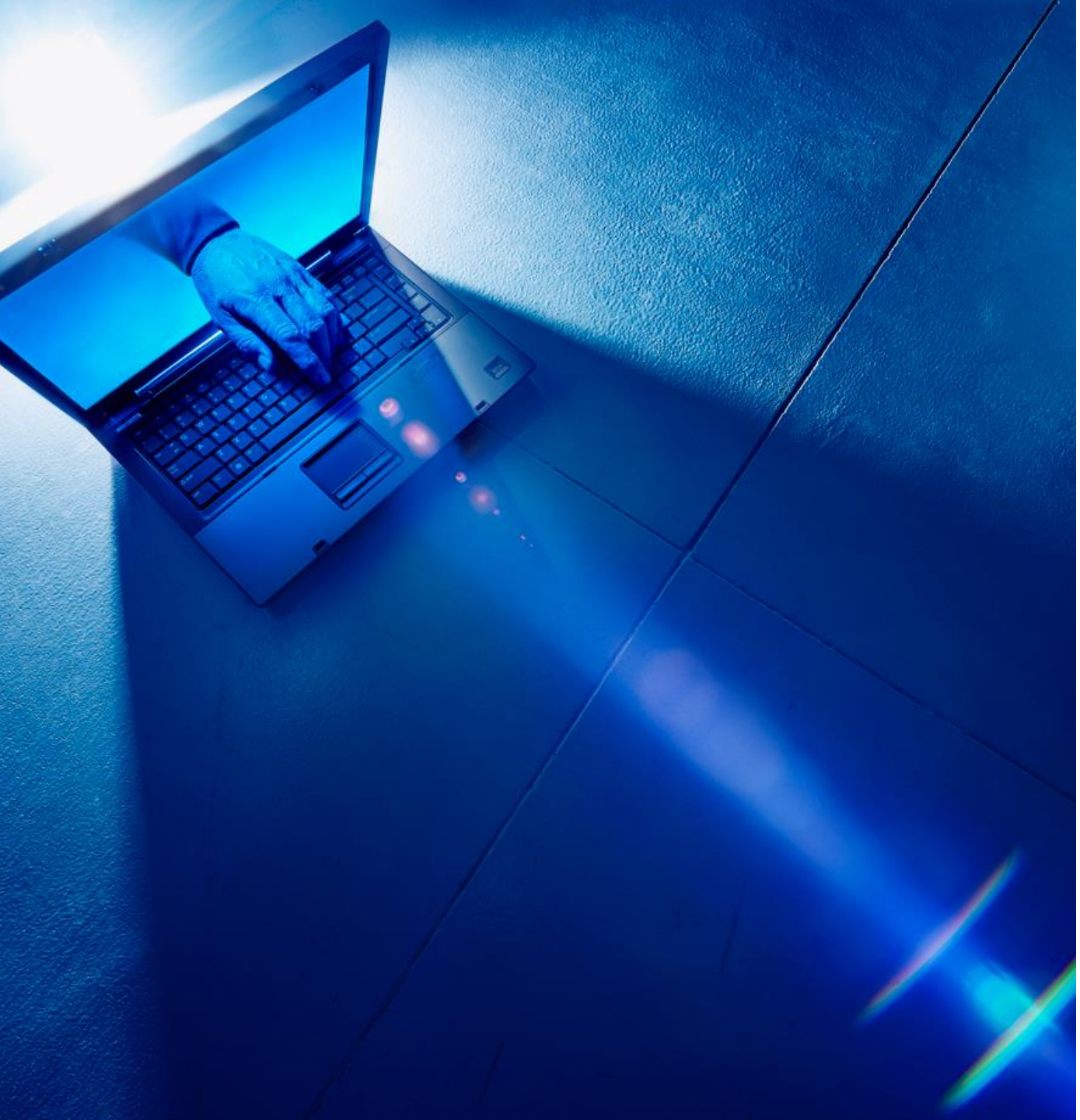
VULNERABILITIES OF ARCHITECTURES, DESIGNS, & SOLUTION ELEMENTS

Objectives

- Explore how to assess and mitigate the vulnerabilities of client-based vs. server-based systems
- Examine how to assess and mitigate the vulnerabilities of database, cryptographic, industrial control, and embedded systems
- Learn about assessing and mitigating vulnerabilities of virtualized, cloud-based, distributed, and IoT systems
- Learn how to assess and mitigate vulnerabilities of containerization, microservices, and serverless technologies
- Explore how to assess and mitigate the vulnerabilities of high-performance, edge computing, and trusted execution environments

CLIENT-BASED VULNERABILITIES

- Broken client-side access control
  - Inadequate control of JavaScript access to client-side assets (data and code), exfiltration of sensitive data, or manipulation of the Document Object Model (DOM) for malicious purposes
- DOM-based cross-site scripting (XSS)
  - Allowing XSS attacks through DOM manipulation or misuse
- Leakage of sensitive data
  - Failure to identify or mitigate digital trackers and pixels over a web property to comply with pertinent requirements



CLIENT-BASED VULNERABILITIES

- Vulnerable and outmoded elements
- Failure to detect and update JavaScript libraries that have known weaknesses
- Absence of third-party origin control
  - Origin control enables restriction of specific web assets by comparing the origin of the resource to a third-party library
  - Without these controls, there is a higher risk of supply chain risk issues due to allowing unknown or unrestrained third-party code that has access to data in the site's origin

CLIENT-BASED VULNERABILITIES

- JavaScript drift
  - This is the failure to notice changes at the asset and code level of JavaScript used client-side, including not detecting behavioral changes of the code to determine if the changes are potentially malicious
- Sensitive data stored on the client side
  - This involves storing sensitive data (passwords, secrets, application programming interface tokens, or personally identifiable information) in persistent client-side locations like LocalStorage, browser cache, or transient storage such as JavaScript variables
- Client-side security logging and monitoring failures

CLIENT-BASED VULNERABILITIES

- Not using common standards-based browser security controls
  - Examples include iframe sandboxes, security headers like Content Security Policy (CSP), sub-resource integrity, and many other standard security features
- Injecting proprietary information on the client side
  - This includes the presence of sensitive business logic, programmer comments, proprietary algorithms, or system data contained in client-side code or stored data

MITIGATING CLIENT-SIDE VULNERABILITIES

Subresource Integrity (SRI)

Enables browsers to verify that resources fetched (from a CDN) are supplied without unforeseen manipulation

Content Security Policy

Prevents XSS, clickjacking, and other code injection

Behavioral detection

Tools such as ESLint, Esprima, and Jalangi2

JavaScript analyzers

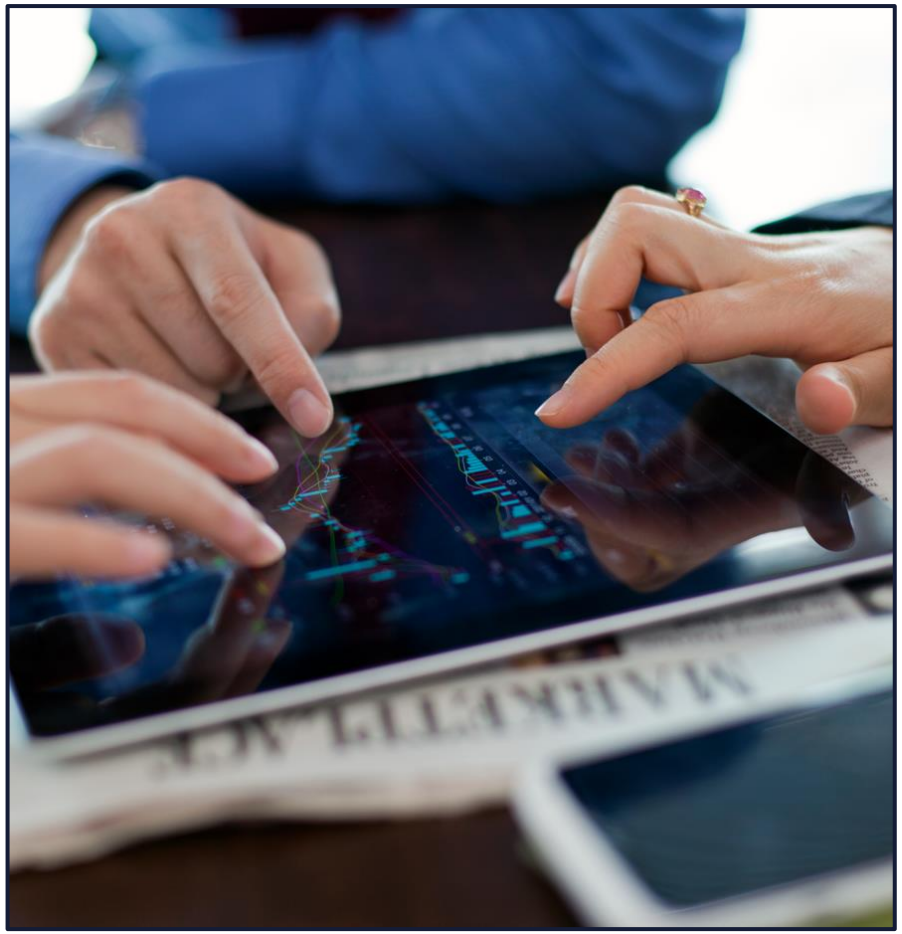
Prevents XSS, clickjacking, and other code injection

Client-side data protection and privacy enforcement

SERVER-SIDE VULNERABILITIES

- SQL injection attacks
  - Attackers exploit web application code to get access to the backend database by injecting
    SQL commands into the uniform resource identifier (URI)
- Server-side request forgery (SSRF)
  - Allows an attacker to induce the server-side application to make requests to an unintended location
  - This can give the threat actor access to internal systems, sensitive information, or other threat vectors

MITIGATING SERVER-SIDE VULNERABILITIES



- The most effective solution to resist an attempted SQL injection is to make use of a safe API that employs a parameterized interface
- Deploy web application firewalls (managed web security gateways)
- Deploy a SIEM)/SOAR system with active defense, such as a CASB for data loss prevention (DLP)
- Audit all privileged insiders and use rotation and separation of duties
- Establish a Zero Trust initiative as opposed to "trust but verify"
- Use honeytokens to discover suspected insiders

MODERN DATABASE TYPES

Relational (RDBMS like MySQL and Postgres)

NoSQL (Non-relational) like AWS DynamoDB or GCP Datastore)

Document/key-value (memory-only like Redis)

Graph (AWS Neptune)

Blockchain

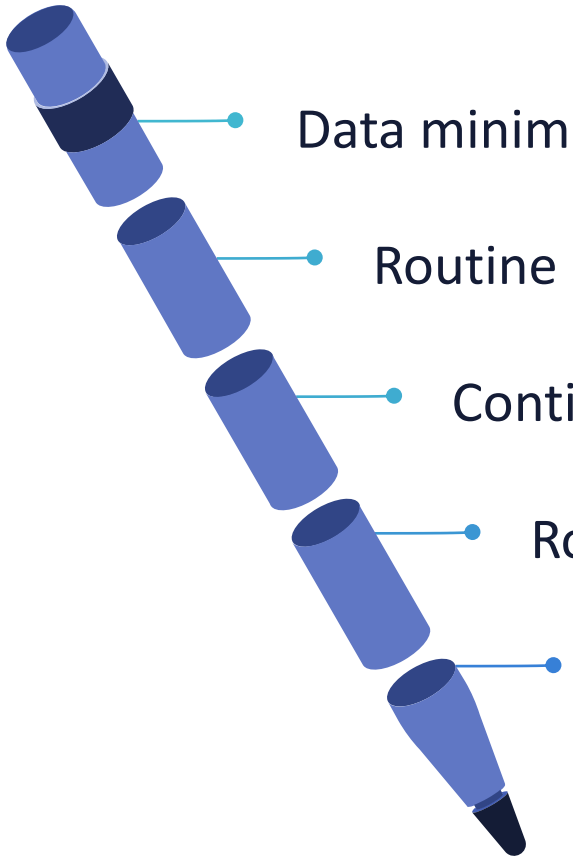
COMMON DATABASE SYSTEM VULNERABILITIES

- Deployment without due diligence and due care
  - Improper scoping and tailoring
- Allowing direct access to data as opposed to abstraction and proxy
- Failing to apply TESTED patch management
- Not testing recovery and restoration of databases on an automated and scheduled basis
- Data remanence

COMMON DATABASE SYSTEM VULNERABILITIES

- Data leaks due to lack of protection of data-in-transit
  - SSH2, IPsec IKEv2, TLS 1.2/3
- Forgetting to apply the most robust post-quantum countermeasures like AES-GCM-256
- Lack of visibility into internal privileged users
- Absence of incident response practices for data leaks/loss and stolen components
- Forgetting to protect archived data

DATABASE SECURITY COUNTERMEASURES



- Data minimization (GDPR directive)
- Routine auditing checks of activity and transaction logs
- Continuous monitoring implementation (DAM systems)
- Robust access controls (MAC, ABAC, risk-based)
- Data flow and information flow visibility (CASB with SaaS providers)

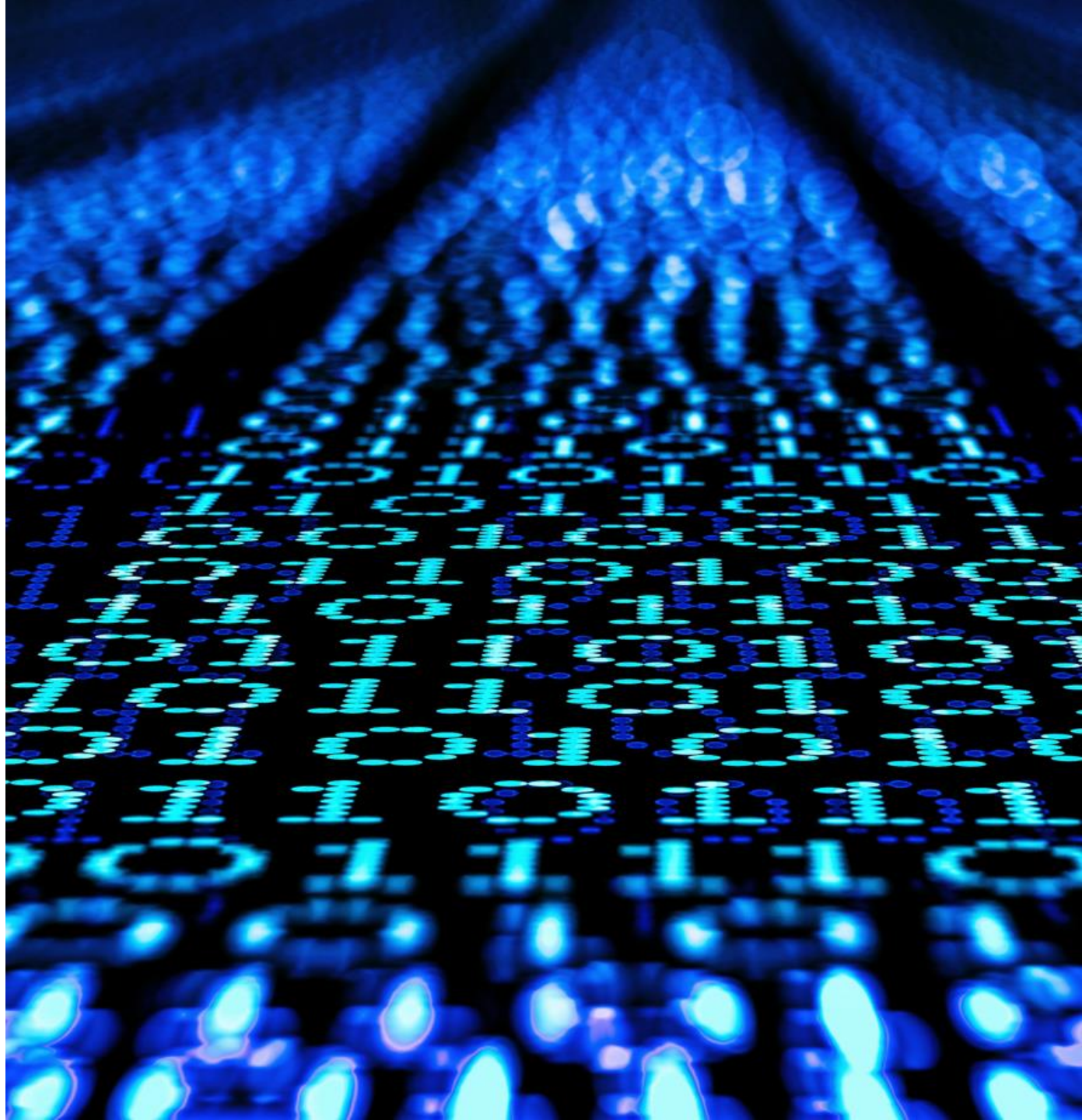
MODERN CRYPTOGRAPHIC VULNERABILITIES

- New ransomware campaign variants
- Edge device misconfiguration
- WPA3 Dragonfly

- Exploiting zero-day vulnerabilities in enterprise products
- Keys and secrets stored in Git, databases, and APIs
- Cracking post-quantum algorithms like Supersingular Isogeny

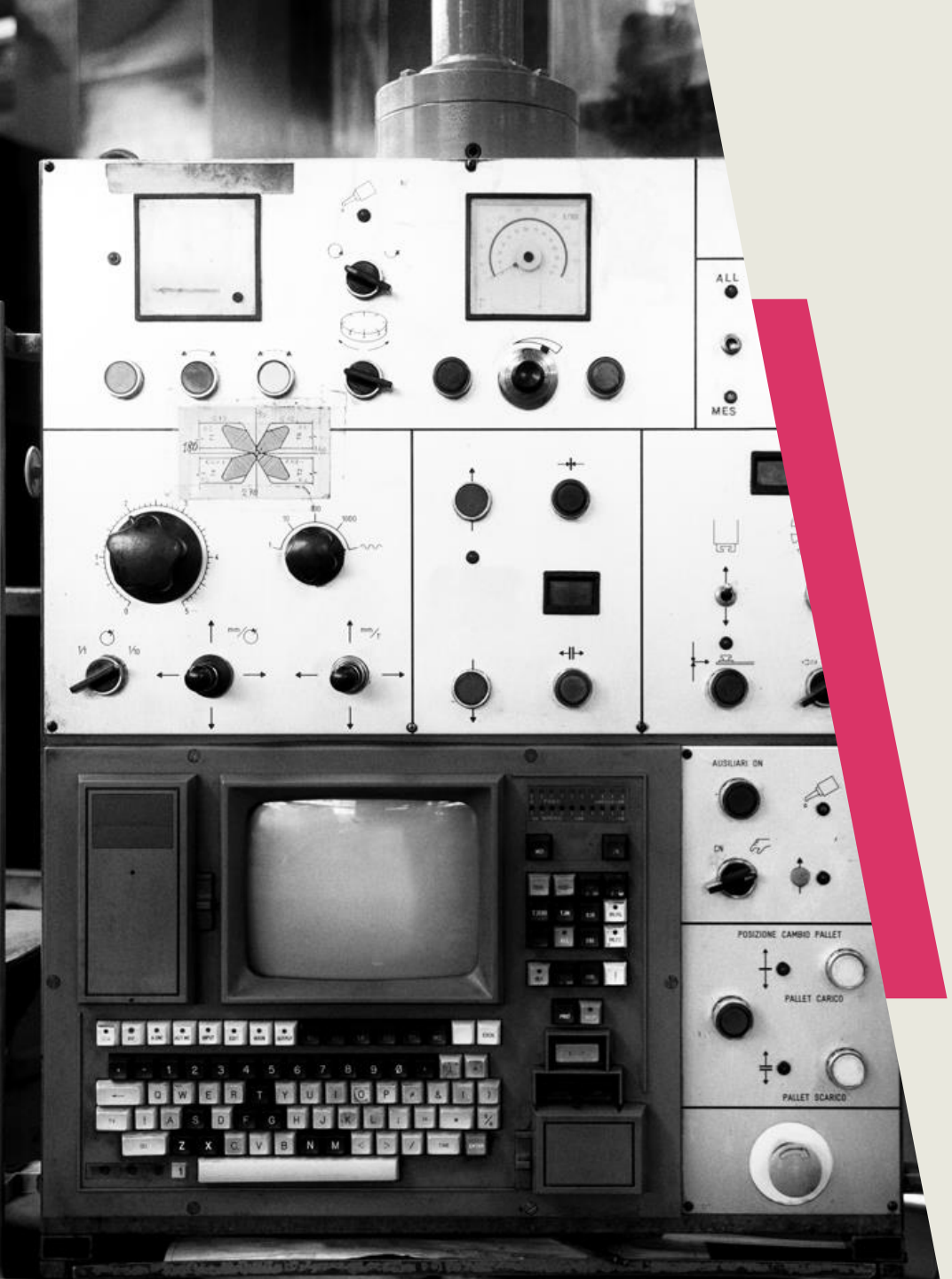
COUNTERMEASURES FOR CRYPTOGRAPHIC ATTACKS

- Ensuring that the cryptographic system is implemented properly
- Updating and testing cryptographic algorithms and protocols (not deprecated)
- Never embedding cryptographic material in repositories, databases, or in API calls
- Using trusted execution environments such as Trusted Platform Module (TPM), hardware security module (HSM), and secure enclaves



COUNTERMEASURES FOR CRYPTOGRAPHIC ATTACKS

- Tested encryption processes on data-at-rest, data-in-transit, and data-in-use
- Usage of ephemeral and temporary session keys whenever possible (forward secrecy)
- Staying up-to-date on new malware, variants, and zero-day threats



INDUSTRIAL CONTROL (ICS) AND EMBEDDED SYSTEMS

- Devices in ICS and embedded environments make a striking target for attackers due to their common absence of strong security controls and insufficient testing for vulnerabilities
- Threat agents have increased attacks targeting these components, particularly in areas like food and agriculture, chemical, water treatment, and manufacturing

