TC digest

WINTER 1998

IBM'S QUARTERLY INFORMATION SOURCE FOR THE TECHNICAL COMMUNITY

WHAT'S INSIDE

Bulletproof SNA Internetworking

If your information systems environment is like most, many of your current business-critical applications still require SNA protocols. Early attempts to carry SNA traffic over multiprotocol router networks have been disappointing. However, with IBM's Advanced Peer-to-Peer Networking (APPN) and High Performance Routing (HPR) extensions to SNA, multiprotocol routers can handle SNA as easily as they do TCP/IP. This article covers some early "trial and error" approaches to SNA internetworking, then explains the APPN/HPR solution and its advantages.

SNA carries over 70% of Earth's financial transactions.
Paraphrase from Gartner Group Report

Is your organization running applications built around the Systems Network Architecture (SNA) protocol? Have you given up on reliably integrating your SNA traffic over your multiprotocol router network? For years now, router vendors have been promising to carry SNA traffic over multiprotocol router networks, only to deliver disappointing results – ranging from periods of unacceptable response times to intermittent SNA session failures.

A new approach to integrating SNA traffic with that of other protocols may well be the best solution for many companies to solidly handle their mission-critical SNA while still

continued on page

Demystifying Subnet Masks

BRUSHING UP ON YOUR TCP/IP BASICS
PART 2

In the Fall issue, the author covered "IPv4 Addressing." This article explains implementing IPv4 subnet masks, including methods for determining subnet mask results, such as command output, a Subnet Mask Conversion Chart, and mathematical formulas. Readers can reinforce their understanding with exercises posted on TCdigest OnLine. Though the author's examples are based on IBM's AIX V4 platform, most terminology and commands cited are common across TCP/IP implementations.

In the Fall issue, we reviewed the basics of IPv4 addressing. We identified three types of addresses involved in delivering information to an end station – a symbolic name *(hostname)* for end users' convenience, an *IP address* for the TCP/IP protocol, and a *MAC address* for delivery on the physical wire.

continued on page

PART I

Cryptography and SET — Safe Surfing?

Part I of this article describes and compares the Secure Socket Layer (SSL) protocol and the Secure Electronic Transaction (SET) protocol. The author covers secure electronic transaction essentials: authentication; hierarchies of trust; certificates; and nonrepudiation. Part II — "Cryptography and SET: Under the Hood" covers encryption, public/private keys, hashing, and digital signatures. Go straight to Part II.

You've been surfing the Web and found some really cool stuff you've just gotta have. You notice that with a few mouse clicks, the object of your heart's desire could be FedEx'ing its way to your home.

continued on page

SET: Safe Surfing? *continued from page 1*

But, are you really ready to send your credit card number over the Internet and run the risk of broadcasting it to every hacker from here to Timbuktu? At the end of the month, are you going to end up with a credit card bill equal to a small country's gross national product? Be afraid. Be very afraid. . . .

If the Internet is today's wild Western frontier, then where is the sheriff who will protect peaceful, law-abiding netizens from the hacker outlaws? What's going to make the world-wide Web safe for e-commerce? (Note: In IBM parlance, *e-commerce* refers to connecting organizations with their customers and business partners; *e-business* refers to solutions
that create new value for business via the Internet.)

But back to safe surfing: one emerging answer is the Secure Electronic Transaction (SET) protocol from Visa and MasterCard (with help from IBM, Microsoft, Terisa Systems, GTE, VeriSign, and SAIC). SET is designed to provide a mechanism for secure electronic payment by credit card over an otherwise very insecure public Internet.

This article first describes and compares two protocols: *Secure Socket Layer (SSL)*, which encrypts/decrypts HTTP data between client and server; and *SET*, an open standard for conducting secure bank card payments over the Inter-net. Next, the article deals with secure electronic transaction essentials– authentication, hierarchies of trust, certificates, and nonrepudiation.

