# 10.2 Asymmetric Encryption and Hashing

**Cybersecurity**

Cryptography Day 2

© 2020 Trilogy Education Services, a 2U, Inc. brand. All Rights Reserved.

# Class Objectives

By the end of today's class, you will be able to:

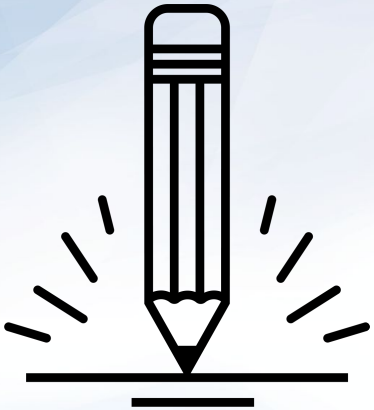
Calculate the required number of symmetric and asymmetric keys based on the number of people exchanging secure messages.

Use GPG to generate keys, and encrypt and decrypt private messages.

Use hashes to validate the integrity of data.

Use digital signatures to validate the authenticity of data.

# Cryptography Review



# **Activity:** Cryptography Refresher

In the activities today, you will continue your role of security analyst at Hill Valley Police Department.

In this review activity, you must encrypt and decrypt a message using OpenSSL.

**Suggested Time:**
20 Minutes

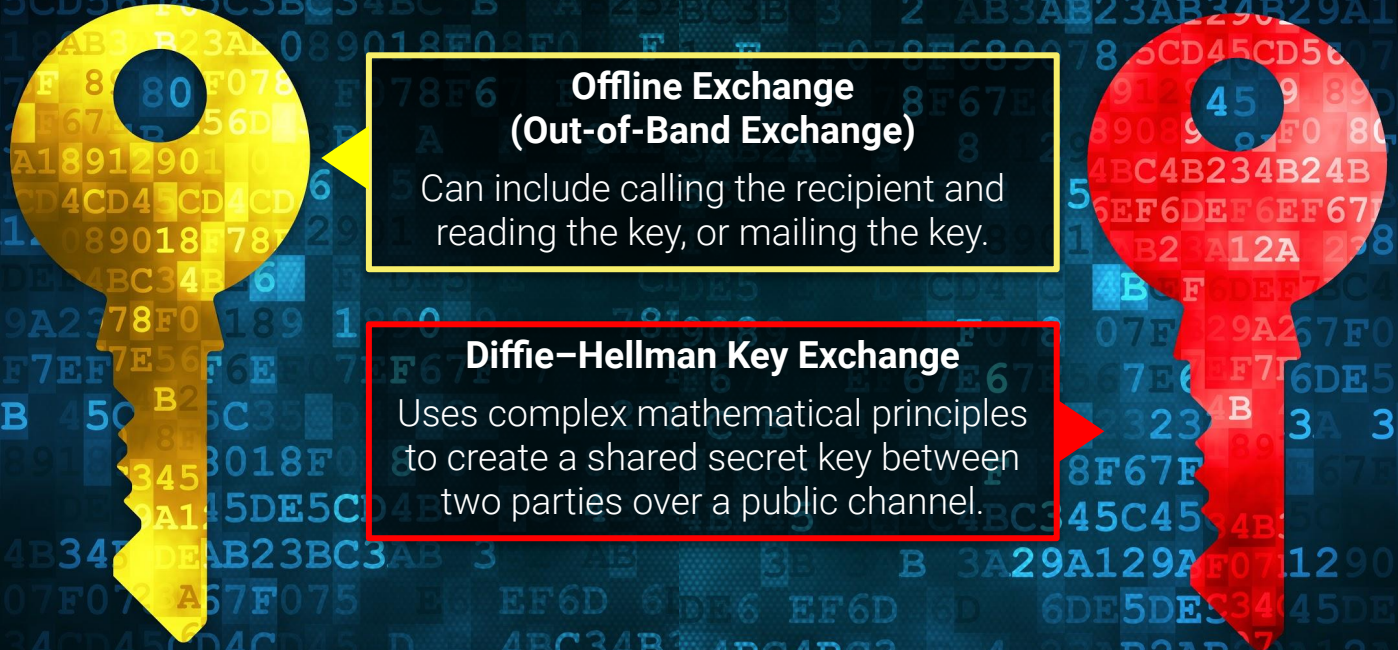
# **Time’s Up!** Let’s Review.

# Key Management and Exchange

In the previous activity, we used **symmetric key encryption**.
While its benefits include speed, efficiency and simplicity of use, symmetric encryption also has its disadvantages.

# Disadvantage One: Secure Key Exchange

As more individuals are included in the key exchange, there needs to be a key for each combination of individuals.



**Offline Exchange (Out-of-Band Exchange)**

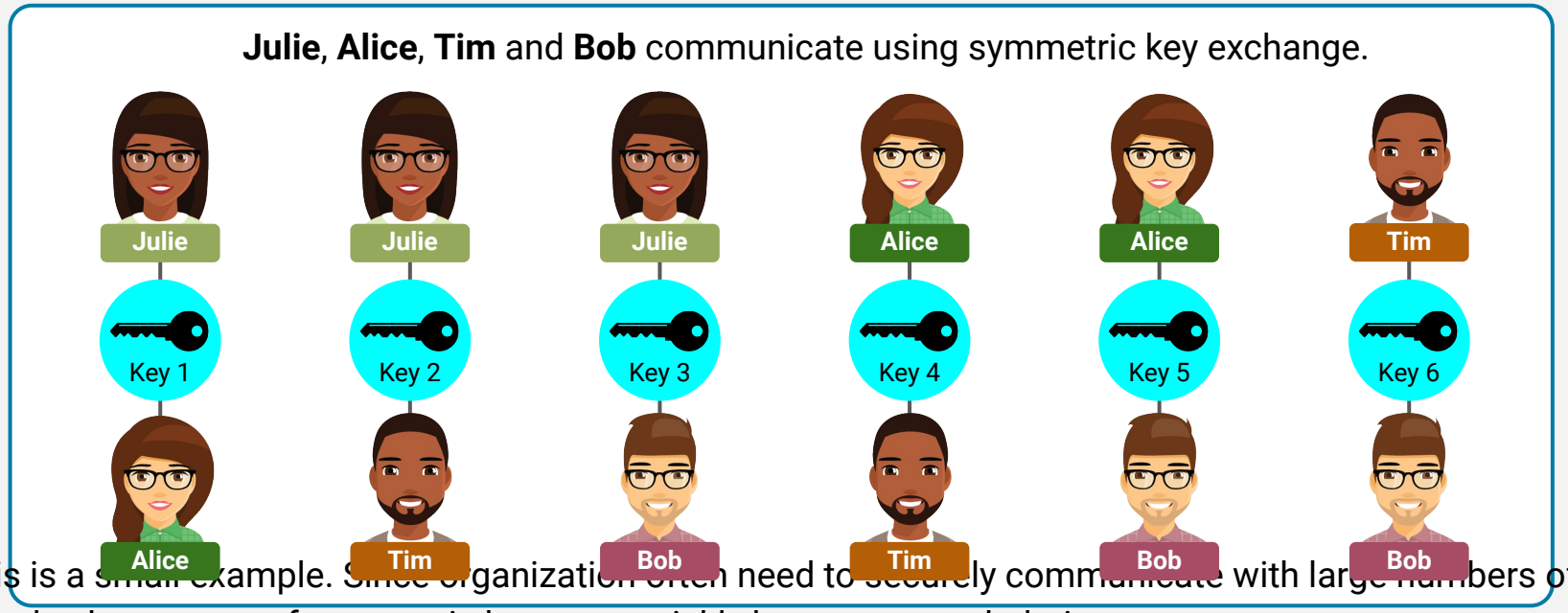
Can include calling the recipient and reading the key, or mailing the key.

