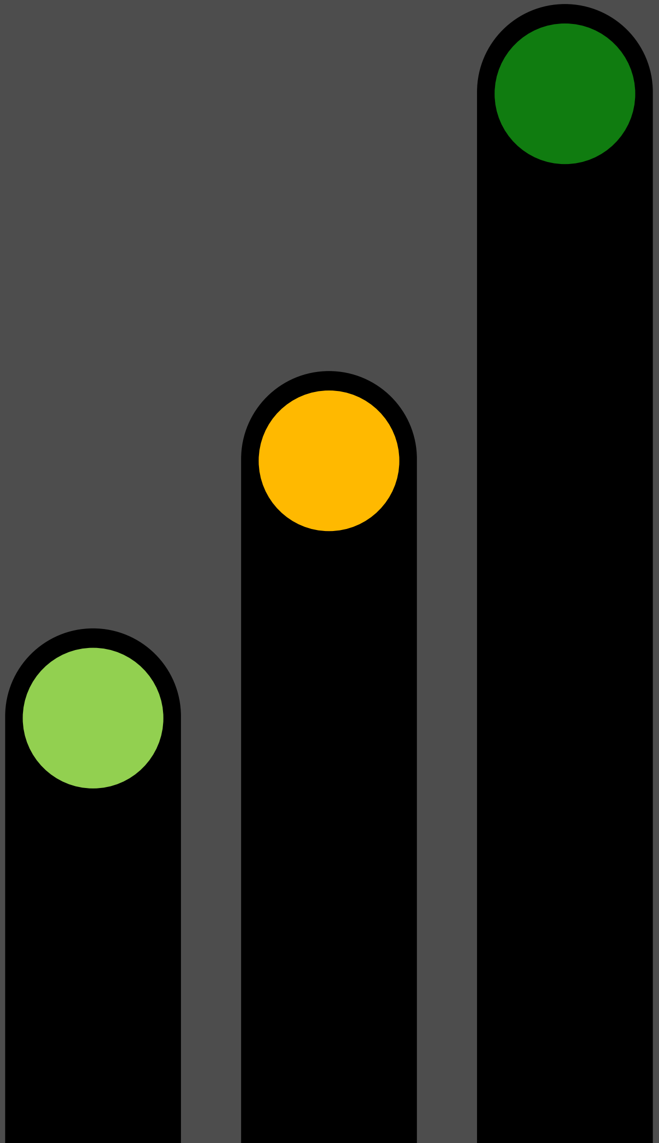
# 2024

# The Passwordless Future with Passkeys

Alex Seigler, CISSP, Enterprise Security Architect, Electronic Arts

# Agenda

- Who is this guy?
- Some history
- Passkeys!



