

Secure Coding for the Enterprise

*.NET Framework / .NET Core
2-Day Workshop*

Stephen Haunts
www.stephenhaunts.com
@stephenhaunts



STEPHEN HAUNTS
TRAINING



No Planned Fire Alarms





STEPHEN HAUNTS
TRAINING



PLURALSIGHT



EIDOS
I N T E R A C T I V E



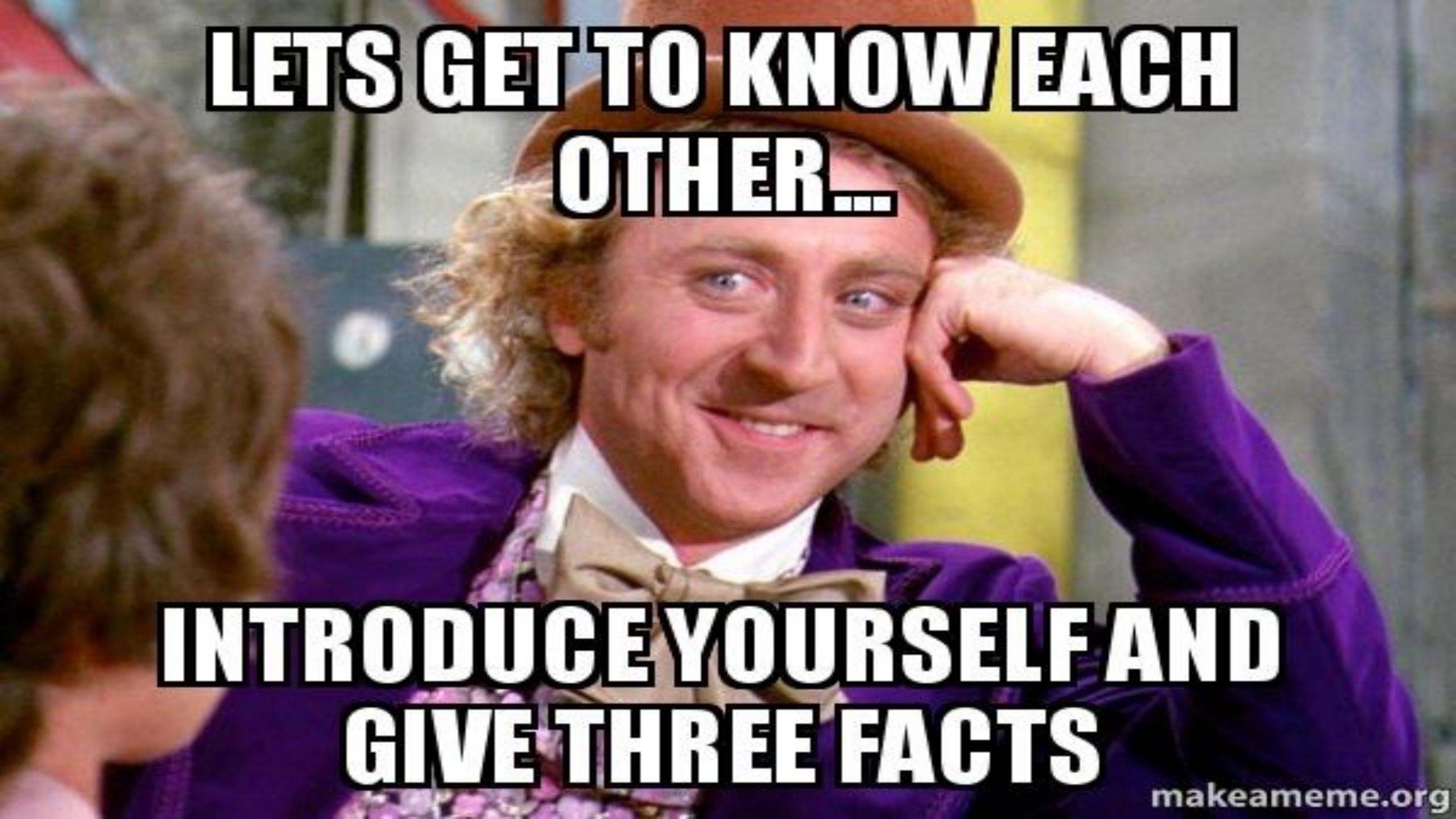
egg

citi®

DOLLAR
FINANCIAL CORP

Boots

A green checkmark icon followed by the word 'RIGHTINDEM' in a bold, black, sans-serif font.



**LETS GET TO KNOW EACH
OTHER....**

**INTRODUCE YOURSELF AND
GIVE THREE FACTS**

Workshop Setup

I provide all the cryptography code

You will be stepping through code

All samples are built for .NET 5+

PC – Visual Studio 2019.2022, JetBrains Rider

Mac – Visual Studio Mac, JetBrains Rider

Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

 master ▼

 1 branch

 0 tags

Go to file

Add file ▼

Code ▼

About



Source code for the Advanced Secure Coding Workshop

 Readme

 MIT License

Releases

No releases published

[Create a new release](#)

Packages

No packages published

[Publish your first package](#)

Languages

 C# 100.0%

README.md



SecureCodingWorkshop

Source code for the Advanced Secure Coding Workshop

<https://github.com/stephenhaunts/SecureCodingWorkshop>

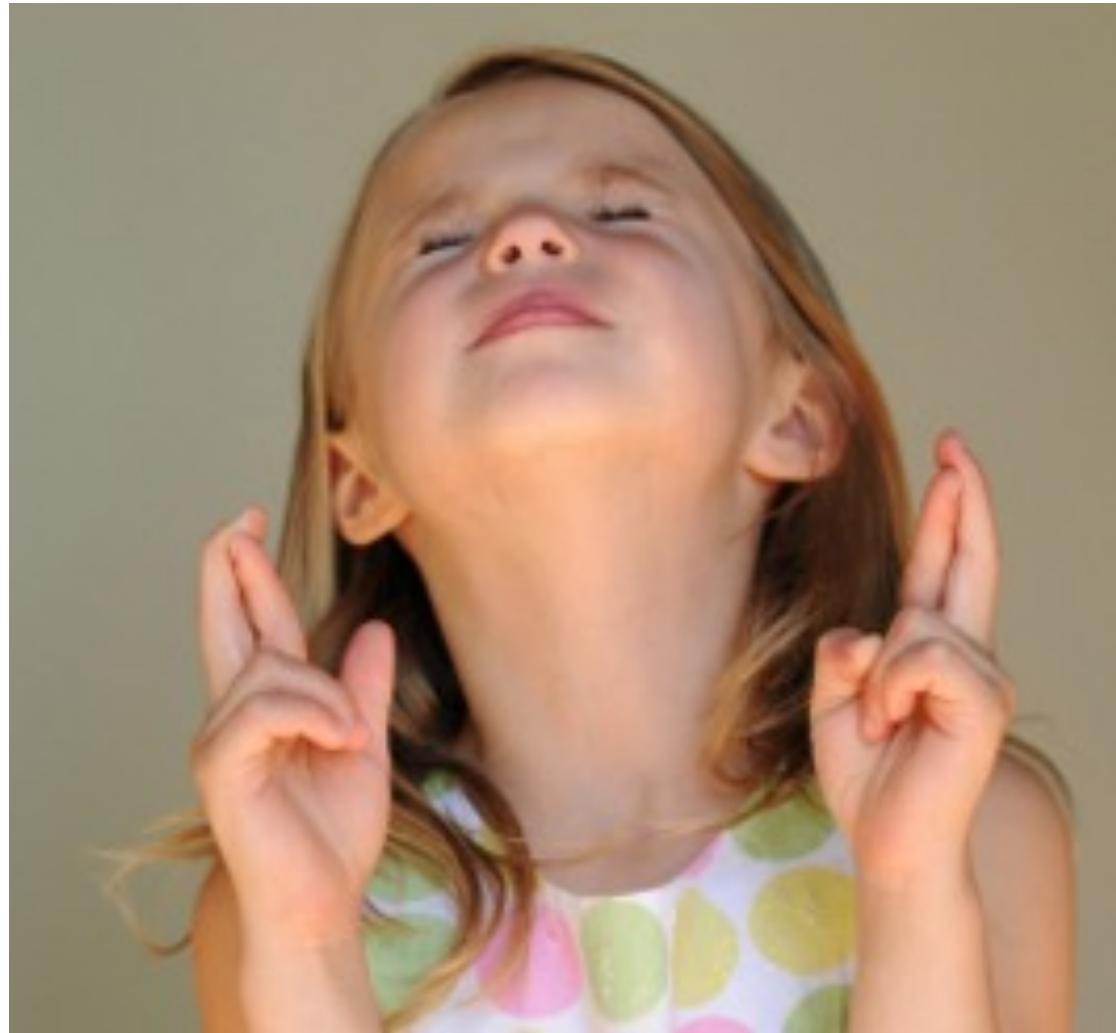


Now it's
your turn!

Download the code
Load up the solution files
Make sure they build

<https://github.com/stephenhaunts/SecureCodingWorkshop>





 **KEEP
CALM
AND
FINGERS
CROSSED**

**Hope is not a
Strategy!**

The definitive guide to all project managers



What the fuck is security

How to ignore it and deliver your project

O'RLY

:P orly.coloncapitalp.com

Awn Thyme



Secure PIN Number
Storage

Secure Hashed Password
Storage

HSM Integration



Secure PIN Number
Storage

Secure Hashed Password
Storage

HSM Integration



PCI Card Token Vault
HSM Integration



Secure PIN Number
Storage

Secure Hashed Password
Storage

HSM Integration



PCI Card Token Vault
HSM Integration



RIGHT IN DEM

Secure Hashed Password
Storage

Record Level Encryption in
Multi Tenant Environment

Azure Key Vault Integration



Secure PIN Number
Storage

Secure Hashed Password
Storage

HSM Integration



PCI Card Token Vault
HSM Integration



RIGHTINDEM

Secure Hashed Password
Storage

Record Level Encryption in
Multi Tenant Environment

Azure Key Vault Integration



Encryption of Patient PIII
Data Sent from Stores to
Support Centre



Secure PIN Number
Storage

Secure Hashed Password
Storage

HSM Integration



PCI Card Token Vault
HSM Integration



RIGHTINDEM

Secure Hashed Password
Storage

Record Level Encryption in
Multi Tenant Environment

Azure Key Vault Integration



Encryption of Patient PII
Data Sent from Stores to
Support Centre

Regulated Environments

FCA, PCI DSS, Health (NHS)

25 1n 34 6





The General Data Protection Regulation

What This Workshop Isn't

$$\frac{\sqrt{3}}{4} = (\alpha^2)$$
$$a^2 + b^2 = c^2$$

$$\frac{1}{2} + B = 0 \Rightarrow B = -\frac{1}{2}$$

$$A = \frac{1}{2}AB \sin C = \frac{1}{2}a^2b^2 \sin C$$

$$c^2 = a^2 + b^2 - 2ab \cos C$$

$$27/32 = 0.845 \approx$$
$$Y = \frac{1}{4}x - \frac{1}{8}(\frac{1}{xy}) + x^2 - 10$$

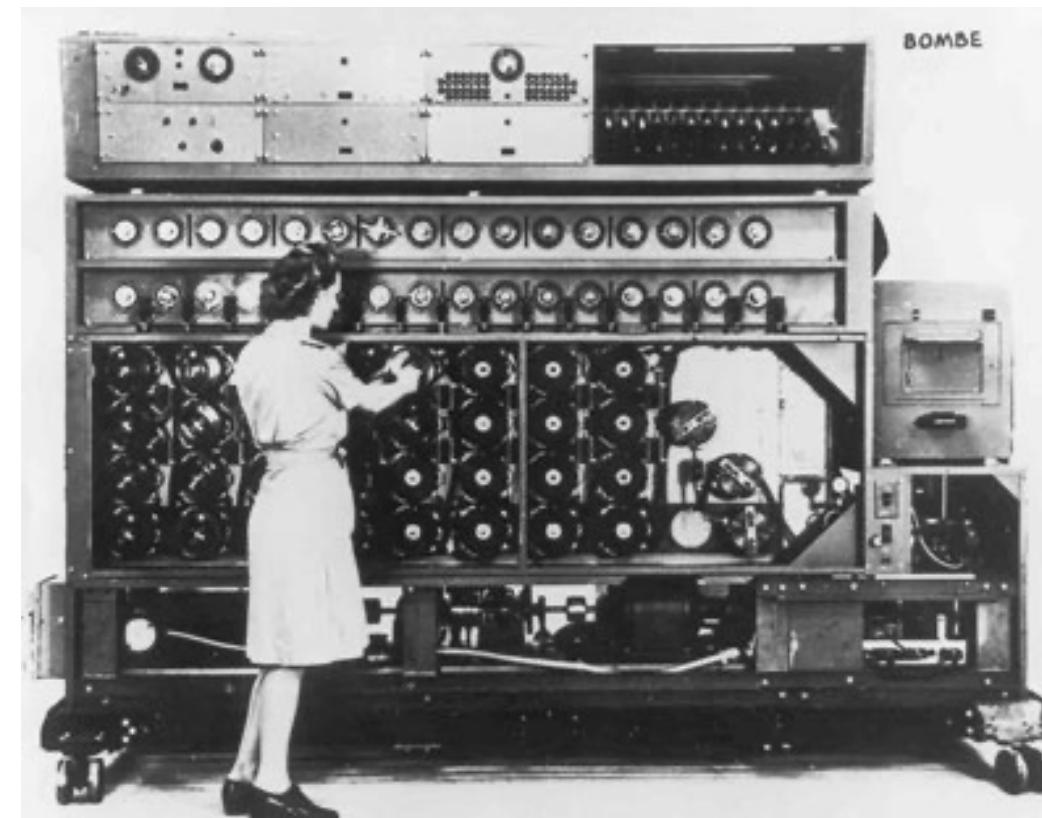
$$\pi = 3.14$$
$$z = \sqrt{2}\pi e$$

$$A = \pi r^2 \frac{10}{25}$$

$$b^2 = \cos^2 C - 2ab \cos C$$

$$= a^2(\cos^2 C + \sin^2 C)$$

$$= a^2 + c^2 - 2ac \cos C$$



What This Workshop Is



What This Workshop Is

.NET Framework / Core
Server Side or Client Code

What This Workshop Is



.NET Core 1.1
.NET Standard 1.6

What This Workshop Is



.NET Core 2.0+
.NET Standard 2.0+

What This Workshop Is



.NET Core 5.0+

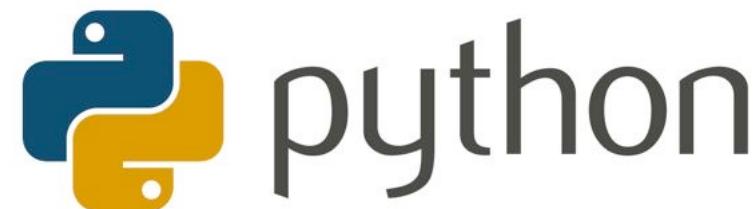
What This Workshop Is



What This Workshop Is



Microsoft



python



Ruby



What We Will Cover

Random Numbers

What We Will Cover

Random Numbers

Hashing + HMACS

What We Will Cover

Random Numbers

Hashing + HMACS

Secure Password Storage

What We Will Cover

Random Numbers

Hashing + HMACS

Secure Password Storage

Symmetric Encryption - AES

What We Will Cover

Random Numbers

Hashing + HMACS

Secure Password Storage

Symmetric Encryption - AES

Asymmetric Encryption - RSA

What We Will Cover

Random Numbers

Hashing + HMACS

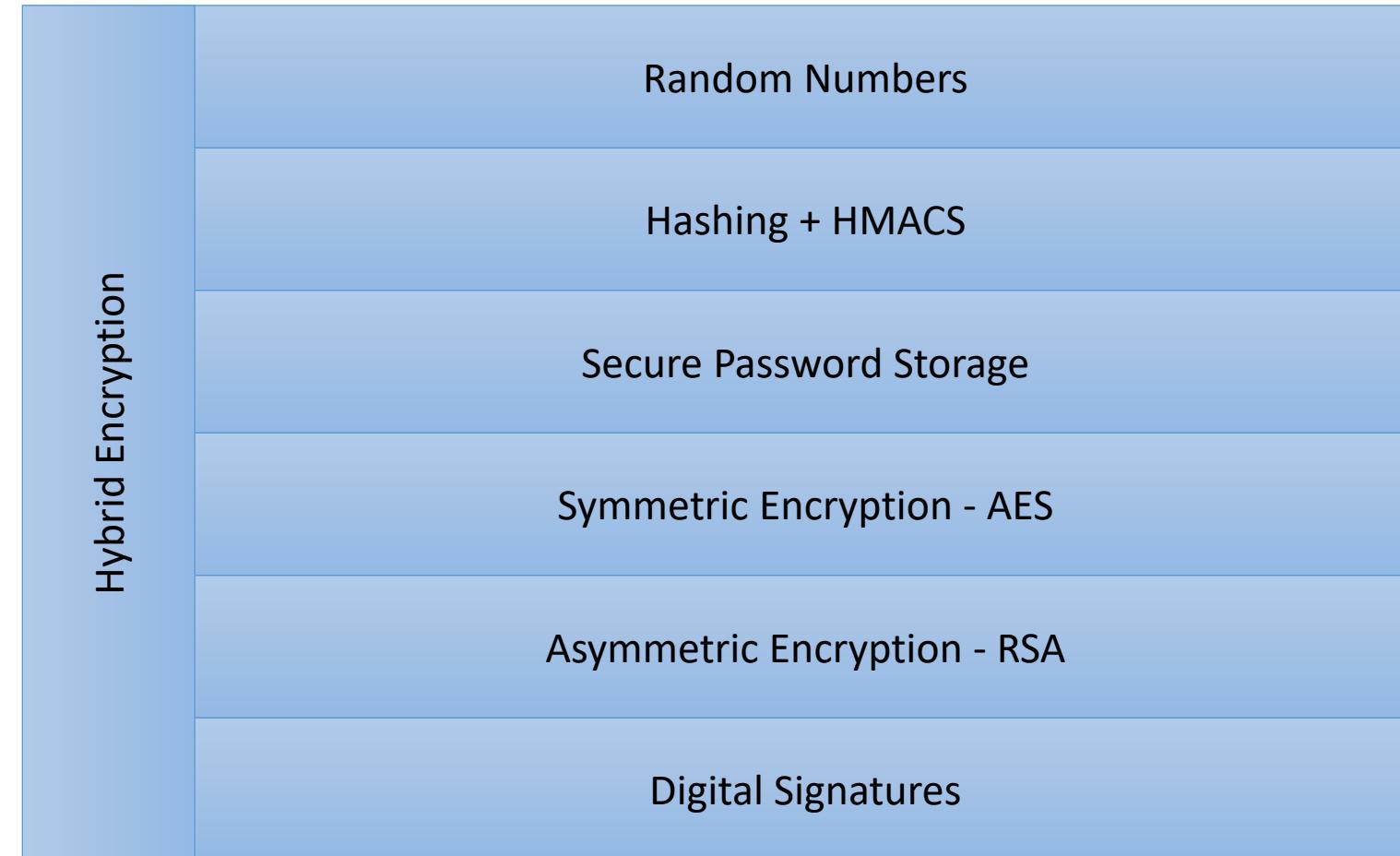
Secure Password Storage

Symmetric Encryption - AES

Asymmetric Encryption - RSA

Digital Signatures

What We Will Cover



What We Will Cover

Time	Day 1	Day 2
9 – 10.30am	Introduction, Setup, Random Numbers	Recap from Day 1
10.45 – 12.30am	Hashing and HMACS, Passwords	Hybrid Encryption, Azure Key Vault
12.30 – 1.30pm	LUNCH	LUNCH
1.30 – 3.15pm	Symmetric Encryption	Azure Key Vault, Blockchain Fundamentals
3.30 – 5pm	Asymmetric Encryption + Digital Signatures, Project Time, Wrap Up	Blockchain Fundamentals, Summary, Wrap Up

What Is Cryptography?



Protecting information

Encryption

Keys

Cipher text

Cryptanalysis



Now it's
your turn!

Discuss some example of
Cryptography everyday use?

Examples of Cryptography





Now it's
your turn!

What data should we encrypt?

What Is Cryptography?

Confidentiality

Integrity

Authentication

Non-Repudiation

Data Breaches

DATA BREACH STATISTICS

DATA RECORDS LOST OR STOLEN SINCE 2013

13,443,149,623

ONLY 4% of breaches were “Secure Breaches” where encryption was used and the stolen data was rendered useless.

DATA RECORDS ARE LOST OR STOLEN AT THE FOLLOWING FREQUENCY



EVERY DAY

6,077,373

Records



EVERY HOUR

253,224

Records



EVERY MINUTE

4,220

Records



EVERY SECOND

70

Records

<http://breachlevelindex.com>

261
pwned websites

4,859,012,158
pwned accounts

60,200
pastes

58,307,714
paste accounts

Top 10 breaches

-  711,477,622 Onliner Spambot accounts 
-  593,427,119 Exploit.In accounts 
-  457,962,538 Anti Public Combo List accounts 
-  393,430,309 River City Media Spam List accounts 
-  359,420,698 MySpace accounts
-  234,842,089 NetEase accounts 
-  164,611,595 LinkedIn accounts
-  152,445,165 Adobe accounts
-  112,005,531 Badoo accounts 
-  105,059,554 B2B USA Businesses accounts 



<https://haveibeenpwned.com>



Now it's
your turn!

What are the consequences
of data breaches?



Now it's
your turn!

How can data breaches occur?

Effects of Data Breaches



Reputational damage to the organization

Loss of sales

Customers switching to a competitor

Legal Action from customers or other companies

Compliance costs (PCI DSS, HIPAA)

Safety of compromised customers

Effects of Data Breaches



Loss of personal data

Identity theft

ID theft leads to impersonation

Loss of financial data

Compromised card data

Compromised transaction history
(could be embarrassing)

Data Breaches are
Inevitable

Not if.... when

Don't Just Focus on
Perimeter Security

Cryptographic Primitives

Encoding

Encoding

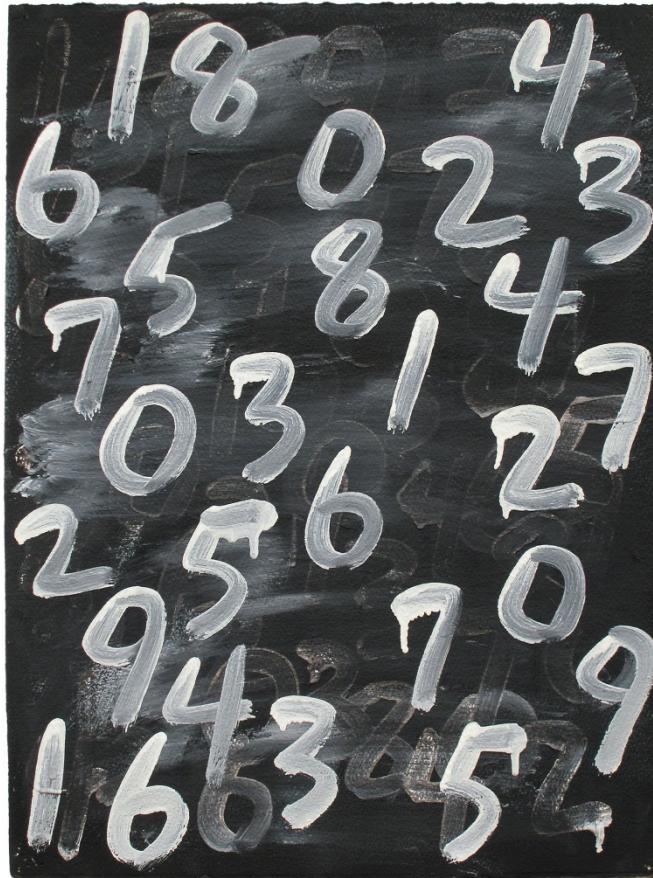
- All the API's we are looking at take/return byte arrays
- This is great for files etc.
- Most of the time we are encrypting and storing text
- So we need to use encodings

Encoding

- Encoding.UTF8.GetBytes("Hello World");
- Convert.ToString(myByteArray);
- Convert.FromBase64String("tv3785e478f5e5hy");

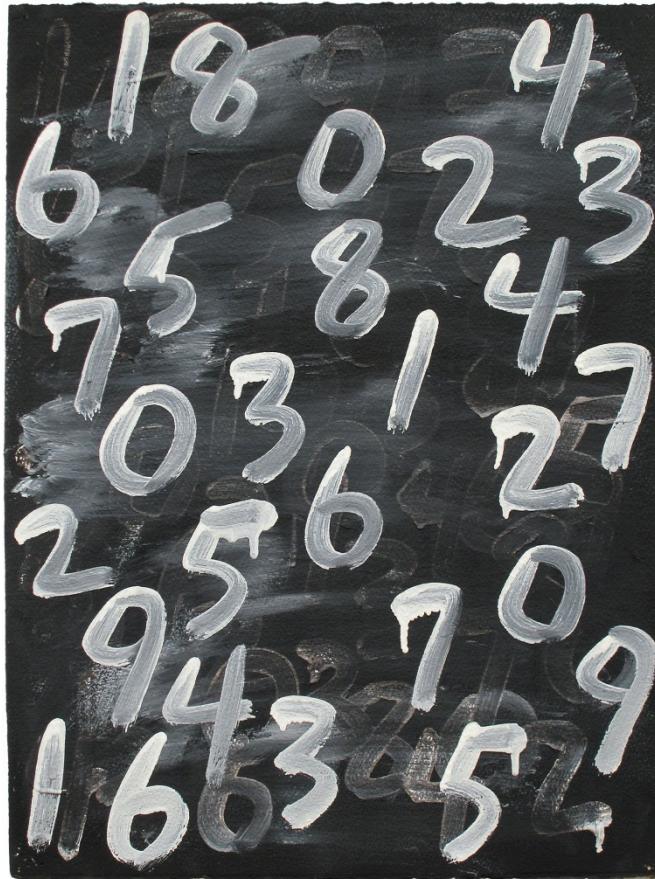
Random Numbers

Why Are Random Numbers Important?



- Used for generating encryption keys
- Software based random numbers are not always truly random
- Randomness can be created from human interaction
- Not practical for server applications

Why Are Random Numbers Important?



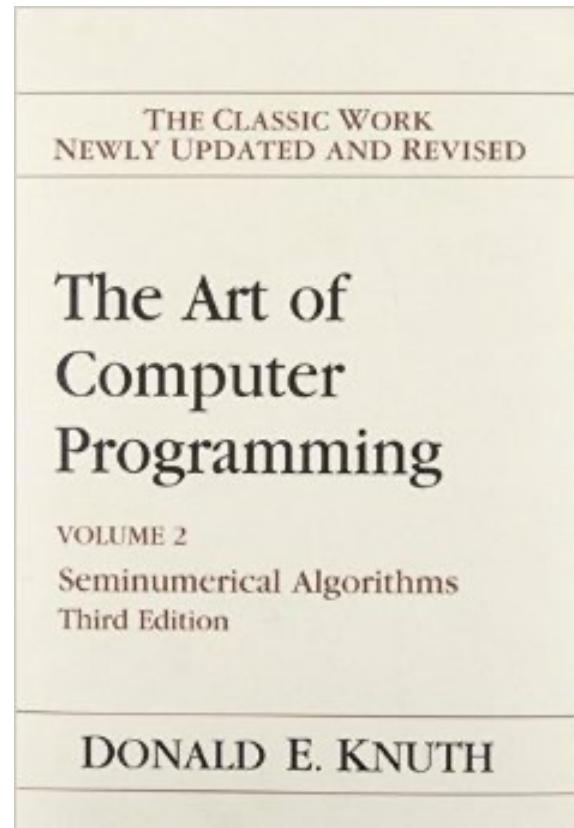
- Can use dedicated hardware or a specifically designed algorithm

System.Random and Its Problems

- ***System.Random*** is a pseudo random number generator
- A seed value is passed into the constructor
- The seed value should be different each time
- ***System.Random*** is deterministic and predictable

System.Random and Its Problems

- Based on the Subtractive Random Number Generator by Donald E. Knuth

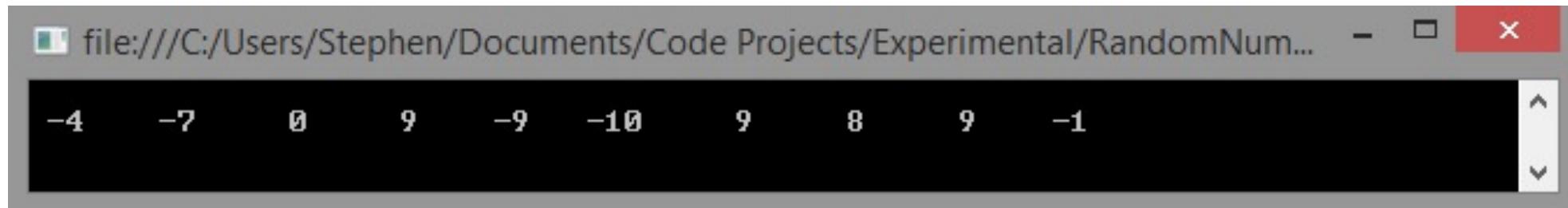


System.Random and Its Problems

```
Random rnd = new Random(250);

for (int ctr = 0; ctr < 10; ctr++)
{
    Console.Write("{0,3} ", rnd.Next(-10, 11));
}
```

System.Random and Its Problems



System.Random and Its Problems

- Microsoft recommends creating 1 instance of ***System.Random*** to generate numbers for your application
 - <http://bit.ly/1CKgPUf>
- ***System.Random*** is not thread safe

RNGCryptoServiceProvider

- Good random numbers are important in Cryptography
- Random numbers used for creating encryption keys and for hashing
- *System.Random* is not good for non-deterministic random numbers
- *RNGCryptoServiceProvider* is a more secure way to generate random numbers
- *RNGCryptoServiceProvider* is slower to execute than *System.Random*
- Performance is a small trade-off for generating encryption keys



Now it's
your turn!

Step through the code. Make note of the code that Base64 encodes the results.

Hashing

What is Hashing?



What is Hashing?



- It is easy to compute the hash value for any given message
- It is infeasible to generate a message that has a given hash
- It is infeasible to modify a message without changing the hash
- It is infeasible to find two different messages with the same hash

What is Hashing?

MD 5

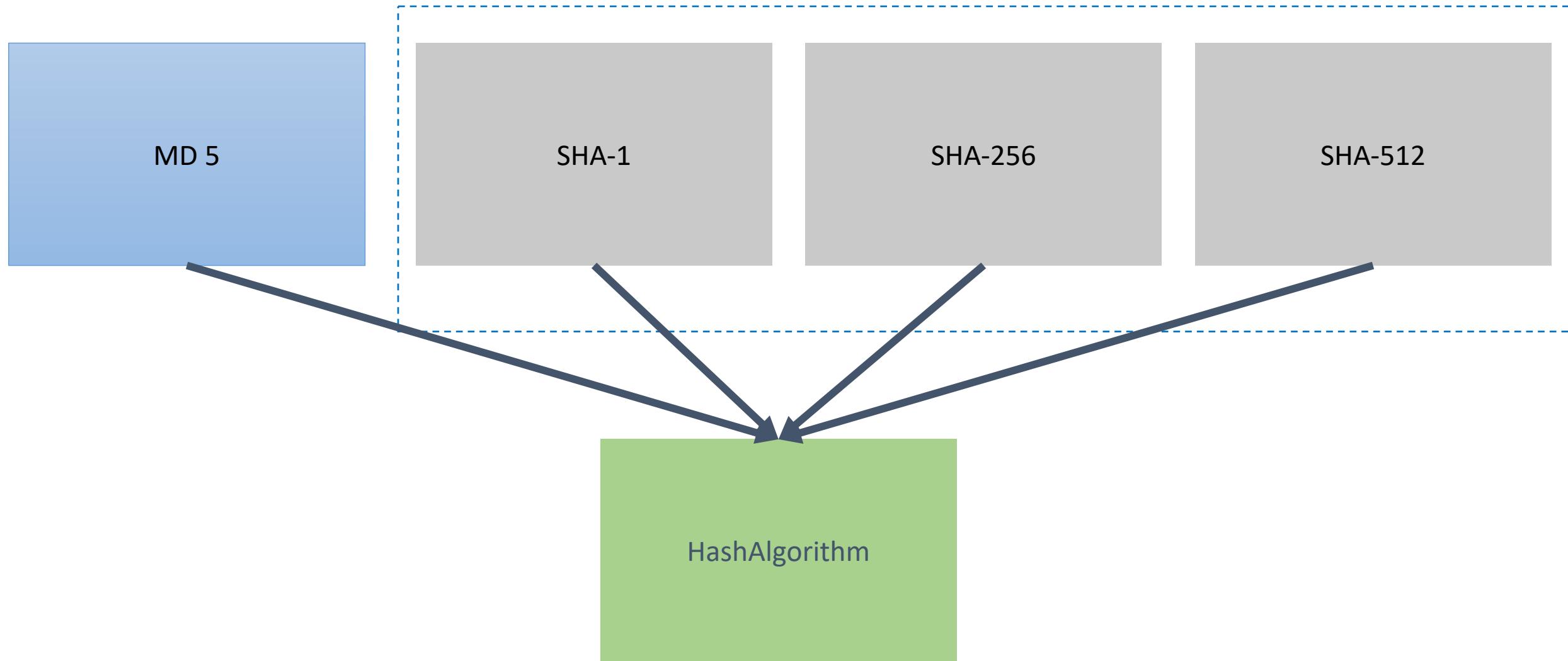
SHA-1

SHA-256

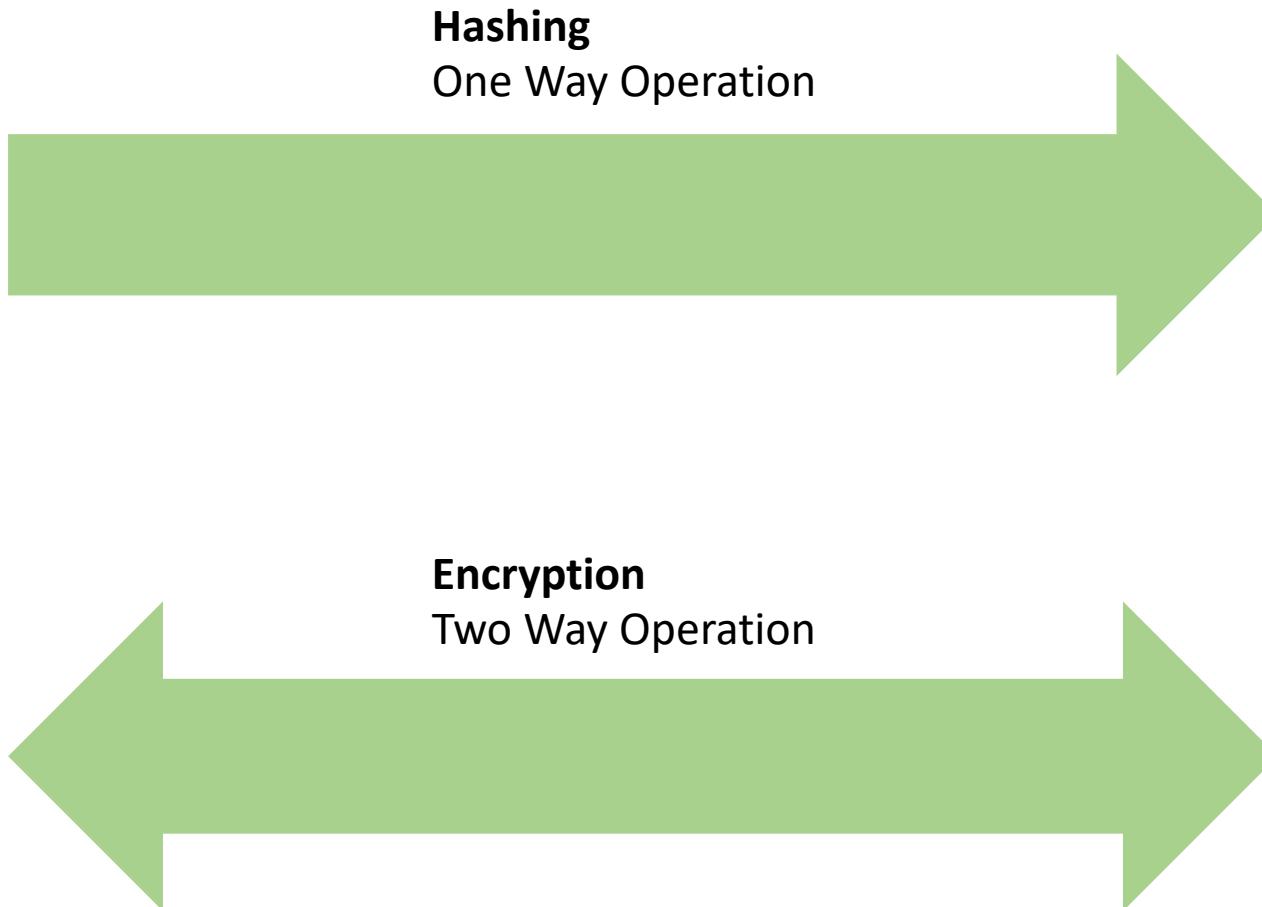
SHA-512

Secure Hash Family

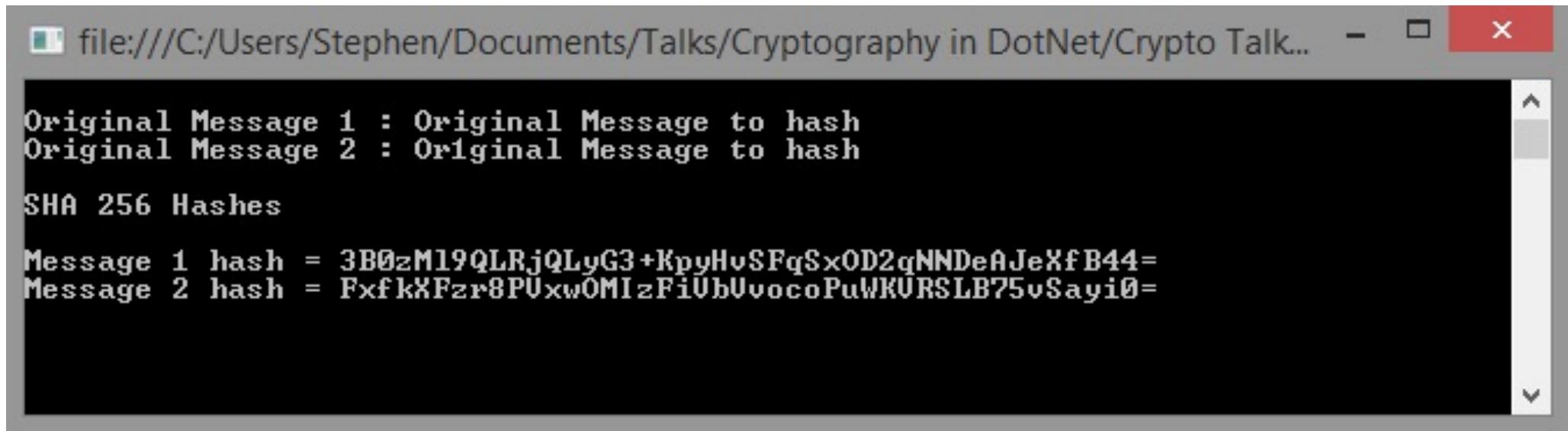
What is Hashing?



What is Hashing?



What is Hashing?



A screenshot of a terminal window titled "file:///C:/Users/Stephen/Documents/Talks/Cryptography in DotNet/Crypto Talk...". The window displays the following text:

```
Original Message 1 : Original Message to hash
Original Message 2 : Original Message to hash

SHA 256 Hashes

Message 1 hash = 3B0zM19QLRjQLyG3+KpyHvSFqSxOD2qNNDeAJeXfB44=
Message 2 hash = FxfkXFzr8PUxwOMIzFiUbUvocoPuWKURSLB75vSayi0=
```

What is Hashing?



```
file:///C:/Users/Stephen/Documents/Talks/Cryptography in DotNet/Crypto Talk...
Original Message 1 : Original Message to hash
Original Message 2 : Original Message to hash
SHA 256 Hashes
Message 1 hash = 3B0zM19QLRjQLyG3+KpyHvSFqSxOD2qNNDeAJeXfB44=
Message 2 hash = FxfkXFzr8PUxwOMIzFiUbUvocoPuWKURSLB75vSayi0=
```

MD5

- Designed by Ron Rivest in 1991 to replace MD4
- Produces a 128 bit (16 byte) hash value
- Commonly used to verify file integrity
- First collision resistance flaw found in 1996
- Recommendation was to move over to the Secure Hash Family
- Further collision resistance problems found in 2004
- Still needed when integrating with legacy systems

Secure Hash (SHA) Family

SHA-1

SHA-2

SHA-3



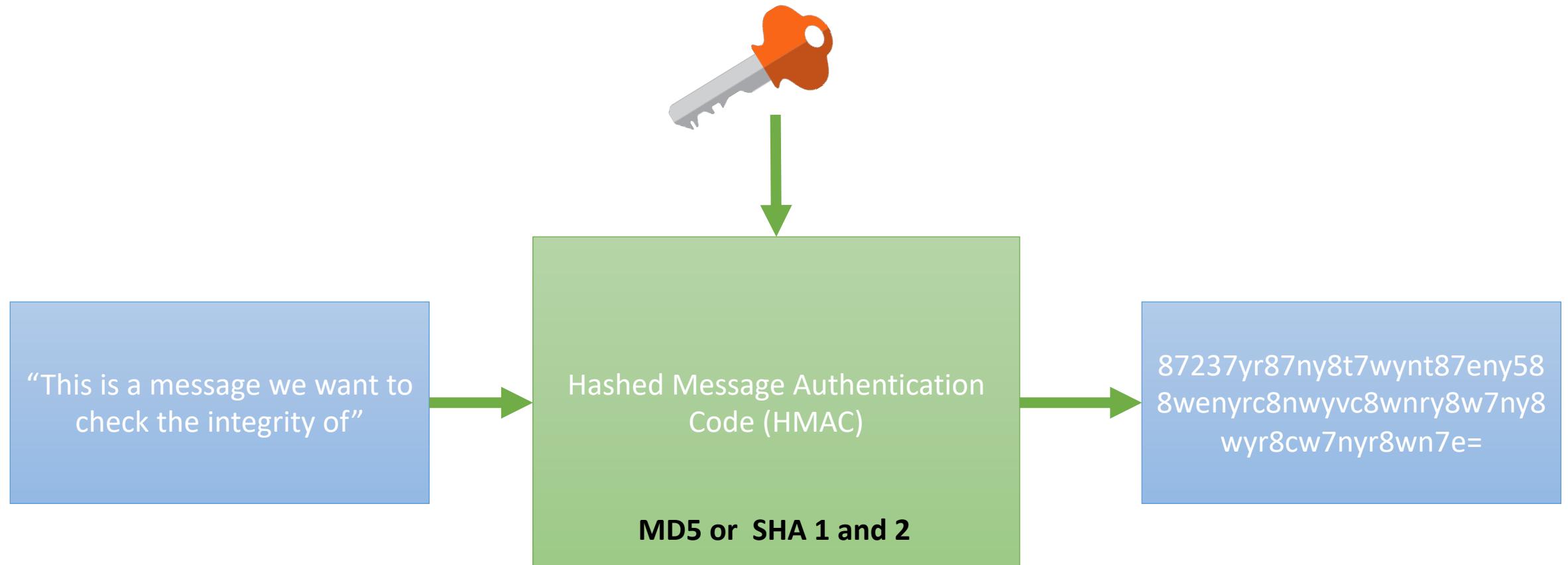
Now it's
your turn!

Step through the hashing code to make sure you understand it.

ADVANCED : Create small console app that will load a file and print it's SHA-256 hash to the console.

Have a 2nd mode where you pass in a filename and a hash and the app says if the file matches the hash.

Hashed Message Authentication Codes





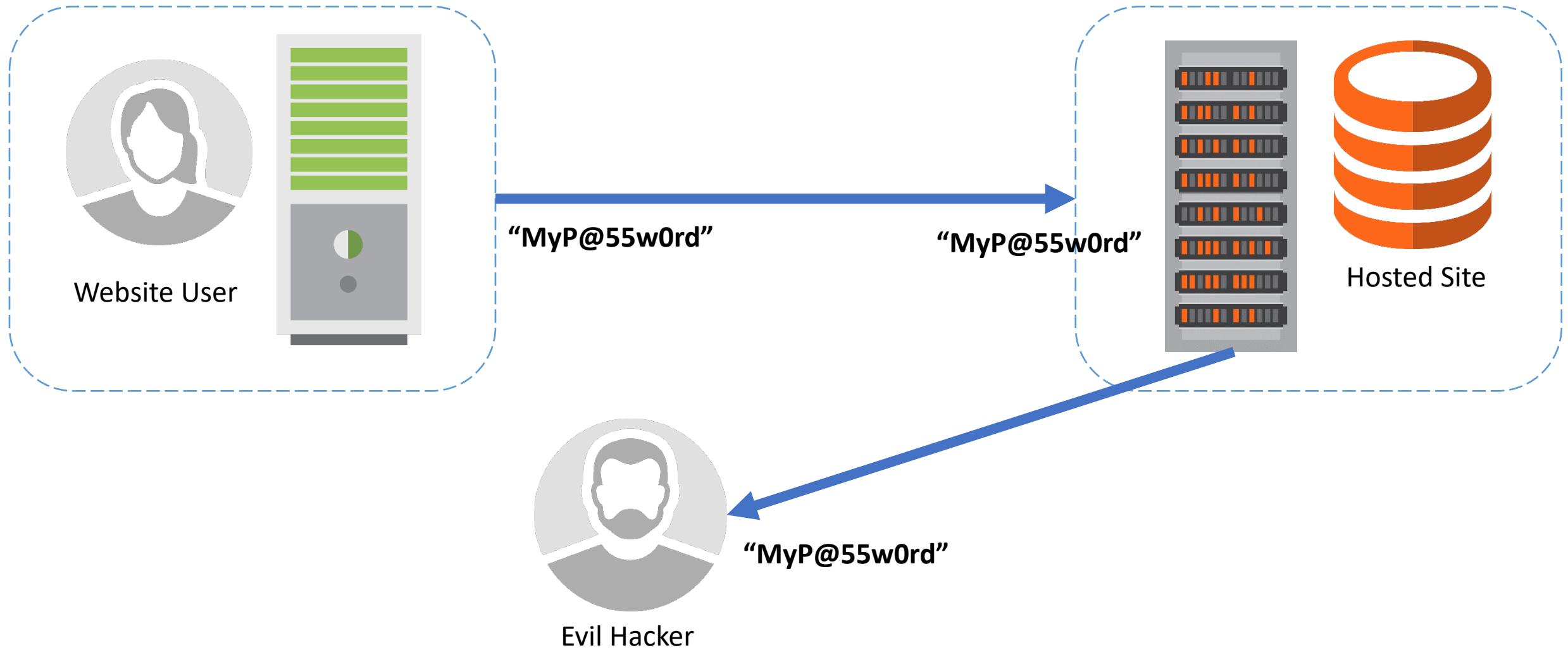
Now it's
your turn!

Step through the hmac code.

Advanced : Take the file hash example and add password support to it.

