Buster Buys a Comic Book: An Illustrated PKI Story

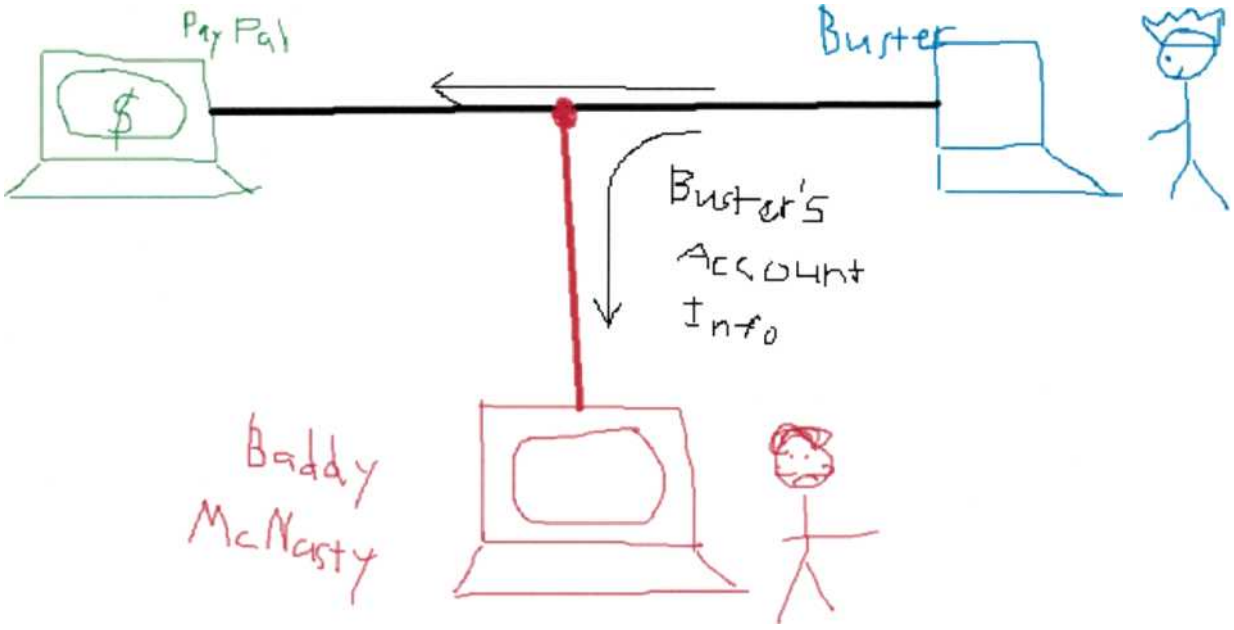
The Set-up

This is Buster's lucky day. While surfing through eBay, he found a collection of vintage 1940's horror comic books in mint condition. He's been looking for some of these for quite some time.

In the meantime, Baddy McNasty wants to update the clothes for his Barbie & Ken collections (They are just *so* last year's fashion!), but he doesn't want to pay for them himself! He wants to use someone else's money to pay for them, and poor Buster just happens to be online right now. What will he do?

