Algorithms, cryptography and protocols

DON'T EVER ROLL YOUR OWN PROTOCOL, CRYPTO ALGO, CRYPTO IMPLEMENTATION, OR CRYPTO RNG

ALSO, KEY MANAGEMENT IS VERY VERY HARD

August 2019

Security.ac.nz

Who?

Kate Pearce - Head of Security at Trade Me (@secvalve)

I work to ensure that the data Trade Me holds for our customers, and the services it provides them, are trusted, trustworthy, and trusty (resilient).

Trade Me

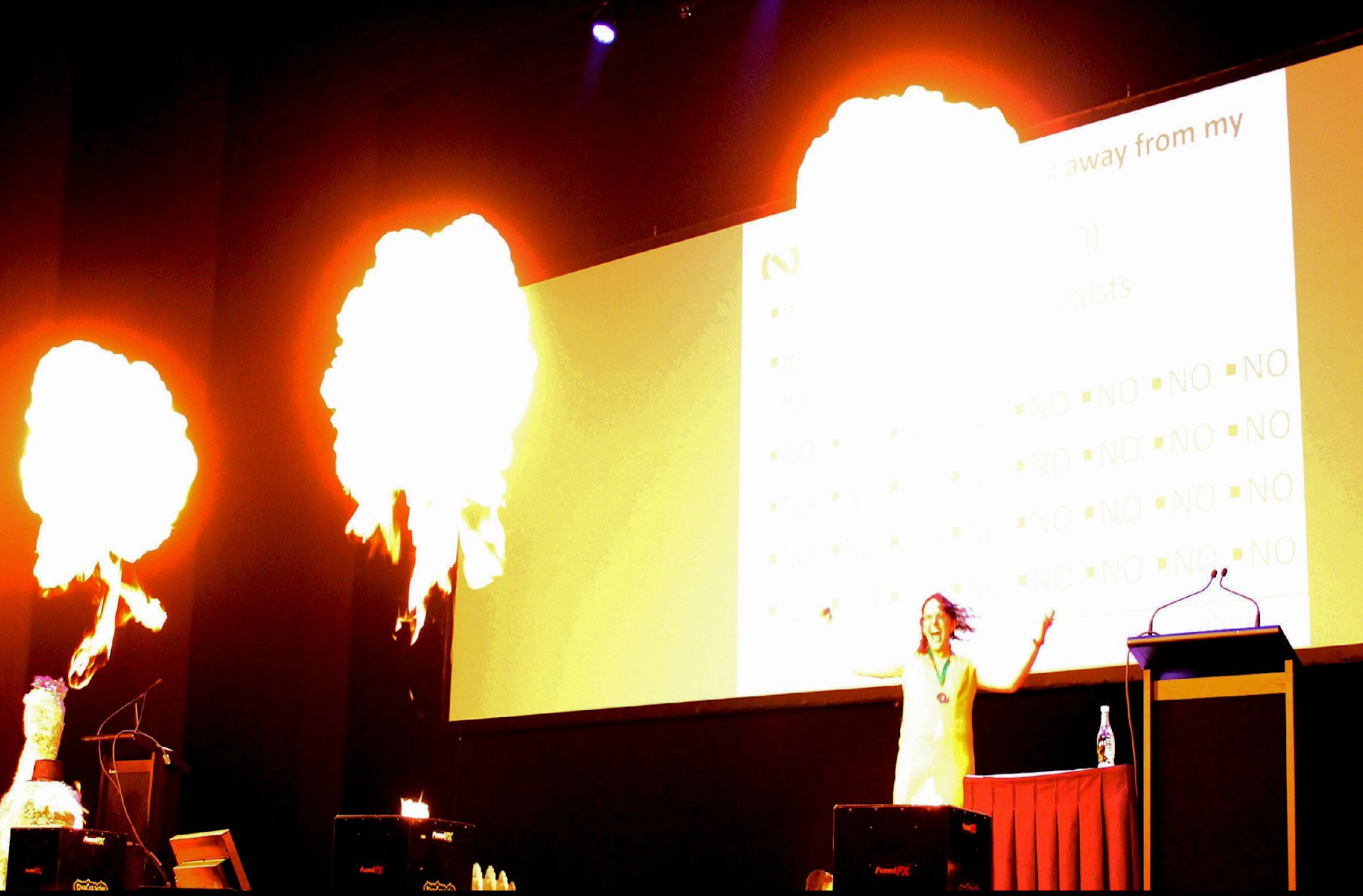
Trade Me and its systems are incredibly prevalent in New Zealand:

- Marketplace (Auctions, listing goods new & secondhand)
- Motors (New and used car listings)
- Property (Rental, Purchase, & Commercial)
- Jobs (Job Listings)
- Payments (Credit Card Processor)
- Holiday Houses
- Dating

Trade Me has unparallelled Brand Presence in New Zealand, and the vast majority of New Zealand's adult population in our systems.

Multiple millions of accounts in a country of 4.8 Million (~around 1M under age 18)

> 2 Million Daily interactions



CCSA, https://m.flikr.com/#/photos/4nitsirk

1. **Principles & Goals**
2. **Building Blocks**
3. **Protocols**

tldr;

DO Use Public Algorithms	DO NOT Roll-your-own Algo/Function	CONCENTRATE ON Key Distribution
DO Use Public Protocols	DO NOT Roll-your-own Protocol	CONCENTRATE ON Key Management
DO Use Secure PRNG for Keys	DO NOT Roll-your-own PRNG OR Use a non-secure PRNG	
DO Use a Secure Implementation	DO NOT Implement your own	
DO Use Recommended Cipher Suites	DO NOT Use Bad, Weak, or Null Suites	
DO Use Slow Algorithms and Salt Secret Hashes	DO NOT Hash Secrets with simple or fast hashes	

This Presentation

- **Is aiming at the key things people make mistakes with**
- **Is not going deep into details**
  - Will not tell you which tech or configuration to use
- **May have errors because cryptography is hard to do well**

Principles & Goals

Protocols - 3 way handshake

Principles - 3 Way handshake

Hello, shall we talk?

Sure, still good to talk?