Storing Passwords

Storing Passwords in the Clear

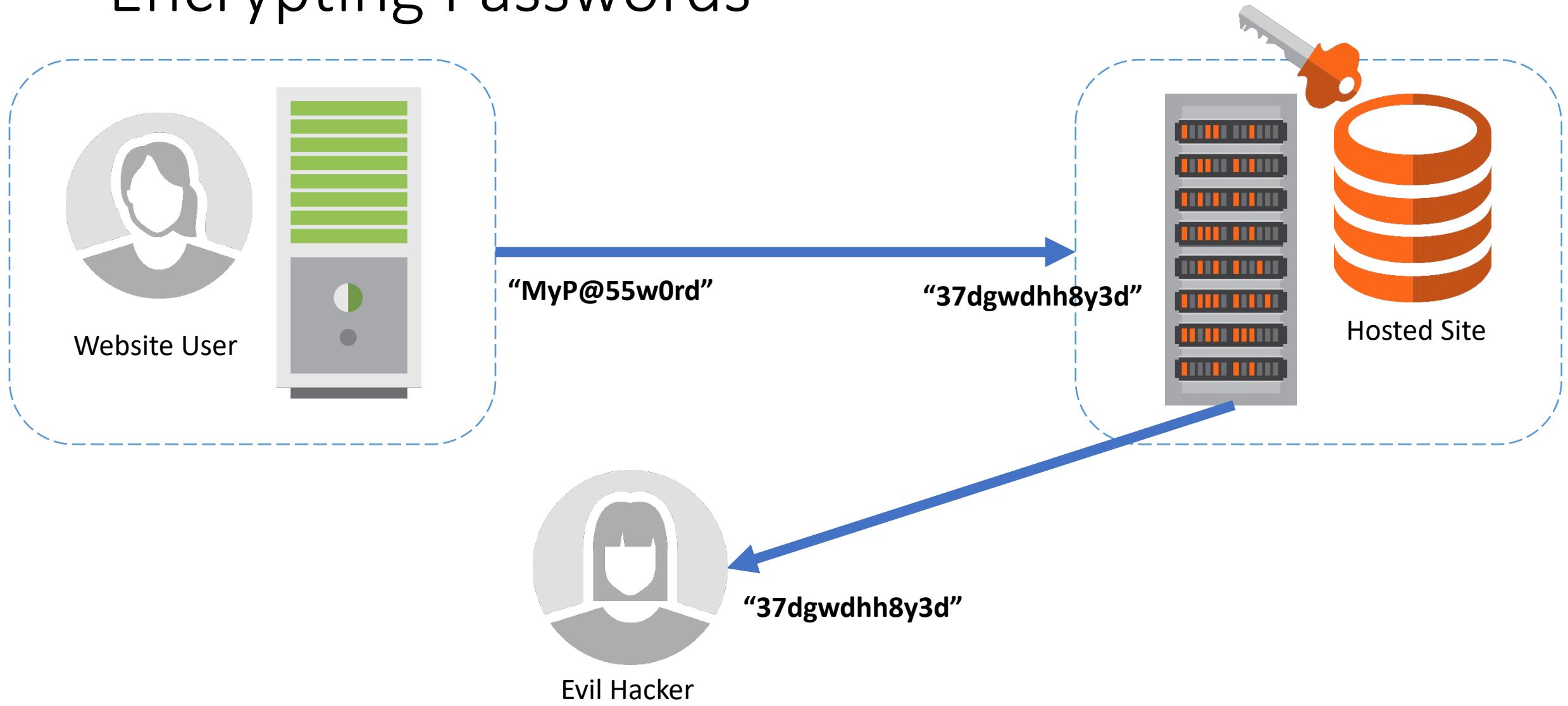




Evil Hacker

- Financial loss
- Reputational damage
- Legal action
- Loss of market share
- Regulatory fines

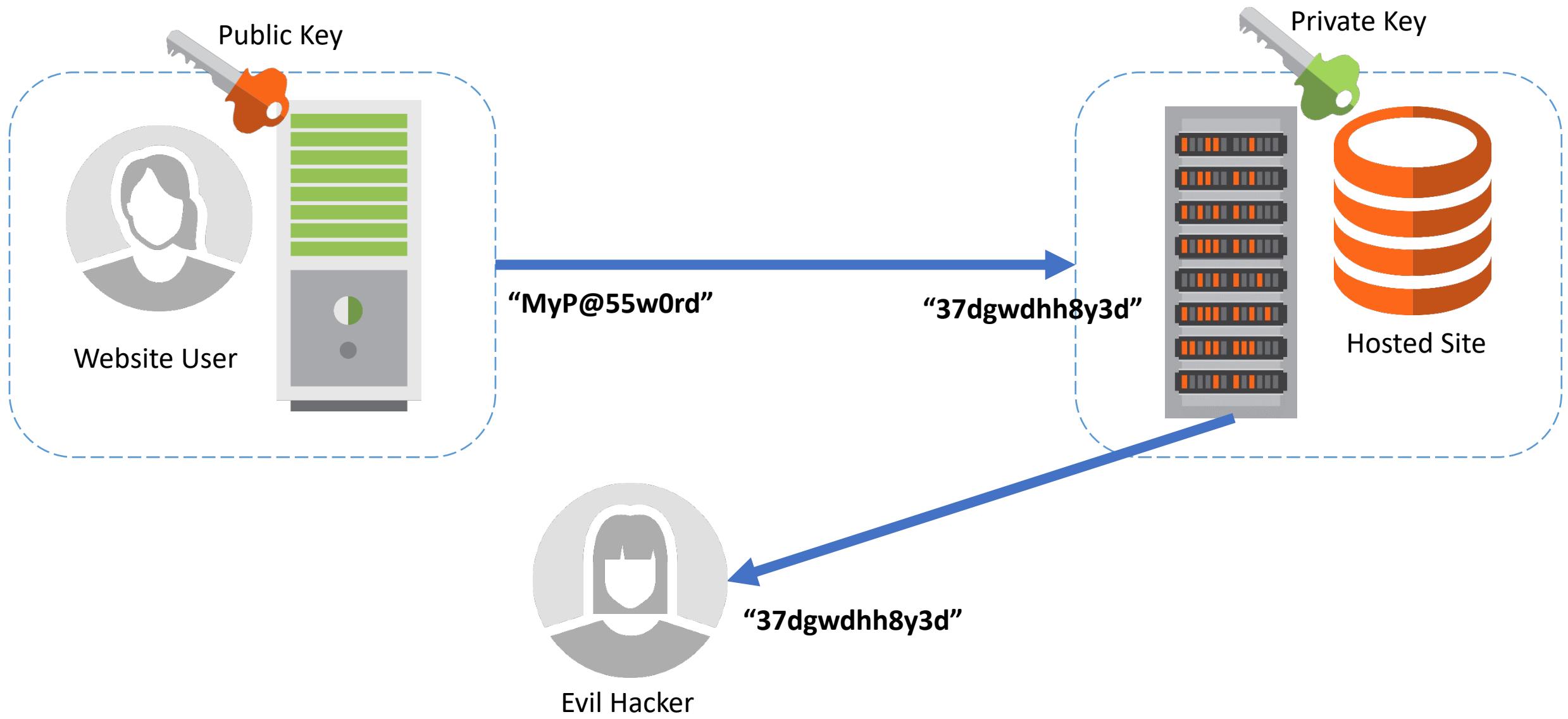
Encrypting Passwords



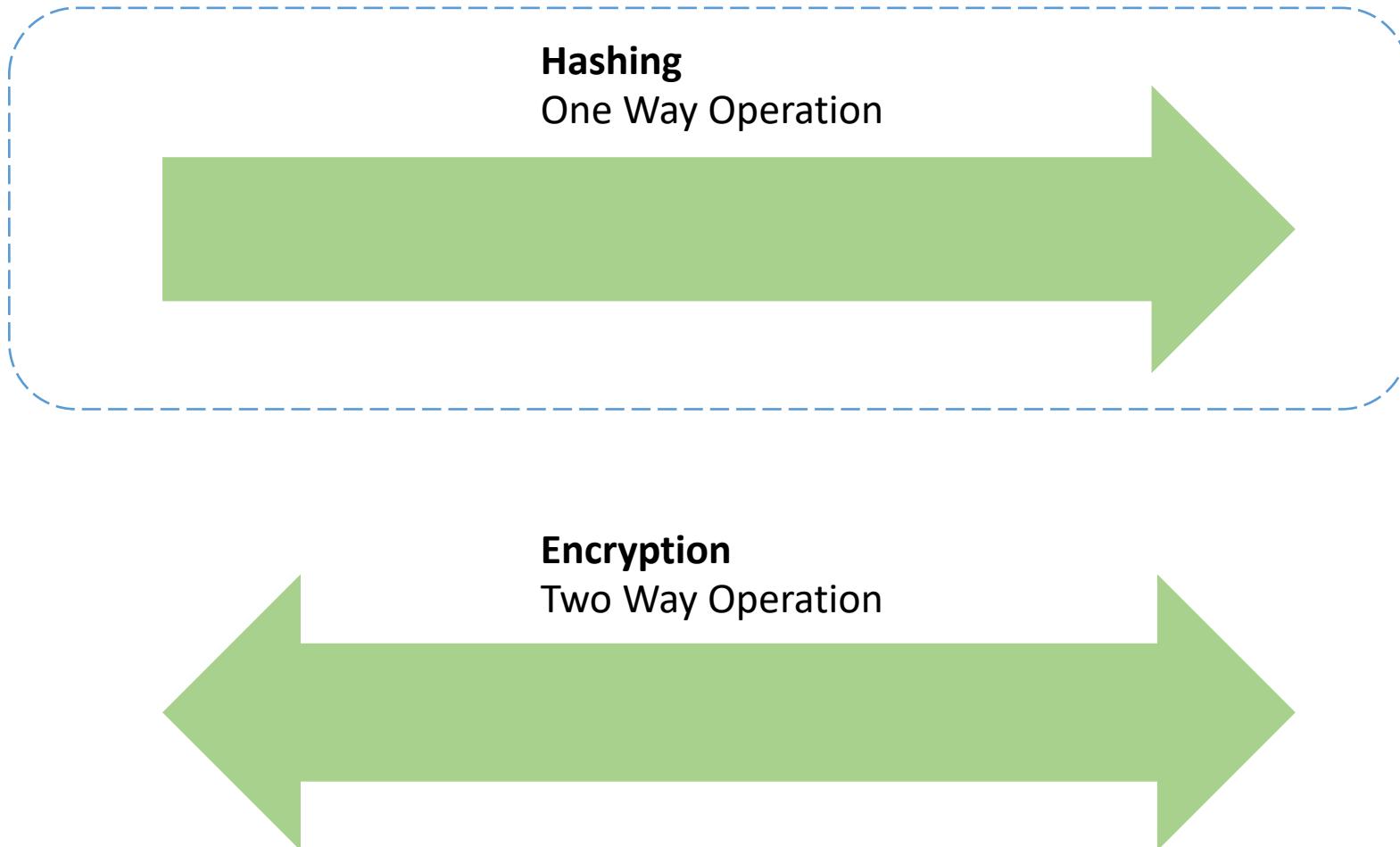


Evil Hacker

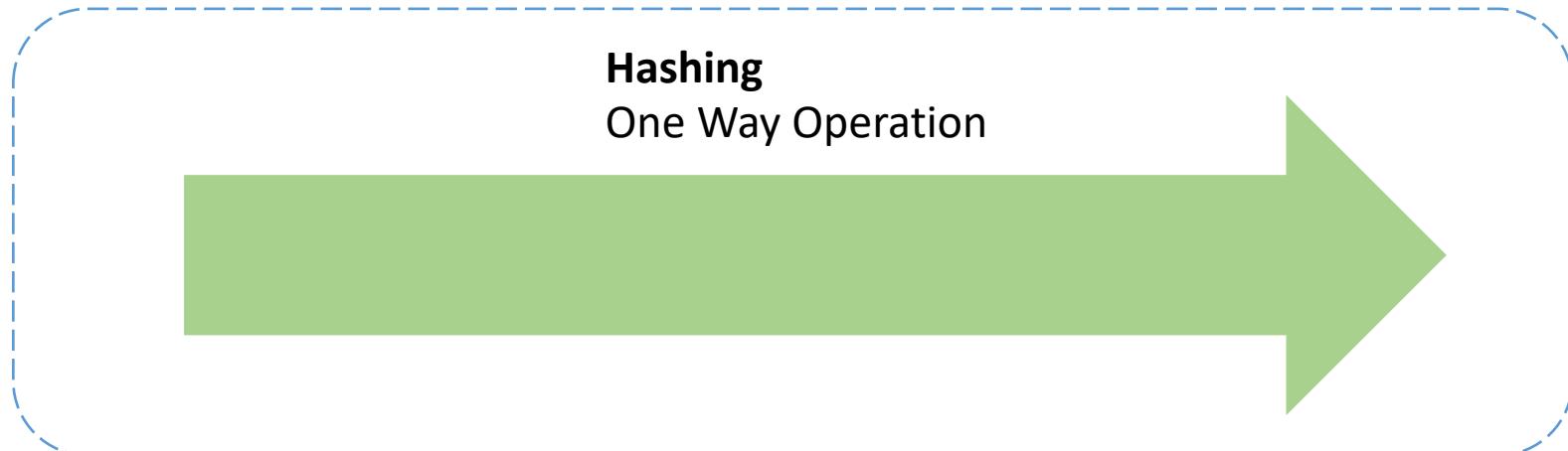
- Key management and storage
- Compromised keys



Using Hashes to Store Passwords

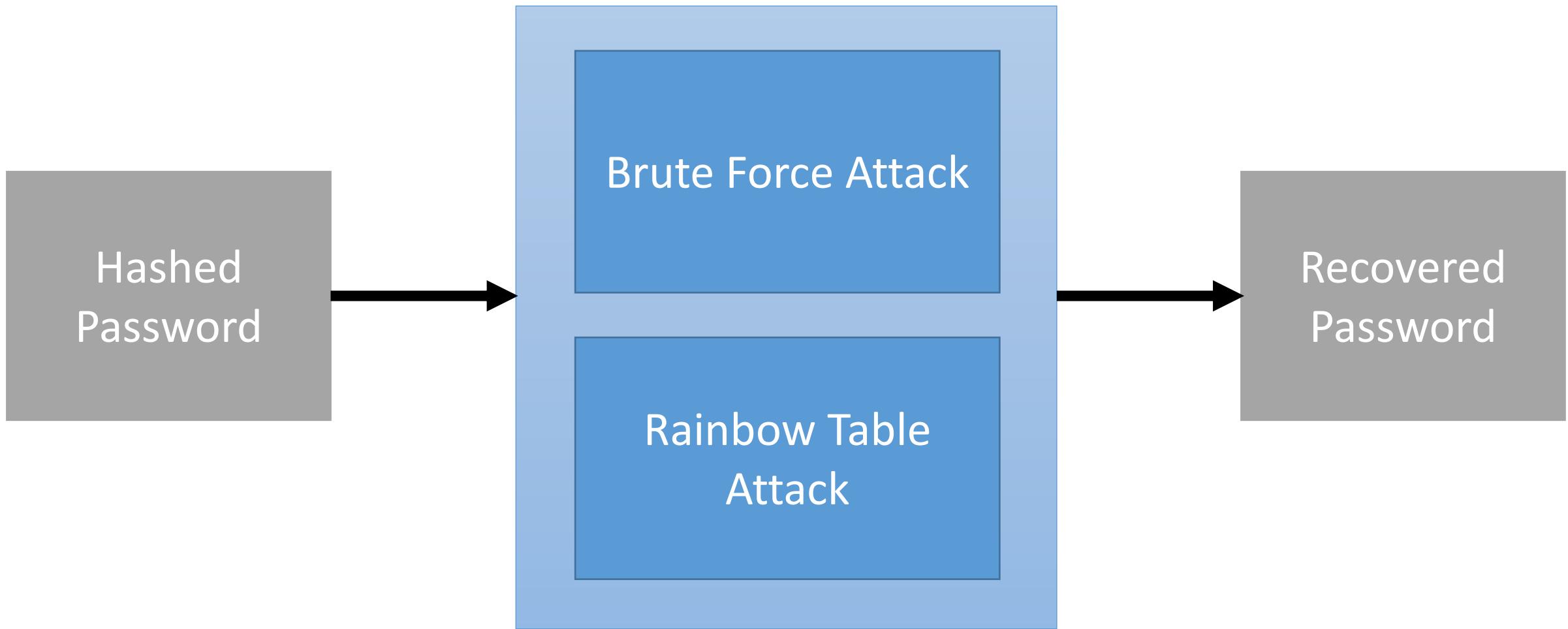


Using Hashes to Store Passwords



- It is easy to compute the hash value for any given message
- It is infeasible to generate a message that has a given hash
- It is infeasible to modify a message without changing the hash
- It is infeasible to find two different messages with the same hash

Using Hashes to Store Passwords



Free Password Hash Cracker

Enter up to 10 non-salted hashes:

89c30f594cd6163641bd1d8e227402b7ac9f7e77650cfef43b9940433e0b37



Supports: LM, NTLM, md2, md4, md5, md5(md5), md5-half, sha1, sha1(bin()), sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+

Hash	Type	Result
89c30f594cd6163641bd1d8e227402b7ac9f7e77650cfef43b9940433e0b37	sha256	Secret69

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.



Advanced Password Recovery

<https://hashcat.net/hashcat/>

```
hashcat -- -bash -- 73x41
[s]tatus [p]ause [b]ypass [c]heckpoint [q]uit => s

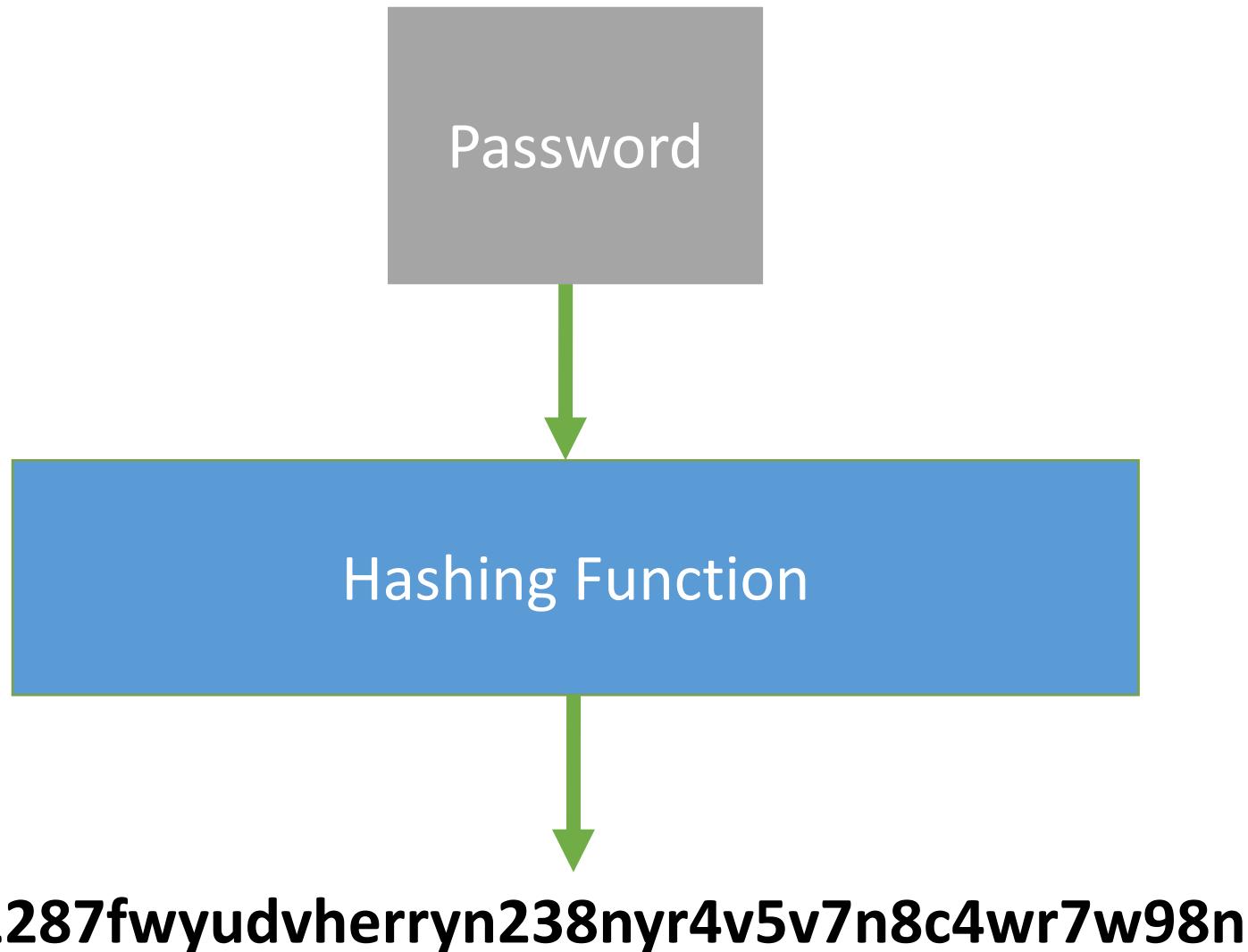
Session.....: hashcat
Status.....: Running
Hash.Type....: MD5
Hash.Target...: example0.hash
Time.Started...: Tue Jan 16 12:09:17 2018 (30 secs)
Time.Estimated.: Tue Jan 16 12:19:27 2018 (9 mins, 40 secs)
Guess.Base....: File (example.dict), Right Side
Guess.Mod.....: Mask (?a?a?a?a) [4], Left Side
Guess.Queue.Base.: 1/1 (100.00%)
Guess.Queue.Mod.: 1/1 (100.00%)
Speed.Dev.#2....: 223.1 MH/s (5.27ms)
Recovered.....: 880/6494 (13.55%) Digests, 0/1 (0.00%) Salts
Recovered/Time...: CUR:0,N/A,N/A AVG:0,0,0 (Min,Hour,Day)
Progress.....: 6681722880/136302297088 (4.90%)
Rejected.....: 0/6681722880 (0.00%)
Restore.Point...: 49152/1048576 (4.69%)
Candidates.#2...: saza22081954 -> 6ome222ajs222

ff891088d44e1b616e0b34a2d3aa7986:2522rowena
fc6fee769723de6186552093c572462e:1644runner
d3186dc073017cff2f4057ff506f89d8:agnes040916
e9b9edb6ff96ec1f05619a77eb9976b2:kobisabine
6867476756abc50656c7bc13821fa470:1192sasha
66343b7ce9c64da8a2ad0fe997d2adb5:heatscreen
89c5d376bdec09fc9d4e894c96fad28:britserega
de28d304f42e8793a081a503d9e65738:6892serge
91eae0025ba47a1737d9afa119cdf2a8:ftwbserver
662d471cfaf58fae3536b8a53ec1f068:coleshay
8ec6963eb3a70aa0b8fff7bdedc3d6f6:antiskill
c60d31ac627828462e81d60677a54a42:8492smirnoff
3278113d5630945d4a69f545a8555812:lelespecial
9fad03ddaa4ce1aa4e4610f2a9bc41cc6:asenstefan1
dc6cce17cfcfd209f1cb04ed3a3fea16:skittoctoc
a5707a73d8bd52292c497f81337b8ee9:yarrtorrents
d2e12757b34d895815951023b962e77c:1chotrains
341bc61f1169d907675ab99925f15451:venetweety
3ee989c9765bb731281d98a100377e9e:0581vanessa
dadf7e87a37ee4282c2533a7d825a27f:5456veronique
3e7b9998bea713ae19f11241da9fc960:77nevidal
```

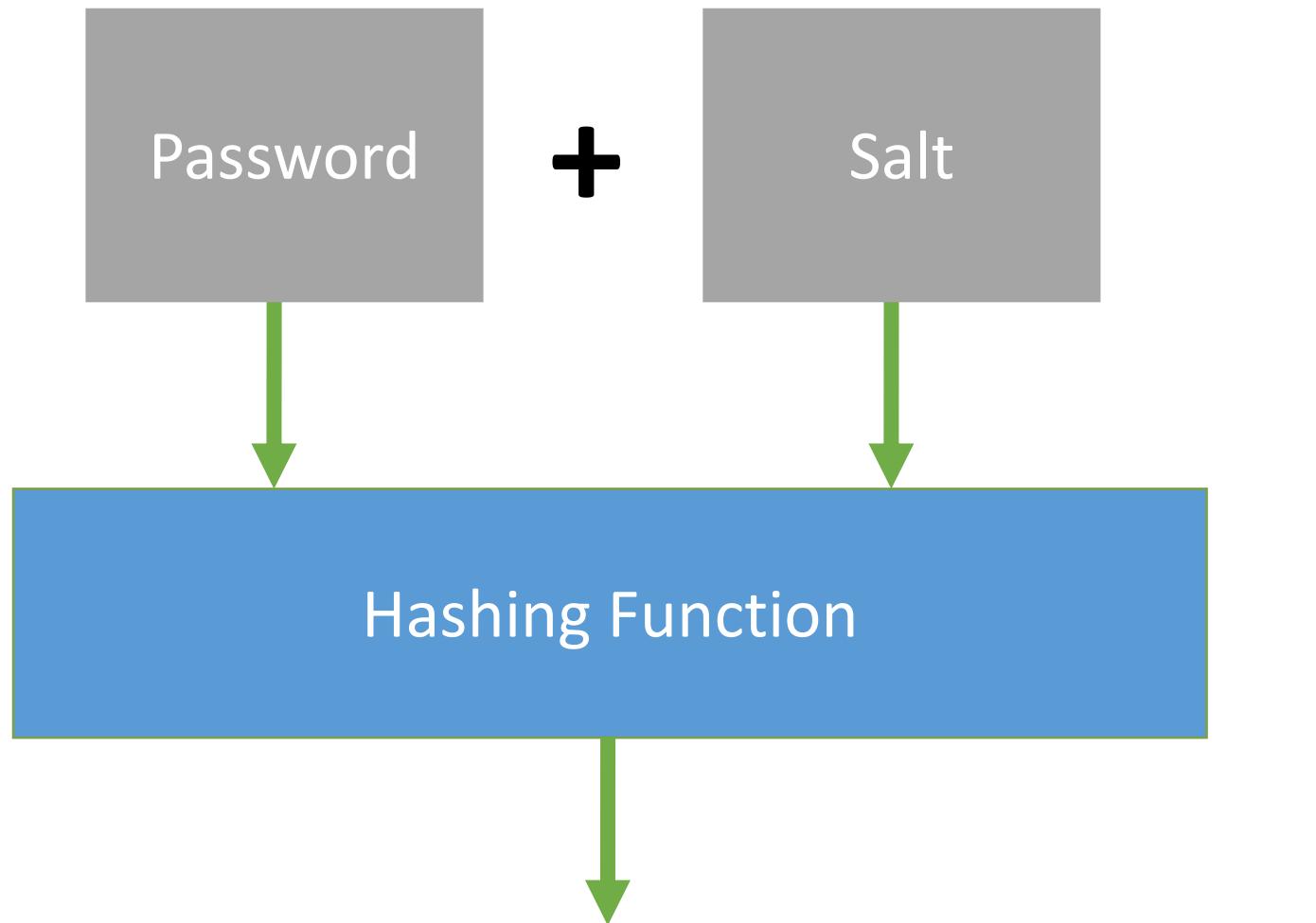
HashCat

- Uses GPU's to crack passwords
- Most commonly used with PC's and high end GPU's
- Exe's for Windows available at <https://hashcat.net/hashcat/>
- For Mac you have to build from source
 - <https://www.phillips321.co.uk/2016/07/09/hashcat-on-os-x-getting-it-going/>

Using Hashes to Store Passwords



Using Hashes to Store Passwords



32287fwyudvherryn238nyr4v5v7n8c4wr7w98n



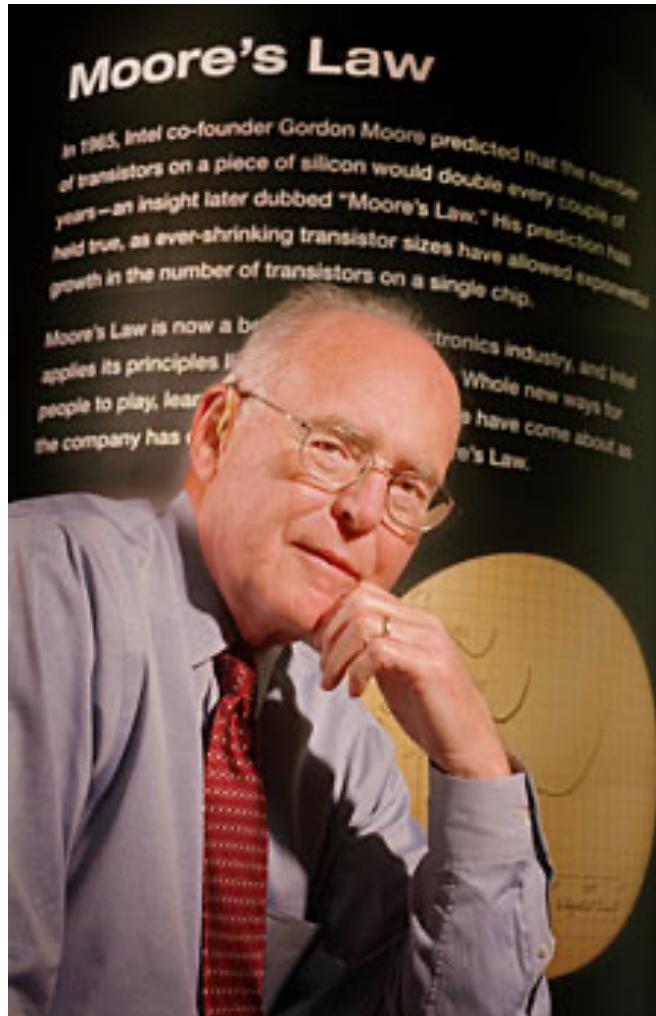
Now it's
your turn!

Generate a salt

Hash the password along with the salt

Display the hashed password and salt
to the console as base 64 strings

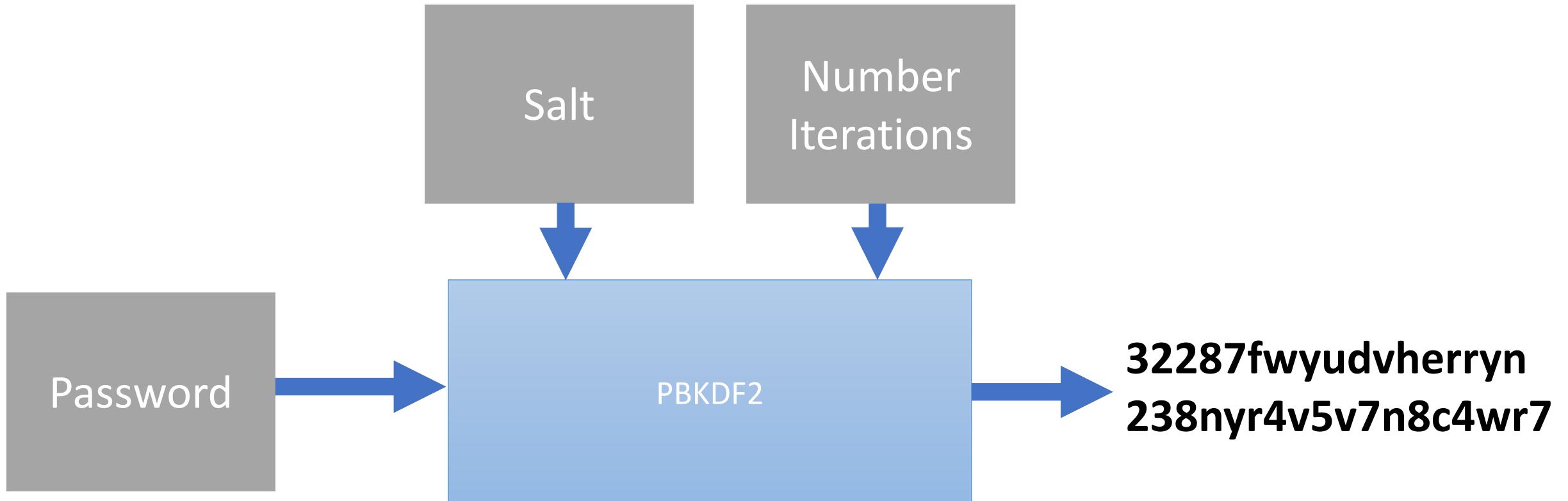
Password Based Key Derivation Functions



Password Based Key Derivation Functions

- Password Based Key Derivation Function (PBKDF2)
- RSA Public Key Cryptographic Standards (PKCS #5 Version 2.0)
- Internet Engineering Task Force RFC 2898 Specification

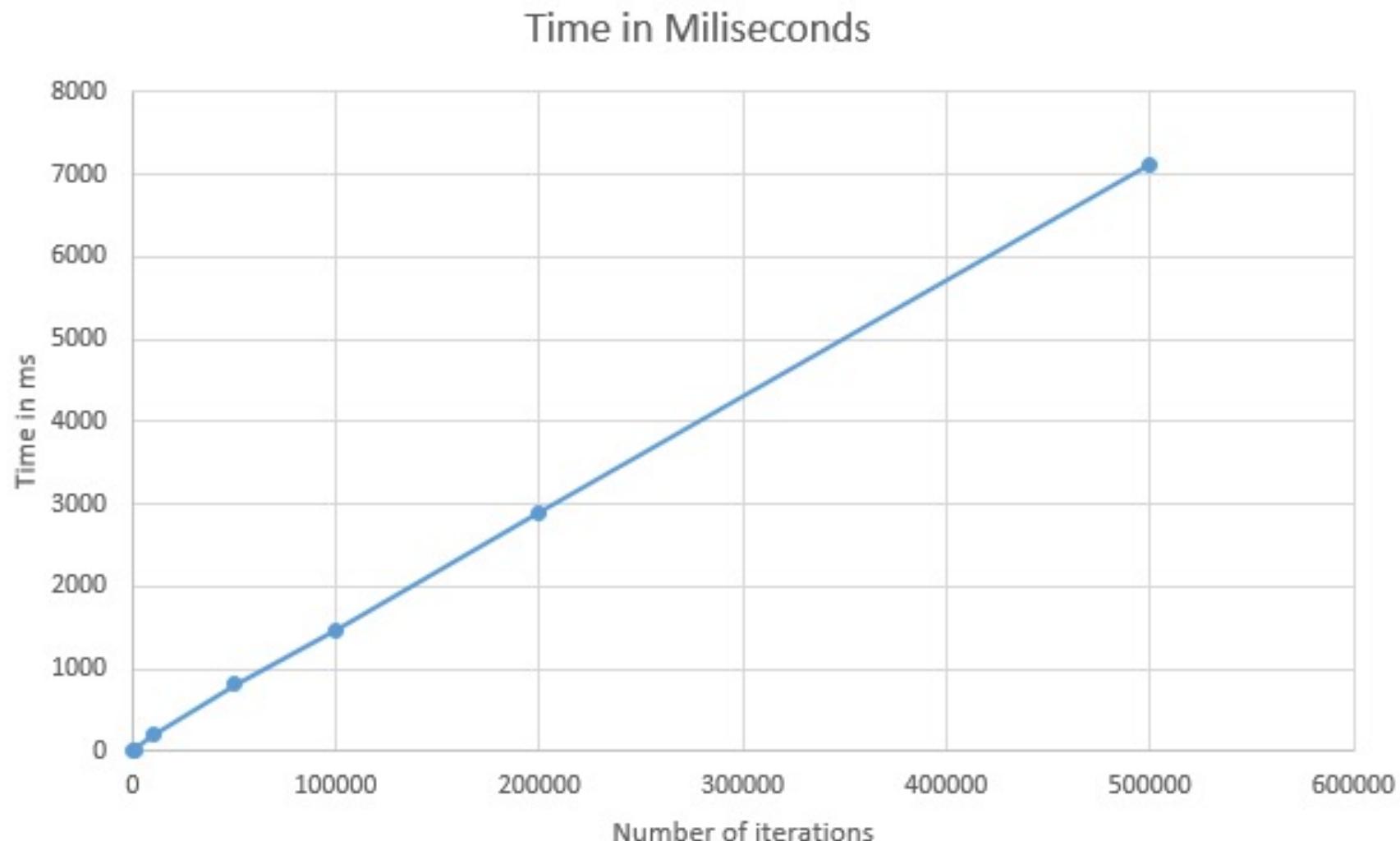
Password Based Key Derivation Functions



Password Based Key Derivation Functions

- Good default is 50,000 iterations
- Balance number of iterations with acceptable performance
- Ideally double number of iterations every 2 years

Password Based Key Derivation Functions





Now it's
your turn!

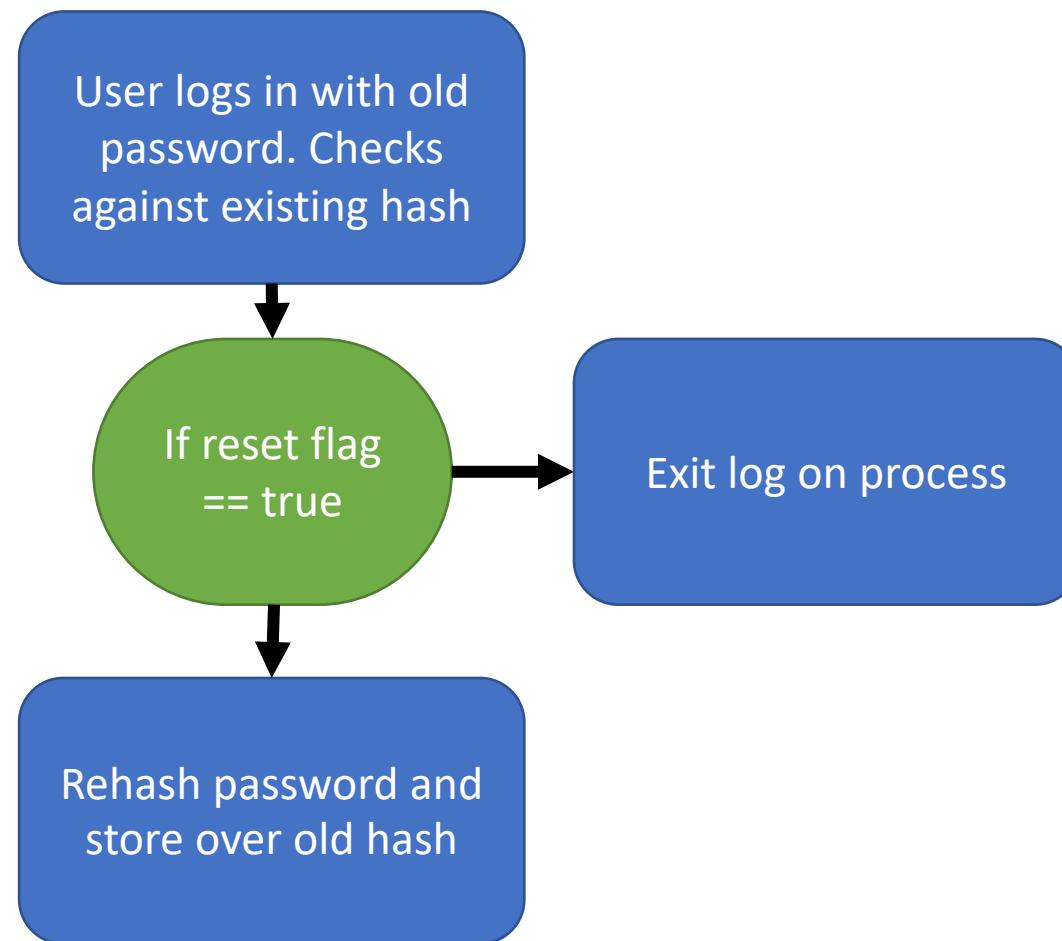
Step through the included code.

Experiment with different iteration values.

What value would be acceptable for your systems?

Upgrading Password Hashes

Moving from one hash algorithm to another



Upgrading Password Hashes

Invalidate individual users

Gradual upgrade of passwords

Invalidate all or group users

Faster update rate

Secure Password Storage



The image shows a close-up of a woman's face. She has dark brown hair and is wearing a gold ring on her left hand. Her right index finger is held vertically against her lips, a gesture commonly used to indicate that something should be kept secret or quiet.

ASHLEY MADISON®

Life is short. Have an affair.®

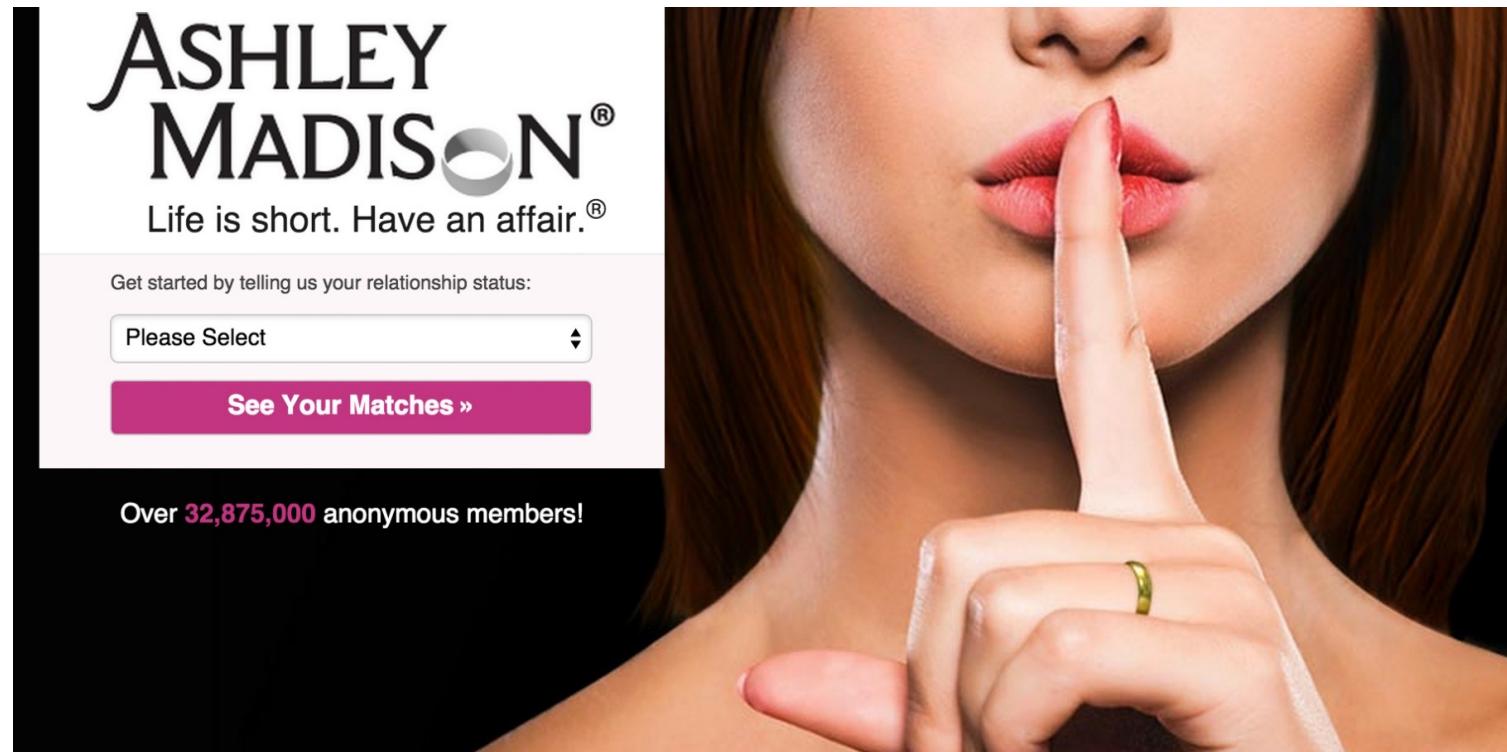
Get started by telling us your relationship status:

Please Select

See Your Matches »

Over **32,875,000** anonymous members!

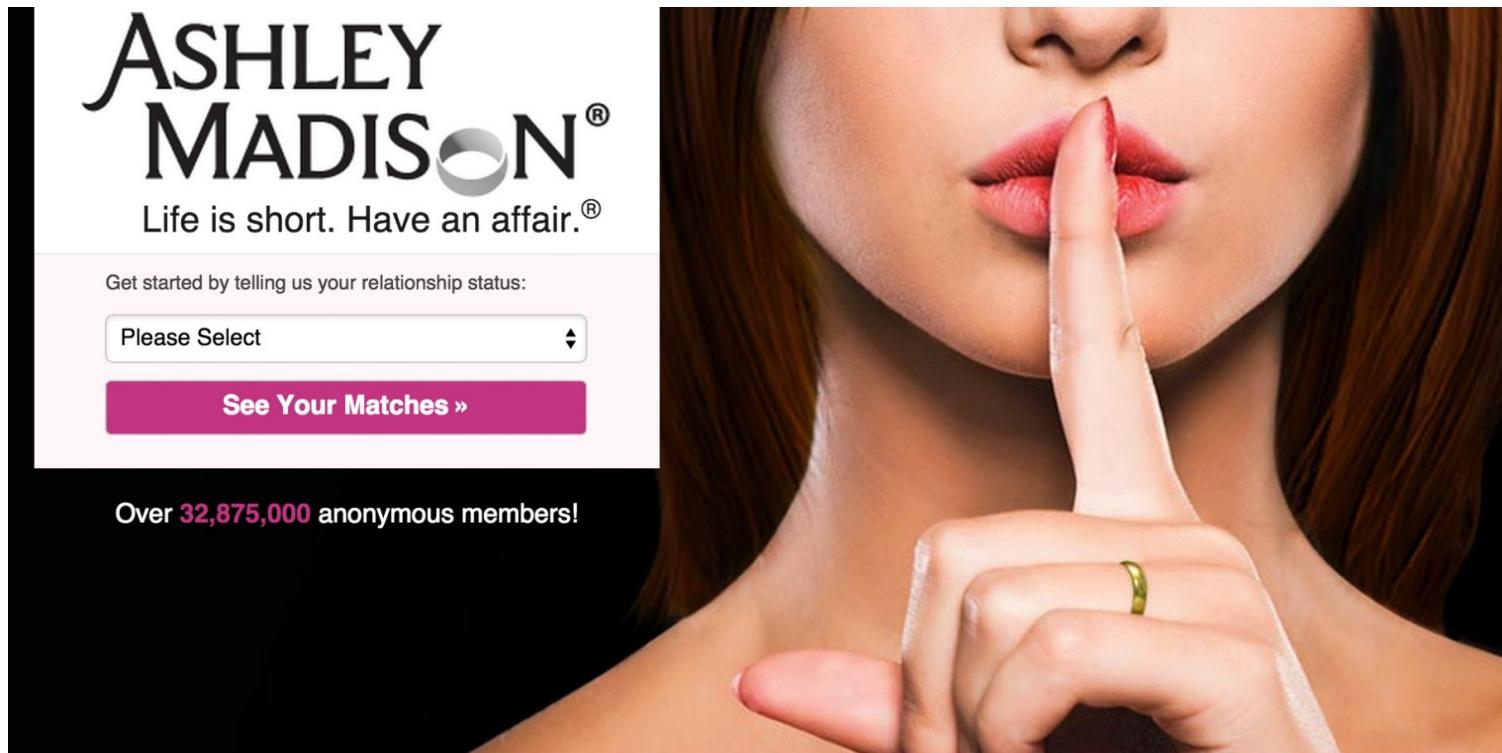
Secure Password Storage



Used BCrypt for Passwords



Secure Password Storage



Used BCrypt for Passwords
Stores local token with MD5 hashed password



Secure Password Storage

The screenshot shows the Ars Technica UK homepage. At the top, there's a navigation bar with links for 'MAIN MENU', 'MY STORIES: 25', and 'FORUMS'. The main headline reads 'RISK ASSESSMENT / SECURITY & HACKTIVISM' in bold capital letters. Below it is a sub-headline: 'Once seen as bulletproof, 11 million Ashley Madison passwords already cracked'. A small blurb states: 'Programming errors make 15.26 million accounts orders of magnitude faster to crack.' The author is Dan Goodin (US) from Sep 10, 2015, at 4:25pm CEST. There are social sharing icons for Facebook, Twitter, and Email, along with a link count of 144. To the left of the text is a historical black and white photograph of two men in early 20th-century clothing standing on a dirt path. One man is pointing a revolver towards the other, who is looking back. A large plume of smoke or dust is visible behind them. In the background, there are trees and some industrial structures like power poles. On the right side of the page, there are several sidebar sections: 'LATEST FEATURE STORY' with a thumbnail of a grand hall and the title 'Open access: All human knowledge is there—so why can't everybody access it?', 'STAY IN THE KNOW WITH' social media links, 'LATEST NEWS' with a link to 'EU REFERENDUM' about Brexit voter registration, and 'GAME ON' with a link to 'Import tax on high-tech goods to be wiped out after MEPs vote for cuts'.

<http://bit.ly/1U7scgd>

What We Have Covered so Far?