The First Try - No Encryption

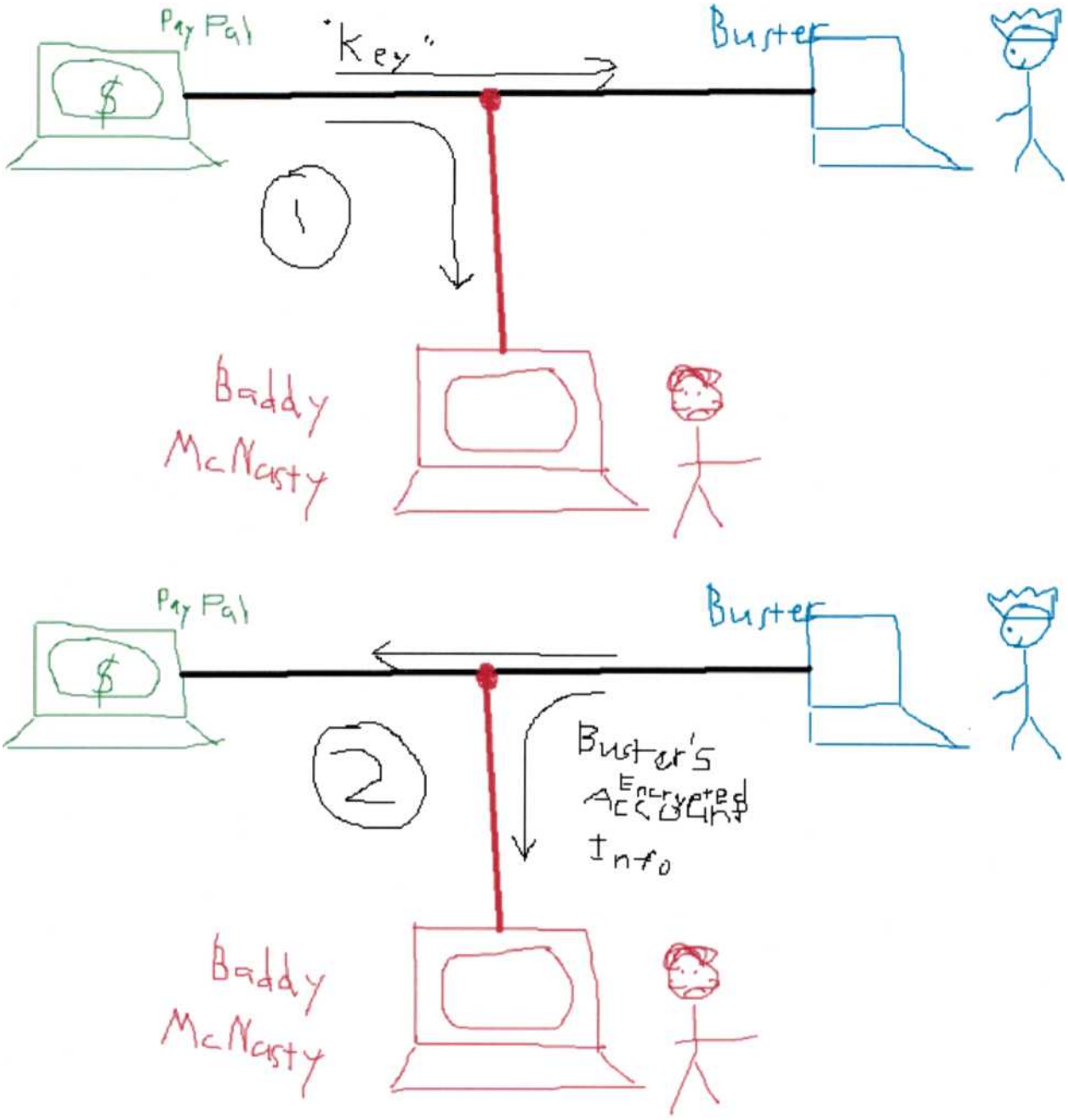
Buster goes to PayPal's server to enter his account credentials, but this was before encryption on the internet was common. Back then, anyone whose network connection was between Buster and PayPal could read everything that was sent between them.[1] Even Baddy McNasty could read it! That's just not good!



[1] I know that encryption existed before PayPal. Don't get so hung up on the details that you miss the bigger point!

Next, Symmetric Two-Way Encryption

Some clever person had the idea that if we scramble Buster's data up on one end ("encrypt" it) and descramble it on the other end, then people in the middle like Baddy won't be able to read it. So that person made a "key" (kind of like a digital version of a Captain Marvel decoder wheel) that could be used to encrypt and decrypt the message. PayPal then sent the key to Buster's computer (1) which then encrypted Buster's account information to send back to PayPal (2).



That's when everyone realized the fatal flaw in the plan: How could PayPal send the key used to encrypt and decrypt Buster's data so that Buster's computer could read and use the key but Baddy's computer couldn't? If Baddy got the key, then he would be able to decrypt Buster's account data too!

Next Step: Asymmetric Public/Private Key Encryption

At this point, something remarkable happened! A magic PiKsI came to both PayPal and Buster and gave them each two keys. One, the PiKsI said, was a *public* key that needed to be given to anyone and everyone. The other was a *private* key that should be kept to yourself and never given away. She then explained that anything that was encrypted with one key could *only* be decrypted with the other.