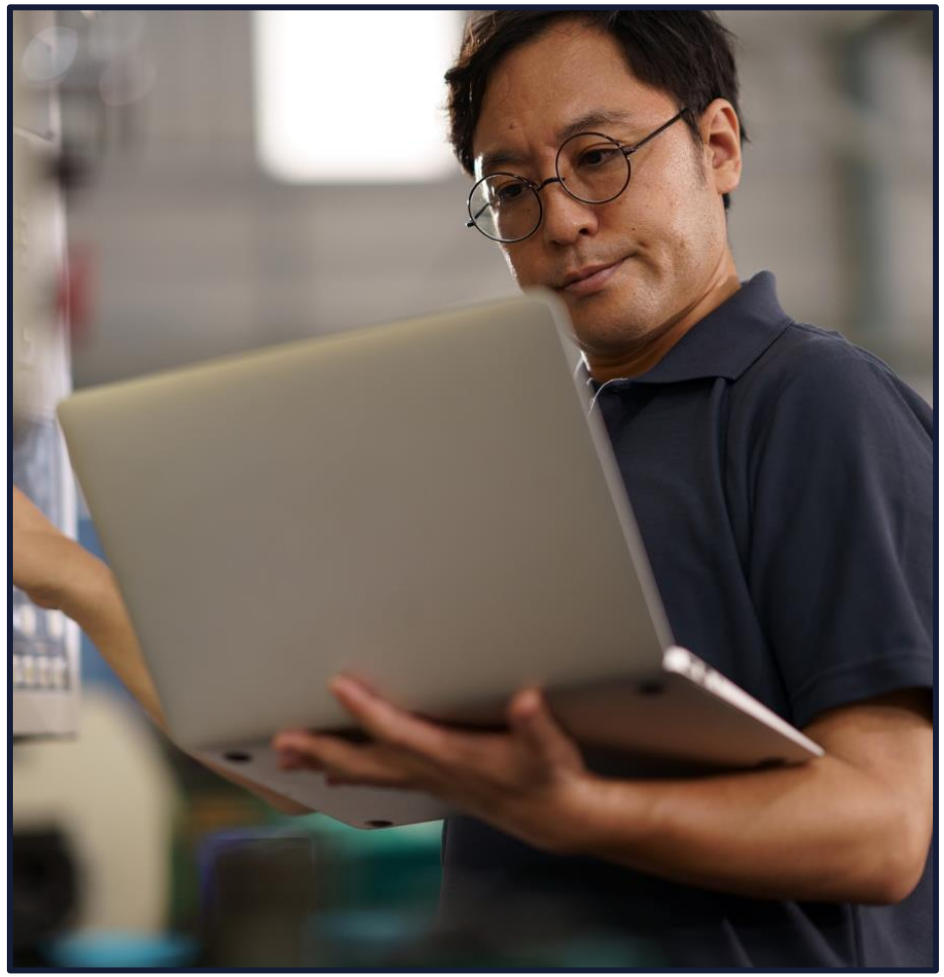
INDUSTRIAL CONTROL AND EMBEDDED SYSTEMS

- Many of the new vulnerabilities to embedded devices allow for:
  - Remote exploitation
  - Execution with low attack complexity
  - Taking control of affected systems
  - Manipulating and modifying settings
  - Escalating privileges
  - Bypassing security controls
  - Stealing data
  - Crashing systems

INDUSTRIAL CONTROL (ICS) AND EMBEDDED SYSTEMS

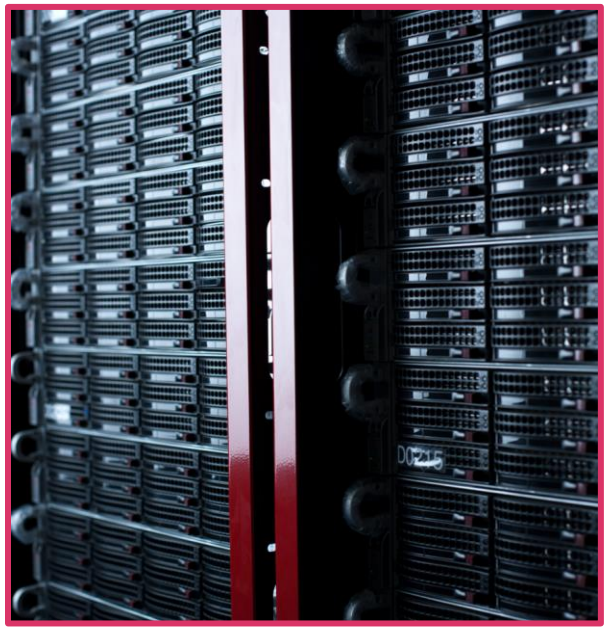
- Real-world use cases
  - Unpatched exploits in Siemens' Ruggedcom APE1808, an industry-grade application processing engine (APE) module
  - Siemens' Scalance W-700 devices, an industry-grade suite of networking and bus systems, have been widely targeted lately

COUNTERMEASURES FOR ICS ATTACKS



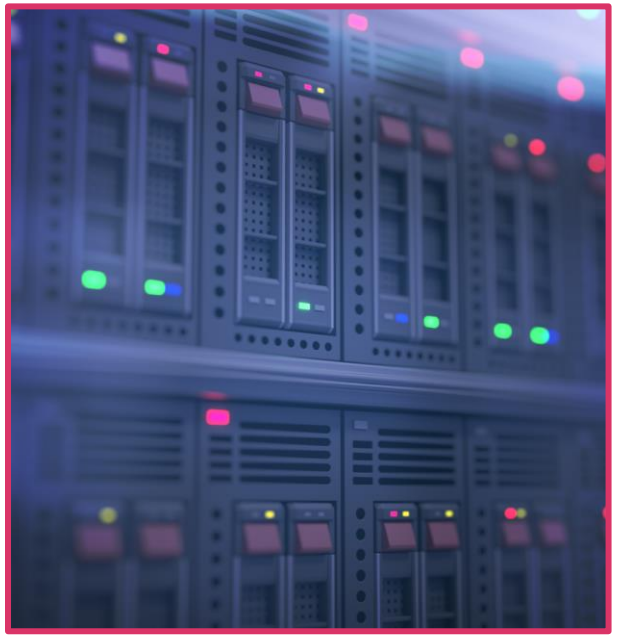
- The **EMB3D** threat model magnifies several prevailing models, including common weakness enumeration (CWE), MITRE ATT&CK, and common vulnerabilities and exposures (CVE)
- It has a precise focus on embedded devices and fosters a knowledge base of cyber threats to devices
- Advantages include observing "in the field" environments
- It uses proof-of-concept and/or theoretic research

VIRTUALIZATION RISKS



- Virtual machine (VM) sprawl and lack of management
- VM escape (hopping)
- Sensitive data in a VM
  - Passwords, personal data, bash profiles, bash history files, encryption keys, license keys, data, and image captures
- Lack of visibility into and control over virtual networks
  - Hinders existing security policy enforcement in most organizations
  - Traffic over virtual networks may not be visible to physical network security protection devices, such as network-based intrusion detection system (NIDS)/network-based intrusion prevention system (NIPS)

VIRTUALIZATION RISKS



- Security of offline and dormant VM
  - Security of pre-configured (Golden Image) VM/active VMs
- Resource exhaustion
  - Resource-intensive software can consume resources in a physical server when it is deployed in multiple VMs. For instance, antivirus and other security software interrupt every call to disk or memory in order to monitor and prevent security incidents such as cracking or viruses
- Hypervisor security
  - Tested patch management, bug bounties
- Unauthorized access to hypervisor
  - **Hyperjacking**

HYPERJACKING

- Hyperjacking defined:
  - An attack that is targeted at exploiting a system from the outside to disrupt or inject a rogue module into the system
  - The attacker can control the virtual machines running on top of the same host and monitor their network traffic
  - Normal security controls are futile since the operating system will not be aware that the machine has been compromised
- Hyperjacking countermeasures
  - Hypervisor security management must be kept separate from regular traffic
  - Tested patch management

VIRTUALIZATION RISKS

- Account or service hijacking through the self-service portal
- A self-service portal is often used to delegate specific parts of virtual infrastructure provisioning and management to assigned self-service administrators
- Generous use of self-service portals in cloud computing services will increase susceptibility to security risks, including account or service hijacking
- Workload of different trust levels located on the same server
- Risk due to cloud service provider API

VM SPRAWL COUNTERMEASURES

- Implement policies, guidelines, and processes to govern and control VM life cycle management
- Control the creation, storage, and use of VM images with a formal change management process and tools
- Retain known-good and timely patched images of a guest OS separately
- Utilize virtualization management solutions to examine, patch, and apply security configuration changes to VMs

SECURING OFFLINE/DORMANT VMS AND GOLDEN IMAGES

- Make certain appropriate hardening and protection techniques are deployed for VM instances using VM guest hardening
- Supplement VM operating systems with built-in security measures, leveraging third-party security technology like visibility and monitoring tools
- Implement integrity checksums for all VM images
- Encrypt VM images to counter unauthorized alteration
- Introduce strict controls for access, generation, and deployment of VM images/instances

THE CSA TREACHEROUS 12

- Data breaches
- Weak identity, credential, and access management
- Insecure APIs
- System and application vulnerabilities
- Account hijacking
- Malicious insiders

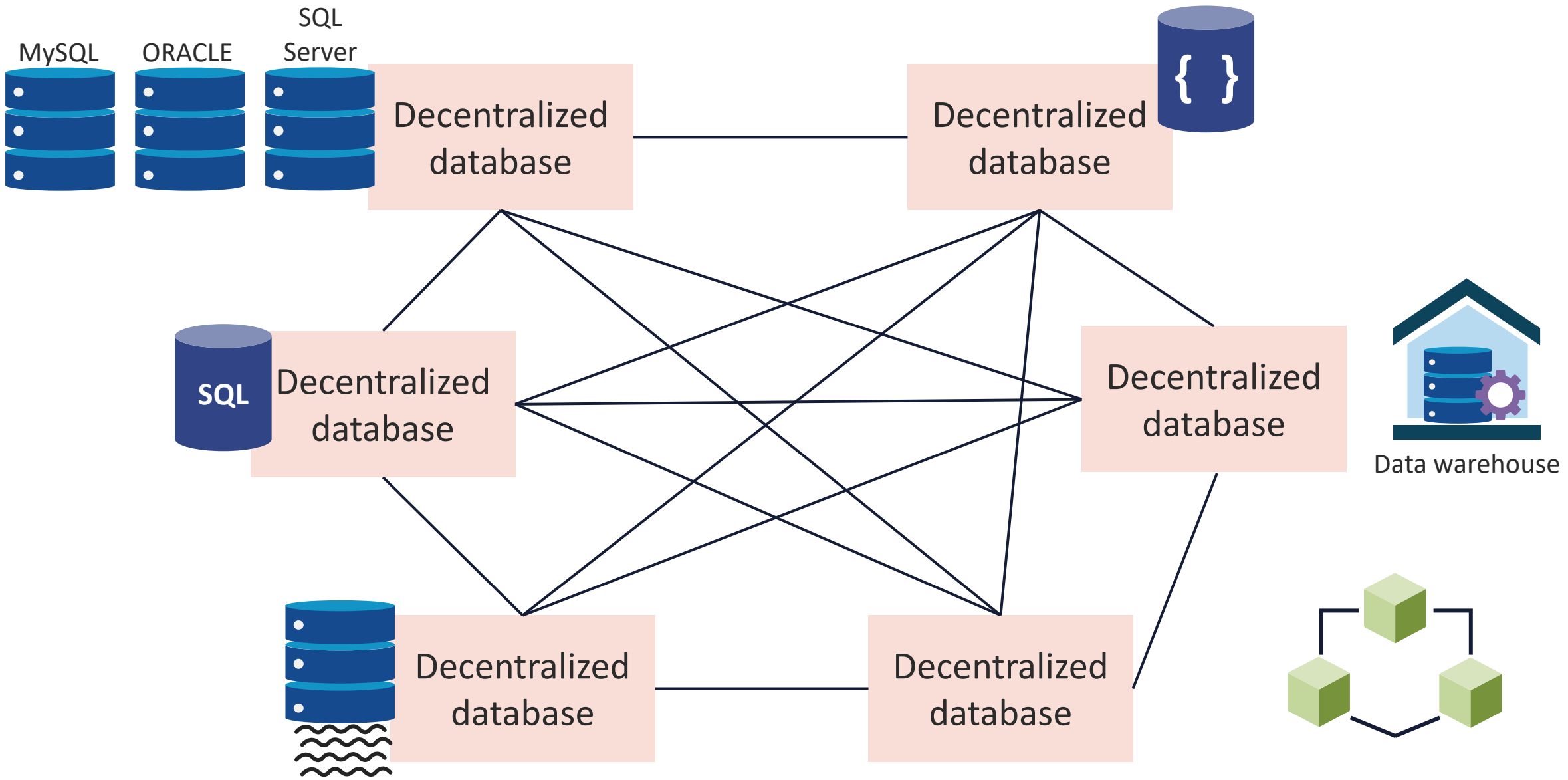
THE CSA TREACHEROUS 12

- Advanced persistent threats (APTs)
- Data loss
- Insufficient due diligence
- Abuse and nefarious use of cloud services
- Denial of service
- Shared technology issues

DISTRIBUTED SYSTEMS

- Unique attributes of distributed systems often include:
  - No centralized servers to concentrate the countermeasures
  - Resource sharing means that data often resides in multiple locations accessed by more than one user, app, or system
  - Concurrency leads to processes and tasks running simultaneously yet not in exact timing
  - Scalability is an enabler for distributed denial-of-service (DoS) attacks
  - Fault tolerance is a benefit for resilience but adds additional targets

DISTRIBUTED DECENTRALIZED DATABASES



DISTRIBUTED SYSTEM WEAKNESSES

- A distributed system needs more security measures than centralized systems, as there are many users, differentiated data, multiple sites, and distributed control
- Security engineers must consider many permutations of failures and errors that can happen at any time, independently or in combination with other error conditions

DISTRIBUTED SYSTEM EXPLOITS

- Passive eavesdroppers that monitor messages and collect private information – information leakage
- Active attackers that not only eavesdrop but further corrupt messages by inserting new data or modifying existing data
- Distributed denial-of-service (DDoS) and botnets
- Unauthorized access through poor access controls

HARDENING DISTRIBUTED SYSTEMS

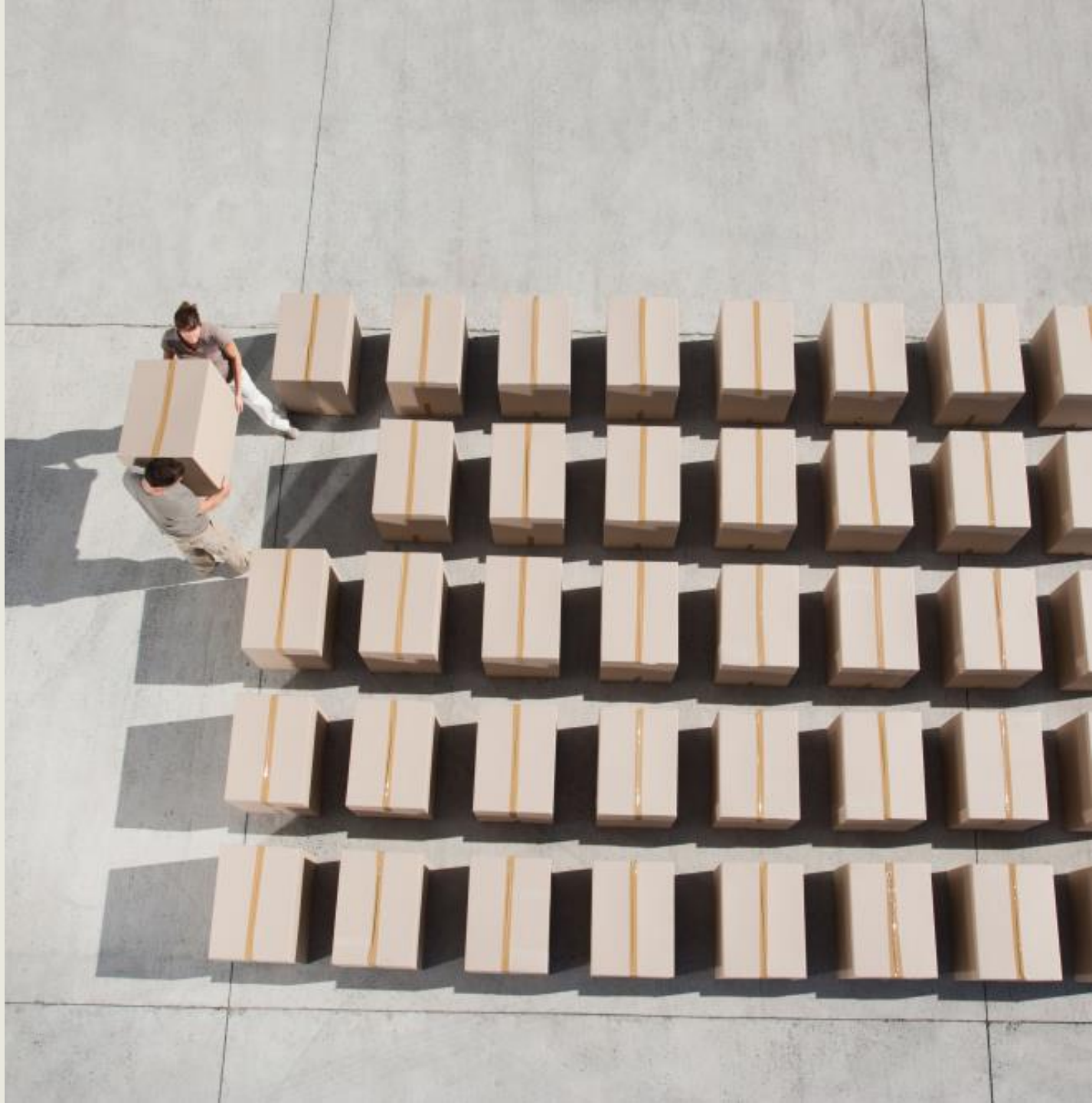
- Communications must be secured with IPsec or Transport Layer Security (TLS) certificates
- Data at rest is authenticated, authorized, validated, and encrypted with strong algorithms and modes (AES-GCM-256)
- Layer 2 can be secured (i.e., SANs and distributed databases) with MACsec
- Cloud-based SIEM/SOAR for hybrid cloud and edge computing environments using:
  - Multicloud
  - Peer-to-peer file sharing
  - Data dispersion

CONTAINERS

- A container is a discrete environment within an operating system (or a serverless architecture) where one or more applications can run and that is typically assigned all the resources and dependencies needed to function
- It is a modular and portable environment that includes the application binaries, software dependencies, and hardware requirements wrapped up into an independent, self-contained unit

MICROSERVICES

- Microservices are specific service-oriented application components made up of small independent services that communicate over well-defined APIs for notification and process queueing
- They make applications and apps faster to develop and easier to scale by small, self-contained teams of developers
  - **Microservices are about the design of software**
  - **Containers are about packaging software for deployment**



CONTAINER DEVELOPMENT VULNERABILITIES

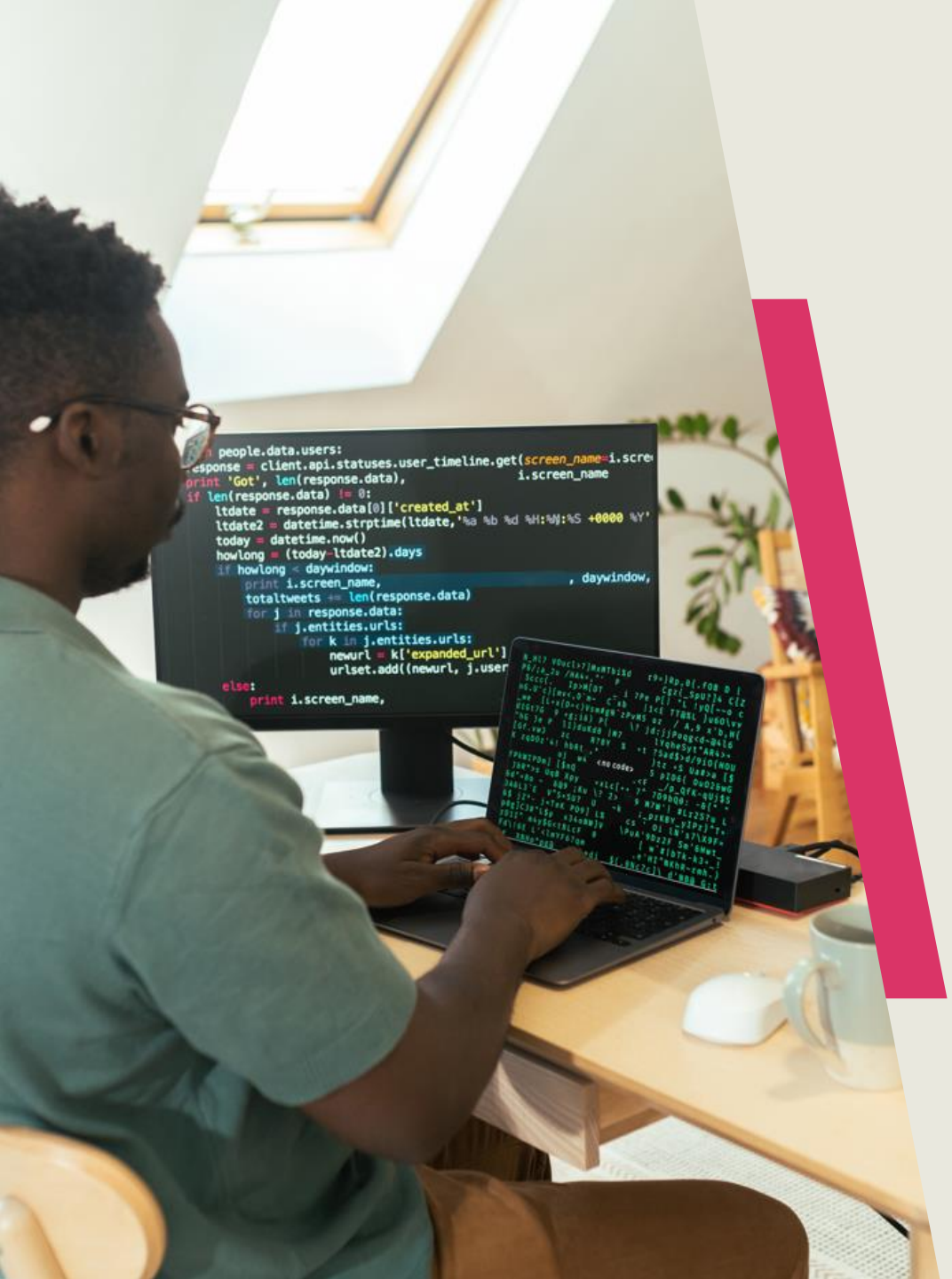
- Unsecure images
  - Containers or microservices that have legacy code with known vulnerabilities
  - They derive from registries that are known to be suspect or have not been verified
- Container breakout (escape)
  - When malicious code or attackers leak from their confined container into the host operating system or hypervisor