**Diffie–Hellman Key Exchange**

Uses complex mathematical principles to create a shared secret key between two parties over a public channel.

# Disadvantage Two: Key Management

As more individuals are included in the key exchange, a key must be created for each combination of individuals.




This is a small example. Since organizations often need to securely communicate with large numbers of people, the amount of symmetric keys can quickly become overwhelming.

# Disadvantage Two: Key Management

Count of symmetric keys = (N * (N-1))/ 2

**For an organization with 7 employees**

```
(7 * 6) / 2 = 42/2 = 21
```

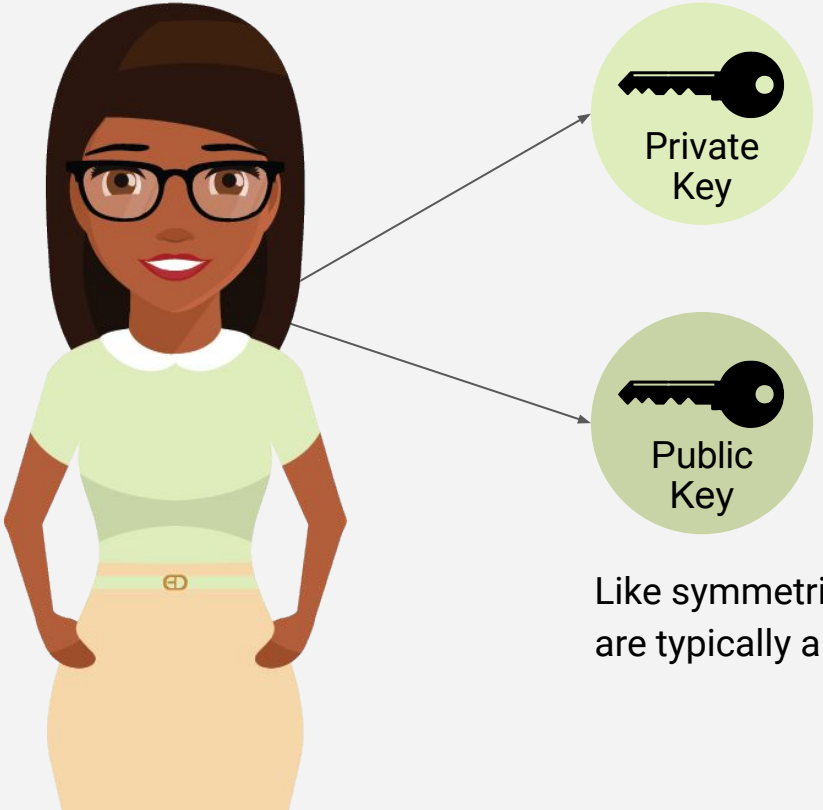
**For an organization with 1,000 employees**

```
(1000 * 999) / 2 = 499,500
```

These disadvantages can be addressed with **asymmetric key encryption**.

# Asymmetric Key Encryptions

Each individual possesses a two-key pair:



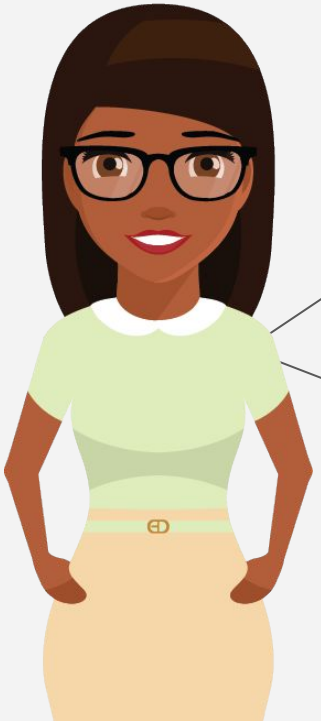

Private keys are kept secret and can affect confidentiality of messages if exposed.

Public keys are public and accessible to all.

Like symmetric keys, these public and private keys are typically a random alphanumeric string.

# Asymmetric Key Encryptions

**Scenario:** Tim wants to send Julie his bank account number using asymmetric key encryption.



Julie will have a public and a private key.

Private Key

Public Key

The public key is truly public, accessible on Julie's website.

# Asymmetric Key Encryptions

**Scenario:** Tim wants to send Julie his bank account number using asymmetric key encryption.

Tim creates a **plaintext message** containing his bank account number.


Tim goes to Julie's website, gets her **public key,** and uses it to encrypt his message.

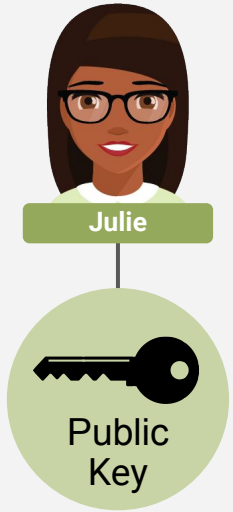
# Asymmetric Key Encryptions

**Scenario:** Tim wants to send Julie his bank account number using asymmetric key encryption.


Tim sends the encrypted message to Julie.

slack

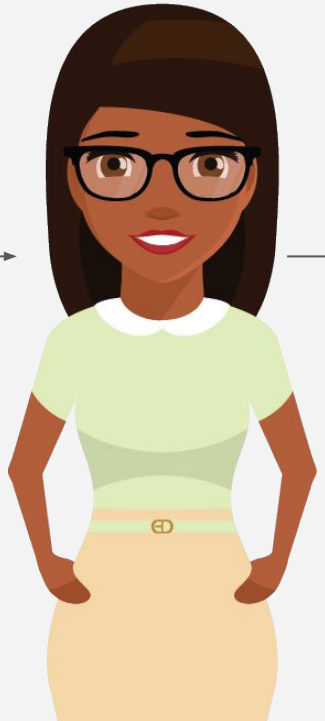
This can be sent over email or Slack without fear of interception.




The message can only be decrypted by Julie with her matching private key.

# Asymmetric Key Encryptions

**Scenario:** Tim wants to send Julie his bank account number using asymmetric key encryption.


Julie receives Tim's encrypted message and decrypts it using her matching private key.