Encryption

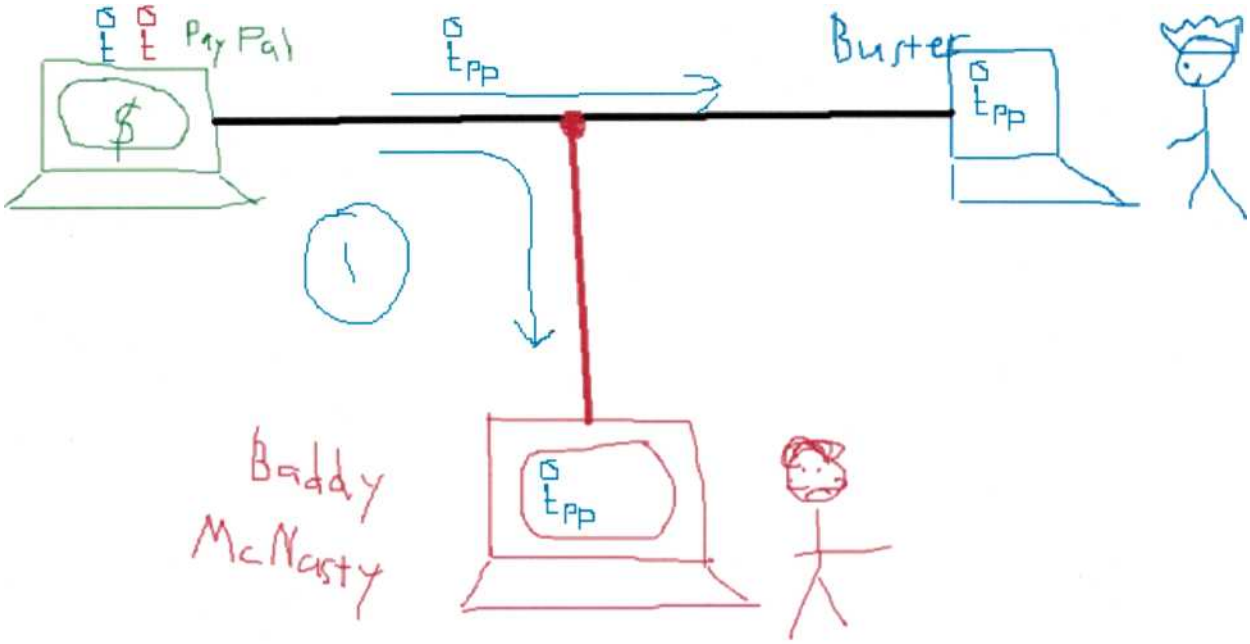
The PiKsI told Buster, "If you want to send PayPal a message that only PayPal can read, get PayPal's *public* key from them, encrypt your message with it, and then send it to PayPal. Only PayPal's *private* key can decrypt it."

