CONFLUENT

Confluent E2E Encryption Accelerator

Pascal Vantrepote, Chris Urban, and Paul Earsy

Table of Contents

Introduction

Most organizations today are leveraging cloud computing or have plans to do so in the near future. The primary business drivers of cloud-computing adoption are cost savings and increased agility. A leading reason organizations have hesitated to adopt cloud technologies or move more of their workloads to the cloud is security, in particular data protection, i.e. how to properly secure sensitive data. This concern is driven by a number of factors:

- Compliance with data protection regulations, for example: General Data Protection Regulation (GDPR), Health Insurance Portability and Accountability Act (HIPAA), Schrems II Judgment, California Consumer Privacy Act (CCPA), and Brazilian General Data Protection Law (LGPD)
- Contractual obligations with customers and business partners
- Internal policies intended to protect revenue by guarding brand reputation and limiting liability

Encryption for Data Protection

Many data protection regulations require generating *ciphertext*, or encrypted data, as part of data protection controls. Encryption is the process of taking raw data, often referred to as *plaintext* in cryptography, combining it with a data encryption key, which depending on the encryption algorithm may require some additional parameters, and feeding it through an algorithm that outputs ciphertext. Provided that one uses a strong encryption algorithm such as AES (Advanced Encryption Standard), the original plaintext cannot be derived from the resulting ciphertext by a human or computer without access to the encryption key. This is why it is important to protect encryption keys and limit their access using a proper key management service (KMS). A KMS provides management of a cryptographic key throughout its lifecycle, including its secure generation, storage, distribution, use, and destruction [see NIST 800-57]. Without a secure key management system, key material can be compromised, which makes the use of strong encryption algorithms and larger key lengths moot.

There are three categories of data protection controls that use encryption:

- Controls for data in motion (also referred to as *data in transit*)
- Controls for data at rest
- Controls for data in use (often referred to as *end-to-end encryption*)

Data-in-Motion Protection

One of the most widely adopted methods of data-in-motion protection is TLS (Transport Layer Security), often referred to as *SSL* (Secure Sockets Layer). SSL has been deprecated in favor of the

more secure TLS, but in conversation the acronyms are often used interchangeably: Where people use SSL, they should use TLS. The data-in-motion protocol, TLS, protects data as it traverses networks and is something we all use everyday in our web browsers and other Internet applications. The "https" in front of the URLs we use is the acronym for "Hypertext Transfer Protocol Secure," a protocol that uses TLS to encrypt the communication between the client web browser and the web servers. TLS is considered a fundamental requirement, and most public and private/internal networks require the use of TLS to protect their data. Confluent supports TLS for protecting the data between components, and Confluent Cloud mandates TLS (there is no mechanism to disable it).