SET uses cryptography to:

- ▶ Provide confidentiality of information
- ▶ Ensure payment integrity
- ▶ Authenticate both merchants and cardholders for credit card purchases via the Internet
- ▶ Make payment processing on the 'Net faster, safer, and more secure — with IBM's CommercePoint Payment software

Isn't SSL enough?

The SSL protocol, widely deployed today on the Internet, has helped create a basic level of security sufficient for some hearty souls to begin conducting business over the Web. SSL is implemented in most major Web browsers used by consumers, as well as in merchant server software, which supports the seller's virtual storefront in cyberspace. Hundreds of millions of dollars are already changing hands when cybershoppers enter their credit card numbers on Web pages secured with SSL technology.

In this context, SSL provides a secure "electronic pipe" between the consumer and the merchant for exchanging payment information. Data sent through this pipe is encrypted, so that no one other than these two parties will be able to read it. In other words, SSL can give us *confidential* communications.

Sounds good, right? But what else is needed?

Is that really you?

Let's say that you (the consumer) want to buy something from my electronic store. You use your Web browser to interact with my merchant server to select the items you want to buy, then you save your selections in an *electronic shopping cart*. Finally, the moment of truth has arrived and you need to send your credit card information to me to pay for your purchase. Your browser and my merchant server set up a secure SSL pipe with you at one end and me at the other. We have a secure means for exchanging data, but no way to verify each other's identity.

How do you (the consumer) know that I am who I claim to be? After all, anyone can put up a snazzy Web site and profess to be the XYZ Corp., but how do you really know that they are legit? How do you know that this isn't simply some hacker impersonating the XYZ Corp. and collecting credit card numbers for personal use?

For that matter, how do I (the merchant) know that you are who you say *you* are? How do I know that the credit card number you just sent really belongs to you and that you didn't pick it up from a receipt you found while rummaging through the garbage? The simple truth is, that while we may have a secure communications pipe, we have no way of knowing who we are dealing with.

What we need is a system of *credentials* that we can present to each other to establish that we are, in fact, who we claim to be. In other words, I need to *authenticate* your identity and you need to do the same for mine.

Plastic Credentials

When you write a paper check in a "bricks and mortar" store, the merchant may ask you to present your driver's license as proof of your identity. Although the merchant doesn't

www.internet.ibm.com/commercepoint/payment/whatisset.html

know you from Adam, he or she is willing to defer to the Department of Motor Vehicles (DMV) in your state as a trusted third party that can vouch for you. The assumption, of course, is that to get a driver's license from the DMV, you had to prove your identity to them with a birth certificate (or a similar legal document). The DMV issued you a plastic card identifying you (*e.g.,* name, address, personal photo), then signed the card with either a preprinted signature of some government official or the state seal (or both) to prove that this license really did come from the DMV.

With this system, the merchant doesn't have to know you in order to trust that you are who you claim to be. The DMV becomes the trusted third party, whose authoritative credentials "prove" that you really are you.

Digital Credentials

Great, but how do you verify someone's identity in cyberspace, where instead of plastic drivers' licenses, we deal with electrical impulses? The answer is, that we need to establish a similar system of trust, but instead of plastic IDs, we will use *digital certificates* containing the essential elements for authenticating your transactions. Since this digital certificate is comprised of a bunch of bits arranged in a standardized format, you can send it over the 'Net to any merchant who needs to know who you are.

First, you (the consumer) must prove your identity to the bank that issued the credit card to you. That bank will, in turn, vouch for you by putting its *digital signature* on your certificate. I (the merchant) will get my bank to digitally sign my certificate as well. But, since you and I are unlikely to use the same bank, we both need a trusted third party that can vouch for both banks. Since the credit card company (*e.g.,* Visa, MasterCard, etc.) is best positioned to do this, it becomes the common entity that you and I both trust.

You might think of this as a *hierarchy of trust* in which your bank trusts you, and therefore signs your certificate. The credit card company trusts your bank because of the business relationship they have established, and therefore signs your bank's certificate.

