CSAP

COMMON SECURITY ARCHITECTURE *for* PRODUCTION

VERSION 1.2

PART 5C: IMPLEMENTATION CONSIDERATIONS – APPROACHES

Motion Picture Laboratories, Inc.

December 21, 2022

Contents

1 Introduction

CSAP v1.2 is presented in six parts:

Part 1: Architecture Description the main architecture document.

Part 2: Interfaces describes the possible interfaces between the modules in a canonical form.

Part 3: Security Levels presents a metric-based approach to scaling security.

Part 4: Securing Software-Defined Workflow discusses how the security architecture can be applied to software-defined workflows that are managed using a service bus.

Part 5: Implementation Considerations is broken into sub-documents (5A, 5B, and 5C), which cover different aspects of CSAP implementation. This part is a new addition to CSAP version 1.2.

Part 6: Policy Description describes how policies and rules are defined. This part has not been published as of December 2022.

It is assumed that the reader is familiar with the previously published parts of CSAP, 1 to 5, and we do not reiterate the concepts described in those parts.

1.1 How to Use Part 5

In creating the CSAP architecture, the designers wanted the result to be implementable using existing technologies and with the least development possible. Part 5 provides a perspective on the way the CSAP designers ensured those goals were met. However, the descriptions of implementation approaches may not represent the optimal approach and are not detailed enough to serve as implementation guides.

Part 5 is broken into sub-parts, each of which covers a particular aspect of implementation. This initial release as an addition to CSAP v1.2 consists of three parts that cover some aspects of implementing CSAP, by no means all of them. We expect to release additional parts as developers gain experience in implementing CSAP.

- In Part 5A subtitled “Starting Out,” Section 2 “CSAP Recap” and Section 3 “The Path to CSAP Implementation” provide general guidance and set the stage. Section 4 “Trust” looks at the meaning of trust, which is a core concept of CSAP, and of zero-trust architectures in general.
- In Part 5B subtitled “CSAP Core,” Section 2 “Identity and the Authentication Service” and Section 3 “Authorization and Authorization Rule Distribution Services” discuss implementation considerations for CSAP core security components. Section 4 “The User Experience” is a lesson in a way to create a good user experience.
- In Part 5C (this document) subtitled “Approaches,” Section 2 “The Network” covers ways in which networks may be used to support CSAP functions. Section 3 “Access Controls” discuss ways access to assets and resources can be controlled. Section 4 “End to End Security” looks at ways that the CSAP architecture can be used to facilitate end-to-end security on untrusted infrastructure.

1.2 Terms

Authentication is the security mechanism used to validate an entity's identity by a trusted authority. The entity might be a user, a service, a device, an application, etc.

Authorization is the security mechanism used by a trusted authority to determine whether an entity can perform an action.

An *Asset* is the broad term we use to mean any data and metadata that is part of the process of media creation including image data, sound data, and metadata. *This is the media definition of the word "asset," and not the definition used in cybersecurity where the word asset means any data, device, or other component (hardware or software) that supports information-related activities.*

The use of *Context* is the normal definition of that word, the setting, circumstances, or environment of an event. The MovieLabs Ontology for Media Creation has a specific and different meaning for *context,* which is not used in this document.

Policies are the abstract representation of what is to be authorized.[1]

Rules are the actionable representation of a policy.

A *Device* is a piece of infrastructure in the form of a physical or virtual system that serves as a platform for the execution of software.

Mutual authentication occurs when each entity that is part of forming a trust relationship can authenticate all the others. (As will be discussed later, in this context a user and their system may each be an entity.)

mTLS (mutual TLS) is a form of TLS with additional steps that authenticate the client to the server. In TLS, the server is authenticated to the client, but out-of-band authentication is required to authenticate the client to the server, e.g., using managed device services.

We use the terms for the structure of the organizations creating a creative work that are the common parlance of Hollywood, but they map directly to the terms used elsewhere.

The Studio is the entity that owns the rights to the creative work, is responsible for funding production and has a say (usually creative) in the production process. This is the same role as a commissioning broadcaster or a network (in the way that the term is used when referring to US linear broadcasters.) View the word "Studio" as a shorthand construct for anything that fits the definition.

Depending on the context, the term *The Production* is used to mean either:

- The entity responsible for carrying out production. This may be a production company, an organization set up to produce one creative work or a department or business unit that is part of the studio.

[1] Or, in the very specific case of Global Security Policies, what is to be denied. Global Security Policies are the only place where a "deny" construct is needed since everywhere else, CSAP is deny by default.

Or

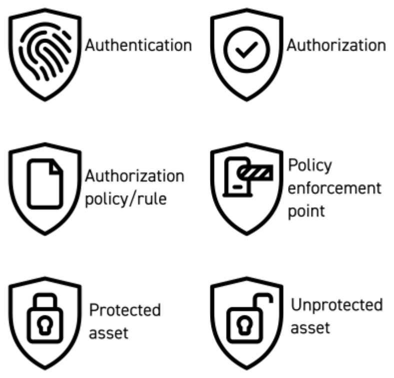
- The complete process of producing the creative work.

Vendors are companies that provide services to the production. They may also be called production service providers. Examples are a VFX company, a transportation company, and a cloud infrastructure provider.

Please do not assume that our use of these terms means that CSAP is only for Hollywood studios or only for motion picture production. CSAP is for all types of media production including scripted and unscripted television.

1.3 Visual Language Security Icons

The shapes and icons used in the diagrams in this document are part of the MovieLabs Visual Language. Rather than add a key for the security icons to each diagram, we include it here.



Further information on the MovieLabs Visual Language can be found here at www.movielabs.com/production-technology/visual-language-for-media-creation/

1.4 Choice of Examples

In this document we present examples using commercial products and services. The choice of technology vendor, for example a cloud provider, is in no way an endorsement and does not imply that one product is better than another. In fact, where possible, we have steered away from examples where there is only one vendor.

We also discuss Google's BeyondCorp, and when we do, unless we say otherwise, we are talking about the zero-trust security solution Google implemented in their own offices and about which they have published a range of papers. These papers are cited frequently in the literature. Consider this to be an academic reference.

1.5 References

1.5.1 MovieLabs Publications

The Evolution of Production Workflows, MovieLabs, 2020

The Common Security Architecture for Production, Parts 1 to 4, MovieLabs.

1.5.2 Publications from Government Organizations

Zero Trust Architecture, NIST Special Publication 800-207, https://doi.org/10.6028/NIST.SP.800-207

2 The Network

Restricting network access to authenticated and authorized entities is a zero-trust requirement and it is a level 100 requirement.

One way of looking at a zero-trust network is that network security perimeters are collapsed to the point where each individual system has its own security perimeter. While that is the goal, CSAP can be implemented without going that granular. If networks are defined such that joining the network requires authentication and authorization, then the organization may choose[2] to regard the network as a security domain. While this document does not contain best practices, it behooves us to say that such a choice should be taken with a great deal of care as to how well it can be enforced and what the ramifications are – after all, a tenet of zero-trust is the assumption that your network has been breached.

The larger the security domain – for example, the more systems in it – the worse the possible consequences of a security breach are since the breach can propagate to many different systems. Large security domains diverge from a zero-trust model and converge a with perimeter security model.

2.1 Mutual Authentication

Mutual authentication is a level 100 requirement. (It is an inherent part of CSAP authentication.)

The concept of mutual authentication is simple: each party to the connection authenticates the other before proceeding with the connection.