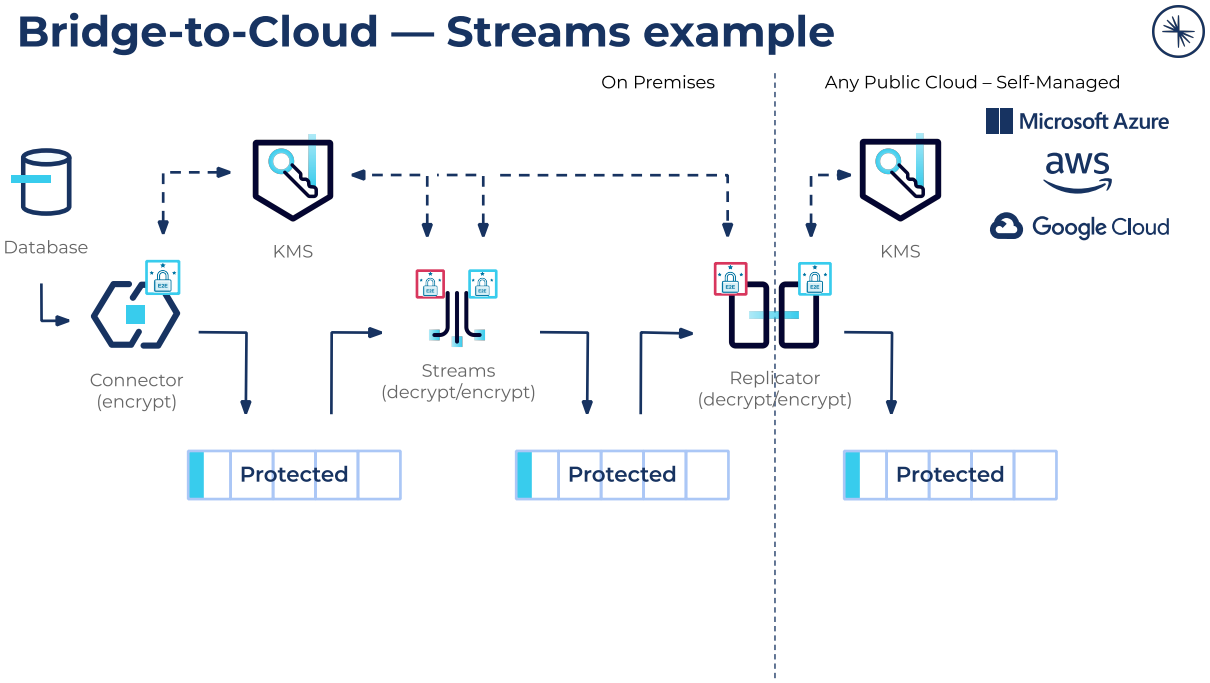
Data-at-Rest Protection

Data at rest is data that is written to persistent storage in a computer. The local hard drive on your computer, a file server that your team uses, the storage used by your cloud compute instances, or the file system that a database uses to store data are all examples of data at rest.

It is considered a best practice to encrypt the underlying storage associated with your Confluent deployment (e.g., the Kafka log files on your Kafka brokers). By default, Confluent Cloud provides data-at-rest encryption for customer clusters, using the storage-level AES-256 encryption mechanisms of the respective cloud providers (AWS, GCP, and Azure). For a Confluent Cloud dedicated cluster on AWS or GCP, Confluent provides the option of using a customer-provided key, which is also referred to as *BYOK*, or "bring your own key." Azure bring-your-own-key (BYOK) support will be available soon. See Encrypt Confluent Cloud Clusters using Self-Managed Keys for additional information.

Data in Use: Bridge to Cloud

For organizations that have adopted cloud computing or are actively planning to, a common use case is "Bridge to Cloud." When first adopting the cloud, sometimes an organization's applications can run independently there, but oftentimes they still need data from an on-premises data center. So existing applications can slowly migrate to the cloud, but they need a strategy, as well as the technology to enable a multiyear migration. The Confluent End-to-End Encryption Accelerator provides the data protection required for organizations to leverage the cloud in a secure manner, while maintaining on-premises source systems. This is achieved by encrypting or tokenizing sensitive data as it moves to the cloud.

