**Data in Transit**

***Channel Encryption***

Channel encryption is a process of encrypting transmitting data in client/server communications. This security layer lies between application and transport layer, most common TLS and HTTPS. A channel encryption restricts overhearing and tampering in data communication. Most of the common channel encryption are HTTPS and TLS.

* **Transmission Layer Security 1.2**: The TLS is the most widely used protocol for implementing cryptography in web. A TLS provides secure communication channel to traditional TCP/IP protocol suite.
* **TLS (server-side):** Transport Layer Security (TLS) is an advanced version of Secure Socket Layer (SSL). It’s a cryptographic protocol, which is designed to provide secure communications over a computer network.
* **mTLS (server/client mutual auth):**Mutual Transport Layer Security (mTLS) is a process that establishes an encrypted TLS connection in which both parties use X.509 digital certificates to authenticate each other. MTLS can help mitigate the risk of moving services to the cloud and can help prevent malicious third parties from imitating genuine apps.

Table: TCP/IP Protocol stack with TLS

|  |  |
| --- | --- |
| **TCP/IP Layer** | **Protocol** |
| Internet layer | IP |
| Transmission Control Protocol | TCP |
| Transport Security Layer | TLS |
| Application Layer | NNTP, HTTP, FTP & Telnet |

TLS uses public-key cryptography to provide authentication, and secret-key cryptography with hash functions to provide for privacy and data integrity.

The reason behind the TLS use is that various cryptographic algorithms are being involving in the protocol. TLS uses secret-key and public-key cryptography to provide privacy and data integrity and, authentication respectively.

**SECRET KEY ALGORITHM**

In secret key algorithm, sender and receiver should have same secret key to perform encryption and decryption. Before start sending message over a secure communication channel, secret key must be available at both parties to do encryption and decryption. The below section depicts sample encryption and decryption flows.

1. **Encryption Flow**
2. Obtain an ASCII values of the plain text
3. Generate binary for given text.
4. Calculate 1’s complement and reverse the number
5. Take any four digits as a key e.g., 1011 and then divide the reserved number with chosen key
6. Obtain and store quotient and remainder in first 5 and last 3 digits. If those are less than 5 and 3 digits, then add sufficient number of 0’s. The result would be the cipher text.
7. **Decryption Flow**
8. Obtain first 5 digits and multiply with key and add last 3 digits cipher to result
9. Reverse the output and then get the 1’s complement of it.
10. Identify the decimal equivalent to get the actual ASCII, which is plain text.

As the above algorithm uses same key for both encryption and decryption, it is called as symmetric algorithm. Well-known cryptographic algorithms are Advanced Encryption Standard (AES), Rivest Cipher 4 (RC4) and Triple Data Encryption Standard (3DES).

For more details on Experian Policy and TSB – [Experian policy and TSB](https://experian.sharepoint.com/sites/GlobalSecurityOffice/SitePages/Home.aspx) [🔗](https://experian.sharepoint.com/sites/GlobalSecurityOffice/SitePages/Home.aspx)

For data in transit, ascend shall leverage hybrid encryption (a combination of asymmetric key and symmetric key). The keys that shall be used for this purpose are

* PGP-based (for legacy support) OR
* Combination of asymmetric key and symmetric key

***PGP Encryption Algorithm***

**Encryption Flow**

1. For data being sent as files from Ascend to Client (external 3rd party or internal systems at Experian)
   1. A secure channel, e.g. SFTP, is used for data transfer
   2. Client sends their PGP public key to the Ascend authorized representative
   3. The file(s) are encrypted with the shared Client-specific PGP public key
   4. The file(s) are either pushed to a pre-authorized Client location or pulled by the Client from an authorized Ascend location (S3 bucket) over the secure channel.