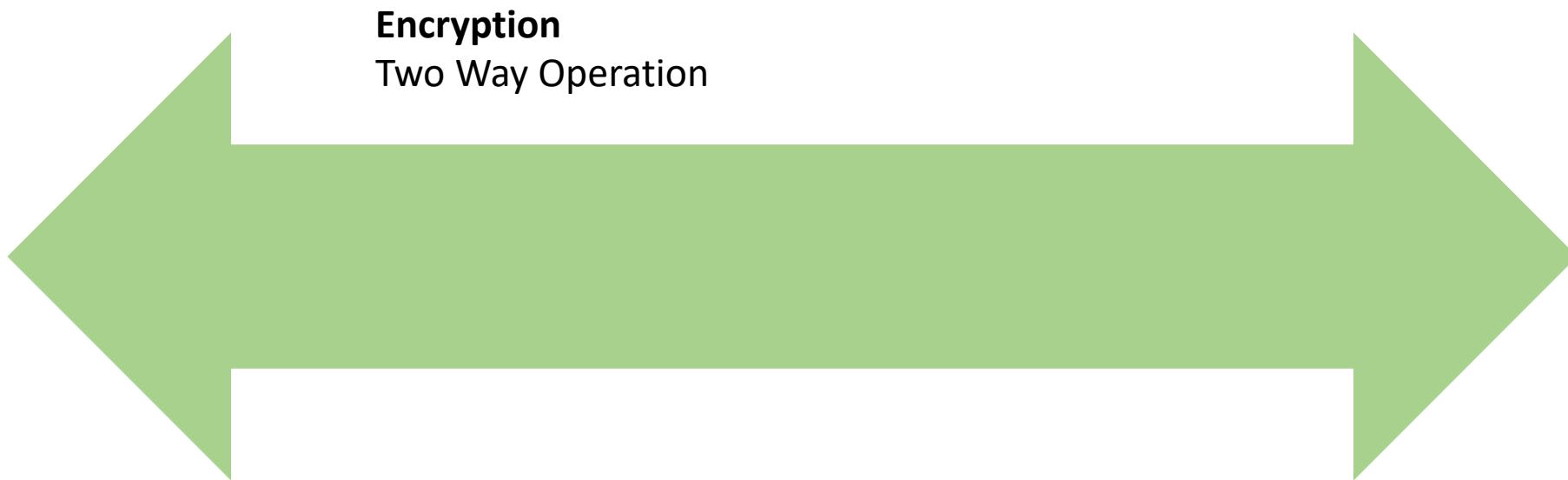
Secure Random
Number Generation

Hashing of Data

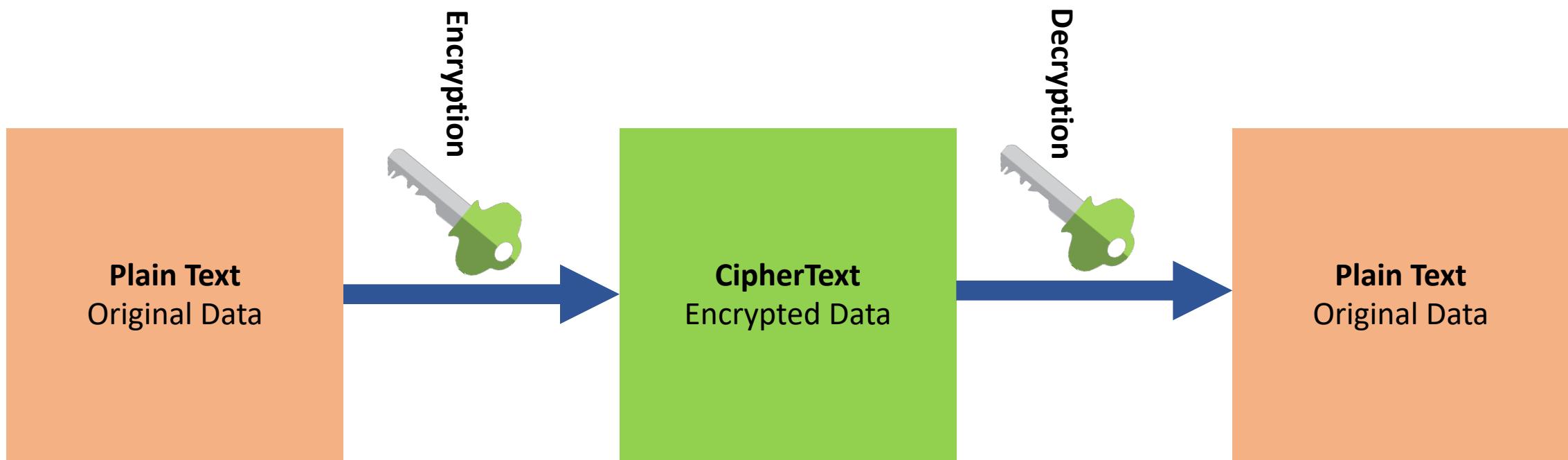
Secure Password
Storage

Symmetric Encryption

What is Symmetric Encryption?



What is Symmetric Encryption?



Symmetric Encryption Advantages



- Extremely secure
- Relatively fast

Symmetric Encryption Disadvantages

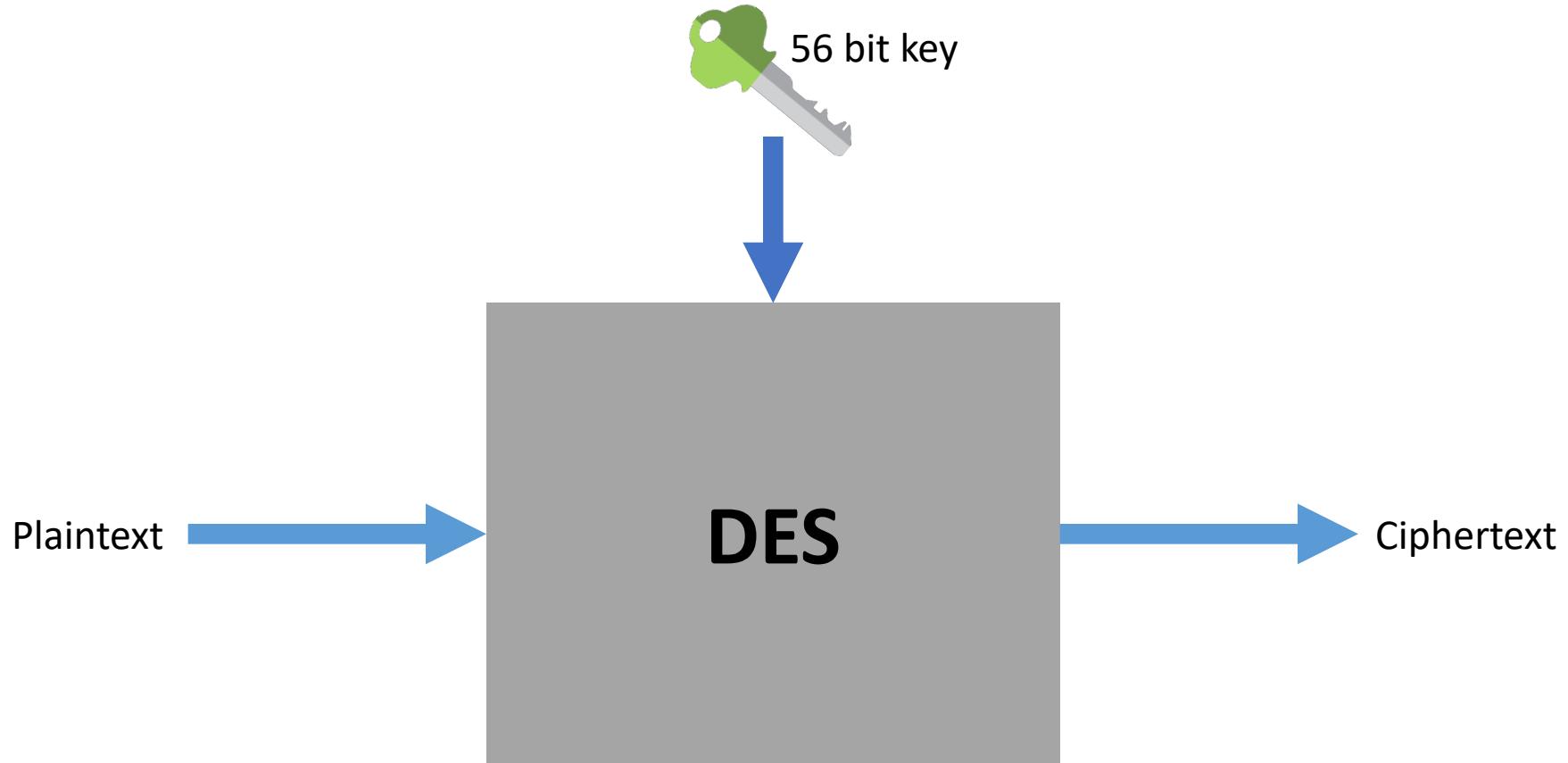


- Key sharing
- More damage if compromised

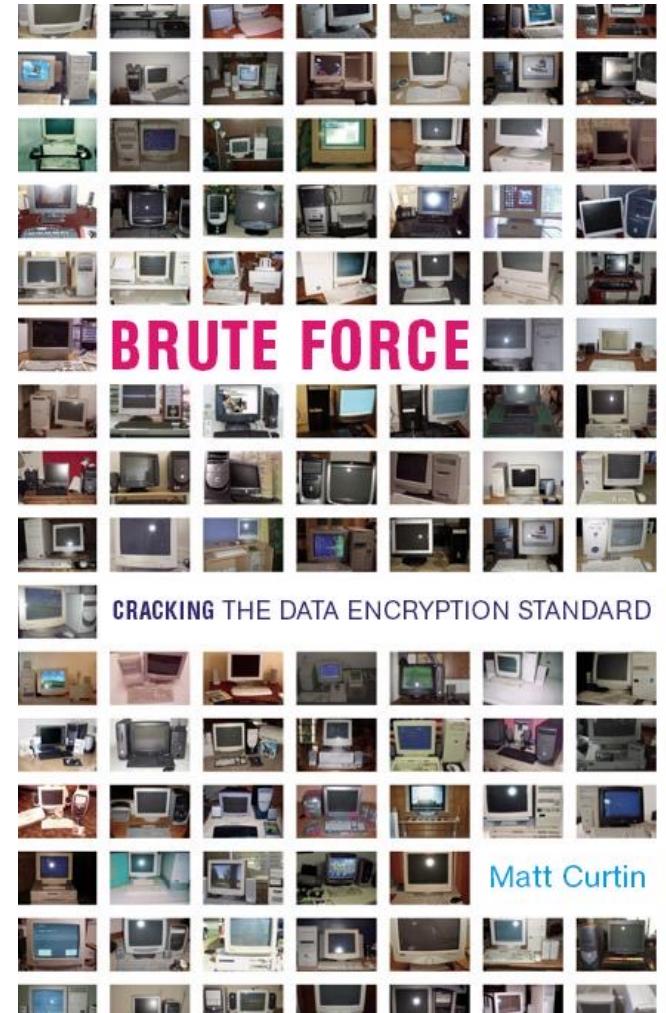
The History of DES and Triple DES?

- Data Encryption Standard (DES) was developed in early 1970's at IBM
- Submitted to the National Bureau of Standards for approval
- Approved at Federal Information Processing Standard 46 (FIPS 46)
- Consultation with the National Security Agency (NSA)
- Provide security for the unclassified electronic data for the US government

The History of DES and Triple DES?



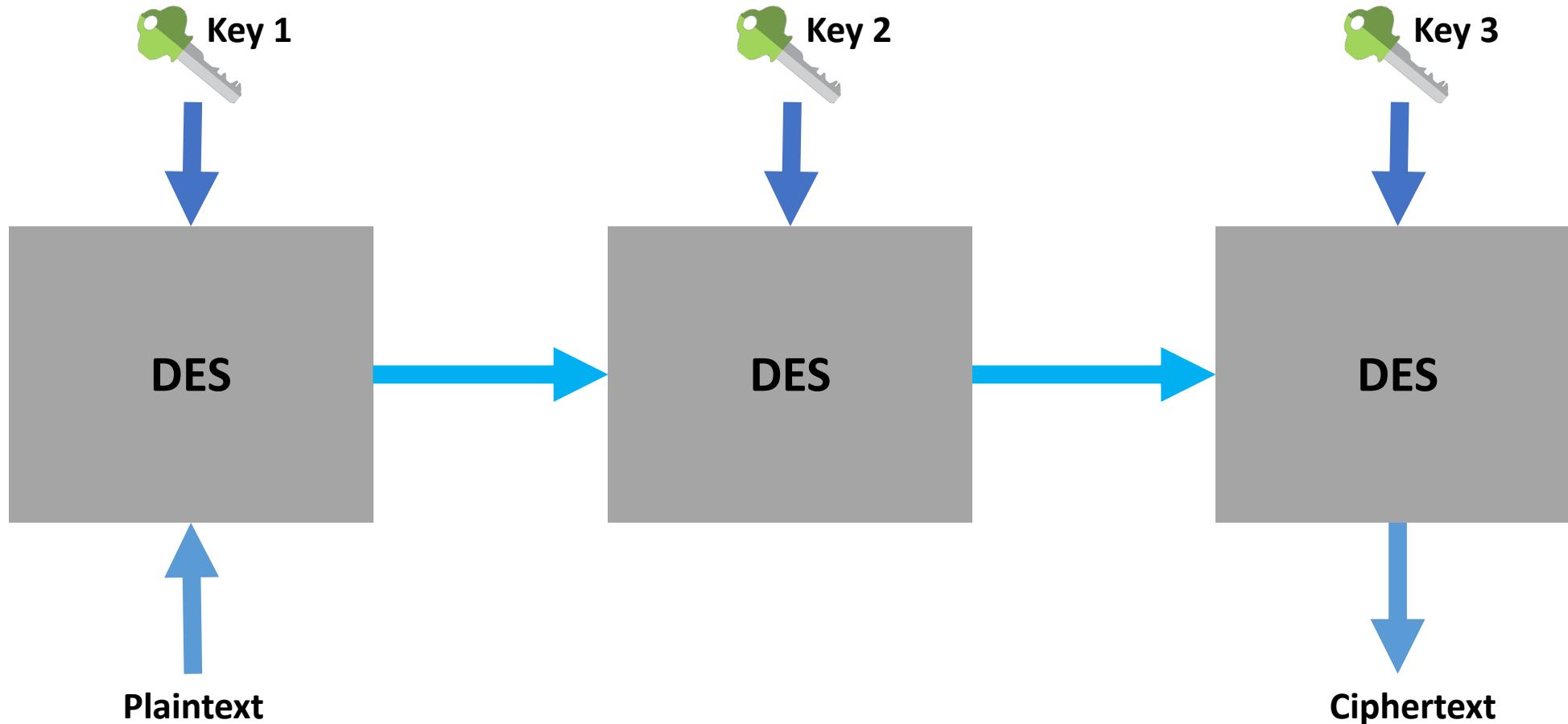
The History of DES and Triple DES?



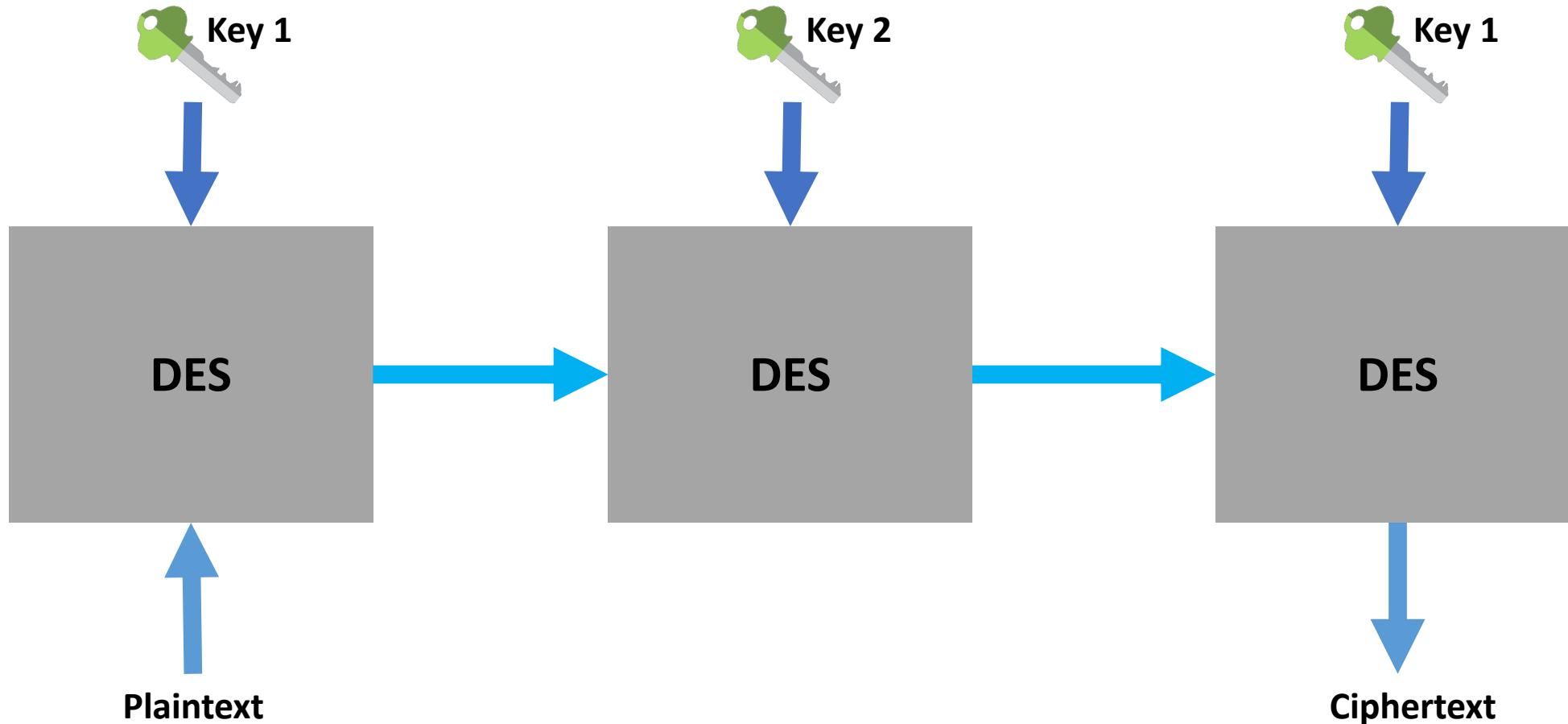
The History of DES and Triple DES?

- A new variant designed called Triple DES
- A simple way to increase key size without redesigning a new cipher
- Many former DES users now use Triple DES
- Triple DES involved applying DES three times with 2 or 3 different keys
- Triple DES was regarded as adequately secure, although it is quite slow

The History of DES and Triple DES? - 3 Key



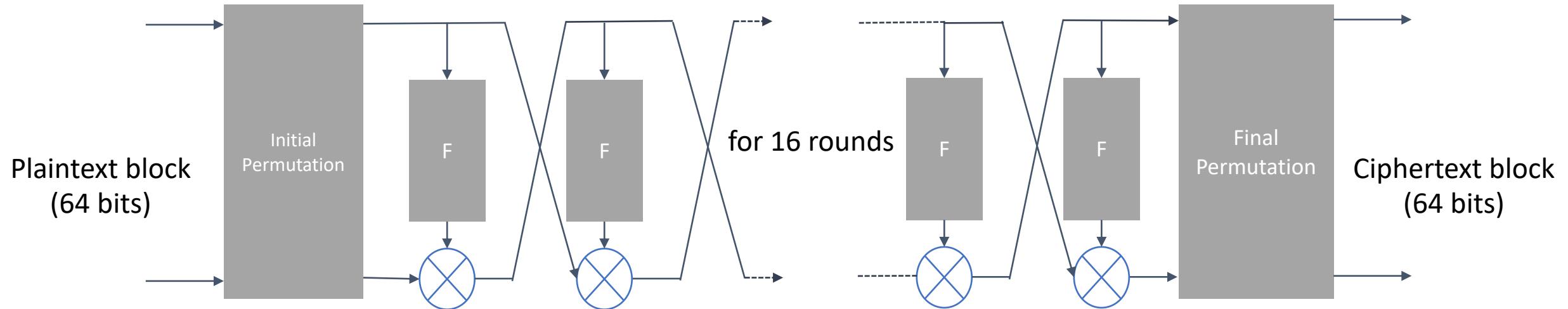
The History of DES and Triple DES? - 2 Key



How Does DES and Triple DES Work?

- DES is a block cipher that transforms plaintext into ciphertext
- DES uses a block size of 64 bits
- Uses a 64 bit key but only 56 bits are used by the algorithm
- Supports different modes of operation

How Does DES and Triple DES Work?



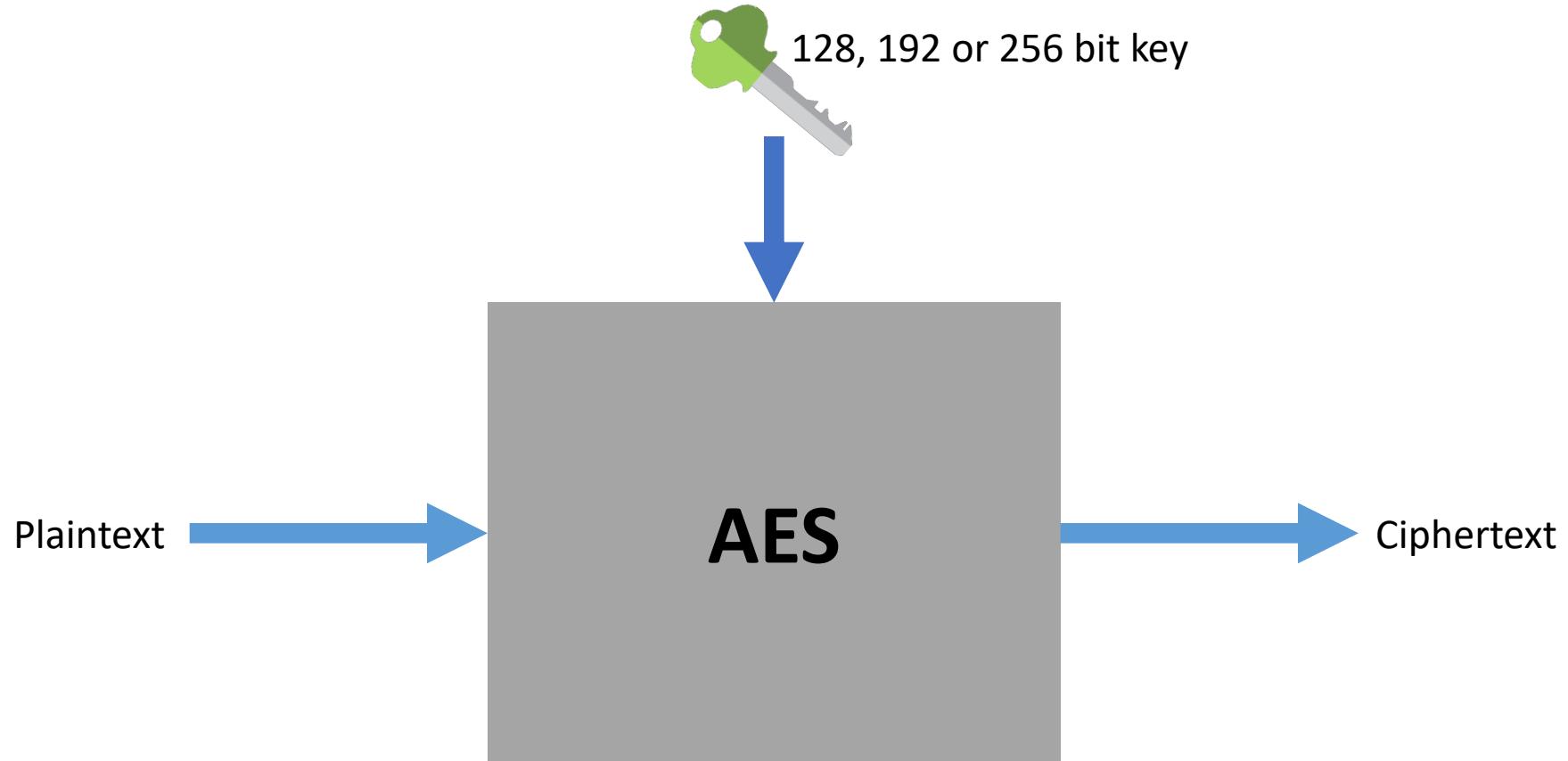
How Does DES and Triple DES Work?

- DES uses a key schedule for encryption
- The key schedule generates sub keys for each of the 16 rounds
- 56 bit key is split in half
- For each round sub keys are bit rotated left
- Decryption key schedule is similar but in reverse order

The History of AES

- Advanced Encryption Standard adopted by NIST in 2001
- Selected by a contest to replace the Data Encryption Standard (DES)
- AES is based on the Rijndael cipher
- Rijndael is a family of ciphers with different key and block sizes

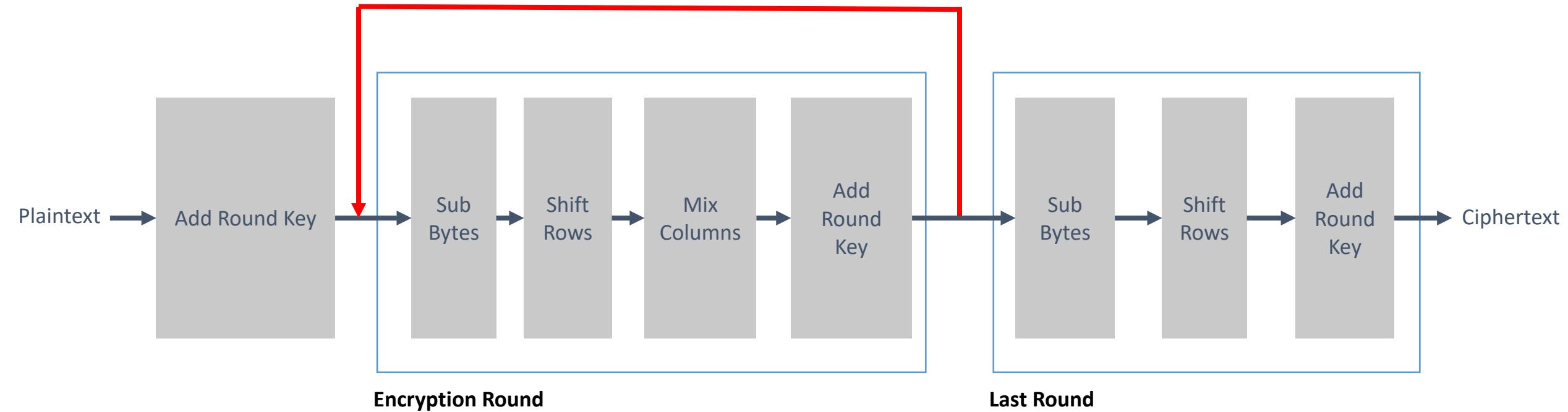
The History of AES



How Does AES Work?

- Unlike DES, AES does not use a Feistel network
- Uses 128 bit block size and 128, 192 or 256 bit keys
- Based on a design known as a substitution – permutation network
 - S-Box performs substitutions
 - P-Box performs bit shuffling to transpose bits across S-Box inputs

How Does AES Work?



10 rounds of repetition for **128-bit** keys
12 rounds of repetition for **192-bit** keys
14 rounds of repetition for **256-bit** keys

How Does AES Work?

- AES key lengths are 128, 192 or 256 bits
- Every key is expanded so a separate sub-key can be utilized for every round
- The Number of rounds of AES generally depends on the length of the key

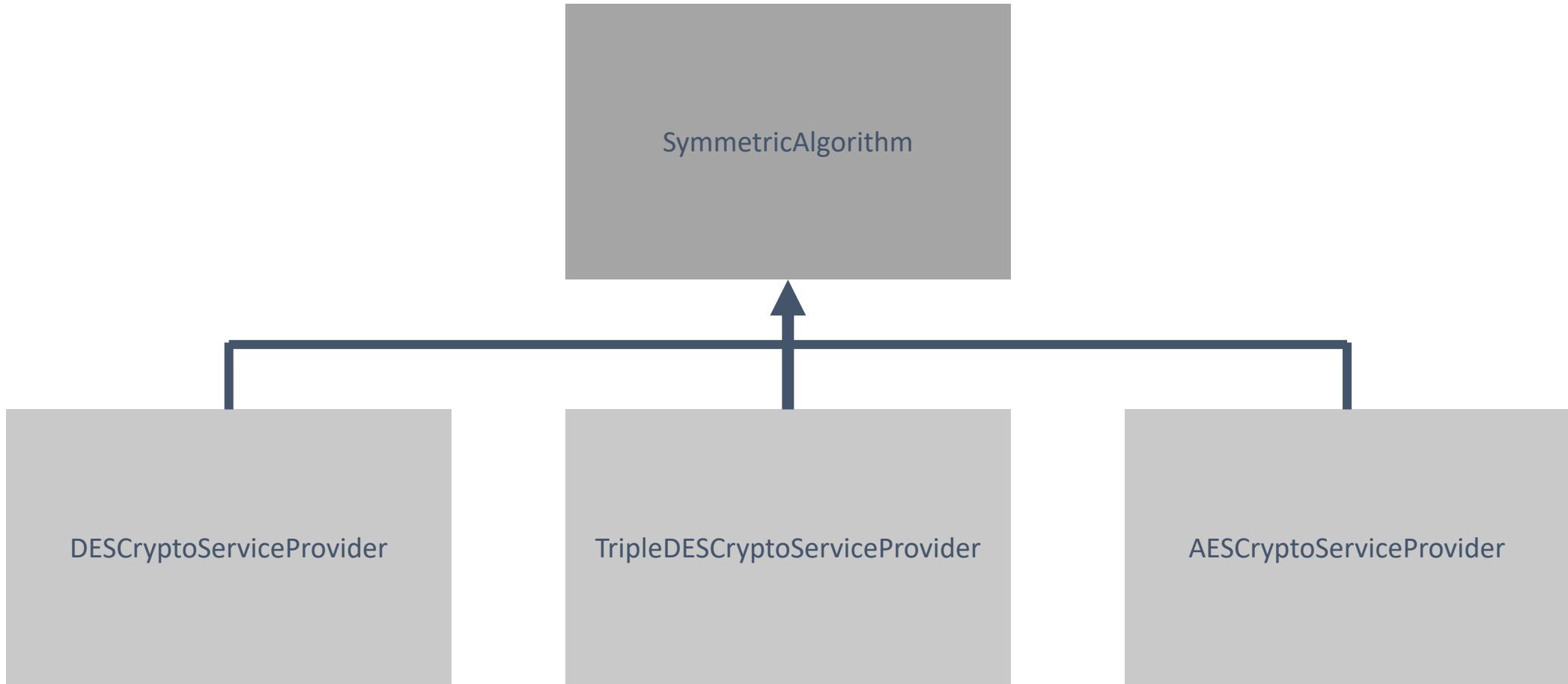
How Secure Is AES Against Brute Force Attack?

Key Size	Possible Combinations
1 bit	2
2 bit	4
4 bit	16
8 bit	256
16 bit	65536
32 bit	4.2×10^9
56 bit (DES)	7.2×10^{16}
64 bit	1.8×10^{19}
128 bit (AES)	3.4×10^{38}
192 bit (AES)	6.2×10^{57}
256 bit (AES)	1.1×10^{77}

How Secure Is AES Against Brute Force Attack?

Key Size	Time to Crack
56 bit	399 sec (6.65min)
128 bit	1.02×10^{18} years
192 bit	1.87×10^{37} years
256 bit	3.31×10^{56} years

Using the .NET Framework Libraries



Using the .NET Framework Libraries

Encryption Mode

- Cipher block chaining (CBC)
- Ciphertext feedback (CFB)
- Ciphertext stealing (CTS)
- Electronic codebook (ECB)
- Output feedback (OFB)

SymmetricAlgorithm

CipherMode Mode

PaddingMode Padding

byte[] Key

byte[] IV

Using the .NET Framework Libraries

Padding

- ANSI X923
- ISO 10126
- None
- PKCS7
- Zeros

SymmetricAlgorithm

CipherMode Mode

PaddingMode Padding

byte[] Key

byte[] IV

Using the .NET Framework Libraries

Key

- Byte array to store encryption key
- Generate secure keys
 - RNGCryptoServiceProvider or
 - GenerateKey()

SymmetricAlgorithm

CipherMode Mode

PaddingMode Padding

byte[] Key

byte[] IV

Using the .NET Framework Libraries

IV

- InitializationVector is a byte array
- Also called a nonce or number once
- IV prevents repetition in encryption
- IV does not have to be kept secret

SymmetricAlgorithm

CipherMode Mode

PaddingMode Padding

byte[] Key

byte[] IV

AesManaged or AesCryptoServiceProvider

- .NET provides 2 implementations of AES
 - AesManaged
 - AesCryptoServiceProvider

AesManaged or AesCryptoServiceProvider

- .NET provides 2 implementations of AES
 - AesManaged
 - AesCryptoServiceProvider
- AesManaged : .NET specific implementation

AesManaged or AesCryptoServiceProvider

- .NET provides 2 implementations of AES
 - AesManaged
 - AesCryptoServiceProvider
- AesManaged : .NET specific implementation
- AesCryptoServiceProvider : Uses windows cryptography libraries.
 - FIPS 140-2 certified

CryptoStream

- CLR uses a stream oriented design for cryptography

CryptoStream

- CLR uses a stream oriented design for cryptography
- Core of this design is CryptoStream



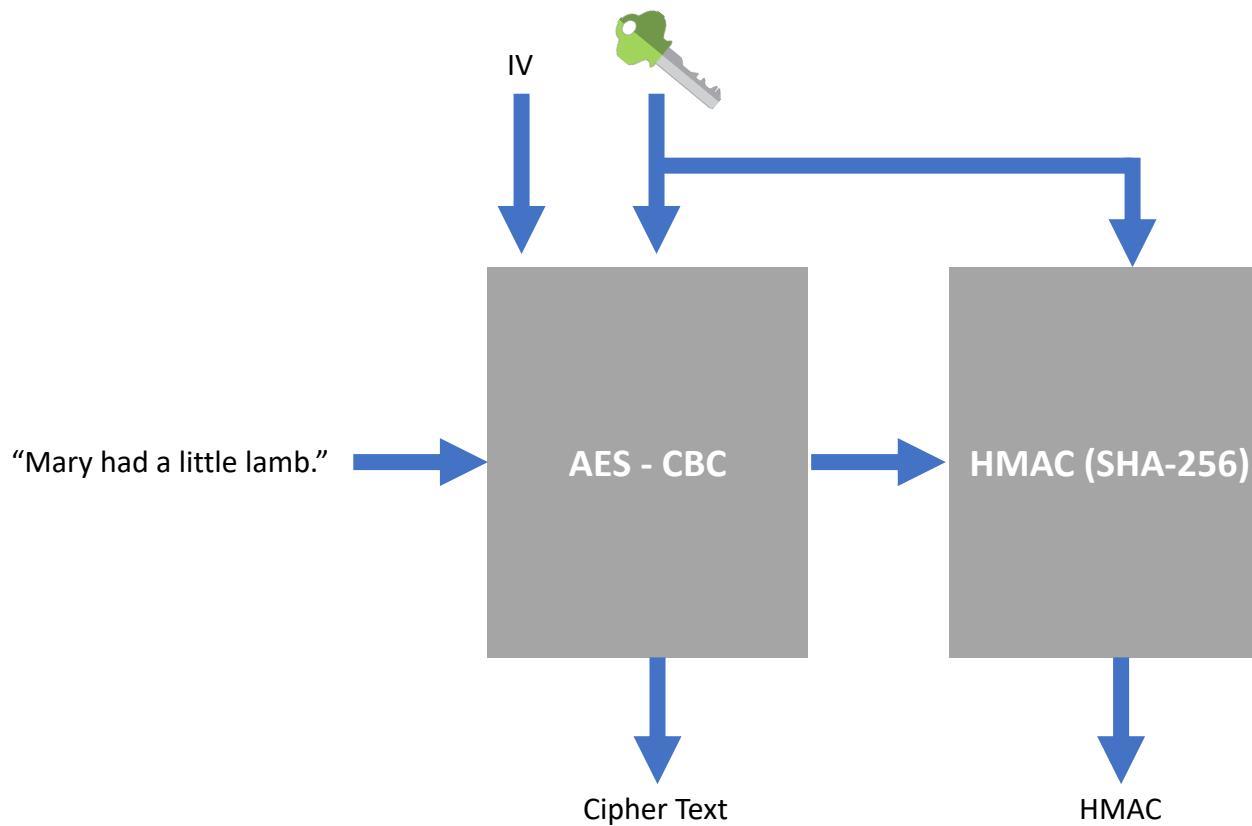
Now it's
your turn!

Step through the DES and AES code.

ADVANCE: Write a console app that takes a filename and a password. Encrypt that file using the password as the key.

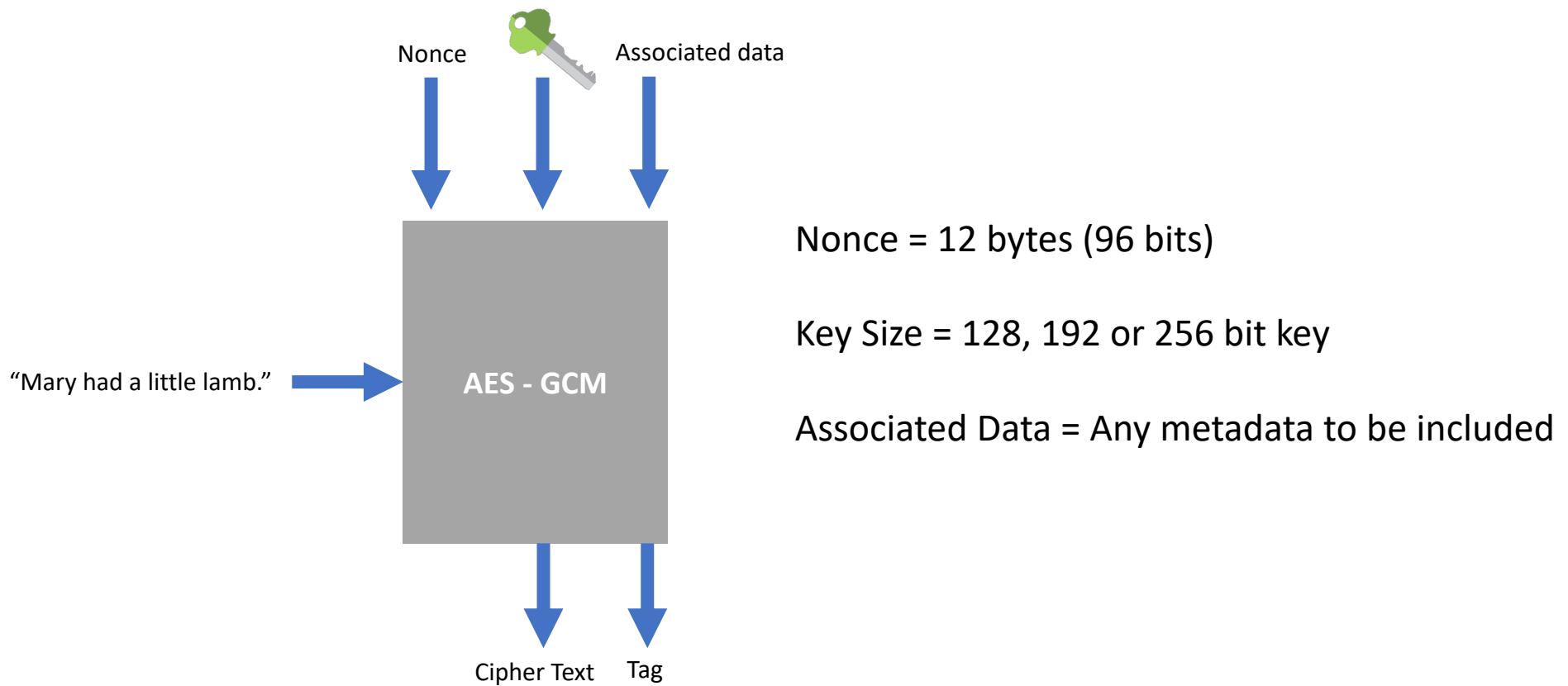
AES – GCM Mode (Galois Counter Mode)

Common Practice to do an authenticated hash after encrypting with AES - CBC



AES – GCM Mode (Galois Counter Mode)

AES – GCM produces the has as it encrypts





Now it's
your turn!

Step through the sample code.

Key Management with Protected Data

ProtectedData Class

Uses the Windows Data Protection API (DPAPI)

Key Management with Protected Data

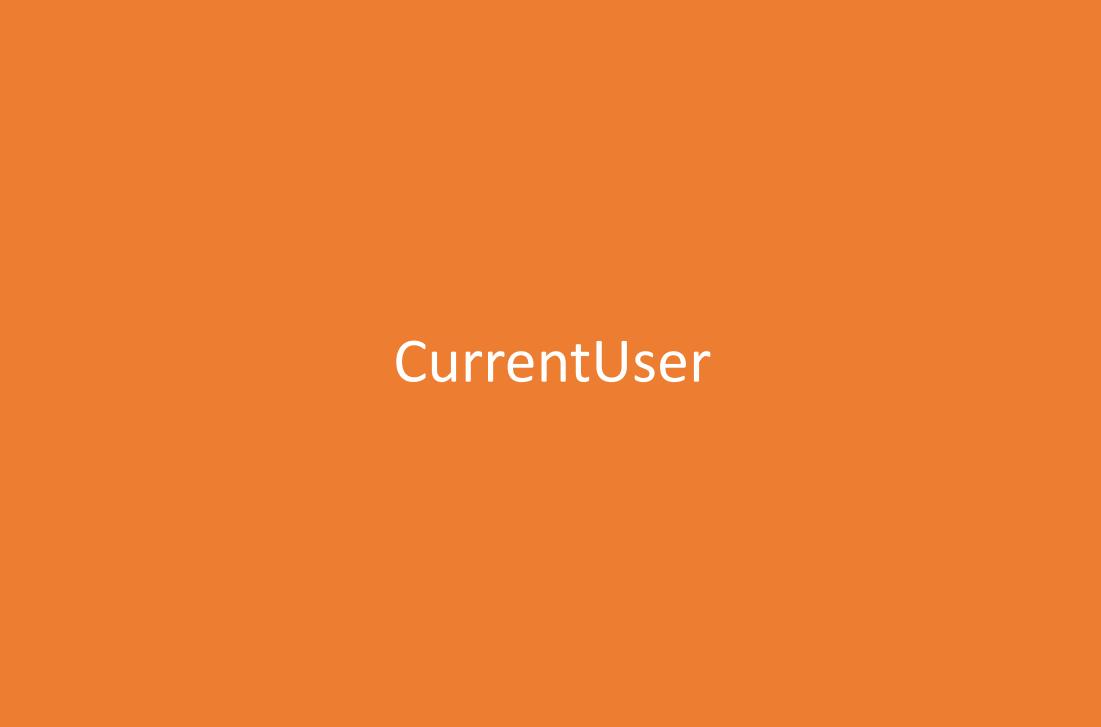
The screenshot shows the 'Manage NuGet Packages – ProtectedDataExample' window. The top navigation bar includes 'nuget.org', 'Browse', 'Installed' (which is underlined), and 'Updates'. A search bar is also present. The main content area displays the details for the 'System.Security.Cryptography.ProtectedData' package.

Property	Value
ID	System.Security.Cryptography.ProtectedData
Author	Microsoft
Published	03/12/2019
Downloads	0
License	View License
Project Page	Visit Page
Dependencies	None

Show pre-release packages

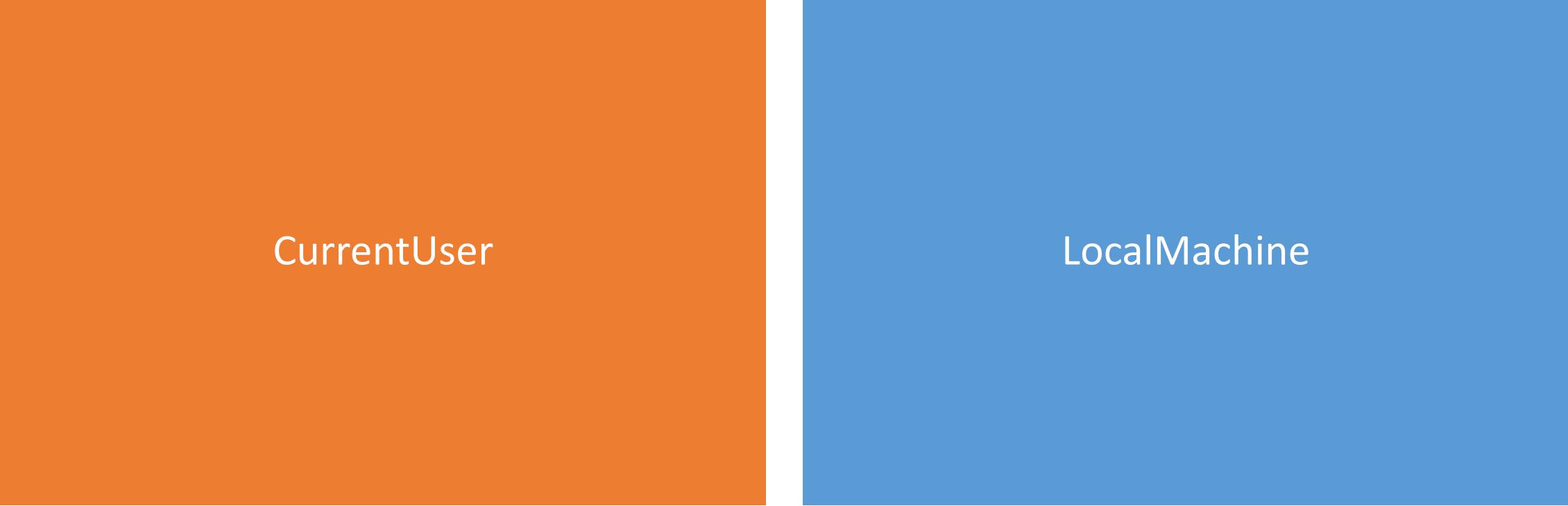
Key Management with Protected Data

Key Management with Protected Data



CurrentUser

Key Management with Protected Data

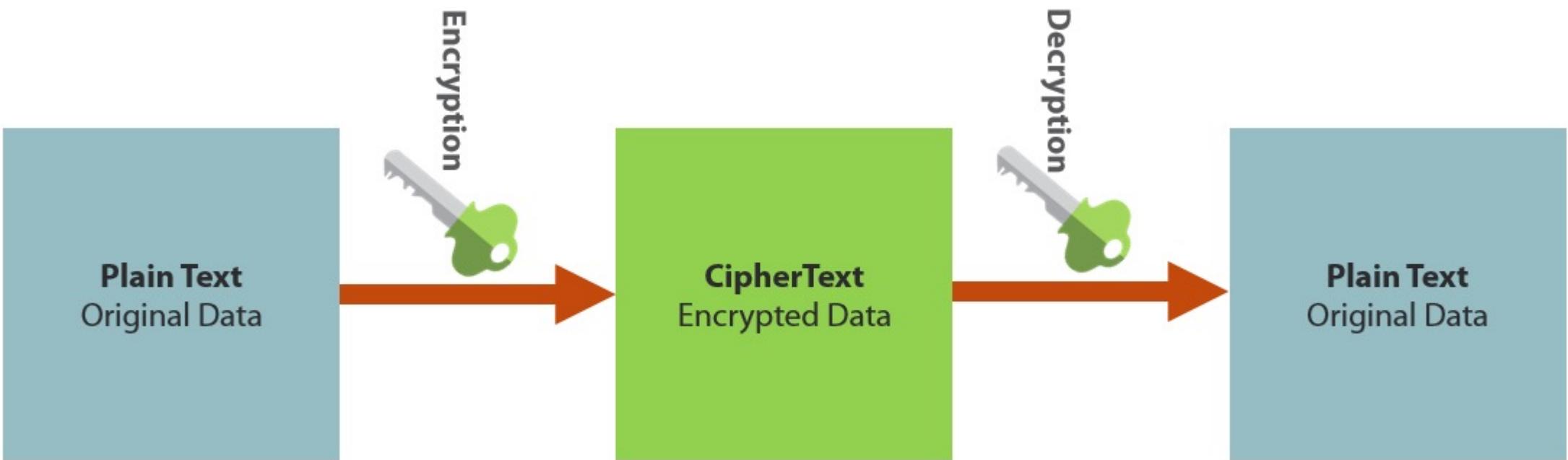


CurrentUser

LocalMachine

Asymmetric Encryption

Symmetric Encryption Recap



Symmetric Encryption Advantages



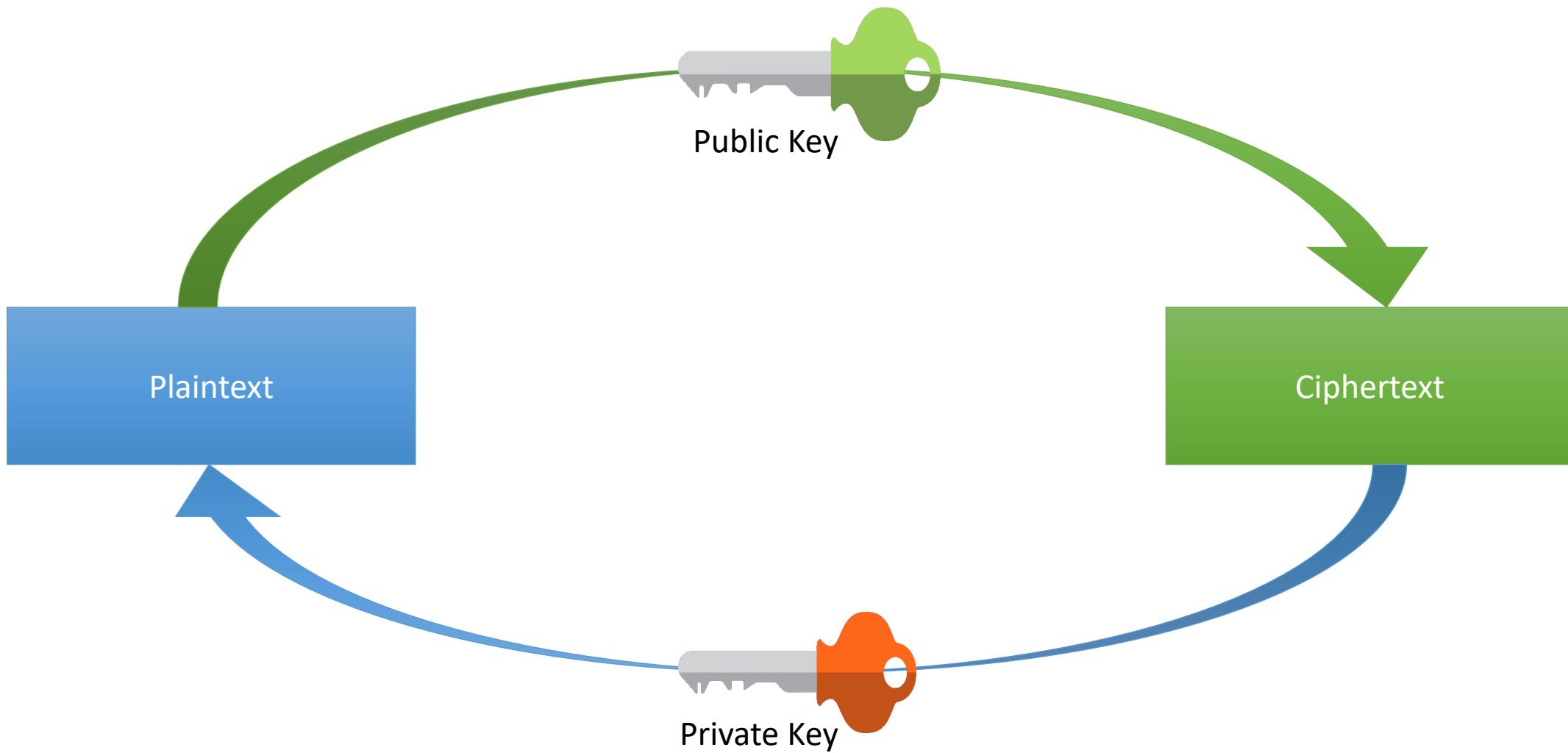
- Extremely secure
- Relatively fast

Symmetric Encryption Disadvantages



- Key sharing
- More damage if compromised

What is Asymmetric Encryption



What is Asymmetric Encryption



- Sender and receiver don't need to share keys prior
- Sender only need the recipients public key

What is Asymmetric Encryption



- Asymmetric encryption is slow compared to symmetric encryption.