Yep! ….*starts talking*

talking intensifies

Principles - 3 Way handshake

SYN

SYN ACK

ACK

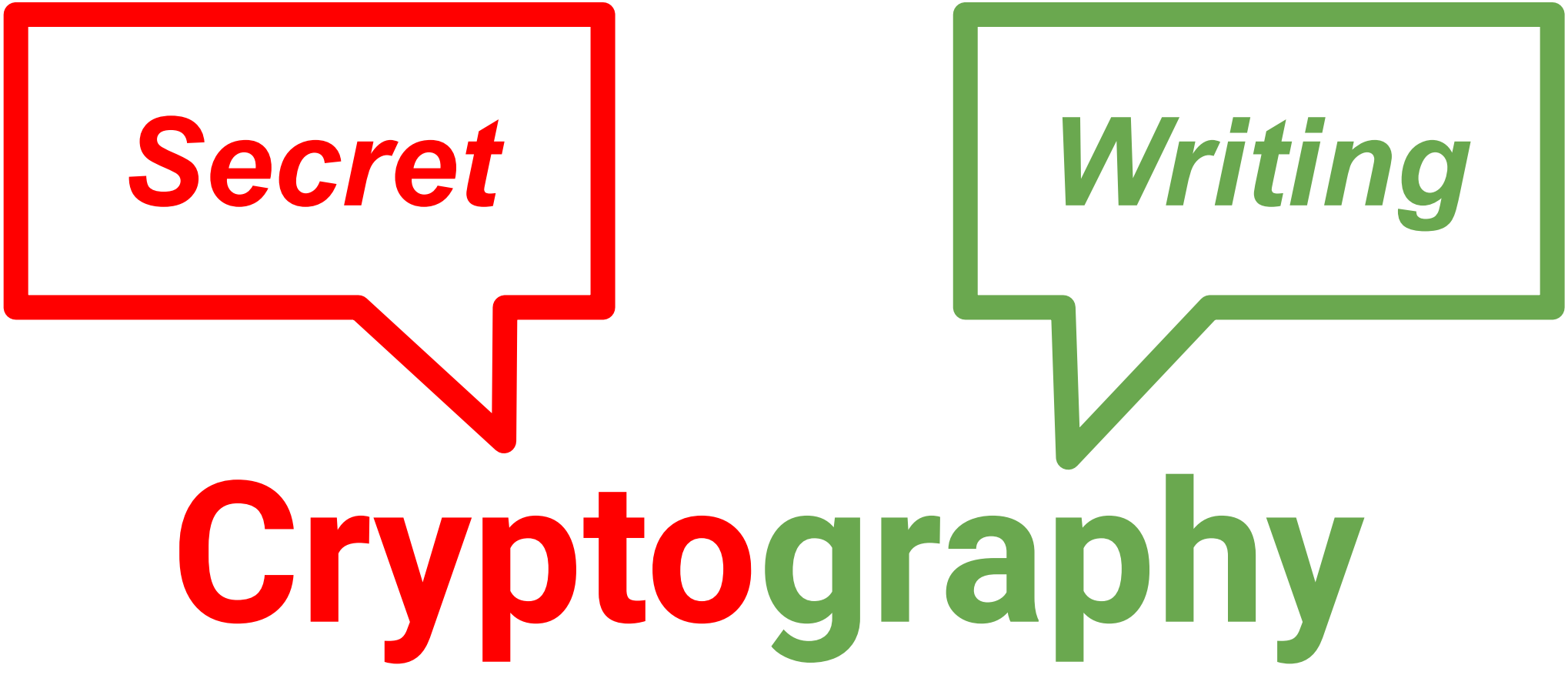
ACK ACK ACK ACK ACK ACK ACK

Cryptography

Cryptography

Cryp


Kate Pearce
@secvalve
Me, realising that I have to use the full word "cryptography" in my talk and can't use the short form "crypto" any more:
NOT GONNA MAKE IT
9:33 AM · Aug 24, 2019 · Twitter for iPhone


Secret
Writing
Cryptography

Cryptography is Control

Cryptography is Economics

Cryptography is Openness

Kerckhoffs's Principle
- *"A cryptosystem should be secure even if everything about the system, except the key, is public knowledge."*