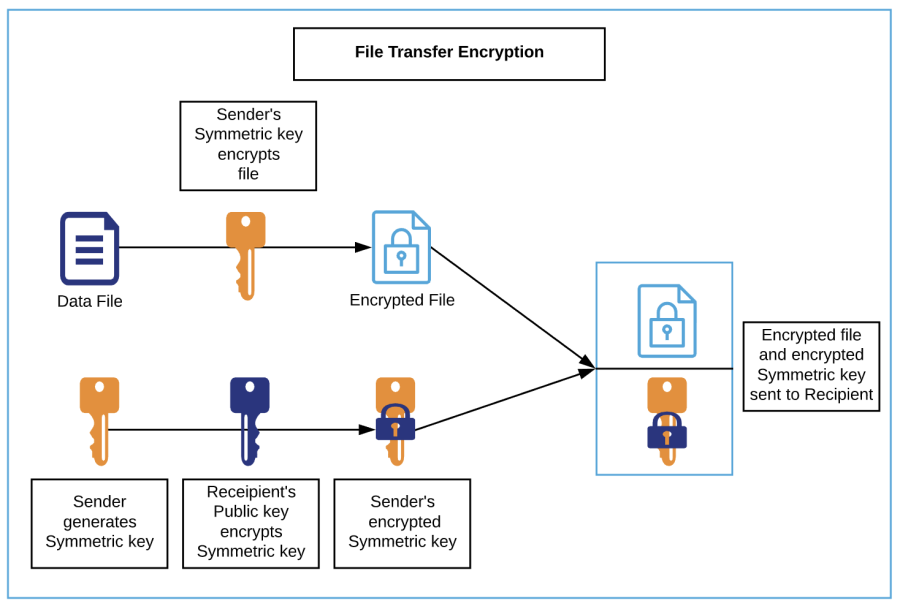
**Decryption Flow**

1. For data being sent as files from Client to Ascend
   1. A secure channel, e.g. SFTP, is used for data transfer
   2. Ascend generates PGP key pair and stores PGP private key in AWS Secrets Manager
   3. Ascend sends their PGP public key to the client authorized representative
   4. Client encrypts file(s) with Ascend PGP public key
   5. Client sends file(s) to the pre-authorized Ascend location in AWS.
   6. On receipt of encrypted file(s) at the authorized Ascend location, the file can be decrypted with the Ascend PGP private key (held in AWS Secrets Manager).

***Hybrid Encryption (combination of Asymmetric and Symmetric key)***

**Encryption flow**:

1. For data being sent as files from Ascend to Client (external 3rd party or internal systems at Experian) OR Client to Ascend
   1. A secure channel, e.g., SFTP, is used for data transfer
   2. Recipient generates Asymmetric key pair, secures private key and sends their public key to the sender's authorized representative
   3. Sender generates a Symmetric key
   4. Sender encrypts file(s) with Symmetric key
   5. Sender encrypts Symmetric key with Recipient's public key
   6. The file(s) and the encrypted Symmetric key are sent to the Recipient to the pre-authorized location.



**Decryption flow**:

1. For data received as files by Ascend from Client (external 3rd party or internal systems at Experian) OR Client from Ascend
   1. A secure channel, e.g., SFTP, is used for data transfer
   2. Recipient decrypts the encrypted Symmetric key with their private key to extract the Symmetric key
   3. Recipient decrypts file(s) with the Symmetric key.

***Payload Encryption (combination of Asymmetric and Symmetric key)***

This scenario shall apply when APIs are used for data exchange. The payload to be encrypted could be a record made up of multiple fields or select sensitive fields within the payload.

The data communication between client and Experian is always encrypted to secure the content using SSL/TLS, which is being transmitted across the networks. Alongside of this, some of the Experian services uses end-to-end payload encryption to secure the payload. For example, if some services using sensitive data like PII must adhere to the company standards and requirements.