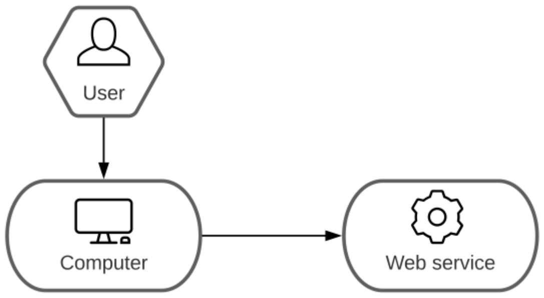
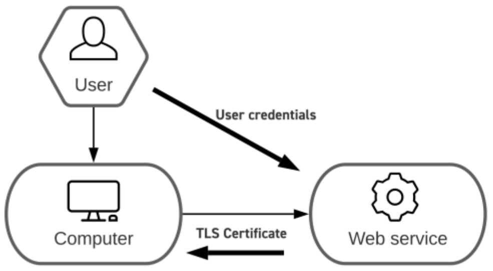


Figure 2-1 Using a web service

If we take the simple case of a user using a computer to access to web server. In this example, we are including the user's browser in their computer and treating the server and the service that runs on it as a single thing.

If we add the authentication into this diagram that is typical of users' access to web services, we get this:

[2] As with trust, the criteria are outside of the scope of CSAP.



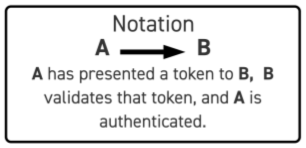

Notation
A → B
A has presented a token to B, B validates that token, and A is authenticated.

Figure 2-2 Typical authentication

In this example, the computer knows it is talking to the web service if the certificate presented by the web service is valid. And the web service knows that it is talking to the user if the user's credentials are valid, however the web service does not know it is talking to the user's computer. The computer knows it is talking to the web service, but the user does not know that directly. However, if the user trusts their computer that may be inferred from their computer authenticating the web service.

What is missing is the web service authenticating the user's computer which is mutual authentication. It is appropriate if the user chooses to trust their computer, but it is not advisable for the web service to trust the user's computer. After all, it may not be talking to the user's computer. It could be a man-in-the-middle attack.[3]

2.1.1 Mutual Authentication Protocols

Since TLS, the standard secure protocol of HTTPS connections on the Internet, is commonly used let us view it from the point of mutual authentication.

When a TLS connection is set up between a client (e.g., a user) and a service (e.g., a website), the service presents its certificate to the client and the client must confirm the certificate is valid, typically by contacting the certificate authority. However, nothing in this process authenticates the client to the service. If that is necessary, for example a bank website needs to authenticate its customers before giving them access to their accounts, a separate unrelated mechanism (e.g., a log in) is used to authenticate the client. Note, that in this example, nothing is authenticating the user's computer. It is assumed that the user trusts their computer[4] so a significant part of secure interactions with a website isn't authenticated.

A version of TLS called mTLS adds another step to the TLS connection set up where the client presents its certificate to the service which confirms the certificate is valid with the appropriate certificate authority. Once an mTLS connection is established, the authentication is mutual and, depending on the circumstances, further authentication of the client may not be necessary. For example, if a service and

[3] TLS protects, to some extent, against a malicious web service masquerading as a legitimate one. That's not universal because, if were, HTTPS proxies would not work. But TLS does nothing to assure the web service it is talking to a trusted machine.

[4] The UK bank NatWest website performs checks as to whether a customer's machine is trustworthy when a customer logs in. Using a form of trust inference, it will restrict what a customer can do after a successful authentication if the server detects remote control software installed on the customer's computer. Interestingly, this means that a trustworthiness criterion for the customer's device is applied by the bank.

the server it runs on can be treated as an atomic entity then authenticating either the service or the server can be taken as authenticating the atomic entity.

The protocol operates this way:

Client	Server
Client "hello" →	
	← Server "hello"
	← Server sends certificate
(Validates server certificate with CA)	
	← Server requests client certificate
Client sends certificate →	
	(Validates client certificate with CA)
Client sends cryptographic info →	
Client finished →	
	← Server finished

Communications can now be encrypted.

Checking the validity of the certificates is not part of the protocol but is required for meaningful authentication and the prevention of man-in-the-middle attacks. Failing to check the validity of a certificate is often identified to be the root cause of a failure of TLS.

Several approaches to security, including service messages and software-defined perimeters use mTLS. What they are doing is creating networks of authenticated devices that are authorized to connect to each other.

Exactly what is being authenticated by mTLS must be identified. A software service or application running on a platform is two components: the software and the platform. If they are an immutable whole, as would be the case with a hardened VM with the application or service pre-installed, they can be authenticated together,[5] otherwise they must be authenticated individually. Our authentication will likely be with the platform or a PEP acting as a proxy.

2.2 Mutually Authenticated Networks

A mutually authenticated network is a network (or mesh) of nodes has these characteristics:

1. Nodes are authenticated and authorized to join the network.
2. Before communication can take place between two nodes, there is mutual authentication resulting in an encrypted connection between the two nodes.
3. Attempted communication with a node other than from a member of the network is ignored.

A generic process for a node to join a network is this:

[5] Two variations are (1) a trusted platform authenticates the software at initialization time and is protected from the installation of unauthorized software and (2) the software is deemed to be trusted because it was installed in a 'clean' environment and no other software is installed.

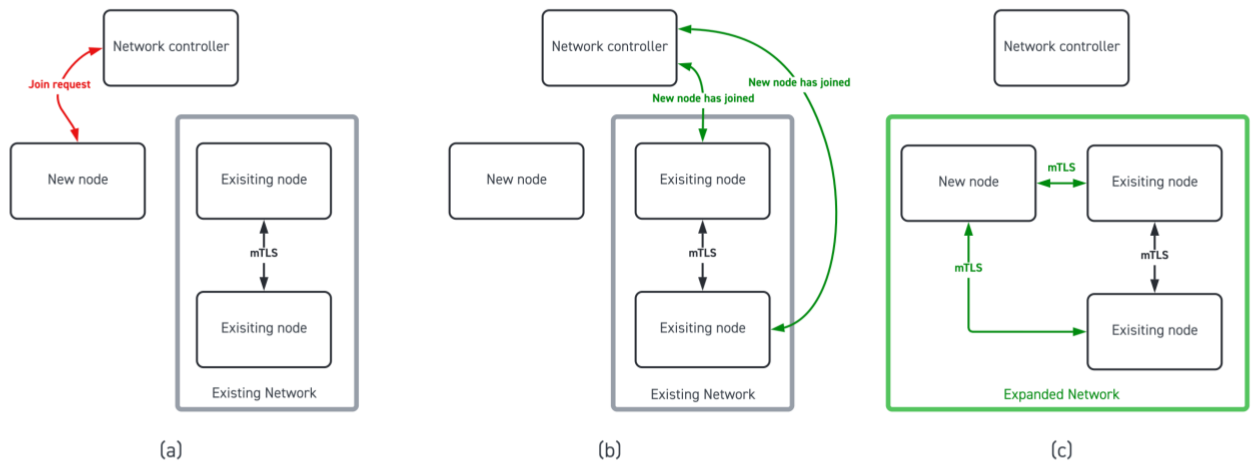


Figure 2-3 Network join

In Figure 2-3(a), the new node contacts the network controller requesting permission to join the network. In Figure 2-3(b), the network controller allows the new node to join the network and informs the existing nodes. In Figure 2-3(c), the new node connects to the existing nodes extending the network. The result is a network consisting of nodes that (a) can mutually authenticate with everything else in the network and (b) are authorized to join the network. Once the new node has joined the network, the controller's job is finished until a change is required including the case where a node should be expelled.