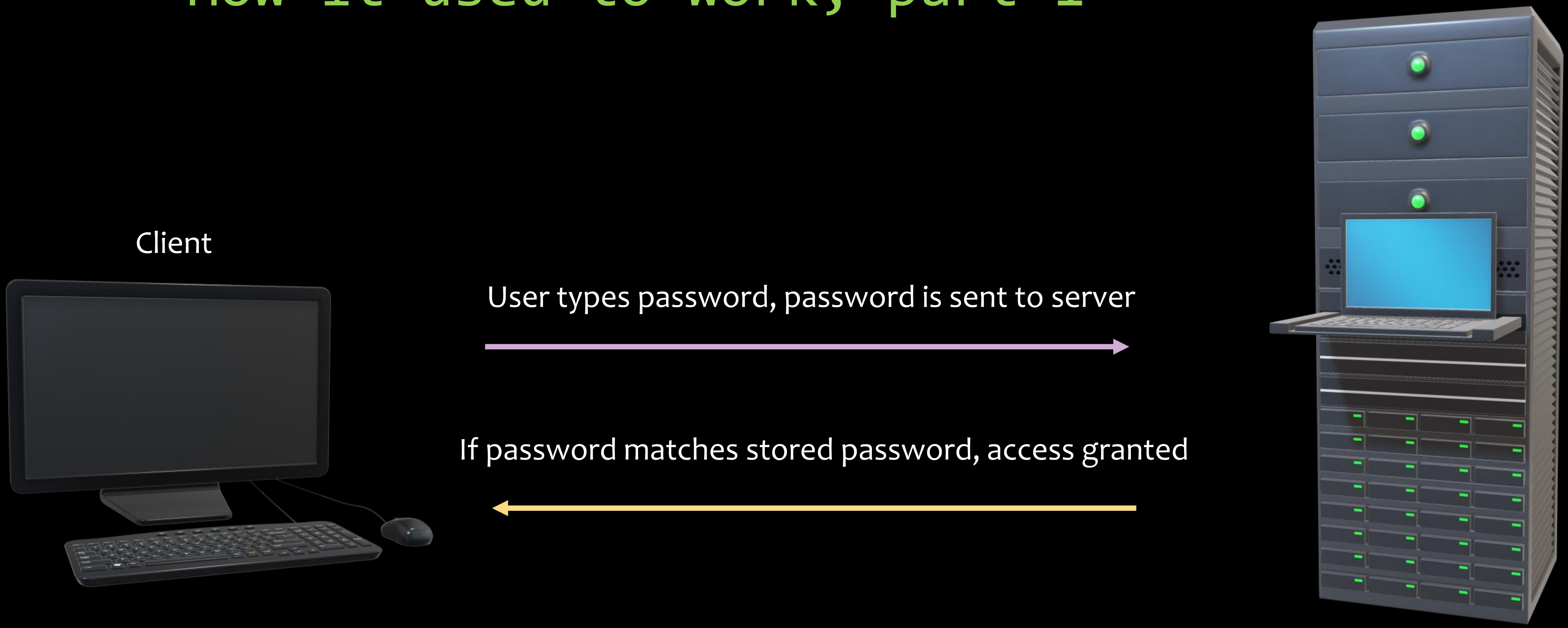
# $whoami

- Alex Seigler
- Husband, father, Sanford native, technology enthusiast, SCPS alumni
- Former Cybersecurity Adjunct Instructor, University of Central Florida
- Work primarily in authentication engineering
- FIDO2/WebAuthn/Passkey expert
- Open source proponent
- linkedin.com/in/aseigler/
- github.com/aseigler
- x.com/alexseigler

# History

How did we get here?

# How it used to work, part 1



Server

Client

User types password, password is sent to server

If password matches stored password, access granted

Password1

Client password = Password1

# How it used to work, part 2

Client

User types password, password hashed on client, *digest* sent to server

If hashed password digest matches stored digest, access granted

Server

HASH(Password1) =
64F12CDDAA88057E06A81B54E73B949B

Client1 hashed password =
64F12CDDAA88057E06A81B54E73B949B

# What is hashing?

The process of converting arbitrary data into a fixed length string of letters and numbers

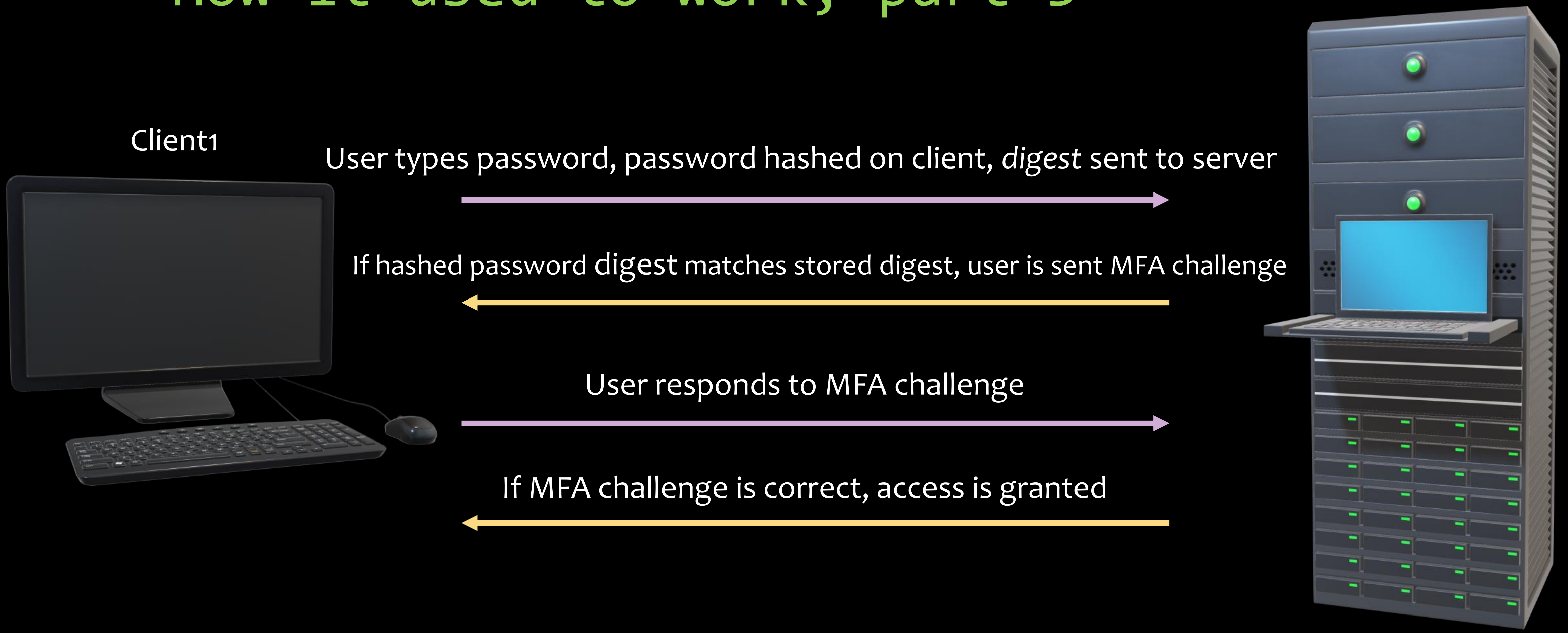
# Hash algorithm properties:

- Deterministic
  - If you use the same data and the same hashing algorithm, the hash digest should always be the same
- Fast
  - Hash algorithms must be very fast
- Irreversible
  - One–way function – impossible to regenerate the original message or data from the hash digest

# Links:

https://www.okta.com/identity-101/hashing-algorithms/
https://www.codecademy.com/resources/blog/what-is-hashing/

# How it used to work, part 3



Client1

Server

User types password, password hashed on client, *digest* sent to server

If hashed password digest matches stored digest, user is sent MFA challenge

User responds to MFA challenge

If MFA challenge is correct, access is granted

HASH(Password1) =
64F12CDDAA88057E06A81B54E73B949B

Client1 hashed password =
64F12CDDAA88057E06A81B54E73B949B

# What is MFA?

Multi-factor authentication is a security process which requires more than one method to verify a user's identity, adding an additional layer of protection

## Factors:

- Something you know
  - A password, passphrase, PIN
- Something you have
  - Phone text/app notification, number generating app, email link
- Something you are
  - Biometrics

## Links:

https://www.onelogin.com/learn/what-is-mfa
https://www.seminolestate.edu/cts/mfa

password

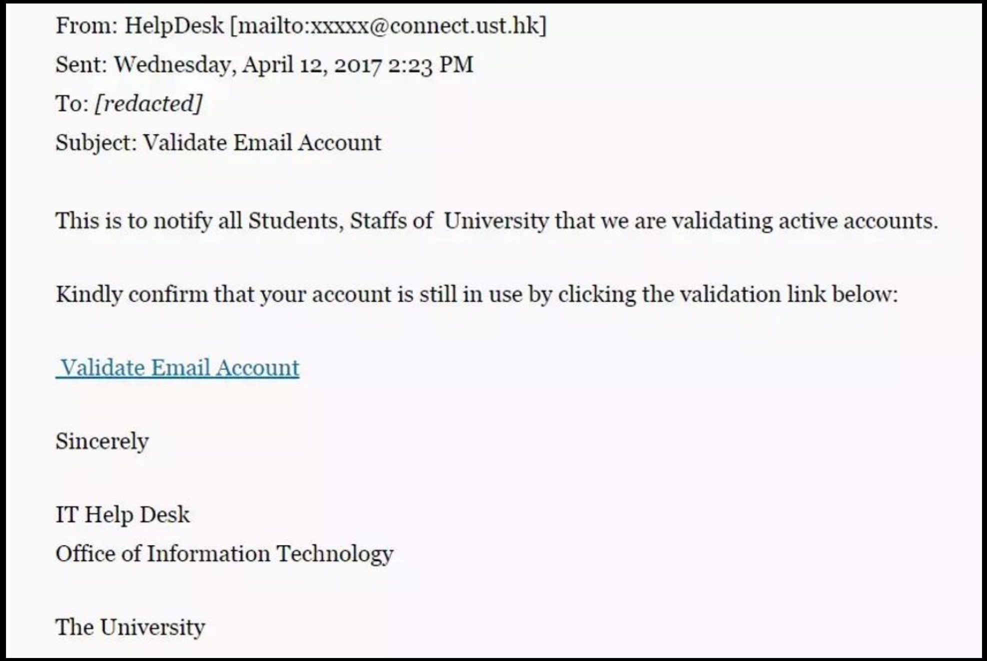
phishable

| | |
|---|---|
| SMS OTP | phishable |
| email OTP | phishable |
| magic link | phishable |
| app push | phishable |

# What is phishing?

Phishing is **an attempt by cybercriminals posing as legitimate institutions**, usually via email, to obtain sensitive information from targeted individuals

## Example:



From: HelpDesk [mailto:xxxxx@connect.ust.hk]
Sent: Wednesday, April 12, 2017 2:23 PM
To: *[redacted]*
Subject: Validate Email Account

This is to notify all Students, Staffs of University that we are validating active accounts.