***Encryption Flow***

Following are the steps for encryption:

1. A session key is generated along with encryption parameters
2. PII or sensitive data are encrypted using the key.
3. Session key encrypted using recipient’s public key.
4. Session key is sent along with payload and remaining parameters

Diagram

Description automatically generated

**Decryption Flow**

Following are the steps for decryption:

1. The session key will be decrypted using receiver’s private key
2. PII or sensitive data will be decrypted using session key.

**Data in use**

**Protection of Data in Use**

Securing sensitive data is an important and essential, when it’s in use. Usually, there will be number of people, who can access and alter the sensitive data with elevated privileges. In this state, data is more vulnerable than usual because data would be in decrypted state.

In order to protect the data in use, access control mechanism need to be implemented before accessing data. There are some mechanisms: Identity management tools and Role based access control tools. These mechanism alone cannot protect the database or document by limiting who can and can’t access the content. For instance, once an user accessed a content with unauthorised access we cannot prevent the person what they want to do with that data.

We can use digital rights management or protection to control the rights such as preventing downloading, only viewing, downloading, etc. However, if malicious user has downloaded document, it is considered to be highly vulnerable. If we apply digital right management protection directly on any file or document, we can apply the protection that will be carried over and limits the users with specific permissions. Even if data is in cloud or downloaded, we can allow only specific users with elevated privileges. These mechanism allows us to secure the data with some vulnerability. In order to eliminate them, we use homographic encryption standard.

***Homomorphic encryption***

Traditional encryption systems are not adequate to safe guard the privacy of sensitive data in cloud servers. Homomorphic encryption standard is an unique encryption mechanism that protect the security and privacy of target data. In homomorphic encryption scheme, there are four procedures that includes evaluation algorithm shown in Fig 1.

Diagram

Description automatically generated

Fig 1. Homomorphic encryption

The homomorphic encryption allows the users perform the operations on encrypted data without decrypting it. This is the unique feature and ensures the security and privacy of the sensitive data. In homomorphic encryption, user encrypts the query and stores somewhere in the cloud and then sends encrypted query information to server as an input. The server runs the prediction algorithms on stored encrypted data using homomorphic encryption without any knowledge on data. The server returns encrypted prediction result to the user and user decrypts the received data using their secret key. In homomorphic encryption, operation on plaintext during encryption process is similar to another operation performed on the encrypted data.

**Homomorphic encryption Flow**

1. Run key generation algorithm for secret key, public key and evaluation key.
2. Encrypt the plaintext P with public key and generate encrypted text i.e., cipher text
3. Decrypt the cipher text with secret key and obtain original data as an output.
4. Run evaluation algorithm using evaluation key and generate the output.

The homomorphic encryption can be applied on different levels of data because of its ability to perform computation on encrypted data. A homomorphic encryption is using in different areas such as Financial sector, forensic application, medical applications, social networking , smart vehicles and healthcare, where users privacy will be maintained and protected.

**Use of isolated compute environment.**

Experian using AWS nitro enclaves to secure sanative data when data is in use. Nitro enclave is an isolated environment, where users can run their applications securely without any involvement from the platform. This allows us to use the data securely in various situations, especially when we deal with sensitive data.

***Enclaves workflow*** [[1]](https://docs.aws.amazon.com/enclaves/latest/user/flow.html)

This section explains the workflow of AWS nitro enclaves using Simple Storage Service(S3) as storage service and KMS as key management service. Here, we have some involved parties in the workflow. The parties are data owner, application developer and Parent instance administrator. *Data owner* is owns sensitive data and KMS key. Data owner is responsible for creating keys using KMS, encrypt the data with key and make encrypted data available. The *Parent instance administrator* is responsible for manging the parent instance and nitro enclaves steps. Initially, this party launches an instance and creates an enclave using docker image. Here, docker image or enclave image file will be provided by application developer. However, the parent administrator must not have permission to perform any cryptographic operations using KMS key, which has been created by data owner and not allowed to modify the KMS key policy. An *Application developer* is responsible to run applications inside the enclaves on the parent instance. The developer binds the application and enclave using docker file or enclave file and gives it to parent administrator. The parent administrator uses the enclave file and create an enclave.