Shannon's Maxim
- *"The enemy knows the system"*

Confidentiality - Privacy

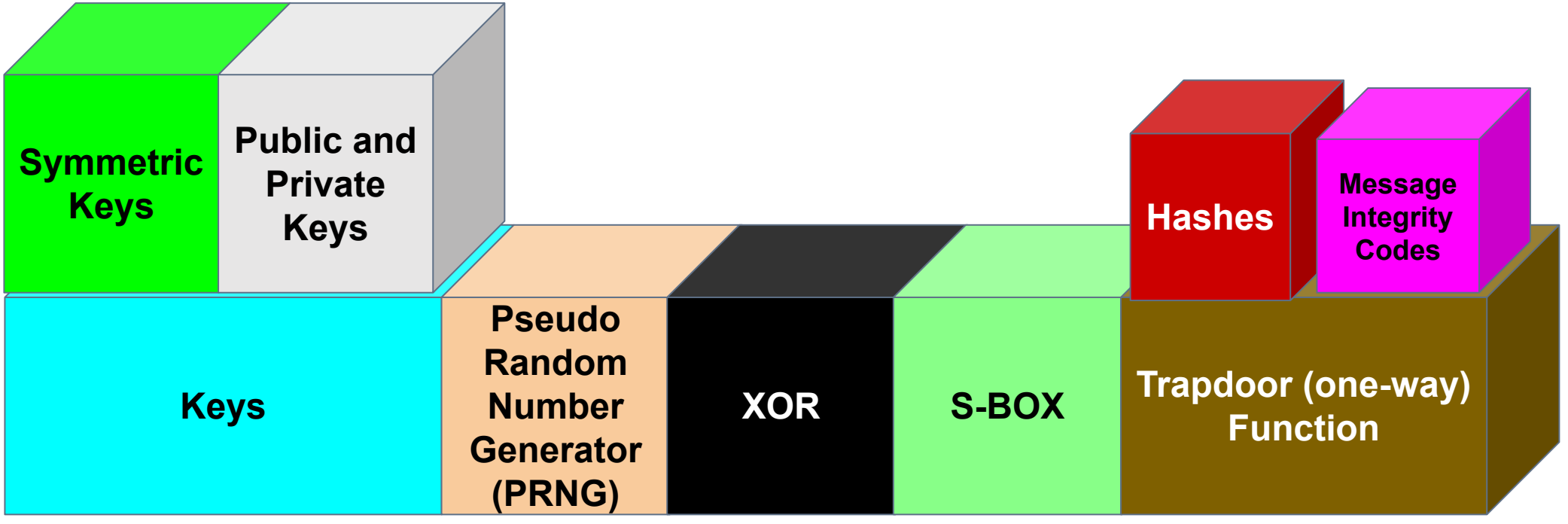
Authenticity - Sender

Integrity - Message

Primitives, and Building Blocks

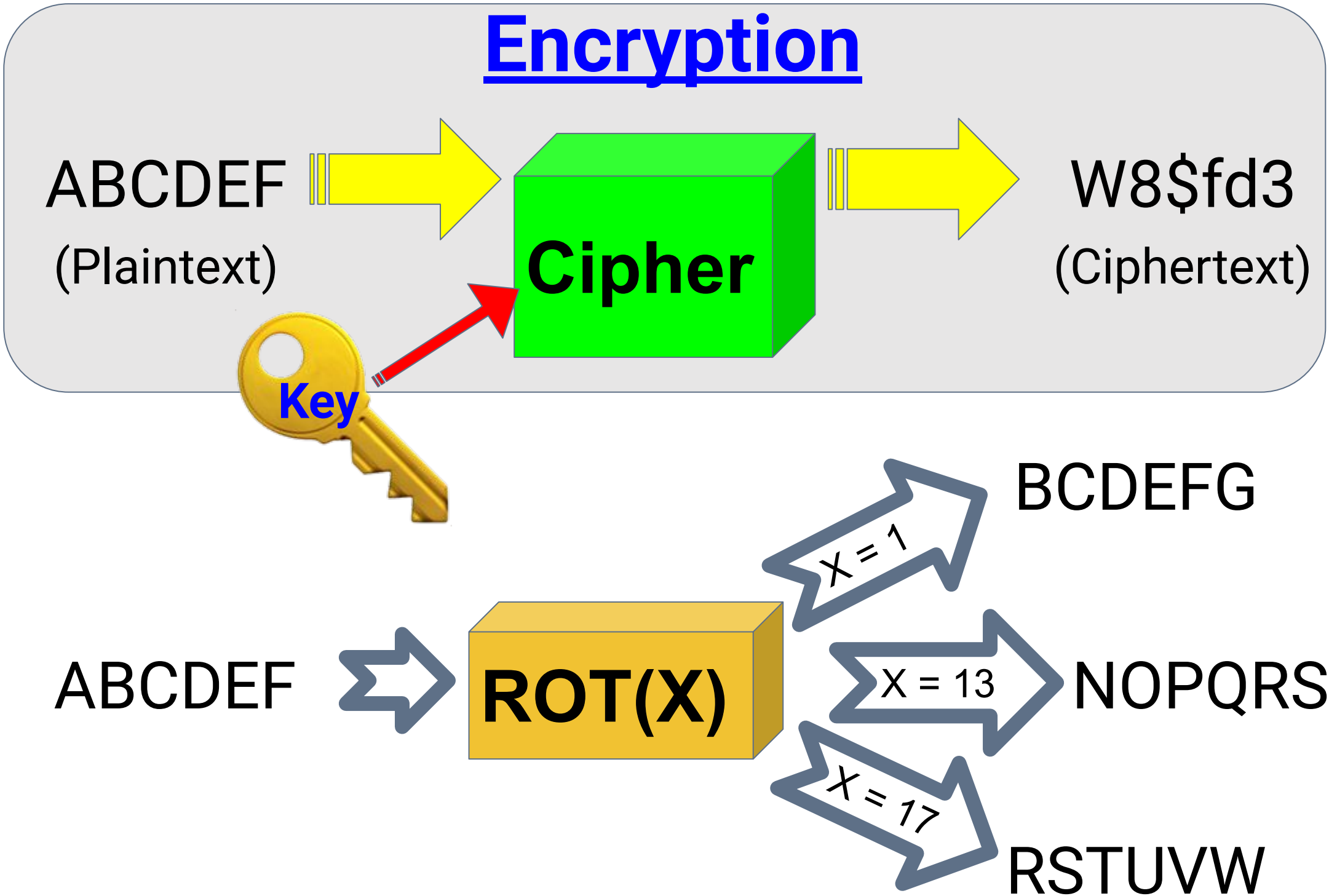
Some Key Primitives (and components)