Kindly confirm that your account is still in use by clicking the validation link below:

Validate Email Account

Sincerely

IT Help Desk
Office of Information Technology

The University

## Links:

https://www.phishing.org/what-is-phishing
https://www.cloudflare.com/learning/access-management/phishing-attack/

# Passkeys!

# Why passkeys?

1:1

Unique Credential
Per Service

Standards-based

Phishing
Resistant

Asymmetric Cryptography
without a bunch of extra stuff

Native
Support

# The Vision

as easy to use
as a password

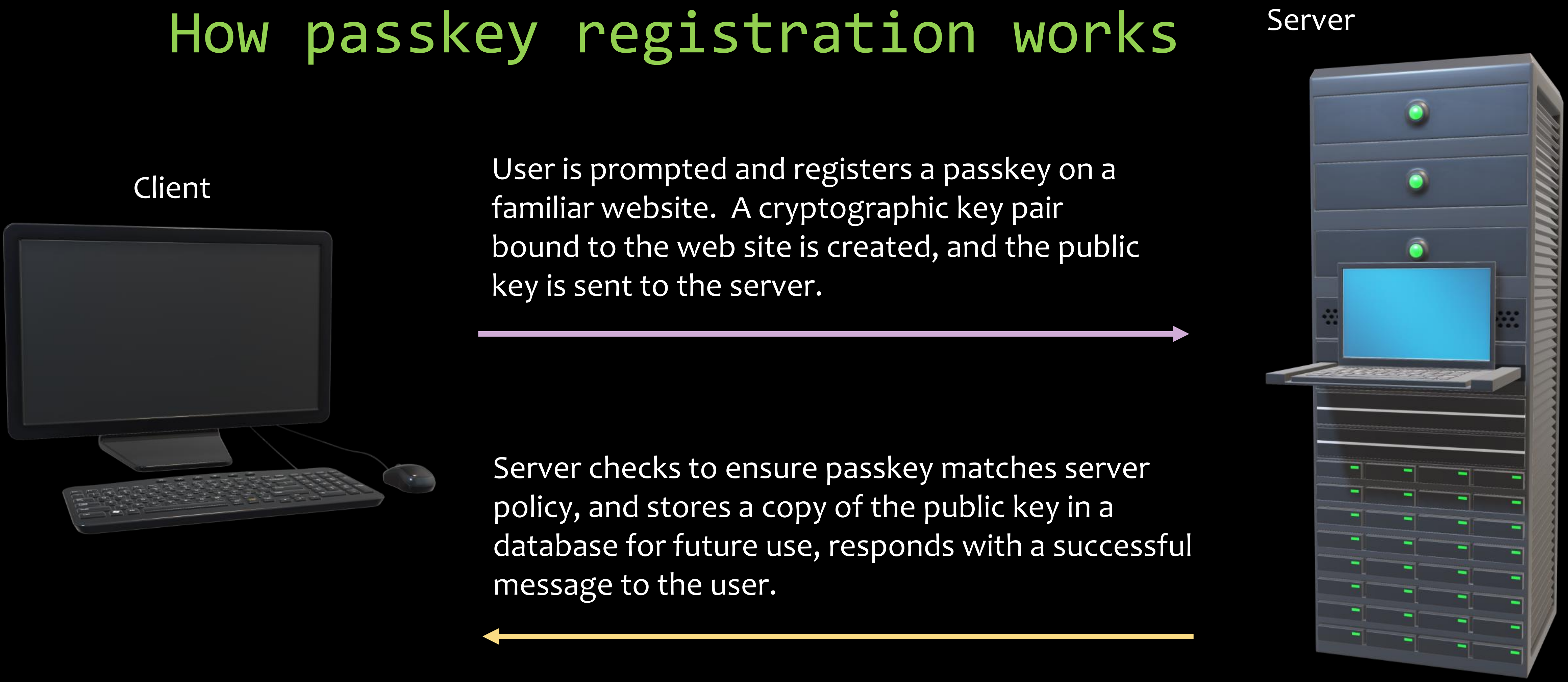
easy to recognize
and understand

leverages existing
investments

durable across
device loss

works at
global scale

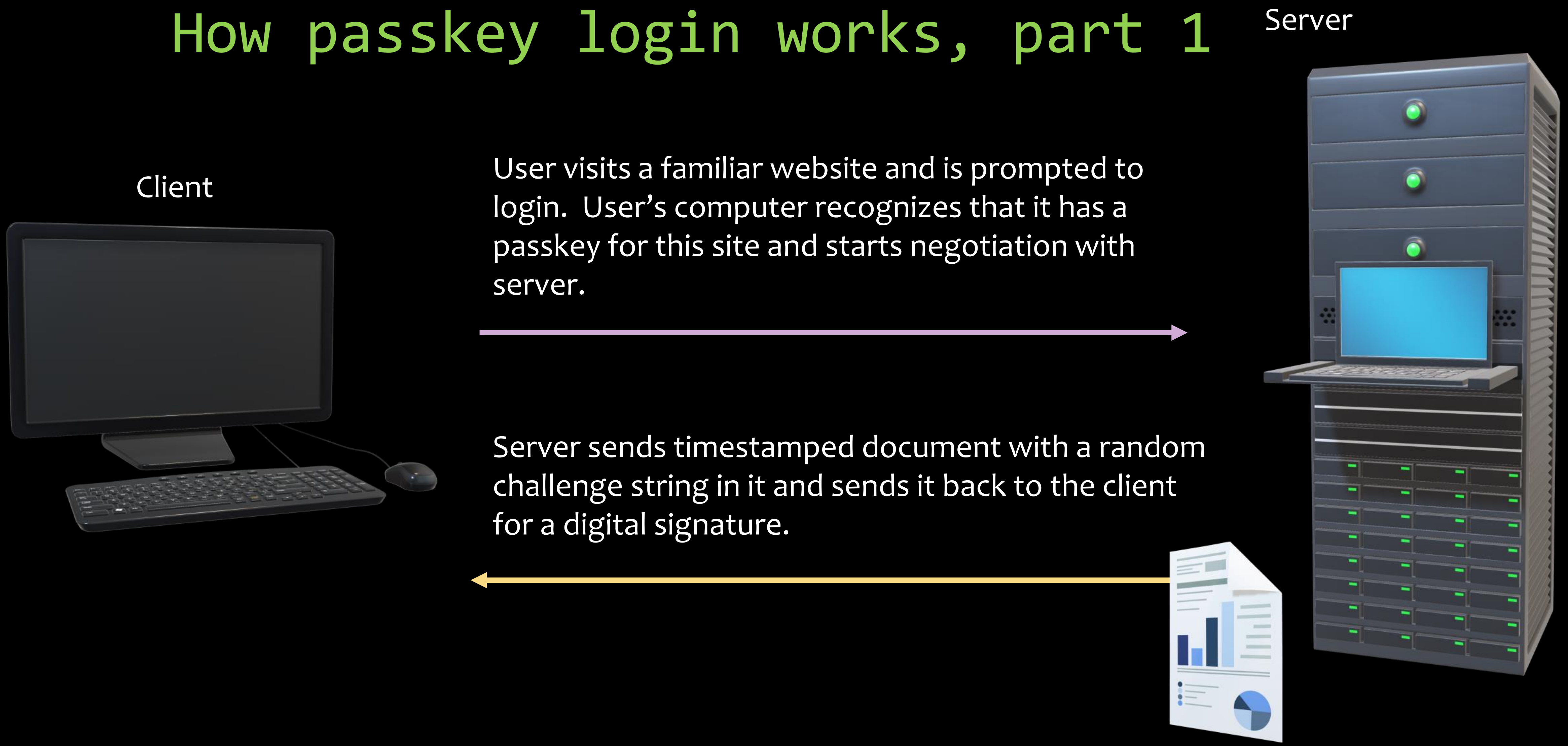
# How passkey registration works



User is prompted and registers a passkey on a familiar website.  A cryptographic key pair bound to the web site is created, and the public key is sent to the server.

Server checks to ensure passkey matches server policy, and stores a copy of the public key in a database for future use, responds with a successful message to the user.