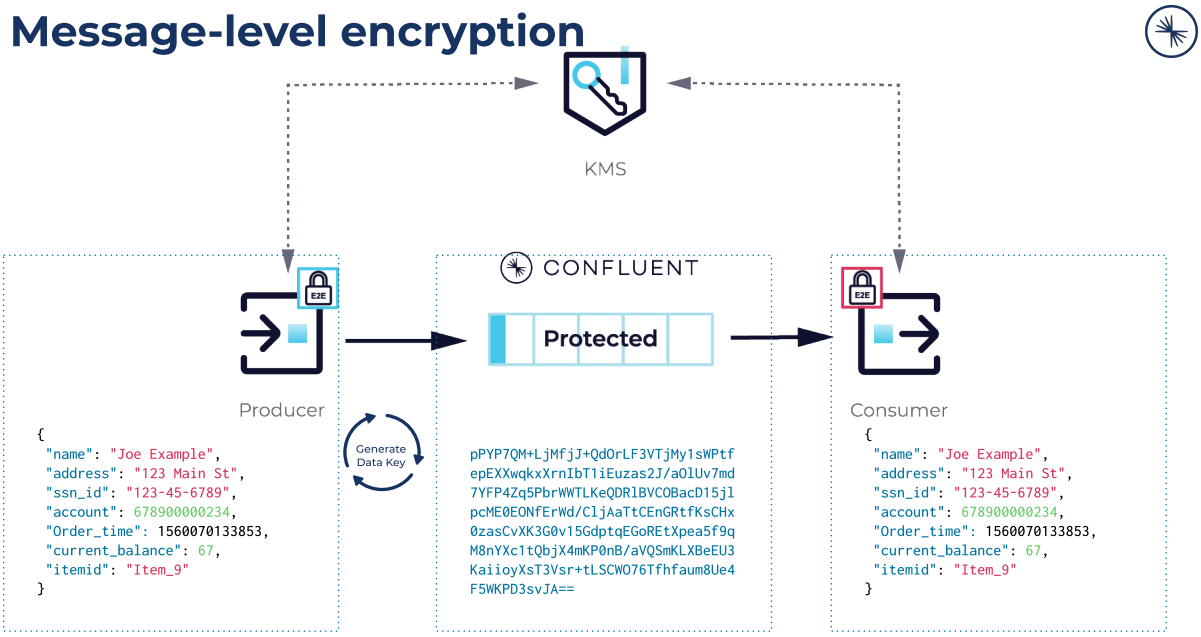


How the Confluent Solution Works: Message-Payload Encryption

The only requirement for using Confluent end-to-end (E2E) message-payload encryption is the Confluent E2E Encryption Accelerator. Libraries are installed for the Confluent components, services and applications that interact with topics containing encrypted messages, which includes connectors, ksqlDB, Replicator, and applications producing or consuming messages—including Kafka Streams applications. The Confluent Encryption Accelerator includes its own serializers and deserializers, which encrypt or decrypt as well as serialize or deserialize messages. Thus this solution supports existing Kafka applications and components without significant code changes for enabling encryption, and also integrates within components (e.g., connectors, ksqlDB, and Replicator), with no code changes. While Schema Registry is optional for message-payload level encryption, it should be considered for managing and sharing E2E configurations across Confluent components and applications, and provides the added benefits that come from leveraging a schema registry.

In addition to making the Confluent Encryption Accelerator libraries available to the components and applications producing encrypted messages, the components and/or applications need to be configured to specify that the payload is to be encrypted. The value converter, cipher provider details, and encryption keys to use should be indicated. When the producing applications or components are started, they will use the specified keys to produce encrypted messages, along with encryption metadata, which

is stored in the message headers. The metadata contains details about the keys used for encrypting the message payload (key name, version, etc.). Consumers of the encrypted topics that need to access the message payload in the clear will use this metadata for decrypting the data, provided they have access to the encryption keys.



How the Confluent Solution Works: Field-Level Encryption

In addition to full message payload encryption, the Confluent E2E Encryption Accelerator also supports encryption or tokenization at the field level for additional granularity. This approach provides the ability to use different policies depending on data classification and it extends additional protections against sensitive data inadvertently being presented in the clear.

Consider that an application that is consuming a topic with message-level encryption has to decrypt entire message payloads if it needs to process the data in the clear. Thus if this application has had its log level increased to *debug* or *trace* for troubleshooting purposes (or by human error), potentially the entire payloads are being logged in the clear to a file or other destination—including sensitive data fields. This risks the possibility of a data breach if the sensitive cleartext data were to get into the wrong hands. However, when implementing encryption or tokenization at the field level, only those applications or systems with access to the keys or tokenization systems associated with sensitive fields would then

have access to the sensitive fields in the clear for processing. Not only does this help to minimize the potential for leaking sensitive data in the clear, but many applications may only need access to a single field or handful of fields, further limiting exposure of sensitive data.

Aside from doing the right thing and protecting your end customer's sensitive data, one of the benefits of encrypting or tokenizing data is that many data protection laws have "safe harbor" language where you do not have to disclose/notify in the event of a data breach if your data was properly protected using encryption or tokenization, and thus you minimize your liability and associated costs.

With the Confluent solution, data is only returned to the clear when absolutely necessary, i.e. when there is a business processing requirement to work with that data. An example of this would be if a customer service representative needed to verify a customer's card number without viewing it on the screen. In this case, the application that the customer service representative is using would require access to return the original plaintext value. If the data is encrypted, it would need access to the encryption key, or in the case of tokenization, access to the tokenization system. Even with this use case, working with the data in the clear is potentially unnecessary, as the customer service representative could input a card number (or partial card number) provided by the customer, and that could be hashed and verified by comparing against the stored hash value of the card. Whenever humans interact with data in the clear, these interactions should be logged for audit purposes, so that in the event of a breach/leak, it can be traced back to "insiders"—who may have purposefully or inadvertently shared sensitive data.

Hashed values of sensitive data, for example a national id or a social security number, can be helpful when performing operations like joining two different data sets contained in distinct Kafka topics. They can be used to alleviate potentially expensive operations from a data processing standpoint, especially with larger event volumes, where billions of attributes may need to be decrypted or detokenized on a daily basis. Note that to achieve this potentially expensive join, you want to make sure to use the same encryption or tokenization scheme on both sides of the fields required to satisfy the join.