Not going over all this in detail

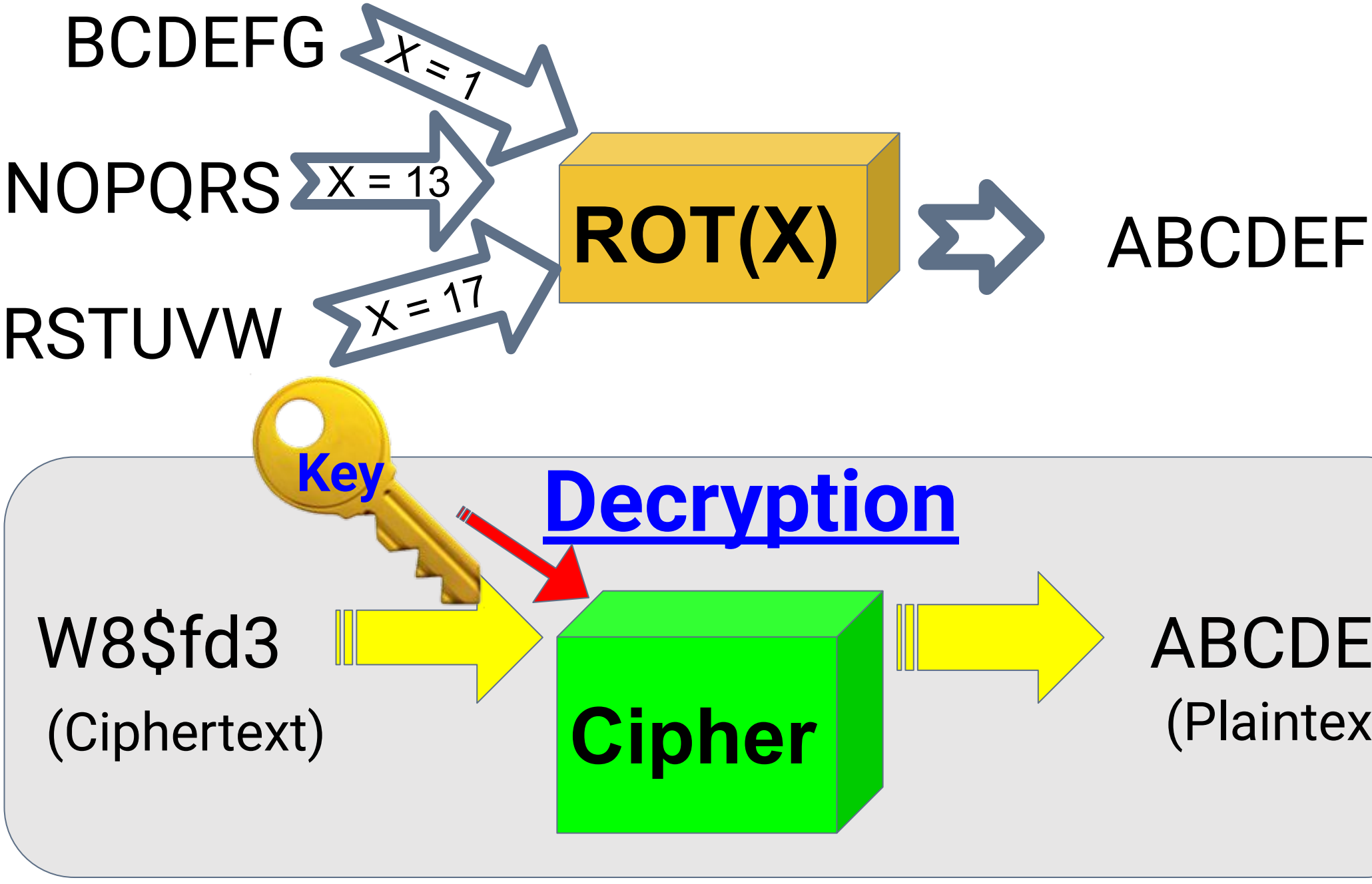


Symmetric Encryption

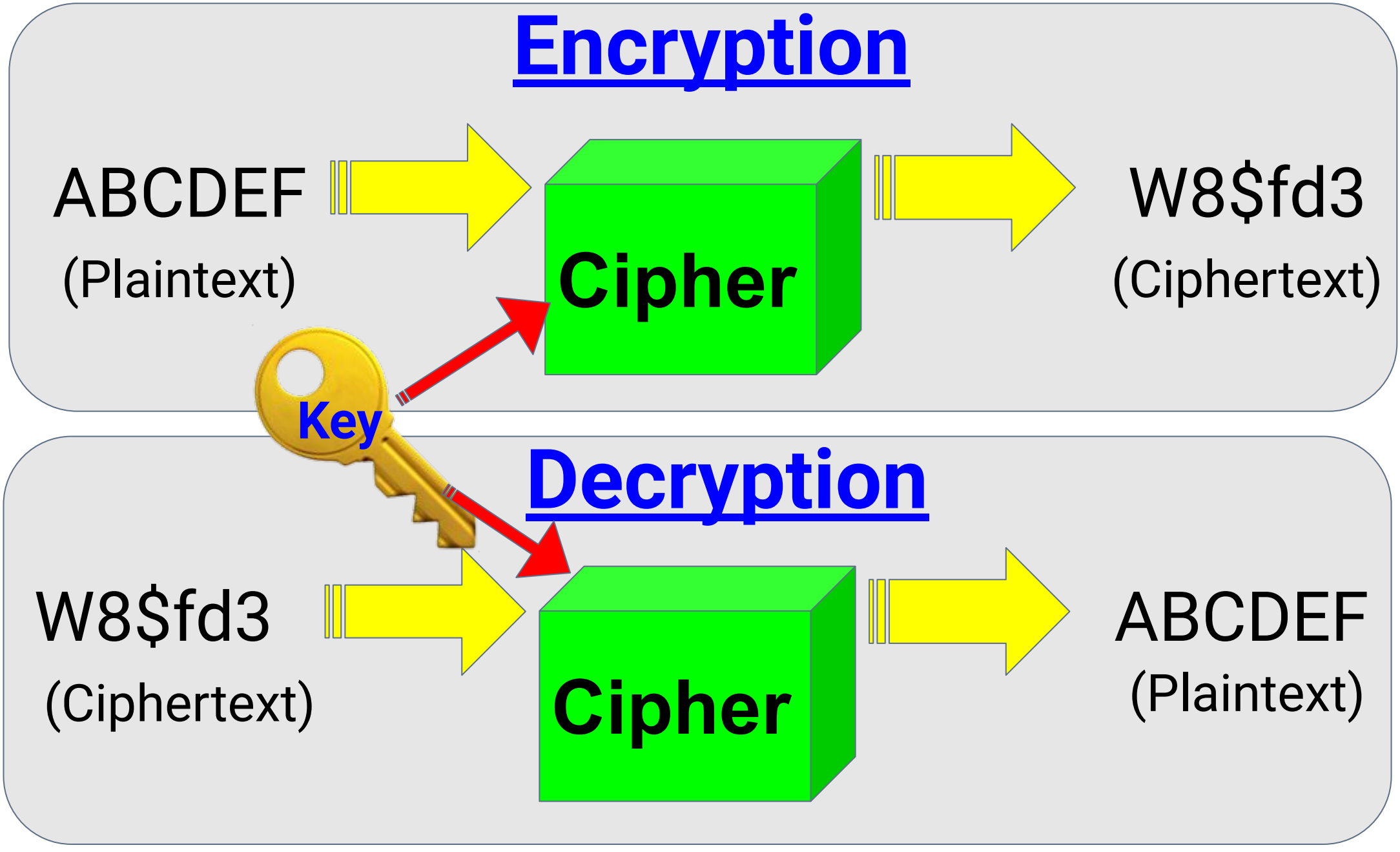
Symmetric Cryptography



Symmetric Cryptography



Symmetric Cryptography



Hashing and Trapdoor Functions

Is this the same?

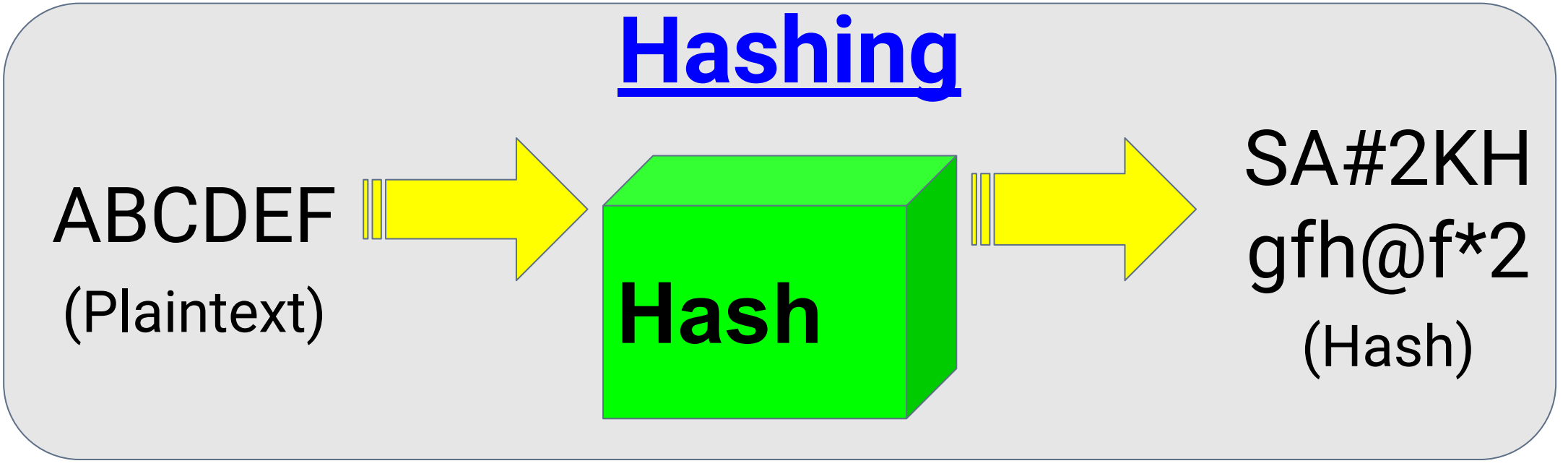
They had a red shirt

The number has a remainder of 1 when divided by 2

The number has a remainder of 5 when divided by 15

The number has a remainder of 11 when divided by 73

They had a red shirt
And green gumboots
And a lot of hair
And mittens
And were a cat


Hashing
ABCDEF
(Plaintext)
Hash
SA#2KH
gfh@f*2
(Hash)