However, there is no reason for the new node to establish connections with the other nodes before it needs to communicate with them. Whether the controller is involved in every node-node connection set up or whether it uses something like a revocable list of network members is a matter for implementation.

2.3 Virtualized Networks

The are several ways that virtualizing networks are used to implement zero-trust networks. The basic principle is that access to the virtual network is limited to things authorized to join the virtual network.

2.3.1 VLANs

A VLAN (the origin of the term is a virtual local area network) is a group of devices configured to communicate as though there were on the same LAN segment.[6] As such, the traffic between them is not, and does not need to be, routed. The configuration is not dependent on the actual LAN configuration.

In a physical network, packets are tagged with the identifier of the VLAN. The standard for VLAN tagging for Ethernet-like networks is IEEE 802.1Q and the tag is a 32-bit field added to the packet. The packets are either tagged by the device that sends the packet or by the network component at the boundary of

[6] This means it is a link layer (layer 2) connection.

the VLAN environment (which might be the switch the host is connected to.) Steps must be taken to prevent malicious tagging of packets where a host tags a packet for a VLAN it is not authorized to join.

A Cisco switch that supports VLANs will have several types of port and two of those types are particularly relevant here:[7]

- Access Port – Any VLAN tag in a packet is replaced with the tag for the VLAN that port is configured for. Access ports are used primarily for hosts and can only carry traffic for a single VLAN.
- Trunk Port – The frames received on the interface are assumed to have VLAN tags although there may be a default tag for untagged packets. Trunk ports are for links between switches or routers that carry traffic for multiple VLANs.

Connecting a malicious device to a trunk port will mean it can send packets tagged with whatever VLAN it wants.

2.3.2 Virtual Private Clouds

Amazon Virtual Private Cloud (Amazon VPC)[8] enables AWS resources to be launched into a virtual network that the user has defined. This closely resembles a traditional physical network. An AWS VPC is logically isolated from other virtual networks in the AWS Cloud, meaning that it has the properties of a VLAN where devices connected to a VLAN are isolated from other traffic in the enterprise network.

Communication between AWS VPCs is through a VPC peering connection, and Instances in either VPC can communicate with each other as if they are within the same network. A VPC peering connection can be created between your own VPCs, with a VPC in another AWS account, or with a VPC in a different AWS Region.

When connecting to an AWS VPC using Direct Connect,[9] a unique VLAN tag that's not already in use on the customer side must be used and this tag is required for any traffic traversing the AWS Direct Connect connection.

2.4 Microsegmentation

Microsegmentation can be a form of access control. Here is how Cisco defines microsegmentation:

Micro-segmentation creates secure zones across cloud and data center environments to isolate application workloads from one another and secure them individually. With micro-segmentation, firewall policies limit east-west traffic between workloads based on a zero-trust security approach to reduce attack surfaces, prevent the lateral movement of threats to contain breaches, and strengthen regulatory

[7] As an example, see VLAN membership section of the Cisco 300 Series Managed Switches https://www.cisco.com/c/en/us/support/docs/smb/switches/cisco-small-business-300-series-managed-switches/smb1837-view-vlan-memberships-on-300-series-managed-switches.html

[8] https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html

[9] https://docs.aws.amazon.com/directconnect/latest/UserGuide/Welcome.html

compliance. Micro-segmentation is also referred to as application segmentation or east-west segmentation in a multi-cloud data center.

https://www.cisco.com/c/en/us/products/security/what-is-microsegmentation.html

Microsegmentation can be implemented using VLANs, VPCs, or with granular rules within a firewall particularly one used in a hybrid or multi-cloud environment. Regardless of how they are implemented, the question is how to create them.

By restricting access to the microsegment to authenticated and authorized participants and devices, the microsegment has become a form of access control to the systems on it.

The challenge is mapping out the microsegments that are required.

Hausman[10] states that “locking down application workloads [in microsegmentation] without a deep, thorough understanding of exactly what communications are taking place and how data is flowing could result in failures and outages which will stall or result in the cancellation of the microsegmentation project all together”. In our context, we would say “workflow” rather than “workload”. For Hausman this means starting with a discovery process to map out how applications, services and systems communicate with each other. This is a bottom-up approach – find out what’s happening and construct the microsegment around it.

Another approach, a top-down approach that aligns with CSAP as a workflow managed security architecture, is to make the workflow management the source of truth for how applications, services and systems should/will communicate and create microsegments that support those workflows.

Neither approach is the right one for every situation and one of the factors in deciding which approach to use is how well the workflows are understood. Arguably, in an enterprise the IT infrastructure is general purpose providing a set of resources like storage and email for a broad set of constituents, and the exact way it is used by different departments and business units is somewhat removed from those tasked with provisioning the infrastructure. In a production environment, that gap between resource provider and resource consumer is narrower.

Bottom up will mean that microsegments are configured around what is happening, top down will mean microsegments are configured around what should be happening. So, if you understand the workflows then top down would be the place to start, if you don’t understand the workflows then bottom up is probably the only approach.

On a historical note, microsegmentation could also be achieved with an air-gapped network[11] although an air-gapped network is antithetical to the use of cloud resources.

[10] Starting Your Microsegmentation journey, Christina Hausman, Cisco security blog, 2020, https://blogs.cisco.com/security/microsegmentation-part-i-starting-your-microsegmentation-journey

[11] In the simplest case, an air-gapped network is a network with no connections to any external system that is in a physically secure location where protecting physical access to network ports is the primary method of defense against unauthorized devices attaching to the network.

2.5 The Software-Defined Perimeter (SDP)

Software-Defined Perimeter (SDP) is a network security architecture developed by the Cloud Security Alliance (CSA) that provides security at Layers 1-7 of the OSI network stack (Layers 1-4 of the TCP/IP network stack.) An SDP implementation hides assets and uses a single packet to establish trust via a separate control and data plane prior to allowing connections to hidden assets.

The specification can be found at https://downloads.cloudsecurityalliance.org/initiatives/sdp/SDP_Specification_1.0.pdf

2.5.1 How it works

The primary components of the SDP architecture include the client (known as the Initiating Host or IH), the server (known as the Accepting Host or AH), and the SDP controller. All control channels use mTLS.

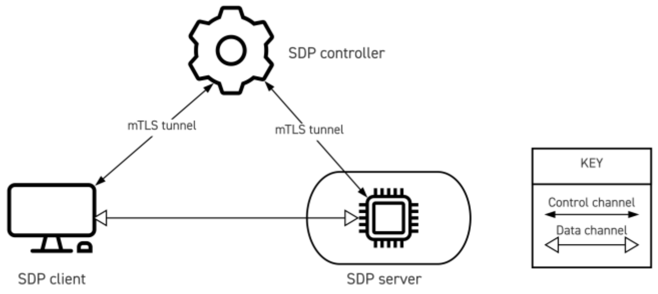


Figure 2-4 SDP Architecture

The operation is straightforward.

1. An SDP controller is added and connected to authentication and authorization services.
2. An AH registers with the SDP controller and is authenticated. The AH does not acknowledge traffic from anywhere other than the SDP controllers it is registered with and will not respond to any non-provisioned requests.
3. An IH registers with the SDP controller and is authenticated.
4. The SDP controller determines which AHs the IH is authorized to connect to.
5. The SDP controller notices those AHs to accept communication from the IH and initiates any optional policy required from encrypted communications.

2.5.2 SPA

One of the most critical elements of SDP technology is that it requires and enforces an “authenticate before connect” model. SDP accomplishes this through single packet authorization (SPA), a lightweight security protocol that validates a device or user’s identity before permitting network access to the relevant system component (e.g., the SDP controller.)