How the Confluent Solution Works: Field-Level Tokenization

Another alternative to encrypting individual fields or attributes is tokenization. Tokenization is a data protection approach in which the original sensitive data element is replaced with a non-sensitive, arbitrary string called a ***token***. The tokenization system maintains a mapping between the token and the original value, which would be protected in the tokenization system, typically using encryption. As with encryption, there is no mechanism to derive the original plaintext value from the token. Tokenization systems often have additional options so that the token does not break downstream data validations (in which case the token is not a completely arbitrary string). For example, for PCI use cases, the token that replaces the original credit card number will pass downstream validation checks for properly formatted card numbers.

Local Versus Remote Crypto Operations

When using the Confluent E2E Encryption Accelerator, some KMSes support offloading crypto operations, with the encrypt or decrypt operations performed on the KMS. Many hardware security modules (HSM) support this capability, where the plaintext data is securely transferred over the network to where the key is securely stored on the HSM to be encrypted. Conversely, the ciphertext data is brought to where the key is securely stored to be decrypted. Because the data encryption keys never leave the appliance, this model provides the highest level of protection. (Note that if this is the policy for the key, a caveat is that keys may be securely replicated and/or backed up for recovery purposes, etc.)

There are some considerations when deciding to use crypto offload, the first and most important is whether your KMS or HSM supports the feature. Another consideration is latency: The KMSes (or HSMs) that support this feature are typically highly performant, provided they are properly sized for the volume of crypto operations, so network latency will probably be the primary consideration. Yet another is availability—the KMS will need to be highly available and network connectivity will need to be reliable.

The alternative to performing the crypto operation remotely is to do it locally, where the key is securely transferred from the KMS to the client by the Confluent Encryption Accelerator library, and the crypto operations are performed by the client with the symmetric data encryption key (DEK) cached in memory. The secure transfer of the data encryption key is facilitated by "wrapping" the DEK using an asymmetric key, which is referred to as a *key encryption key* (KEK).

Confluent also supports performing the crypto operations locally using a data encryption (DEK) symmetric key generated by the Confluent Encryption Accelerator at run time and used for encrypting either the entire payload or specific fields within the message (AES-256 is the default, but it is configurable if a different strength symmetric key is required). The client-library generated DEK is wrapped with an asymmetric key, which was previously created on the KMS and stored as a new attribute in the message/event associated with the encrypted data. You can specify an interval to generate new data encryption keys as per your organization's security policy. For consumers of events/messages with encrypted payloads or specific attributes that need to access the sensitive data in the clear, the encryption library associated with the consumer will decrypt the DEK(s) associated with the message/event using the private KEK, and use the DEK to decrypt the encrypted message payload or attributes.

Confluent Cloud Security Controls

For details on the security controls available for Confluent Cloud, please take a look at our esteemed-

colleague Peter Gustafsson's blog and associated white paper: An Overview of Confluent Cloud Security Controls. Peter provides an excellent overview of Confluent Cloud and goes into details on the following Confluent Cloud security topics:

- Internal Confluent Cloud infrastructure security
- Internal Confluent Cloud service security
- Customer security controls
- Business continuity and disaster recovery
- Support coverage
- Compliance
- Trust and security program

Take Action

If the End-to-End Encryption Accelerator sounds like something that could benefit your organization, reach out to your Confluent Account team if you are an existing customer, or click on Contact Us to get more information.

Confluent Customer Solutions & Innovation Division: Accelerators

An Accelerator is software developed and refined by Confluent's Customer Solutions & Innovation Division. Accelerators function alone or in concert with Confluent and they provide the following capabilities:

- Secure and protect Kafka Clusters (e.g., the End-to-End Encryption Accelerator discussed in this white paper)
- Migrate and upgrade from legacy systems
- Improve Kafka consumer efficiency
- Assist with cluster planning and operations

More information on the Confluent CSID Accelerators can be found here.

Resources

Kafka expertise from the inventors of Kafka. Start your event streaming journey with Confluent. For more information, please visit confluent.io or contact us at info@confluent.io.

- Confluent CSID Accelerators
- Confluent Cloud Security Addendum
- Confluent Cloud DPA
- Confluent Cloud Free Trial – Sign Up
- Streaming Resources – Apache Kafka Resources, Tools, and Best Practices

Confluent, founded by the original creators of Apache Kafka®, pioneered the enterprise-ready event streaming platform. With Confluent, organizations benefit from the first event streaming platform built for the enterprise with the ease of use, scalability, security, and flexibility required by the most discerning global companies to run their business in real time. Companies leading their respective industries have realized success with this new platform paradigm to transform their architectures to streaming from batch processing, spanning on-premises and multi-cloud environments. Confluent is headquartered in Mountain View and London, with offices globally. To learn more, please visit www.confluent.io. Download Confluent Platform and Confluent Cloud at www.confluent.io/download.