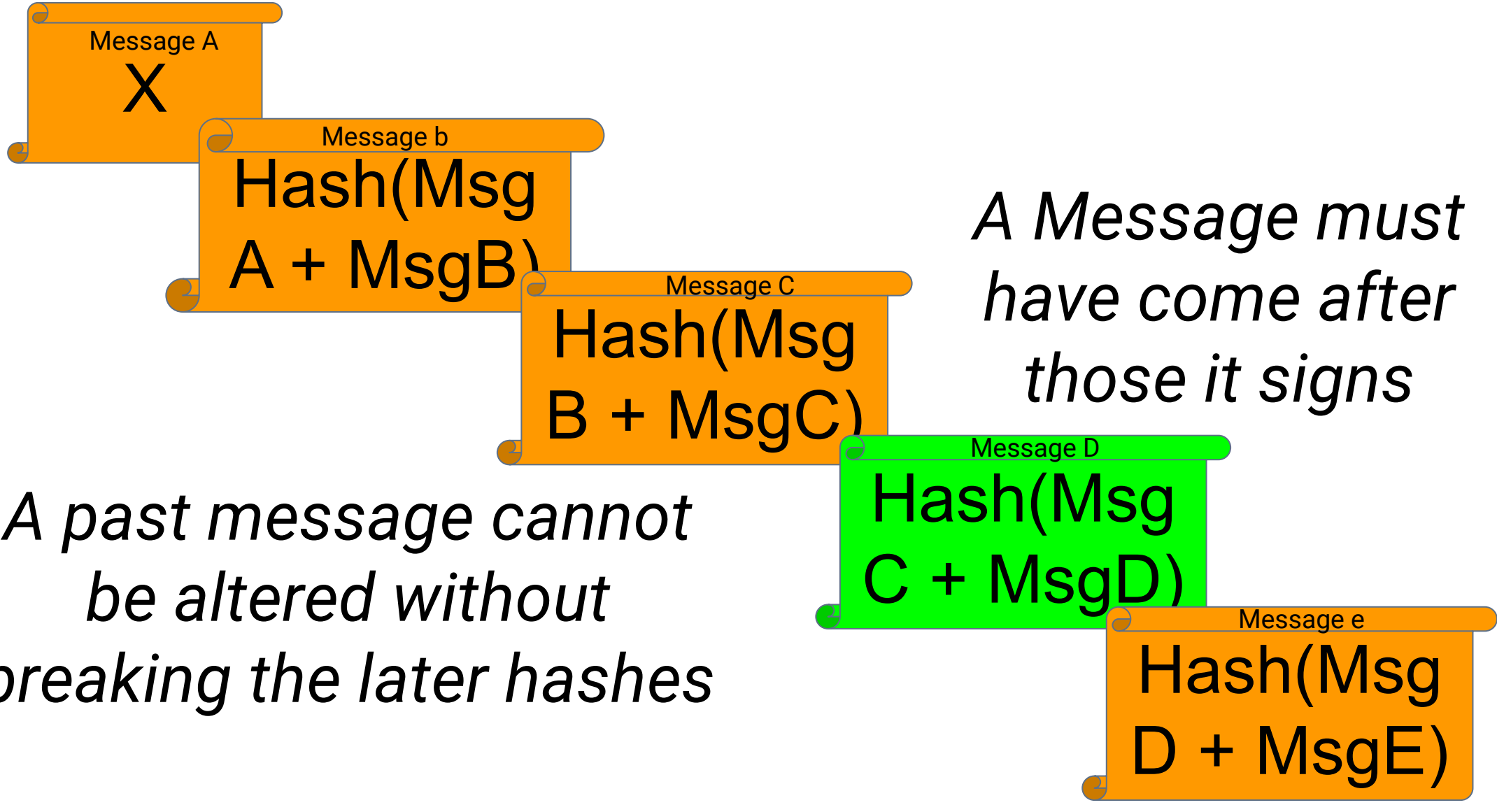
Hashing cannot go the other way, as information is lost

Red Shirt?

Hashing cannot go the other way, as information is lost

But it may tell you enough to be confident something is the same to the hashed thing

Hashing can be used to verify authenticity



A Message must have come after those it signs

A past message cannot be altered without breaking the later hashes

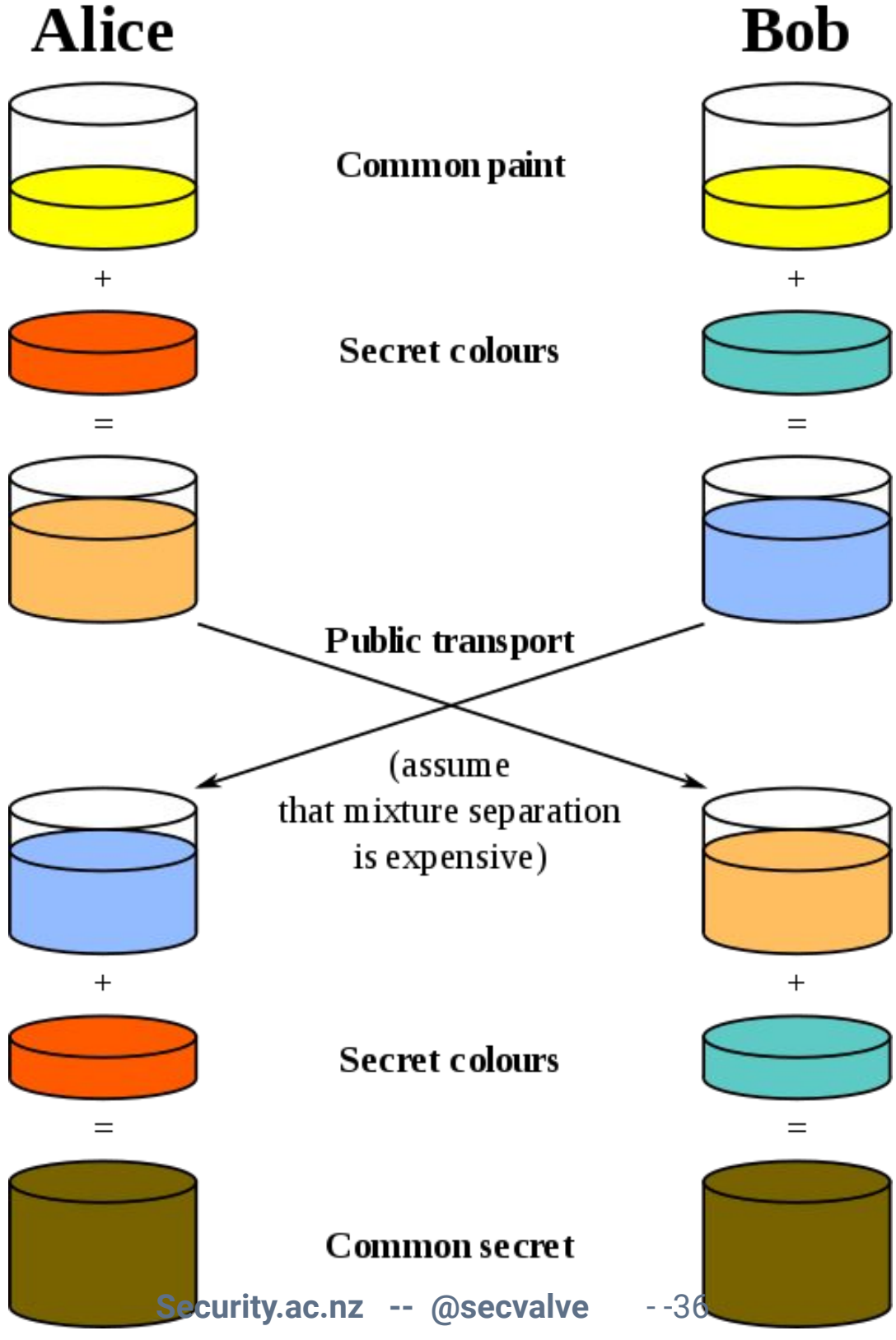
Asymmetric Encryption

Asymmetric Encryption?

We can gain security from with operations that are vastly more difficult to reverse without some useful information

Asymmetric Encryption?

We can gain security from with operations that are vastly more difficult to reverse without some useful information



Asymmetric Encryption?

We can gain security from with operations that are vastly more difficult to reverse without some useful information

Go through the hidden trapdoor activated by the statue's eye

Or, in mathematics: ***factoring numbers***

Asymmetric Encryption?

How do we protect our communications if we've never met?

How do we share a key without observers being able to use it?