The information for a connection request, including the requester’s IP address, is encrypted and authenticated in a single network message. The purpose of SPA is to allow a service to be darkened via a default-drop firewall. The system should drop all TCP and UDP packets and not respond to those attempts, providing no information to potential attackers that the port is being monitored. After authentication and authorization, the user is granted access to the service. SPA is integral to SDP, where it initiates communication in the connections made between clients and controllers; gateways and controllers; and clients and gateways.

While implementations of SPA may differ slightly, they should share the following features:

1. Packet must be encrypted and authenticated.
2. Packet must self-contain all necessary information; packet headers alone are not trusted.
3. Packet must not depend on admin or root-level access to generate and send; no raw packet manipulation allowed.
4. Server must receive and process packets as silently as possible; no response or verification will be sent.

2.5.3 SDP and CSAP

As with BeyondCorp's trust tiers, SDP can be used to implement a network consistent with CSAP.

Adding CSAP services to Figure 2-4 gives us:

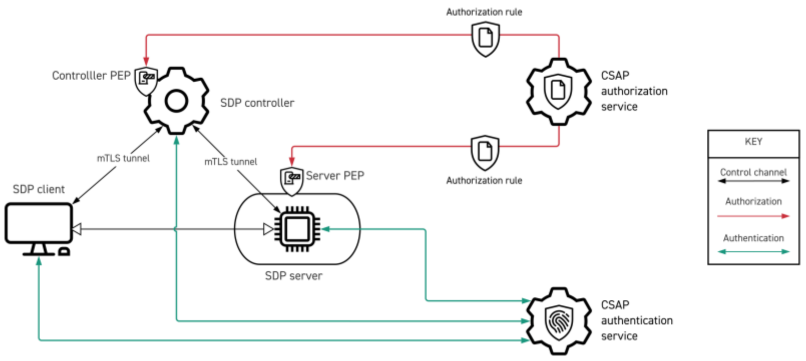


Figure 2-5 SDP architecture with CSAP PEPs

We can now annotate the SDP operation from Section 2.5.1 to show the operation of the PEPs which we show as *underline italics*. Note that the controller PEP would probably be implemented within the SDP controller. The server PEP is a CSAP requirement in any event, and some extra functionality is necessary to support SDP. Note that there is no change to the operation of SDP registration process although there are additional authentication and authorizations.

1. An *authenticated* SDP controller is added and, *if it is authorized*, connected to authentication and authorization services.
2. An AH registers with the SDP controller and is authenticated. *The registration is rejected if the AH is not authorized to register with the SDP.* The AH does not acknowledge traffic from anywhere other than the SDP controllers it is registered with and will not respond to any non-provisioned requests.
3. An IH registers with the SDP controller and is authenticated.
4. The SDP controller determines which AHs the IH is authorized to connect to.

5. The SDP controller notices those AHs to accept communication from the IH and initiates any optional policy required from encrypted communications.

Some policy considerations to be aware of in implementation are:

1. How is authentication and authorization kept current, and how is membership of an SDP revoked?
2. Since the SDP controllers authenticates everything before it can register, is membership of an SDP sufficient to say something has been authenticated?
3. Since the SDP controller determines which AHs an IH is authorized connect to, does that mean that all connections between that IH and those AH are authorized?

2.5.4 SDP and BeyondCorp

Cloud Security Alliance (CSA)[12] says that BeyondCorp differs from SDP in several ways, and we offer this quotation without commentary:

BeyondCorp is a web proxy-based solution that supports HTTP, HTTPS, and SSH protocols. SDP implementations generally support more (and in some implementations, all) IP protocols. SDP also supports more fine-grained access controls than BeyondCorp. In Google's system, applications are assigned to a small number of "trust tiers." SDP supports finer-grained and individualized access controls, driven by user and device context.

2.6 Service Meshes

A service mesh is a way of implementing device and service authentication which are level 100 requirements.

BeyondCorp's proxy solution is a form of service mesh.

Let us define a service mesh.[13]

"A service mesh is a dedicated infrastructure layer for handling service-to-service communication. It's responsible for the reliable delivery of requests through the complex topology of services that comprise a modern, cloud-native application. In practice, the service mesh's implementation is an array of lightweight network proxies deployed alongside microservices, without the applications needing to be aware." William Morgan, https://buoyant.io/what-is-a-service-mesh/

A service mesh is a combination of smart endpoints and dumb pipes. Service-to-service communication is done through a smart endpoint which is a URL that resolves to a microservice and communicates

[12] Software-Defined Perimeter Architecture Guide, Jason Garbis, Juanita Koilpillai, et al. 2019.

[13] William Morgan, the creator of Linkerd and the person who coined the term service mesh wrote, "The service mesh was born in the murky, trend-infested waters of the cloud native ecosystem, which unfortunately means that a huge amount of service mesh content ranges from 'low-calorie fluff' to—to use a technical term—'basically bullshit.' But there's some real, concrete, and important value to the service mesh, if you can cut through all the noise."

using basic layer 7 network protocols such as HTTP, REST, and so on. This approach is opposed to central smart pipe messaging systems such as ESB/MQ.[14]

Obviously, we are describing service meshes in this document because they offer a security model that is highly compatible with CSAP. There however other reasons to use a service mesh, for example they have reliability and observability features.

- A service mesh can add reliability such as requesting retries, managing timeouts, handle traffic splitting/shifting.
  - For example, Linkerd's retries are parameterized with things like retry budgets because the naive approach to retries can be a path to "retry storms" and other distributed system failure modes.
- A service mesh can have excellent observability such as by aggregation of success rates, latencies, and request volumes for each service, or individual routes.

Software-defined workflows are architected as distributed collections of communicating services, with each performing some discrete function in the overall workflow.

A service mesh can secure the service-to-service communications, that are part of software-defined workflows, by proxies paired with each service. As the word proxy implies, communications between services are intercepted by the proxies which process them according to security policies. In CSAP, the proxy is a policy enforcement point.

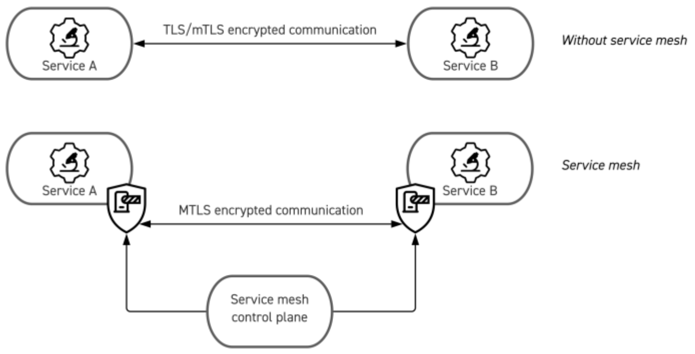

TLS/mTLS encrypted communication
Service A
Service B
Without service mesh
Service A
Service B
MTLS encrypted communication
Service mesh
Service mesh control plane

Figure 2-6 Inter-service communication

The difference between direct inter-service communication, the upper part of Figure 2-6 and the service mesh is that the service mesh provides a security control plane that manages the security data plane so that only authenticated and authorized services can join. The control is exercised through the policy enforcement points attached to each service. Using the policy enforcement point means that no security features need to be built into the service.

