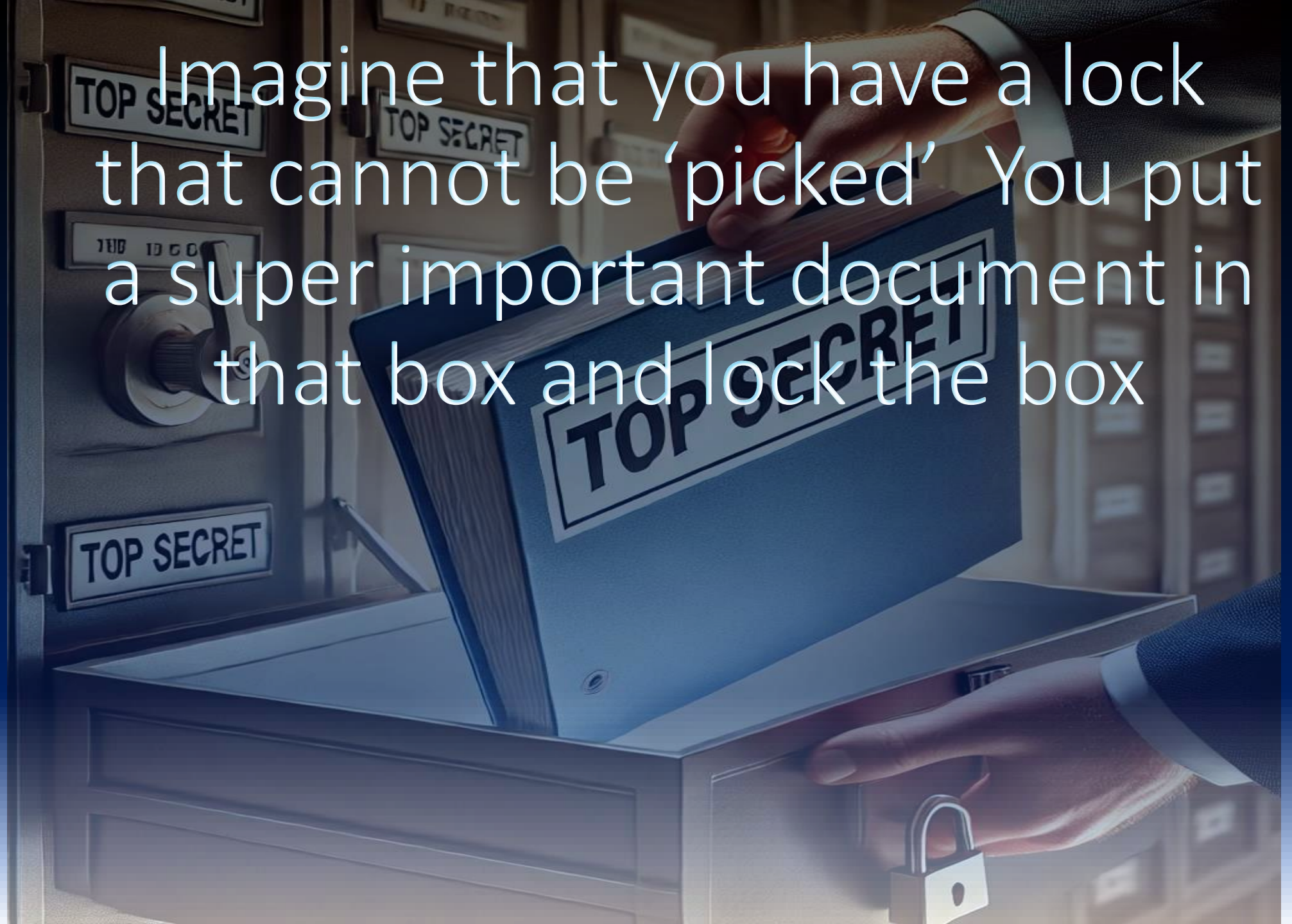


Imagine that you have a lock that cannot be ‘picked’ You put a super important document in that box and lock the box

Now you lock that box with your super strong lock and send it across the country

Uh oh, bad news. You have the only key to the lock. Your document is safe but the person you sent the box to can’t open it.

That is essentially the problem faced with modern encryption:

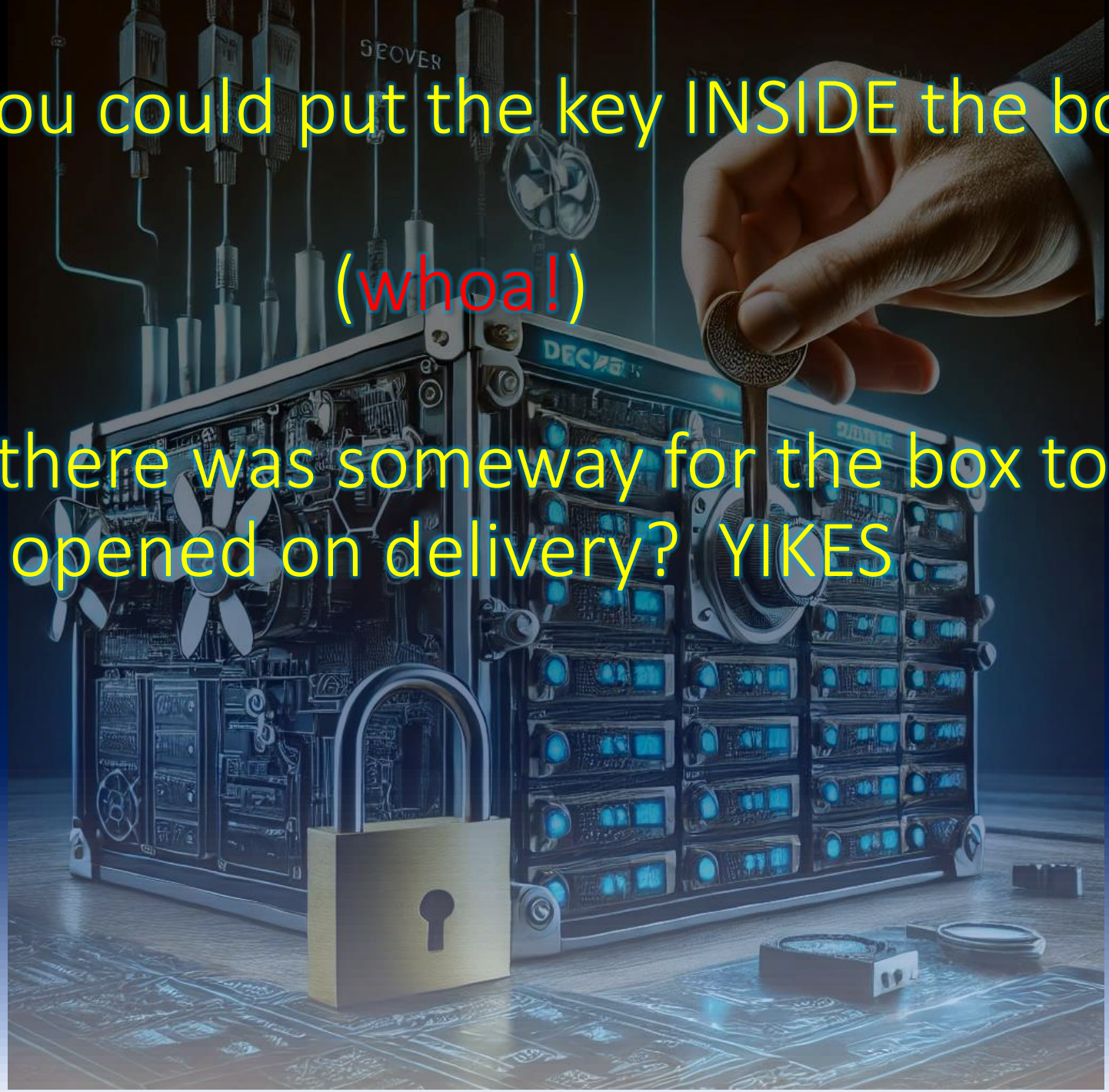
If you make a super-duper-strong key that is awesome……

…but it doesn't help you if the person you're sending your 'package' to doesn't have a copy of the key. Afterall, just how secure is it if you have multiple copies of your super duper strong key floating around?

What if you could put the key INSIDE the box?

(whoa!)

And that there was someway for the box to be opened on delivery? YIKES



That’s exactly what happens in modern encryption. Your browser creates a temporary super-strong key.

The bank already has a super-strong public key.

You’ll use the bank’s super-strong key to encrypt your key and send your super-strong key to the bank

Let’s review what happens in a conversation between your browser and your bank’s server:

Your browser initiates a ‘conversation’ with the bank’s server….

- **Your browser checks to see if the bank’s certificate is valid.**
  - **It is valid so the conversation continues.**
  - **Your browser retrieves the bank’s public key.**

- **Remember, the bank has a private key that is not shared.**

Your browser receives the bank's public key.

Your browser generates a 'session' key

Your browser uses the bank's public key to encrypt your session key using RSA encryption (NASTY 1-way math function)

Ths is an example of "asymmetric encryption" since 2 keys are used)

Your browser is now ready to send your encrypted session key to the bank

That common key is now sent back to the bank all safe and snug and encrypted!

- Remember, there is a (currently) unbreakable connection between the bank's private key and its public key.
- Because your session key was encrypted with the bank's public key, the bank's computer can use the relationship between its public and private keys to decrypt your session key. No other server on the planet can do that.
- Now both you and your bank have a common key which is your session key.
- From now on the bank uses RSA to encrypt all further communication using your super secret session key (symmetric encryption)

Since you and the bank both have a copy of your super secret session key. All further back and for the encryption of your data layers in your transport packets are encrypted using RSA to encrypt using your session key.

Once your session with the bank is ended, your session key is discarded so it is never used again to communicate with a secure server!
Pretty slick huh?