With Public-Key Cryptography

Asymmetric Cryptography

PRIVATE KEY

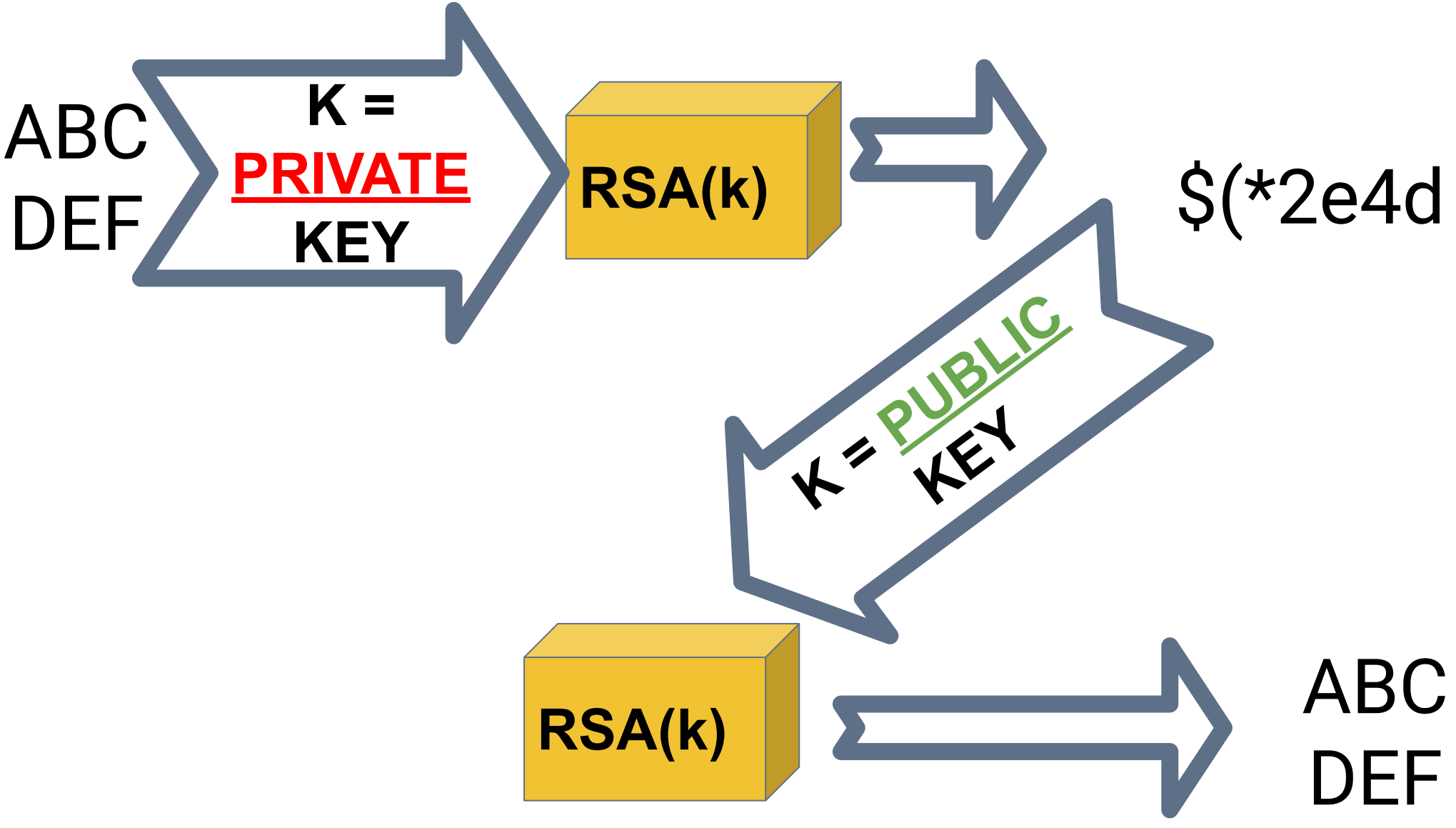
PUBLIC KEY

PRIVATE NEEDS TO BE SHARED

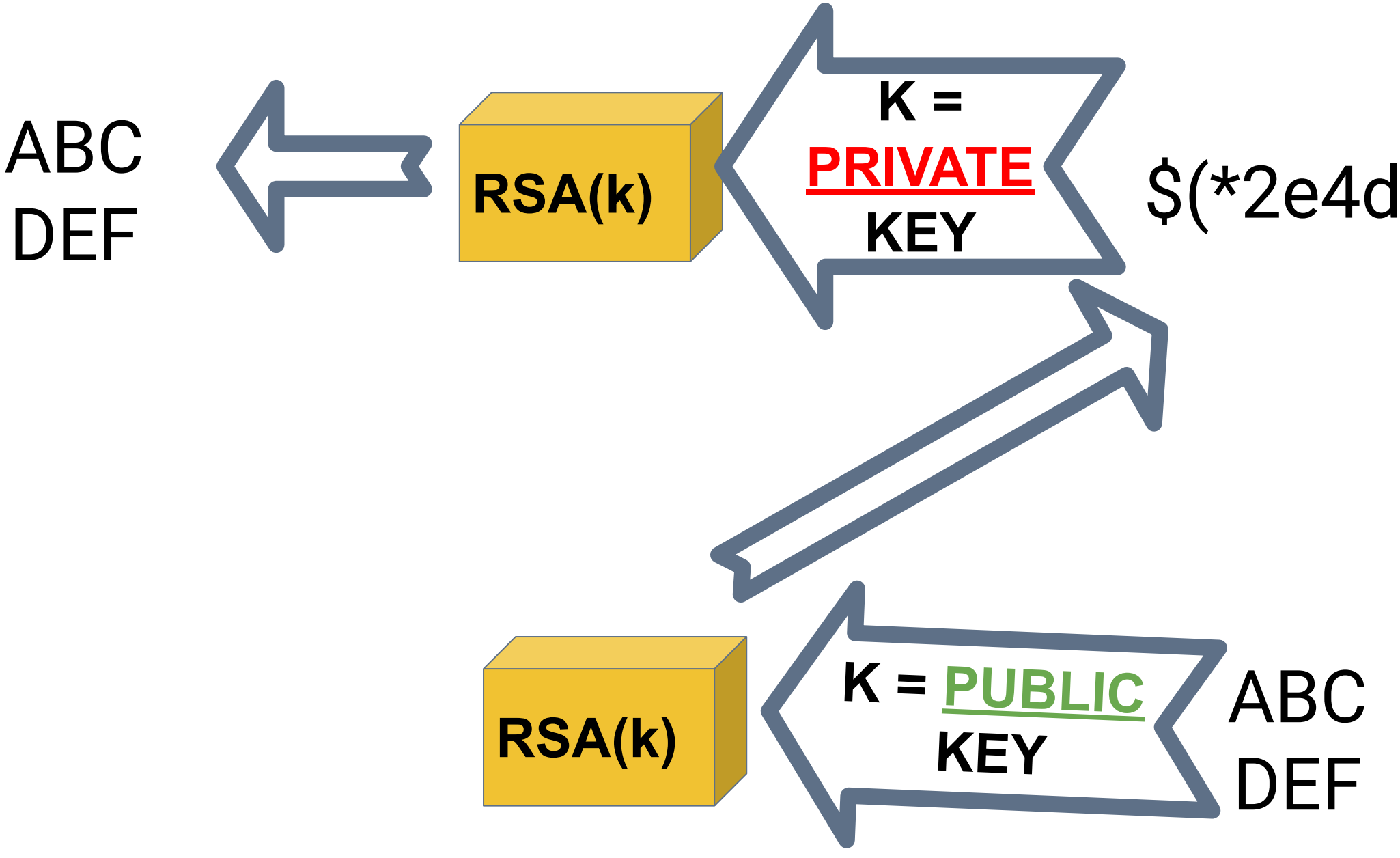
NEVER SHARED EVER Shared PUBLICLY

PRIVATE NEEDS TO BE SHARED

Asymmetric Cryptography



Asymmetric Cryptography



SO WHAT?