Signing

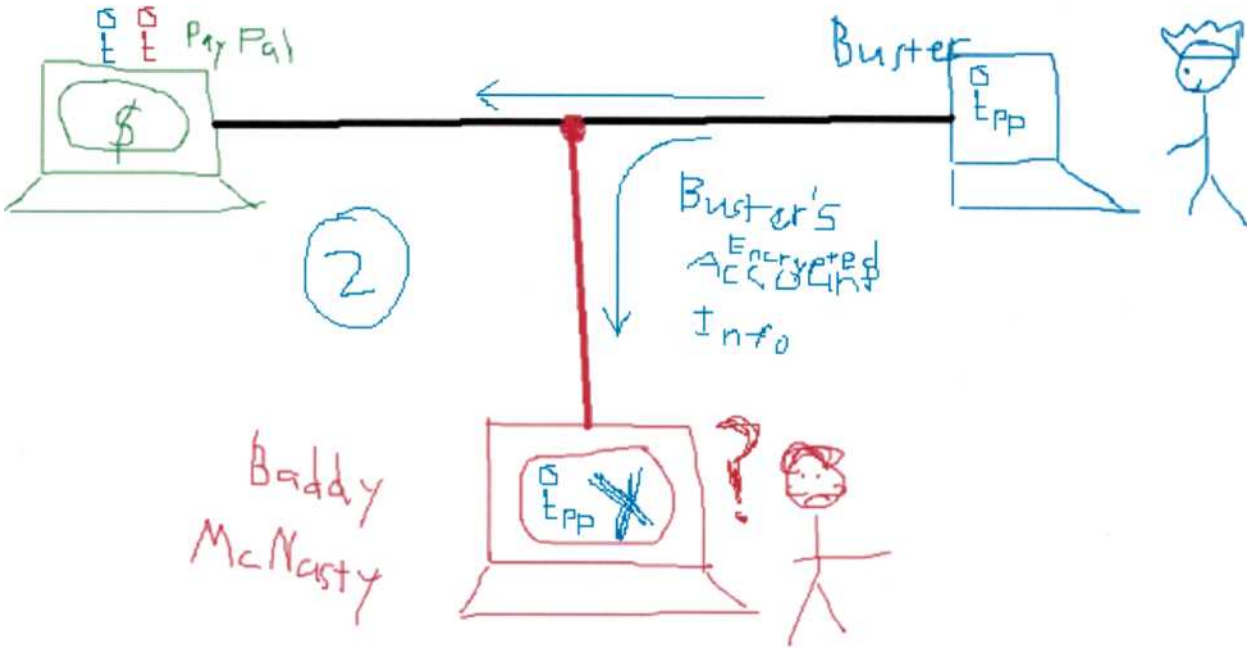
The PiKsI then turned to PayPal. "If you want to send Buster a message that he'll know without a doubt came from you, encrypt your message with your *private* key and send it. Anyone with your *public* key will be able to decrypt the message, but they will know it came from you because only you have the *private* key that was used to encrypt it."

Happy Ending?

So PayPal sent Buster its *public* key which Baddy got a copy of too.



Buster's computer then encrypted his account information with PayPal's *public* key and sent it to PayPal. Baddy got the encrypted data too, but he couldn't decrypt it because only PayPal's *private* key could decrypt it, and Baddy didn't have that!