CONTAINER DEVELOPMENT VULNERABILITIES

- Poisoned orchestration API
  - Unsecure Kubernetes (K8) environments leading to unauthorized access, data theft, or worse
- Denial-of-service
  - A particular container is inundated with too many requests, forcing it to buckle
  - It could possibly cascade into negatively affecting communication with other containers or microservices

CONTAINER DEVELOPMENT VULNERABILITIES

- Untrusted images
  - Unreliable images can carry malicious code or hidden backdoors
- Lack of resource limiting
  - A poorly written container consumes too many processor, memory, or network resources
  - This could trigger application or system instability as other microservices on the same host system get starved



CONTAINER DEVELOPMENT VULNERABILITIES

- Misconfigured access controls
  - Failing to implement due diligence with least privilege and segregation of duties principles
- Insufficient isolation
  - At their core, containers, and microservices are designed to provide a certain degree of isolation – from the host system and other containers
  - Without solid configuration, this isolation shield issue can become problematic

CONTAINER SECURITY COUNTERMEASURES

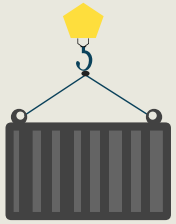
- Introduce a **DevSecOps** environment
- Initiate software assurance like the OWASP Software Assurance Maturity Model (**SAMM**)
- Employ Architecture, Threats, Attack Surfaces, and Mitigations (**ATASM**)
  - A newer threat-modeling approach that focuses on the structural understanding of containers
- Deploy a **robust solution** with the following features:
  - Security scanning and monitoring
  - Configuration defect detection and misconfiguration management
  - Embedded secrets and vulnerability detection

SERVERLESS TECHNOLOGIES

- Modern serverless solutions leverage modern cloud infrastructures to emulate the network operating system (NOS) environment without the need for a Windows or Linux-based server
- These run code, manage data, and integrate applications, all without underlying servers
- They feature automatic scaling, built-in high availability, and a pay-for-use billing model to increase agility and optimize costs
  - Functions, containers, and databases are common serverless solutions

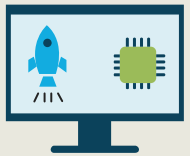
CASE STUDY: AWS FARGATE SERVERLESS CONTAINERS

Without Fargate



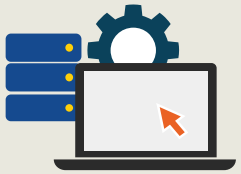
Build your container image

Define and deploy the EC2 Instances



Provision and manage compute and memory resources

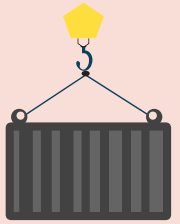
Isolate application in separate virtual machines



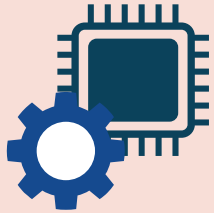
Define and deploy the EC2 instances

Pay for EC2 Instances

With Fargate

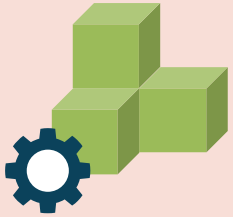


Build container image

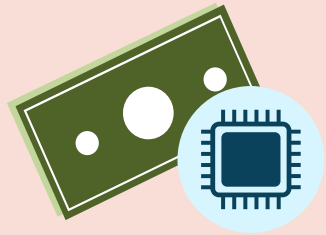


Define memory and compute resources required

AWS Fargate



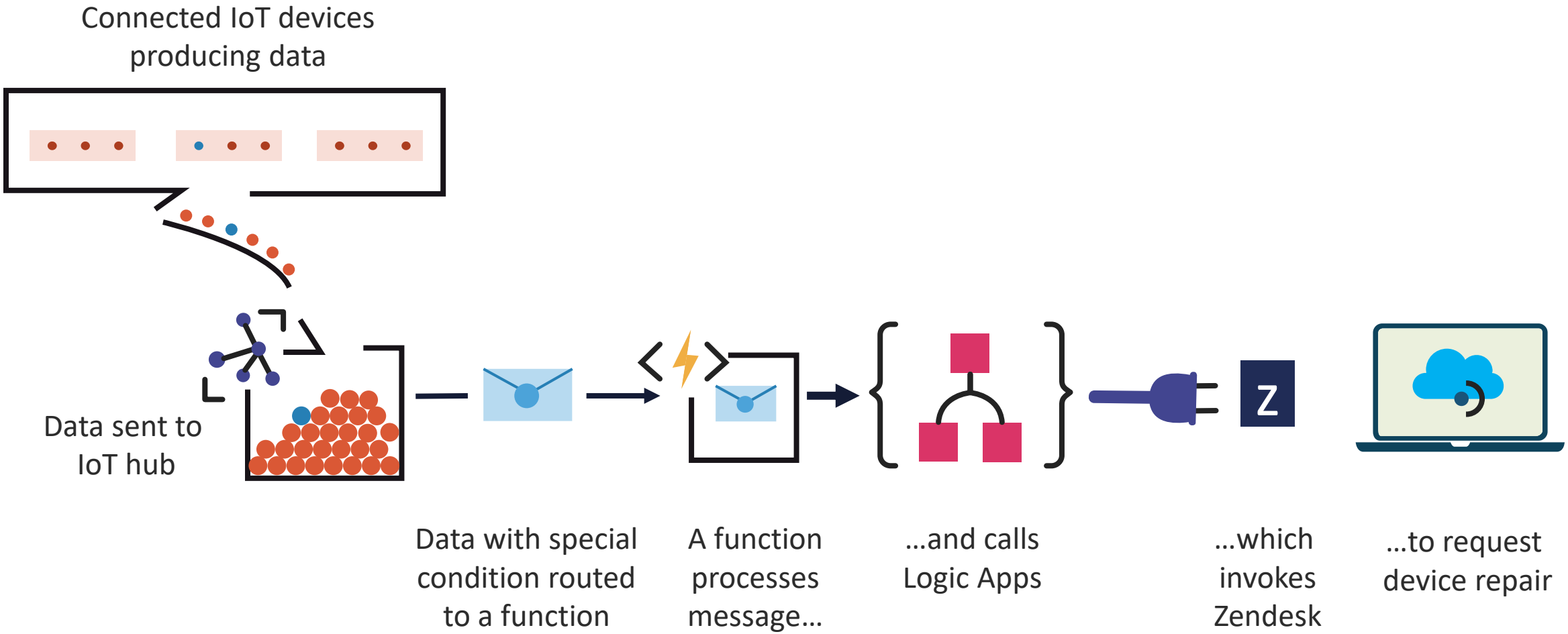
Run and manage applications



Pay for requested compute resources when used

Application isolation by design

CASE STUDY: AZURE SERVERLESS FUNCTIONS

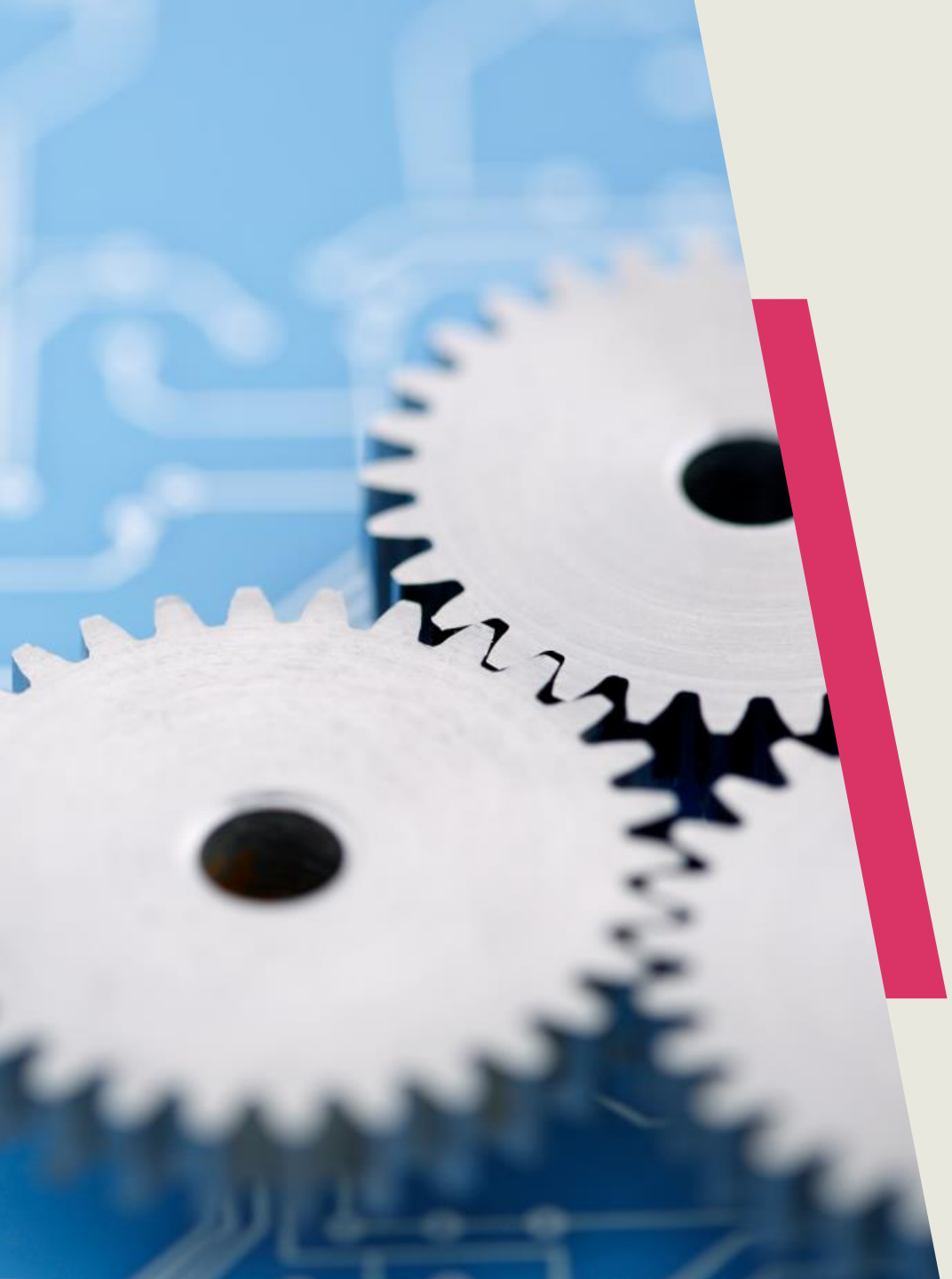


SERVERLESS VULNERABILITIES

- Lack of visibility and control of the cloud computing environment
- Absence of source code security
- Using obsolete or unpatched libraries, packages, and dependencies
- Forgetting to integrate the newest cryptographic protections into the code
- Too much information being disclosed in error messages

HIGH-PERFORMANCE COMPUTING (HPC)

- An HPC cluster is a set of independent computing systems, called compute nodes, which are interconnected via high-speed networks (100 Gbps)
- An example of large-scale HPC would be a Dell solution with 9,216 CPU cores, and 1.28PB Power Scale Network File Storage (NFS)
  - These are usually purchased on a 1/3/5-year subscription from various vendors such as Dell or HP



HIGH-PERFORMANCE COMPUTING

- If customers decide to "rent" an HPC solution instead, they may use a CSP like AWS
- Amazon Elastic Compute Cloud (EC2) Hpc6a instances offer 3rd Gen AMD EPYC processors
- These are built for tightly coupled, compute-intensive workloads such as:
  - Computational fluid dynamics (CFD)
  - Weather forecasting
  - Multiphysics* simulations
- Elastic Fabric Adapter (EFA), powered by AWS Nitro, delivers low-latency 100 Gbps network bandwidth for internode communication

NIST SP 800-223

- **High-Performance Computing Security: Architecture, Threat Analysis, and Security Posture**
- According to NIST: "An HPC system is designed to maximize performance, so its architecture, hardware components, software stacks, and working environment are very different from enterprise IT.

  Additionally, most HPC systems serve multiple projects and user communities, leading to more complex security concerns than those encountered in enterprise systems. As such, security solutions must be tailored to the HPC system's requirements."

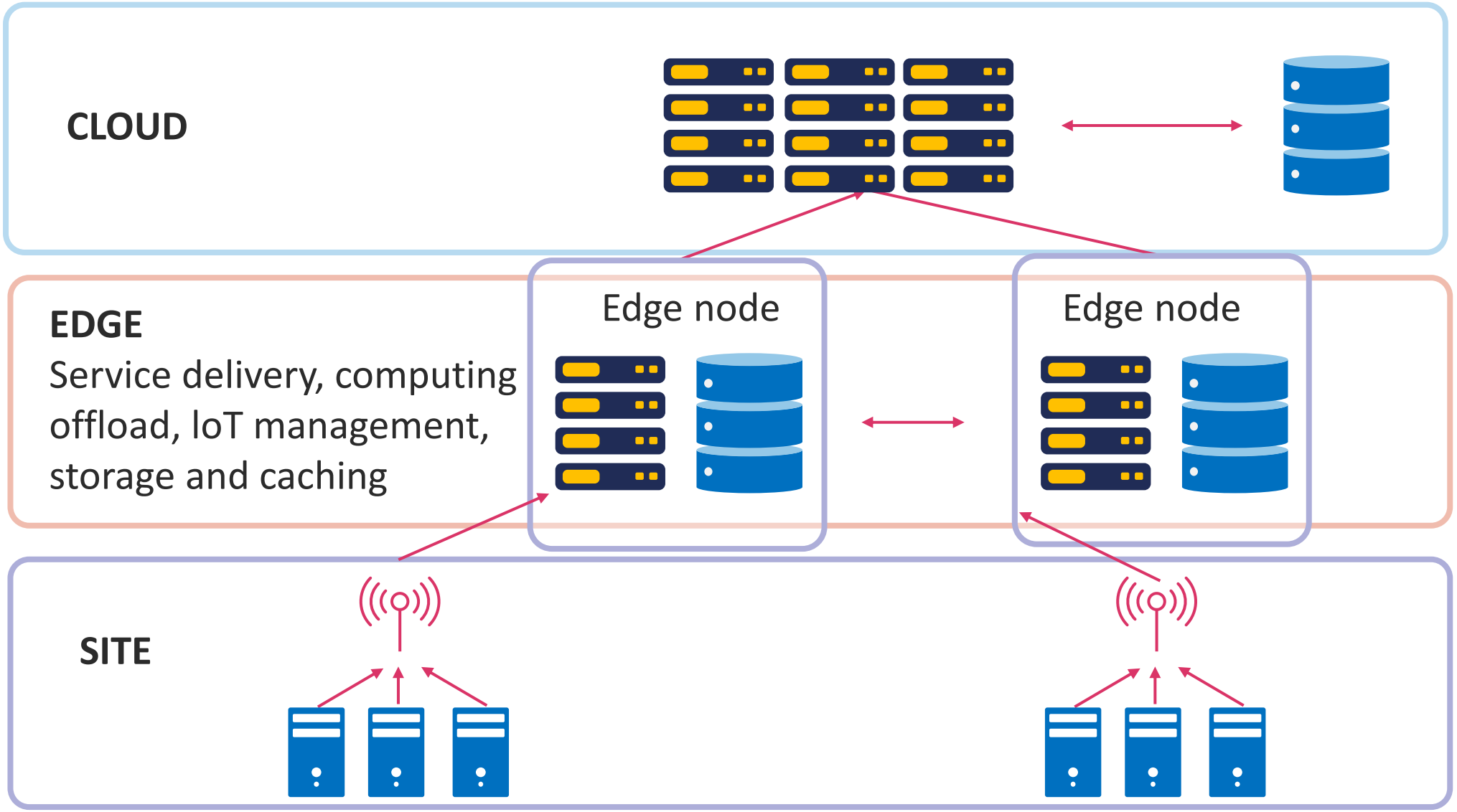
EDGE COMPUTING (FOG)

- Edge computing is a distributed computing standard that brings compute services and data storage near to the customer location where it is
- The goal is to speed up response times and preserve bandwidth (**hybrid cloud and local zones**)
- Content delivery network (CDN) solutions place cached versions of content (often in elastic Redis in-memory storage clusters) at metro area edge locations

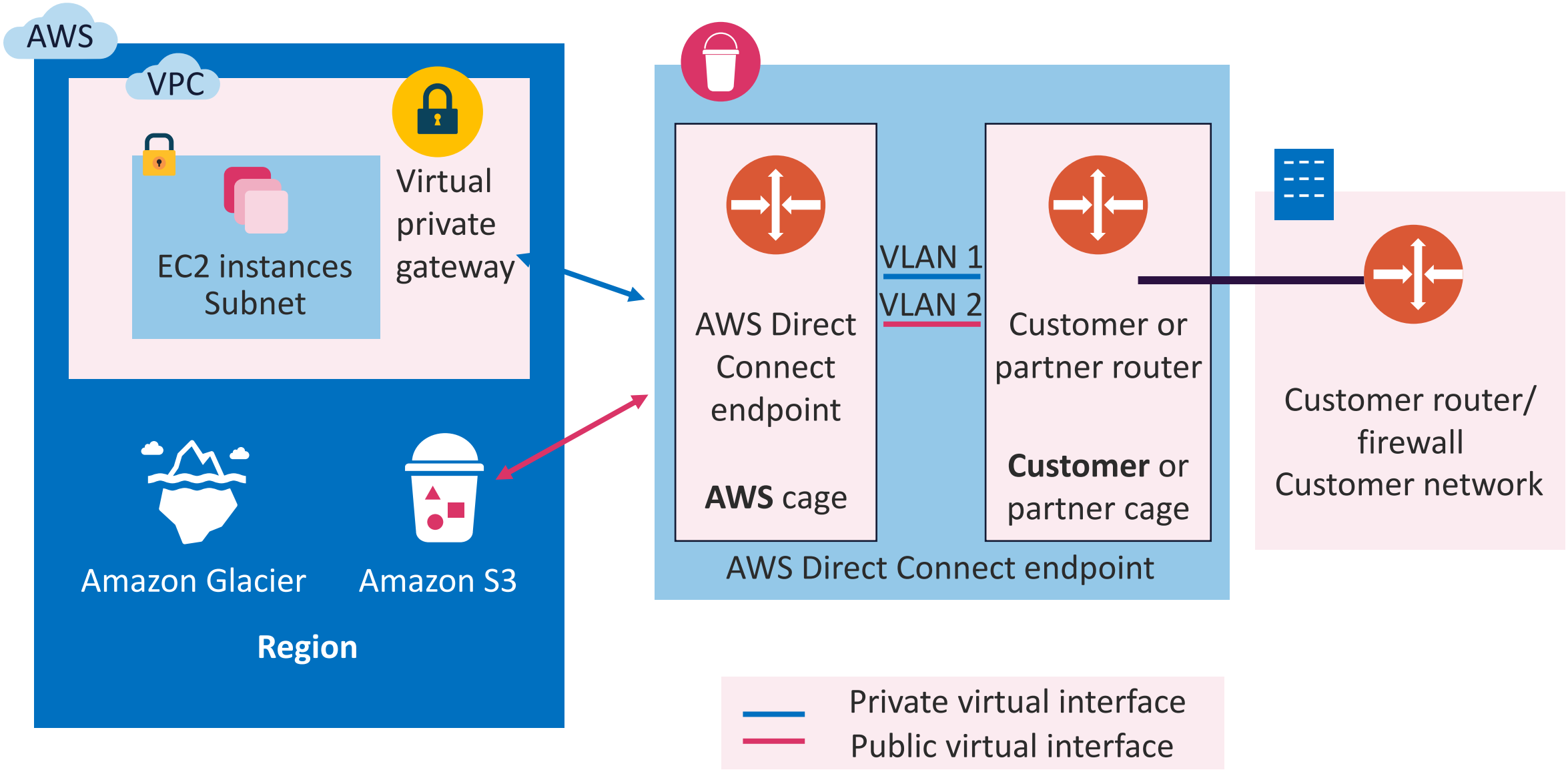
EDGE COMPUTING (FOG)

- Security is a shared responsibility model between customers, internet service providers (ISPs), and CDN providers
- Edge computing can also be supported by using direct solutions like AWS Direct Connect, Azure ExpressRoute, and Google Interconnect (10/50/100 Gbps with MACsec)

EDGE COMPUTING

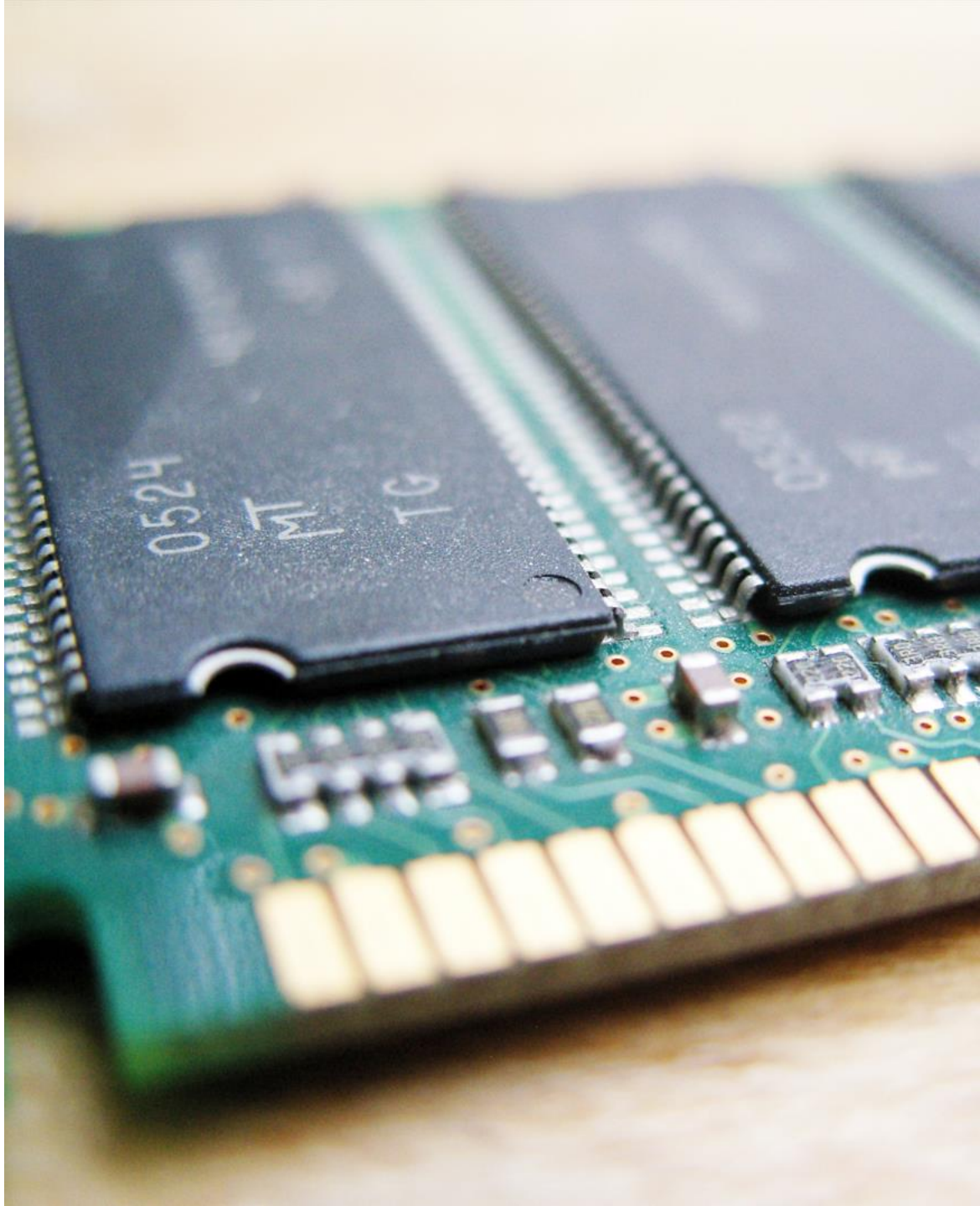


DIRECT CONNECT SUPPORTS EDGE (FOG) COMPUTING



TRUSTED EXECUTION ENVIRONMENTS (TEE) – MEMORY PROTECTION

- In a nutshell, Trusted Execution Environments (TEEs) are CPU-encrypted isolated private enclaves inside the memory
- The goal is to protect data-in-use at the hardware level
- Unauthorized subjects cannot remove, change, or add more data while the sensitive data is inside an enclave
- A TEE can greatly boost mobile (iOS SEP) and cloud data security by isolating sensitive functions and providing a secure environment for analyzing data