We now know:

- If something is encrypted with a Public Key it can only be read with the corresponding private key
- If something decrypts with a Public Key it was encrypted with the corresponding private key

Now each party has a way to communicate to the other party secretly.

Now each party has a way to communicate to the other party secretly.

Example: (NOT HOW Diffie-Hellman Key Exchange WORKS)

1. **BOTH Publicly:** Let's use our a common word "peregrine"

2. Alice sends a message [encrypted with Bob's public Key] to use the secret word "**Opossum**"
   a. Only Bob can read this

3. Bob sends Alice a message [encrypted with his private key and then her public key] and then his to use the secret word "**WeaselSquawk**"
   a. Only Bob can have sent this, Only Alice can read it

They now have a key to use for symmetric encryption:

peregrineOpossumWeaselSquawk

Exercise: Find the vulnerability in this method (Hint: how does Bob Auth Alice?)

Why not use Public-private cryptography all the time?

It is thousands of times more computationally intensive (And key reuse should be avoided)

Signing and Message Integrity Codes

We also now have a way to validate the authenticity of something!

If i send you a hash result that has been put through my private key (signed) then you can compare the value i sent with the value you get checking yourself!

If they’re the same then you know it came from me.

Public Key Infrastructure

How do we know the public key is the right one?

We could share it in advance

...

BUT THAT'S THE SAME PROBLEM AS BEFORE!

How do we know the public key is the right one?

With Public-Key Infrastructure

How do we know the public key is the right one?

With Public-Key Infrastructure

(We have common friends)

How do we know the public key is the right one?

With Public-Key Infrastructure

(We have common friends)
(Who have common friends)

Public Key Infrastructure

Elizabeth Has Signed the Certificate

Alice Asks Elizabeth to Verify Her Public Key

I attest Alice's Key

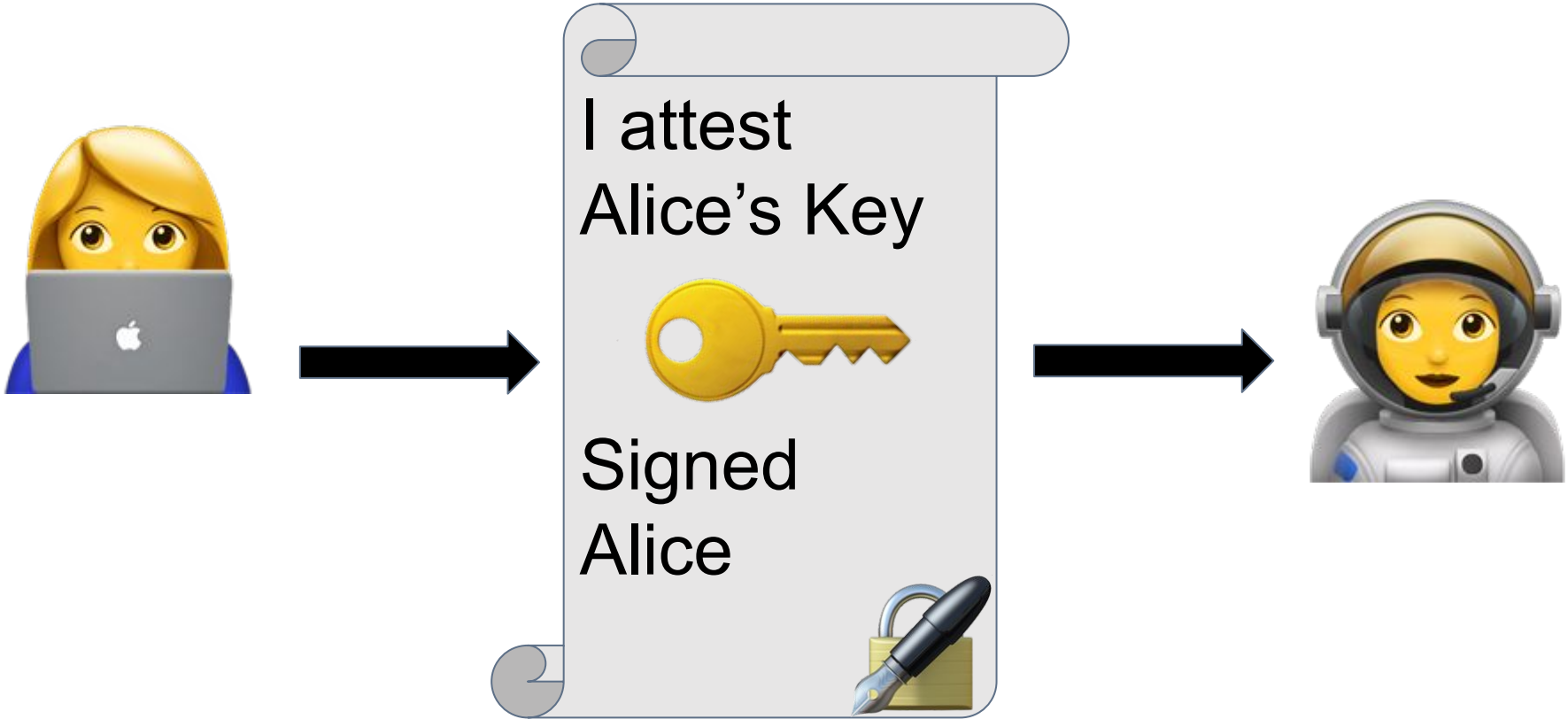
Signed Elizabeth

Becky Already Trusts Elizabeth and Her Public Key

(It was in her web browser)

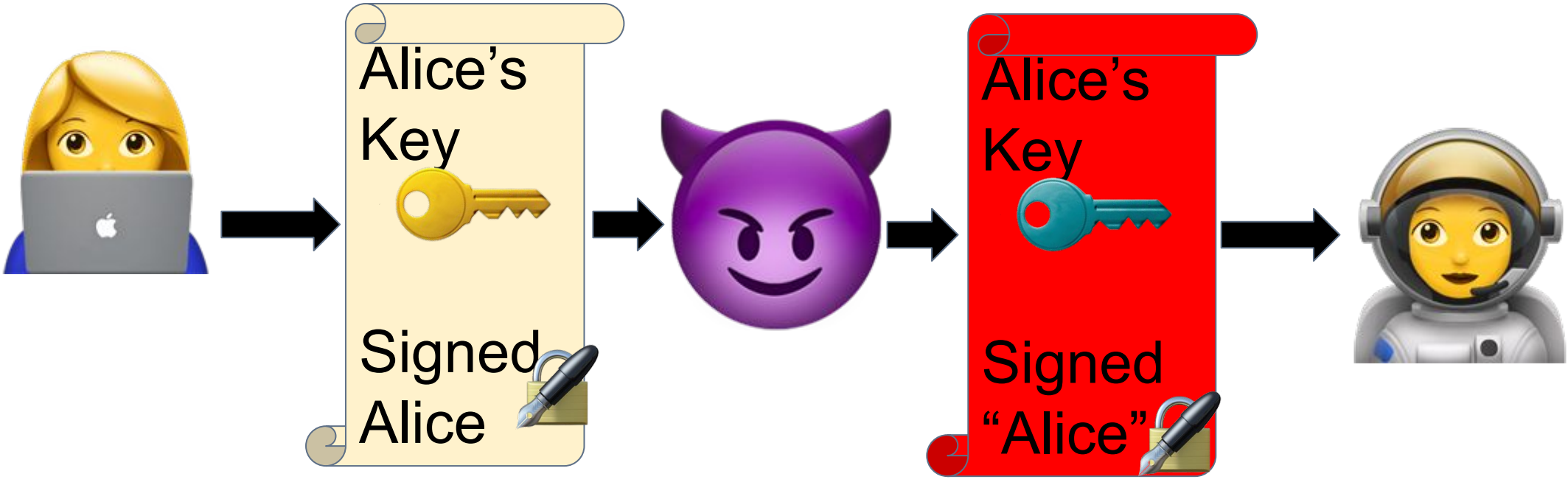
Self-Signing

Cryptography Gotchas



Self-signed certificates break the whole system as you can't tell if someone is in the middle

Cryptography Gotchas



Self-signed certificates break the whole system as you can't tell if someone is in the middle

Computers and Randomness

It doesn't matter how good your encryption algorithm is if your key is easily guessed

...but...