By the same token, my bank trusts me (the merchant) and signs my certificate. And the credit card company trusts my bank and signs my bank's certificate. Now we all have digital certificates that have been signed by a trusted third party whose signature is easily recognizable by all. We have established a system of trust, as shown in Figure 1. (Note: Figure 1 shows a simplified version of the hierarchy of trust. The actual hierarchy used in SET is somewhat more complicated.)

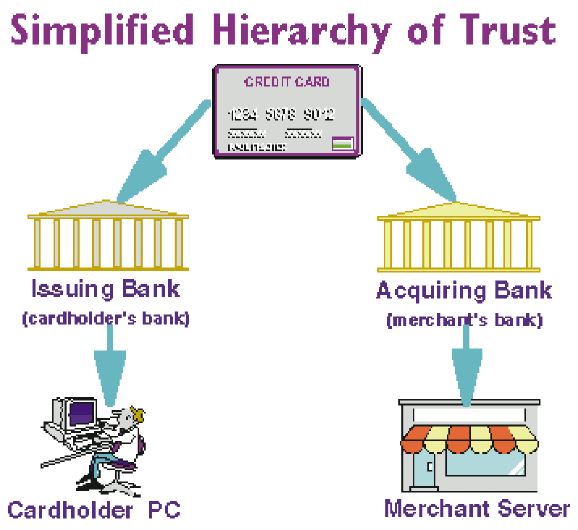

Figure 1. Simplified Hierarchy of Trust

Digital Ink?

How does one "sign" a digital certificate? Clearly, an ink pen won't work when we're dealing with bits of 1's and 0's. The short answer is that the banks and credit card companies must maintain specialized software called a *certificate authority* that creates these digital signatures, based on cryptographic techniques covered in Part II of this article.

At this point, the important thing to understand is that SET defines in great detail exactly what these digital certificates look like, the communication flows necessary to get them signed, the hierarchy of trust, and when and to whom you must present your certificate when making a purchase.

On the other hand, SSL, as it is generally deployed on the Internet today, does not require an analogous system of credentials and is, therefore, subject to attack by impostors posing as merchants, consumers, or banks.

Now some will say that SSL does, in fact, allow for digital certificates, but these certificates are optional and can't begin to match the robustness of the SET credentialing system. For example, *there is no single, internationally recognized hierarchy of trust for today's SSL certificates.*
A number of companies will issue a certificate to you which they have signed, but since they have failed to establish a common *root certificate* that applies to all SSL certificates (as SET has done), the result is a proliferation of certificate authority signatures that must

continued on page

SET: Safe Surfing? *continued from page 1*

be recognized by every consumer's browser, every merchant's server, and every bank's payment system. Also, since SSL certificates are not tied to a specific credit card account number, they really only serve to identify the *machine* of the parties involved, not their right to debit the account to complete the sale.

If you order five CDs from my store, then a hacker changes that "5" to "500" before it gets to me, then you're going to be one unhappy customer when your shipment and credit card statements arrive.

Another SET advantage is that merchant banks can determine whether or not the merchant will see the consumer's account number. Since SET has such a strong system of credentials, many in the credit card industry hope that this account number-hiding option will be used extensively. This would limit the number of people who know your account number, therefore resulting in less potential for fraud.

Even if both you (the consumer) and I (the merchant) are upstanding citizens who would never get involved in credit card fraud, remember that when I receive your account number, I will have to store it (at least temporarily) before passing it along to my bank to capture the funds. While your number is on my system, a hacker could break in and steal this number and . . . well, you know the way that story ends.

Finally, SET (unlike SSL) defines all the necessary protocols – not only for exchanging payment data between the consumer and merchant, but also for completing the picture by specifying how this data will be passed along to the merchant's bank. Once the bank receives the data, it simply uses its existing back-end systems to interface with both the credit card company and the consumer's bank to collect the funds. While it is possible to set up a similar system with SSL, the fact remains that no internationally recognized system exists today. This means that each merchant must independently deal with his or her bank. As you can imagine, the security implications of such a "free for all" aren't very comforting.

This is one reason why, in most countries, merchants must bear the majority of the financial risk for Internet-based transactions. This risk has, in fact, caused some merchants to simply be unwilling to enter cyberspace, thereby, limiting your choices for e-commerce shopping.