TRUSTED PLATFORM MODULE (TPM)

- A Trusted Platform Module 2.0 (TPM 2.0) is a hardware component that generates and stores cryptographic keys, secrets, and certificates
  - It enhances the system security by preventing unauthorized access and tampering (confidentiality and integrity)
- To use a TPM, a software library that communicates with the device and a user-friendly API is needed

TRUSTED PLATFORM MODULE 2.0 TYPES

- **Discrete** TPMs are the most secure dedicated chips that deliver TPM capabilities in their own tamper-resistant semiconductor package
- **Integrated** TPMs are part of another chip
- **Firmware** TPMs (fTPMs) are firmware-based UEFI solutions that run in a CPU's TEE
- **Virtual** TPMs (vTPMs) are hypervisor solutions in isolated execution environments
- **Software** TPMs are software emulators of TPMs that run in an operating system and are the least secure option

TPM 2.0 ENCRYPTION AND DECRYPTION

- The **PC Client Platform TPM Profile (PTP) Specification** requires:
  - SHA-1 and SHA-256 for hashes
  - RSA, ECC using the NIST P-256 curve for public-key cryptography and asymmetric digital signature generation and verification
  - HMAC for symmetric digital signature generation and verification
  - 128-bit AES for symmetric-key algorithm
  - ECC-based 256-bit curve is optional for the TCG PC Client Platform PTP Specification
- **Homomorphic encryption** is being introduced in cloud computing environments

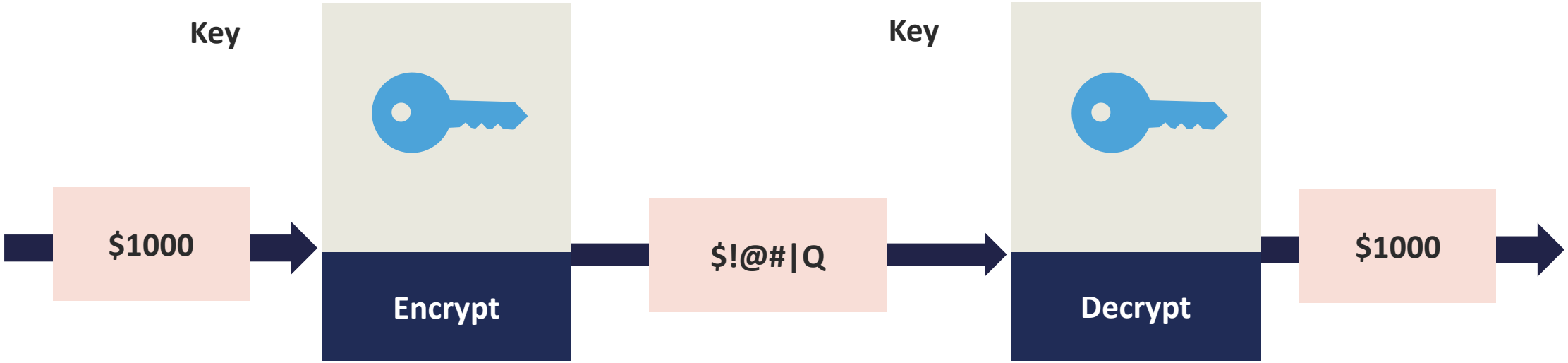
CRYPTOGRAPHIC SOLUTIONS & CRYPTANALYTIC ATTACKS

Objectives

- Learn about cryptographic methods and life cycle
- Describe key management, digital signatures, certificates, and public key infrastructure
- Explain brute force, implementation, side-channel, an on-path attacks against cryptosystems
- Learn about Kerberos exploitation
- Understand ransomware attacks

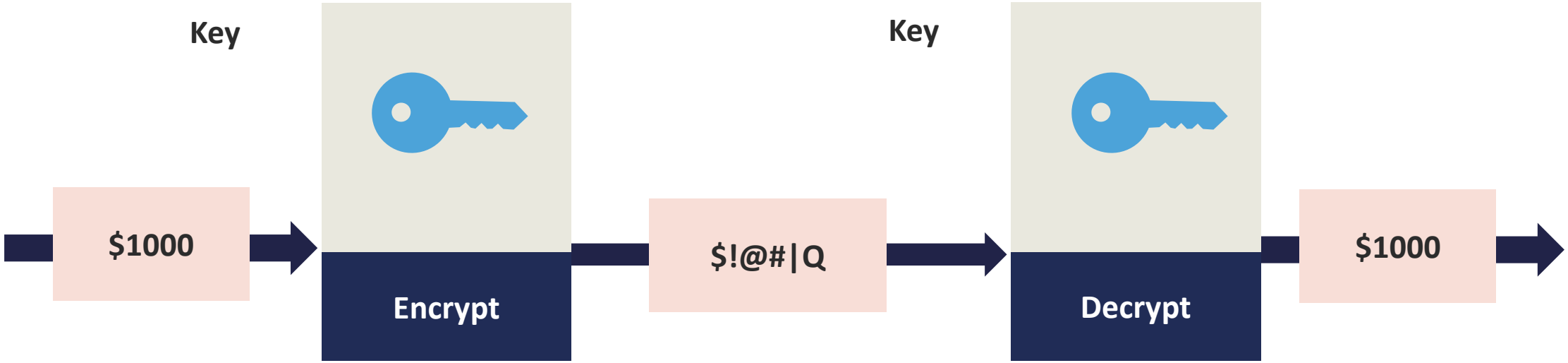
SYMMETRIC KEY CRYPTOGRAPHY

- The same secret key is used for encryption and decryption
- The secret key must be shared between sender and receiver securely
- Strength is related to management, size of keys, and how they are shared
- Keys are typically from 40 to 512 bits in length
- Longer keys are less susceptible to a successful brute force attack



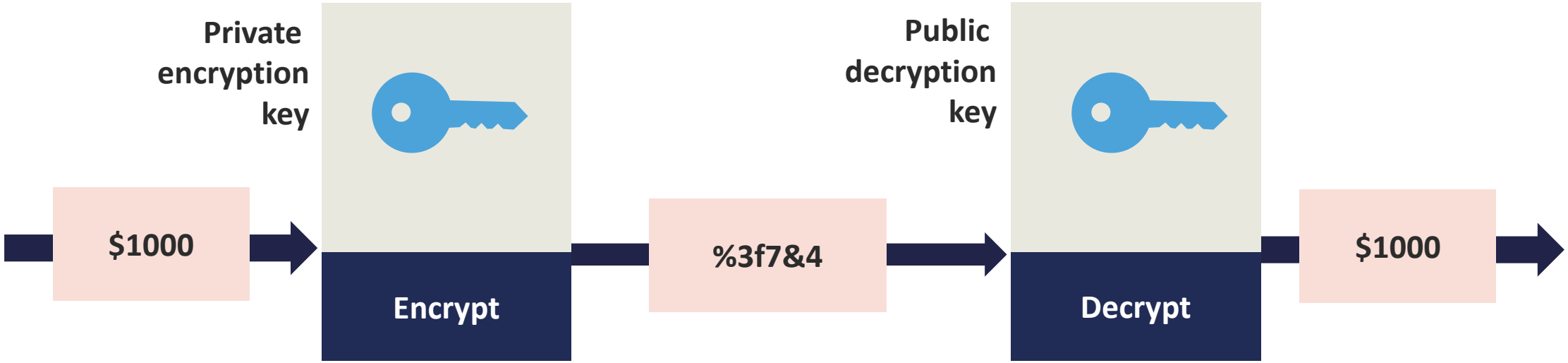
SYMMETRIC KEY CRYPTOGRAPHY

- Symmetric algorithms deliver wire-speed encryption
- Commonly used for bulk data encryption (e.g., VPNs, data at rest, CMKs at CSPs)
- They utilize strong confusion and diffusion techniques
- Algorithms can be accelerated by hardware security module (HSM)
- Can use both stream and block ciphers (block are most common)



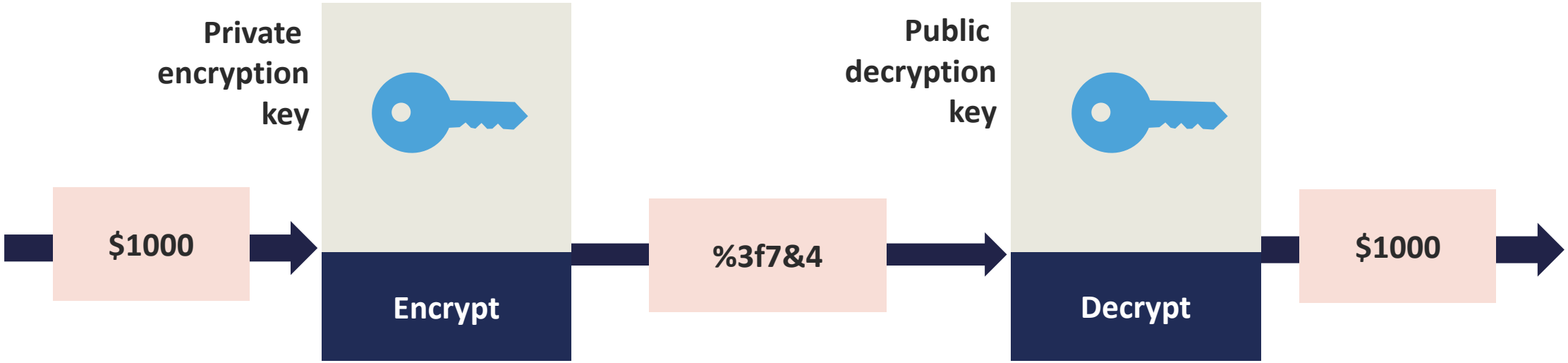
ASYMMETRIC KEY CRYPTOGRAPHY

- Different keys are used for encryption and decryption
- They are generated together and are mathematically related
- The public key is shared with multiple parties
- The private key must be kept private and confidential by the owner or custodian
- Keyspace ranges from 512 to 4,096 bits in length



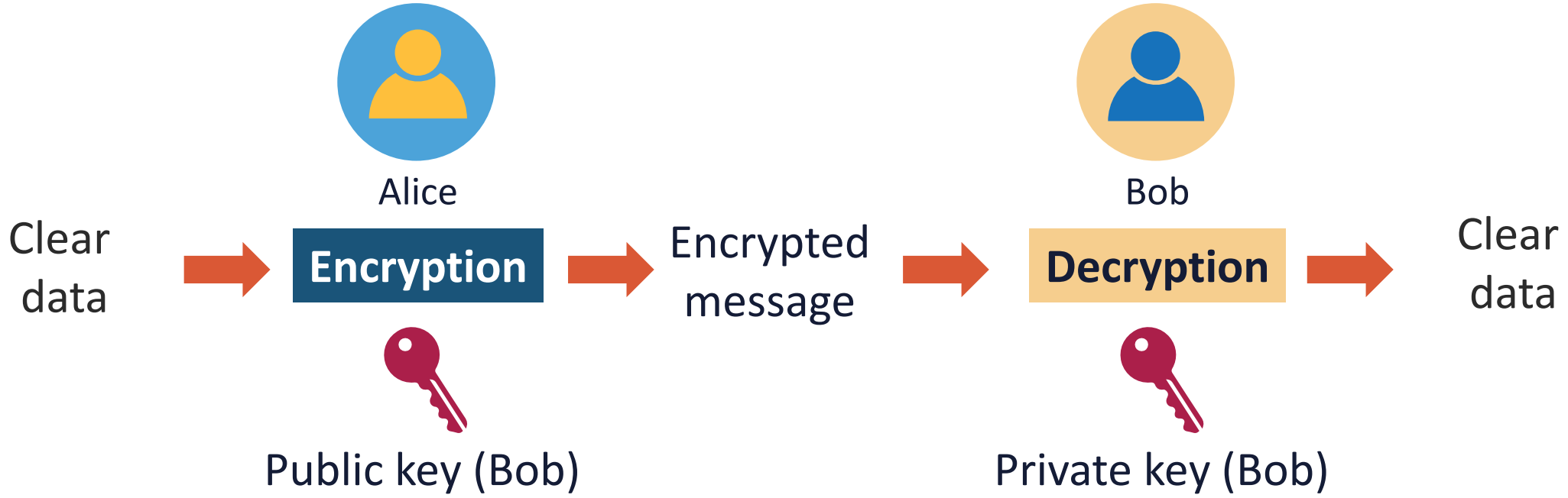
ASYMMETRIC KEY CRYPTOGRAPHY

- Asymmetric cryptosystems are slower and not suitable for bulk data operations
- The design is based on factoring the product of large prime numbers
- Key management is simpler and more secure (PKI and HSM)
- Best suited for digital signatures and session key exchange or agreement protection services (RSA, DSA, Elliptic curve DSA, PGP/GPG, Diffie-Hellman)



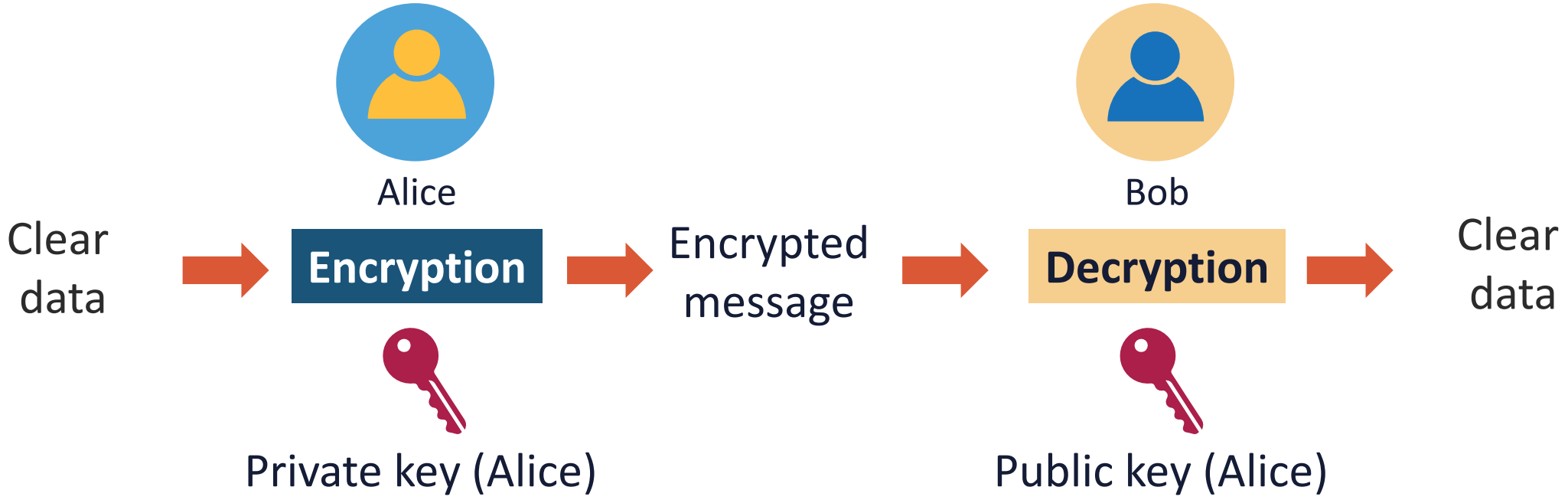
ASYMMETRIC CRYPTOSYSTEMS FOR PRIVACY

Privacy (confidentiality) = Alice encrypts with Bob's public key, then Bob decrypts with his private key



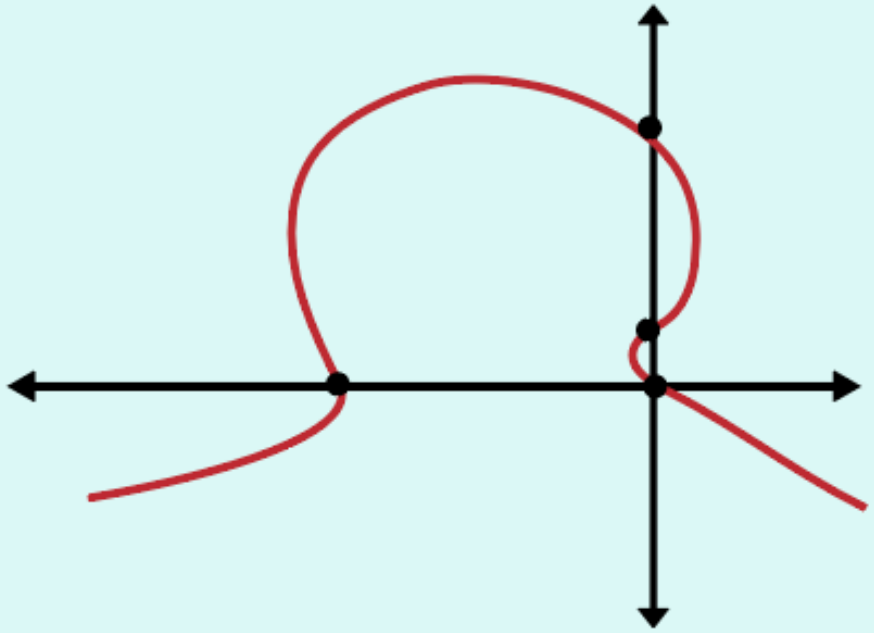
ASYMMETRIC CRYPTOSYSTEMS FOR AUTHENTICATION

Origin authentication = Alice encrypts with her private key, then Bob decrypts with Alice's public key



ELLIPTIC CURVE CRYPTOGRAPHY

- Elliptic curve cryptography (ECC) involves rich mathematical functions based on points on a curve
- The algorithm computes discrete logarithms of elliptic curves, which is different from calculating discrete logarithms in a finite field
  - The smaller and more efficient keys offer exceptional speed and strength (e.g., a 256 elliptic curve key is equivalent to a 3,072 normal key)
- It is commonly used in digital signatures, key distribution, and encryption
- It is an excellent choice for mobile and Internet of Things (IoT) devices



ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM (ECDSA)

- It is a cryptographically secure digital signature system based on elliptic curve cryptography
  - It leverages the math of the cyclic groups of elliptic curves over finite fields and on the difficulty of the elliptic-curve discrete logarithm problem
- It is popular in the field of digital signatures built on the complexity of calculating a private key from which a public key is based
  - ECDSA is also used by Bitcoin to assure that cryptocurrency can only be used by the lawful owner(s)

ELLIPTIC CURVE DIFFIE–HELLMAN (ECDH)

- ECDH is a key agreement protocol that enables two entities, each having an elliptic curve public–private key pair, to generate a shared secret over an unsecure channel like the public Internet
- This shared secret may be directly used as a key or to derive another ephemeral key
- The key (or derived key) can then be used to encrypt successive communications using a symmetric-key cipher