RSA History

- RSA was developed by RSA Security LLC
- RSA stands for Rivest, Shamir and Adelman, the inventors of the technique
- No efficient way to factor large numbers
- RSA is the de-facto standard for industrial strength encryption
- There are limits to the amount of data you can encrypt in one go
- RSA is commonly used to encrypt symmetric encryption keys

How Does RSA Work?



1024 bit keys

2048 bit keys

4096 bit keys

Key Derivation

- Public and private keys are based on prime numbers
- Factoring a number back into constituent prime numbers is hard

Key Derivation

- Public and private keys are based on prime numbers
- Factoring a number back into constituent prime numbers is hard

$$23 \times 17 = ?$$

Key Derivation

- Public and private keys are based on prime numbers
- Factoring a number back into constituent prime numbers is hard

$$23 \times 17 = 391$$

Key Derivation

- Public and private keys are based on prime numbers
- Factoring a number back into constituent prime numbers is hard

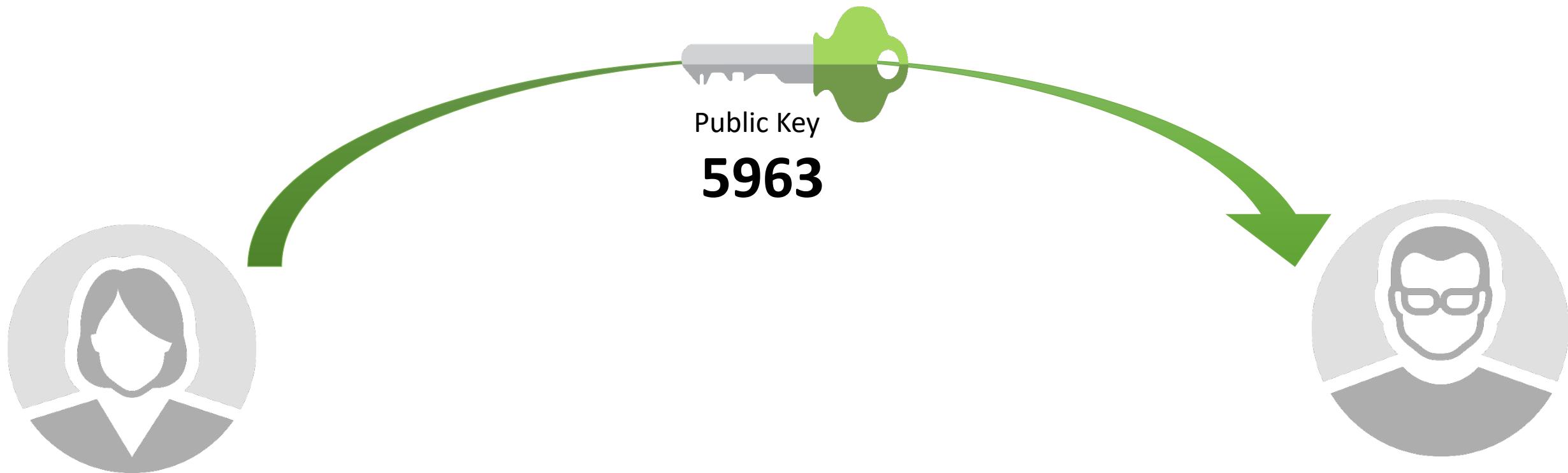
$$? \times ? = 5963$$

Key Derivation

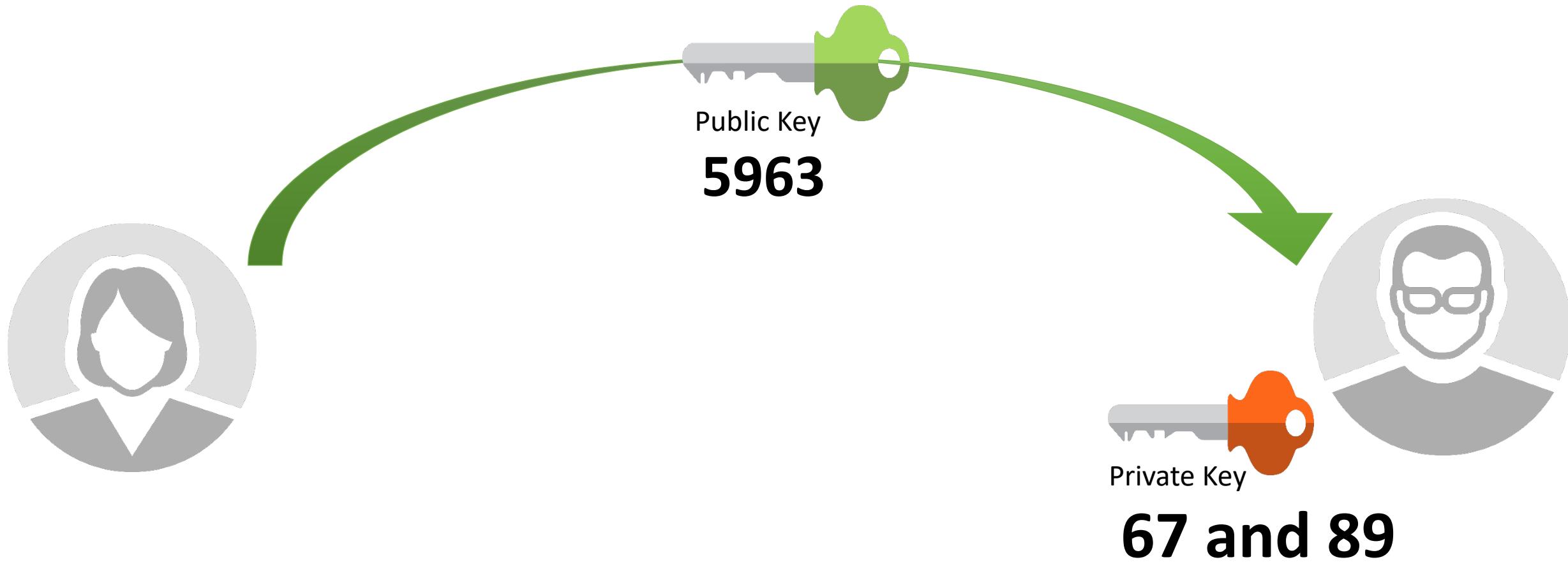
- Public and private keys are based on prime numbers
- Factoring a number back into constituent prime numbers is hard

$$67 \times 89 = 5963$$

Key Derivation



Key Derivation



Key Derivation

Name	Value	Type
publicKey	{System.Security.Cryptography.RSAParameters}	System.Security.Cryptography.RSAParameters
D	null	byte[]
DP	null	byte[]
DQ	null	byte[]
Exponent	{byte[3]}	byte[]
InverseQ	null	byte[]
Modulus	{byte[256]}	byte[]
P	null	byte[]
Q	null	byte[]
privateKey	{System.Security.Cryptography.RSAParameters}	System.Security.Cryptography.RSAParameters
D	{byte[256]}	byte[]
DP	{byte[128]}	byte[]
DQ	{byte[128]}	byte[]
Exponent	{byte[3]}	byte[]
InverseQ	{byte[128]}	byte[]
Modulus	{byte[256]}	byte[]
P	{byte[128]}	byte[]
Q	{byte[128]}	byte[]

Encryption and Decryption

- RSA encryption and decryption is a mathematical operation

Encryption and Decryption

- RSA encryption and decryption is a mathematical operation
- Based on modular math



Now it's
your turn!

Step through the
RSAWithRSAParameterKey example

Using the RSA Class to Export Keys

Exporting RSA Keys

ExportPKCS8PrivateKey

ExportEncryptedPKCS8PrivateKey

Exporting RSA Keys

```
RSA rsa;

PbeParameters keyParams = new
    PbeParameters(PbeEncryptionAlgorithm.Aes256Cbc,
                  HashAlgorithmName.SHA256,
100000);

encrypted = rsa.ExportEncryptedPkcs8PrivateKey(
    Encoding.UTF8.GetBytes("384w5n73489in57v"),
    keyParams);
```

Exporting RSA Keys

```
RSA rsa;

PbeParameters keyParams = new
    PbeParameters(PbeEncryptionAlgorithm.Aes256Cbc,
                  HashAlgorithmName.SHA256,
                  100000);

encrypted = rsa.ExportEncryptedPkcs8PrivateKey(
    Encoding.UTF8.GetBytes("384w5n73489in57v"),
    keyParams);
```

Exporting RSA Keys

How can you protect the private key password?

Exporting RSA Keys

How can you protect the private key password?

A large orange rectangular box occupies the lower-left portion of the slide, serving as a placeholder for content.

ProtectedData

Exporting RSA Keys

How can you protect the private key password?

ProtectedData

Store as a secret in a Key Store

Digital Signatures

Digital Signatures

- Claiming authenticity of a message
- Digital signatures give both authentication and non-repudiation
- Based on asymmetric cryptography
- Digital signatures consist of
 - Public and private key generation
 - Signing algorithm using the private key
 - Verification algorithm using the public key

Digital Signatures



1. Alice encrypts her data



Digital Signatures



1. Alice encrypts her data
2. Alice takes a hash of her data



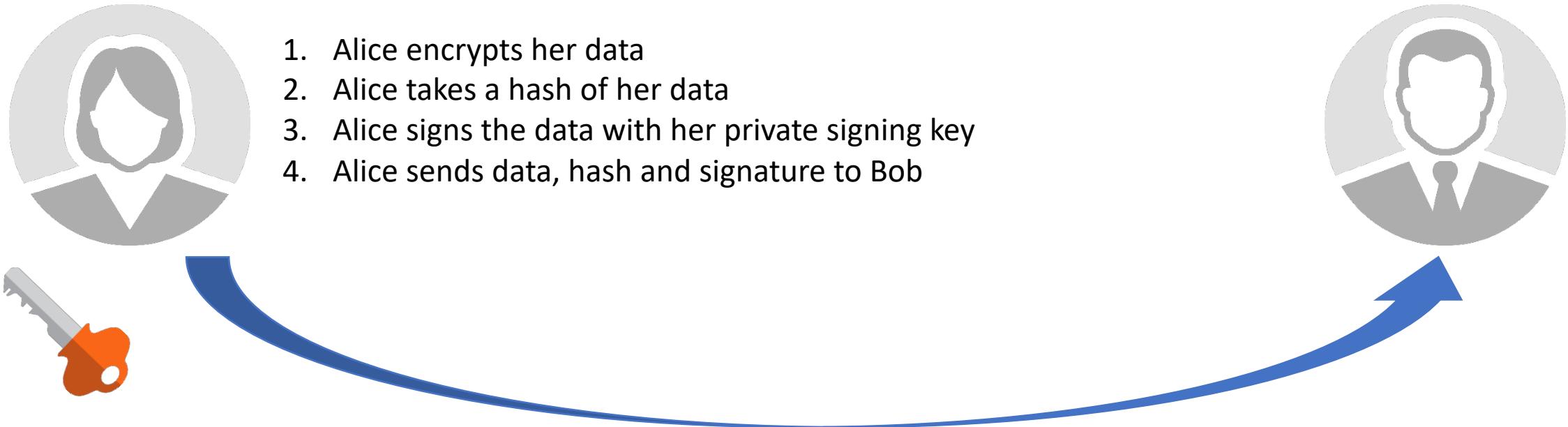
Digital Signatures



1. Alice encrypts her data
2. Alice takes a hash of her data
3. Alice signs the data with her private signing key

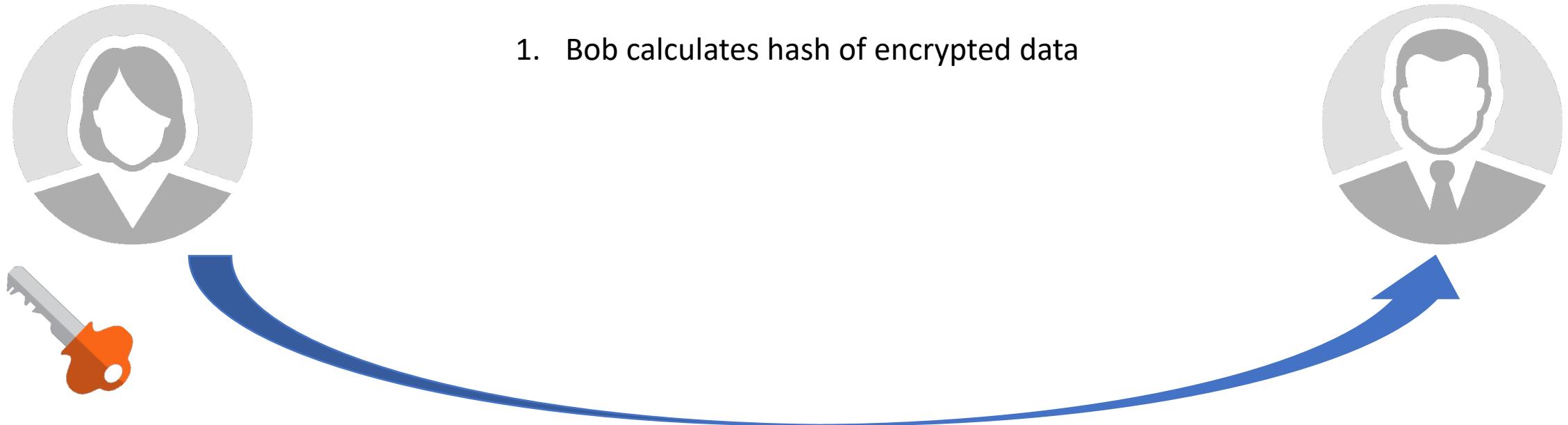


Digital Signatures

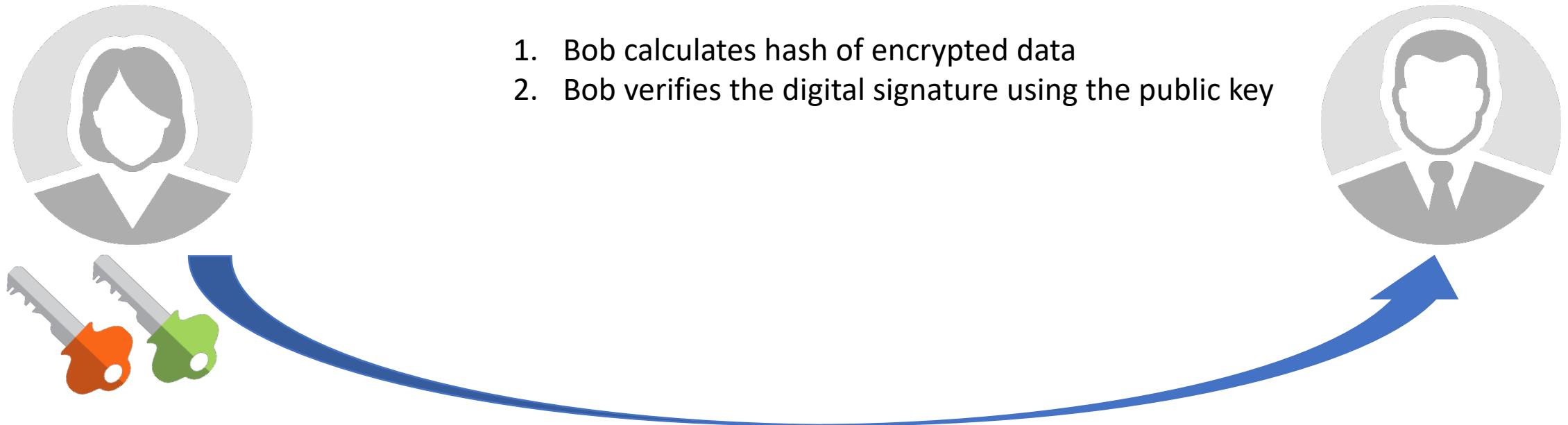


Alice sends encrypted data, the hash and the digital signature to Bob

Digital Signatures



Digital Signatures



Alice sends encrypted data, the hash and the digital signature to Bob

Digital Signatures

	Public Key	Private Key
Encryption (RSA)	Encrypt	Decrypt
Digital Signatures	Verify Signature	Sign Message

Digital Signatures in the .NET Framework

- Digital signatures use 3 main classes
 - RSACryptoServiceProvider
 - RSAPKCS1SignatureFormatter
 - RSAPKCS1SignatureDeformatter

Digital Signatures in the .NET Framework

- You can also use the RSA class for signing and Verifying which gives you the same key import and export benefits.



Now it's
your turn!

Step through the Digital Signature example code

What We Will Cover

Time	Day 1	Day 2
9 – 10.30am	Introduction, Setup, Random Numbers	Recap from Day 1
10.45 – 12.30am	Hashing and HMACS, Passwords	Hybrid Encryption, Azure Key Vault
12.30 – 1.30pm	LUNCH	LUNCH
1.30 – 3.15pm	Symmetric Encryption	Azure Key Vault, Blockchain Fundamentals
3.30 – 5pm	Asymmetric Encryption + Digital Signatures, Project Time, Wrap Up	Blockchain Fundamentals, Summary, Wrap Up

Quick Recap

Confidentiality

Integrity

Authentication

Non-Repudiation

Quick Recap

Confidentiality

Integrity

Authentication

Non-Repudiation

AES and RSA

Quick Recap

Confidentiality

Integrity

Authentication

Non-Repudiation

AES and RSA

SHA-256

Quick Recap

Confidentiality

Integrity

Authentication

Non-Repudiation

AES and RSA

SHA-256

HMAC (SHA-256)

Quick Recap

Confidentiality

Integrity

Authentication

Non-Repudiation

AES and RSA

SHA-256

HMAC (SHA-256)

Digital Signature

Quick Recap

Confidentiality

Integrity

Authentication

Non-Repudiation

AES and RSA

SHA-256

HMAC (SHA-256)

Digital Signature

Hybrid Encryption

Hybrid Encryption

Quick Recap

Confidentiality

Integrity

Authentication

Non-Repudiation

AES and RSA

SHA-256

HMAC (SHA-256)

Digital Signature

Hybrid Encryption

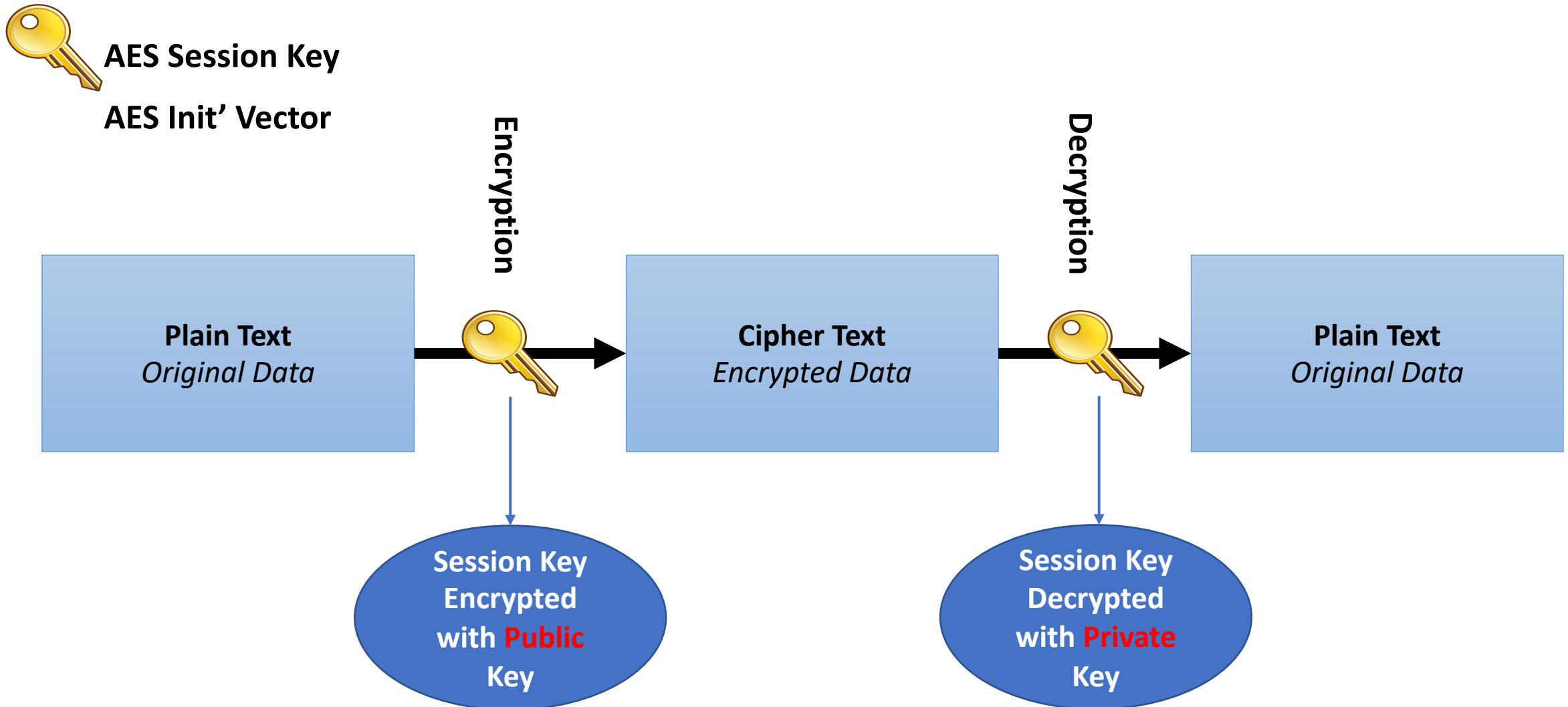
Hybrid Encryption

RSA has limits on the amount of data you can encrypt

You want the flexibility of AES but secure key exchange is hard

Combine RSA and AES into a hybrid scheme

Hybrid Encryption



Hybrid Encryption

Encrypted Data Packet

RSA Encrypted Session Key
AES Initialization Vector
AES Encrypted Data

Hybrid Encryption

Alice generates 32 byte AES key



Hybrid Encryption

Alice generates 16 byte IV



Hybrid Encryption

Alice encrypts her data with AES using session key and IV



Hybrid Encryption

Alice encrypts the session key with RSA and **Bobs** public key



Hybrid Encryption

Alice stores encrypted data, encrypted session key and IV in a structure



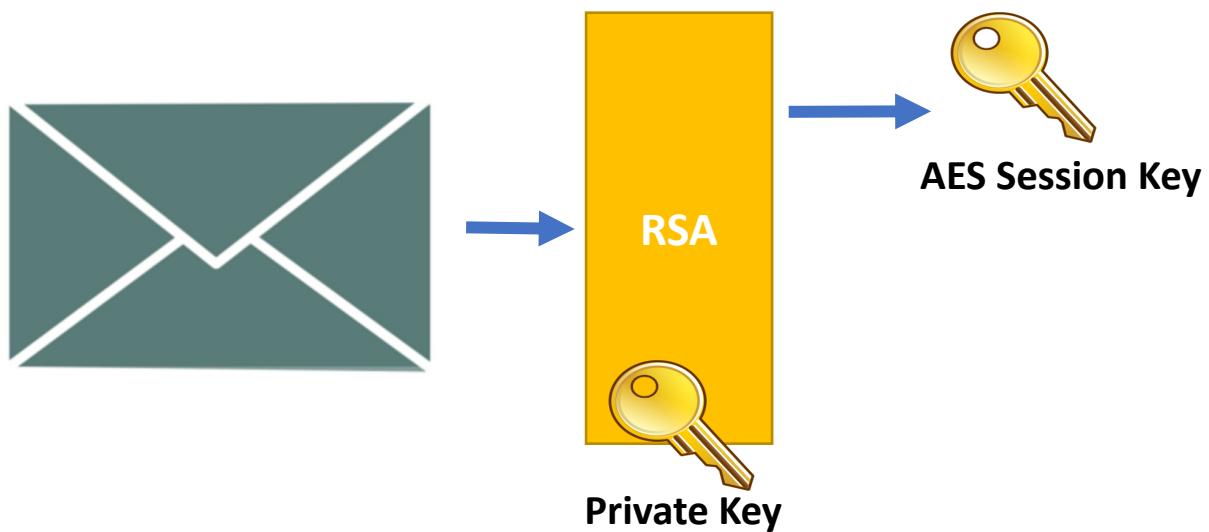
Hybrid Encryption

Alice sends data to Bob



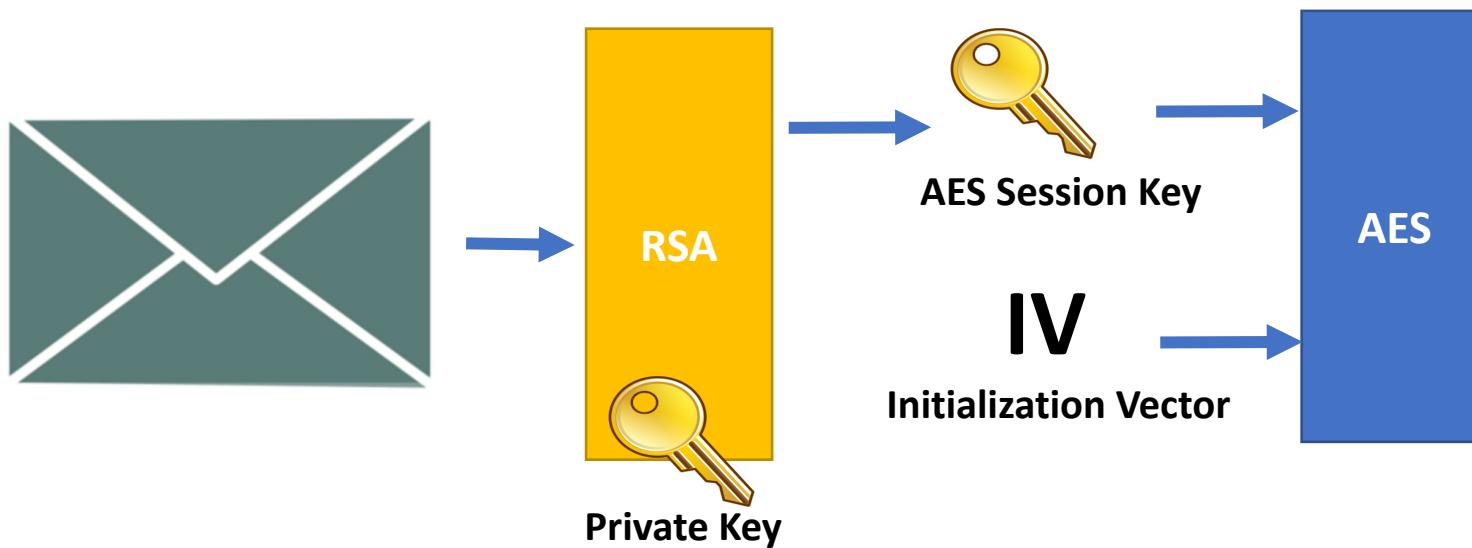
Hybrid Encryption

Bob decrypts the encrypted AES session key using RSA and **Bobs** private key



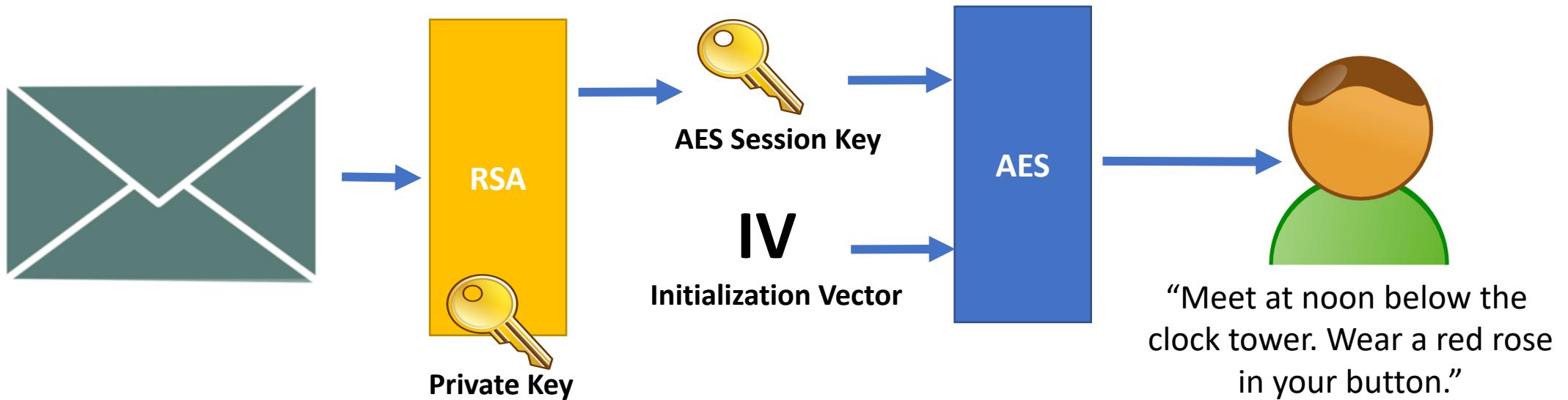
Hybrid Encryption

Bob decrypts the encrypted data using the decrypted AES key and the IV



Hybrid Encryption

Bob reads the decrypted message



Hybrid Encryption

Bob generates 32 byte AES key



Hybrid Encryption

Bob generates 16 byte IV



Hybrid Encryption

Bob encrypts his data with AES using session key and IV



Hybrid Encryption

Bob encrypts the session key with RSA and **Alices** public key



Hybrid Encryption

Bob stores encrypted data, encrypted session key and IV in a structure



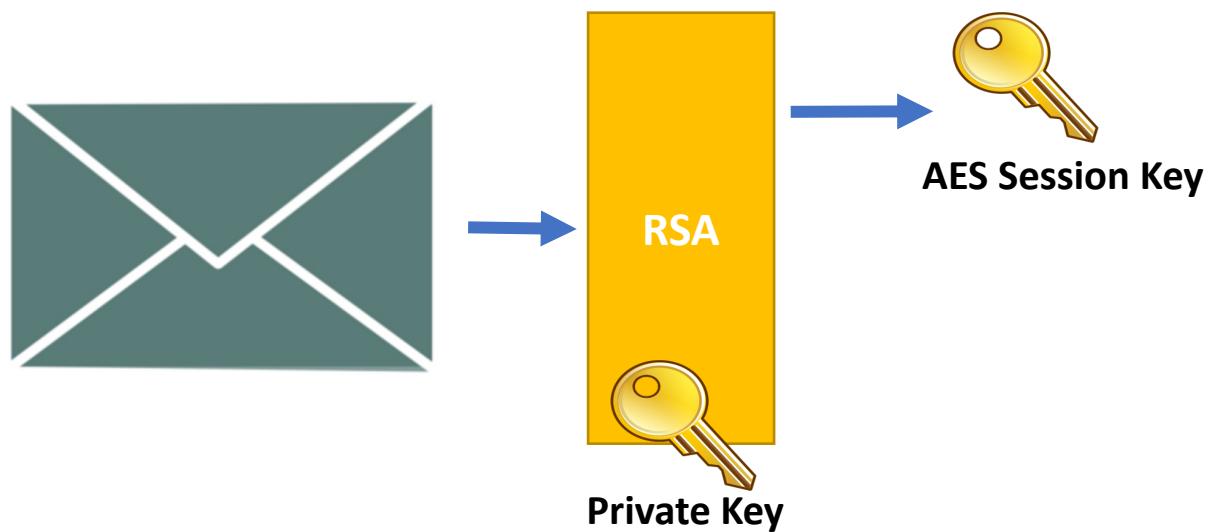
Hybrid Encryption

Bob sends data to **Alice**



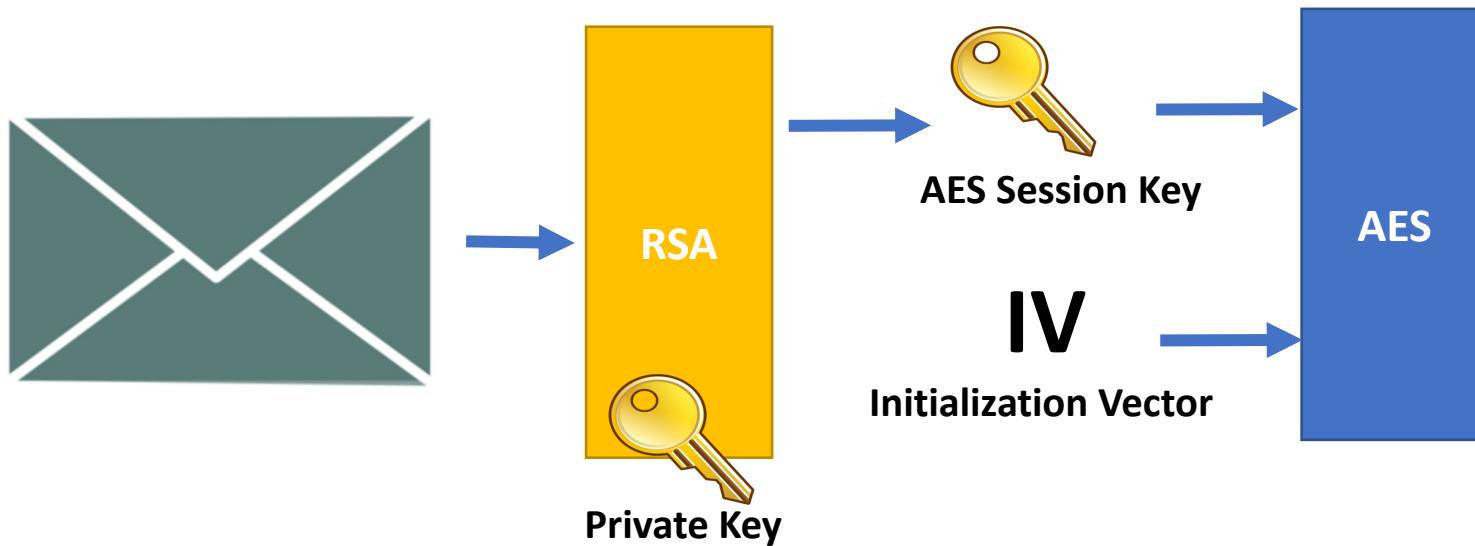
Hybrid Encryption

Alice decrypts the encrypted AES session key using RSA and **Alices** private key



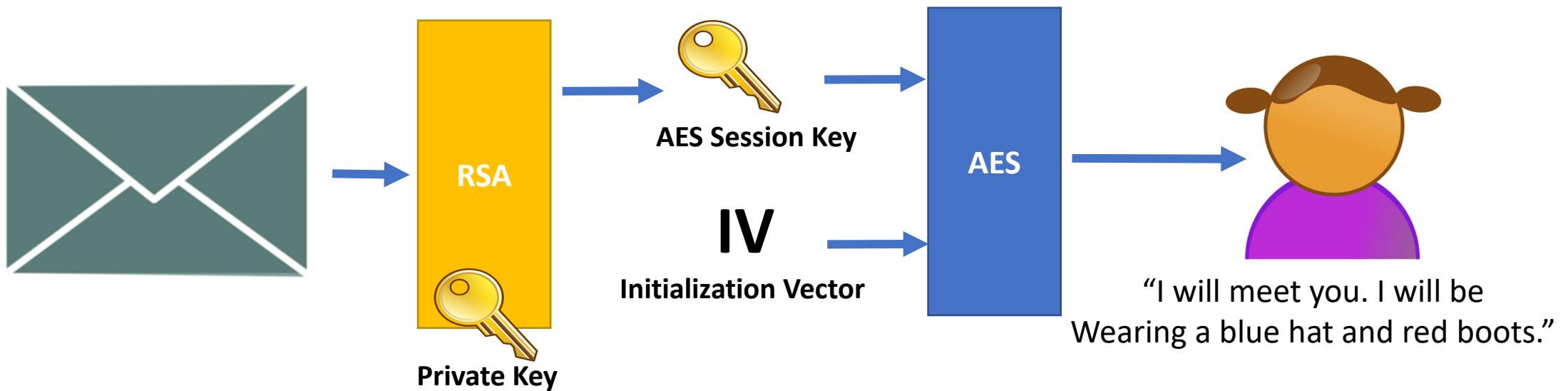
Hybrid Encryption

Alice decrypts the encrypted data using the decrypted AES key and the IV



Hybrid Encryption

Alice reads the decrypted message

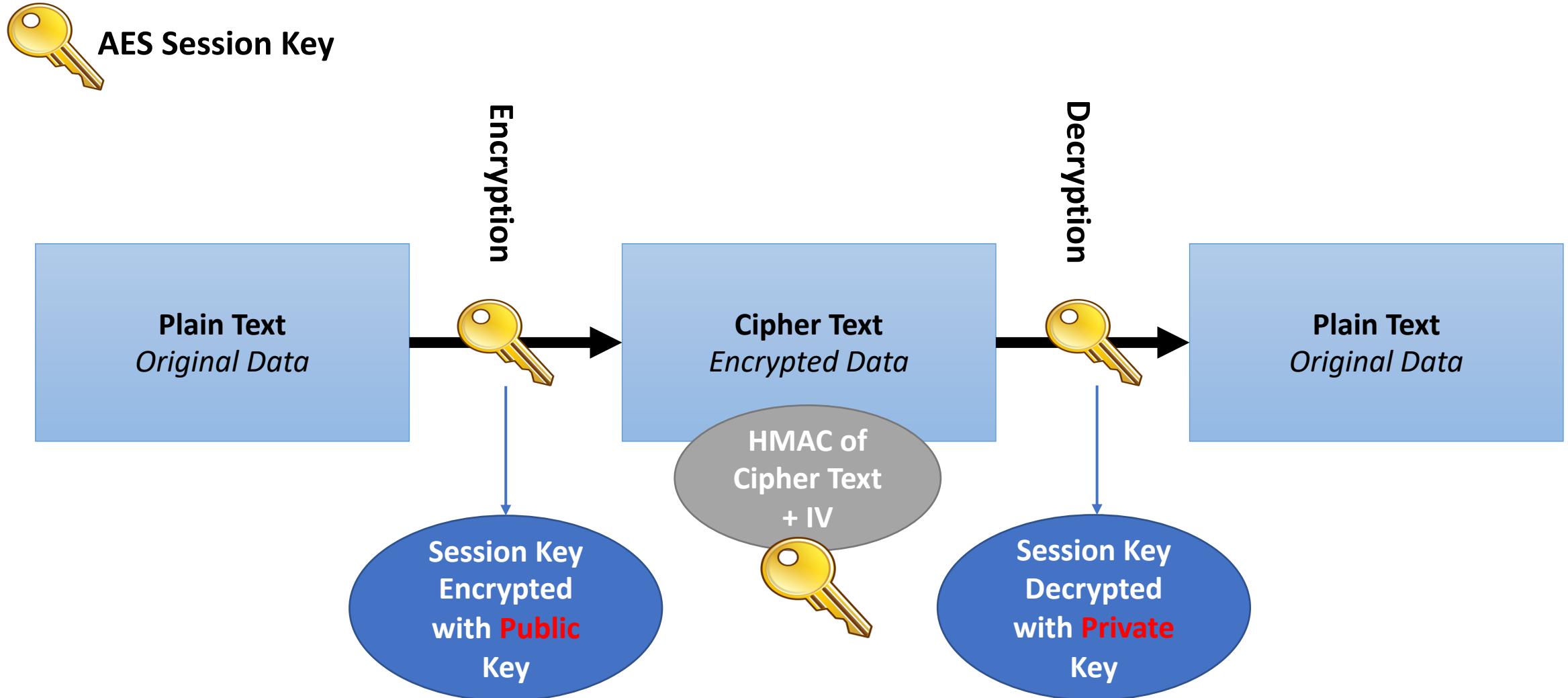




Now it's
your turn!

Step through the code in the Hybrid example.

Hybrid Encryption – Integrity (AES-CBC)



Hybrid Encryption – Integrity (AES-CBC)

Encrypted Data Packet

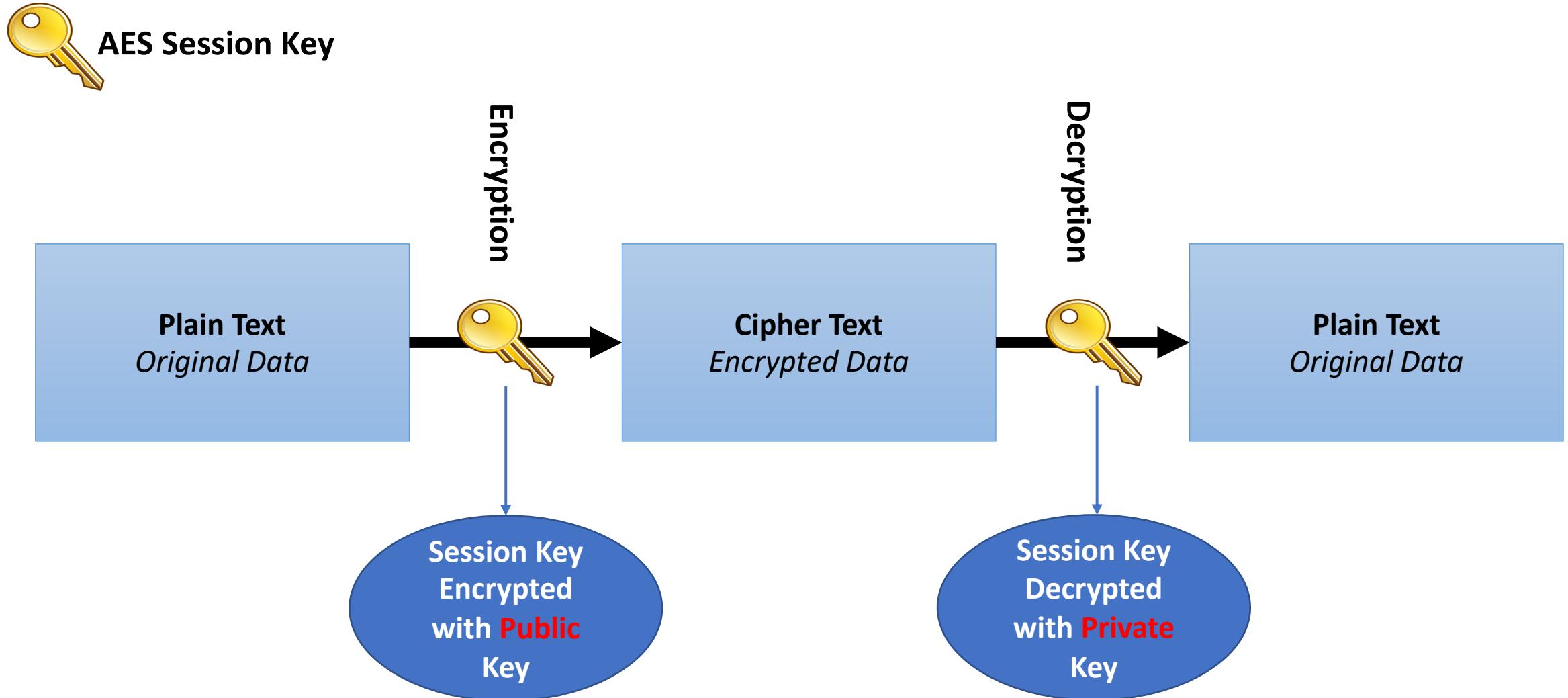
RSA Encrypted Session Key
AES Initialization Vector
AES Encrypted Data
HMAC (SHA-256) of Encrypted Data



Now it's
your turn!

Step through the code in the Hybrid with Integrity example.

Hybrid Encryption – Integrity (AES-GCM)



Hybrid Encryption – Integrity (AES-CBC)

Encrypted Data Packet

RSA Encrypted Session Key

AES Nonce

AES Encrypted Data

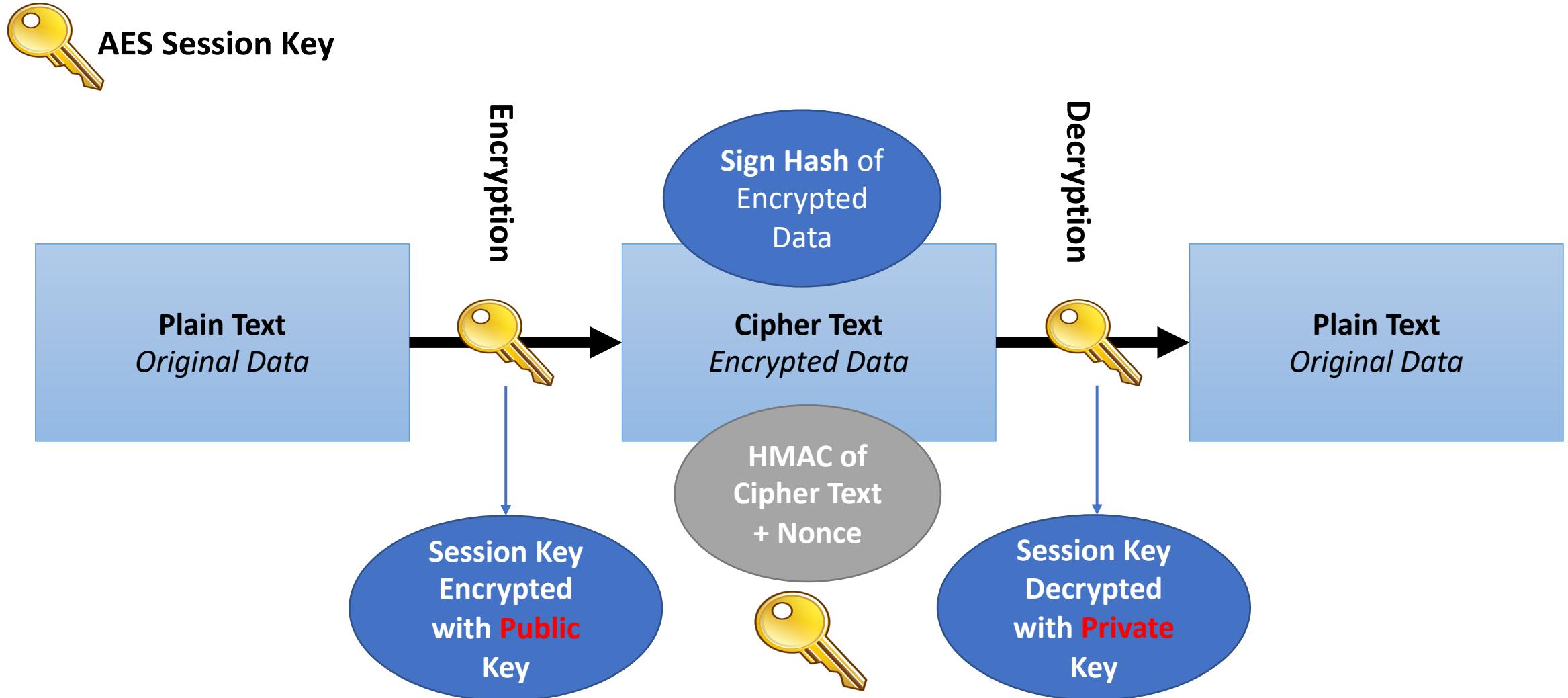
AES Tag



Now it's
your turn!

Step through the code in the Hybrid with Integrity GCM example.