[14] See Microservices Guide, Martin Fowler. https://www.martinfowler.com/microservices/

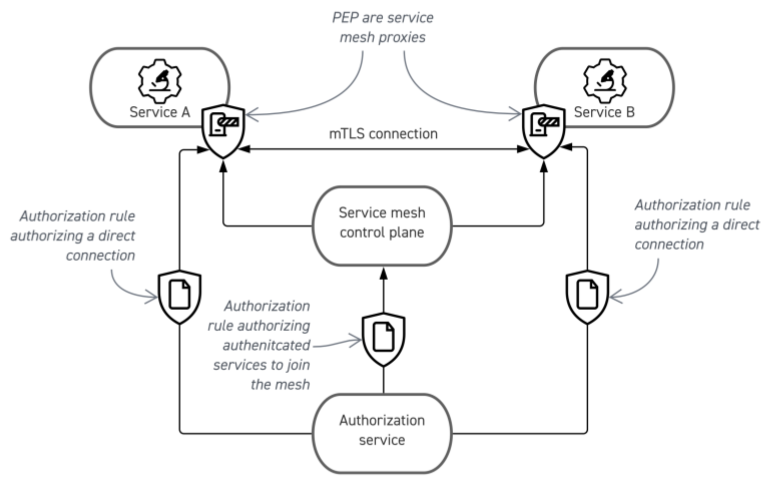

PEP are service mesh proxies
Service A
Service B
mTLS connection
Service mesh control plane
Authorization rule authorizing a direct connection
Authorization rule authorizing a direct connection
Authorization rule authorizing authenitcated services to join the mesh
Authorization service

Figure 2-7 Service mesh and CSAP

The service mesh architecture is implemented by several software products including:

- Istio, https://istio.io/
- Linkerd, https://linkerd.io/
- Consul, https://www.consul.io/
- Kuma, https://kuma.io/
- Traefik Mesh, https://traefik.io/traefik-mesh
- Open Service Mesh, https://openservicemesh.io/

And as services from the three major cloud providers:

- Google Anthos Service Mesh, https://cloud.google.com/anthos/service-mesh/
- AWS App Mesh, https://aws.amazon.com/app-mesh
- Open Service Mesh on Azure Kubernetes Service (AKS), https://docs.microsoft.com/en-us/azure/aks/open-service-mesh-about

When deploying a service mesh, there are three factors important to how the proxies (policy enforcement points) are implemented.

- The data plane proxies must be fast because they add two proxy hops to every call, one on the client side and one on the server side.
- The proxies need to be small and light because they consume memory and processing, and the consumption will scale linearly with the application.
- Deploying and updating proxies must be automated. Doing so manually will not scale.

2.7 SDN and IBN

Software-defined networking (SDN) and the related intent-based networking (IBN) are approaches to networking that we mention for completeness, but we do not regard either as a security solution.

3 Access Controls

Access controls are a level 100 requirement

CSAP must work with systems that have native authentication and/or access controls. By “native” we mean built in the system.

3.1 Access Control Basics

Storage systems implement some form of access controls to restrict access to files and directories only to authorized users. In UNIX, access controls are specified as the right to read, write, and execute for each of an owner ID, a group ID, and all others. In this table, the owner class can read, write, and execute the file, the group class can read and execute the file and others cannot access it.

Owner class	Group class	Other class
`r w x`	`r – x`	`- - -`

This is the most basic form of an access control list (ACL.)

3.2 Access Controls in Cloud Storage

A cloud identity and access management (IAM) system controls access to the resources within a project including storage buckets and objects stored within buckets. The set of access rules applied to a resource is called an IAM policy (not to be confused with a CSAP authorization rule.) An IAM policy applied to a project defines the actions that users can take on all objects or buckets within that project and an IAM policy can be applied to a single bucket defining the actions that users can take on that specific bucket and objects within it.

An IAM policy is defined in terms of roles, a bundle of one or more permissions. Roles are granted to principals (a user account, service account, group, or domain) which allows them to perform actions.

The first step in mapping CSAP authorization rules to cloud storage is the creation of an IAM policy for each of the actions that an authorization rule is likely to authorize. Fortunately, since we are talking about asset storage here, the possible combinations are small since an authorization rule would allow an asset to be read and an asset to be written (including created or updated.)

The second step is creating a principal in the IAM system for each of the participants that the authorization rule applies to. This can be done when the authorization rule is received or preferably ahead of time to reduce the latency in establishing the access permissions. An implementation would define how participants (which might be a user, a group, an organization, a service, etc.) is mapped to principals.

3.3 Asset Protection

An identity and access management (IAM) service can be used to hold participants’ access rights. A participant’s access to an asset can be set by the asset protection service using the IAM system.

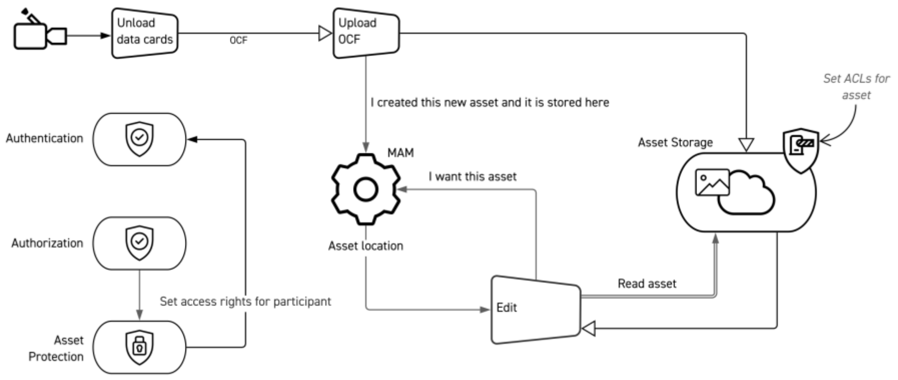


Figure 3-1 Assets protected by ACLs

A participant can be authorized to access an asset either individually or, more likely if ACLs are used to protect assets, through membership of a group. Let's assume that we are using group rights in the ACLs and our edit task is ready to access an asset and they need to be granted access. There are two ways to accomplish that:

- The participant can be given membership of a group that has access to the asset. This method might be used when, for example, a participant commences a new task such as edit starts on a new scene.

Or

- A group that the participant belongs to can be added to the asset's ACL. The second method might be used when, for example, a workflow event changes who can access an asset such as the asset being approved.

In the first method, neither the MAM nor the asset protection service is involved. The asset's ACL have already been set and do not change. In the second method, the asset protection is changing the asset's ACL and it will call on the MAM to locate the assets described in the authorization rule.

If asset protection is by asset encryption, asset protection is done at the point of production (encryption) and consumption (decryption.)

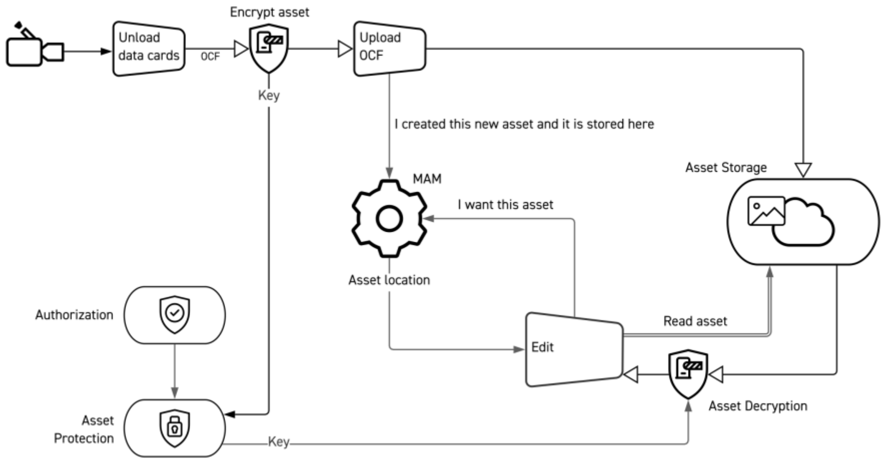


Figure 3-2 Assets protected by encryption

In this case, asset encryption keys are stored in a key repository associated with the asset protection service. See section 4 in the End-to-End Security section.