Computers and Randomness

Computers:

- Are terrible at randomness
- Do exactly what they are told

Given the same input, they do the the same thing every. single. Time.

So… how do we get a good key?

So… how do we get a good key?

With a Random Number Generator (RNG)?

- *No - Computers don't [usually] have those*

So… how do we get a good key?

With a Pseudo Random Number Generator (PRNG)?

- *Maybe, but probably not*

So… how do we get a good key?

With a Cryptographically Secure Pseudo Random Number Generator (CSPRNG/CPRNG)?

YES!

You get a biscuit:

Don’t use a normal random generator for cryptography. Ever.

(Also, don’t use the wrong Datatype for a key. Ever)

STORY TIME!

STORY TIME!

Blockchain Bandit and How to lose millions of dollars of crypto coins

https://www.wired.com/story/blockchain-bandit-ethereum-weak-private-keys/

This brings us to another point.

Hashing does not provide privacy if the input values can be predicted.

Hashing *does not* provide privacy or security if the input values can be predicted, or if values can be tested rapidly.

- Hashes can be tested at speeds of millions to billions of values per second
- Some things come in only a limited number of values.

Never ever simply hash secrets, or things with predictable values, for "security" or privacy reasons:

- Names
- Usernames
- User ID's
- Passwords
- Credit Card Numbers
- Email Addresses
- Phone Numbers
- IP/Mac Addresses

But, i haven’t discussed how to store important secrets yet have i?

Secret Storage!

Here's the thing about user passwords.

You do NOT need to store them

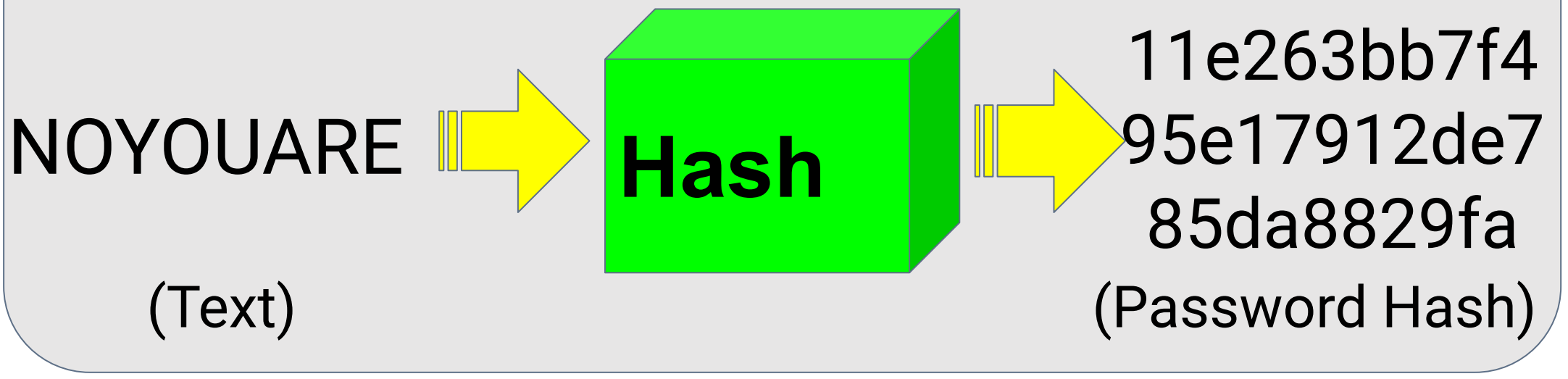
- NEVER EVER store raw or encoded passwords
- Never Reversibly Encrypt Passwords

You only need to know if a given password is correct

You only need to know if a given password is correct

So, we use hashes!

BAD PASSWORD Hashing



By storing the hash we do not know user's password, and cannot leak it

But, DON'T USE A NORMAL HASH For PASSWORDS. See next slide

Secret Storage

But, DON'T USE A NORMAL HASH For PASSWORDS.

Presume attackers will compromise them, and:

- DO NOT Truncate, or change the case of, passwords before hashing
- Use a SLOW & computationally intensive hash
  - (Argon2, PBKDF2,Scrypt - or bcrypt if you have to)
  - NEVER USE MD5, SHA-X, or FOR PASSWORDS
- Use a complex, user-specific, SALT in your calculated hash value

https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Password_Storage_Cheat_Sheet.md

Protocols

I haven’t actually mentioned a lot of protocols have i?

Protocols

Here's a few protocols you may want basics on:

ARP / DHCP

802.11

TCP/IP

FTP

UDP

ICMP

SMTP

HTTP / HTTP2 / HTTP3|QUIC

DNS

SSL/TLS

But, no time for that today!

Because here's the thing...

These building blocks in various combinations are what makes the algorithms:

SSH -> Public/Private Authentication (without Certificates to verify)

HTTPS -> HTTP Protected with SSL/TLS (Which is the certificate-based encryption)

Bitcoin & Crypto Currencies -> Hash Chains (and a bit more stuff)

Conclusion

Cryptography is Control

Cryptography is Economics

Cryptography is Openness

Kerckhoffs's Principle
- *"A cryptosystem should be secure even if everything about the system, except the key, is public knowledge."*

Shannon's Maxim
- *"The enemy knows the system"*

Some things i didn't cover but wanted to:

- Digital Rights Management
- Web Of Trust
- Ransomware
- Steganography
- Forward Secrecy
- Quantum Computing
- Specific Protocol Recommendations
- Cryptanalysis and Cryptographic Attacks
  - Ciphertext-only,
  - Known Plaintext,
  - Chosen plaintext,
  - Chosen ciphertext
- Implementation and Key Attacks
  - Birthday Attacks,
  - Key and Plaintext Guessing Attacks,
  - Side Channel Attacks,
  - Rainbow Tables

tldr;

DO Use Public Algorithms	DO NOT Roll-your-own Algo/Function	CONCENTRATE ON Key Distribution
DO Use Public Protocols	DO NOT Roll-your-own Protocol	CONCENTRATE ON Key Management
DO Use Secure PRNG for Keys	DO NOT Roll-your-own PRNG OR Use a non-secure PRNG	
DO Use a Secure Implementation	DO NOT Implement your own	
DO Use Recommended Cipher Suites	DO NOT Use Bad, Weak, or Null Suites	
DO Use Slow Algorithms and Salt Secret Hashes	DO NOT Hash Secrets with simple or fast hashes	