Hybrid Encryption - Signature



Hybrid Encryption - Signature

Encrypted Data Packet

RSA Encrypted Session Key

AES Initialization Vector

AES Encrypted Data

HMAC (SHA-256) of Encrypted Data

Digital Signature of HMAC

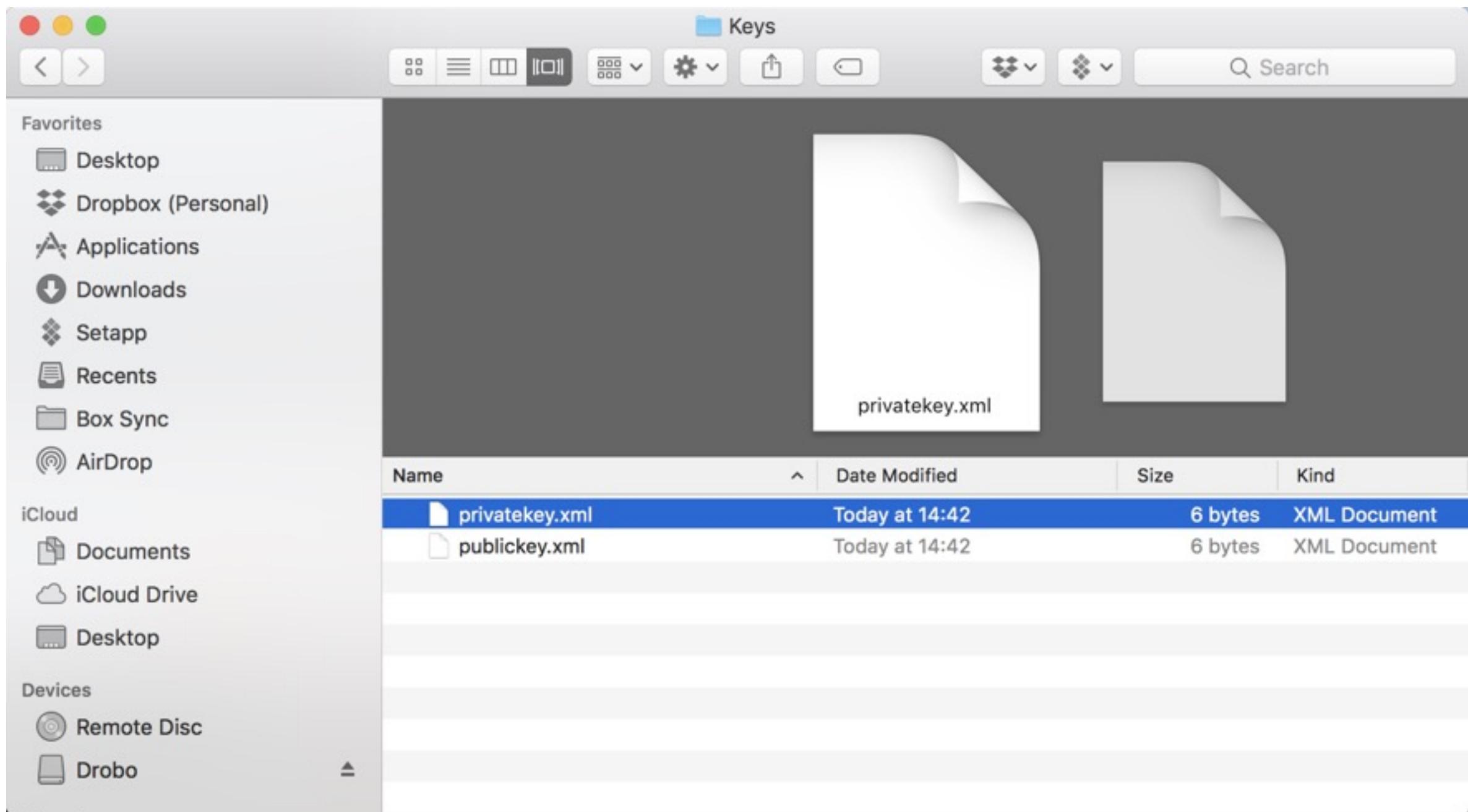


Now it's
your turn!

Step through the code in the Hybrid with Integrity and Signature GCM example.

Azure Key Vault

Key Management Options





Identity of issues to / from

Public key verification

Installed directly onto servers

Robust and proven mechanism

Benefits



1. Minimal involvement from end users
2. No additional hardware required
3. Easy to manage
4. Natively supported by many enterprise systems



Tamper proof

Central key repository

Can also store Certs (Private Keys)

Tamper proof

Central key repository

Can also store Certs (Private Keys)

Traditionally cost prohibitive



HSM cost no longer an issue
in the world of Azure

Introducing Azure Key Vault



Key Vault

Safeguard cryptographic keys and other secrets used by cloud apps and services



Azure Key Vault provides an abstraction over the
Thales NShield hardware

<https://www.thalesesecurity.co.uk/products/general-purpose-hsms/nshield-connect>

Software vs Hardware

Both store keys at rest in HSM Hardware

Software: Encrypt and Decrypt performed in software on compute VM's

Hardware: Encrypt and Decrypt performed in the HSM Hardware

Recommend : Use software in dev/test. Hardware in production

Non Key Operations

Service	Standard	Premium
Secret operations	\$0.03 per 10,000 operation	\$0.03 per 10,000 operations
Certificate operations	Renewals \$3 per request All other operation \$0.03 per 10,000	Renewals \$3 per request All other operation \$0.03 per 10,000

Software Key Operations

Service	Standard	Premium
RSA 2048-bit keys	\$0.03 per 10,000 operations	\$0.03 per 10,000 operations
RSA 3072 – 4096bit keys	\$0.15 per 10,000 operations	\$0.15 per 10,000 operations

Hardware HSM Key Operations

Service	Standard	Premium
RSA 2048 – bit keys	N/A	\$1 per Key + \$0.03 per 10,000 operation
RSA 3072 – 4096 bit keys	N/A	First 250 keys = \$5 per key 251 – 1500 keys \$2.50 per key 1501 – 4000 keys £0.90 per key 4001+ keys \$0.40 per key + \$0.15 per 10,000 operations

**Once a Key goes in the Key Vault,
it never comes out again**

Setting up a Key Vault

The screenshot shows the Microsoft Azure documentation page for 'Get started with Azure Key Vault'. The page has a dark header with navigation links like 'SALES 1-800-867-1389', 'CONTACT SALES', 'MY ACCOUNT', 'PORTAL', 'Search', and 'FREE ACCOUNT'. The main content area has a title 'Get started with Azure Key Vault' with a date '20/11/2017', a reading time of '11 minutes', and a note about contributors. It includes a list of activities: 'How to create a hardened container (a vault) in Azure.', 'How to use KeyVault to store and manage cryptographic keys and secrets in Azure.', and 'How an application can use that key or password.' Below this is a note about Azure Key Vault availability across regions and a link to the 'Key Vault pricing page'. A 'Requirements' section lists prerequisites: 'An Azure subscription' and 'Azure PowerShell, minimum version of 1.1.0'. On the right side, there are buttons for 'Feedback', 'Edit', 'Share', and 'Theme' (set to 'Light'), and a sidebar titled 'In this article' with links to various Key Vault management tasks.

<https://docs.microsoft.com/en-gb/azure/key-vault/key-vault-get-started>

Setting up a Key Vault

Using Powershell

Create resource group

Create a Key Vault (Specify resource group, region, hw/sw)

You can create software / hardware keys

Write / read secrets

Upload certificates

```
Connect-AzureRmAccount
```

```
Get-AzureRmSubscription
```

```
Set-AzureRmContext -SubscriptionId <subscription ID>
```

```
New-AzureRmResourceGroup -Name 'MyResourceGroup' -Location 'East US'
```

```
New-AzureRmKeyVault -VaultName 'MyVault' -ResourceGroupName 'MyResourceGroup' -Location 'East US'
```

```
New-AzureRmKeyVault -VaultName 'MyVault' -ResourceGroupName 'MyResourceGroup' -Location 'East US' -SKU 'Premium'
```



+ Create a resource

All services

★ FAVORITES

Dashboard

Resource groups

All resources

Recent

App Services

Virtual machines (classic)

Virtual machines

SQL databases

Cloud services (classic)

Security Center

Subscriptions

Application Insights

Azure Active Directory

Monitor

Cost Management + Billing

Help + support

Advisor

Home > RIKeyVault

RIKeyVault
Key vault

Search (Ctrl+Shift+F)

Delete Move

Resource group (change)

RIVaultTestRG

Location

North Europe

Subscription (change)

Microsoft Partner Network

Subscription ID

7408974f-4567-4913-87ed-dfb7474bd6fa

DNS Name

https://rikeyvault.vault.azure.net/

Sku (Pricing tier)

Standard

SETTINGS

Keys

Secrets

Certificates

Access policies

Properties

Locks

Automation script

MONITORING

Alert rules

Diagnostics logs

Log analytics (OMS)

Log search

Metrics (Preview)

SUPPORT + TROUBLESHOOTING

New support request

Monitoring

Show data for last:

1 hour

6 hours

12 hours

1 day

7 days

30 days

Click for additional metrics.

Total requests (RIKeyVault)



Average latency (RIKeyVault)



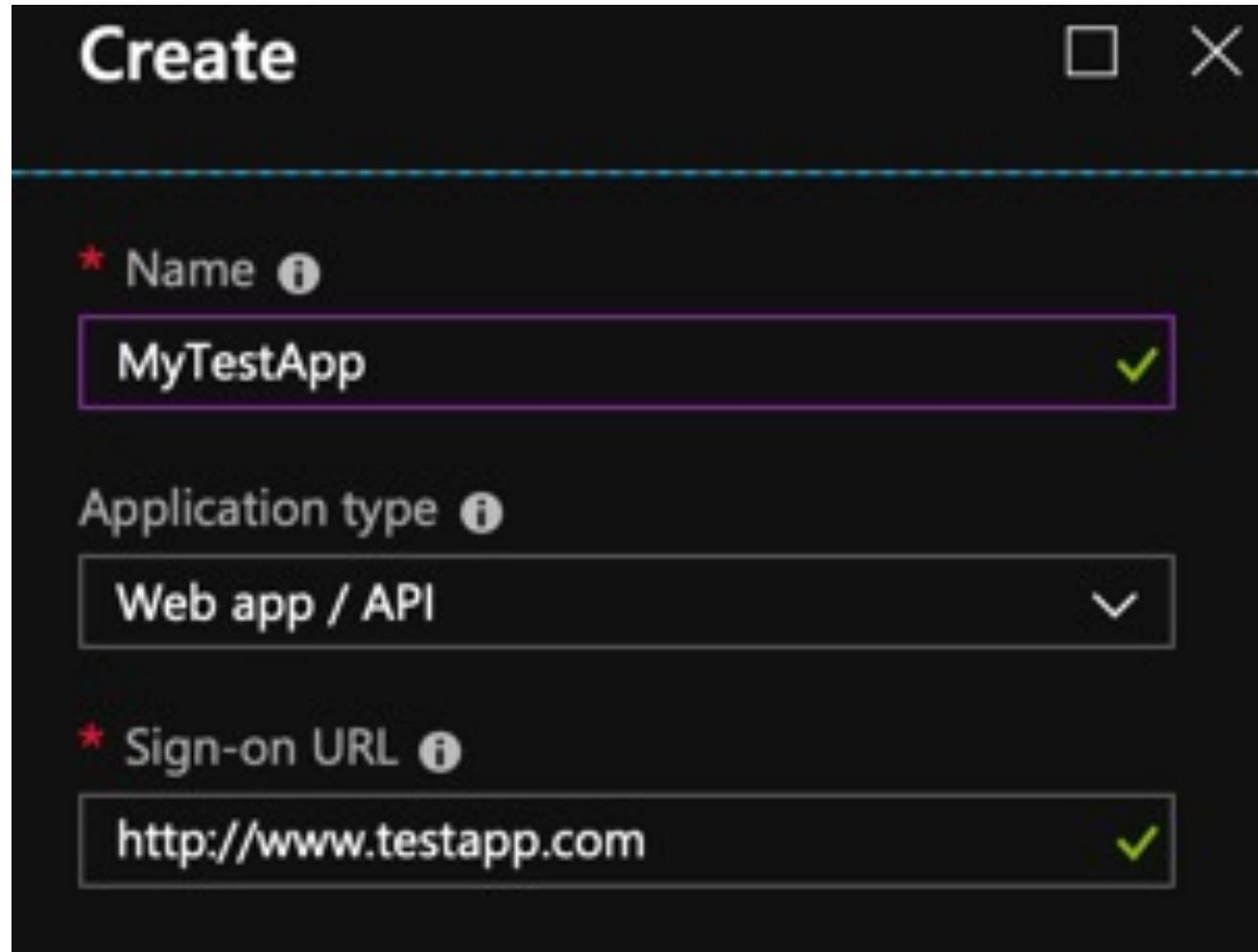
Setting up a Key Vault

To use your application, you need to register with AzureAD

ApplicationId

Shared Secret

Need to be presented by your application to get a
Token to access the vault



Creating an Application in Azure AD

MyTestApp
Registered app

Settings Manifest Delete

Display name	Application ID
MyTestApp	5c60ef25-17cd-433f-b665-db0509c5aa14
Application type	Object ID
Web app / API	b8a1172b-326c-448b-93ba-aea27be2ccb5
Home page	Managed application in local directory
http://www.testapp.com	

Make a note of the Application ID

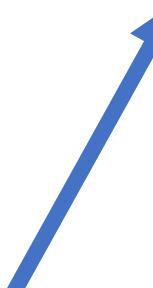
Settings

Filter settings

GENERAL

- Properties >
- Reply URLs >
- Owners >

API ACCESS

- Required permissions >
- Keys** > 

TROUBLESHOOTING + SUPPORT

- Troubleshoot >
- New support request >

Keys

Save Discard Upload Public Key

⚠ Copy the key value. You won't be able to retrieve after you leave this blade.

Passwords

DESCRIPTION	EXPIRES	VALUE
my Key	31/12/2299	UqOBR+xUGImTslm6ufhiCnXjJpEHU6jKs7C7tyMsdtw=

Key description Duration Value will be displayed on save ...

**Make a note of the ClientSecret
Can only view this once**

 RIKeyVault - Access policies
Key vault

Search (Ctrl+ /) << Save Discard Refresh

Click to show advanced access policies

Add new

 Stephen Haunts
USER

 SteveTestVault
APPLICATION

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Settings

Keys

Secrets

Certificates

Access policies

Firewalls and virtual networks

Properties

Locks

Automation script

You then need to assign permissions

Add access policy

Add a new access policy

Configure from template (optional)

Select principal
None selected

Key permissions
16 selected

Secret permissions
8 selected

Certificate permissions
16 selected

Authorized application i
None selected

Principal

Select a principal

+ Invite

Select i

Search by name or email address

Microsoft.SupportTicketSubmission

MicrosoftAppPlatEMA
Microsoft AppPlat EMA

MicrosoftAzureBatch
Microsoft Azure Batch

MyTestApp

RI Test Key Vault

You then need to assign permissions

Keys vs Secrets

Add Packages

nuget.org

Search: keyvault

<input type="checkbox"/>		Microsoft.Azure.KeyVault	3,036,382	Microsoft.Azure.KeyVault Azure Key Vault enables users to store and use cryptographic keys within the Microsoft Azure environment. Azure Key Vault supports multiple key types and algorithms and enables the use of Hardware Security Modules (HSM) for high value customer keys.
<input type="checkbox"/>		Microsoft.Azure.KeyVault.WebKey	2,129,471	Microsoft Azure Key Vault WebKey Class Library
<input type="checkbox"/>		Microsoft.Azure.Management.KeyVault.Fluent	96,325	Provides Key Vault service management (Fluent) capabilities for Microsoft Azure.
<input type="checkbox"/>		Sandboxable.Microsoft.Azure.KeyVault	446	This library is adjusted to run in a Partial Trust environment. Provides key and secret operation capabilities for Azure Key Vault.
<input type="checkbox"/>		Lingmin.NuGet.Services.KeyVault	127	Key vault access for NuGet services
<input type="checkbox"/>		Sandboxable.Microsoft.Azure.Management.KeyVault	409	This library is adjusted to run in a Partial Trust environment. Provides Microsoft Azure Key Vault management functions for managing key vaults.
<input type="checkbox"/>		KeyVaultBuildTask	510	Pulls values from KeyVault during the build process
<input type="checkbox"/>		Microsoft.Azure.KeyVault.Core	9,834,930	Microsoft Azure Key Vault Core Class Library
<input type="checkbox"/>		Microsoft.Azure.KeyVault.Extensions	381,537	Class library for Azure Key Vault Extensions

Show pre-release packages

Version 2.3.2

Close **Add Package**

HSM Patterns and Practices

Multiple Environments

Configuration as
Secrets

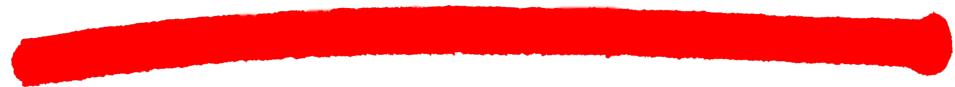
Local Key Wrapping

Key Versioning

Password Protection

Digital Signing

Multiple Vaults for Environments





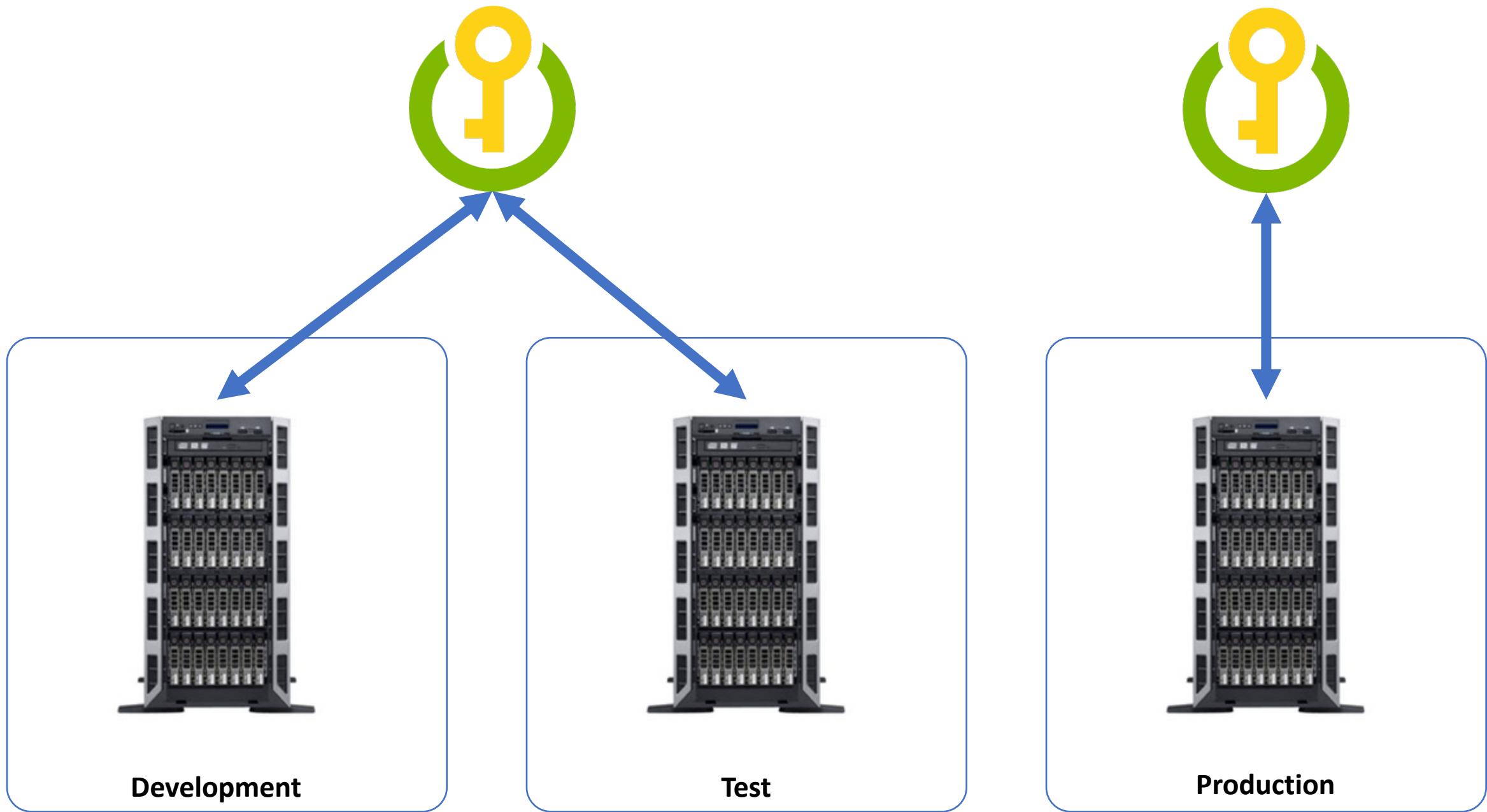
Development

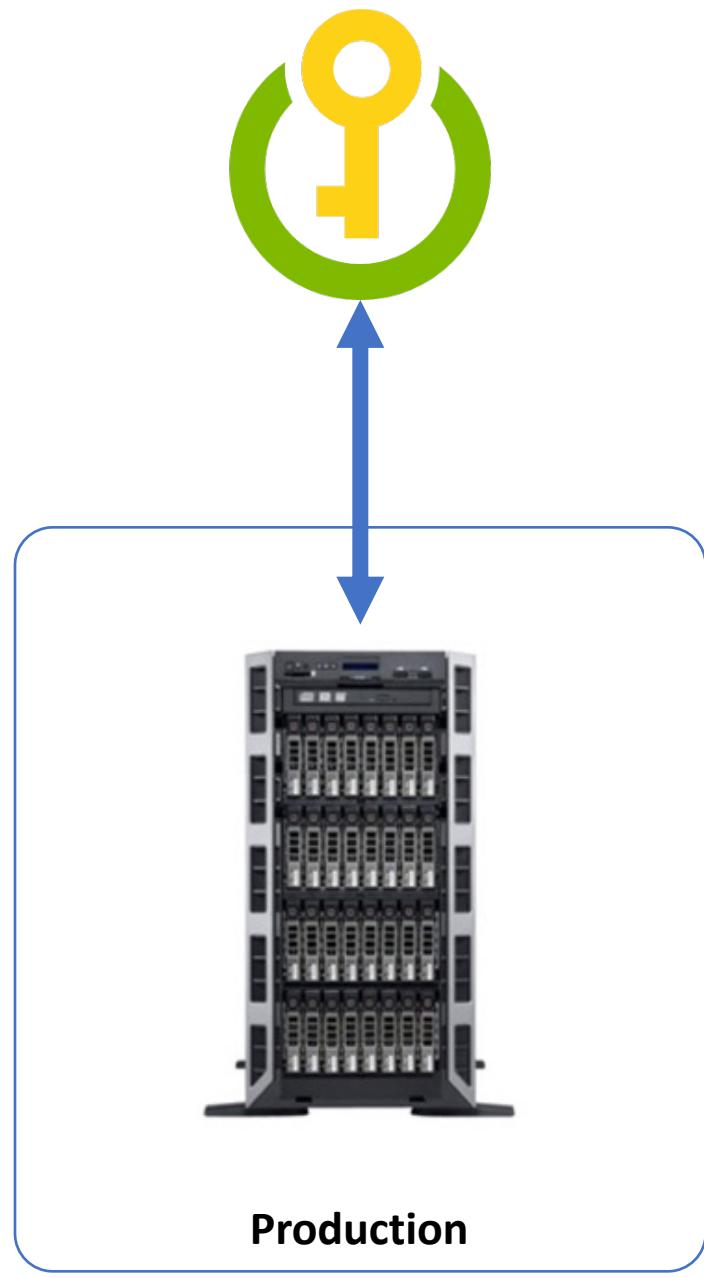
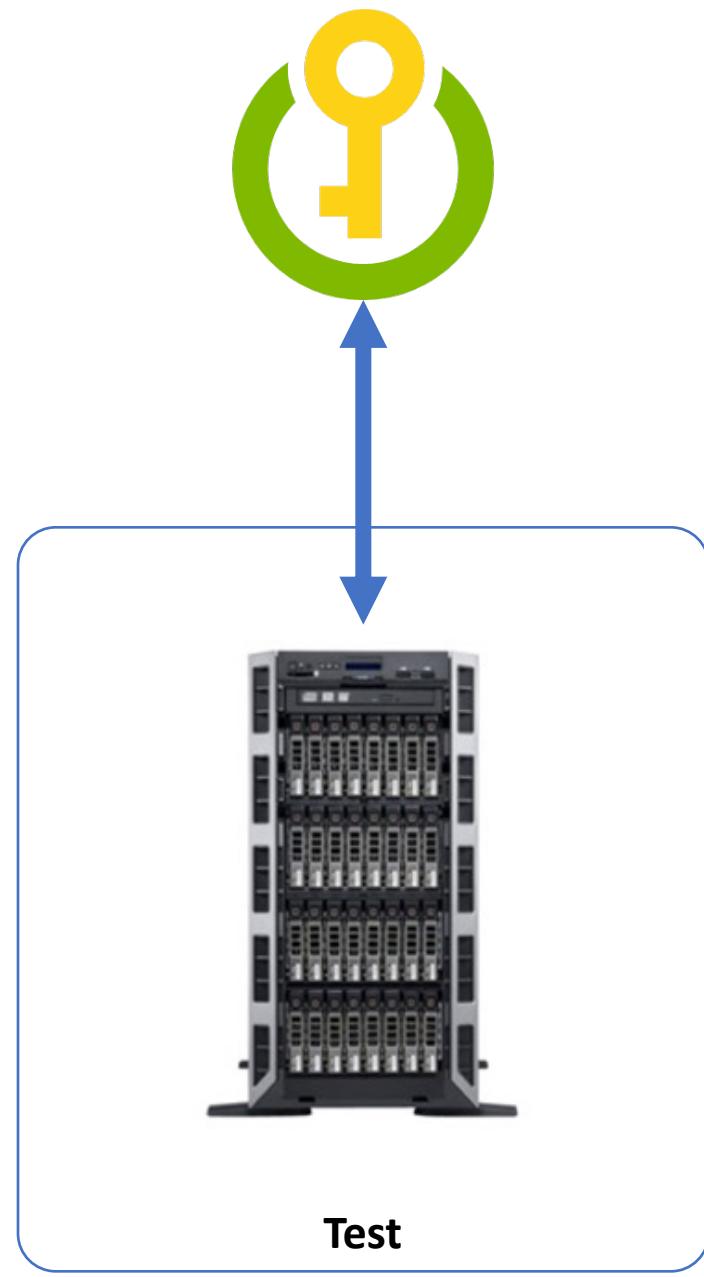
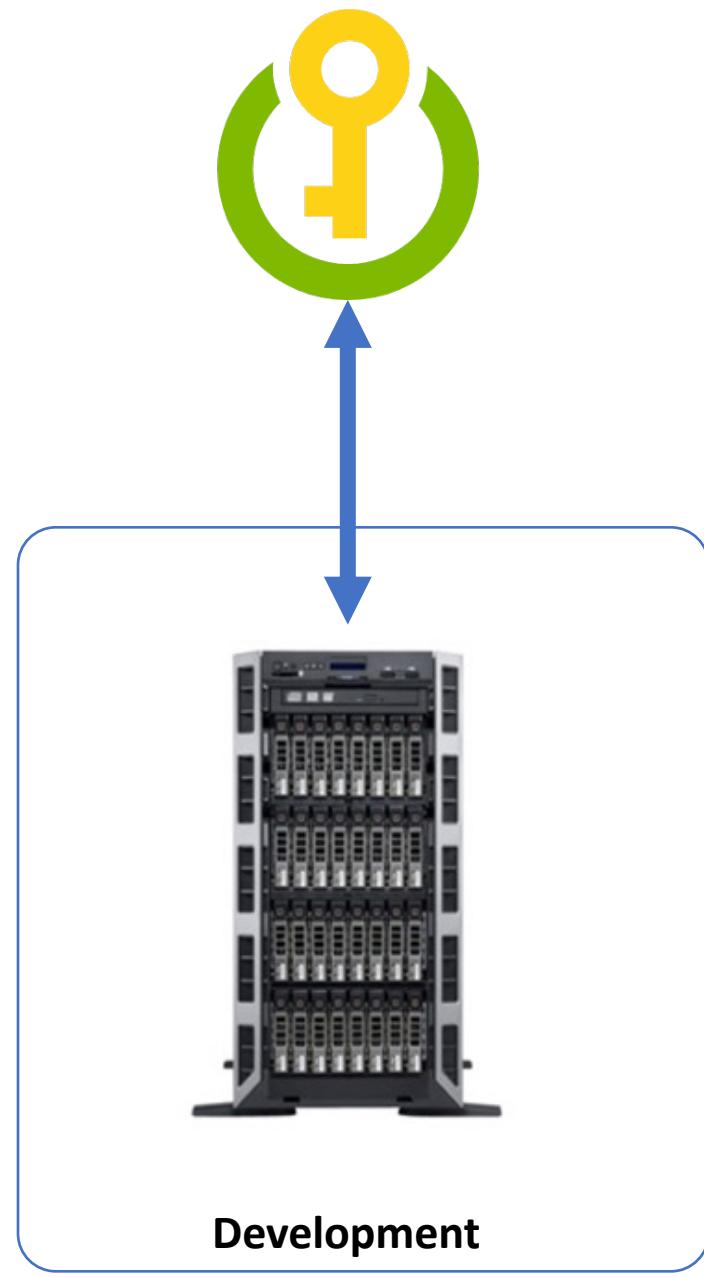


Test



Production





NEVER share keys from
production in any other
environment

Configuration as Secrets



Secrets are max 25k blobs of text

Stored in software but encrypted
with key in HSM

Connection Strings

API Keys

Redis Cache
Connection Strings

Confidential
Application Settings

Anything Else
Secret.....

Secrets are versioned in the Key
Vault



Local Key Wrapping



Encrypting on the HSM is expensive

Use a hybrid solution

Encrypt locally with AES

Encrypt AES Key with the HSM

Store Encrypted AES key in the database



SaaS claims management system

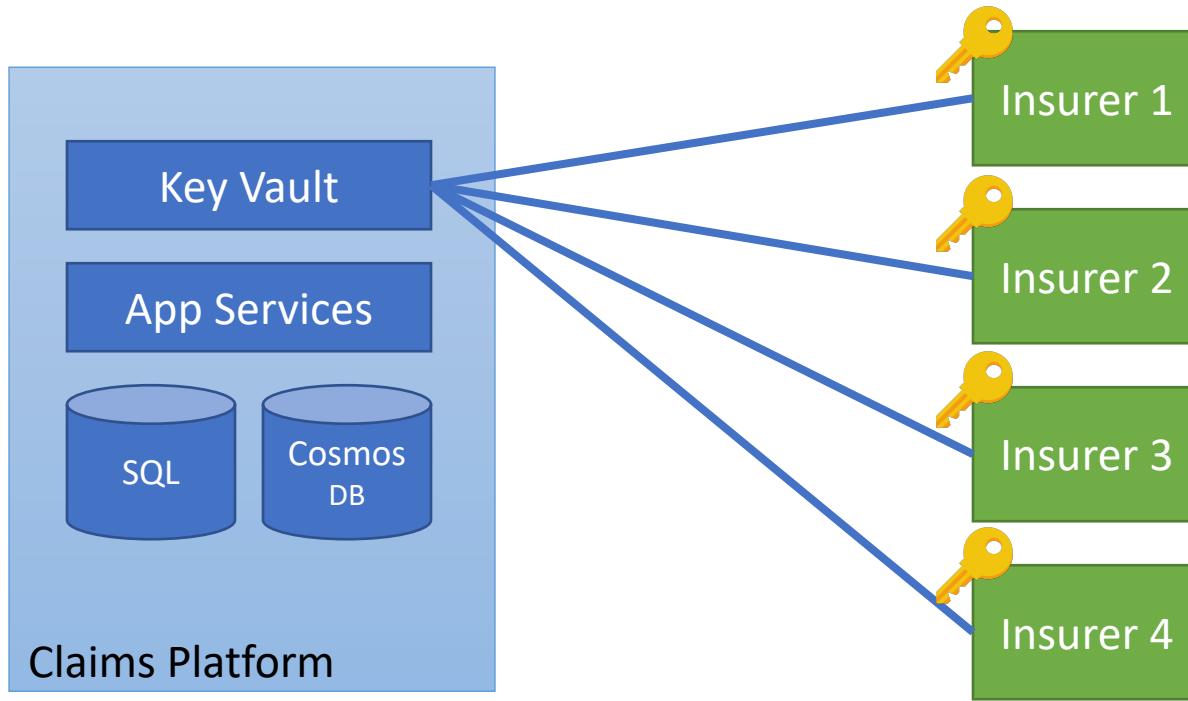
Multiple companies sharing same platform

Multi tenant with custom branding

Insurers had own master keys in Key Vault

Insurers PII data encrypted using their own keys

Partitioning via encryption



Tenant Id	Key Id	Encrypted AES Session Key
1	1	38r7289j5n79840n58v40m580v4043n85v0=
2	2	Hdkjfrwi5yuioweurowveirunvowiortvno98v=
3	3	kjdfngkdfmuto8wenrowven75ownro8ownr=
4	4	No3874n85748r898908wtirhgiwmc8wun9t=



Local encryption operations are faster

Decrypt key with the HSM and store in memory

Reduces network hops and latency

Easy to rotate keys



Create new key on the HSM

Re-encrypt the local AES key only

Doesn't require re-encrypt of all data

Key Versioning

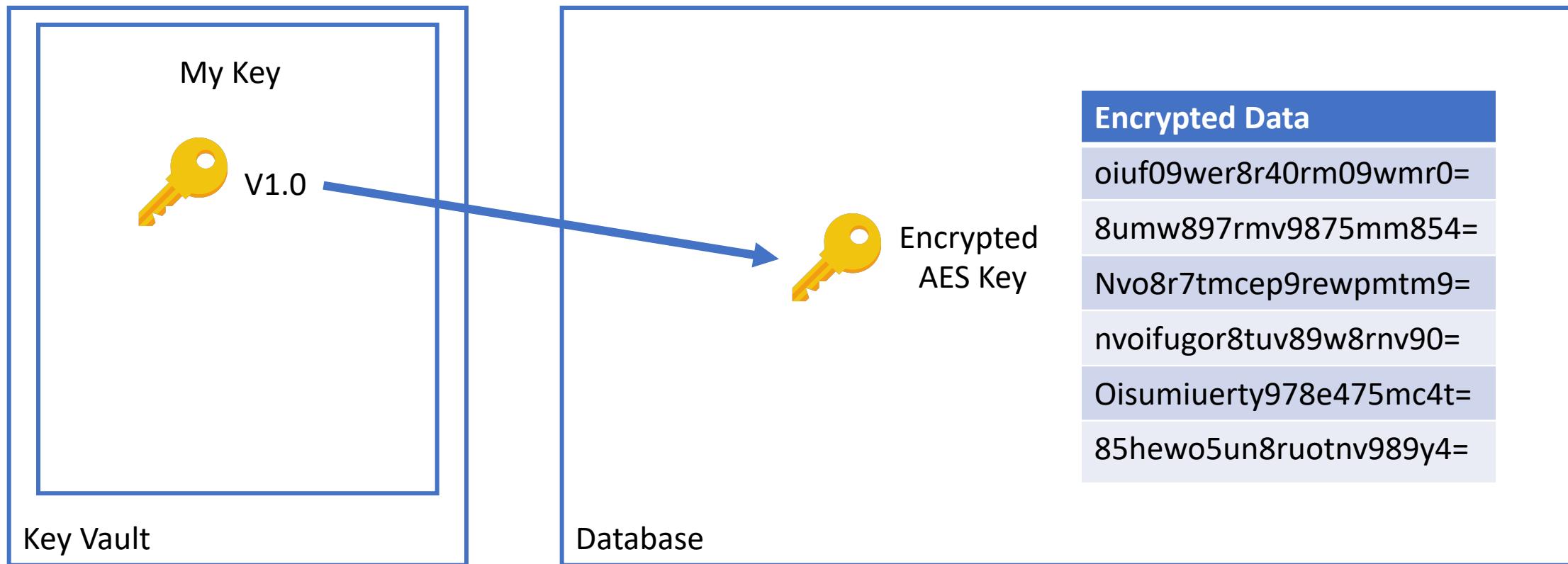




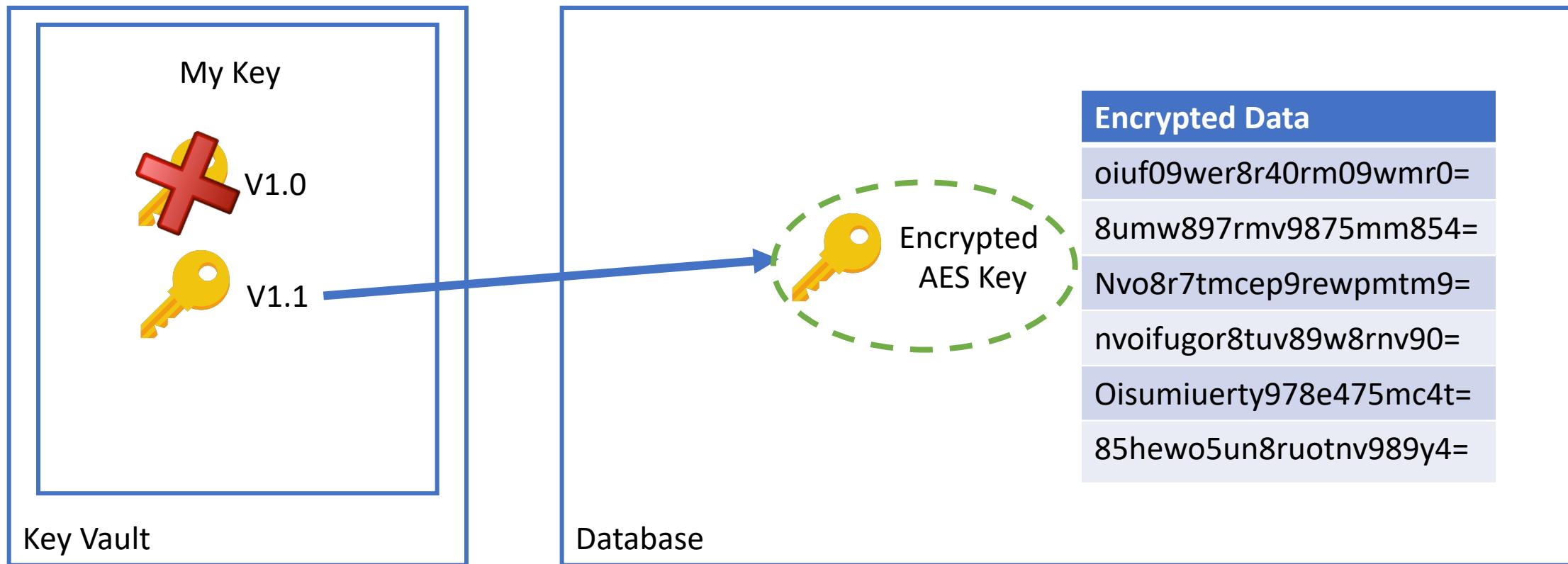
Changing keys over time is important

Digital equivalent of changing the locks

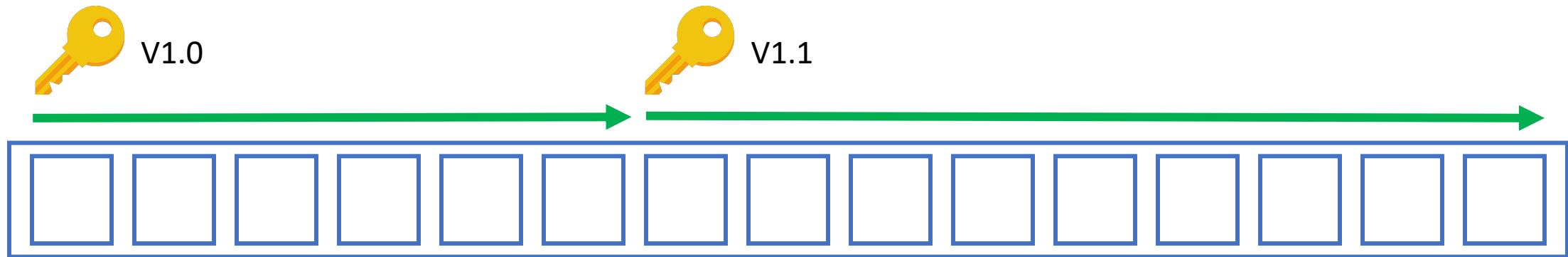
Easy for Local Key Wrapping



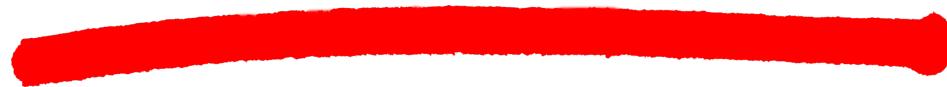
Easy for Local Key Wrapping



Incremental Keys



Password Protection

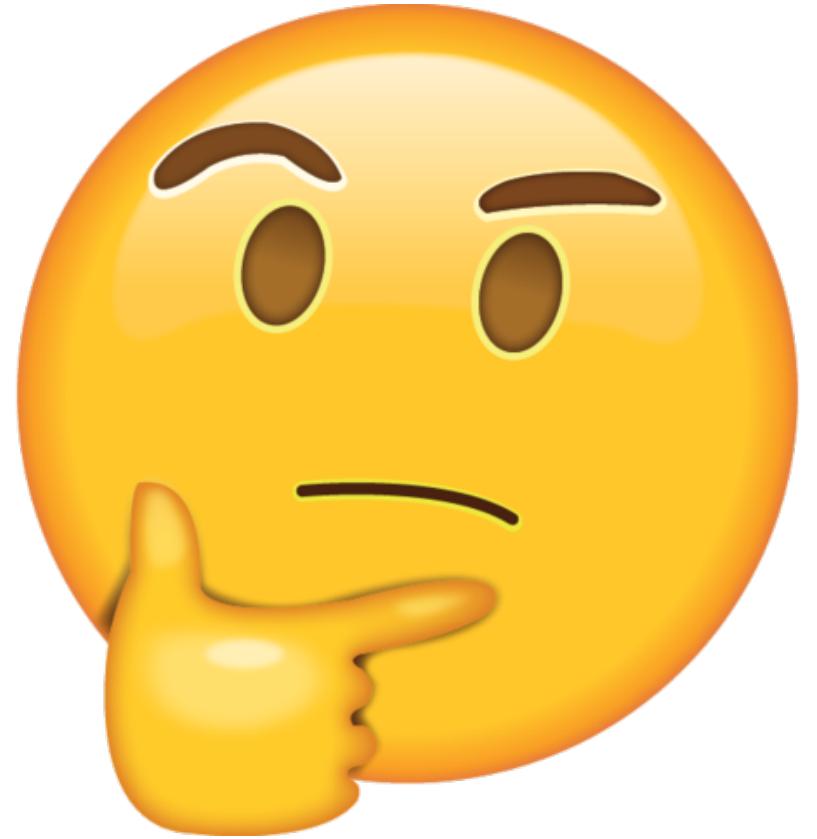


Passwords aren't great

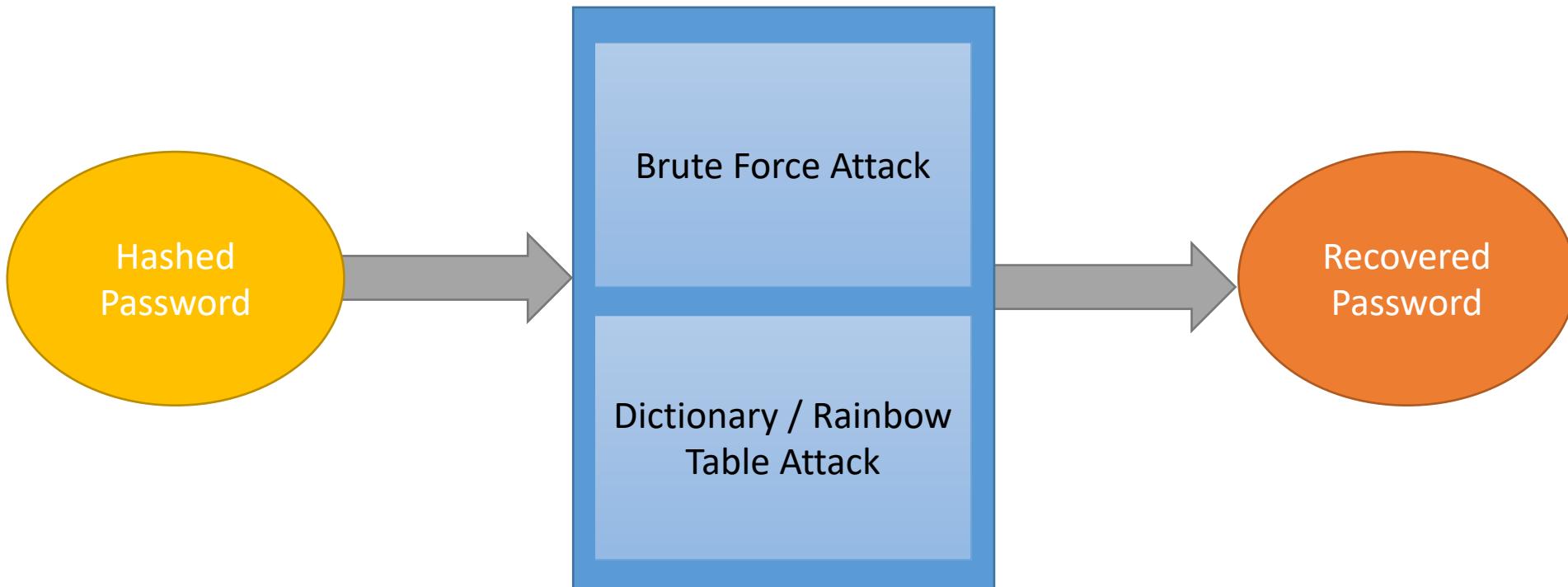
BUT, they are here to stay



Plaintext
Passwords



Hashes
Passwords



Free Password Hash Cracker

Enter up to 10 non-salted hashes:

```
89c30f594cd6163641bd1d8e227402b7ac9f7e77650cfef5ef43b9940433e0b37
```



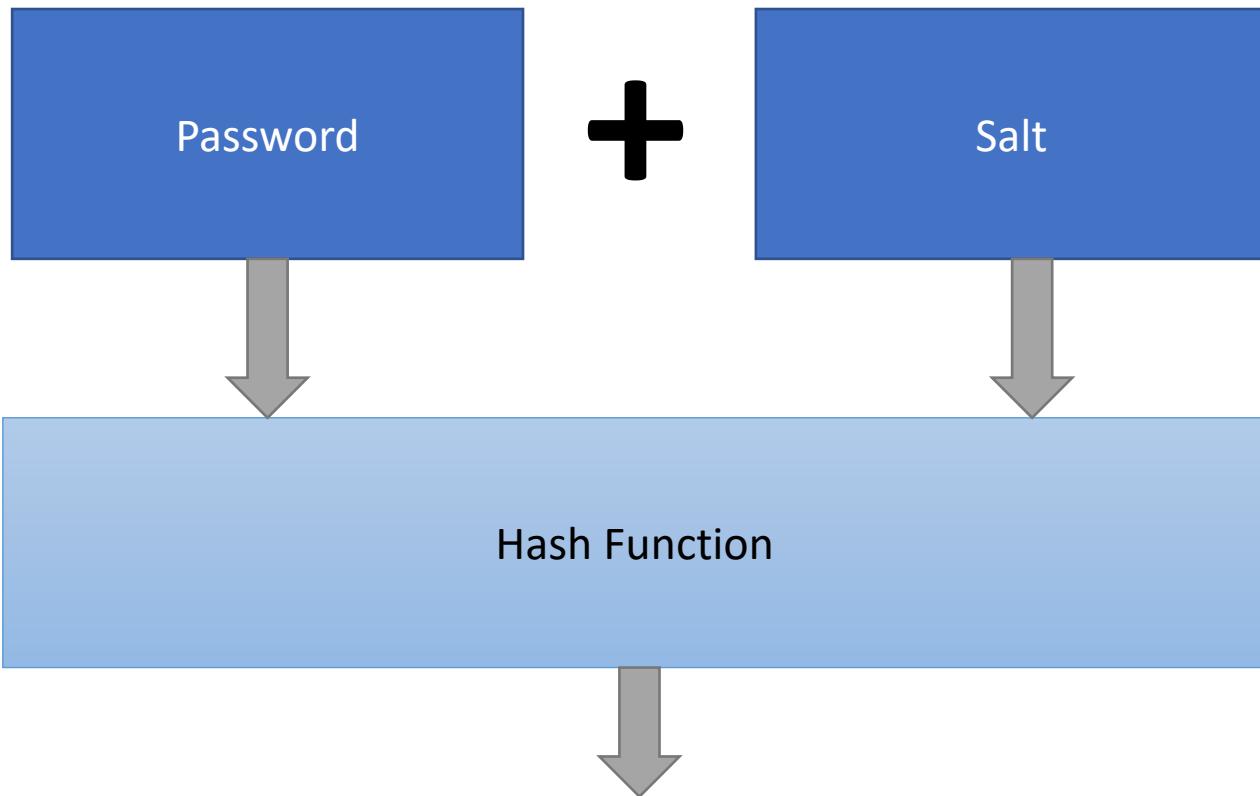
Supports: LM, NTLM, md2, md4, md5, md5(md5), md5-half, sha1, sha1(sha1_bin()), sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+

Hash	Type	Result
89c30f594cd6163641bd1d8e227402b7ac9f7e77650cfef5ef43b9940433e0b37	sha256	Secret69

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.