Download the actual SET specs from www.visa.com/set/ or from www.mastercard.com/set/

View the IBM Redbook, *Secure Electronic Transactions: WWW Credit Card Payment* (SG24-4978) at www.raleigh.ibm.com:80/cgi-bin/bookmgr/BOOKS/EZ30QD00/CCONTENTS

Read the SET White Paper at www.internet.ibm.com/commercepoint/payment/set-paper.html

Is that really what you sent me?

Both SET and SSL provide a degree of *confidentiality* (*i.e.,* only the intended recipient can read the payment information); and we just covered SET's mechanism for ensuring *authenticity* (*i.e.,* you are who you say you are), but how do I (the merchant) know that the payment message I received from you was not tampered with along the way? If you order five CDs from my store, then a hacker changes that "5" to "500" before it gets to me, then you're going to be one unhappy customer when your shipment and credit card statements arrive.

SET also provides a means for verifying *message integrity*, which helps to ensure that what you sent me is what I received. (Note: Part II explains the underlying mechanism for verifying message integrity.)

Buyer's Remorse

The merchant's ultimate question is, "How can I make sure that you (the consumer) won't later claim that you never agreed to the terms of the sale, and that you aren't going to pay?"

For banks and merchants to feel confident about e-commerce, they prefer a payment system that can establish an environment of *non-repudiation* (*i.e.,* neither you nor I can renege on the deal).

Through its system of digital certificates, SET provides the technological basis for a nonrepudiable transaction, but the bottom line here is that what does and does not constitute nonrepudiation is at the discretion of the business and legal communities involved. In other words, one country's courts may recognize digital certificates as legally binding while another may find this insufficient.

The point is, legally binding purchases potentially exist within a properly implemented SET infrastructure, but the courts (or other government authorities) ultimately have the final word on this debate.

For additional technical information, you can view the IBM Redbook *Secure Electronic Transactions: WWW Credit Card Payment* (SG24-4978) at www.raleigh.ibm.com:80/cgi-bin/bookmgr/BOOKS/EZ30QD00/CCONTENTS

PART II

Cryptography and SET — What's Under the Hood?

Part I of this article, "Cryptography and SET: Safe Surfing?" explained how SET uses cryptography to ensure confidentiality, message integrity, and to authenticate all parties involved in the transaction. Part II covers cryptography fundamentals; how asymmetric crypto differs from symmetric crypto; public and private keys; hashing; plus a concise industry overview called "Is It Safe Yet?"

We saw in Part I how SET provides a basis for confidentiality, authentication, message integrity, and nonrepudiation. Sounds pretty good, right? But how does it do all this? The answer is that these features all derive from special cryptographic techniques. Let's take a look at some fundamental elements ...

What is cryptography?

Simply stated, cryptography is *a method of secret writing*. If I want to send you a confidential message, I can use a computer to *encrypt* (scramble) the original plain text – based on a mathematical algorithm that you and I mutually agree upon. I feed my message into this algorithm and it encrypts my message, based upon a key that I have selected. The new, encrypted message is unreadable, so it can be sent over a public network such as the Internet without compromising the original message's contents.

When you receive the message, you decrypt it by feeding it, along with the appropriate key, through an algorithm that unscrambles the message and restores it to its original form.

Of course, no encryption scheme is completely immune to cracking (*i.e.,* unauthorized decryption). An encryption scheme's relative strength is determined by the underlying algorithm's mathematical characteristics and the key's size. In short, larger keys ensure greater security because they represent more possible combinations that a would-be cracker must try.

The trouble with this, though, is that many governments around the world aren't keen to the notion of drug smugglers, terrorists, and other criminals evading law enforcement using strong encryption to hide their communications. To prevent this, governments often strictly limit cryptographic key sizes (and, therefore, the cryptographic strength). For example, US law limits the strength of cryptographic technology that can be exported, while French law regulates the strength of cryptography that can be imported.