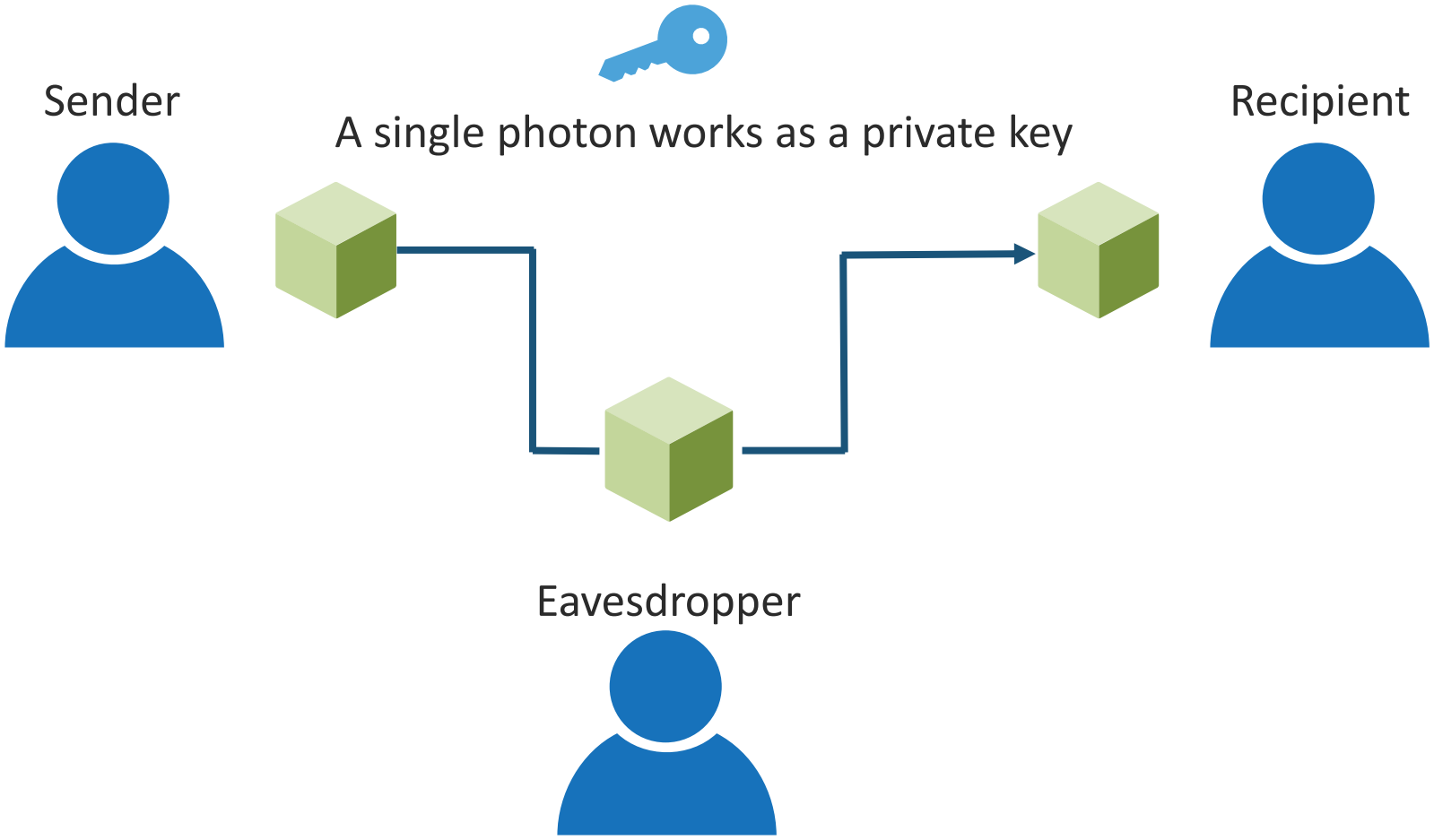
QUANTUM CRYPTOGRAPHY

- Personal computers use bits (1s or 0s) whereas quantum computers use qubits
- These are typically subatomic particles, such as photons typically (or electrons)
- Quantum computing derives its power from the fact that qubits can represent numerous possible combinations of 1 and 0 at the same time
- This ability to simultaneously be in multiple states is called superposition

QUANTUM CRYPTOGRAPHY

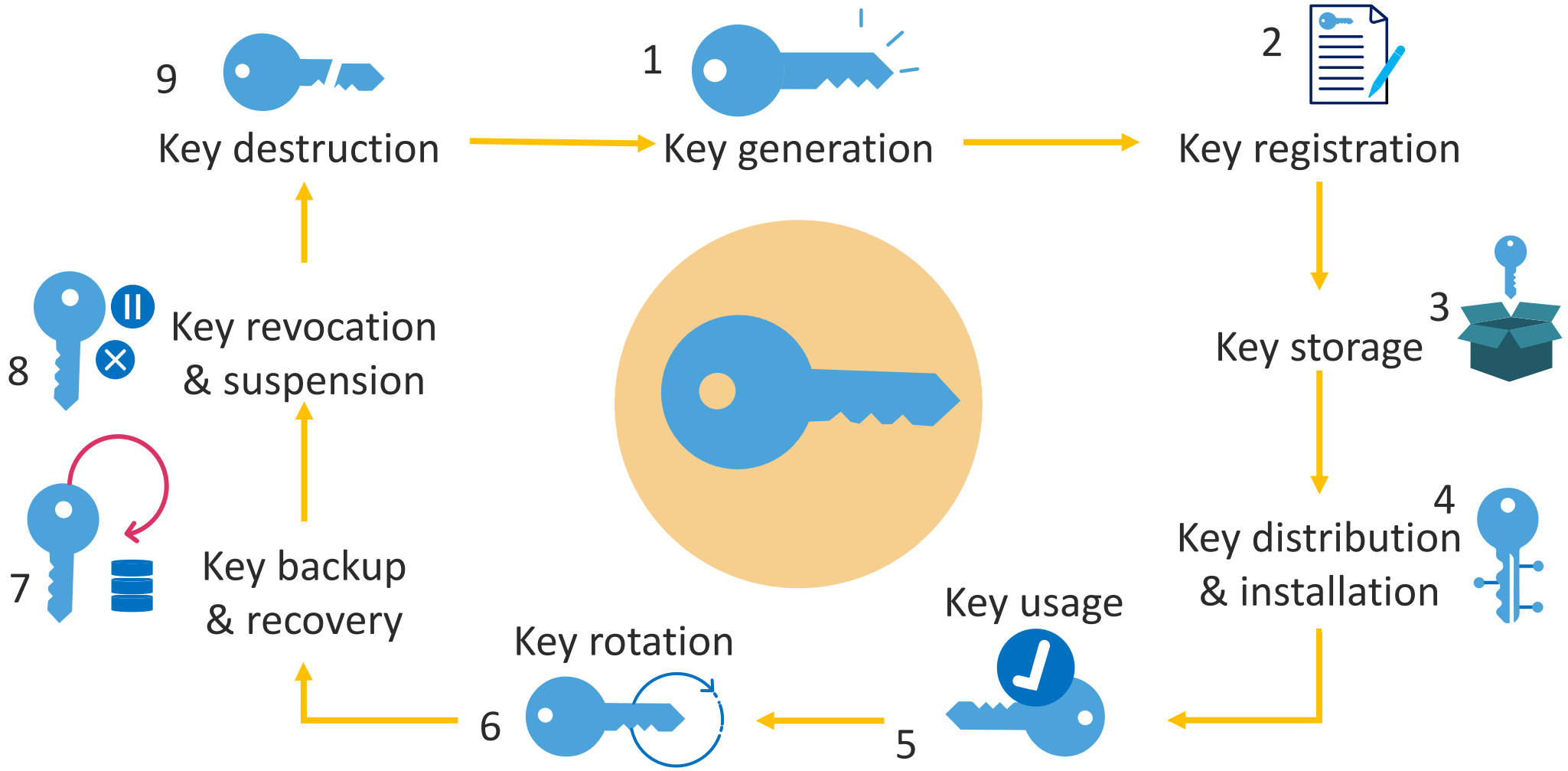
- Quantum cryptography is also known as quantum encryption
- It is an "early-stage" technique for encrypting and conveying secure data based on the naturally existing and immutable laws of quantum mechanics:
  - Particles are inherently uncertain since photons can exist in more than on place (or state) simultaneously (superposition)
  - Photons can be measured randomly in binary positions
  - Quantum system cannot be measured without being altered
  - Particles cannot be completely cloned

QUANTUM CRYPTOGRAPHY SYSTEM



Recipients can discern the presence of eavesdroppers because the quantum state has changed due to observation

CRYPTOGRAPHIC LIFE CYCLE



CRYPTOGRAPHIC KEY LIFE CYCLE: GENERATION

- Keys can be generated using several different modalities:
  - Software program (RSA 2048) generating certificates
  - A key management system (for example AWS KMS)
  - Hardware security modules
  - Trusted third party (TTP)
- Each key should have a key strength (in bits) to offer sufficient protection for the entire useful lifetime of the associated data – with the ability to endure attacks during this lifetime

CRYPTOGRAPHIC KEY LIFE CYCLE: REGISTRATION

- Keys can be associated with accounts, users, systems, code, and applications and registered in a repository like an HSM or KMS
- If keys are tagged or labeled with key/value pairs or other unique identifiers, this metadata must be protected as well
  - AWS uses a pseudorandom key identifier (Access Key ID) for its secret access keys used with the command line interface (CLI) or software development kit (SDK)

CRYPTOGRAPHIC KEY LIFE CYCLE: STORAGE

- The key storage database (which will have a master key for encryption) next stores the keys along with all extensible attributes
  - Qualities include things like name, activation date, size, and instance
- A key can be activated upon creation or set to be activated later either automatically or manually
- Keys are often stored in the same system that generated them as mentioned

CRYPTOGRAPHIC KEY LIFE CYCLE: DISTRIBUTION AND INSTALLATION (LOADING)

- The goal of the deployment and install (loading) stage is to put the new key into a secure cryptographic device, either manually or electronically
- This is a critical phase for key security and should only be done by highly-privileged and audited personnel when conducted manually (the most common method)
- Governance such as the Payment Card Industry Data Security Standard
(PCI-DSS) now mandates that the key usage be equally secured along with encrypting the key material

CRYPTOGRAPHIC KEY LIFE CYCLE: USAGE (UTILITY)

- As with any type of data, keys should have utility, use cases, and meaning or they should not be generated
- The key management system should let activated keys be claimed by authorized users, systems, and apps for encryption or decryption processes, or message authentication code (MAC) creation and authentication

CRYPTOGRAPHIC KEY LIFE CYCLE: ROTATION

- Key managers should replace, and rotate keys, automatically based on an established schedule (according to the key's expiration date or life cycle)
  - This should also occur when compromise is suspected or a device with a certificate is lost or stolen
- When replacing keys, it is common to re-encrypt all stored data under the new key
- Rotating keys can be difficult overhead since it can entail additional procedures and protocols, which may include correspondence with third parties in public-key systems

CRYPTOGRAPHIC KEY LIFE CYCLE: BACKUP AND RECOVERY

- Sometimes a key can be lost during use (like equipment failure or forgotten credentials)
- A secure backup copy should be made available from media such as external media (CD, USB fob, FireWire) or using an existing traditional backup solution
- When a symmetric key or an asymmetric private key is being backed up, it must be encrypted before being stored
- **Backup may be referred to as "Archival"**
- The recovery and restoration process must be tested on a regular or automated basis

CRYPTOGRAPHIC KEY LIFE CYCLE: REVOCATION AND SUSPENSION

- The most common example of this process is when a device storing an X.509v3 certificate is lost or stolen
- A public key infrastructure (PKI) relies on the Certificate Revocation Lists (CRL) or an Online Certificate Status Protocol (OCSP) database to revoke the public/private key pair
- Keep it mind that a 'suspended' public key does not populate the list or database
  - This may be done for a leave of absense or temporary sabbatical of a subject

CRYPTOGRAPHIC KEY LIFE CYCLE: DESTRUCTION

- The final phase is the end of the key's life cycle, where all instances, or just specific instances, are eliminated
- This may occur according to an archive policy (7 to 10 years)
- Recovery of that key may be possible, depending upon the method used

KEY DESTRUCTION (DISPOSITION)

Key destruction

Removes a key instance in an allowable form at a certain location

Data may still exist at the location (remanence) so that the key may be reconstructed for later use

Key deletion

Removes a key instance **and any information** from which the key may be restored from its operational storage/use location

Instances may still exist at other locations (archive)

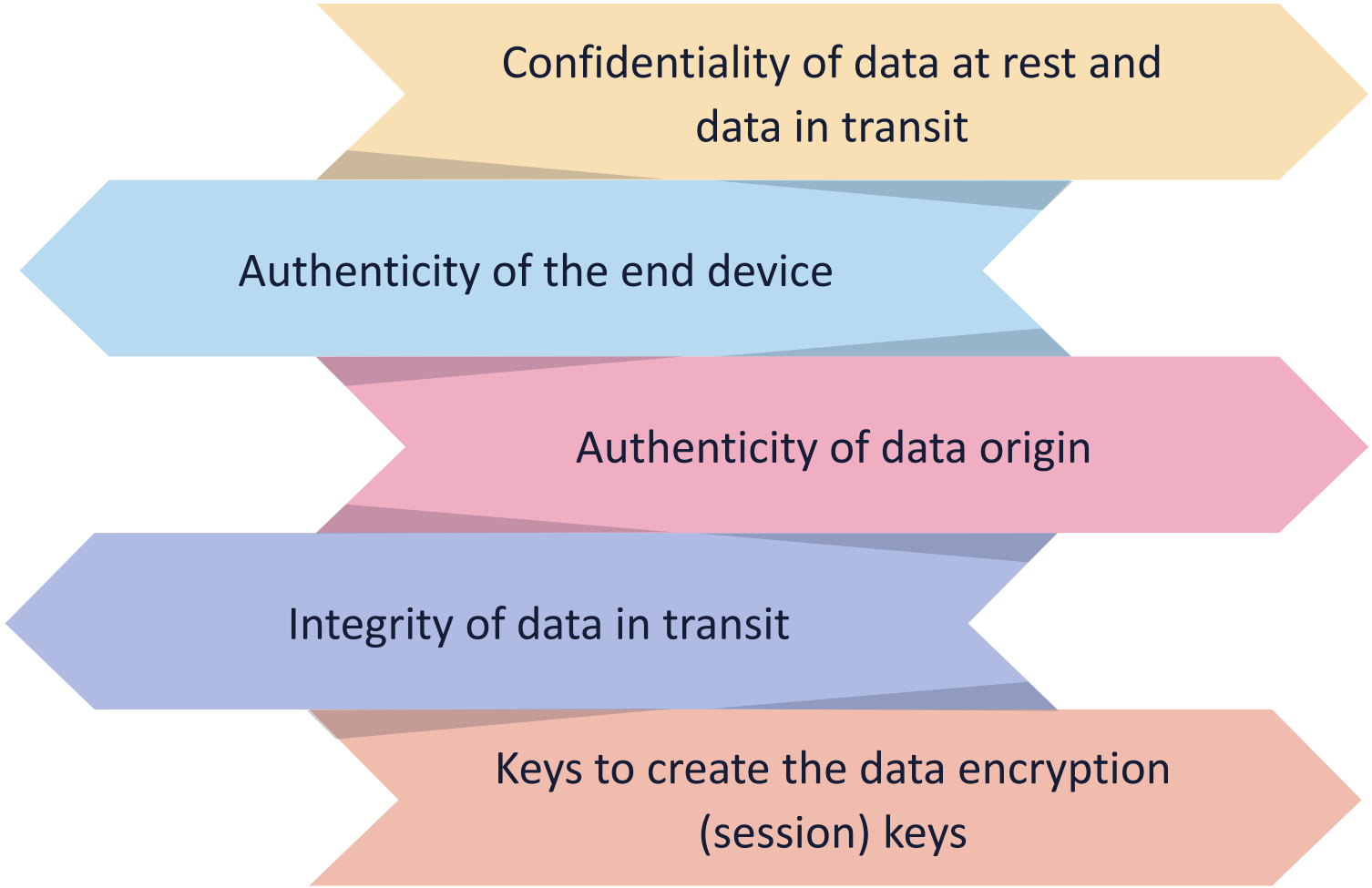
Key termination

All key instances and information are completely removed from all locations, making it unfeasible to regenerate or reconstruct the key (except from a restore from a backup image)

KEY MANAGEMENT BEST PRACTICES

- The cryptographic and key management algorithms being used by an application should start by knowing the objective utility of the application
- A thorough analysis of the use case of the application must be performed to determine the optimal key management method
- Use the most robust and pertinent suite of algorithms that are agreed upon between clients and servers, peers, and distributed systems

APPLICATION CRYPTOSYSTEM REQUIREMENTS



KEY MANAGEMENT BEST PRACTICES

- There are a varied collection of key types and certificates to use:
  - **Encryption**: Symmetric and/or asymmetric encryption keys (public and private)
  - **Authentication of endpoints**: Pre-shared symmetric keys, trusted certificates, **trust anchors***
  - **Data authentication**: Hash-based message authentication code (HMAC)
  - **Integrity protection**: MACs
  - **Key encryption keys** (KEKs)

KEY MANAGEMENT BEST PRACTICES

- Keys stored in memory for a long time can become "burned in" so this must be mitigated by splitting the key into components that are frequently updated (NIST SP 800-57)
- By implementing temporary ephemeral keys, entities can achieve **perfect forward secrecy** protection
  - This means that a compromise of the service or application's long-term signing key does not compromise the confidentiality of past sessions

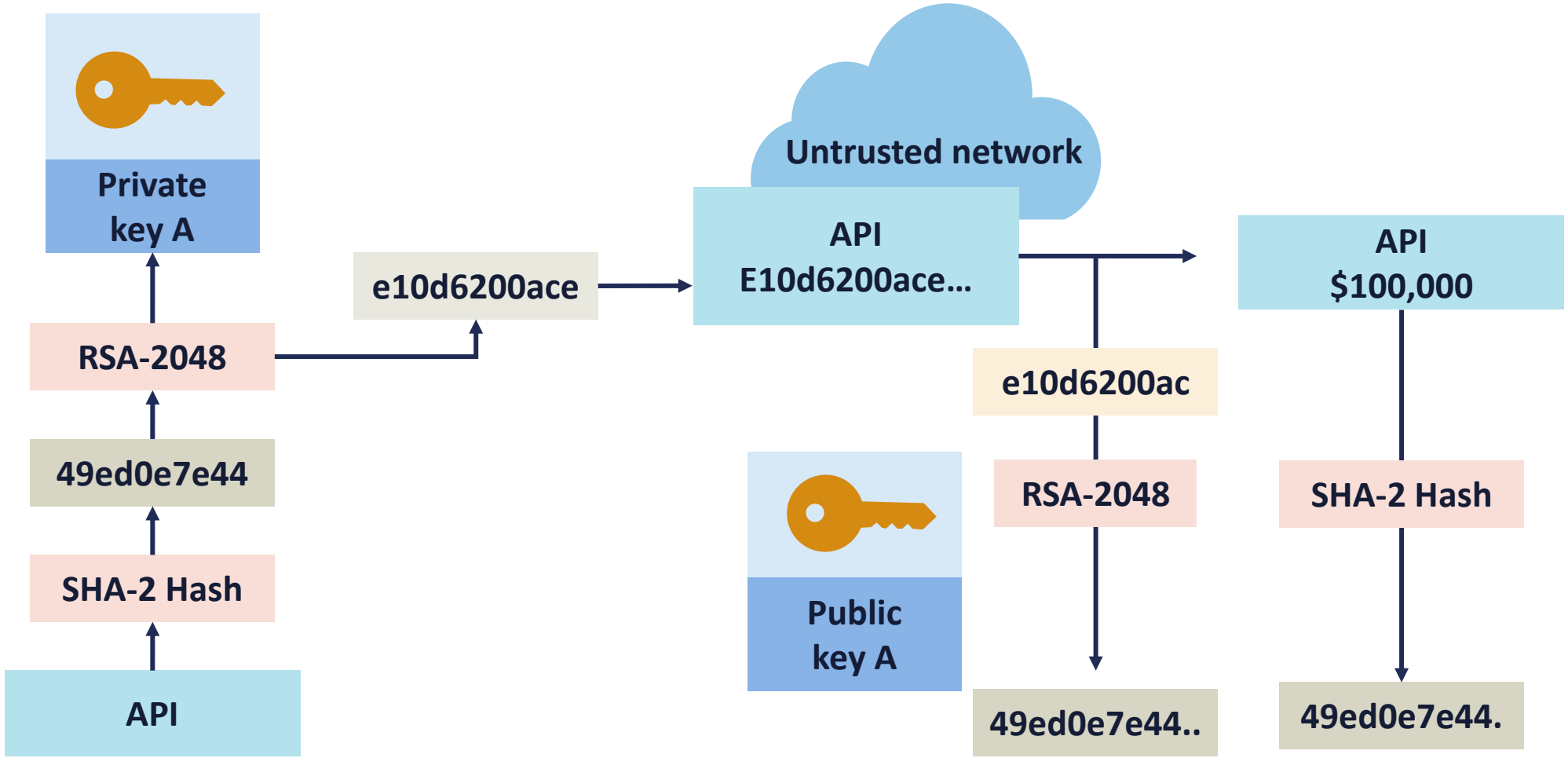
KEY MANAGEMENT BEST PRACTICES

- According to NIST, a distinct key should only be used for one purpose
  - Including encryption, authentication, key wrapping, random number generation, or digital signatures
- Valid reasons include:
  - Using the same key for two different processes may weaken the security provided by one or both use cases
  - Restricting the use of a key limits the damage that could be done if the key is compromised
  - Some usages of keys can interfere with each other

DIGITAL SIGNATURES

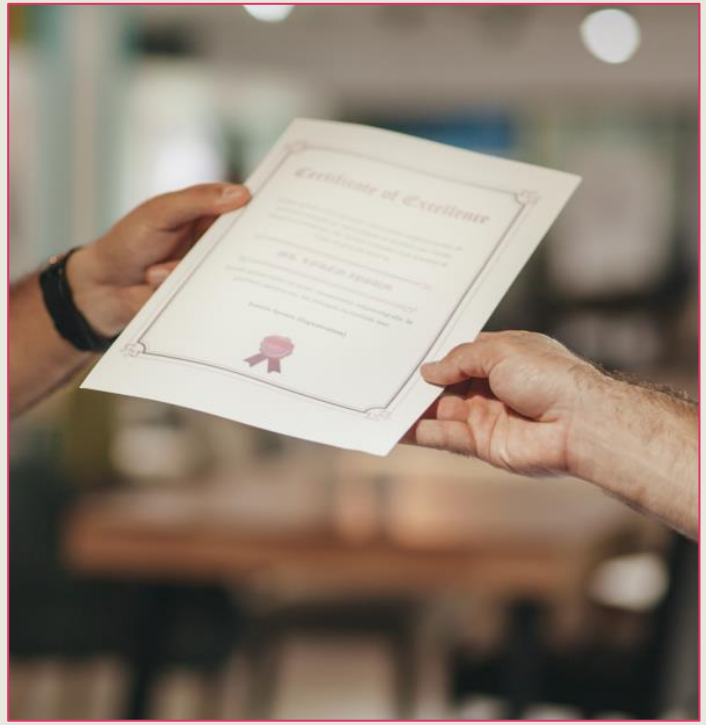
- Digital signatures are a form of electronic signature that was designed to replace and/or supplement a physical handwritten signature
- They leverage a mathematical algorithm commonly used to validate the authenticity and integrity of a message
  - This can include email, credit card, online transaction, or some form of digital document
- They create a virtual fingerprint that is unique to an entity
- It's used to identify principals and protect information in digital messages or documents
- Digital signatures are more secure than other forms of electronic signatures