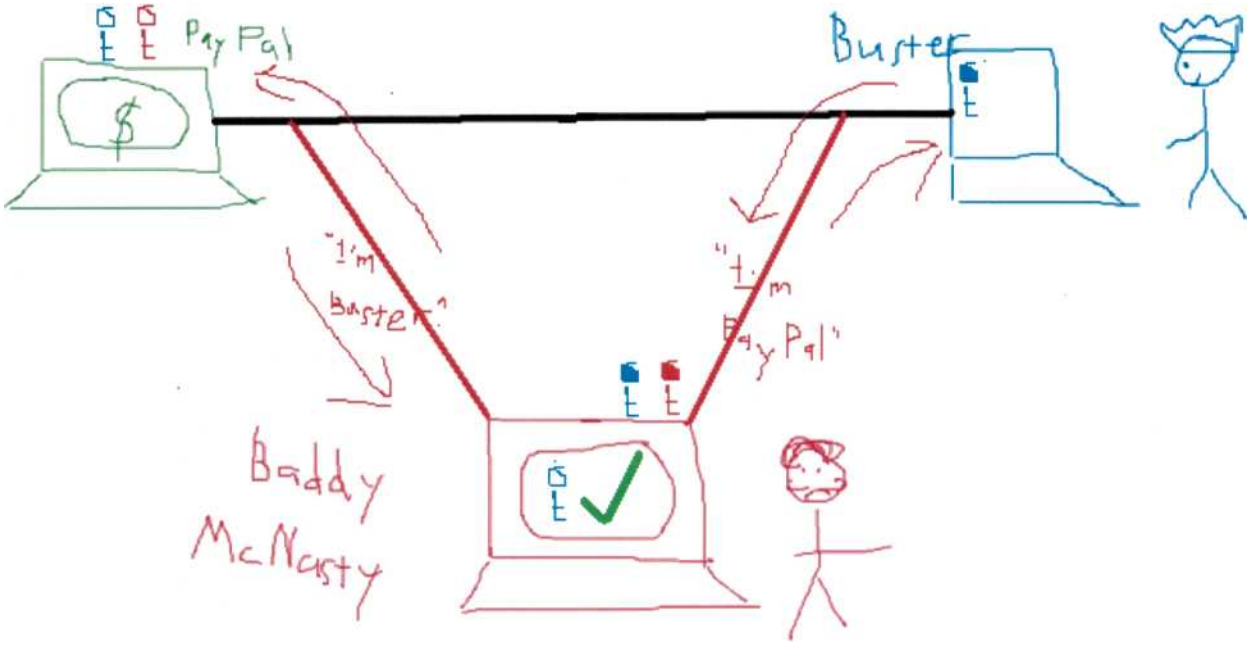


Things seemed pretty good for a while until one day...

The Revenge of McNasty - The Man in the Middle

For all of his faults, however, Baddy McNasty was clever. It was just a matter of time before he figured out a way to get Buster's PayPal account information.

Baddy made his own set *of* *public* and *private* keys. They weren't the same as PayPal's keys, but that didn't matter for what Baddy wanted to do. Baddy was going to get Buster's computer to think his computer was PayPal's server so when Buster's computer asks for PayPal's *public* key, Baddy's computer would send its own *public* key. Baddy's computer would decrypt Buster's data with Baddy's *private* key, save it, and then encrypt it with PayPal's *public* key to send to PayPal pretending to be Buster's computer. When PayPal' server would send a reply, Baddy's computer would do the whole process over in reverse. Buster's computer (and Buster) would never know they were talking to Baddy's computer and not PayPal's server.



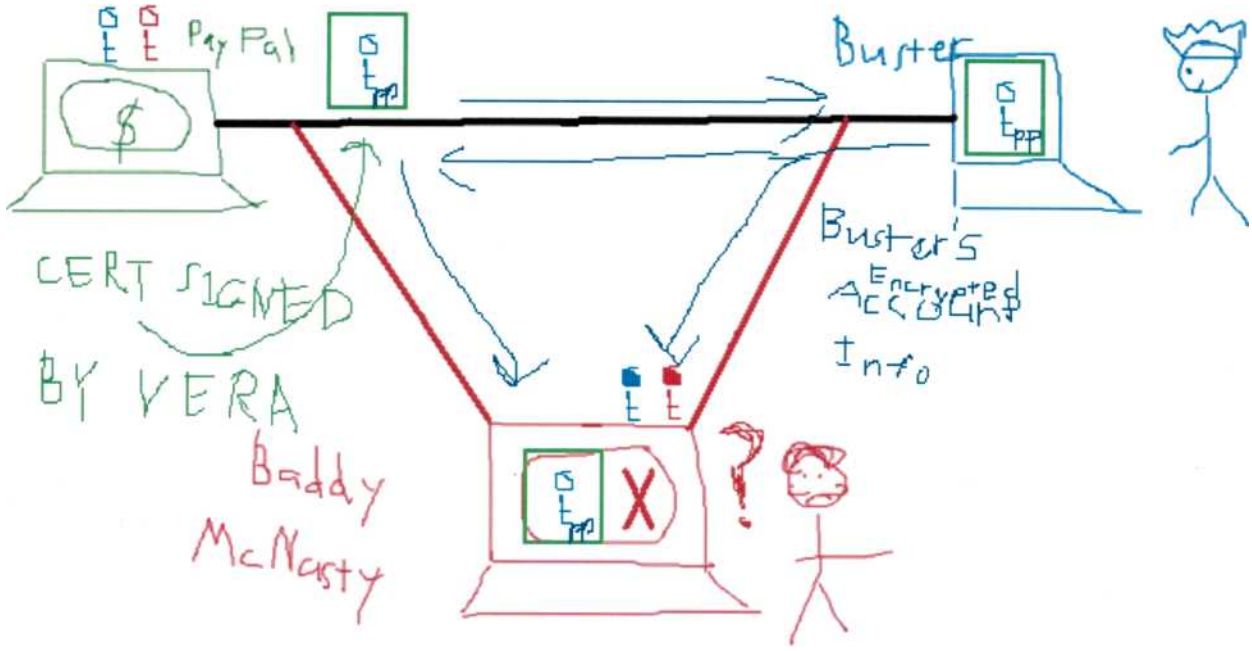
PiKsI to the Rescue Again

When the PiKsI saw this, she just couldn't bear to watch it without doing something, so she called her friend Vera Sine[2] and asked her to come meet Buster and PayPal.