Negotiating this maze of regulations can be downright maddening. The good news is that the key sizes used by SET have been approved by most major countries around the world. Governments tend to look more favorably upon SET-based cryptography, because it was designed only for specific types of financial transactions, not for general-purpose messages.

Presumably, these encrypted messages will ultimately be stored as credit card transactions on existing bank databases and can, therefore, be subpoenaed by court order if necessary. And since few countries (and even fewer merchants) want to be locked out of opening new sales channels around the world through e-commerce, the prospects for widespread approval of SET crypto is virtually assured. Put simply – money talks.

Symmetric Cryptography

The most straightforward form of crypto used in SET is called *symmetric* (or secret key) cryptography. This scheme is called symmetric, because the same key both encrypts and decrypts the message; therefore, you have symmetry. Both sender and receiver have a "shared secret" – the symmetric key – that they both must know.

The most popular form of symmetric crypto, the one used in SET, is the Data Encryption Standard (DES). Invented by IBM, it ultimately became a US government standard in the late 1970s. Relatively speaking, DES is fast, safe, and reliable. It has withstood the test of time – an important criterion in choosing a security technology involving large amounts of money. Recently, however, the strength of DES's 56-bit key length has come under question. In one well publicized case, a single DES message was cracked in just over three months (using spare cycles on over 10,000 computers around the world). Some see this as proof positive that DES is no longer secure. When you consider, however, that a single SET purchase request contains data encrypted by not one, but by three different DES keys, then unless your credit limit is a few

SET: Under the Hood

orders of magnitude higher than mine, you can quickly see that while cracking a SET message may not be impossible, it is highly improbable and, more importantly, impractical.

In fact, a more significant problem with DES (or with any other symmetric key scheme) is the requirement that both sender and receiver know the shared secret. If I want to send you a message, I select a symmetric key, encrypt the message with the key, then send the encrypted message to you. To decrypt the message, you have to know what the key is. How can I give you this information? Off-line methods such as telephone, fax, or mail are too slow, cumbersome, and subject to their own set of attacks. But if I send you the key online, then what's to prevent a hacker from intercepting it?

I suppose that I could encrypt the secret key and then send it to you. But how will I send you the key to unlock the key? You can see the recursive nature of this problem gets out of control quickly unless I use a different encryption scheme – one that doesn't require me to send you my key. No, it's not impossible, read on ...

Asymmetric Cryptography

Given that the previous section dealt with something called symmetric cryptography, you can almost guess that the next section will deal with something called *asymmetric* cryptography – and you'd be right! As its name implies, asymmetric crypto is just the opposite of symmetric crypto, in that we use one key to encrypt and another, different key to decrypt. Figure 2 illustrates the main points of an asymmetric cryptography transaction.

In fact, if I want to send you a message that only you can read, when using this technique, you (not me) would need to compute two keys in advance. These keys would be mathematically related in such a way that anything encrypted with one can be decrypted only with the other and vice versa. Then you would arbitrarily designate one key to be your private key and the other to be your public key.

Your *private key*, as you would expect, would remain private. You would tell it to no one under any circumstances. It has been generated on your computer and should never leave your computer in order for it to truly remain private.

Your *public key*, on the other hand, would be published to anyone who wanted to communicate with you. In no way do you compromise your own security by telling anyone what your public key, is because there is effectively no way for someone to derive your private key from your public key. While the proof of this apparent mathematical paradox is beyond the scope of this article, Figure 2 contains a simplified example of how such a thing could be accomplished.

Since you are the only person in the world who knows your private key, you are also, therefore, the only person in the world who can decrypt a message encrypted with your public key. So if I want to send a private message for your eyes only, I can use asymmetric encryption and your public key to encrypt the message, send it over the public network, and you, and only you, can decrypt the message because you are the only one who knows your private key, which is the only key that can decrypt the message.

Conversely, you can turn the whole process around, and I can know that a message did, indeed, come from you if you have encrypted it with your private key and I can decrypt it with your public key.