Julie can now see Tim's bank number.

# Key Management

While symmetric key count grows exponentially larger as the number of employees increase, asymmetric count stays proportionate and manageable with increases.

**For a company with twelve employees:**

**Symmetric**

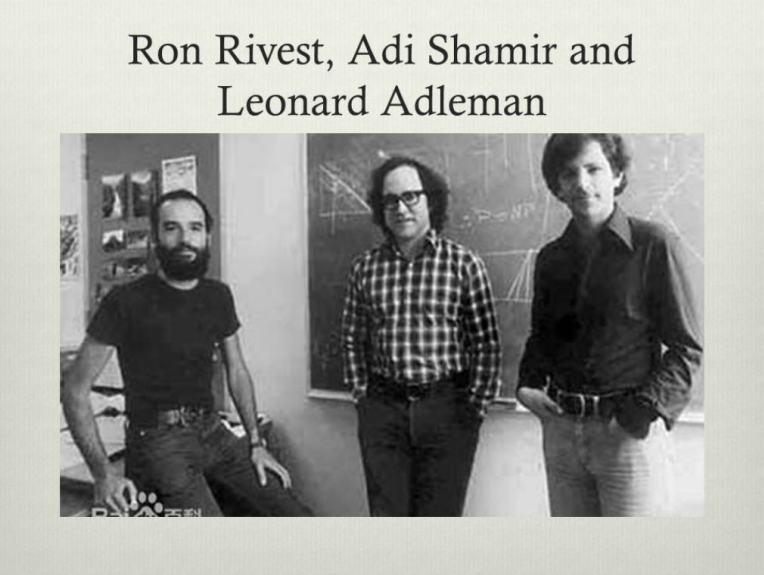
$(12 * 11) / 2 = 66$

**Asymmetric**

$(12 * 2) = 24$

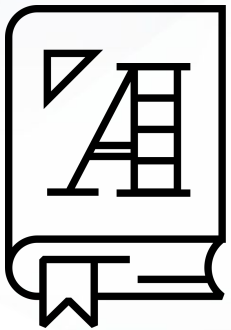
# RSA

Asymmetric encryption uses an algorithm called RSA, introduced in 1977 and named after the last names of its creators.

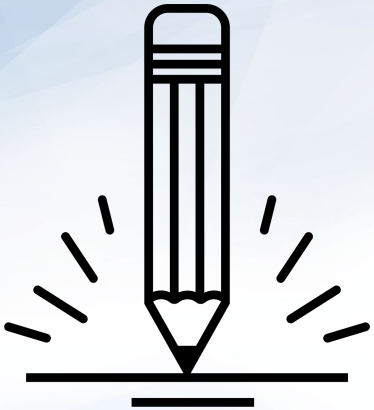



*Popularly known as **Rivest-Shamir-Adleman ( RSA ) algorithm,** it was built after **Whitefied Diffie, Martin Helman** and **Ralph Merkle** introduced public key cryptography in **1976**. **RSA** is an **asymmetric cryptography** popularly known as **public key cryptography.***

(Source: Devyani Vij, Medium)



**RSA** is the asymmetric algorithm standard used around the world. It works by employing the complexity of factoring large prime numbers.



# **[Optional] Activity:** Optimizing with Asymmetric Public Keys

In this activity, you will compare the key counts required for communicating with asymmetric and symmetric key cryptography.

**Suggested Time:**
Complete if class falls on Saturday

# **Time’s Up!** Let’s Review.

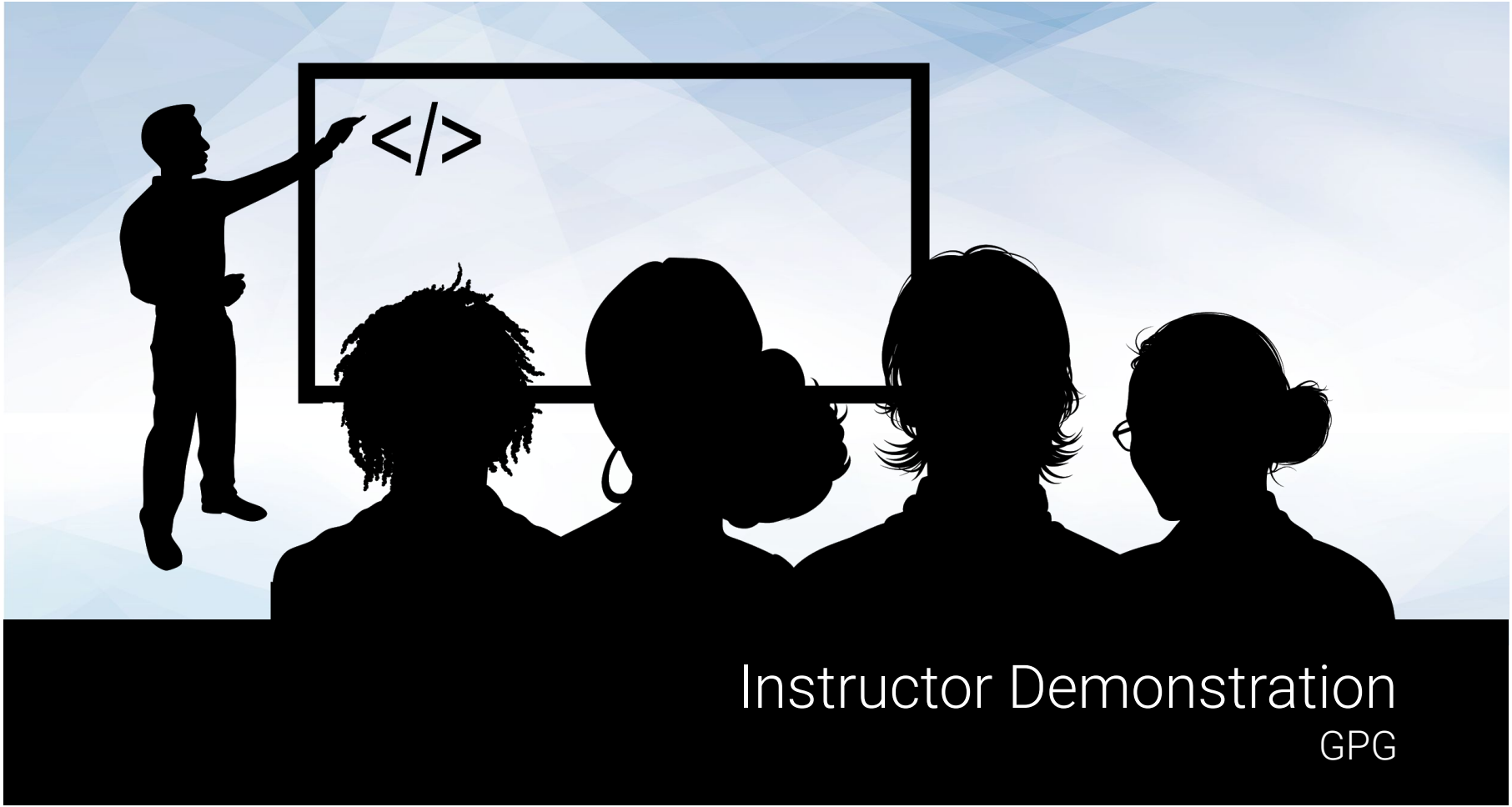
# Applying Public Key Cryptography with GPG

# GNU Privacy Guard (GPG)

GPG is a command-line tool used to simplify the creation, encryption, and decryption of asymmetric key cryptography.

