphertext.

The described *MultiPrime* cryptosystem adds a new dimension to the traditional securityversus-performance paradigm, and provides significant performance advantages while maintaining a prescribed level of security.

For More Information wEBSITE: www.compaq.com

©2000 Compaq Computer Corporation. All rights reserved. January 2000. Compaq, ProLiant, VMS, and the Compaq logo, registered U.S. Patent and Trademark Office. MultiPrime is a trademark of Compaq Information Technologies Group, L.P. Intel and Pentium are registered trademarks of Intel Corporation. Microsoft, Visual C++, Windows, and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other product names mentioned herein may be trademarks and/or registered trademarks of their respective companies. Technical specifications and availability are subject to change without notice.

00-0022 Order number 11NT-0100A-WWEN