In fact, you could take a single message and encrypt it with both your private and my public keys (*i.e.,* two separate encryptions of the same message); I would decrypt it with my private and your public keys, and we could *authenticate* both the sender and the receiver of the message.

Here's the Rub ...

So now you might ask, "Why not simply use asymmetric encryption all the time, since it doesn't have the problem that symmetric schemes do with distributing secret keys?" The reason is simple – asymmetric crypto is about 10 to 1000 times more compute intensive than symmetric crypto. In fact, some have suggested that the form of asymmetric crypto used in SET, which was named RSA (Rivest, Shamir, and Adleman - its inventors), might just as easily stand for "Really Slow Algorithm." Clearly, if your message is of any substantial size, you would limit the amount of asymmetric crypto that must take place unless you have loads of patience and free CPU cycles.

SET strikes a balance between the pros and cons of DES and RSA by using DES to encrypt the bulk of the message payload and RSA to distribute the DES keys, which are only 56 bits long.

Public Key Distribution

For RSA (or any other asymmetric crypto scheme) to work, however, we need to have a means for telling the world what our public keys are. Since offline methods such as mail or phone are impractical, you might suggest e-mail or some other Internet-based method. The trouble with e-mail, though, is that you can't necessarily be sure who it came from. An impostor could send you his or her public key and tell you it was mine. Now we're back to that authenticity issue again.

SET uses the *digital certificates* discussed in Part I to distribute public keys. In other words, your public key is included in your certificate. When you are ready to buy something from me, you send me your certificate (which I know is really yours since it is digitally signed by a trusted third party), and then use the public key that it contains.

By the way, as part of these payment flows, SET also requires that I (the merchant) send you my certificate, so that you can authenticate me and be able to encrypt messages using my public key so that only I can read them.

Signing in Cyberspace

What really constitutes a digital signature, and how does your browser and my merchant server recognize it? Let's back up a bit. When you want to send a payment message to me that only I can read, you encrypt the bulk of the message using a randomly selected DES key. That helps ensure *confidentiality*.

You also want to ensure that the message that I get is the same one you sent (*i.e.,* message integrity) so you also run a *hashing function* (similar to a `CHECKSUM`), which calculates a relatively unique *message digest*. This digest is, as its name implies, a summary of the full message. A good hashing function would yield a significantly different digest value even if the original message had only been changed slightly. Also, it would not allow a hacker to determine the original message based solely on the digest value. In other words, it's an irreversible, one-way function.

After calculating the payment message's digest, you then encrypt this digest, along with the DES key, using your private key. In SET, this is called your *digital signature*. You then send this digitally signed message and your certificate to me. When I receive the message, I run the same hashing function that you ran so that I can determine the message digest myself. Then I decrypt your signature using your public key (which I obtained from your certificate) and see if the digest value that you calculated on the sending side matches the one that I calculated on the receiving side. If they match, then I can believe to a reasonable

ASYMMETRIC CRYPTO EXAMPLE; OR, HOW DO THEY DO THAT?

Let's say that the "secret" message that I want to send you is my unlisted phone number, which is, for the sake of this example, `848-9033`. You calculate a pair of asymmetric keys which have a special complementary mathematical relationship. For example, you arbitrarily select one key to be `1234567` (trivial, but it will work for this illustration).

The encrypt/decrypt function is based on *modulo 10 arithmetic* (which sounds a lot harder than it really is). This simply means that when you add two digits, you divide the sum by 10 and keep only the remainder. Here's an example:

`5+7=12` and `12/10=1` with a remainder of `2`

therefore, in mod 10 arithmetic: `5+7=2`

Here's how I would encrypt the secret message using this trivial algorithm:

`8489033`	the secret message I want to send to you
`+ 1234567`	your public key
`9613590`	the encrypted message *(simply the sum in mod 10 arithmetic)*

Here's how you would decrypt the secret message:

`9613590`	the encrypted message
`+ 9876543`	your private key
`8489033`	the original secret message *(the sum in mod 10 arithmetic)*

Why does this work? You know that adding `0` to anything simply yields that same thing, right? Well, in mod 10 arithmetic, it turns out that adding your public and private keys together is essentially nothing more than adding `0` to the original message, because these keys were carefully chosen to be `10`'s complements of each other. See for yourself:

`1234567`	your public key
`+ 9876543`	your private key
`0000000`	the sum in mod 10 arithmetic

Of course, the actual algorithm used in SET is more complicated than this, but you get the idea.