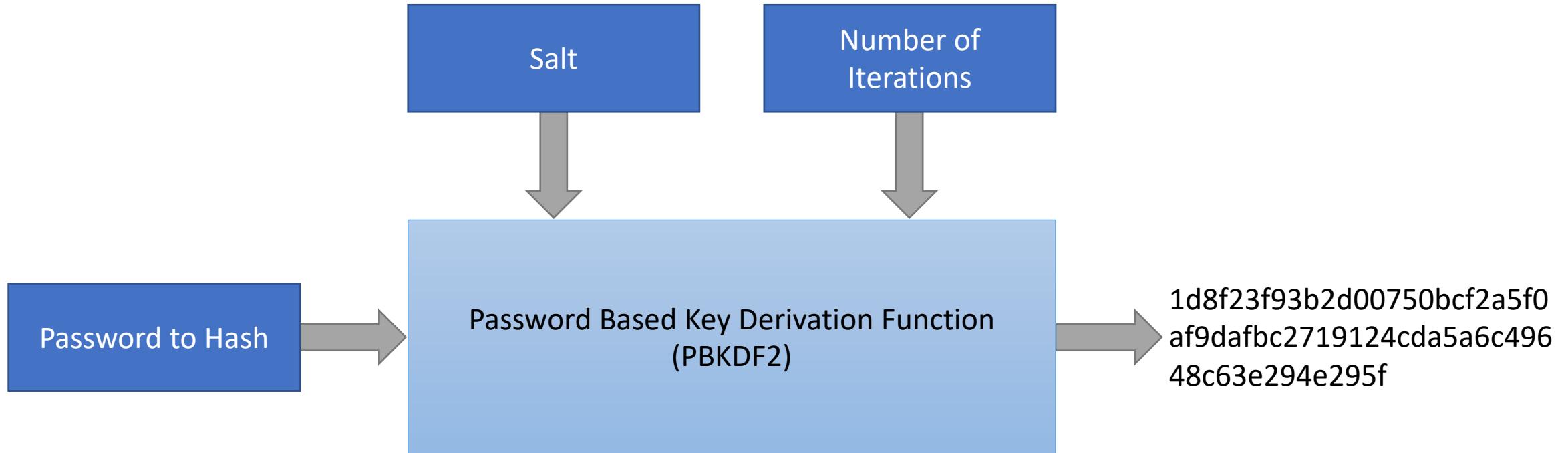
Salted Hash
Passwords



1d8f23f93b2d00750bcf2a5f0af9dafbc2719124cda5a6c49648c63e294e295f

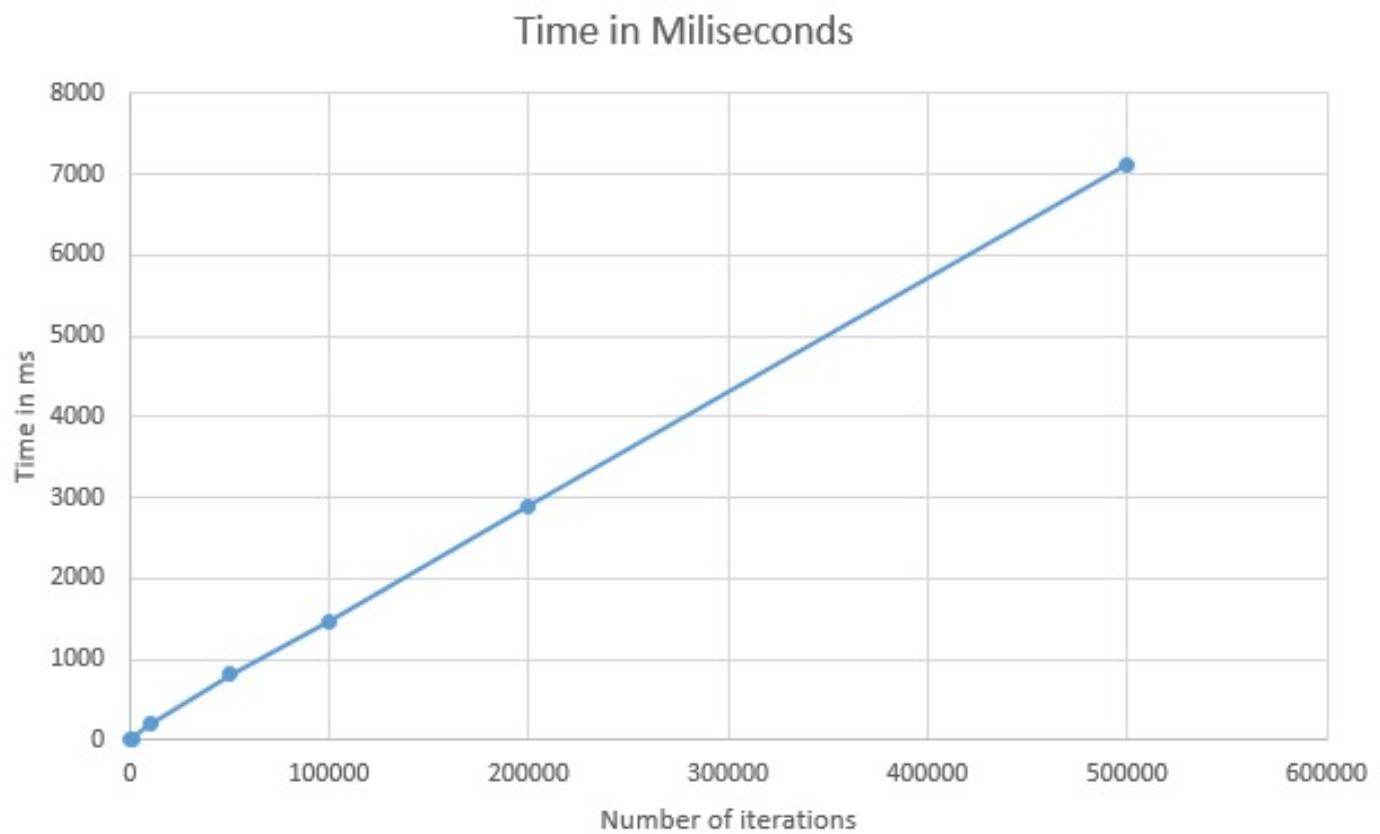


PBKDF2 or
BCrypt
Passwords



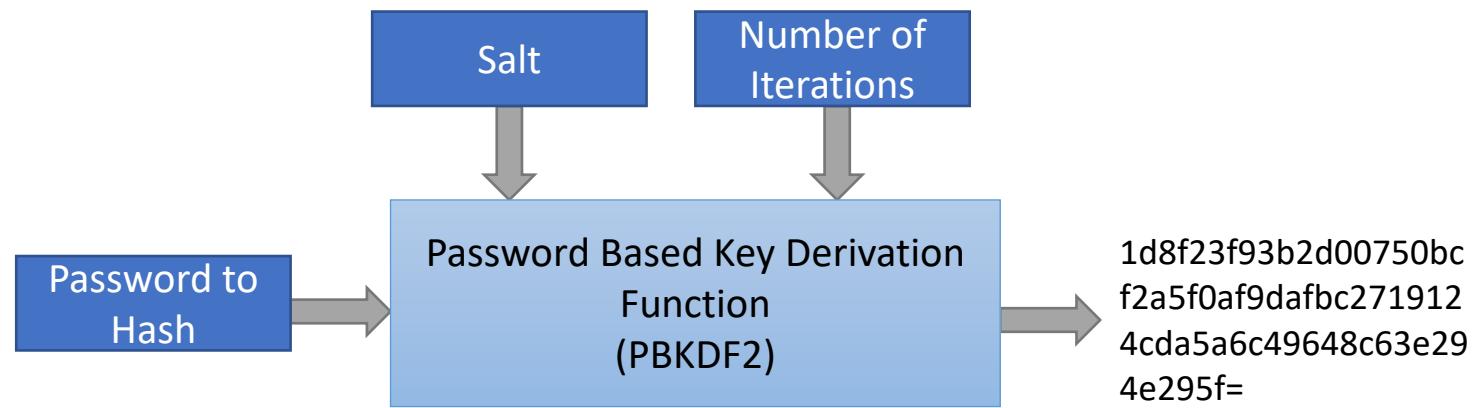
```
public static byte[] HashPassword(byte[] toBeHashed, byte[] salt, int ittr)
{
    using (var rfc2898 = new Rfc2898DeriveBytes(toBeHashed, salt, ittr))
    {
        return rfc2898.GetBytes(20);
    }
}
```

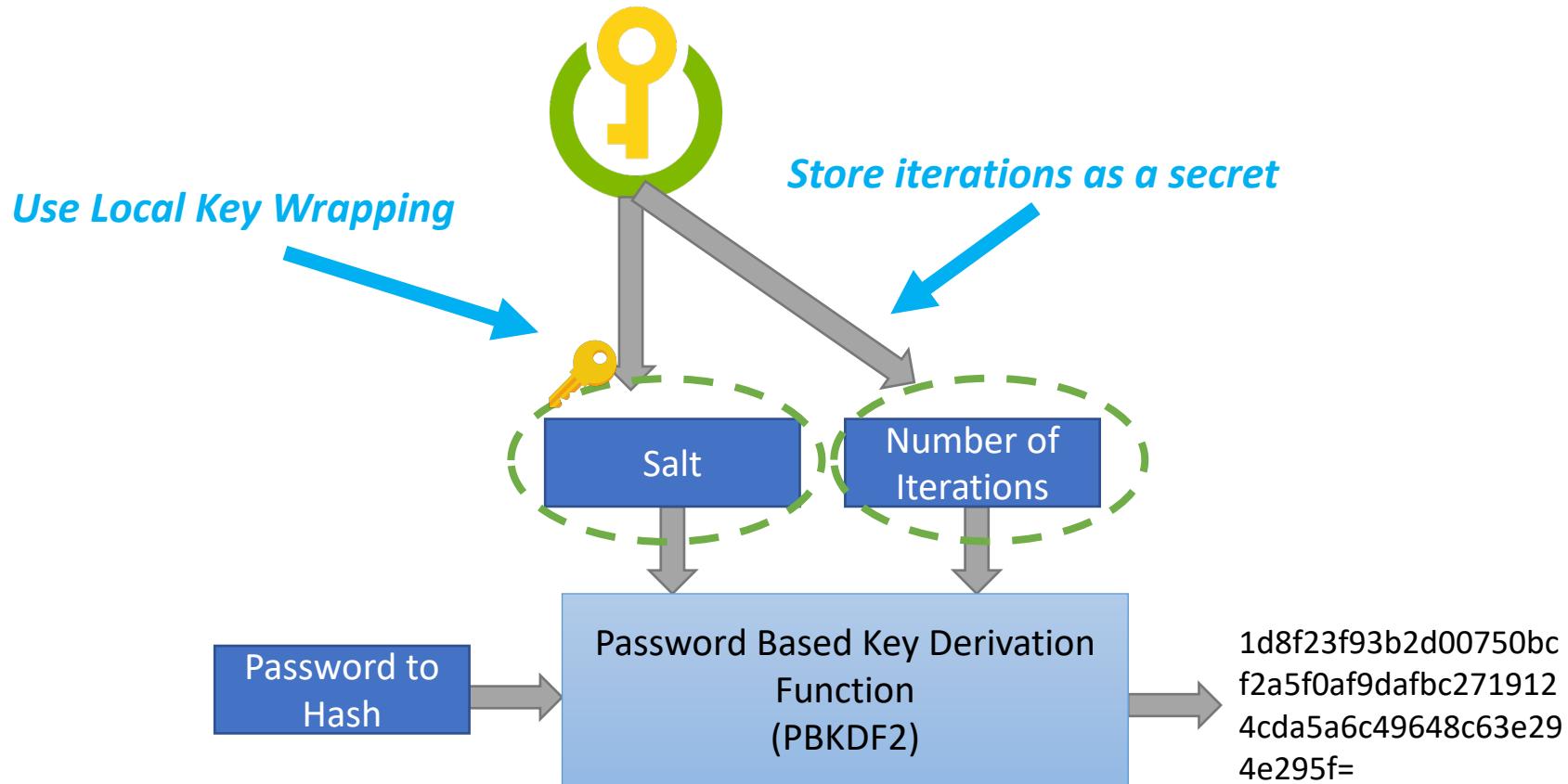
Num of Iterations	Time (ms)
100	2
1000	16
10,000	196
50,000	807
100,000	1474
200,000	2892
500,000	7172





But we are
already good?







Password Table Stolen



Attackers can not even start trying to crack the PBKDF2

They first have to perform the HMAC

But the key is encrypted by Key Vault

Makes dictionary / rainbow table attack much harder

Good practice to use different HMAC /
Vault Keys for different roles, it Admin,
internal employee, customers etc.

Digital Signing



Confidentiality

AES and RSA

Integrity

SHA-256

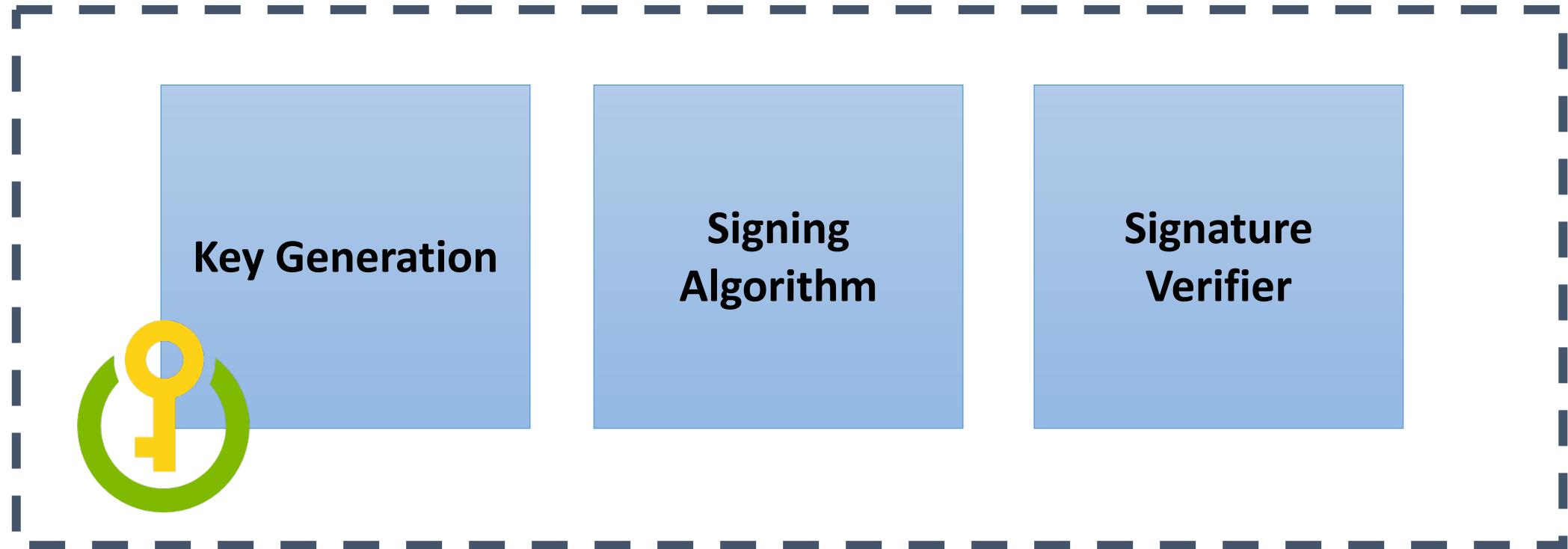
Authentication

HMAC (SHA-256)

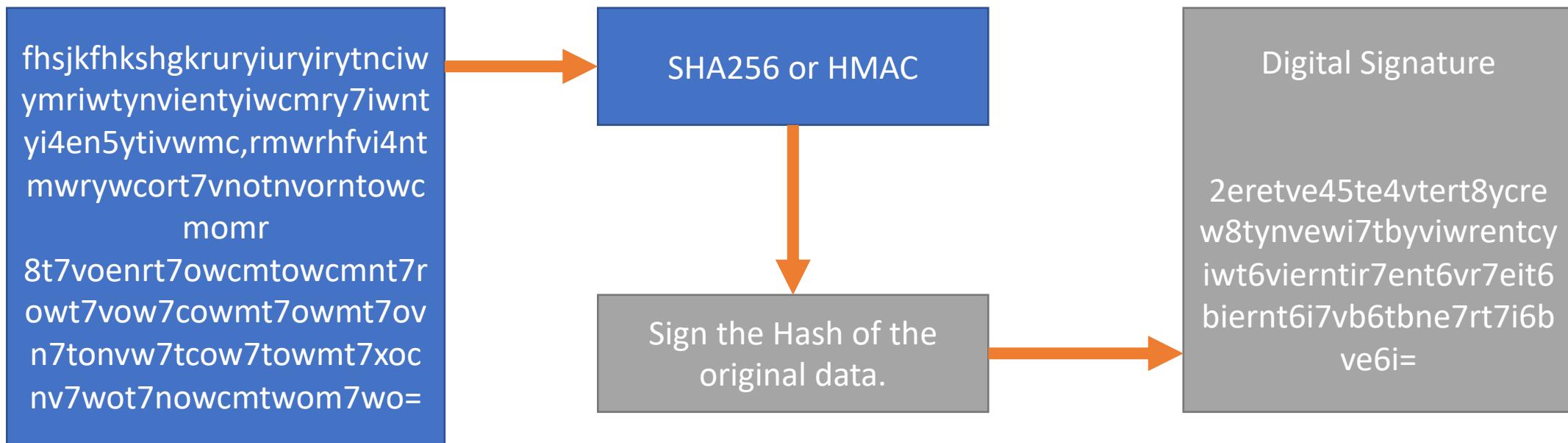
Non-Repudiation

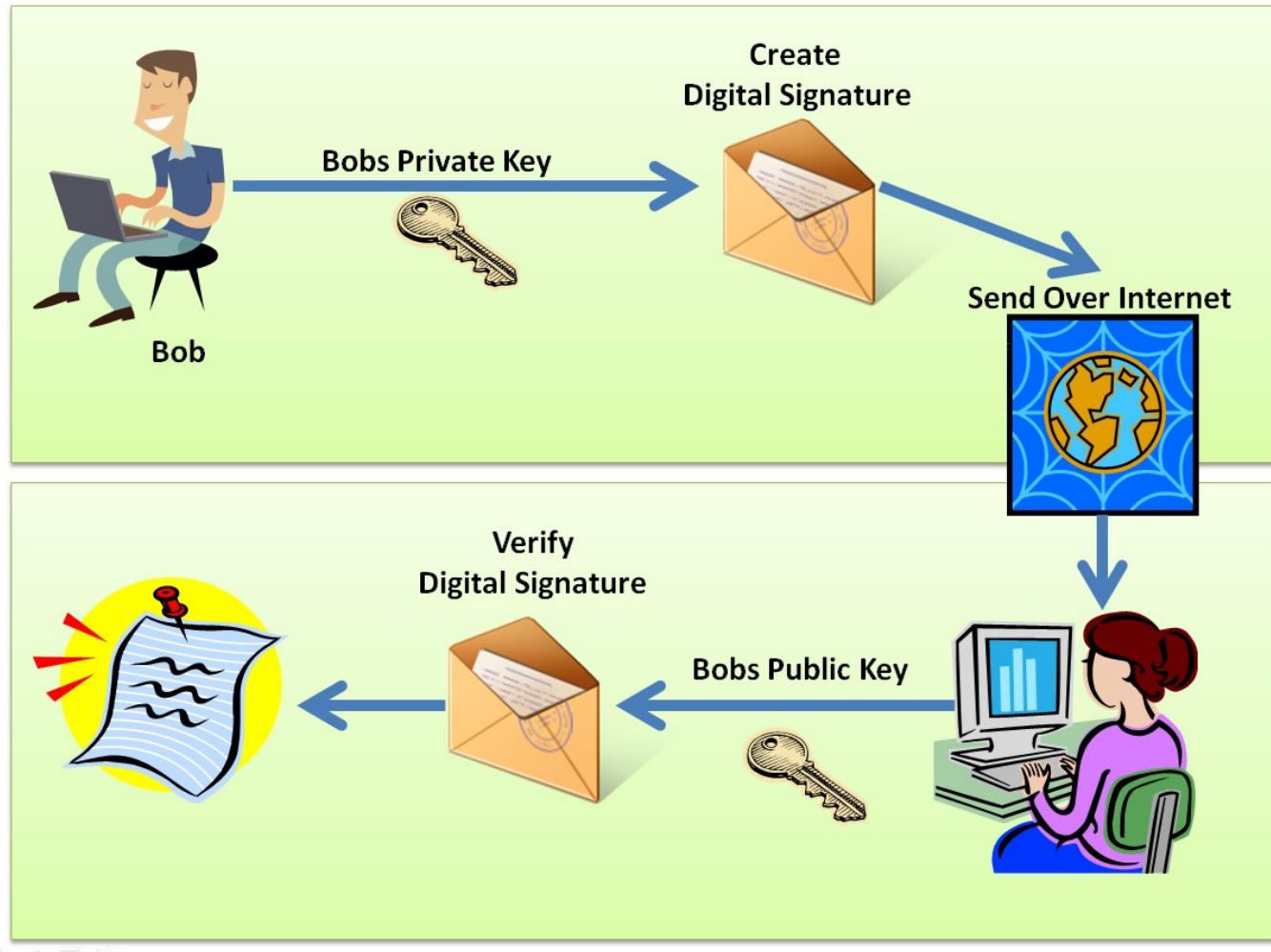
Digital Signature

A digital is a mathematical technique used to validate the authenticity and integrity of a message, software or digital document.



Calculate the Signature from a hash of your data





RSAPKCS1SignatureFormatter

RSAPKCS1SignatureDeFormatter



Azure Key Vault offers Digital Signatures natively

No need to use RSAPKCS1SignatureFormatter





Now it's
your turn!

Play with the Key Vault Example to understand them

Advanced : Update the one of the hybrid encryption examples to use the Key Vault to wrap the symmetric key instead of RSACryptoServiceProvider.



Applied Cryptography

Blockchain Principles and Practices

Thinking About Trust

"The blockchain is an incorruptible digital ledger of economic transactions that can be programmed to record not just financial transactions but virtually everything of value"

Don and Alex Tapscott : Blockchain Revolution

Thinking About Trust

"The blockchain is an incorruptible digital ledger of economic transactions that can be programmed to record not just financial transactions but virtually everything of value"

Don and Alex Tapscott : Blockchain Revolution

"A blockchain is a continuously growing list of records, called blocks, which are linked and secured using cryptography"

Wikipedia

Thinking About Trust

*"The blockchain is an incorruptible **digital ledger** of economic transactions that can be programmed to record not just financial transactions but virtually everything of value"*

Don and Alex Tapscott : Blockchain Revolution

"A blockchain is a continuously growing list of records, called blocks, which are linked and secured using cryptography"

Wikipedia

Thinking About Trust

*"The blockchain is an incorruptible **digital ledger** of economic transactions that can be programmed to record not just financial transactions but virtually everything of value"*

Don and Alex Tapscott : Blockchain Revolution

*"A blockchain is a **continuously growing list of records**, called blocks, which are linked and secured using cryptography"*

Wikipedia

Thinking About Trust

*"The blockchain is an **incorruptible digital ledger** of economic transactions that can be programmed to record not just financial transactions but virtually everything of value"*

Don and Alex Tapscott : Blockchain Revolution

*"A blockchain is a **continuously growing list of records**, called blocks, which are linked and secured using cryptography"*

Wikipedia

Thinking About Trust

*"The blockchain is an **incorruptible digital ledger** of economic transactions that can be programmed to record not just financial transactions but virtually everything of value"*

Don and Alex Tapscott : Blockchain Revolution

*"A blockchain is a **continuously growing list of records**, called blocks, which are **linked and secured using cryptography**"*

Wikipedia

Thinking About Trust

Incorruptible

Digital ledger

Continuously growing list of records

Linked and secured using cryptography

A father stands on a large, textured rock, reaching up with both arms to catch his young son as he jumps off another rock. The son is in mid-air, wearing a dark t-shirt and jeans, with his arms raised. The background shows more large rocks and a bright sky.

TRUST



BANK

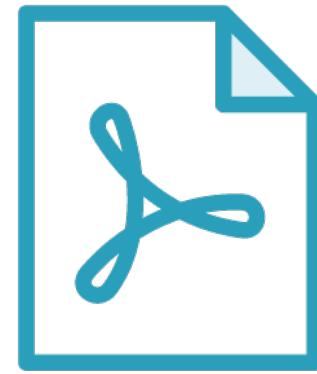




<https://news.bitcoin.com/evolution-bitcoin-vending-machine/>

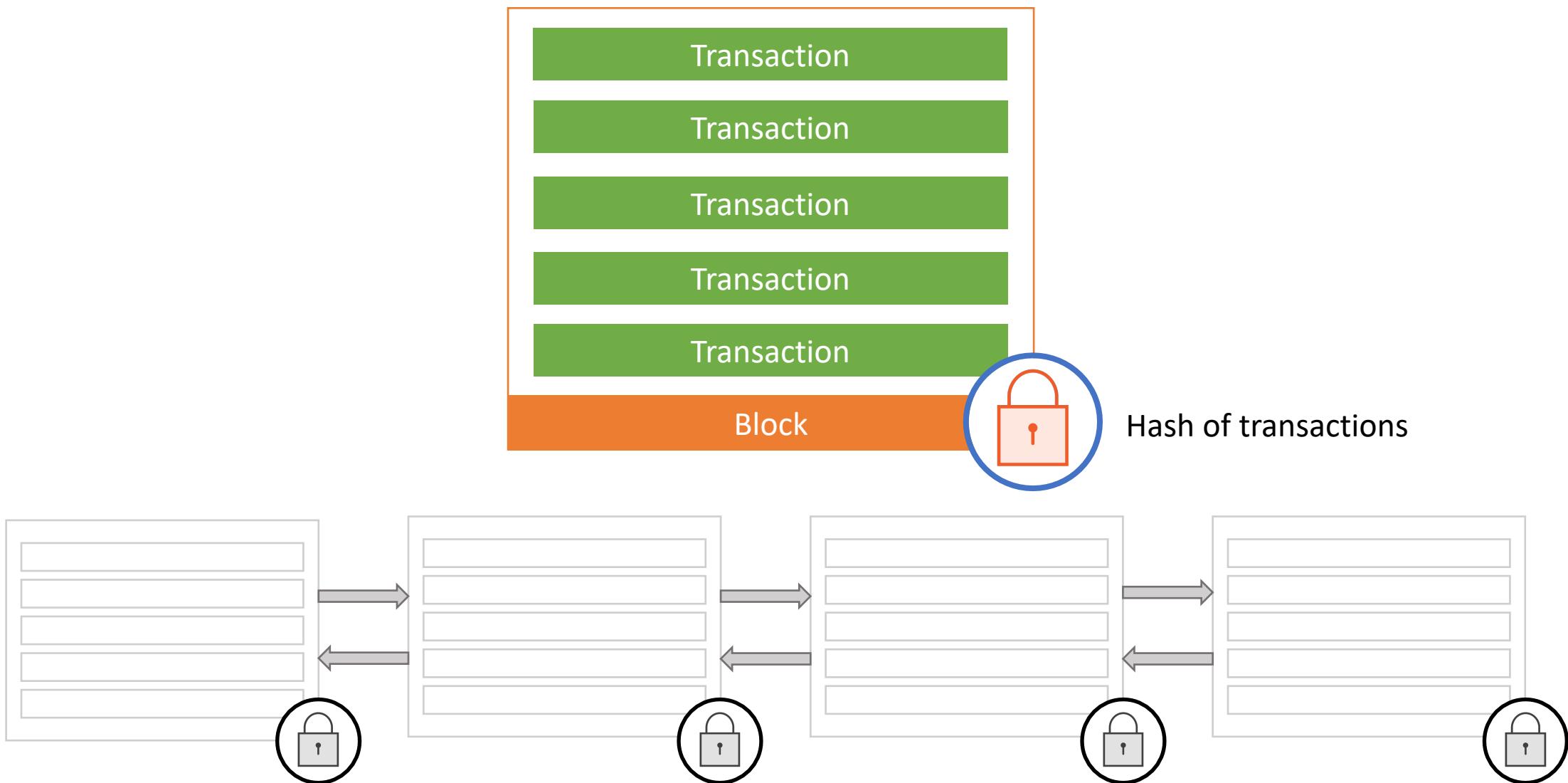


Satoshi Nakamoto



<https://bitcoin.org/bitcoin.pdf>

Introducing Blockchain



Introducing Blockchain

10 minutes to calculate each block

Introducing Blockchain

10 minutes to calculate each block

1000 blocks per chain

Introducing Blockchain

10 minutes to calculate each block

1000 blocks per chain

166 hours to calculate chain

Introducing Blockchain

10 minutes to calculate each block

1000 blocks per chain

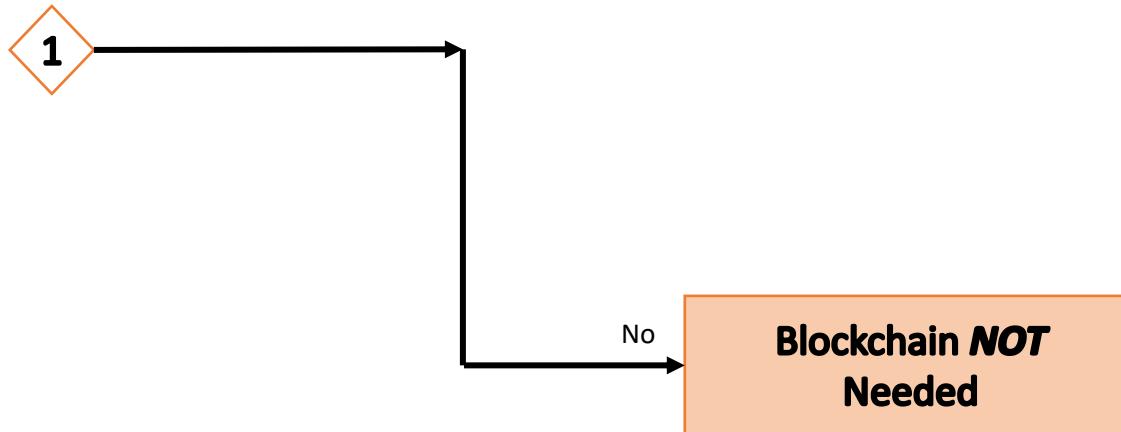
166 hours to calculate chain

Nearly 7 days to recalculate chain

Public vs Private Blockchain

Do I Need a Blockchain?

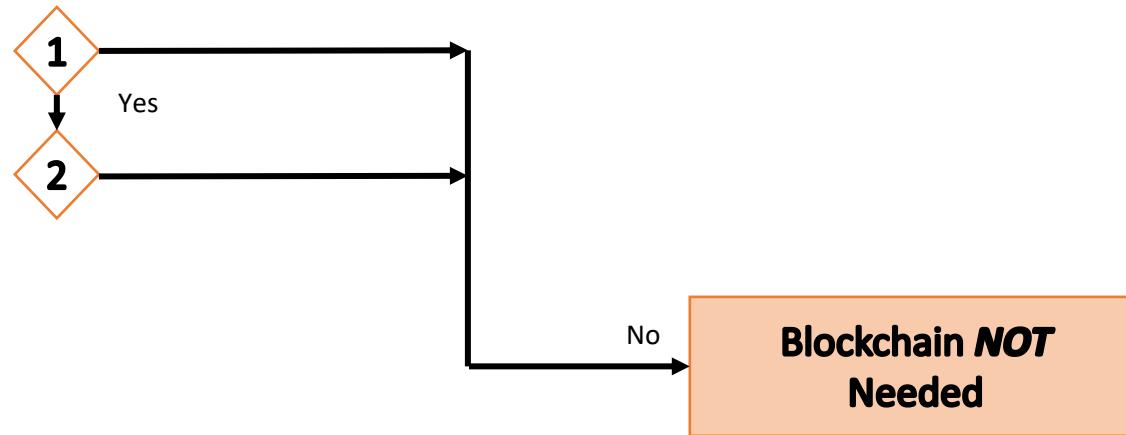
1. Do you need a shared, public database?



Do I Need a Blockchain?

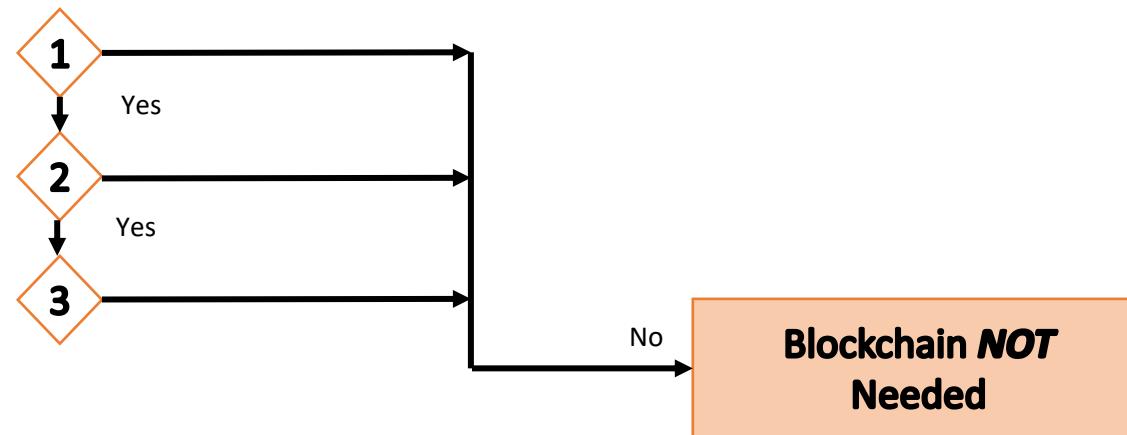
1. Do you need a shared, public database?

2. Are there multiple parties involved?



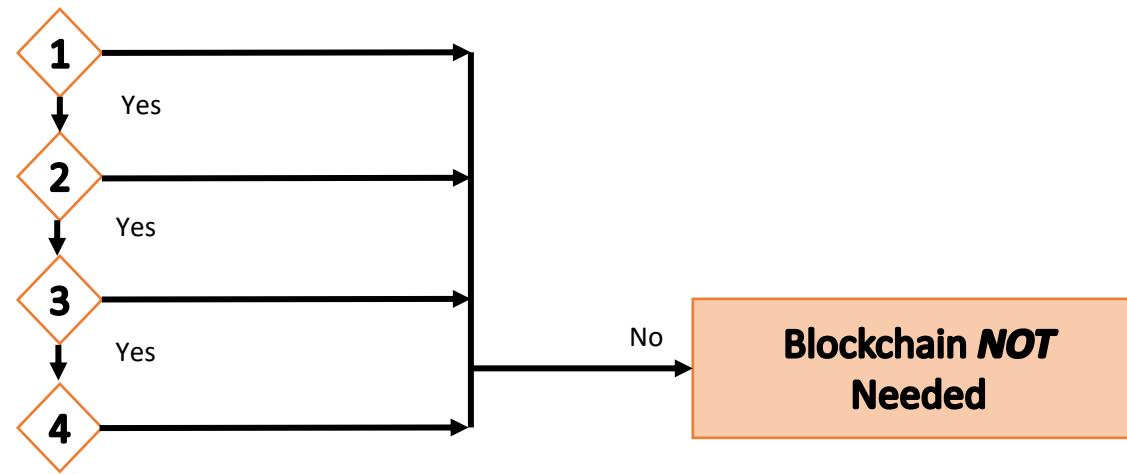
Do I Need a Blockchain?

1. Do you need a shared, public database?
2. Are there multiple parties involved?
3. Do these parties have conflicting Incentive and/or are not trusted



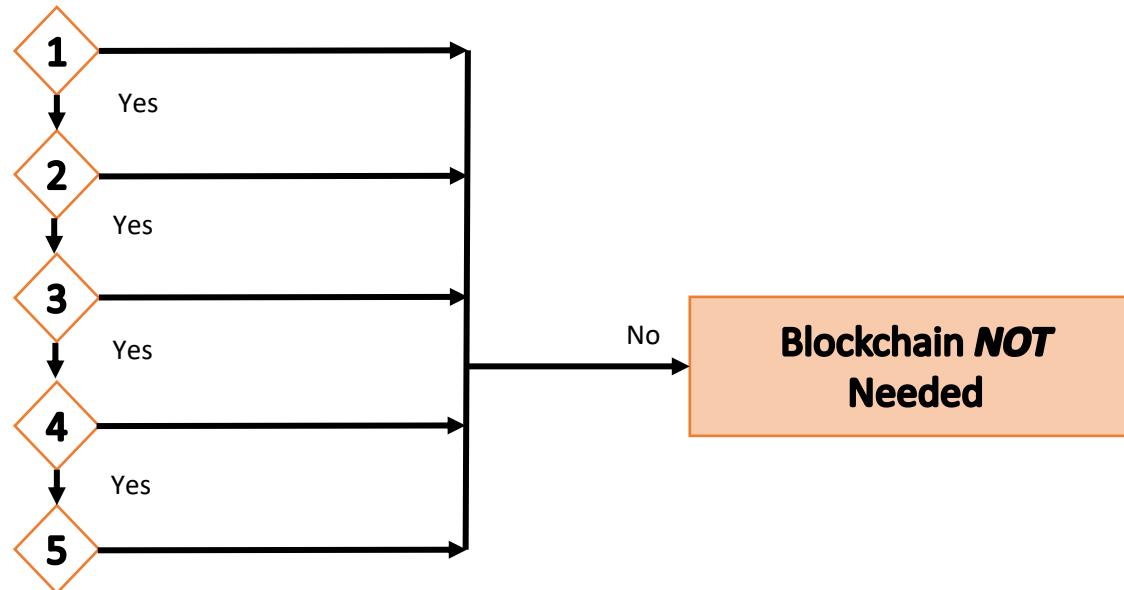
Do I Need a Blockchain?

1. Do you need a shared, public database?
2. Are there multiple parties involved?
3. Do these parties have conflicting Incentive and/or are not trusted
4. Parties able to play by the same rules?



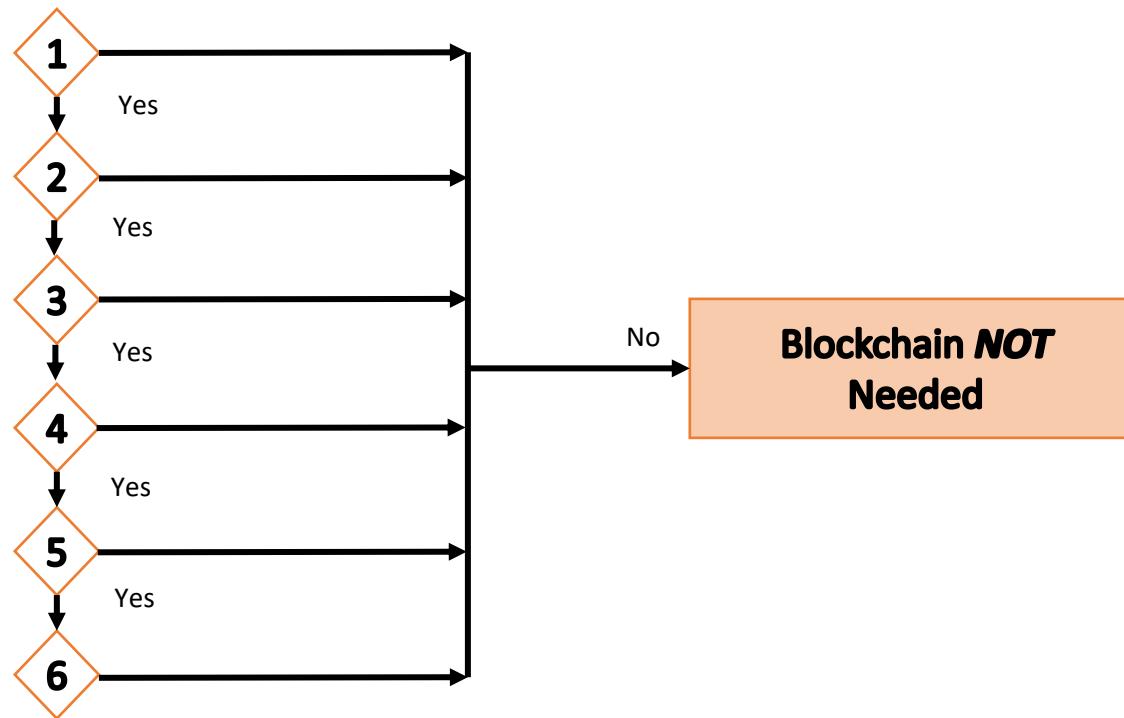
Do I Need a Blockchain?

1. Do you need a shared, public database?
2. Are there multiple parties involved?
3. Do these parties have conflicting Incentive and/or are not trusted
4. Parties able to play by the same rules?
5. Do you need an immutable log of all Actions to be recorded?



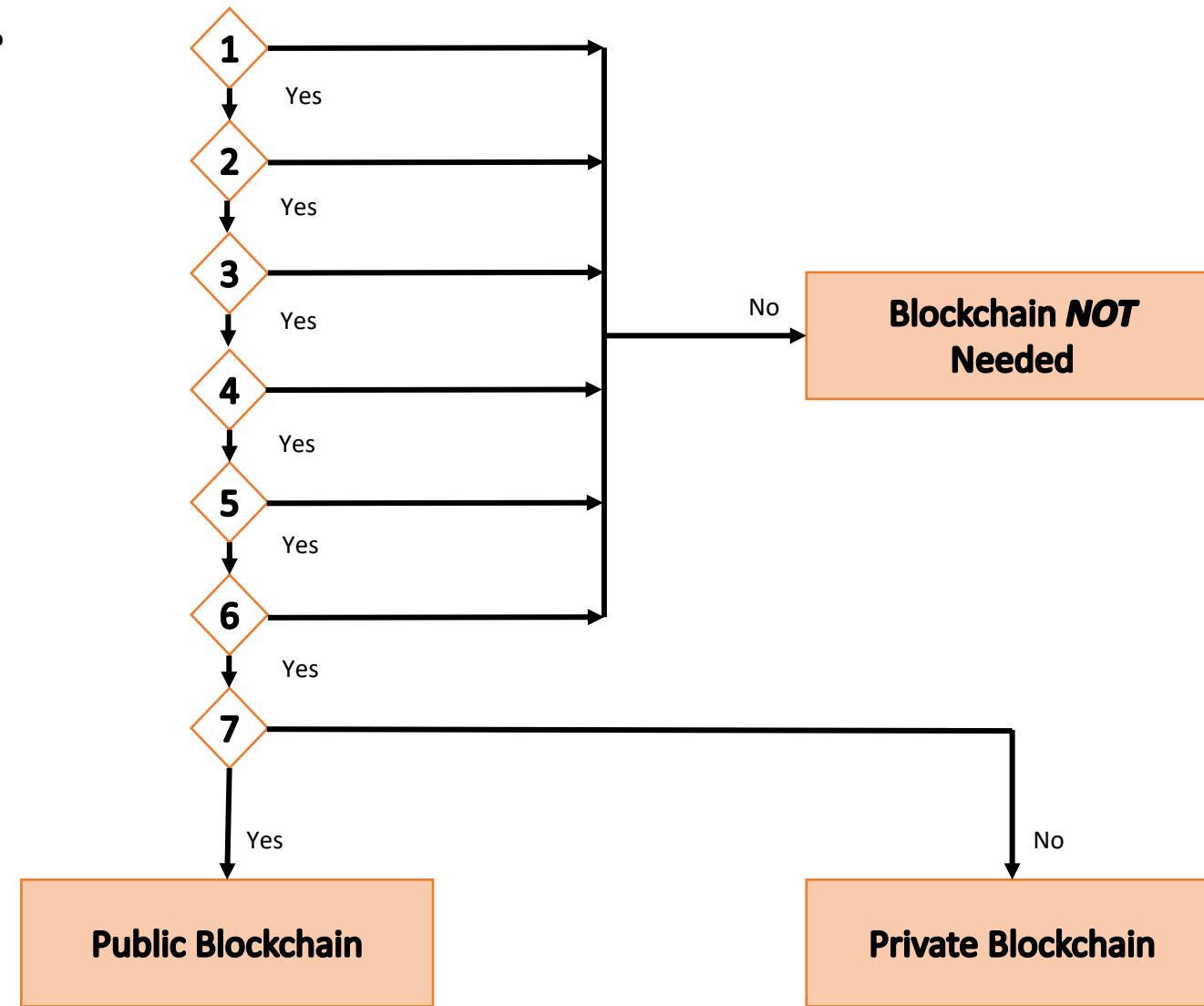
Do I Need a Blockchain?

1. Do you need a shared, public database?
2. Are there multiple parties involved?
3. Do these parties have conflicting Incentive and/or are not trusted
4. Parties able to play by the same rules?
5. Do you need an immutable log of all Actions to be recorded?
6. Transaction rules do not change frequently



Do I Need a Blockchain?

1. Do you need a shared, public database?
2. Are there multiple parties involved?
3. Do these parties have conflicting Incentive and/or are not trusted
4. Parties able to play by the same rules?
5. Do you need an immutable log of all Actions to be recorded?
6. Transaction rules do not change frequently
7. Can transactions be public



Why Use a Blockchain Over a Database?

Blockchain

Decentralized trust as multiple nodes contain data and mine blocks

Block creation requires consensus

Immutable data that can never be changed

Ideal for transactional data

Write speed into the blockchain not an issue

Data integrity guaranteed

Smart contracts turn Blockchain into a computing platform

Why Use a Database Over a Blockchain?

Database

Fast storage and performance

Replication for redundancy

Doesn't have per "transaction" fees like a public Blockchain

Data easily amended or deleted

Not as computationally expensive

Easier to manage

Public vs Private Blockchain

Public Blockchain

Anyone can write to the blockchain

Public vs Private Blockchain

Public Blockchain

Anyone can write to the blockchain

Every node contains a copy of the complete chain

Public vs Private Blockchain

Public Blockchain

Anyone can write to the blockchain

Every node contains a copy of the complete chain

Best security and trust between peers

Public vs Private Blockchain

Public Blockchain

Anyone can write to the blockchain

Every node contains a copy of the complete chain

Best security and trust between peers

Hashing puzzles are time consuming

Public vs Private Blockchain

Public Blockchain

Anyone can write to the blockchain

Every node contains a copy of the complete chain

Best security and trust between peers

Hashing puzzles are time consuming

Some enterprises nervous about public blockchain

Public vs Private Blockchain

Private Blockchain

Private blockchains have been controversial

Public vs Private Blockchain

Private blockchains have been controversial

Private Blockchain

Enterprises favour a more controlled blockchain

Public vs Private Blockchain

Private blockchains have been controversial

Private Blockchain

Enterprises favour a more controlled blockchain

An enterprise writes transactions

Public vs Private Blockchain

Private Blockchain

Private blockchains have been controversial

Enterprises favour a more controlled blockchain

An enterprise writes transactions

Doesn't have same decentralized security as public blockchain

Public vs Private Blockchain

Private Blockchain

Private blockchains have been controversial

Enterprises favour a more controlled blockchain

An enterprise writes transactions

Doesn't have same decentralized security as public blockchain

The company decides who can read and verify blocks

ALRIGHT HIPPIES



Miners need to be paid

Your transactions will incur a
transaction fee

You need to consider costs



Now it's
your turn!

Discuss some different use
cases for a Blockchain?