3.4 SaaS Access Controls

Beyond saying that programmatic/API interfaces will be needed, it is not possible to generalize how to map CSAP authorization rules to a SaaS service's user authentication and access controls because there are many types of SaaS services each with different characteristics, and many ways that a SaaS provider can implement authentication and access controls. Consequently, this section is a series of questions that may help the implementation of a mapping from CSAP authorization rules to a SaaS service:

1. Does the SaaS service support an external authentication service, e.g., an SSO? If not, then a programmatic interface is required to set and remove users in the SaaS service's list of authenticated users and the associated permissions.
2. Are assets and metadata held inside a closed environment that only the SaaS service can access? If so, then a programmatic interface is required to set the SaaS service's access controls so that authorization rules can be mapped to the SaaS access controls.
3. Are access controls on the functions offered by the SaaS service sufficiently expressive to match the authorization rules?
4. If assets or metadata are held outside of the SaaS service, how is access managed? Does the SaaS require an identity it manages?
    a. If so, how can that be reconciled with authorization rules that are specific to a participant?
    b. If not, is it using a user's (i.e., a participants) credentials and if so, what is the minimum level of trust required?

4 End-to-End Security

End-to-end security is a level 300 requirement.

In CSAP, the highest level of end-to-end security means that data is encrypted when it is created and decrypted only by those authorized to consume it. That is the highest level because the attack surface is minimized to the devices managing keys, the systems creating and consuming data, asset management, and the CSAP core security components. In other words, the only things that must be trusted are the things involved in key management and authorization or are a data consumer or producer.

In Section 4.1, we will discuss end to end security in a messaging system as an example; the same concept applies to any movement of data between entities.

4.1 Messaging

Let us look at messaging systems, such might be used in software-defined workflows, as an example of how to approach end to end security. In these diagrams, we use the visual language "thing" shape to avoid assigning a particular view of the implementation to the messaging system, producers, and consumers since they can be manifest in several ways.

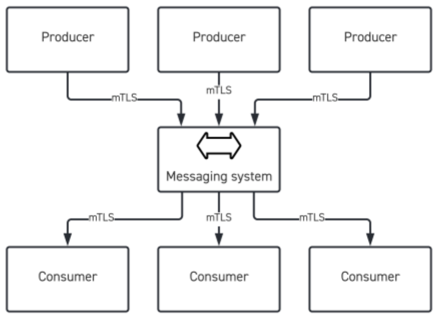


Figure 4-1 Basic messaging system

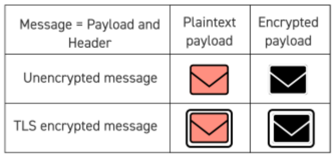
Logically, this type of messaging system is a queue with producers sending messages to the consumers that have subscribed to the queue. The queue is managed by the messaging system and all messages pass through the messaging system. The system receives messages from the producers and sends them individually to the consumers (meaning this isn't a broadcast network in way an Ethernet LAN is where one packet is sent, and everyone receives it.)

How CSAP secures the operation of the message system is dealt with in Part 4 of the CSAP architecture documentation, here we are going one step deeper to look at how security of the messages themselves can be implemented.

As shown in Figure 4-1, messages from the producer to the broker and from broker to consumer are sent over TLS connections where they are encrypted during their flight from one to the other. However, when a message is received by the broker, it is decrypted. TLS is a secure means of communication over an untrusted network, but it obviously does not do anything to protect data once it is received.

4.1.1 Untrusted Message Broker and Cross Domain Messaging.

A message consists of two parts: the header and the payload. The TLS connection between the end points and the broker encrypts the entire message: both header and payload. The payload can also be encrypted separately. We use the following convention for diagrams in this section:



Message = Payload and Header	Plaintext payload	Encrypted payload
Unencrypted message	🟥	⬛
TLS encrypted message	[🟥]	[⬛]

Figure 4-2 Conventions for messaging diagrams

To demonstrate the problem this subsection discusses, let us consider a message sent by one of the producers in Figure 4-1 to one of the consumers.

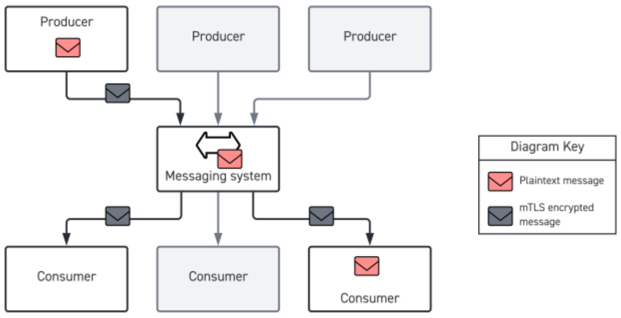


Figure 4-3 A message's journey

The payload isn't encrypted other than when the message is encrypted while it is being sent over the secure connection. Obviously, the producer and the consumer can be trusted with the plaintext payload, but do we trust the broker with the plaintext payload? Do we need to?

Probably the worst case of this is a message sent from one domain to another. In Figure 4-4 there are three mTLS connections (producer to message system 1, message system 1 to message system 2 and message system 2 to consumer) and the message is encrypted only when it is in flight.

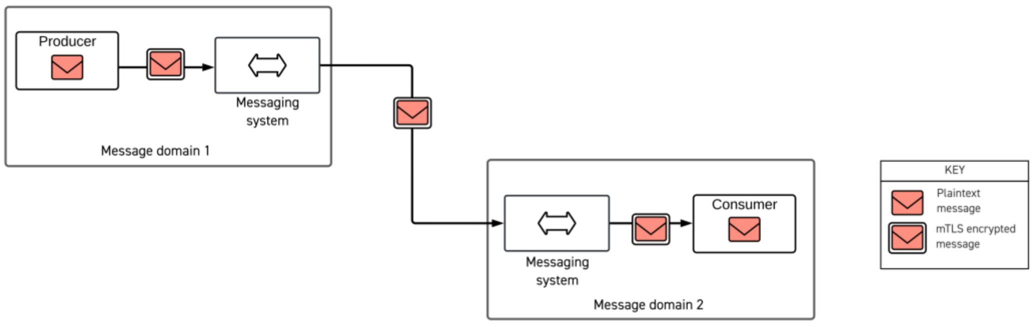


Figure 4-4 Messaging across domains

For this to be secure, either both message systems and any intermediate systems that route messages must be trusted with the plaintext payload or the payload must be encrypted.

End to end security means that the payload of the message is encrypted at the producer and is not decrypted until it gets to the consumer. Like this:

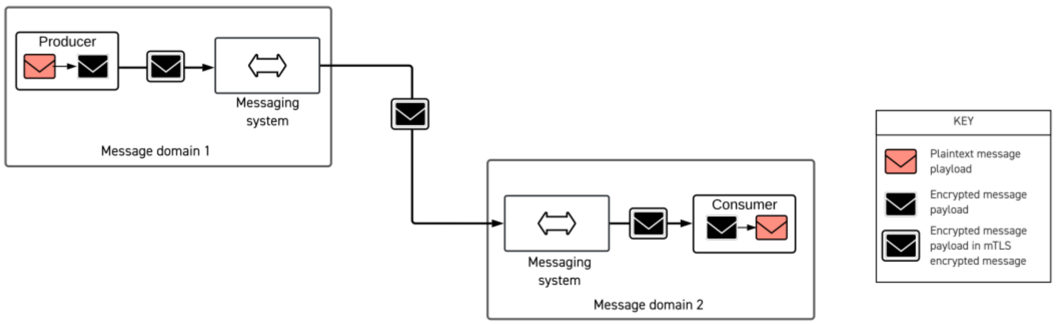

Producer
Messaging system
Message domain 1
Messaging system
Consumer
Message domain 2
KEY
Plaintext message playload
Encrypted message payload
Encrypted message payload in mTLS encrypted message

Figure 4-5 End-to-end encryption across domains

Of course, this also applies within a single domain which means that in Figure 4-3 the message system only sees an encrypted payload.