# How passkey login works, part 1



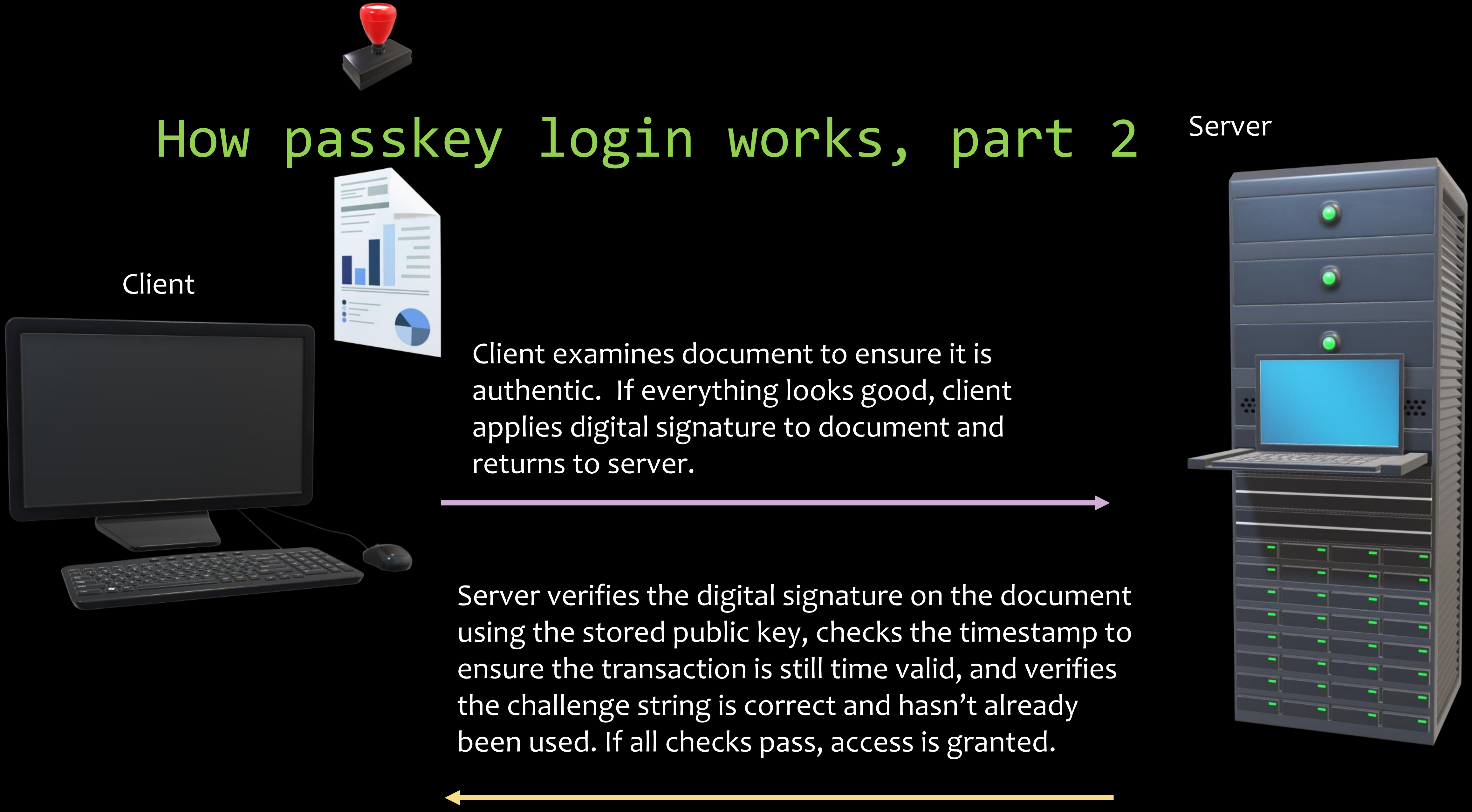
Server

Client

User visits a familiar website and is prompted to login.  User's computer recognizes that it has a passkey for this site and starts negotiation with server.

Server sends timestamped document with a random challenge string in it and sends it back to the client for a digital signature.

# How passkey login works, part 2



Client examines document to ensure it is authentic.  If everything looks good, client applies digital signature to document and returns to server.

Server verifies the digital signature on the document using the stored public key, checks the timestamp to ensure the transaction is still time valid, and verifies the challenge string is correct and hasn’t already been used. If all checks pass, access is granted.

**passkeys**

`are replacements for`

*passwords*

*(and the baggage that comes with them)*

Questions?

# Thanks !

linkedin.com/in/aseigler/
github.com/aseigler
x.com/alexseigler