Blockchain Use Cases

Digital Currency

Blockchain Use Cases

Digital Currency

Voting

Blockchain Use Cases

Digital Currency

Voting

Intellectual Property

Blockchain Use Cases

Digital Currency

Voting

Intellectual Property

AML and KYC

Blockchain Use Cases

Digital Currency

Voting

Intellectual Property

AML and KYC

Identity Management

Blockchain Use Cases

Digital Currency

Voting

Intellectual Property

AML and KYC

Identity Management

Public Record
Registries



The General Data Protection Regulation

Data to Record

Claim number

Settlement amount

Settlement date

Car registration

Mileage

Claim type

Blocks with a Single Transaction

Transaction Details

Claim Number

Settlement Amount

Settlement Date

Registration Plate

Mileage

Claim Type

Transaction Details

Claim Number

Settlement Amount

Settlement Date

Registration Plate

Mileage

Claim Type

Block Header

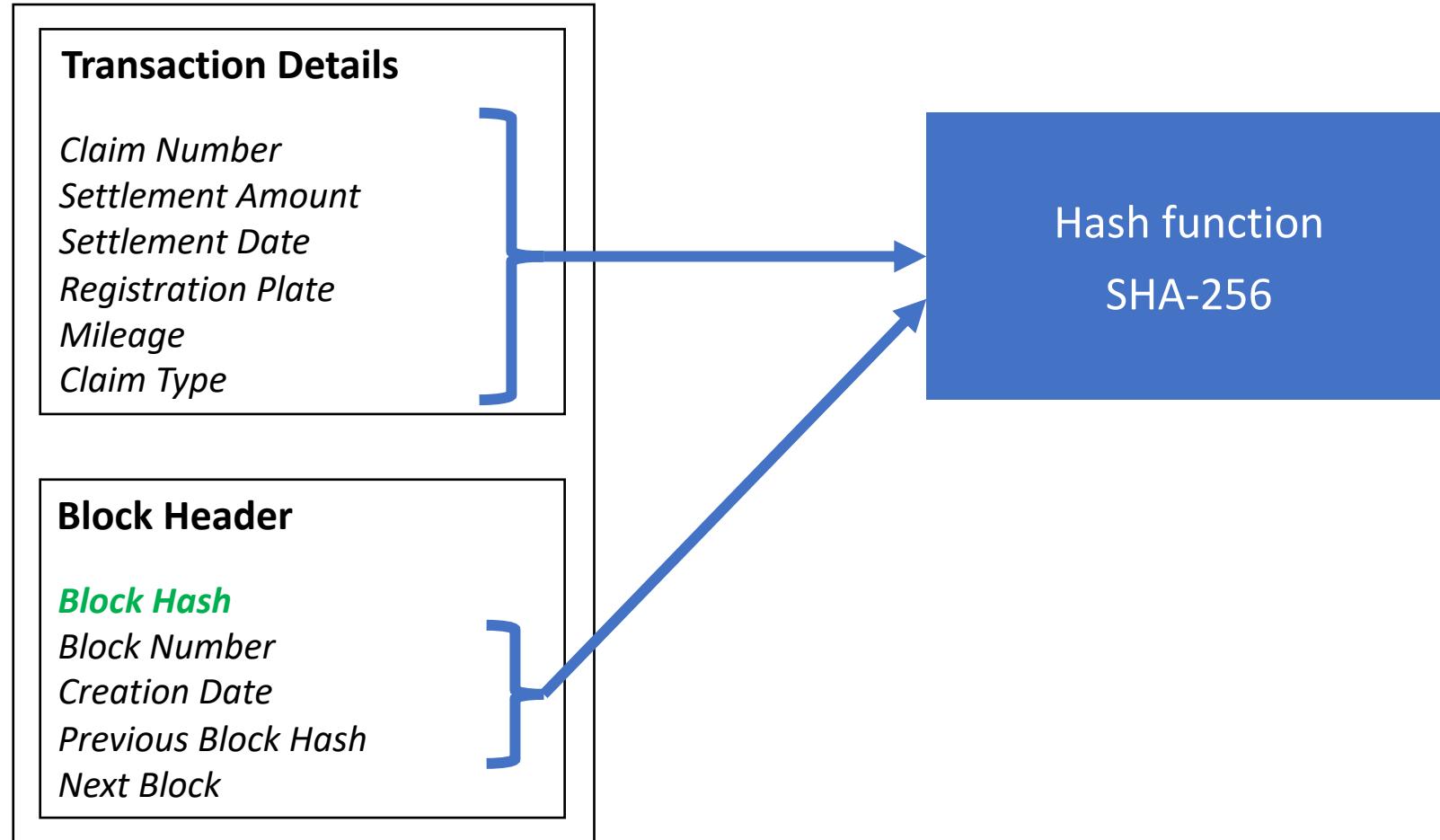
Block Hash

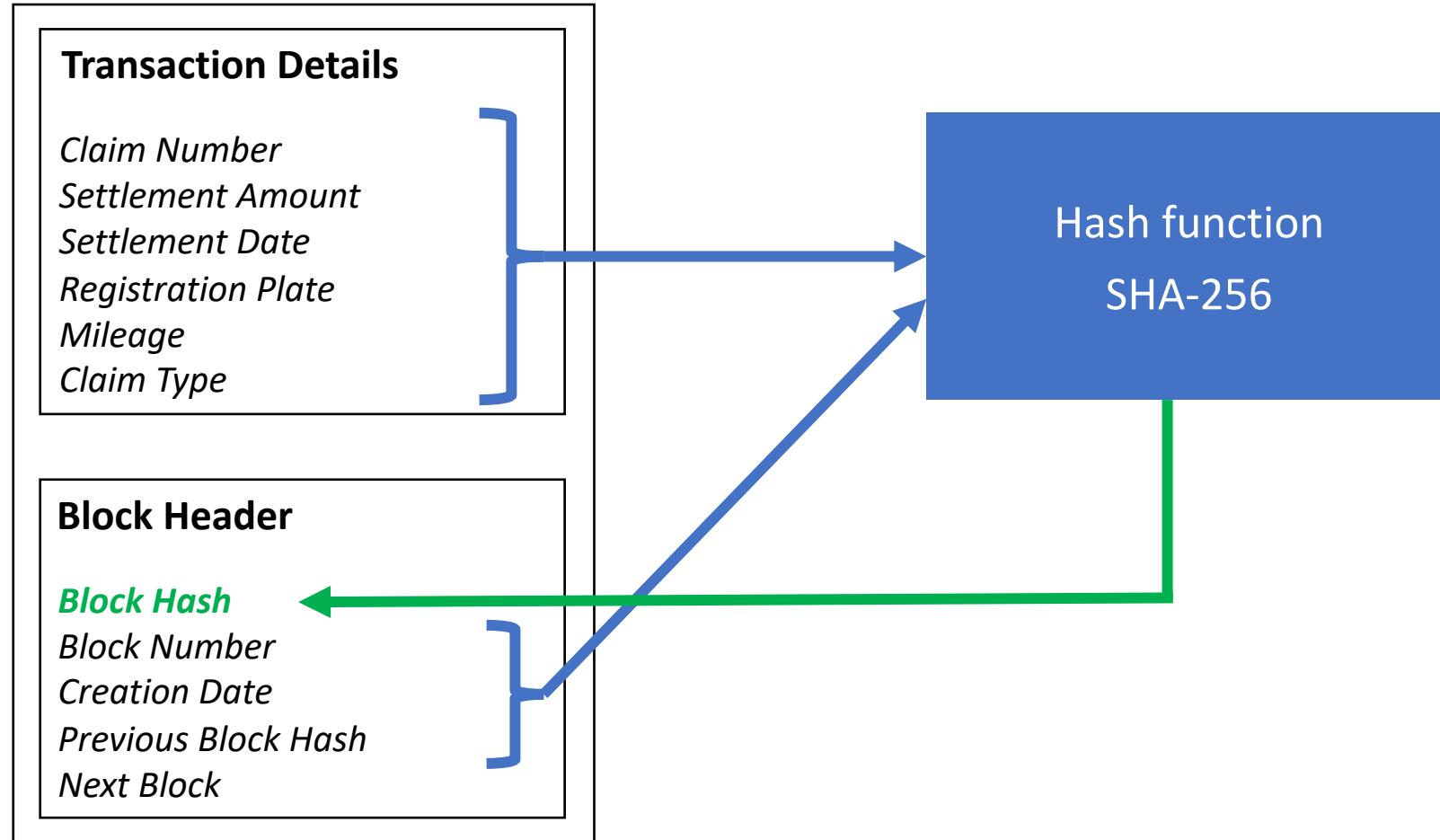
Block Number

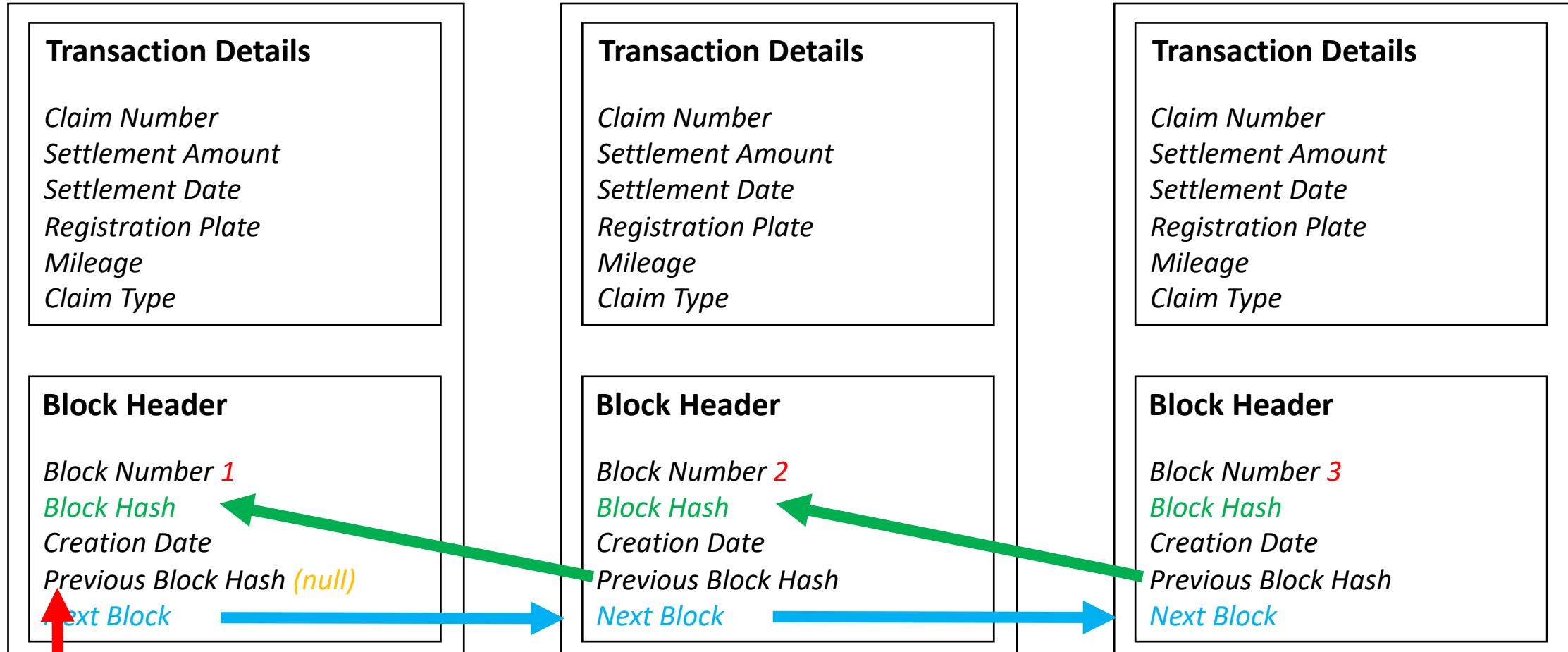
Creation Date

Previous Block Hash

Next Block

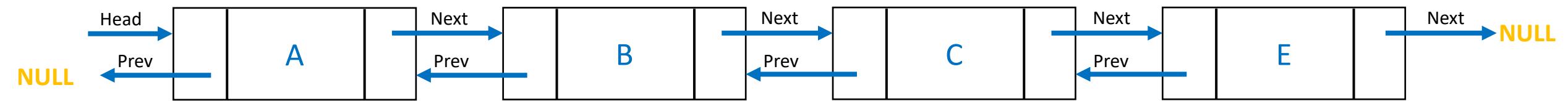




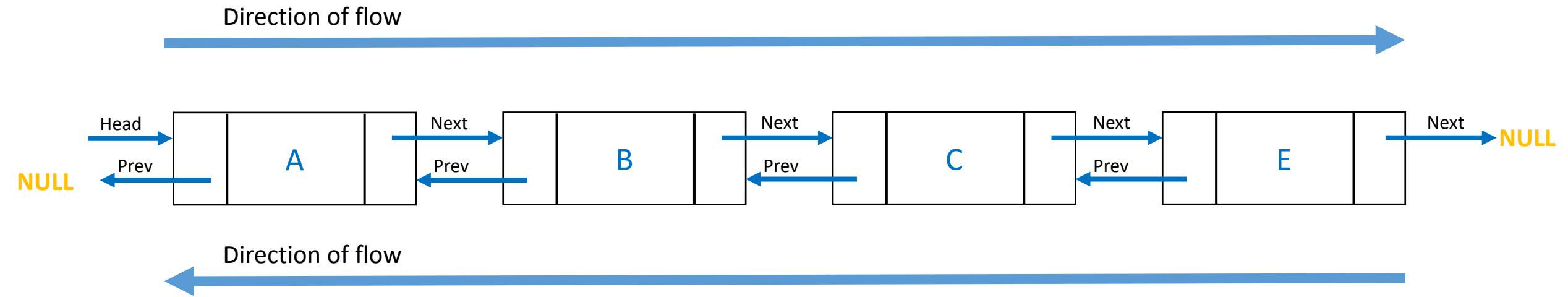


Has no parent so equals null

Double Linked List

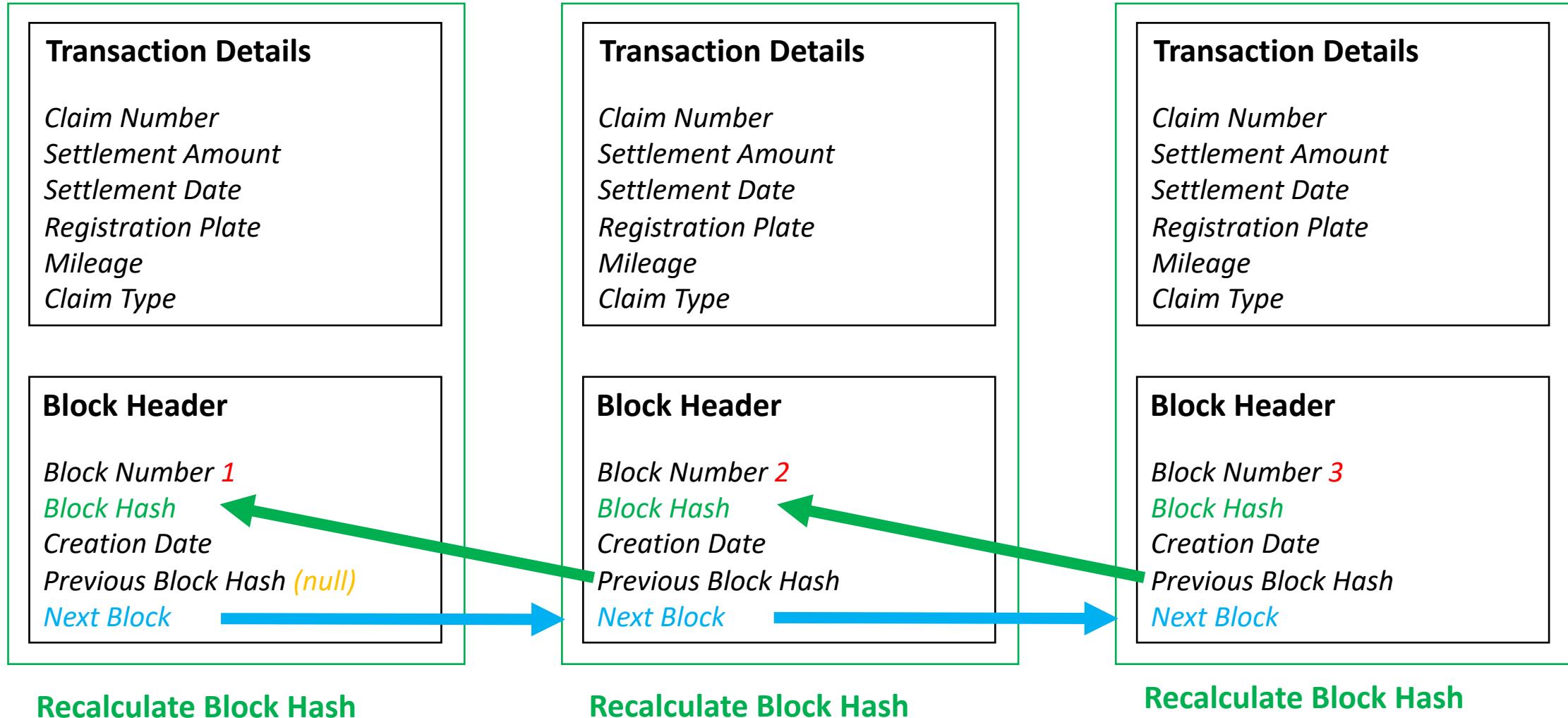


Double Linked List

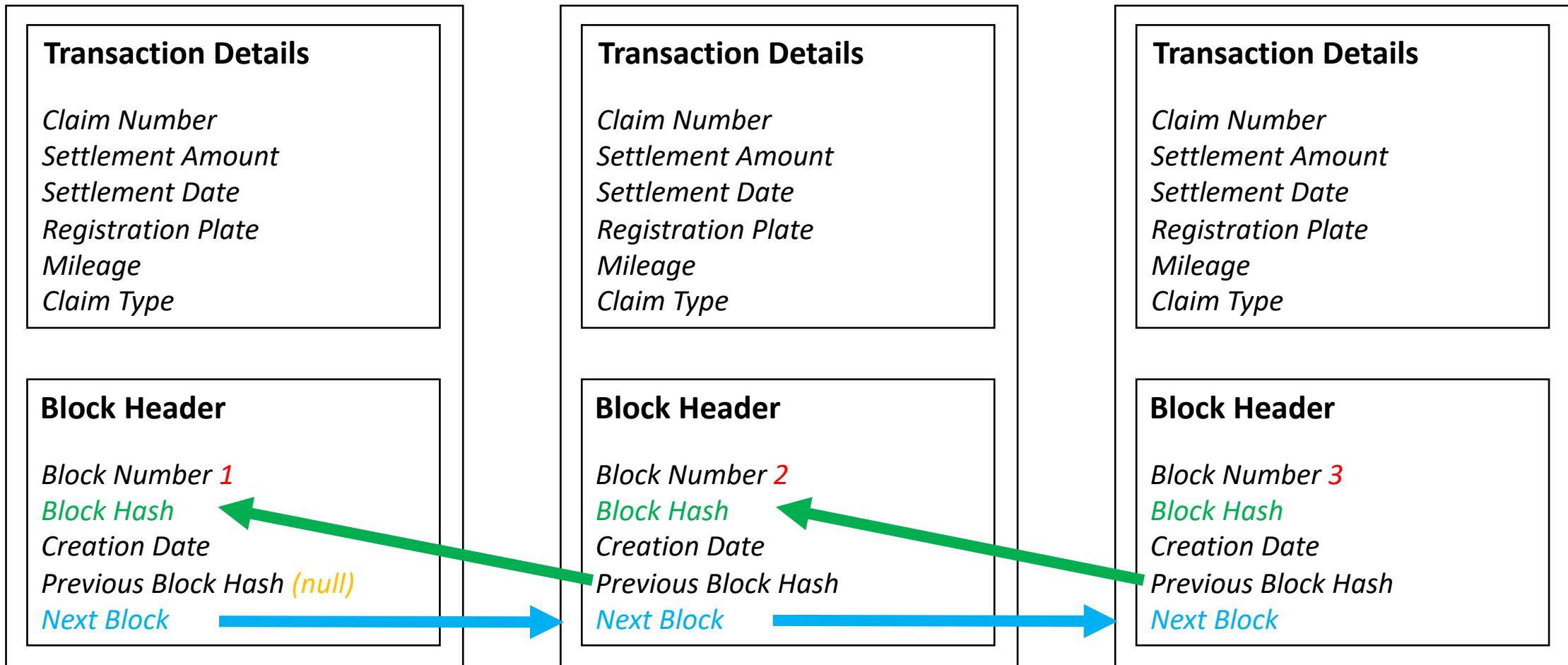


Blockchain flows in one direction

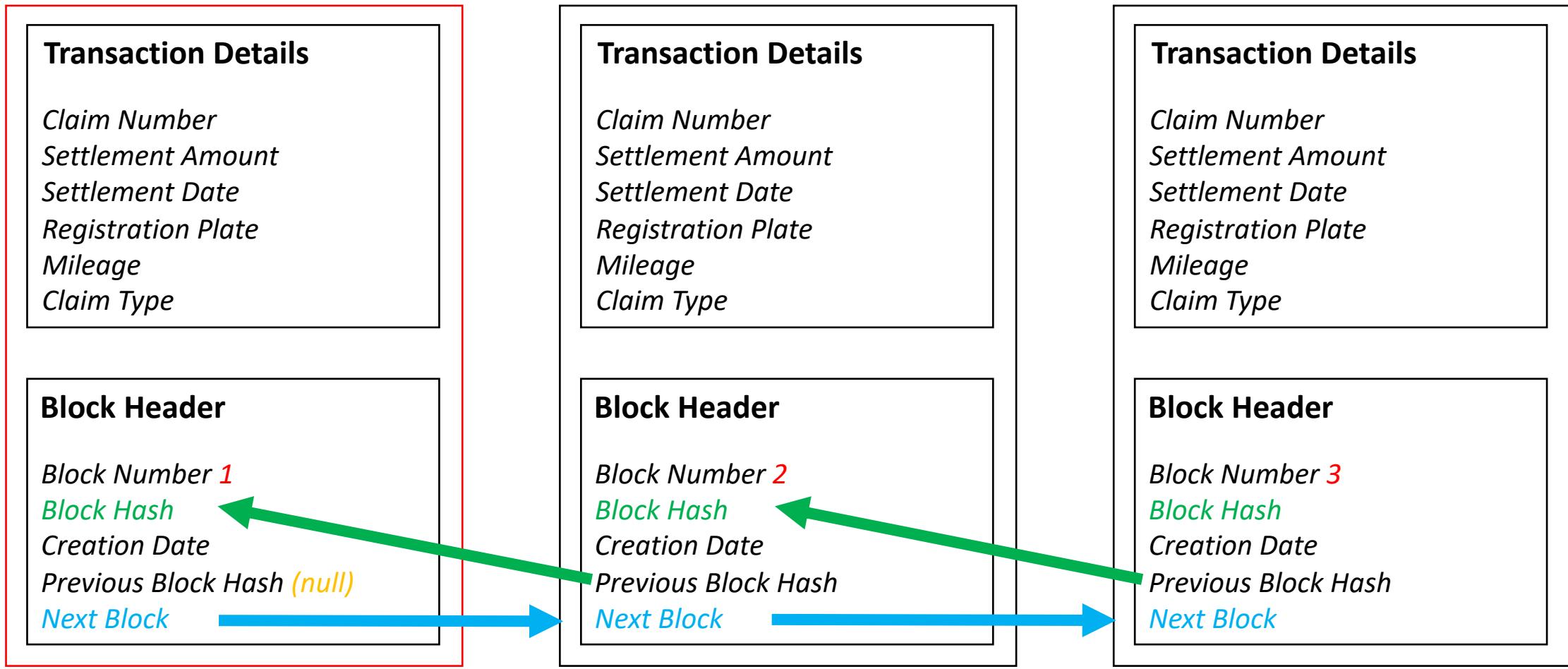
Happy Path



Verification Failure

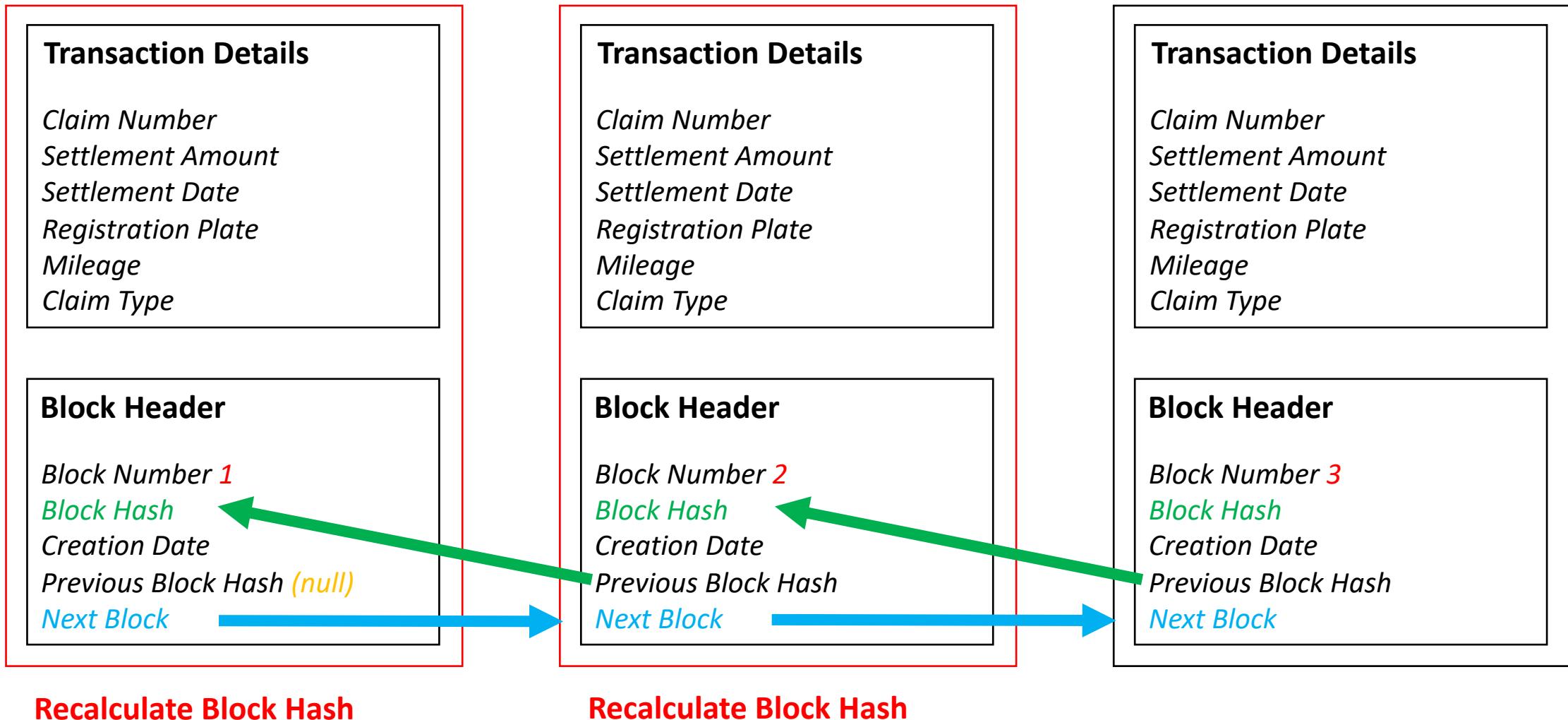


Verification Failure

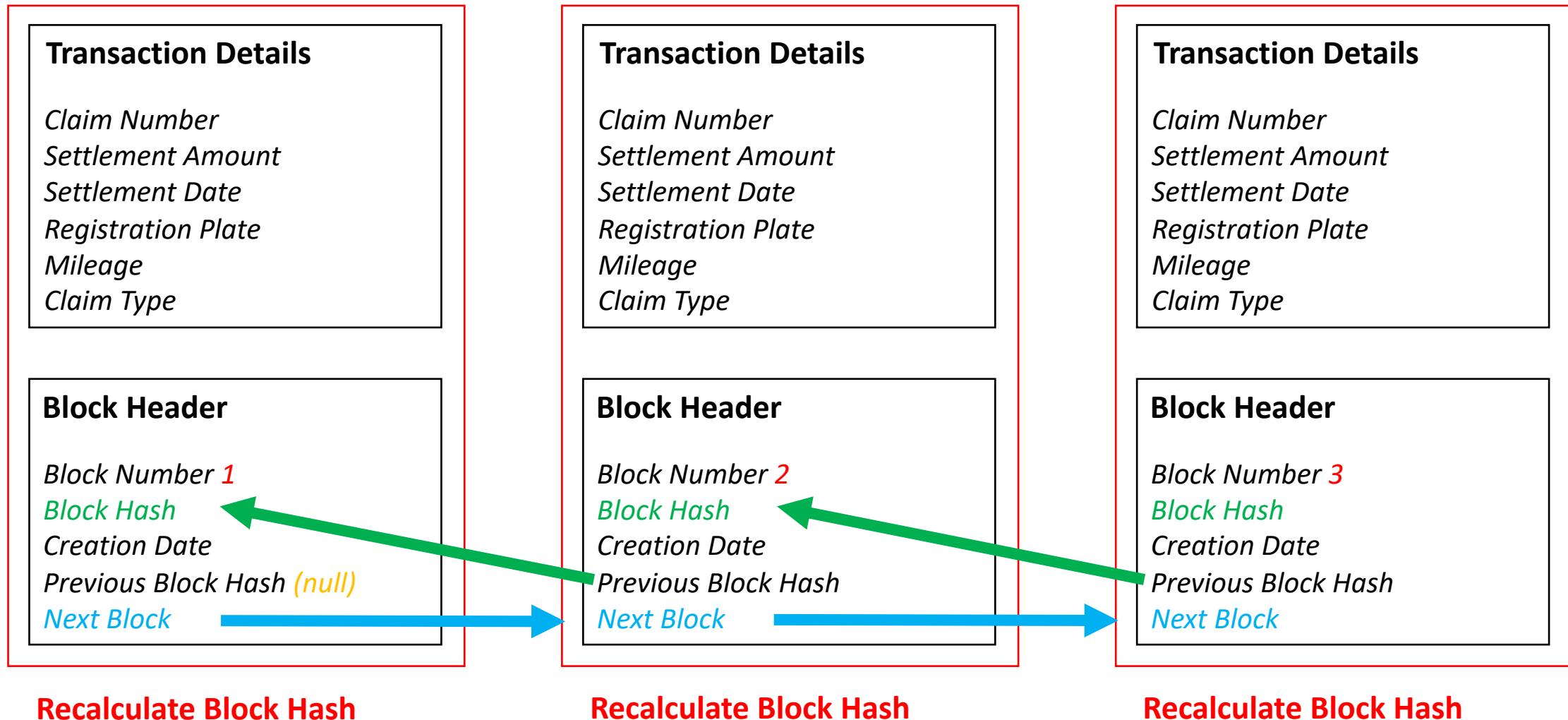


Recalculate Block Hash

Verification Failure



Verification Failure

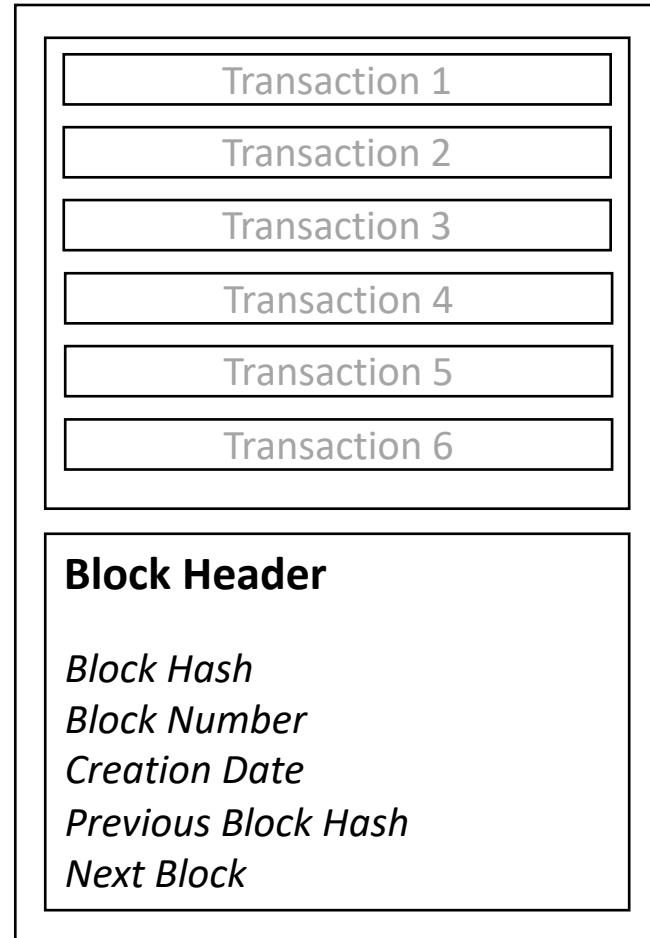


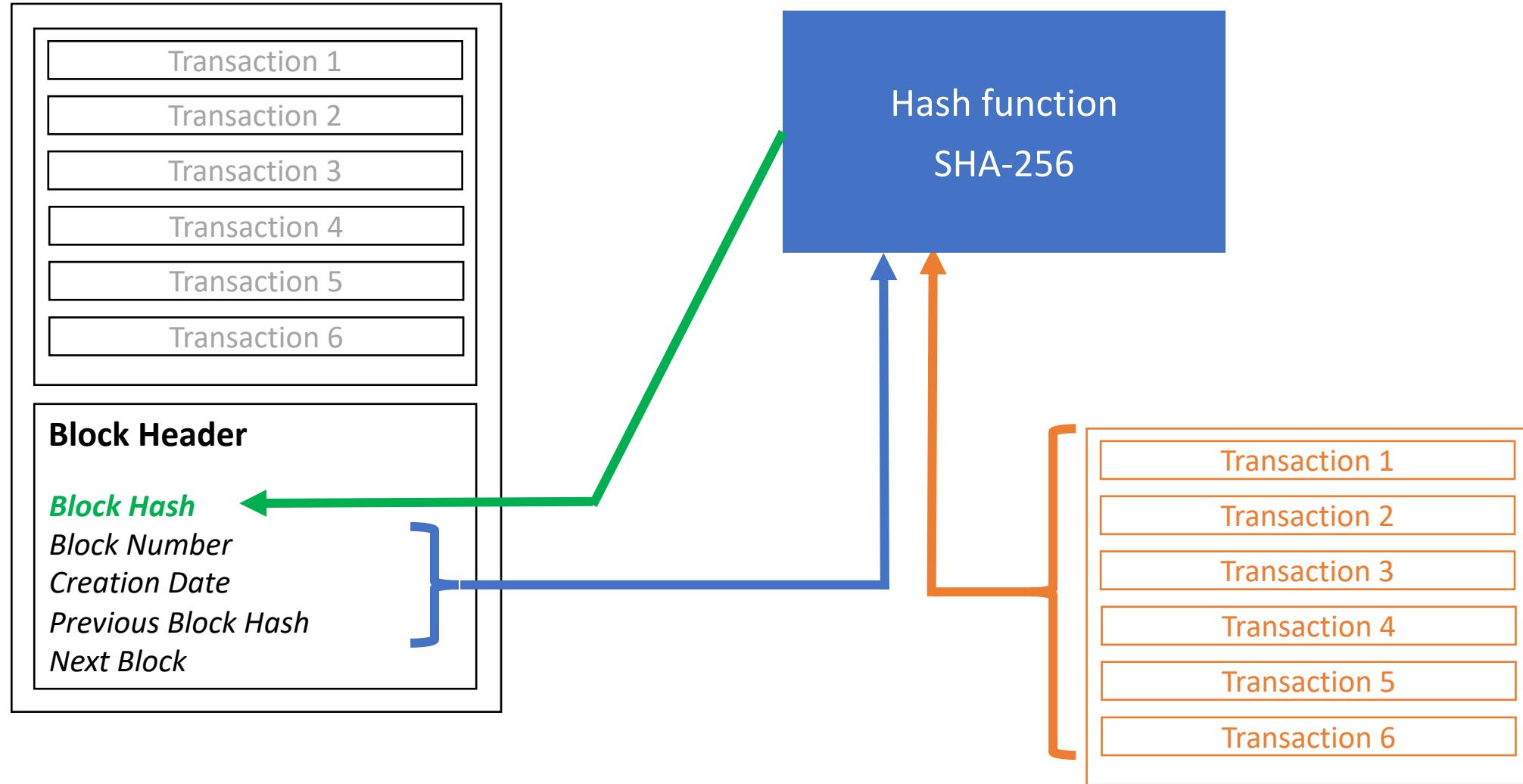


Now it's
your turn!

Spend some time stepping through the
Block with Single Transaction example

Blocks with Multiple Transactions



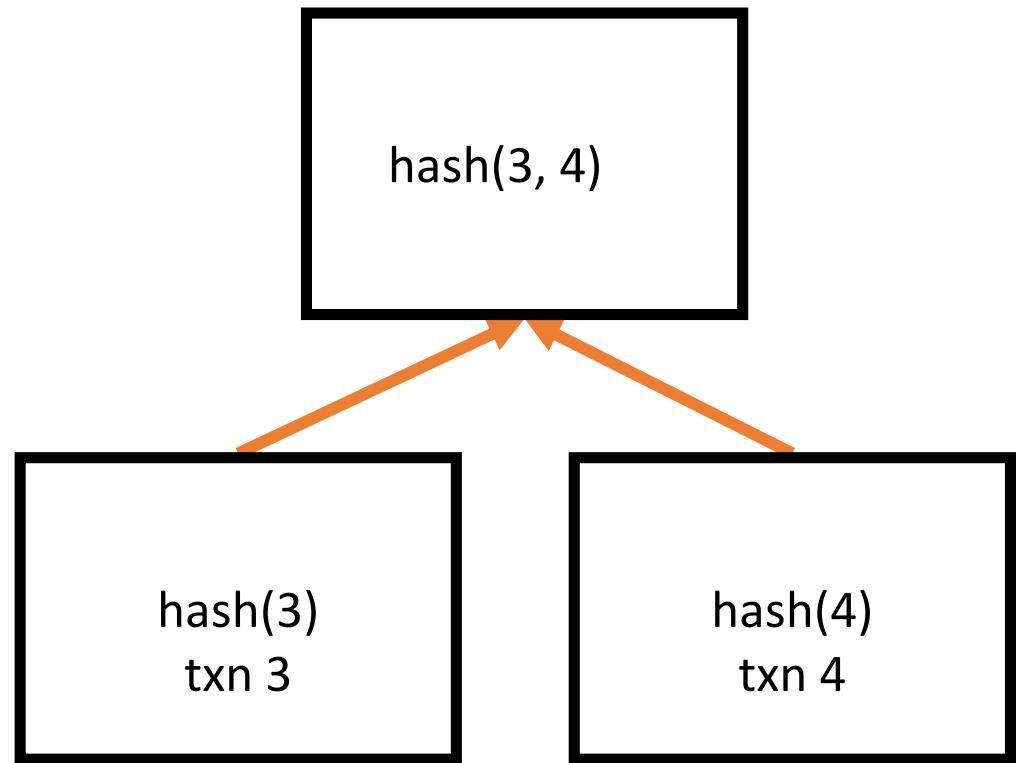
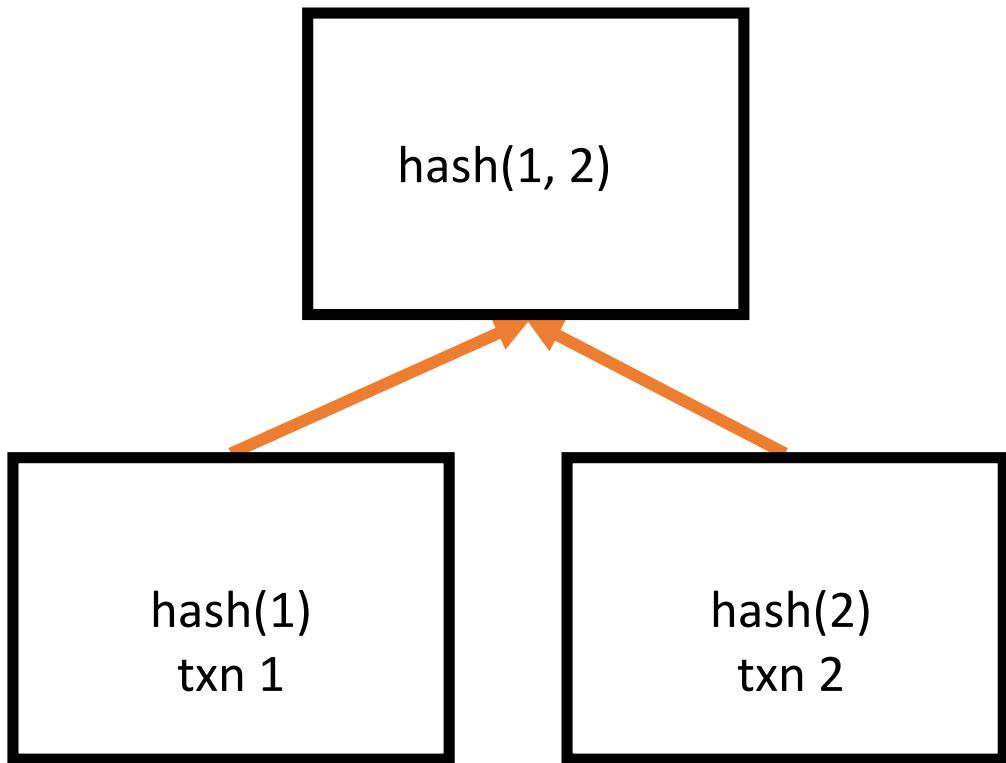