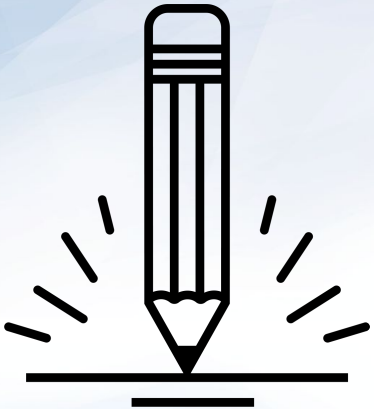
GPG is a free program, available on many Linux distributions, that runs symmetric and asymmetric encryption algorithms.

In the following demonstration, we will use the bank account scenario with GPG to create a key pair for asymmetric encryption and decryption.




# Instructor Demonstration

## GPG



# **Activity:** GPG

In this activity, you will use asymmetric encryption with GPG to encrypt and decrypt text files.

**Suggested Time:**
20 Minutes

# **Time’s Up!** Let’s Review.


Countdown timer

15:00

(with alarm)

# Hashing and Data Integrity

So far, we've used cryptography to protect privacy and confidentiality. It can also be applied to protect the **integrity** of data.

# Hashing

Similar to encryption, hashing uses algorithms to input data and generate a unique output.

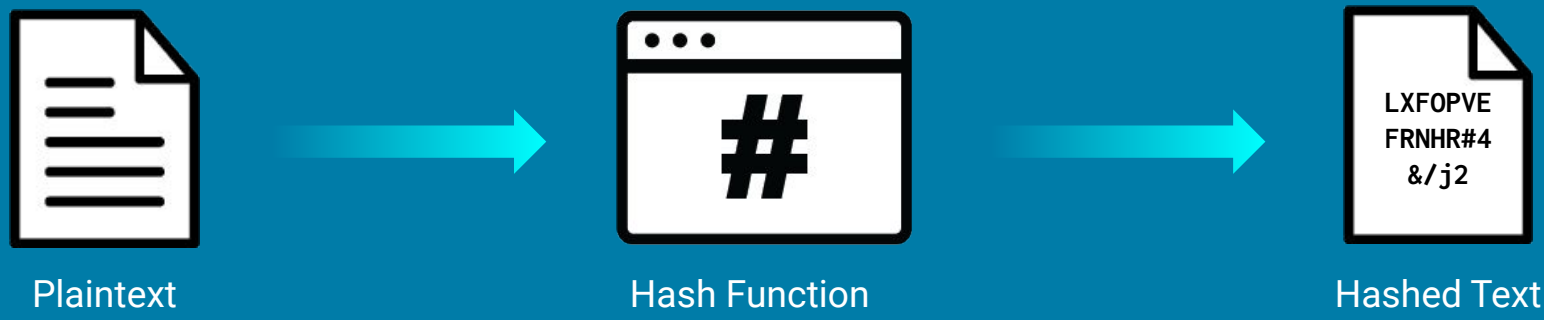
**Hashing** is a cryptographic method used to verify the integrity of data.


While this looks similar to cryptography,
we'll see that there are several significant differences.

# Difference One: Keys vs. No Keys

Encryption takes plaintext and converts it to ciphertext with a key and an algorithm.


Hashing takes plaintext and converts it to a **message digest** with an algorithm, and no key.



# Difference One: Keys vs. No Keys

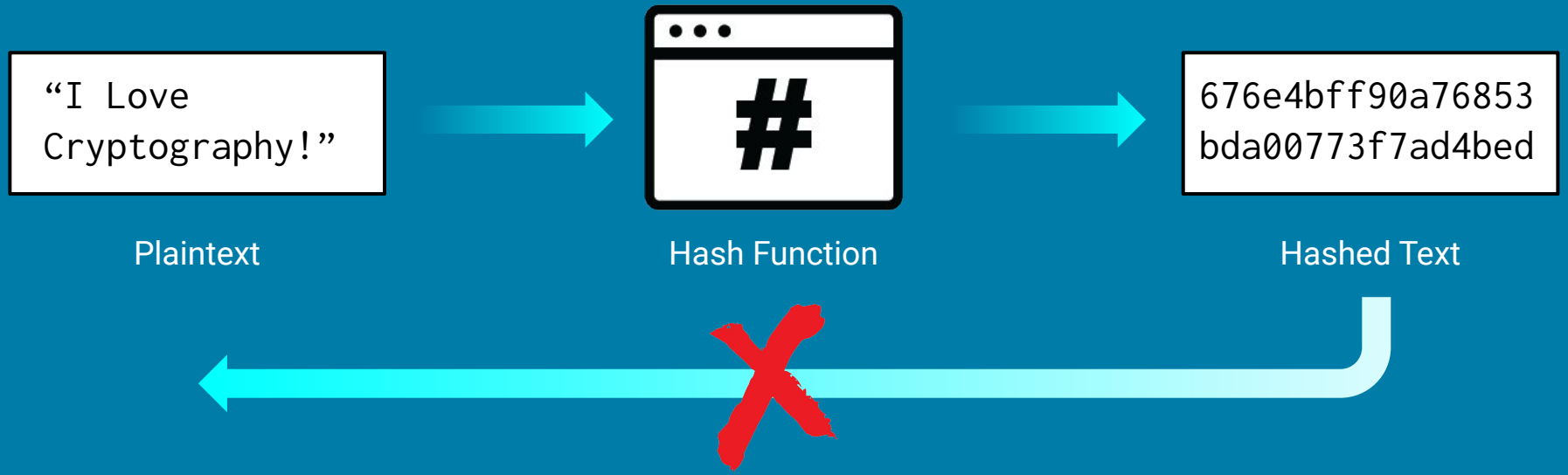
Message digests are also known as fingerprints, hashes, and checksums.

They are unique identifiers of the plaintext that are outputted from the algorithm. For example:

```
676e4bff90a76853bda00773f7ad4bed
```

# Difference Two: Reversible vs. Irreversible

Encryption converts ciphertext and plaintext back and forth using encryption and decryption.




Hashing is a **one-way function,** meaning it cannot be converted back to plaintext.