It is important to note when designing such a system that the entire message cannot be encrypted since fields such as headers used by the messaging systems to route packets must be in plaintext when they are processed by the messaging system.

4.2 Key Management for End-to-End Security

Having decided we want to encrypt payloads end to end, the next problem is that the producer and consumer need a shared key so that the consumer can decrypt the payload in the message from the producer. This requires key management.

One way to implement this is to install a PEP at each consumer and producer. This PEP is a software function inserted between the code in the producers and the consumers that generates and consumes messages, and the code that sends messages to and receives messages from the messaging system.

Messages created by a producer are encrypted by the PEP and then passed to the messaging code. Network stacks are not modified.

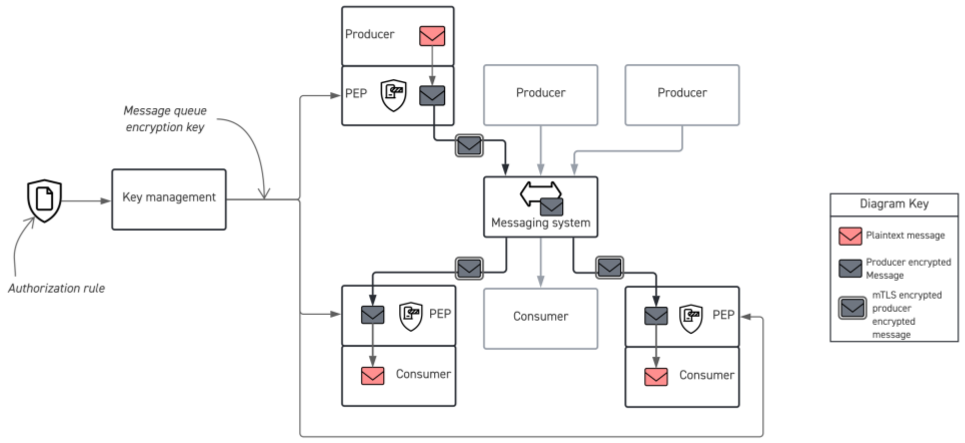


Figure 4-6 Key management for end-to-end security

The key manager creates an end-to-end encryption key for each queue and distributes it to, and only to, the PEPs in the producers and consumers that are authorized to use that queue. It has the information it needs to do that in an authorization rule from the CSAP policy service which is the same authorization rule used by the PEPs that allow consumers to join a queue and producers to send to it.

In this approach to implementation, the messaging system is not modified and does not do anything special to support the end-to-end security.

The key manager can be implemented using CSAP supporting components such as the AWS Key Management Service (KMS). It will create and manage the end-to-end keys required by this approach. Like other cloud provider key management services, AWS KMS is highly secure using hardware security modules.[15]

4.3 Asset Encryption

Local asset encryption is a level 200 requirement.
End-to-end asset encryption is a level 300 requirement.

Let us recap the advantage of asset encryption as it was described in Part 1 of the CSAP documentation.

[15] Statements like that are not enough by themselves. You also need to know who has validated the security of the hardware module. In the case of AWS, it uses "hardware security modules that have been validated under FIPS 140-2 or are in the process of being validated". Source: https://aws.amazon.com/kms/

Traditionally the way assets are protected is by controlling access to the files and storing them on encrypted media. *The Evolution of Production Security* paper points out that the security goal is to protect the asset and not whatever is holding the assets.[16]

CSAP can protect assets individually. Controlling who can access an asset is a better security proposition than controlling who can access the storage that holds it.

Two primary methods of protecting assets are:

- The use of the access permissions supported by the infrastructure (file system, cloud provider, etc.)
- Encrypting the assets with individual keys.

These are not mutually exclusive, and assets can be protected using both methods along their journey.

In Part 1 of the CSAP documentation we showed how asset encryption is very similar to the way DRMs manage encryption of content delivered to consumers. Regardless of whether it is possible to use the encryption function of a commercial DRM, there is mapping between the components of a DRM system and key management in CSAP. This is useful for CSAP implementation because it is a guide to how asset encryption can be implemented.

Function	**DRM**	**CSAP**
Decides who encryption keys are distributed to	Entitlement server	Authorization server
Decrypts the data	DRM component of media player	Policy enforcement point (whether discrete or part of the system or application)
Encrypts the data	Content preparation step[17]	Policy enforcement point (whether discrete or part of the system or application)
Distributes encryption keys	DRM license manager	Key manager
Format for key distribution	DRM license	Authorization rule

In both cases, data encryption functions are within a trusted piece of software. The media player has a DRM module which decrypts the content and outputs it directly to the display hardware. The PEP decrypts and encrypts data as it passes into and comes from the application. The DRM module in the media player has mechanisms for protecting itself. The PEP function, whether integrated into the application and as a layer between the application and I/O, would also need to protect itself.

The next diagram illustrates how asset encryption works.

[16] By "holding" we mean whatever the asset is stored in: local hard drive, USB memory stick, AWS S3, GCP persistent disk, Azure SQL Database, a bread bin in your kitchen, etc.

[17] This assumes content is not encrypted on the fly when it is streamed. For a live broadcast, it would be encrypted on the fly.

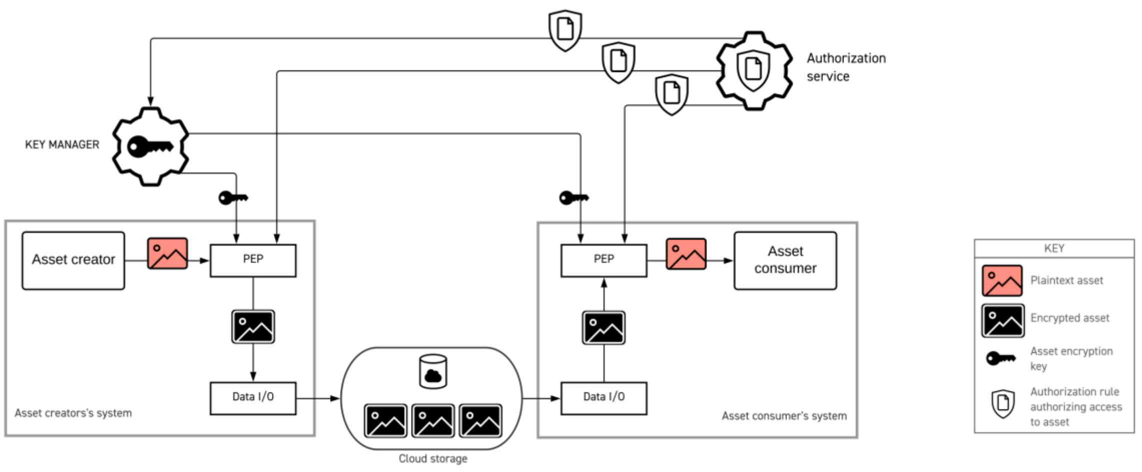


Figure 4-7 Asset encryption

Functionally, this is very similar to Figure 4-6.