DIGITAL SIGNATURES



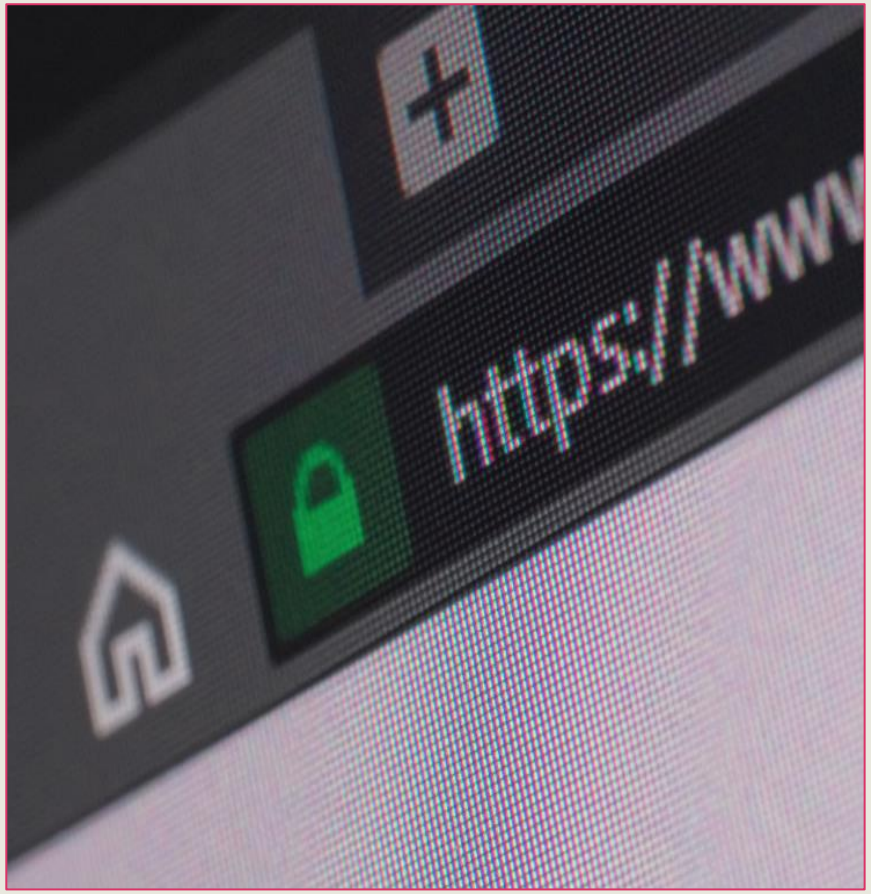
DIGITAL CERTIFICATES

- A digital certificate is a form of file used to map cryptographic key pairs to entities such as individuals, web sites, or organizations
- If public trust is needed, then a trusted certificate authority (CA) will assume the role of a third party to validate, identify, and associate them with cryptographic pairs using the digital certificates
- The owner of the private key can then use it to sign documents, while the public key is used to verify the validity of those signatures

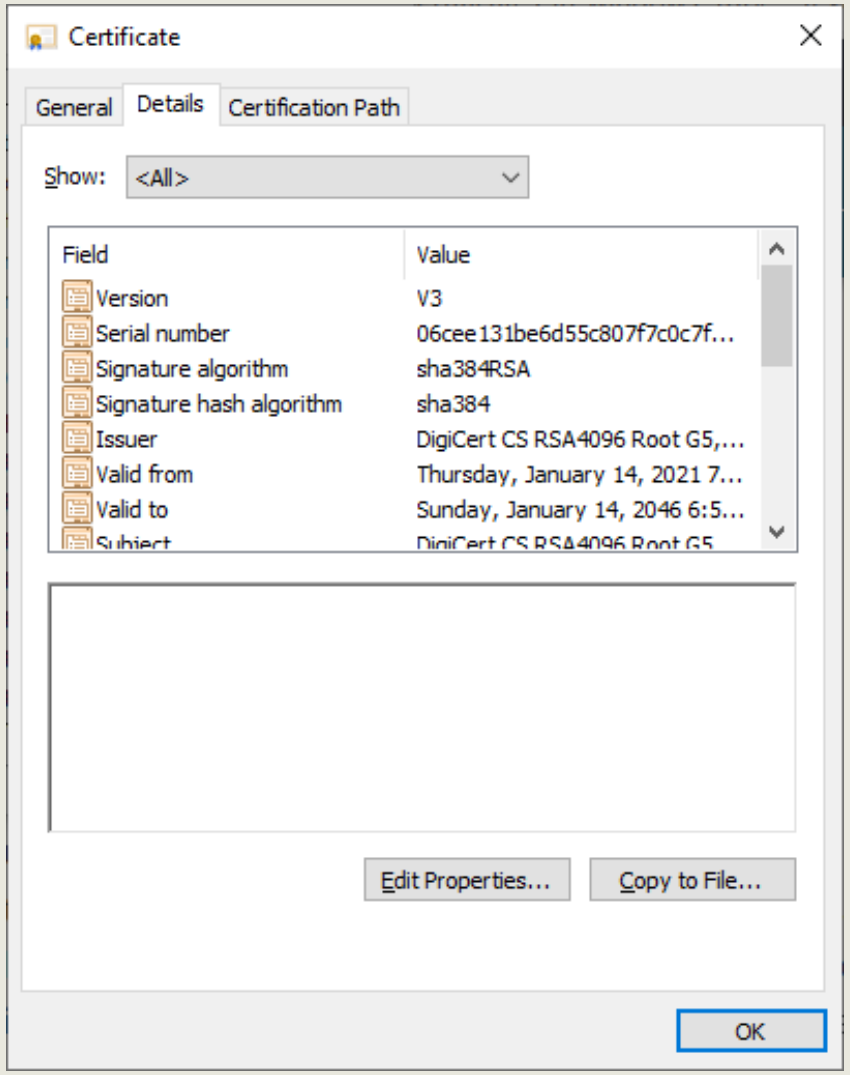


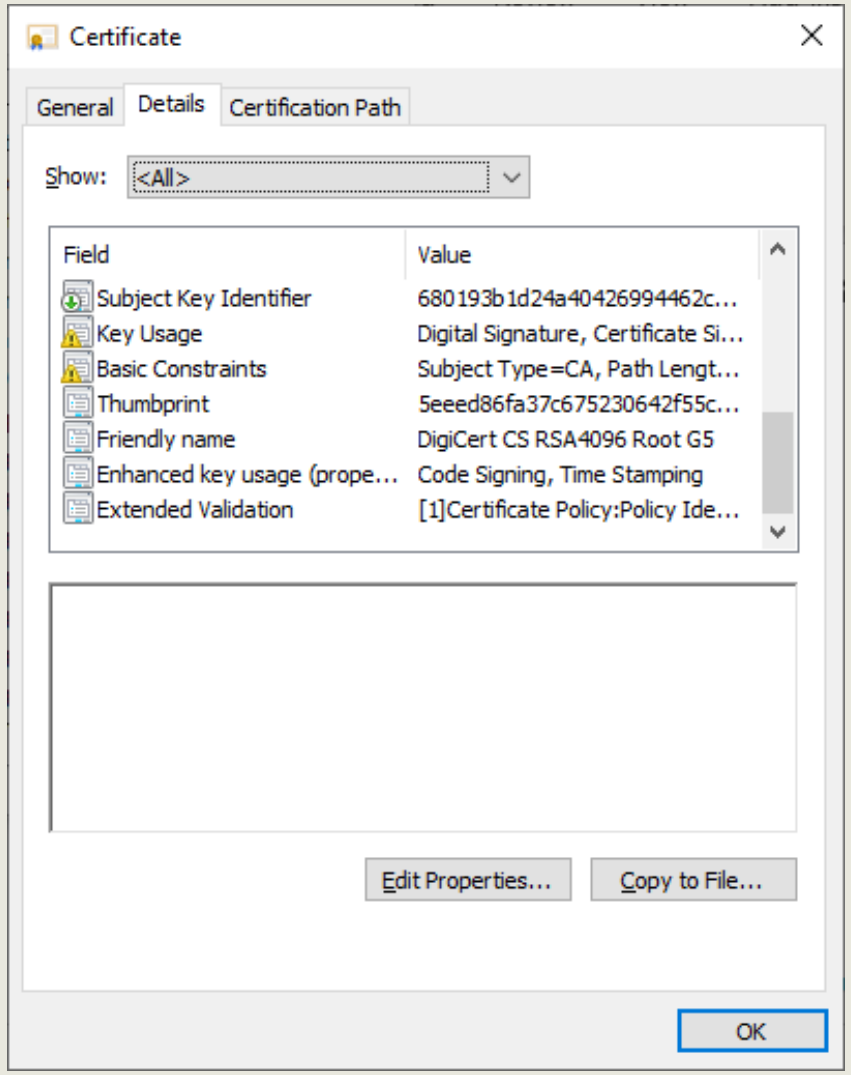
DIGITAL CERTIFICATES

- The most common commercial format for digital certificates is based on the X.509 standard, which consists of the following:
  - A public key
  - A pseudorandom hexadecimal serial number linked to the certificate
  - Information about the issuing CA
  - A lifetime (usually 1 to 5 years)
  - A digital signature
  - The hashing and signing algorithms used
    - Hashing – SHA1, SHA2, SHA384
    - Signing – DSA, RSA, and ECDSA
  - Extensions for more features like security



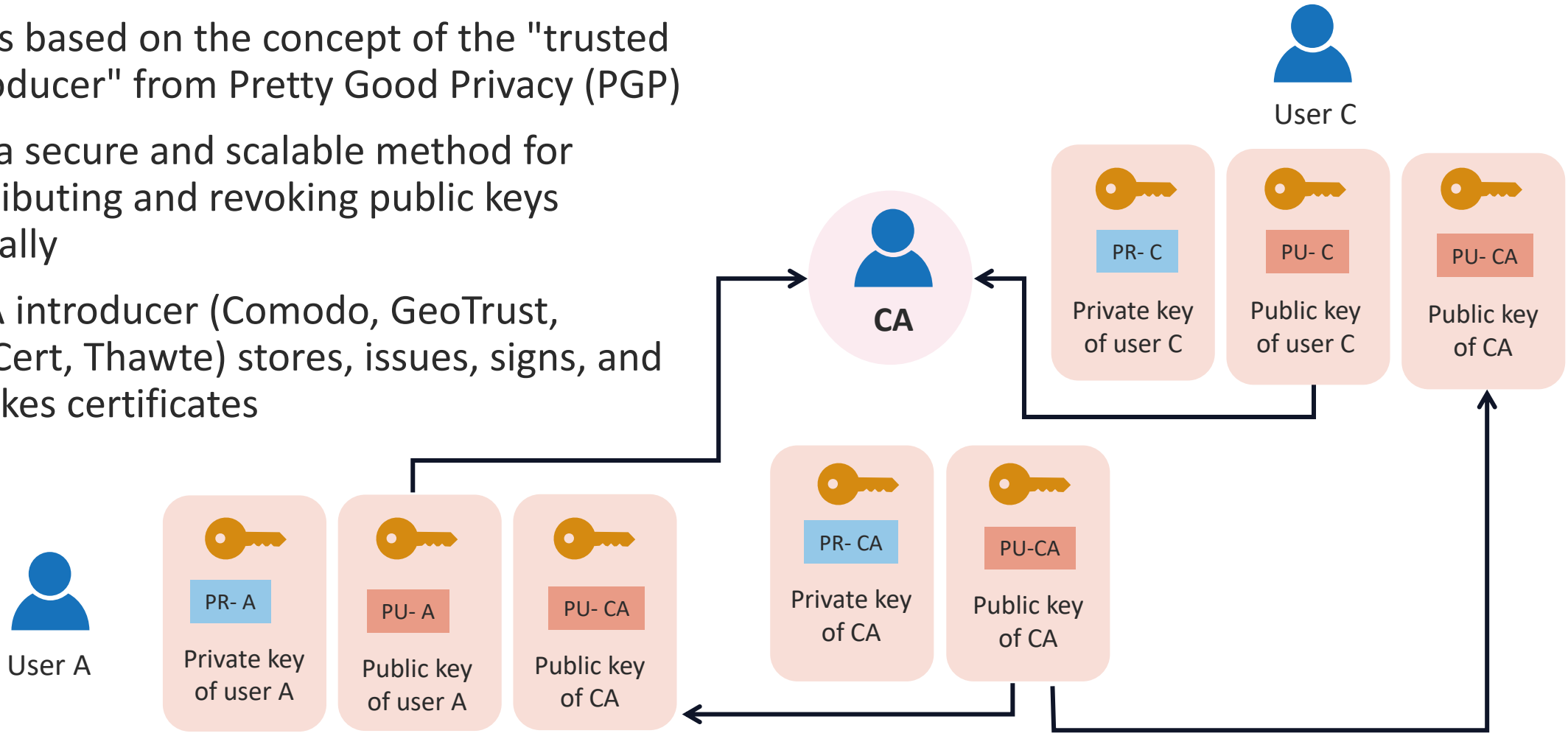
DIGITAL CERTIFICATE





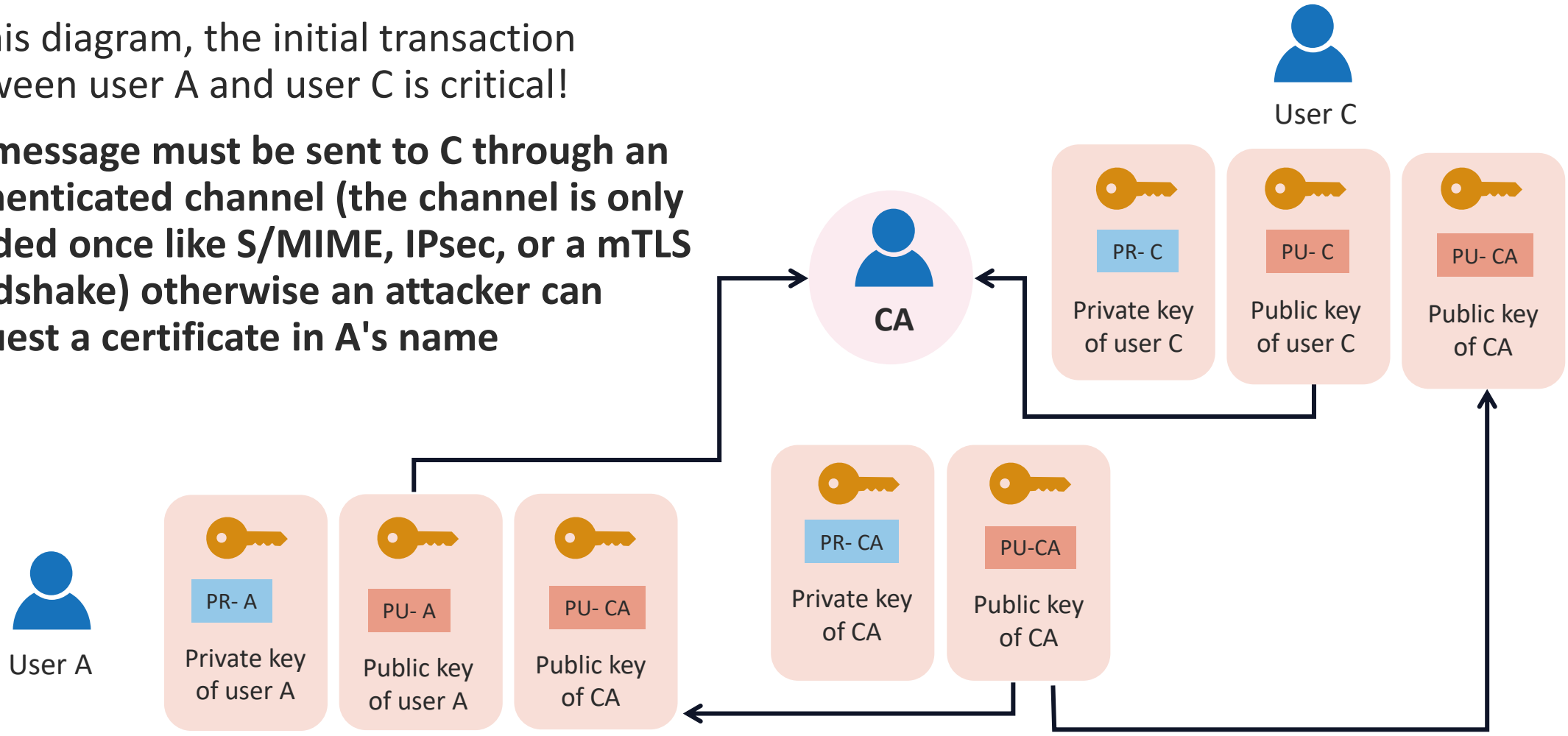
PUBLIC KEY INFRASTRUCTURE

- PKI is based on the concept of the "trusted introducer" from Pretty Good Privacy (PGP)
- It is a secure and scalable method for distributing and revoking public keys globally
- A CA introducer (Comodo, GeoTrust, DigiCert, Thawte) stores, issues, signs, and revokes certificates



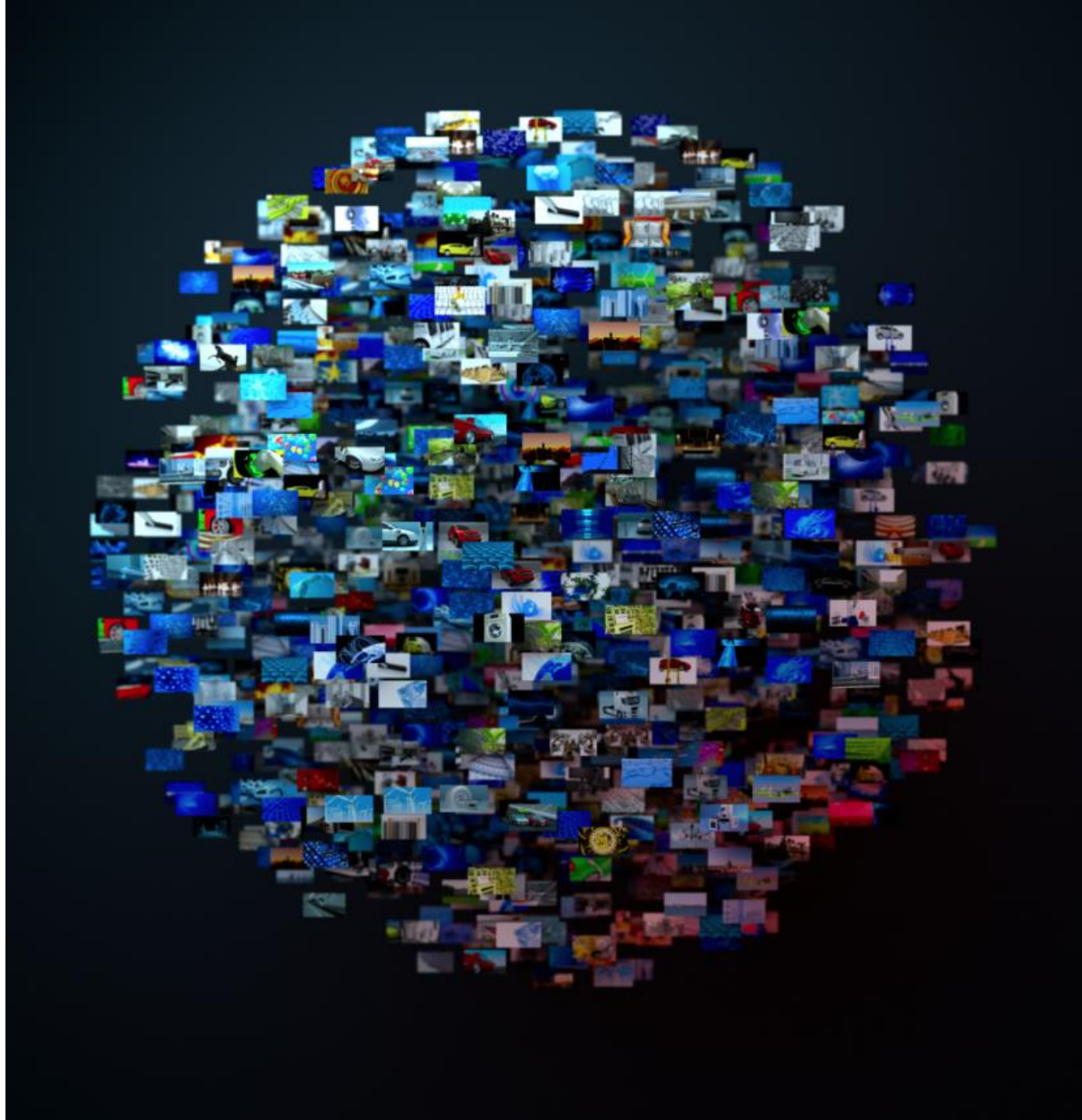
PUBLIC KEY INFRASTRUCTURE (PKI)

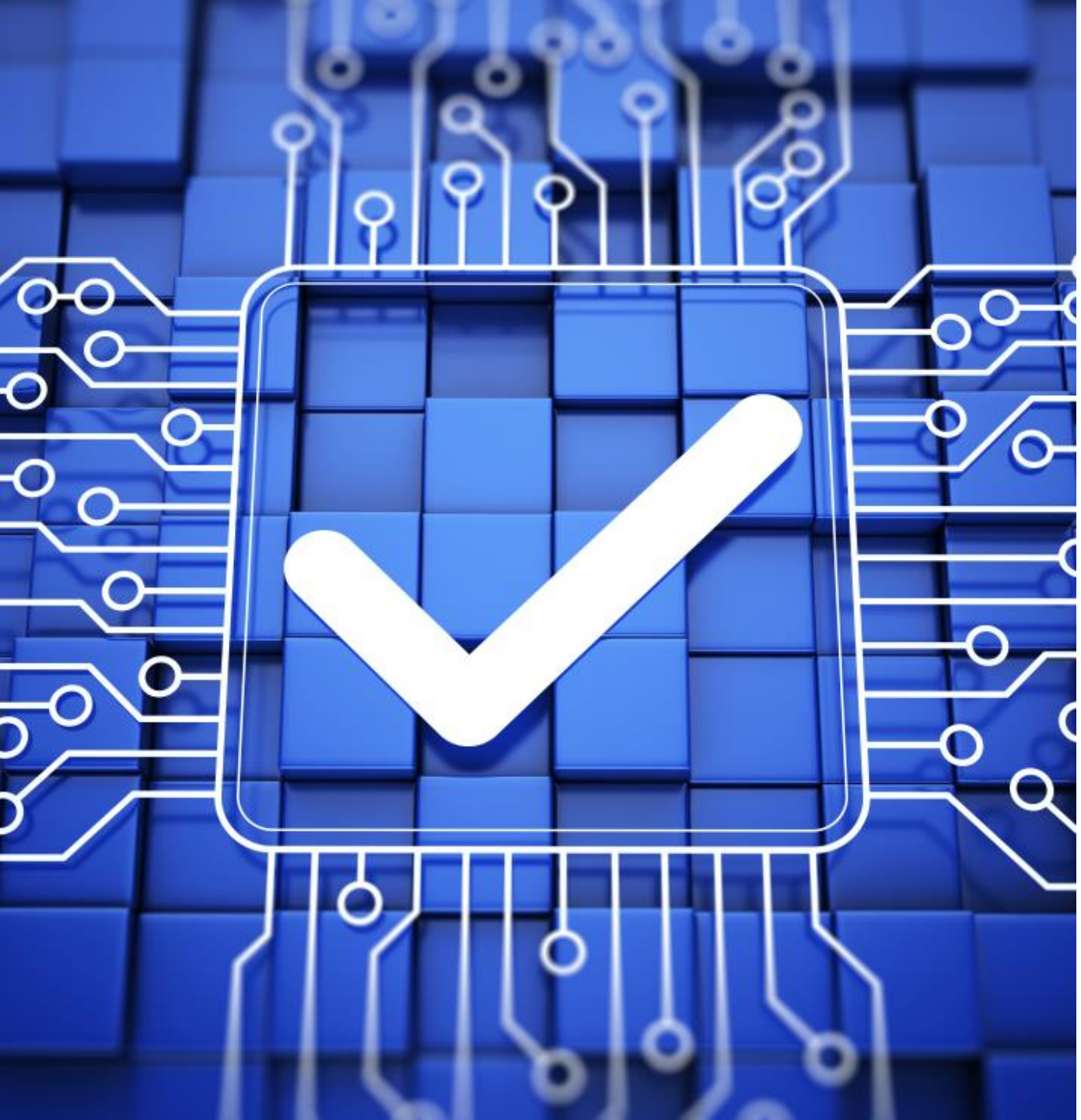
- In this diagram, the initial transaction between user A and user C is critical!
- **A's message must be sent to C through an authenticated channel (the channel is only needed once like S/MIME, IPsec, or a mTLS handshake) otherwise an attacker can request a certificate in A's name**