# Difference Three: Output Lengths

Hashing algorithms output **fixed lengths**. Regardless of the input length, the output length is always the same.

| | | |
|---|---|---|
| `I Love Cryptography!` | # | `676e4bff90a76853bda00773f7ad4bed` |
| `Hi` | # | `c1a5298f939e87e8f962a5edfc206918` |
| *We hold these truths to be self-evident, that all men are created equal, that they are endowed by their Creator with certain unalienable Rights, that among these are Life, Liberty and the pursuit of Happiness...* | # | `124650d689d47f777f715a25260a29c2`<br>This is the hash of the entire Declaration of Independence! |

# Difference Four: Goals

Encryption's main goal is **privacy**.

Hashing's main goal is **integrity**.

# Difference Four: Goals

If a small change is made to the input of a hashing algorithm, the message digest is completely different.

```
I Love Cryptography! = 676e4bff90a76853bda00773f7ad4bed
```

```
I Love Cryptography!! = 4e6fc433ff57a6c4a854cbbeff65f61a
```

```
I Love Cryptography? = d7a5035b338b66bf7f50dc85dbf1a279
```

```
I Luv Cryptography! = ad7e7b47b58e28a2894f5fcf9dc41691
```

# Hashing Algorithms

Hashing has several algorithms:

## SHA

stands for Secure Hashing Algorithms, and includes its successors, **SHA1** and **SHA2.**

## MD

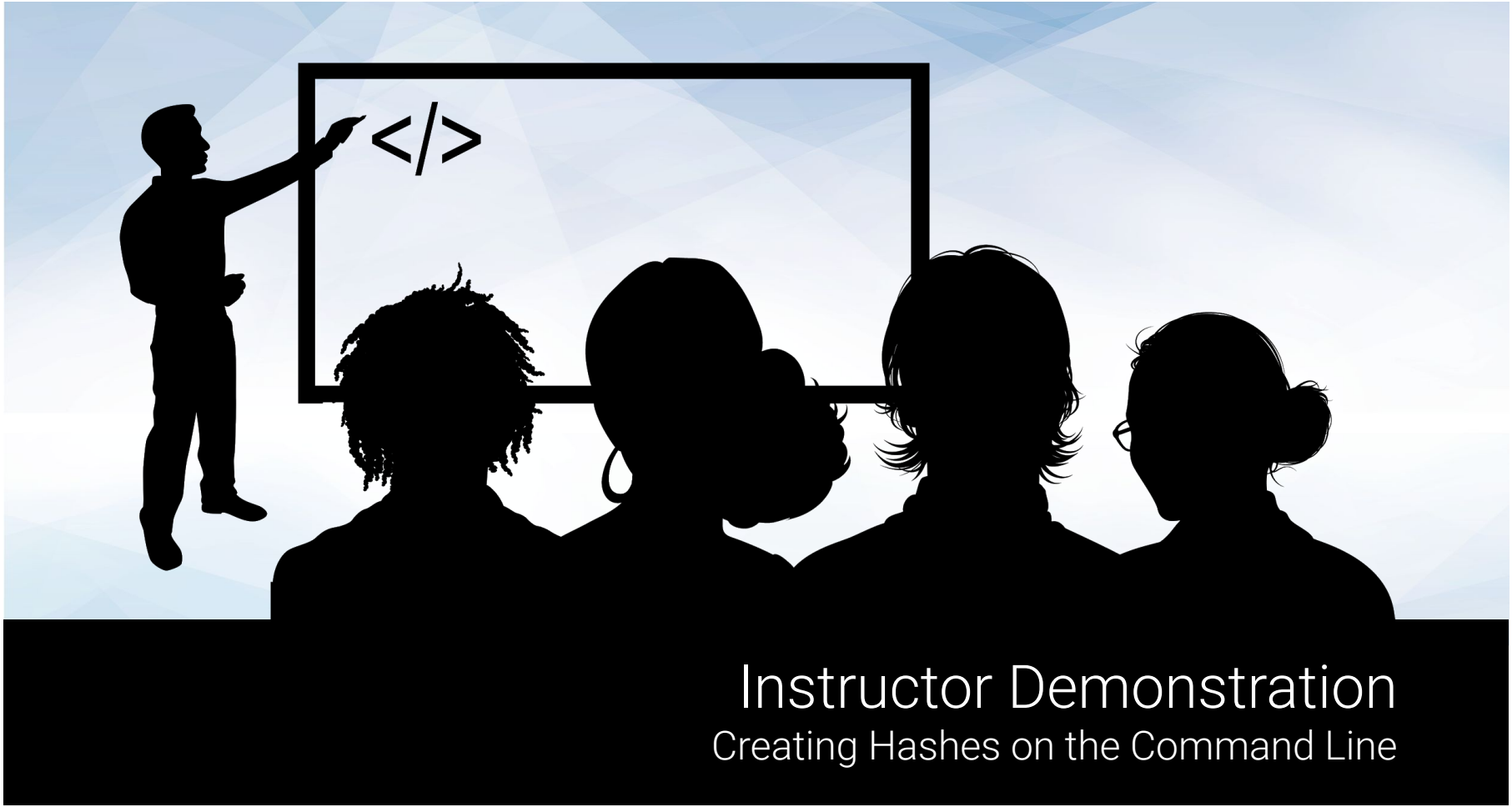
stands for Message Digest and has several variations: **MD2**, **MD4**, and **MD5**.

## SHA2

has variations with different security strengths: **SHA-256** and **SHA-512**.

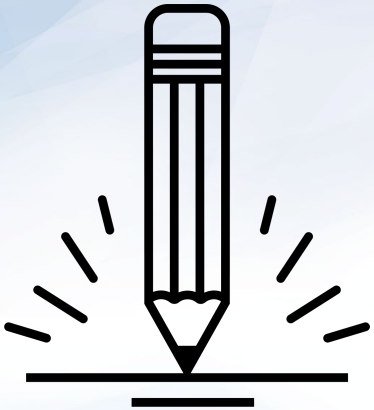
## LM and NTLM

are hashes used by Windows.




# Instructor Demonstration

## Creating Hashes on the Command Line



# **Activity:** Generating Hashes

In this activity, you will verify the integrity of files by generating hashes of current and backup files, and then comparing them.