Asset encryption raises several implementation issues that need to be addressed:

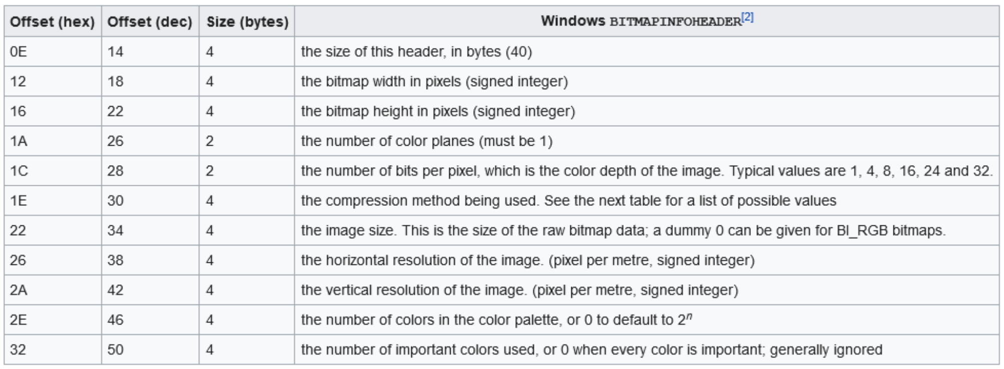
Issue	**Comments**
Key handling	Encryption must be implemented in a way that only trusted software needs to access encryption keys. It may be possible for the key management client to set the keys in, say, a hardware encryption module and then let the application do the encryption and decryption.
Where to implement encryption and decryption	1. In the application – see comment above 2. In the operating system – the driver that access storage contains the encryption code 3. In a security wrapper around an application
Performance	Server and workstation grade processors with hardware assisted AES processing. For example, those that support Intel Advanced Encryption Standard (New Instructions (AES-NI) offer very high performance for encryption operations.
Whether to encrypt intermediate and temporary files	This is a risk management decision. For guidance: if the files are stored in a location inaccessible except by the application, then encryption may not be necessary. If the files are stored in a shared location, then encryption is probably desirable.

Issue	Comments
Using a DRM's key management	The answer would depend on the DRM but it is not expected that this would be the best solution. • Standard DRM licensing terms have compliance robust requirements that are not relevant here • While the authorization model is similar, the mapping is not exact and a DRM based PEP would behave differently than a standard DRM client.

4.3.1 Multi-Part Asset Encryption

Encrypting assets means that a data manager can copy and move the assets without being able to access the contents. That both makes the task of administering ACLs far simpler since an encrypted then asset is useless to someone without the key and means the asset management workflow does not depend on highly trusted individuals in data management.

However, a file used to store an asset contains metadata that allows interpretation of the data in the file. In a simple case, such as a Windows bitmap image, the file contains metadata in the following format:



Offset (hex)	Offset (dec)	Size (bytes)	Windows BITMAPINFOHEADER[2]
0E	14	4	the size of this header, in bytes (40)
12	18	4	the bitmap width in pixels (signed integer)
16	22	4	the bitmap height in pixels (signed integer)
1A	26	2	the number of color planes (must be 1)
1C	28	2	the number of bits per pixel, which is the color depth of the image. Typical values are 1, 4, 8, 16, 24 and 32.
1E	30	4	the compression method being used. See the next table for a list of possible values
22	34	4	the image size. This is the size of the raw bitmap data; a dummy 0 can be given for BI_RGB bitmaps.
26	38	4	the horizontal resolution of the image. (pixel per metre, signed integer)
2A	42	4	the vertical resolution of the image. (pixel per metre, signed integer)
2E	46	4	the number of colors in the color palette, or 0 to default to 2^n
32	50	4	the number of important colors used, or 0 when every color is important; generally ignored

Figure 4-8 Windows BITMAPINFOHEADER header

There may be participants that make use of the metadata, in this case the BITMAPINFOHEADER header, but do not need access to essence (the data or payload part of the file.) In this case we can encrypt the header with one key and the essence with another key.

Participant	Header key	Essence key	Capability
Data manager	No	No	See file and copy it
Metadata consumer	Yes	No	+ can read metadata
Essence consumer	Yes	Yes	+ can read essence

A bitmap image is a "flat" format, the data is interpreted as the individual pixels of the image. Many file formats used in production are more complicated.

JPEG 2000, ISO/IEC 15444, is an image compression standard and coding system. Its compression process decomposes the image into a multiple resolution pyramid representation.

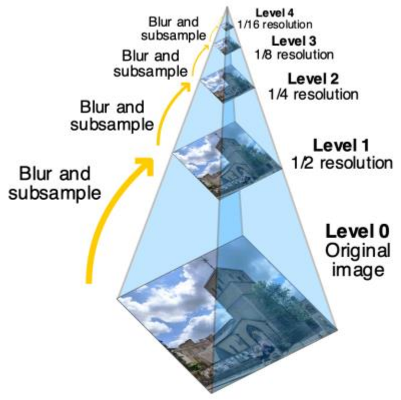


Figure 4-9 Image pyramid representation[18]

JPEG 2000 provides efficient code-stream organizations which are progressive by pixel accuracy and by image resolution (or by image size.) In the case of downloading a JPEG 2000, this means that a lower quality version of the image can be constructed without the complete file having been downloaded. The quality then improves progressively as more data is downloaded.

This property of JPEG 2000 means that by encrypting subsequent levels with different keys than earlier ones, the maximum resolution of the decrypted image can be controlled. Thus, a user or process that only needs a ¼ resolution image is given a key that decrypts the data in the file up to the point where the ¼ resolution image is available. The rest of the file can be encrypted with a different key(s).

Another example of this is OpenEXR, an open-source high-dynamic-range floating-point image file format for high-quality image processing and storage. One feature of OpenEXR is multi-part files:

An OpenEXR file may contain multiple independent images or "parts" with different sets of image channels, resolutions and data compression methods.

Multi-part files are useful when an image contains a large number of channels, but file readers are expected read only a subset of those channels at a time. Placing each such subset in a separate part speeds up file reading, since channels that are not needed by the reader are never accessed.

See the technical Introduction to OpenEXR, Florian Kainz, Rod Bogart, Piotr Stanczyk, Industrial Light & Magic, Peter Hillman, Weta Digital. Updated November 5, 2013

A multi-part OpenEXR file could therefore be encrypted using different encryption keys for different parts. Other features of OpenEXR including deep images and the ability to store additional data can also be encrypted in this way.

[18] Image credit: By Cmglee - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=42549151

There are also file formats that are wrappers for the individual components of an asset (usually an asset that contains both picture and sound.) The Material eXchange Format (MXF), defined by SMPTE-377M, is a file format that encapsulates video, audio, metadata and other bit streams (essences), which are audio-video content that can be continuously decoded through mechanisms such as interleaving essence components with stream-based metadata.

Here again, different encryption keys could be used for each stream.

In this section we have described the concept in terms of the use of different encryption keys however to generalize it, what is required is a multi-part encryption method that allows distribution of a key or key set to participants that allows them to decrypt the component they are authorized to access and no more.

5 Conclusion

In this document we have examined some of the approaches to implementing CSAP using existing technologies and/or proven solutions. Our goal is to assist you in discovering some of the approaches available and we hope you find it useful. However, nothing in this document should be taken as a recommendation and this document does not provide, by any metric, a complete list. Every implementor must make their own decision as to what they choose to use.