***Data encryption flow***

This section provides details of encryption process, attestation setup and enclaves creation in detail.

1. Create a key using AWS KMS.
2. Encrypt the plaintext using generated AWS KMS key.
3. With client-side cryptographic library and plain text data key, sensitive data need to encrypted.
4. Store cipher text and encrypted data key in amazon S3.
5. Detect an enclave application. This is preconfigured enclave package, where users can run new application or existing applications.
6. Check security properties of an application enclave, if is satisfied with application security properties, then bind the application into enclave file.
7. Launch parent instance and run the enclave image.

***Data decryption Flow***

This section provides an overview of how data decryption happening in the nitro enclaves.

* 1. Download the cipher data and encrypted key from storage into parent instance.
  2. Use vsock to transfer the downloaded data into enclave.
  3. Compare PCRs data, if result is positive then encrypted plaintext data key sent to enclave using vsock.
  4. Using enclave private key, the encrypted plaintext data key decrypted.
  5. Plaintext data key used to decrypt the encrypted data.
  6. Now, actual data is available to process inside the enclave.

Reference:

1. [Enclaves workflow - AWS](https://docs.aws.amazon.com/enclaves/latest/user/flow.html)

**Data at Rest**

Product use cases and policies

**Data transfer and disk-level encryption (PII data)**

This scenario is applicable in the scenario where PII data is part of the payload that is being transferred. The sequence of the data flow would include:

1. Using PGP public key to encrypt and transfer the payload to an Ascend landing zone (S3 bucket)
2. Decrypt the payload with the PGP private key
3. PII Data - Transfer and Storage Store the decrypted payload in an S3 bucket encrypted with Experian keys.

Diagram

Description automatically generated

***Database encryption***

A company or any organization storing large data in on-premises or cloud need to be secured with effective database security. This is an essential and important for any businesses across the global to have smooth operations. Basically, this security doesn’t go beyond the basic access control mechanisms and managed services. This is not sufficient to protect the sensitive databases with basic security measures alone. With improper planning of sensitive data maintenance leads to data at high risk. Such that businesses need to maintain robust security measures for the sensitive data are smart to enable more safeguard mechanisms that can protect from external and internal threats: database encryption.

**Database encryption working flow**

A database encryption is a process of converting readable database into ciphertext of unreadable databases. By using key and algorithm, an user can decrypt and perform the operations like, retrieving and modifications. This is an essential when system was breached, the data would be in unreadable and only legitimates can decrypt with corresponding keys.

There are various ways for implementing database encryption standard with varied length of keys. Different databases we can find in the market- SQL, Access, Oracle, etc., each of these offers different database encryption standard for their customers to safeguards the stored data. To prevent the unauthorized access through the key computation, keys must be longer than usual keys. There are various levels to implement an encryption between an application to database engine. Most of the service provider ask customers to select an encryption standard to secure sensitive data with transparency. Transparent Data Encryption is the best for database encryption standard and most of the customers opting to use it.

Transparent Data Encryption (TDE): TDE is used to encrypt the stored data on our database instance, where database engine running like oracle, Microsoft SQL server, etc. TDE automatically encrypt the data before it is stored and automatically decrypts the data before it retrieved from storage.

**Which types of database encryption are most secure?**

Service provider gives a freedom to select cryptographic algorithm among available to encrypt their database. The following are some of the encryption standards offers various levels of protection for the database security.