**Suggested Time:**
17 Minutes

# **Time’s Up!** Let’s Review.

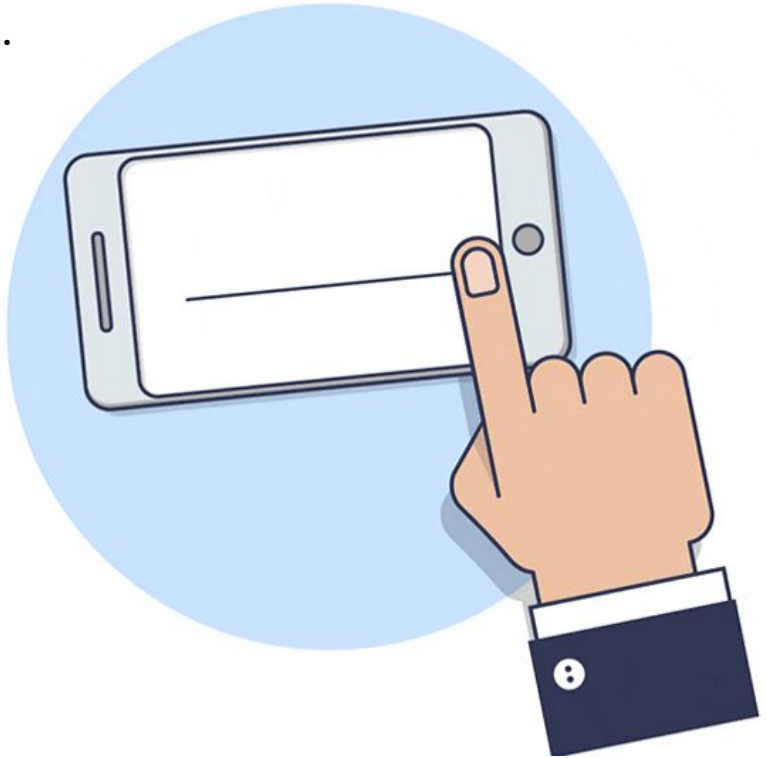
# Digital Signatures

Cryptography can also be used to validate **authenticity** using **digital signatures**.

# Digital Signatures

A digital signature is a mathematical scheme used to verify the authenticity of digital data.

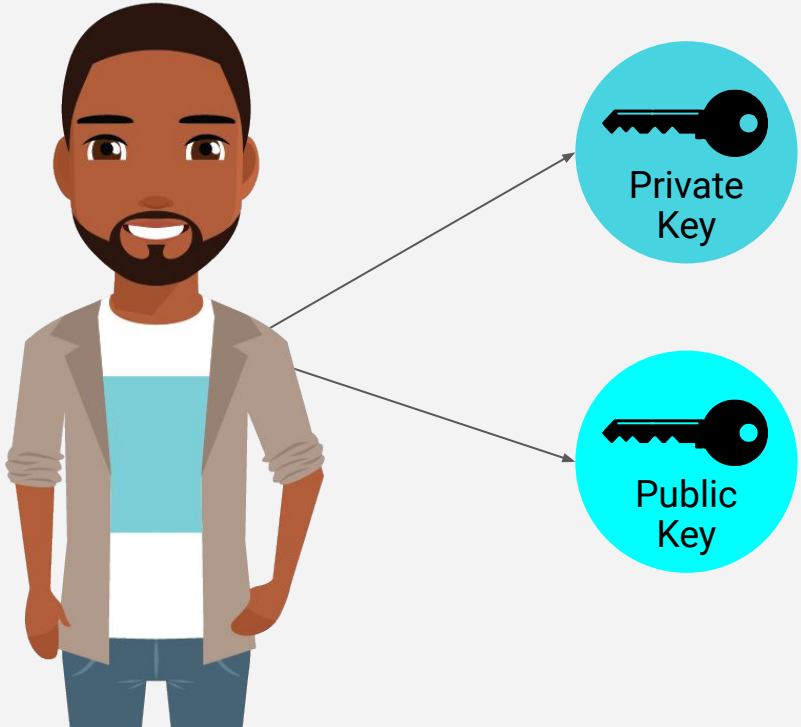
- In the United States and several other countries, digital signatures are considered legally binding.
- Similar to asymmetric encryption, digital signatures also use public key cryptography.
- However, digital signatures use public and private keys in reverse.



(Source)

# Digital Signatures

**Scenario:** Tim wants to send Julie a message that says “Transfer $500 to the account I provided to you,” and then digitally sign the message.




**STEP 1**

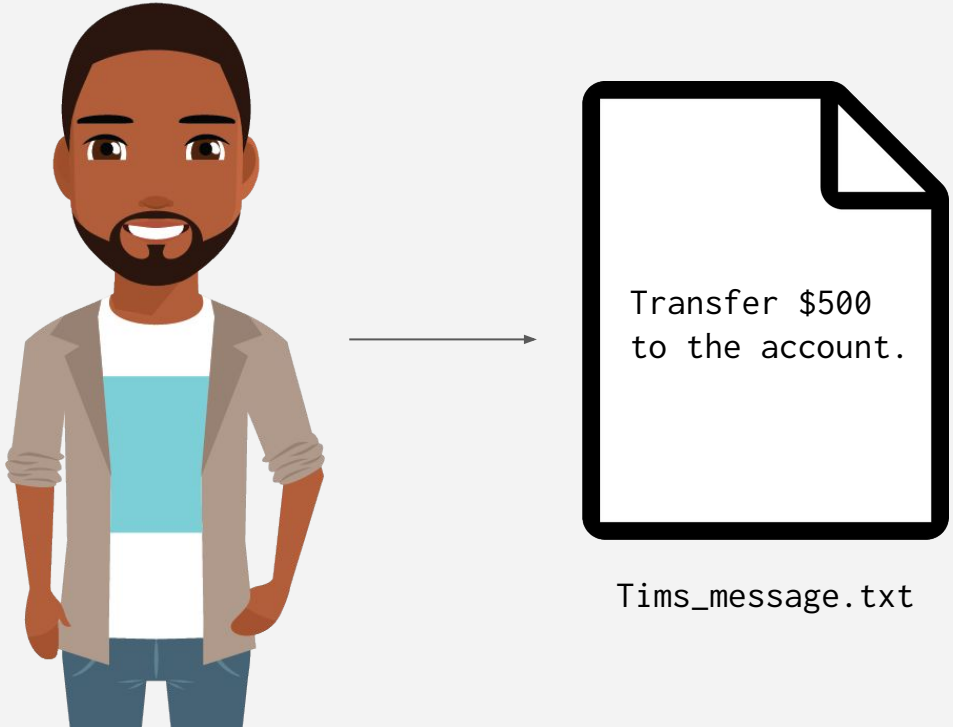
## Key Creation

Tim will have a public and a private key.

The public key is truly public, and accessible on his website.

# Digital Signatures

**Scenario:** Tim wants to send Julie a message that says “Transfer $500 to the account I provided to you.” and then digitally sign the message.