Figure 2. Example of Asymmetric Crypto

SET: Under the Hood

degree of certainty that the message has not been modified along the way. Figure 3 shows an example of just such a transaction.

Another benefit of this scheme is that I have also established that this message did, indeed, come from you because it was signed using your private key, which only you know. This, then, could be the basis for *nonrepudiation*, allowing me to hold you to the terms and conditions of the purchase. As stated earlier, though, nonrepudiation is ultimately a legal condition – not a technical one – but a technological basis does, in fact, exist with SET.

Is it safe yet?

Today's Internet-based payment mechanisms based on SSL (discussed in Part I) are roughly as secure as existing mail order/ telephone order (MOTO) credit card transactions. Clearly, many business are comfortable with this level of risk and are, therefore, moving ahead and deploying e-commerce.

SET, however, can potentially provide an even more robust e-payment infrastructure, resulting in lower fraud rates. The problem? SET, a new technology, is not yet widely deployed. Take heart, though, because at this very moment IBM is running SET pilots all around the world. In fact, we worked with PBS, a Danish Payment System company, to complete the world's first end-to-end SET transaction. Since that time, we've also worked with Fujibank in Japan to extend SET to include debit card transactions. In addition to these and others in Europe and Asia, we are currently running pilots in North America (in both the US and Canada), South America, and Africa, so, hopefully, you will be seeing more about SET from your bank in the not too distant future.

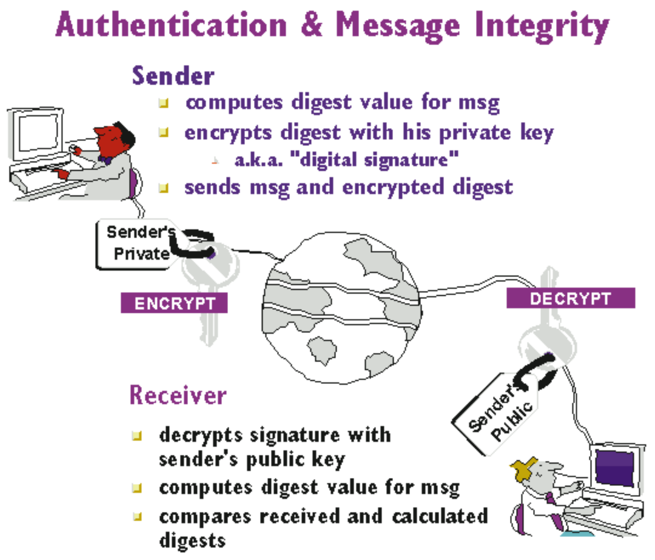
Authentication & Message Integrity
Sender
computes digest value for msg
encrypts digest with his private key
a.k.a. "digital signature"
sends msg and encrypted digest
Sender's Private
ENCRYPT
DECRYPT
Receiver
decrypts signature with sender's public key
computes digest value for msg
compares received and calculated digests
Sender's Public

Figure 3. Signing in Cyberspace

Summary

This article only scratches the surface of SET and its cryptographic basis. Many of the actual mechanisms that the protocol employs have been simplified here to demonstrate the underlying concepts. If you'd like to learn more about SET and IBM's products that implement it, take a look at the IBM Redbook, *Secure Electronic Transactions: WWW Credit Card Payment* (SG24-4978) at www.raleigh.ibm.com:80/cgi-bin/bookmgr/BOOKS/EZ30QD00/CCONTENTS

You can download the actual SET specs from www.visa.com/set or www.mastercard.com/set.

by Jeff Crume
Consulting Internet Specialist, IBM Advanced Technical Support
crume@us.ibm.com