PUBLIC KEY INFRASTRUCTURE

- Note that with X509.v3 certificates – if you look at the fields – 2 public key algorithms are involved
- The one within the certificate, the Subjects Public Key algorithm is often some 160-bit elliptic curve mechanism
- AND the one that was used to SIGN the certificate - like RSA 2048 or ECDSA





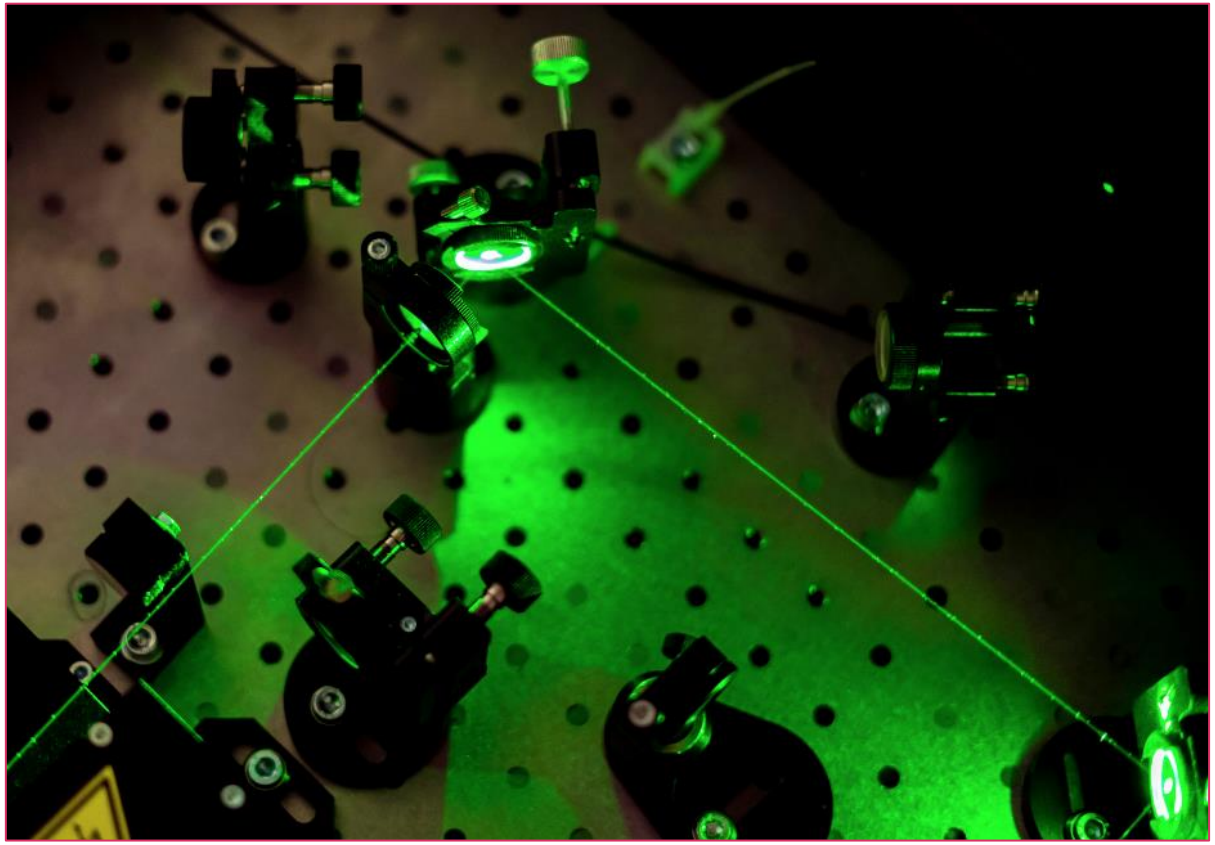
PKI CERTIFICATE REVOCATION

- The **certificate revocation list (CRL)** is the original method using a list of certificates that are invalid serial numbers and it is:
  - Issued by the CA who granted the certificate
  - Generated and published periodically
  - Pushed or polled in defined intervals or immediately – not real-time
  - Downloaded by client regularly but also not in real-time
- A CA can perform suspension of a certificate and that DOES NOT place the serial number on the CRL

PKI CERTIFICATE REVOCATION

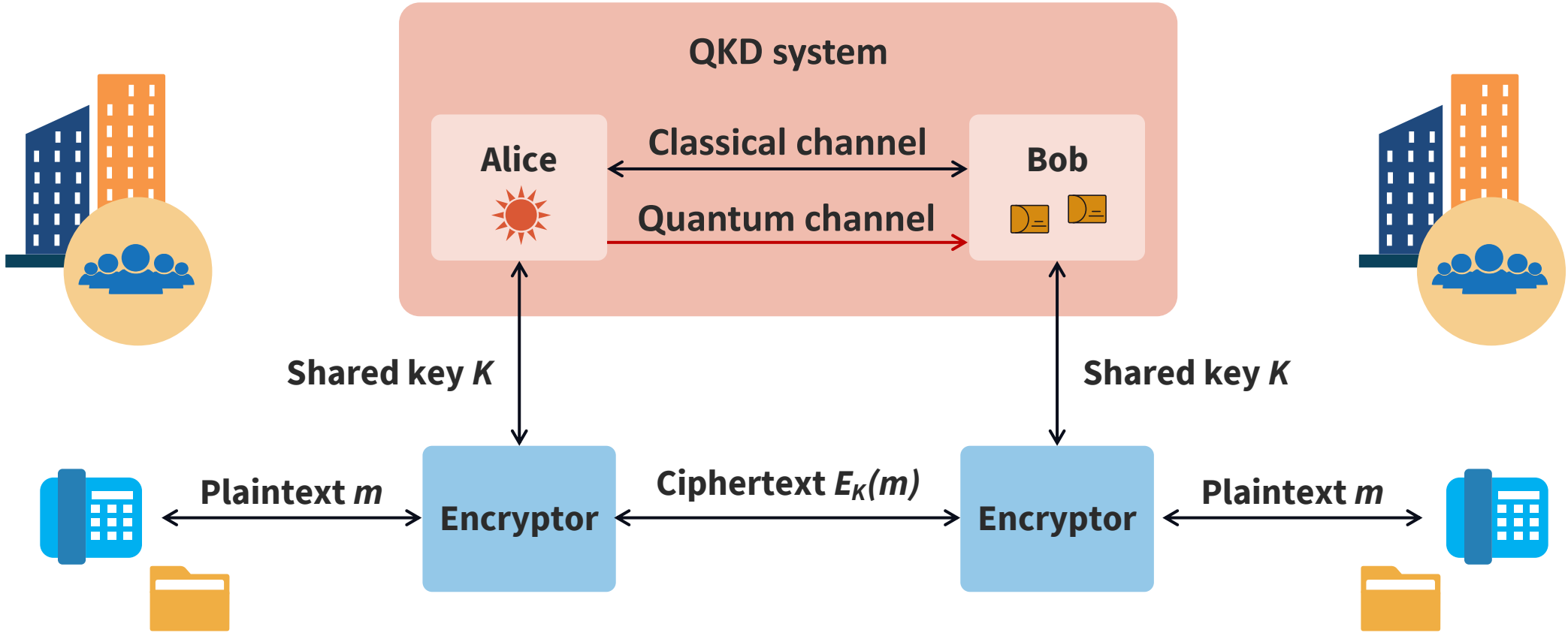
- **Online Certificate Status Protocol (OCSP)** is a database evolution of the CRL
- It is a method for a web browser or other client to determine the validity of a Transport Layer Security (TLS) certificate by verifying with the CA vendor
- It is a dynamic online transactional database of serial numbers that is generated and published immediately
- Clients query database anytime (although many vendors will bypass)
  - OCSP improves security, but causes websites to load slower

QUANTUM KEY DISTRIBUTION (QKD)



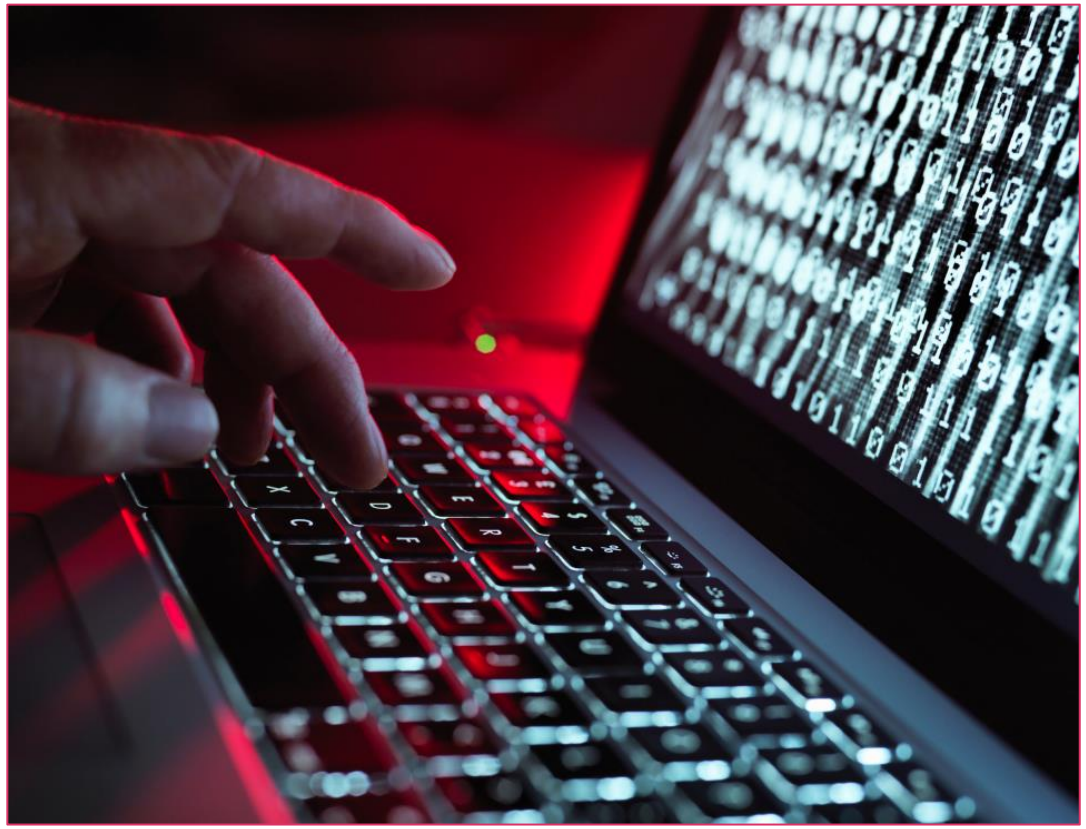
- QKD is a secure communication scheme that uses a cryptographic protocol based on quantum mechanics
- It allows two entities to generate a shared random secret key which can then be used to encrypt and decrypt
- The process of QKD is **not** the same as the quantum cryptography
- It is considered a possible "post-quantum" solution to implement beyond elliptic curve Diffie-Hellman ephemeral (ECDHE)

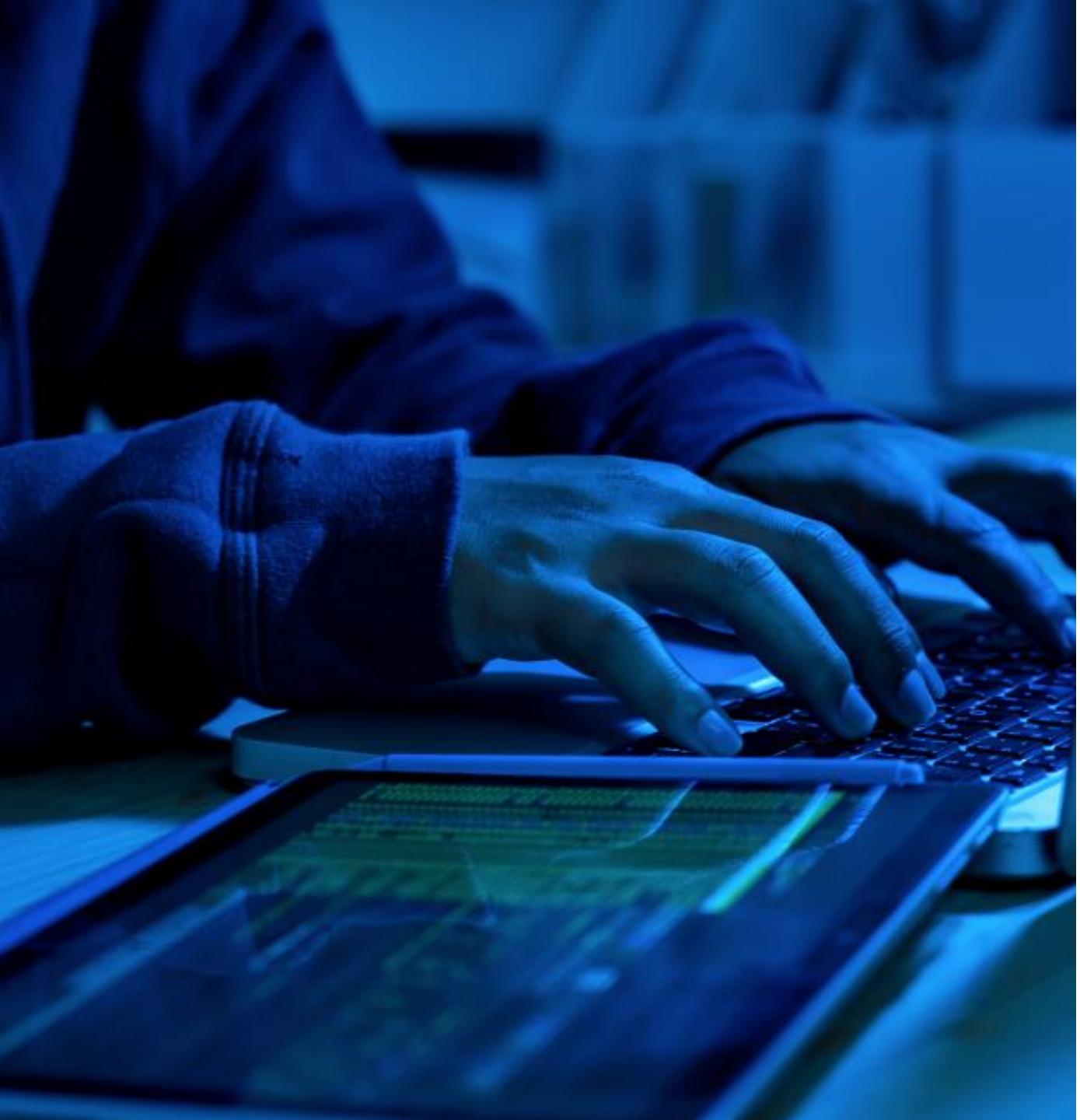
QUANTUM KEY DISTRIBUTION



IMPLEMENTATION ATTACKS

- Implementation attacks will commonly include the following:
  - Taking advantage of weak initialization vectors and nonces in various algorithms
  - Attacking weaknesses in key management such as lack of rotation and unsecure backups
  - Exploiting vulnerabilities in the underlying deployment such as the Dragonblood attack on WPA3
  - A focus on exploiting weaknesses in the software code itself



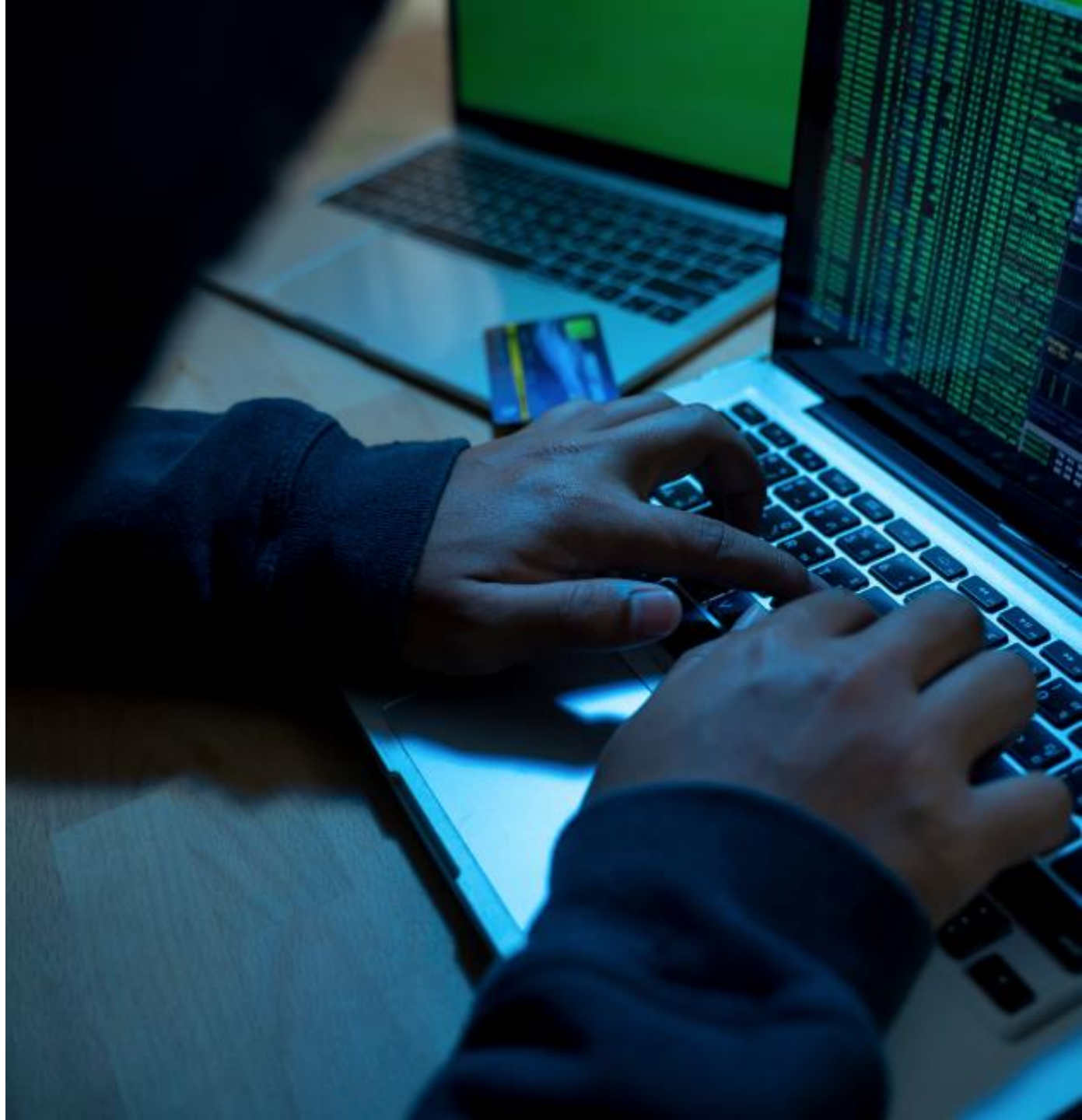


SIDE-CHANNEL ATTACKS

- A side-channel attack (SCA) is a passive attack where the attacker can collect information based on power consumption, electromagnetic radiation, execution time, and other attributes
- The immediate value of each of these sources depends on the data being processed in the module (such as a smart card), which in turn depends on the secret key