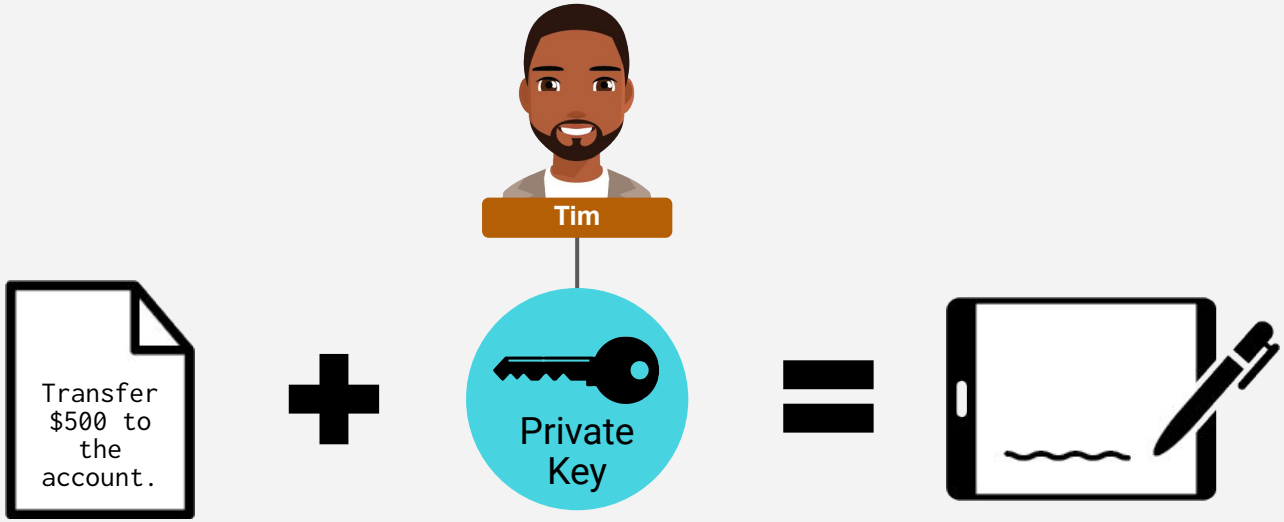

**STEP 2**

## Creating the Message

Tim places his message inside a file.

# Digital Signatures

**Scenario:** Tim wants to send Julie a message that says “Transfer $500 to the account I provided to you.” and then digitally sign the message.




Tims_message.txt

Digital Signature

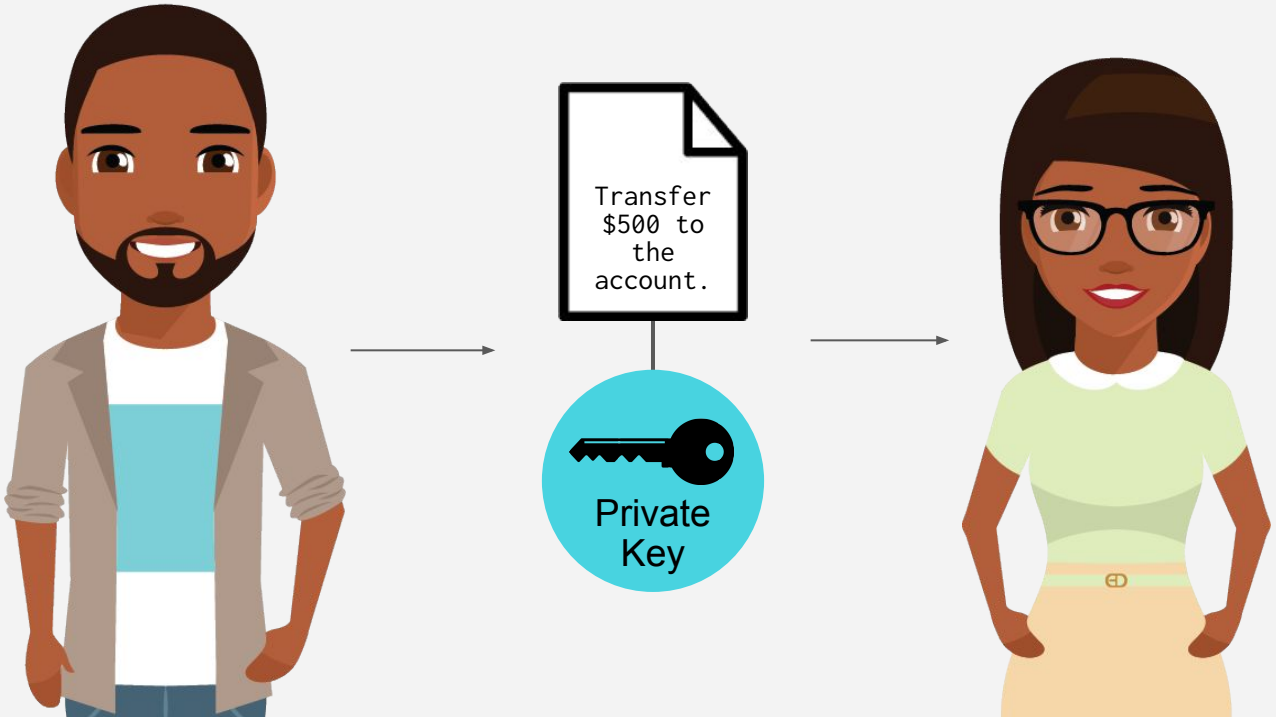
**STEP 3**

## Signing the Message

Tim signs the message with his private key in order to create a digital signature.

# Digital Signatures

**Scenario:** Tim wants to send Julie a message that says “Transfer $500 to the account I provided to you.” and then digitally sign the message.