* AES: Advanced Encryption Standard is a most popular and secure symmetric algorithm. Most of the companies and government in the U.S. uses this algorithm to safeguard their sensitive data. An AES is a block cipher, the block size is either 128,192, or 256 bits. As this a symmetric algorithm, key must be exchanged before data communication happen. So, data and key must be secured to prevent unauthorized access.
* RSA: Rivest-Shamir-Adleman (RSA) is an asymmetric key algorithm that uses a key pair for encryption and decryption. A public key is used for encryption and private key or secret key is used for decrypting a ciphertext. The key size for this algorithm is 1024 and 2048, this gives an high level security for the data but it takes huge computational latencies when compared to other methods. This algorithm mostly used for sharing an information across an insecure network and in database encryption as well.
* Twofish: A twofish is a symmetric algorithm and flexible license-free method. This algorithm takes 128 bits or 256 bits and 16 rounds for encryption. This can be customized and ran quicker than normal process.

**TDE workflow**

To enable TDE in AWS for relational databases for SQL server DB instance, need to check with TDE option in RDS group which is associated with DB instance.

1. Check whether DB instance is enabled with TDE in RDS console.
2. If DB instance not enabled with TDE, we have two options to enable it. Creating a new option group with TDE or modify the existing option group to add TDE to it.
3. Make sure TDE is enabled with the option group to use it for encryption.

**Encrypting data using TDE**

Once TDE is added to option group, amazon RDS issues a certificate to perform the encryption on the given DB instance. This certificate used to execute the SQL commands that automatically encrypt and decrypt the data on DB instance. The performance of TDE completely depends on size of DB instance, amount of data, IOPS and other factors.

* Folder/network share encryption- under process.

***File level encryption***

File level encryption allows us to prevent unauthorized access or stealing while data is at rest or in transit. The file encryption is more sophisticated, more difficult to crack and more secure for file sharing and transfer with secure mechanism.

### ***File Encryption Workflow***

In this encryption, files are encrypted using advanced algorithms that can be decrypted with secret key which is created by file owner. The performance of this encryption is measured by complexity and strength of the algorithm, choosing correct encryption standard and key length.

Until the data being decrypted with appropriate key, encrypted data is confidential and secure from data at rest and in transit. Few encryption standards provides additional file protection and confidentiality by implementing complex mechanisms.

### **File Encryption Methods**

Number of encryption mechanisms are available for sharing encrypted file across the insecure network. Few are very commonly used in organizations, industries, and other areas. Most popular encryption standards are Open PGP, ZIP with AES, SFTP, FTPS, HTTPS, etc. Majority of encryption algorithms filters an information through a series of operations-permutations, additions, substitutions, etc., to secure the content which is available inside the file.

### **What Does File Encryption Do?**

The following is a process of encryption and decryption to attain the privacy and integrity for the organization.

1. Using PGP public key to encrypt and transfer the payload to an Ascend landing zone (S3 bucket)
2. Decrypt the payload with the PGP private key
3. Data - Transfer and Storage Store the decrypted payload in an S3 bucket encrypted with Experian keys.

An open PGP encryption standard used for encryption, decryption and authenticating the content which is being available inside the file. Open PGP uses symmetric and asymmetric cryptography, hashing, data compression to safeguard the data. Usually, PGP encryption done using the software application, it changes the plaintext to unreadable or complex code of characters.

***Field or column level encryption***

Field level encryption is a process of encrypting a specific field and upload it to web server for later use. This enable us to maintain confidentiality and integrity for sensitive fields or columns. These sensitive fields are encrypted at the user side and placed in cloud or web browser and this data remains cipher for the entire application process. This encryption allows only applications have the decryption keys to recover the data from encrypted.

In order to use this field level encryption, you have to configure CloudFront and set some fields in POST requests for encryption. At a time, you can encrypt ten fields or columns with a single request. You can’t encrypt all the fields at a time, you must configure according to need.

***Field-Level Encryption Flow***

The following steps provides an outline for configuring field level encryption.

1. Generate public-private key pair for configuring field level encryption in CloudFront.
2. Field-level encryption profiles need to be created, defines the fields that you want to encrypt.
3. Configure the field-level encryption for processing the encryption profiles. These configurations specify which profile to use based on the request obtained from the query argument.
4. Link the cache behaviour to specify when to encrypt the data by CloudFront.