SIDE-CHANNEL ATTACKS

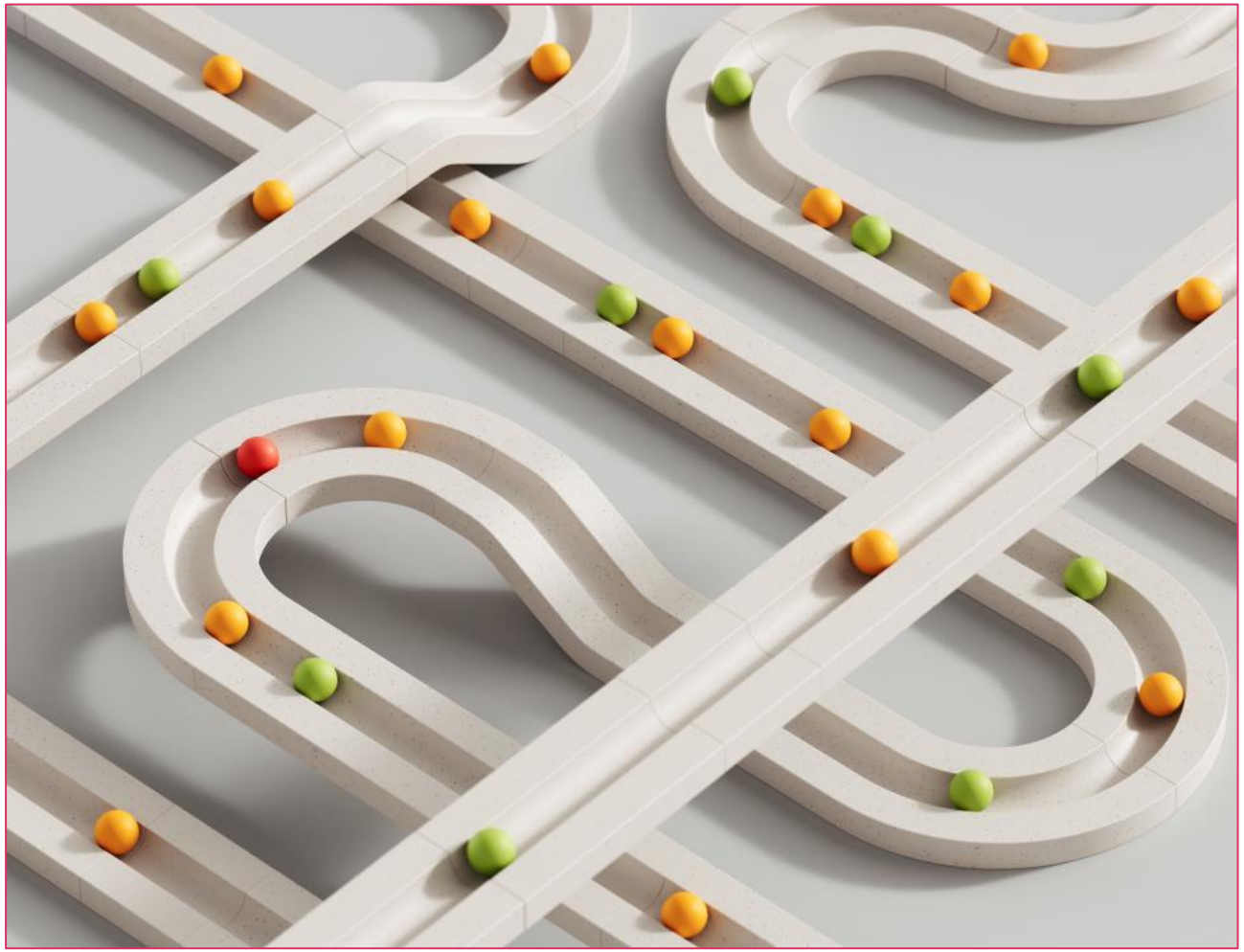
- SCA is a massive security threat since power leakage (for example) grants localized data about every operation of the cryptographic algorithm
- The adversary can target the first or last operation to focus entirely on minute parts of the secret key (sub-keys)
- SCA can also be launched by combining information from many leakage traces collected at different inputs
  - This is called differential power analysis (DPA) – an even more serious threat



ON-PATH ATTACKS

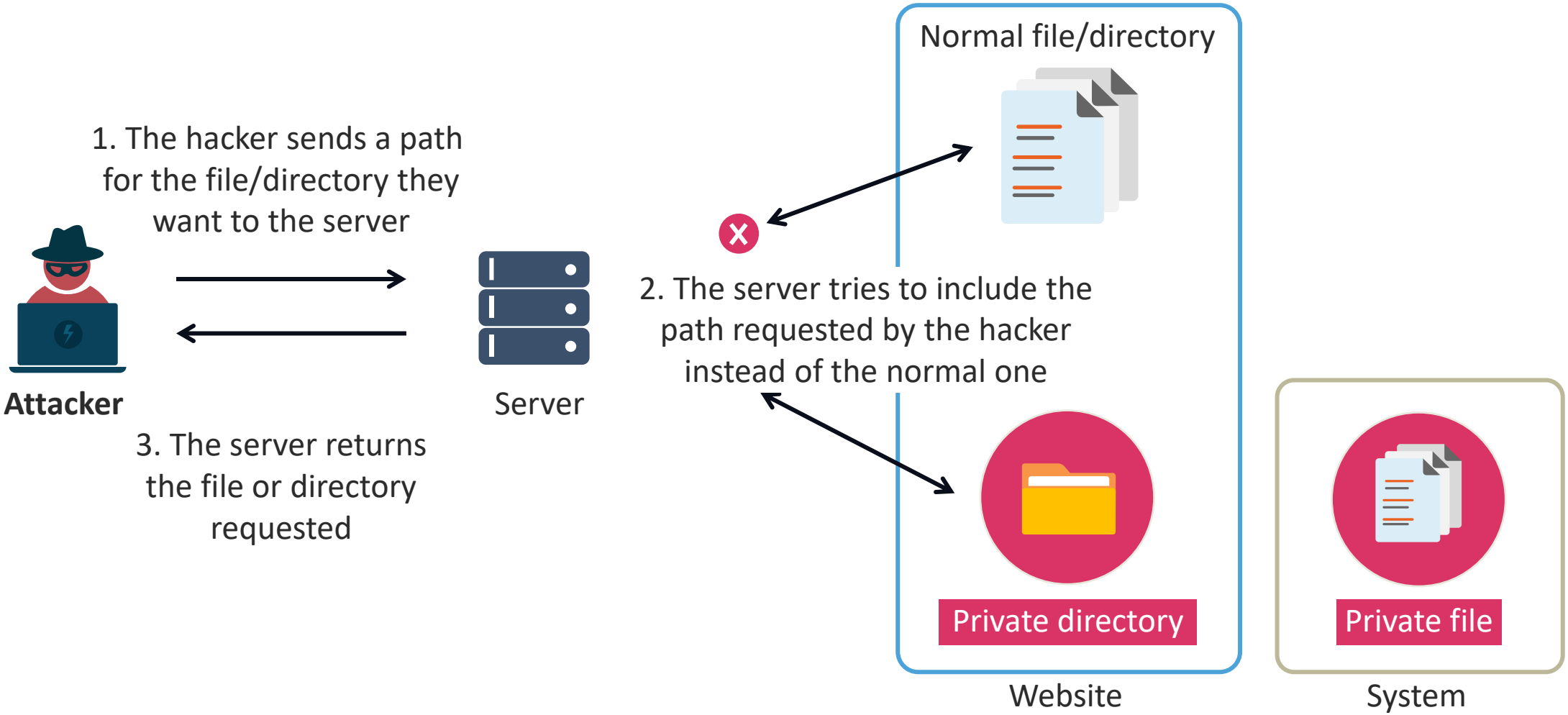
- An on-path attack (**previously called man-in-the middle - MITM**) is when an adversary puts themselves between two devices (such as a web browser and a server) and wiretaps or changes communications between the two hosts
- The attackers can then gather information as well as spoof or masquerade as either of the two hosts

ON-PATH ATTACKS

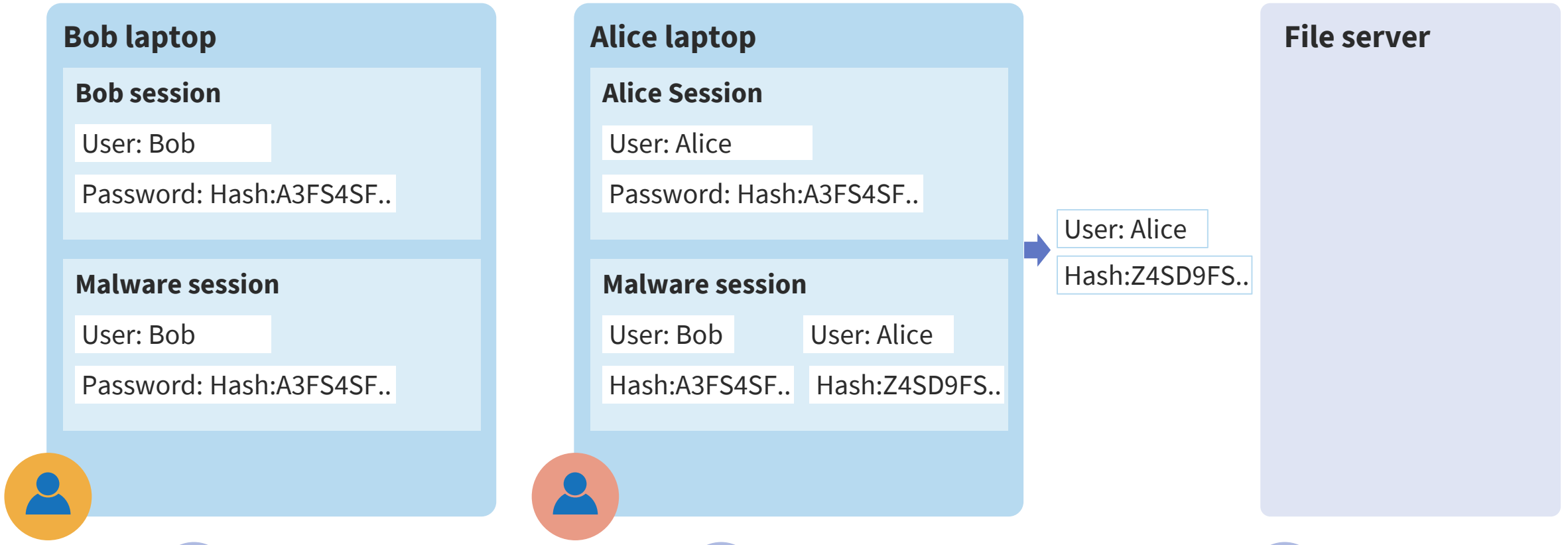


- Besides web applications, on-path attacks can target ARP and ICMP messages, email communications, and public/hotel Wi-Fi networks
- DNS servers are also common as victims suffer DNS spoofing and hijacking, sending users to a rogue site that gathers sensitive data
- The usual targets of on-path attackers are SaaS providers, e-commerce businesses, and financial app users

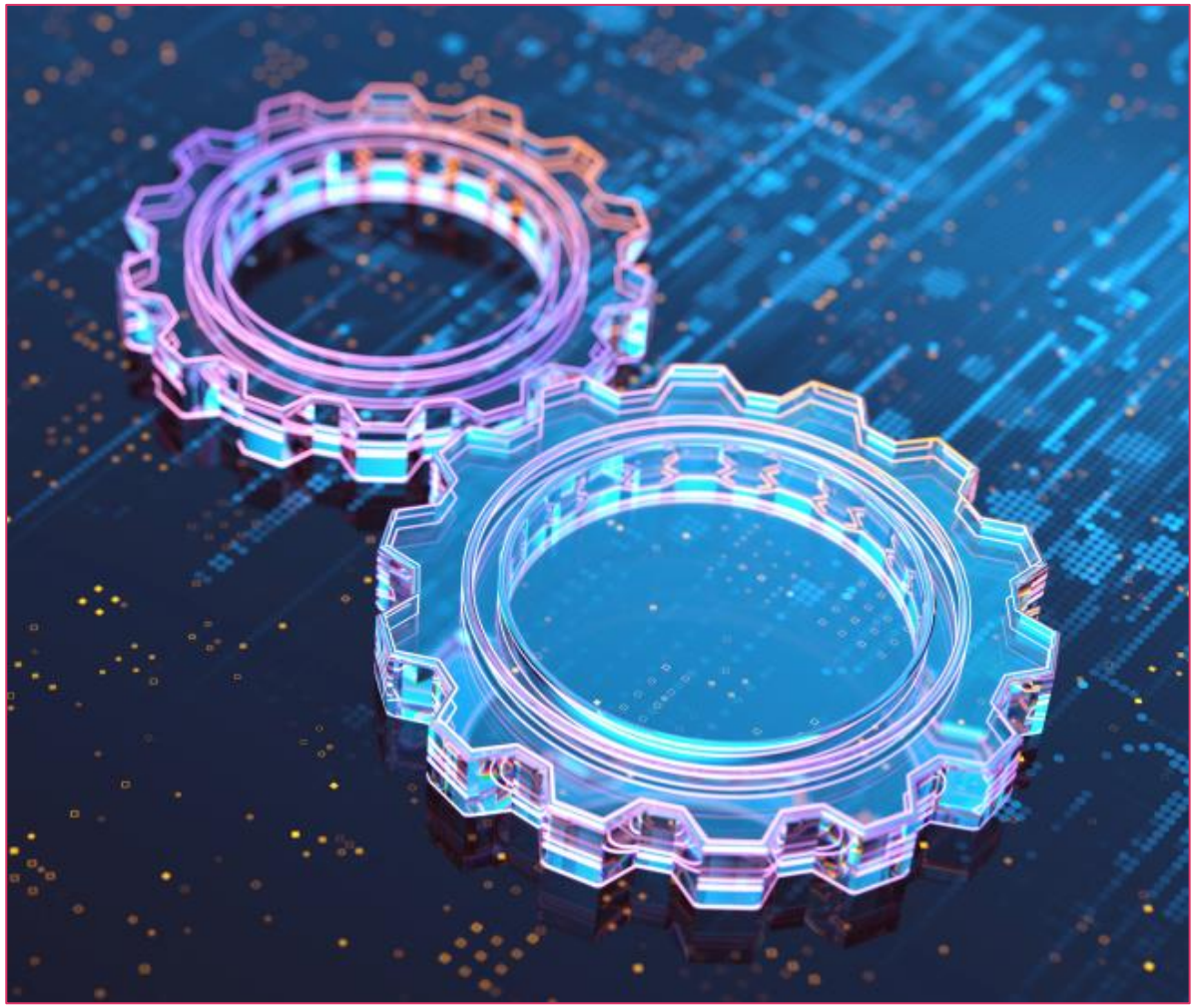
ON-PATH ATTACKS



PASS THE HASH ATTACK

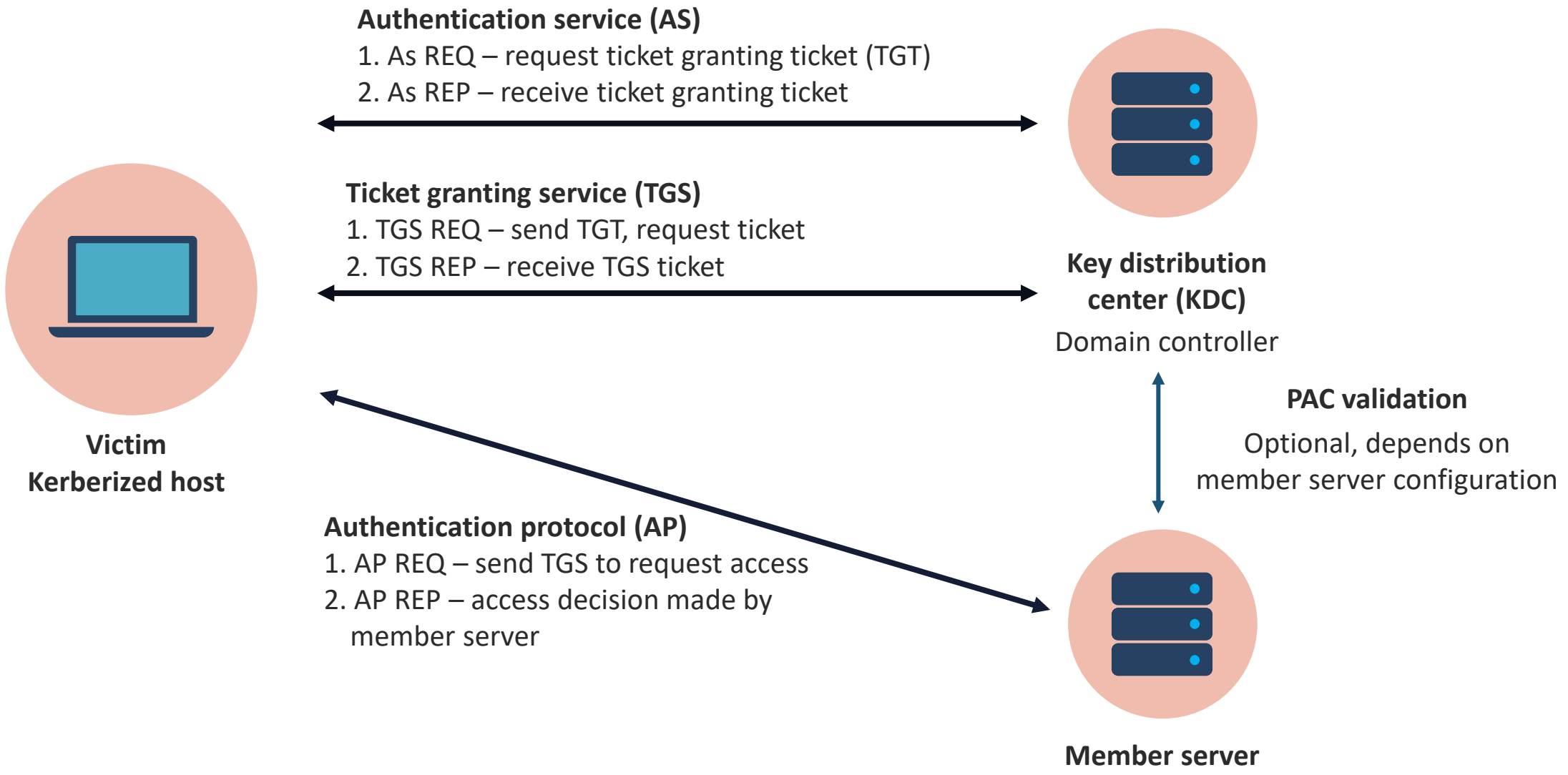


EXPLOITING KERBEROS

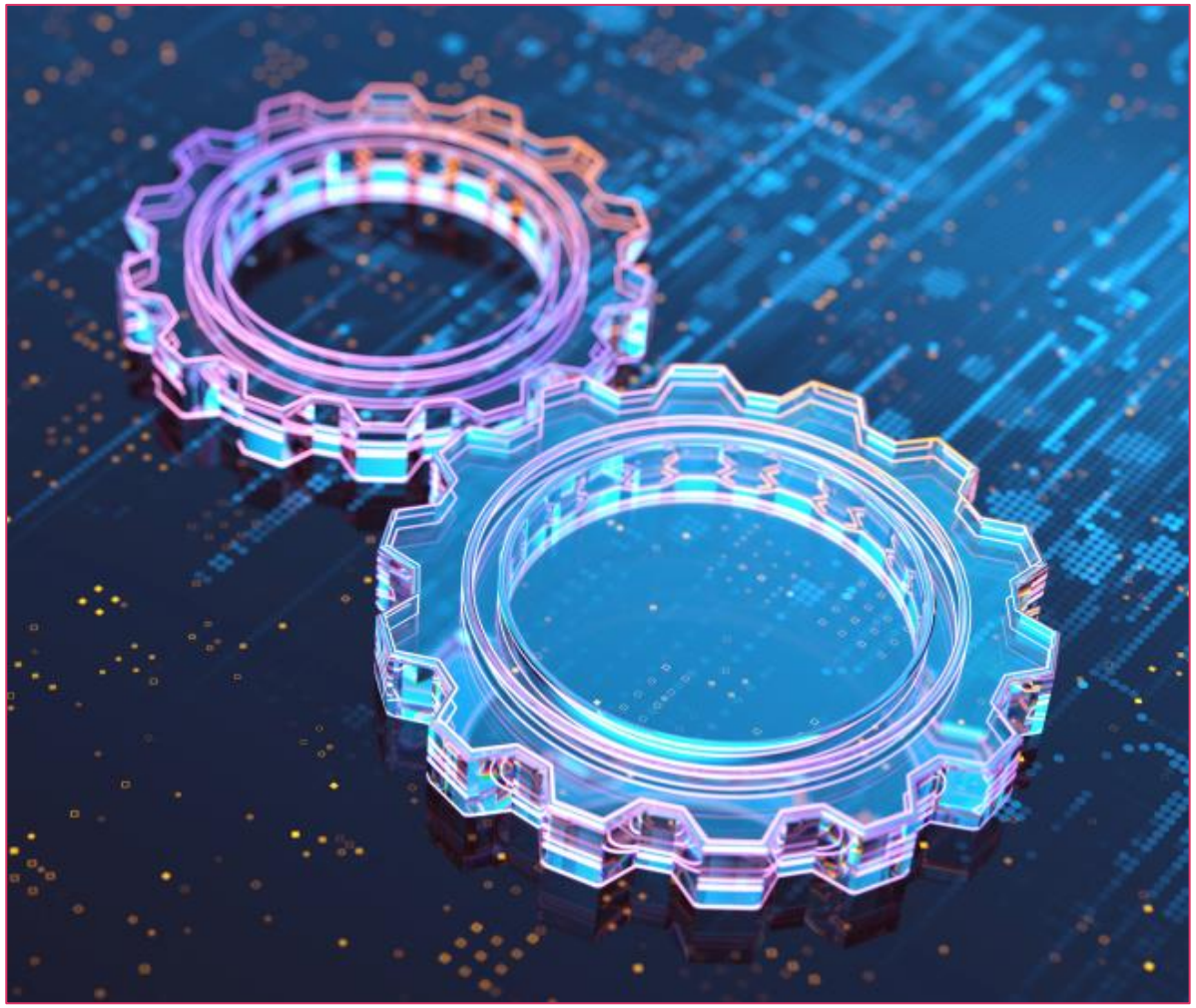


- **Kerberoasting**
  - Where an attacker removes encrypted Kerberos tickets from the network and tries to crack the encryption and password hashes for service accounts in Active Directory
  - If successful, they can get access to sensitive data or network resources

KERBEROASTING ATTACK



EXPLOITING KERBEROS



- **Golden Ticket attacks**
  - Where a threat agent handles the Kerberos authentication protocol to get unrestricted access to an organization's entire realm or domain (including devices, files, and domain controllers)
  - This hack lets unauthorized users exploit the krbtgt account, a key aspect of the Kerberos system responsible for encrypting and signing all tickets

EXPLOITING KERBEROS

- **Pass-the-ticket attacks**
  - A lateral movement exploit where attackers steal a Kerberos ticket from one system and use it to get access to another system by reusing the stolen ticket
- **Elevation of privilege vulnerabilities discovered by Google Project Zero**
- **Downgrade of encryption from AES to MD4-RC4**
- Group Policy, change, configuration, and patch management are the keys to mitigation

RANSOMWARE LIFE CYCLE

User receives spammed message with archived attachment

Attachment is downloader malware that connects to URLs hosting the crypto-ransomware

Crypto-ransomware is downloaded onto the computer

Files in the affected computer are encrypted

A ransom message is displayed, stating the deadline and amount

Victims must use Tor browser to pay using Bitcoins

DEFENDING AGAINST RANSOMWARE ATTACKS

- Maintain a precise and timely inventory of all assets
- Deploy two top-of-the-line antivirus programs
  - Make sure that antivirus is configured to automatically scan emails
- Apply tested patches and updates to all endpoints
- Consider managed security services (MSSP) that block access to known ransomware sites and zero days
- Backup and restore all data to an external site or cloud service
- Conduct security and awareness and enforce AUPs