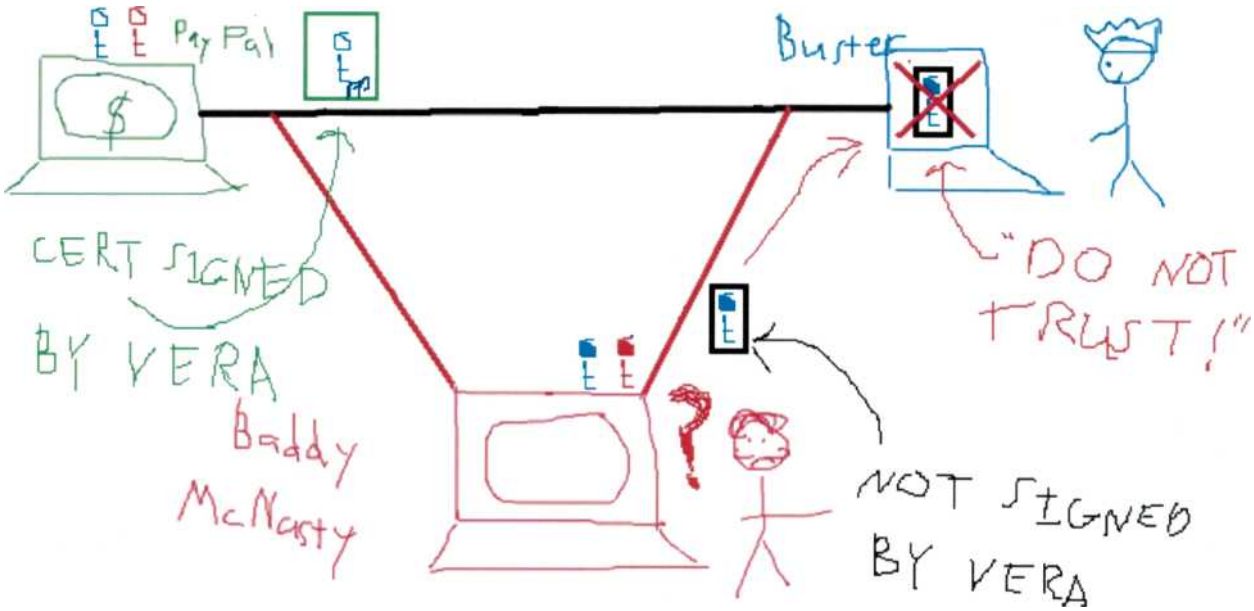
"Vera is going to help deal with that nasty McNasty once and for all. She is someone I *TRUST* and you can *TRUST* her too. She is going to make sure that a server on the internet is exactly who it says it is.
She has her own set of *public* and *private* keys that she can use to encrypt and sign. PayPal, if you will give Vera a copy of your *public* key along with some information about you and the server (like physical location, IP address, domain name, etc.), Vera will make a *certificate* out of it and sign it with her *private* key.

[2] "VeriSign", get it?

"Buster, when your computer asks PayPal for its *public* key, PayPal will send you their *certificate.* Your computer will then use Vera's *public* key to see if she signed it. If she did, then you can *TRUST that* she made sure that the server your computer is talking to is PayPal's server and no one else's. "



Once again, Buster could buy comic books online without fear of Baddy McNasty stealing his PayPal information. When Baddy tried to make his own *certificate,* Buster's computer would see that it wasn't signed by Vera and told Buster not to *TRUST* it!



The End?

Then one day, Baddy McNasty bought a small box from Packet Forensics[3] that used forged certificates that looked like they were signed by Vera to do the same Man-in-the-Middle attack that he did before Vera got involved.

"Well," thought PiKsI, "maybe self-signed certificates would be a better way to go...."[4]

[3] "Law Enforcement Appliance Subverts SSL", http://www.wired.com/threatlevel/2010/03/packet-forensics/

[4] "Are self-signed certificates safer?", http://blogs.techrepublic.com.com/security/?p=3388