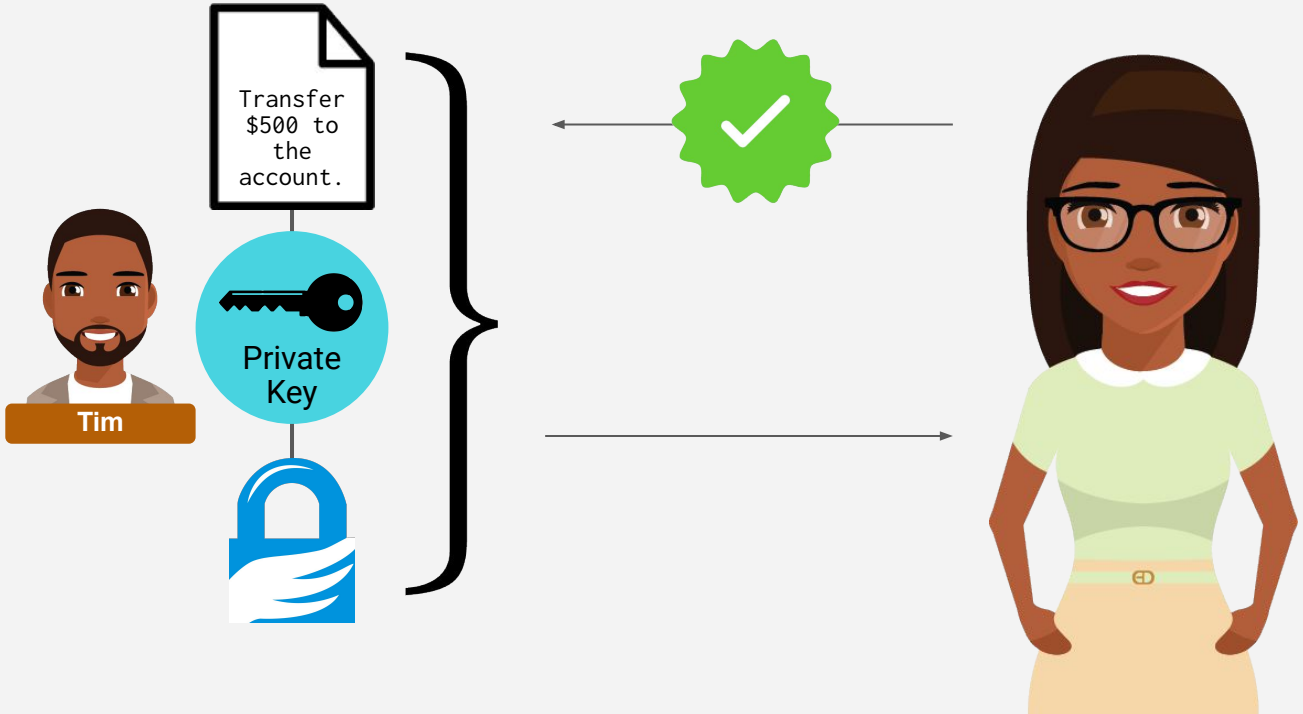

**STEP 4**

## Sending the Message

Tim sends the digital signature to Julie along with his plaintext message.

# Digital Signatures

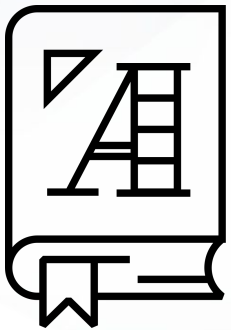
**Scenario:** Tim wants to send Julie a message that says “Transfer $500 to the account I provided to you.” and then digitally sign the message.




**STEP 5**

## Validating the Signature

Julie uses Tim's public key to validate the signature, using a validation tool such as GPG.



That scenario depicted a **detached signature**, in which the message and the signature are sent separately.

# Digital Signatures

Other digital signatures include:

01

**All at Once**
A signature appended to an encrypted message.

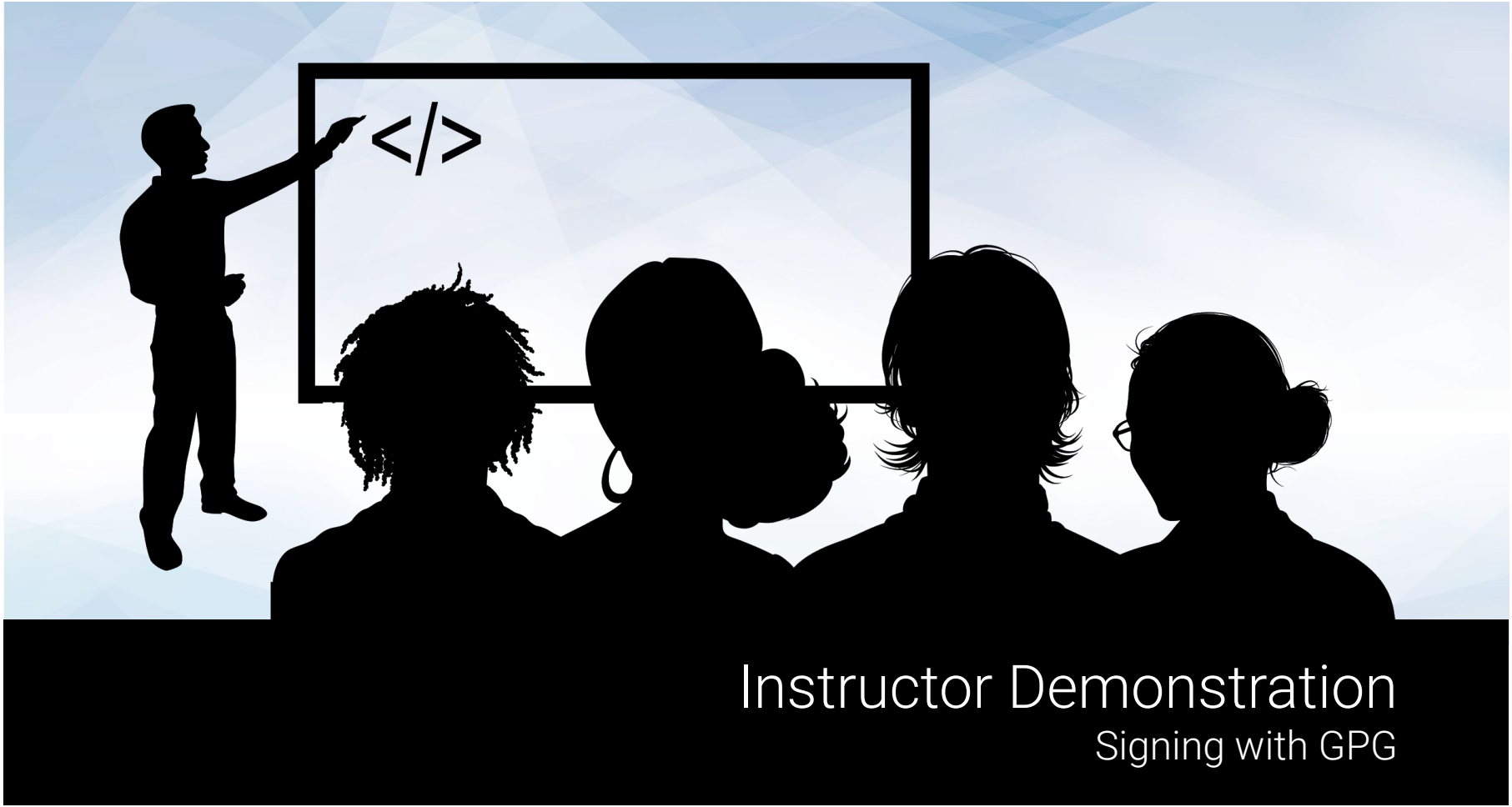
02

**Clearsigned**
A signature appended to an unencrypted message.

03

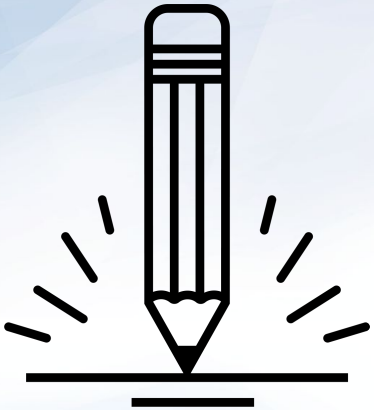
**Signed Hash**
Instead of signing a message, a hash is created first and the hash is signed for verification.

In the next demo, we will apply a detached digital signature with GPG.




# Instructor Demonstration

## Signing with GPG



# **Activity:** Digital Signatures

In this activity, you will use GPG to determine the authenticity of messages.

**Suggested Time:**
20 Minutes

# **Time’s Up!** Let’s Review.

# Any Questions?


The End