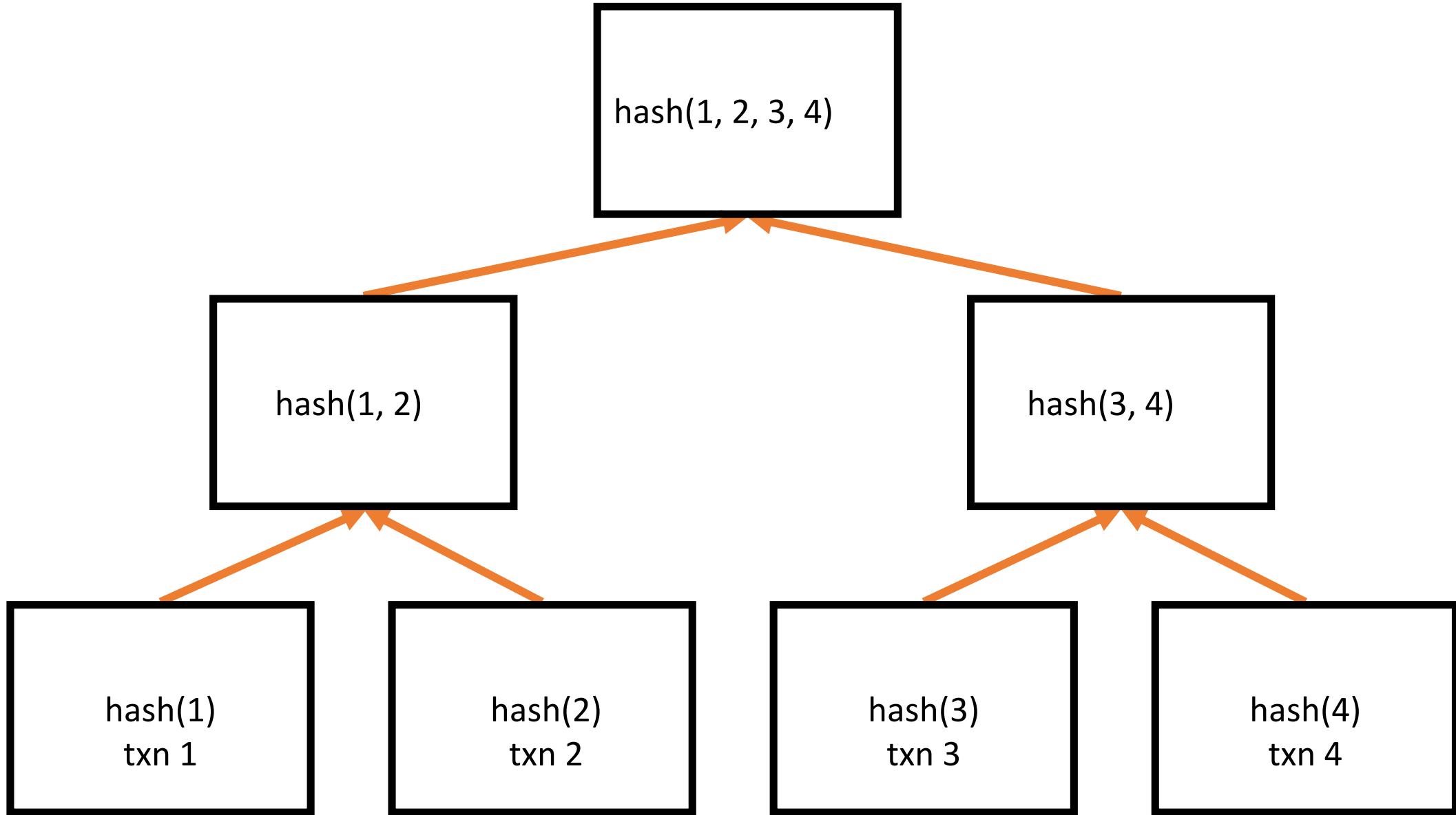
hash(1)
txn 1

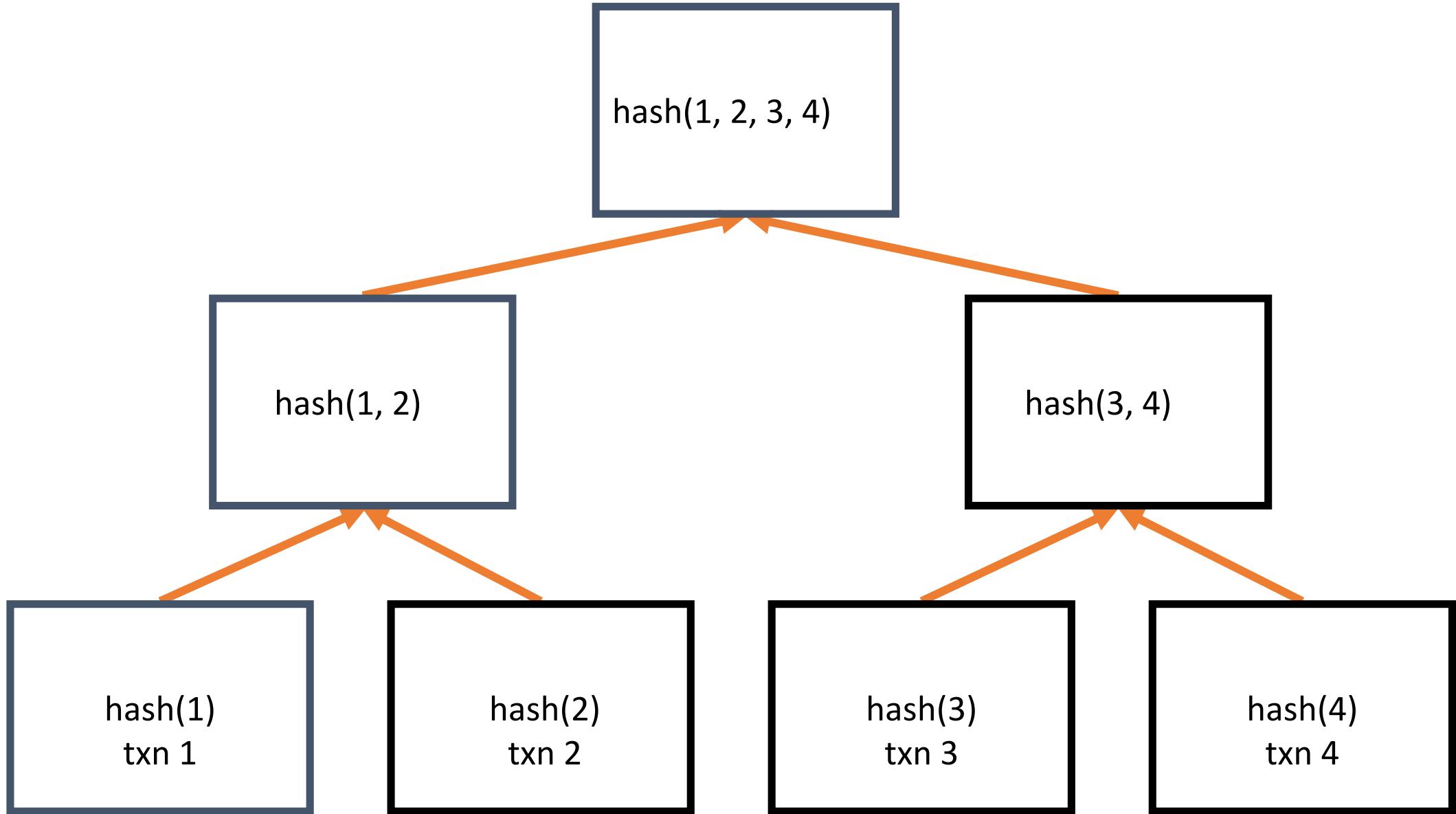
hash(2)
txn 2

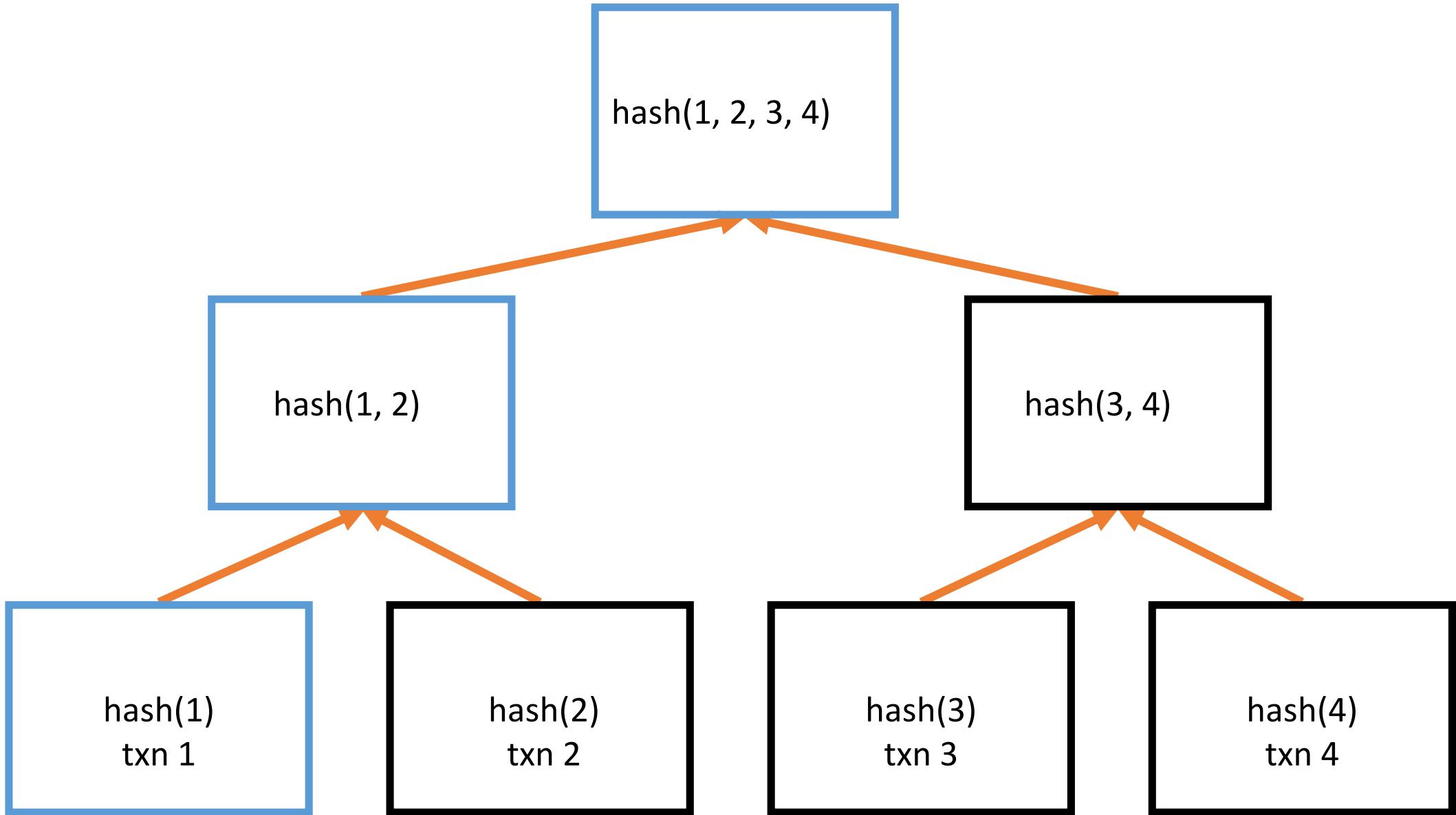
hash(3)
txn 3

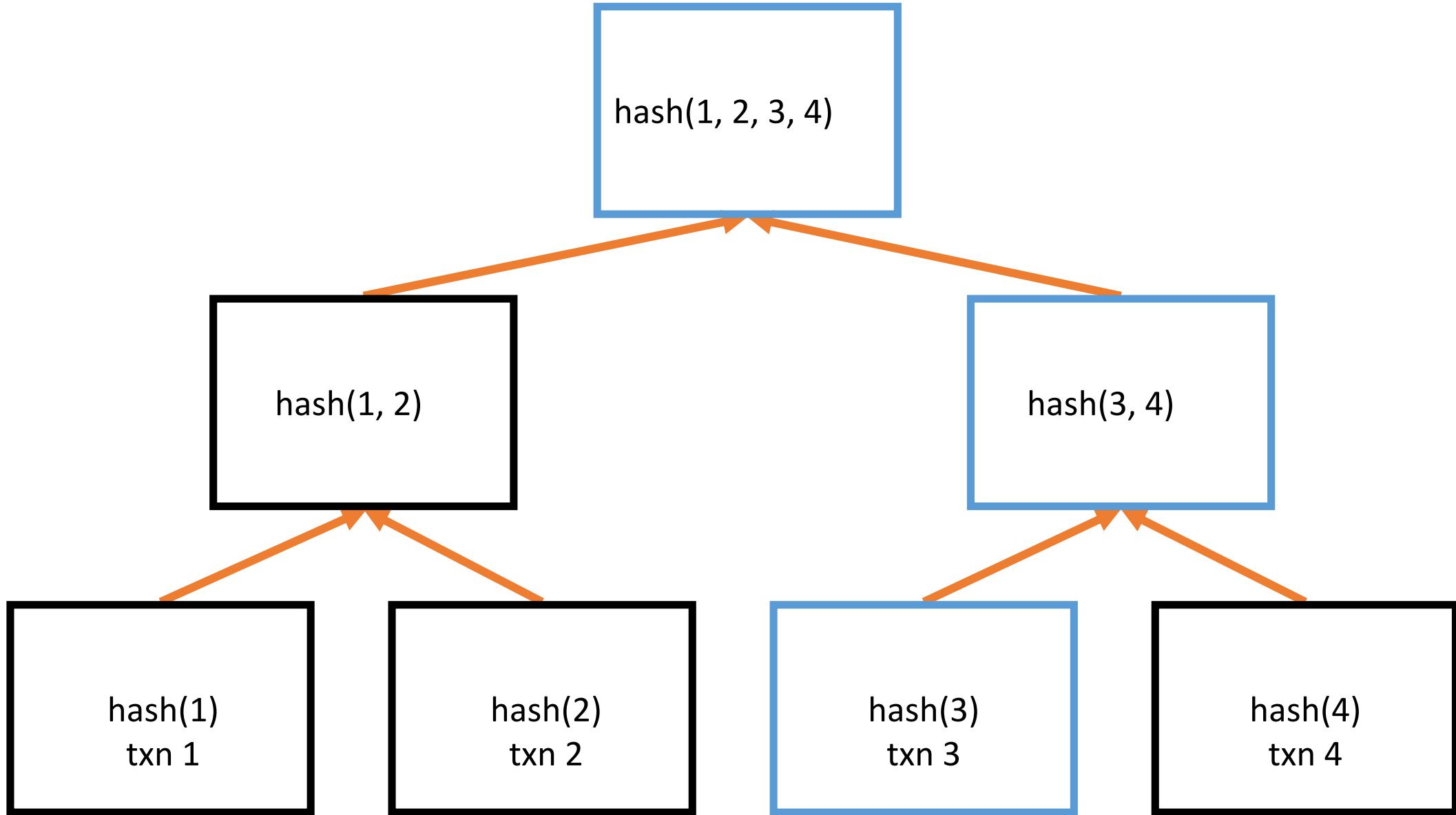
hash(4)
txn 4

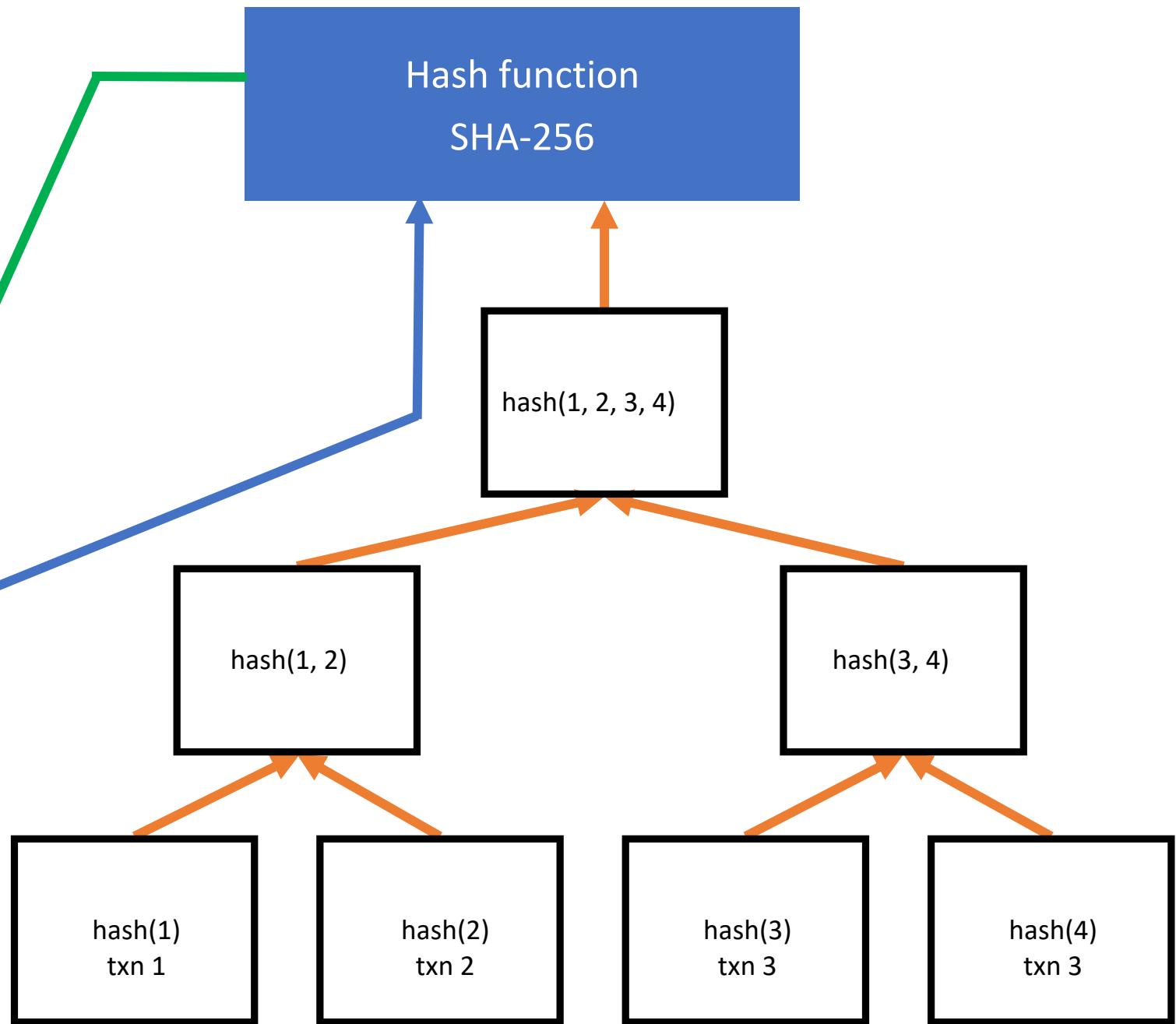
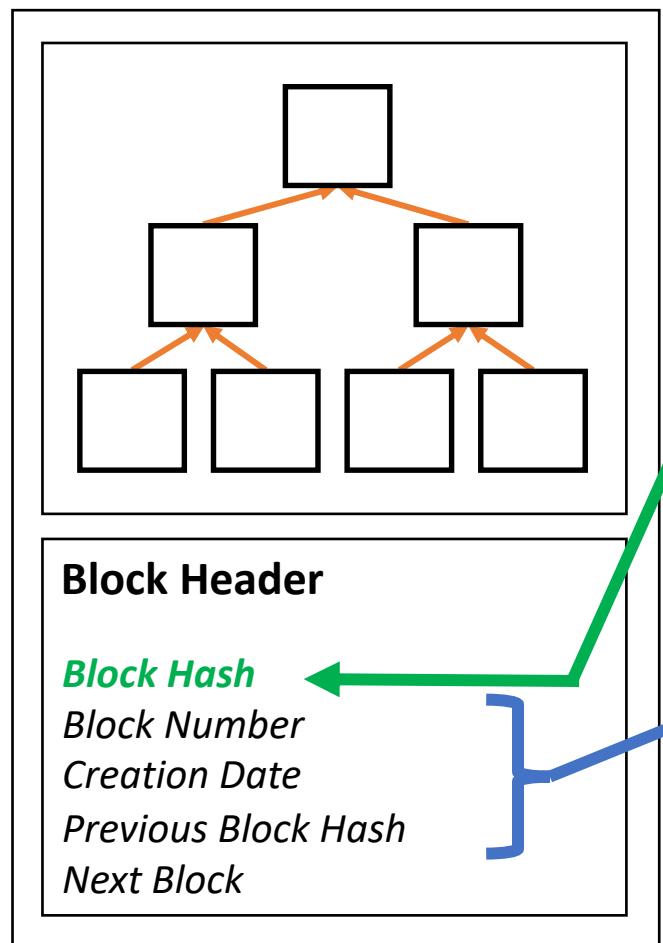














Now it's
your turn!

Spend some time stepping through the
Block with Multiple Transactions example

Applying Proof of Work

Block #514788

Summary	
Number Of Transactions	2328
Output Total	\$ 62,170,297.79
Estimated Transaction Volume	\$ 21,401,766.29
Transaction Fees	\$ 1,614.42
Height	514788 (Main Chain)
Timestamp	2018-03-23 11:49:43
Received Time	2018-03-23 11:49:43
Relayed By	Unknown
Difficulty	3,462,542,391,191.56
Bits	391203401
Size	1101.468 kB
Weight	3993.123 kWU
Version	0x20000000
Nonce	3627070036
Block Reward	\$ 106,077.12

Hashes	
Hash	0000000000000000000000000000000010cd0b6538c7c2e72752e9f7f88ff495086ab067f1ebb3
Previous Block	000000000000000000000000000000001836119518ed7da954694dbc2f29dae66788ee873b2355
Next Block(s)	
Merkle Root	43cdb974d8ed687925b246e8090b0edf2811bbb71e7a817abc8f43ad8f49d1eb

Hashes

Hash 0000000000000000000010cd0b6538c7c2e72752e9f7f88ff495086ab067f1ebb3

Previous Block 000000000000000000001836119518ed7da954694dbc2f29dae66788ee873b2355

Next Block(s)

Merkle Root 43cdb974d8ed687925b246e8090b0edf2811bbb71e7a817abc8f43ad8f49d1eb

Solving the Problem with Hashing

Hash("Attack at 6am Tuesday 467387465 " = 000000283747282918723647

Solving the Problem with Hashing

Hash("Attack at 6am Tuesday 0" = 38472395749325f79385s7394f573495b

Solving the Problem with Hashing

Hash("Attack at 6am Tuesday 1" = 0967094585902409b09d900980c0980a

Solving the Problem with Hashing

Hash("Attack at 6am Tuesday **467387465**" = **000000**283747282918723647

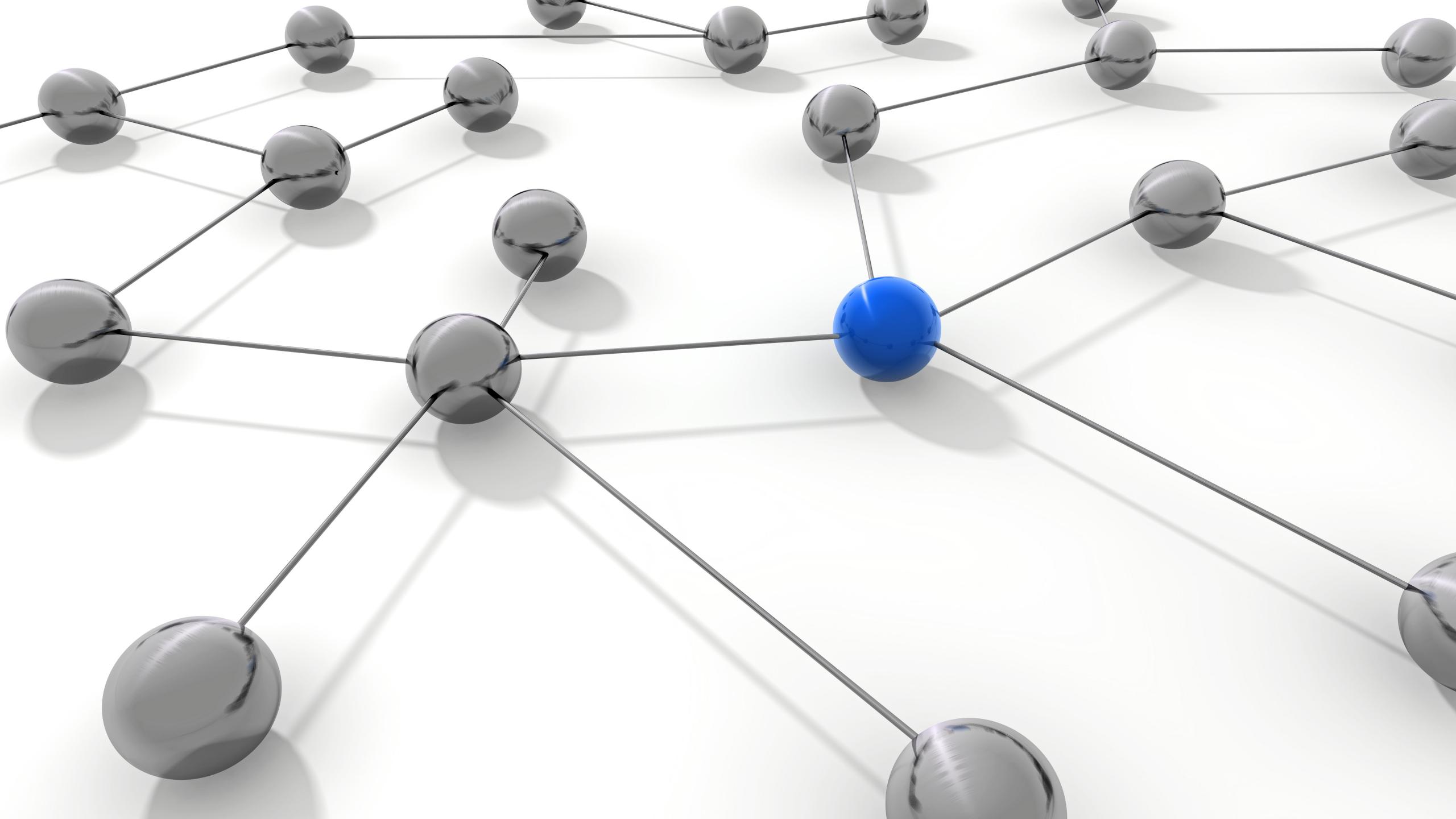
Solving the Problem with Hashing

Expensive to calculate

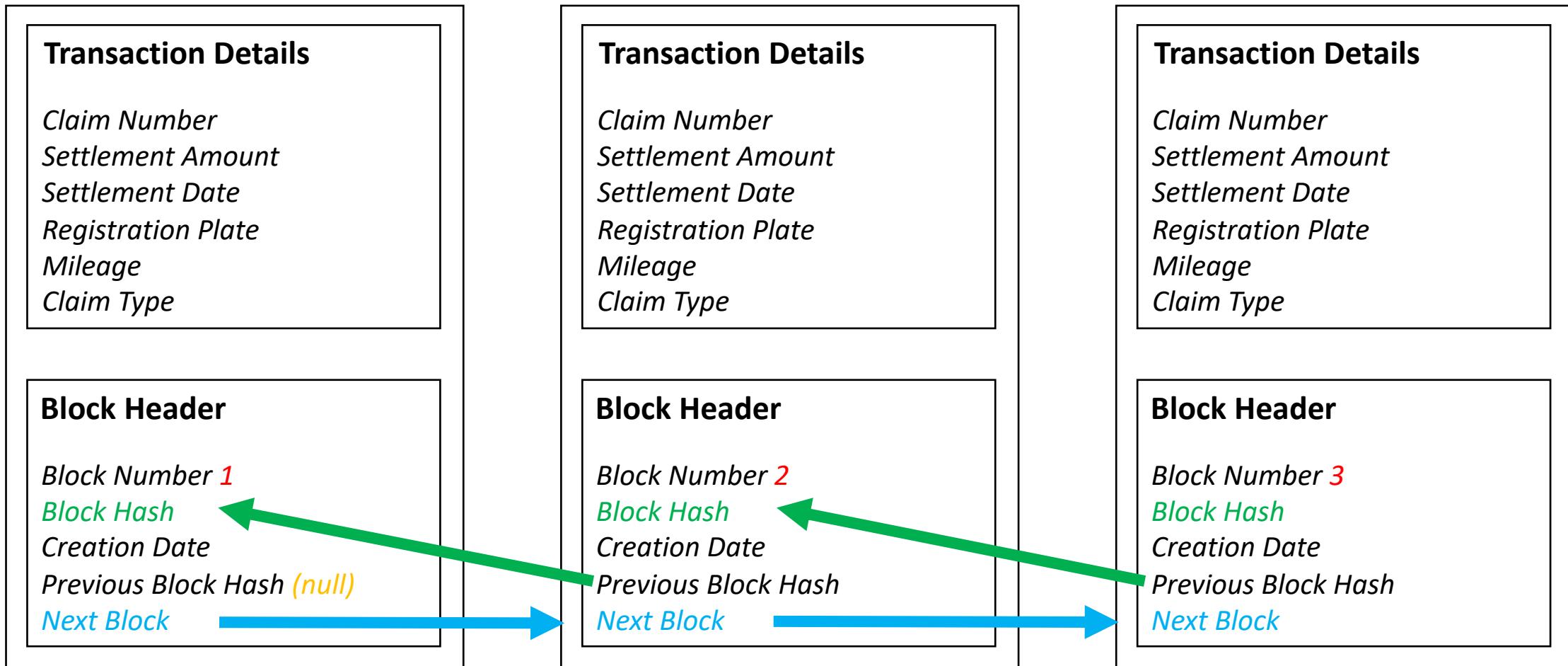
Easy to verify

Solving the Problem with Hashing

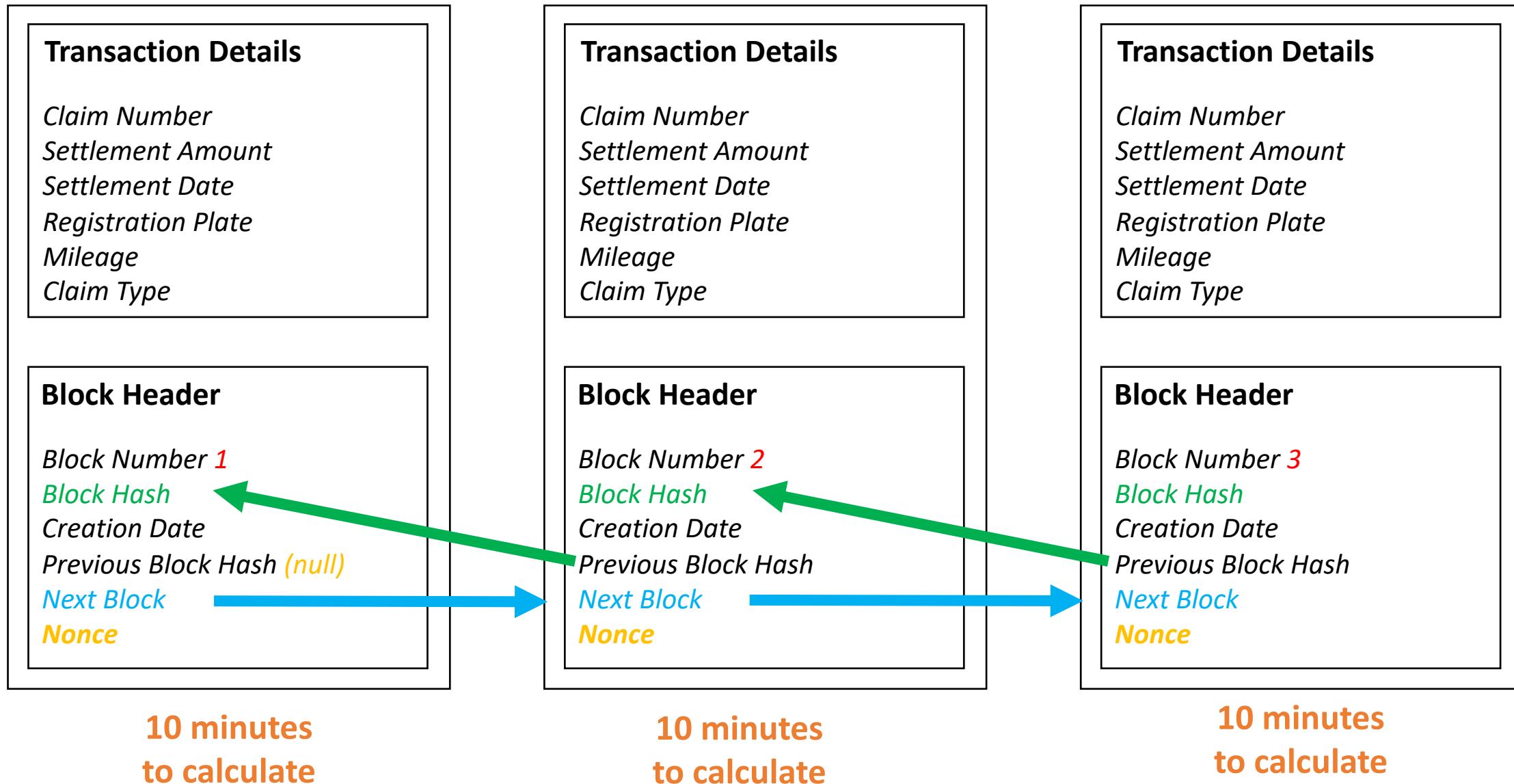
Changing the message
Requires recalculation
of the hash and nonce



Preventing Block Tampering



Preventing Block Tampering



Preventing Block Tampering

5000 blocks at 10 minutes per block

Preventing Block Tampering

5000 blocks at 10 minutes per block

833 hours effort

Preventing Block Tampering

5000 blocks at 10 minutes per block

833 hours effort

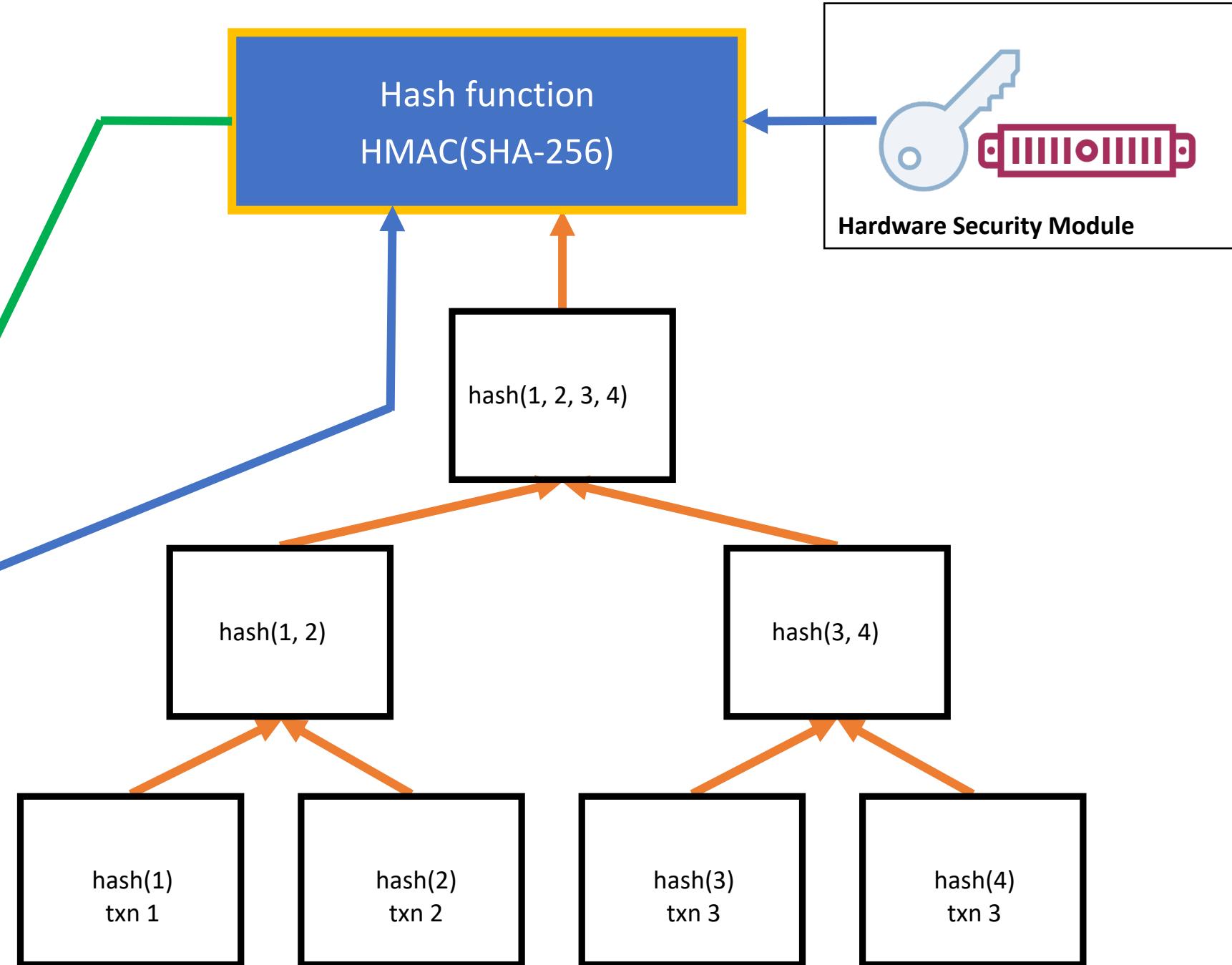
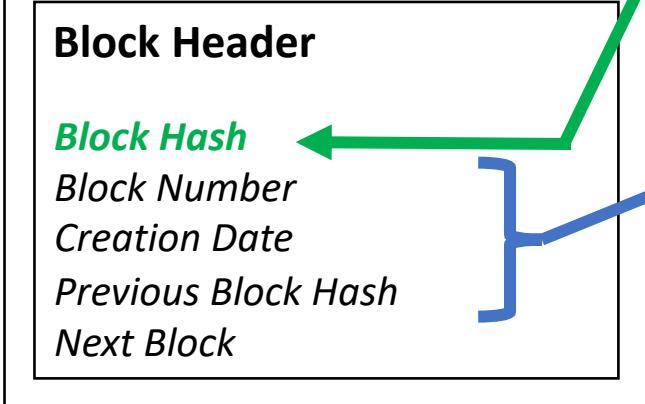
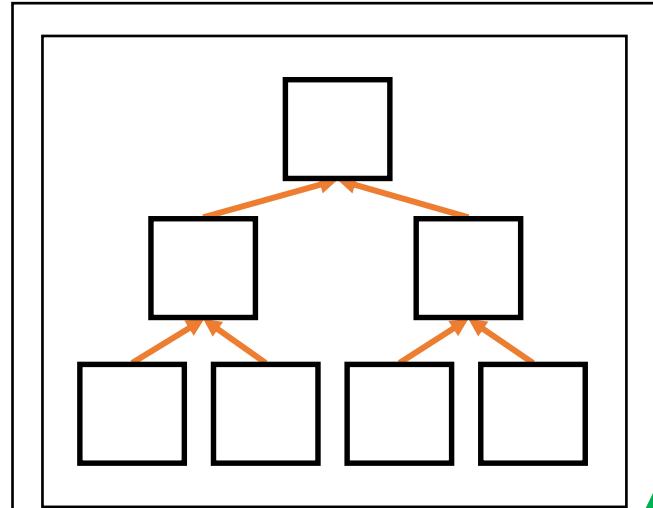
35 days

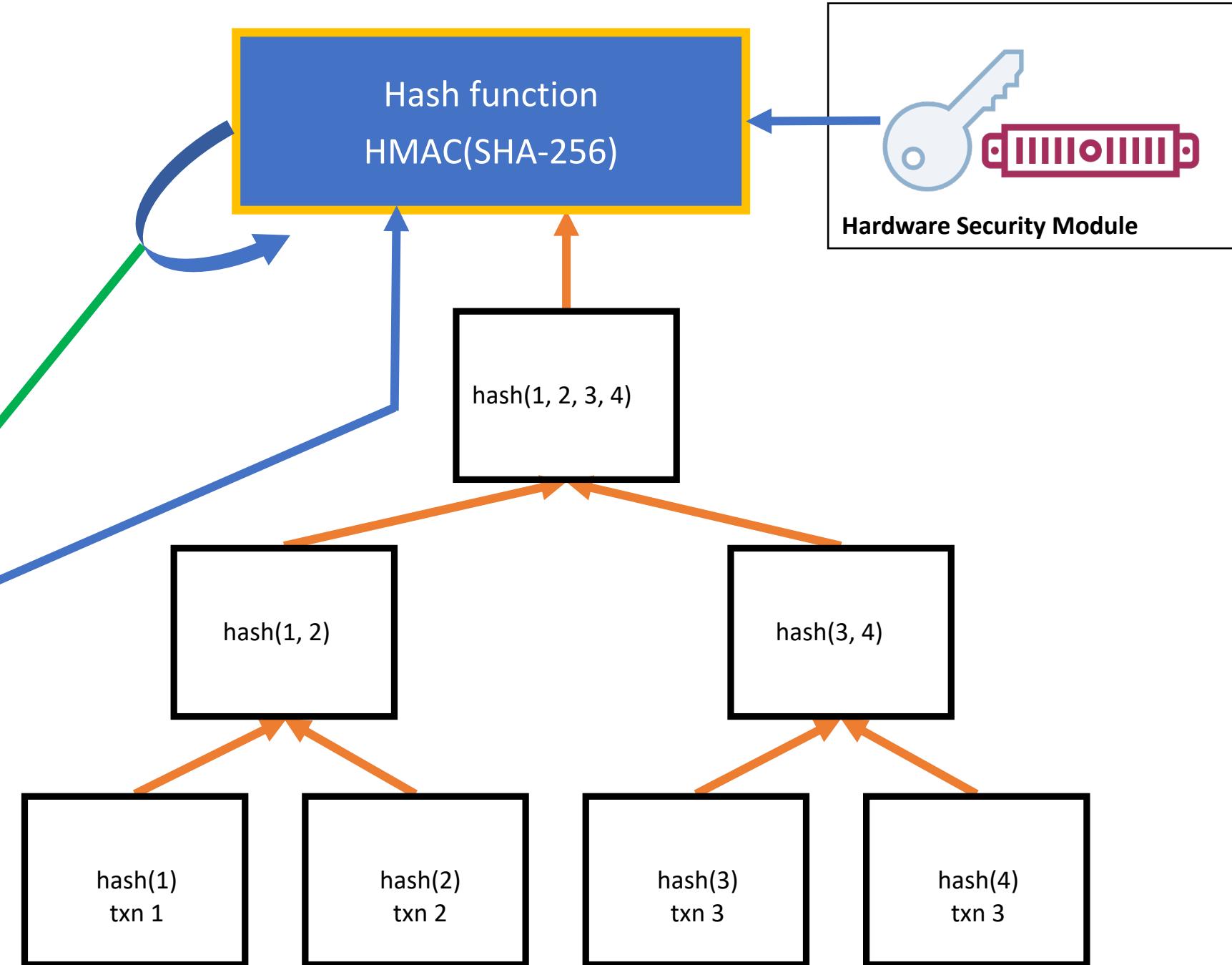
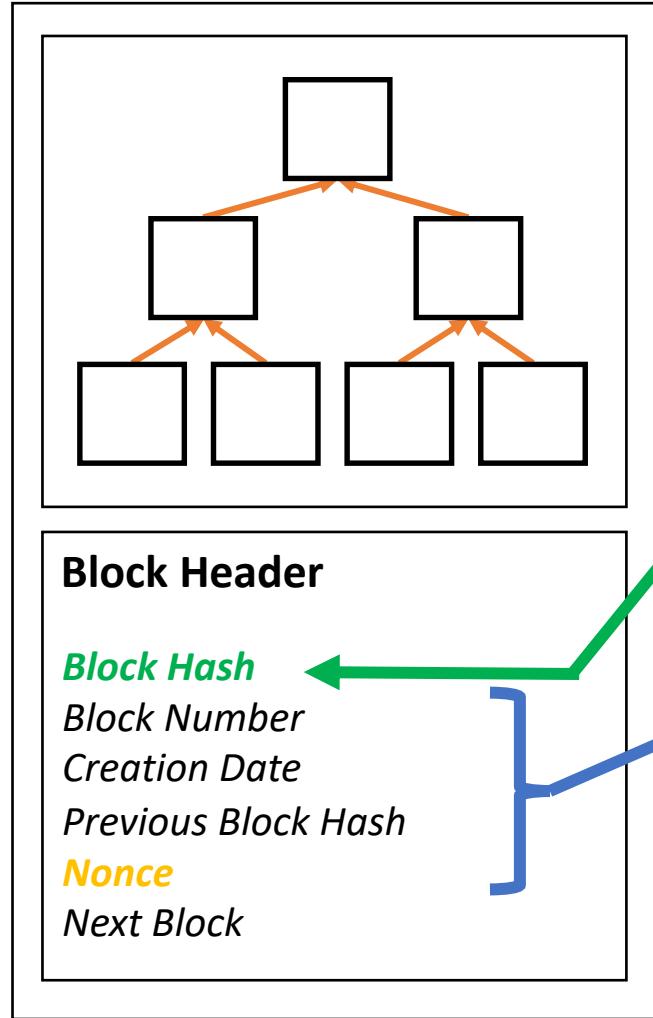


Now it's
your turn!

Spend some time stepping through the
Proof of Work example

Integrating Proof of Work





stephenhaunts — ProofOfWorkTest.dll — dotnet + bash -c clear; cd "/Applications/Visual Studio.app/Contents/Resources/li..."

```
Difficulty Level 0 - Nonce = 0 - Elapsed = 00:00:00.03 - EYEd2NqKf57Cna1I2rk9UCprZTCrhiT3QcRhS1BQJhA=
Difficulty Level 1 - Nonce = 24 - Elapsed = 00:00:00.00 - 0NboeG5IAZduTUQ2WmALzrfN1FoRDg5nLZN8Vhgmgmg=
Difficulty Level 2 - Nonce = 9478 - Elapsed = 00:00:00.01 - 00Ln0xLQjZuZjuZrw14ne2T3R4niaydfrazohLoLx1k=
Difficulty Level 3 - Nonce = 93521 - Elapsed = 00:00:00.12 - 000FtlmdNXKvBYFiAeKGPaLMBIkECs9nIbskXiln4og=
Difficulty Level 4 - Nonce = 2286428 - Elapsed = 00:00:03.10 - 00007D4cJR6eYzCvW82UPRAC/dkxEsV55v7+9bVRH/A=
Difficulty Level 5 - Nonce = 380372972 - Elapsed = 00:08:25.45 - 0000032c57uM3MYWkrnzHNv+efN7SJULrrnzRgDrXXQ=
```

stephenhaunts — ProofOfWorkTest.dll — dotnet + bash -c clear; cd "/Applications/Visual Studio.app/Contents/Resources/li..."

Difficulty Level 0 - Nonce = 0 - Elapsed = 00:00:00.03 - EYEd2NqKf57Cna1I2rk9UCprZTCrhiT3QcRhS1BQJhA=
Difficulty Level 1 - Nonce = 24 - Elapsed = 00:00:00.00 - 0NboeG5IAZduTUQ2WmALzrfN1FoRDg5nLZN8Vhgmgmg=
Difficulty Level 2 - Nonce = 9478 - Elapsed = 00:00:00.01 - 00Ln0xLQjZuZjuZrw14ne2T3R4niaydfrazohLoLx1k=
Difficulty Level 3 - Nonce = 93521 - Elapsed = 00:00:00.12 - 000FtlmdNXKvBYFiAeKGPaLMBIkECs9nIbskXiln4og=
Difficulty Level 4 - Nonce = 2286428 - Elapsed = 00:00:03.10 - 00007D4cJR6eYzCvW82UPRAC/dkxEsV55v7+9bVRH/A=
Difficulty Level 5 - Nonce = 380372972 - Elapsed = 00:08:25.45 - 0000032c57uM3MYWkrnzHNv+efN7SJULrrnzRgDrXXQ=

stephenhaunts — ProofOfWorkTest.dll — dotnet + bash -c clear; cd "/Applications/Visual Studio.app/Contents/Resources/li..."

Difficulty Level 0 - Nonce = 0 - Elapsed = 00:00:00.03 - EYEd2NqKf57Cna1I2rk9UCprZTCrhiT3QcRhS1BQJhA=
Difficulty Level 1 - Nonce = 24 - Elapsed = 00:00:00.00 - 0NboeG5IAZduTUQ2WmALzrfN1FoRDg5nLZN8Vhgmgmg=
Difficulty Level 2 - Nonce = 9478 - Elapsed = 00:00:00.01 - 00Ln0xLQjZuZjuZrw14ne2T3R4niaydfrazohLoLx1k=
Difficulty Level 3 - Nonce = 93521 - Elapsed = 00:00:00.12 - 000FtlmdNXKvBYFiAeKGPaLMBIkECs9nIbskXiln4og=
Difficulty Level 4 - Nonce = 2286428 - Elapsed = 00:00:03.10 - 00007D4cJR6eYzCvW82UPRAC/dkxEsV55v7+9bVRH/A=
Difficulty Level 5 - Nonce = 380372972 - Elapsed = 00:08:25.45 - 0000032c57uM3MYWkrnzHNv+efN7SJULrrnzRgDrXXQ=

stephenhaunts — ProofOfWorkTest.dll — dotnet + bash -c clear; cd "/Applications/Visual Studio.app/Contents/Resources/li..."

Difficulty Level 0 - Nonce = 0 - Elapsed = 00:00:00.03 - EYEd2NqKf57Cna1I2rk9UCprZTCrhiT3QcRhS1BQJhA=
Difficulty Level 1 - Nonce = 24 - Elapsed = 00:00:00.00 - 0NboeG5IAZduTUQ2WmALzrfN1FoRDg5nLZN8Vhgmgmg=
Difficulty Level 2 - Nonce = 9478 - Elapsed = 00:00:00.01 - 00Ln0xLQjZuZjuZrw14ne2T3R4niaydfrazohLoLx1k=
Difficulty Level 3 - Nonce = 93521 - Elapsed = 00:00:00.12 - 000FtlmdNXKvBYFiAeKGPaLMBIkECs9nIbskXiln4og=
Difficulty Level 4 - Nonce = 2286428 - Elapsed = 00:00:03.10 - 00007D4cJR6eYzCvW82UPRAC/dkxEsV55v7+9bVRH/A=
Difficulty Level 5 - Nonce = 380372972 - Elapsed = 00:08:25.45 - 0000032c57uM3MYWkrnzHNv+efN7SJULrrnzRgDrXXQ=

stephenhaunts — ProofOfWorkTest.dll — dotnet + bash -c clear; cd "/Applications/Visual Studio.app/Contents/Resources/li..."

```
Difficulty Level 0 - Nonce = 0 - Elapsed = 00:00:00.03 - EYEd2NqKf57Cna1I2rk9UCprZTCrhiT3QcRhS1BQJhA=
Difficulty Level 1 - Nonce = 24 - Elapsed = 00:00:00.00 - 0NboeG5IAZduTUQ2WmALzrfN1FoRDg5nLZN8Vhgmgmg=
Difficulty Level 2 - Nonce = 9478 - Elapsed = 00:00:00.01 - 00Ln0xLQjZuZjuZrw14ne2T3R4niaydfrazohLoLx1k=
 Difficulty Level 3 - Nonce = 93521 - Elapsed = 00:00:00.12 - 000FtlmdNXKvBYFiAeKGPaLMBIkECs9nIbskXiln4og=
Difficulty Level 4 - Nonce = 2286428 - Elapsed = 00:00:03.10 - 00007D4cJR6eYzCvW82UPRAC/dkxEsV55v7+9bVRH/A=
Difficulty Level 5 - Nonce = 380372972 - Elapsed = 00:08:25.45 - 0000032c57uM3MYWkrnzHNv+efN7SJULrrnzRgDrXXQ=
```

stephenhaunts — ProofOfWorkTest.dll — dotnet + bash -c clear; cd "/Applications/Visual Studio.app/Contents/Resources/li..."

```
Difficulty Level 0 - Nonce = 0 - Elapsed = 00:00:00.03 - EYEd2NqKf57Cna1I2rk9UCprZTCrhiT3QcRhS1BQJhA=
Difficulty Level 1 - Nonce = 24 - Elapsed = 00:00:00.00 - 0NboeG5IAZduTUQ2WmALzrfN1FoRDg5nLZN8Vhgmgmg=
Difficulty Level 2 - Nonce = 9478 - Elapsed = 00:00:00.01 - 00Ln0xLQjZuZjuZrw14ne2T3R4niaydfrazohLoLx1k=
Difficulty Level 3 - Nonce = 93521 - Elapsed = 00:00:00.12 - 000FtlmdNXKvBYFiAeKGPaLMBIkECs9nIbskXiln4og=
Difficulty Level 4 - Nonce = 2286428 - Elapsed = 00:00:03.10 - 00007D4cJR6eYzCvW82UPRAC/dkxEsV55v7+9bVRH/A=
Difficulty Level 5 - Nonce = 380372972 - Elapsed = 00:08:25.45 - 0000032c57uM3MYWkrnzHNv+efN7SJULrrnzRgDrXXQ=
```

stephenhaunts — ProofOfWorkTest.dll — dotnet + bash -c clear; cd "/Applications/Visual Studio.app/Contents/Resources/li..."

```
Difficulty Level 0 - Nonce = 0 - Elapsed = 00:00:00.03 - EYEd2NqKf57Cna1I2rk9UCprZTCrhiT3QcRhS1BQJhA=
Difficulty Level 1 - Nonce = 24 - Elapsed = 00:00:00.00 - 0NboeG5IAZduTUQ2WmALzrfN1FoRDg5nLZN8Vhgmgmg=
Difficulty Level 2 - Nonce = 9478 - Elapsed = 00:00:00.01 - 00Ln0xLQjZuZjuZrw14ne2T3R4niaydfrazohLoLx1k=
Difficulty Level 3 - Nonce = 93521 - Elapsed = 00:00:00.12 - 000FtlmdNXKvBYFiAeKGPaLMBIkECs9nIbskXiln4og=
Difficulty Level 4 - Nonce = 2286428 - Elapsed = 00:00:03.10 - 00007D4cJR6eYzCvW82UPRAC/dkxEsV55v7+9bVRH/A=
Difficulty Level 5 - Nonce = 380372972 - Elapsed = 00:08:25.45 - 0000032c57uM3MYWkrnzHNv+efN7SJULrrnzRgDrXXQ=
```



Now it's
your turn!

Spend some time stepping through the
Blockchain Proof of Work example

Summary

Hybrid Encryption

Random Numbers

Hashing + HMACS

Secure Password Storage

Symmetric Encryption - AES

Asymmetric Encryption - RSA

Digital Signatures

Supplemental Materials

Building Secure Applications with Cryptography in .NET

by Stephen Haunts

In this course, you will learn how to use the .NET Framework to protect your data to satisfy confidentiality, integrity, non-repudiation, and authentication.

[Resume Course](#) [Bookmark](#) [Add to Channel](#) [Download Course](#) [Schedule Reminder](#)

[Table of contents](#) [Description](#) [Transcript](#) [Exercise files](#) [Discussion](#) [Related Courses](#)

This course is part of:  .NET Class Libraries with C# Path

Lesson	Duration
Course Overview	2m 8s
Introduction	10m 6s
Cryptographic Random Numbers	8m 26s
Hashing Algorithms	19m 35s
Secure Password Storage	20m 40s
Symmetric Encryption	45m 28s
Asymmetric Encryption	31m 26s

[Expand All](#)

Course author



Stephen Haunts

Stephen Haunts is an experienced Software Developer and Leader who has worked across multiple business domains including Computer Games, Finance, and Healthcare Retail and Distribution. Stephen has...

Course info

Level	Intermediate
Rating	★★★★☆ (46)
My rating	★★★★★
Duration	3h 10m
Updated	16 Oct 2020

Share course

<https://app.pluralsight.com/library/courses/dotnet-cryptography-secure-applications/table-of-contents>

Supplemental Materials

The screenshot shows a course page from Pluralsight. The main title is "Play by Play: Enterprise Data Encryption with Azure Revealed" by Lars Klint and Stephen Haunts. A brief description states: "In this Play by Play, you'll learn about what enterprise data is critical to encrypt, and robust patterns and practices you can apply in your organization." Below the title are buttons for "Resume Course", "Bookmark", "Add to Channel", and "Live mentoring". A navigation bar includes "Table of contents", "Description", "Exercise files", "Discussion", and "Recommended". The "Table of contents" tab is selected, indicated by an orange underline. At the bottom, there's a "Course Overview" section with a play button, a progress bar showing "2m 4s", and a "Expand all" link. On the right side, there's a sidebar titled "Course authors" featuring profile pictures and names for Lars Klint and Stephen Haunts. The sidebar also contains "Course info" sections for Level (Beginner), Rating (★★★★★), My rating (★★★★★), Duration (0h 57m), and Released (10 Mar 2017). It ends with a "Share course" section and social sharing icons for Facebook, Twitter, Google+, and LinkedIn.

Play by Play: Enterprise Data Encryption with Azure Revealed

by Lars Klint and Stephen Haunts

In this Play by Play, you'll learn about what enterprise data is critical to encrypt, and robust patterns and practices you can apply in your organization.

Resume Course Bookmark Add to Channel Live mentoring

Table of contents Description Exercise files Discussion Recommended

Course Overview Expand all

Course authors

Lars Klint

Stephen Haunts

Course info

Level Beginner

Rating ★★★★☆

My rating ★★★★☆

Duration 0h 57m

Released 10 Mar 2017

Share course

f t g+ in

<https://app.pluralsight.com/library/courses/play-by-play-enterprise-data-encryption-with-azure-revealed>



Stephen Haunts
www.stephenhaunts.com
@stephenhaunts



STEPHEN HAUNTS

TRAINING



block 3