Data flows and methods of Salary Finance's Employee Validation Application (EVA)

Introduction

This document has been written to give employer Information Security and IT teams a full understanding of how Salary Finance's Employee Validation Application ('EVA') works.

The document explains why we validate employee data, what role EVA serves in that validation process and gives a detailed explanation of how EVA works, including a description of the data flows between EVA and the Salary Finance platform.

Why does Salary Finance validate employee data?

Salary Finance validates employee data for three main reasons:

1. To **prevent fraud** by confirming that an applicant is indeed an employee of your organisation
2. To **determine an employee's eligibility** to apply for a Salary Finance product
    - For example, Salary Finance loans are only available to employees who have served with your organisation for a certain period of time
3. To help us to **provide our products responsibly**
    - As a provider of financial products, accurate income data is vitally important. For example, we use salary data to help us determine the affordability of a loan

What is EVA and what does it do?

EVA is an application which allows employers to validate data provided by employees[1] as part of

[1] EVA is a Portal Executable so it does not require installation and can be run from a folder on a Payroll

their application for a Salary Finance product. EVA does not require installation and can be run on a Payroll Administrator's machine or a shared desktop (supports Windows, Mac & Linux). In summary, it connects to the Salary Finance network via HTTPS in order to:

- Retrieve information on how to encrypt and hash the employee data
- Retrieve mapping information to help EVA interpret the payroll file correctly
- Identify joiners and leavers
    - To do this we compare the hashed encryption keys vs. the previous data upload to identify new and deleted keys
- Post encrypted and hashed records to the SF database

EVA access

To use EVA, the payroll user will need to be granted EVA access by your project lead or payroll manager during implementation. We will capture this user information during the implementation project. Accounts are configured using the following rules.

- Usernames must be a unique named work email address
    - We do not accept a group email address
- Passwords must be a minimum of 8 characters and contain 3 of the following...
    - Lower case letter
    - Upper case letter
    - Number
    - Special characters
- Permissions can be set at the payroll process and payroll level

Each user will be required to set their password when they first logon. The password setting/reset journey requires that the user enter their unique telephone number to further verify the user who is requesting a new password.

EVA Encryption and Hashing

EVA was designed with two core principles in mind:

1. to ensure that no employee data ever leaves your network perimeter in readable/plain-text form
2. to prevent Salary Finance from being able to read an employee's data until an employee correctly enters a matching subset of their employment data

1 - Local network encryption and hashing

EVA runs all processes on your local network. This ensures that no employee data ever leaves your network perimeter in readable/plain-text form.

2 - Encryption and hashing method

Administrator's machine or a shared desktop (supports Windows, Mac & Linux).

EVA enables employers to securely encrypt (AES-256) each employee record using a subset of the employee's underlying data plus the employer unique identifier as the encryption key. EVA then hashes each encryption key using SHA-256, rendering all the data unreadable to Salary Finance.

When an employee applies for a Salary Finance product, we use a subset of their employment details to recreate the hashed encryption key. The platform then searches for a matching hashed key in our database. If we can find the hash, then we know that the employee provided encryption key matches and can successfully decrypt the rest of the employee's data. During the registration process, all employees agree that they are happy to share their employment data with SF.

The methods employed by EVA have been described by a top-ranked Data Protection barrister as 'the gold standard' for data sharing.

The encryption and hashing process is presented in detail in the rest of the document.

A Encryption process

1 Create encryption key

The employee encryption key is created by using:

i) a subset of the employee's employment data
ii) an employer unique ID

Each and every row represents an employee and unique encryption key is generated for each row.

2 Encrypt employee data

EVA then encrypts the employee's data points with the encryption key that has been generated in step 1 using the industry standard AES-256 cryptographic algorithm.

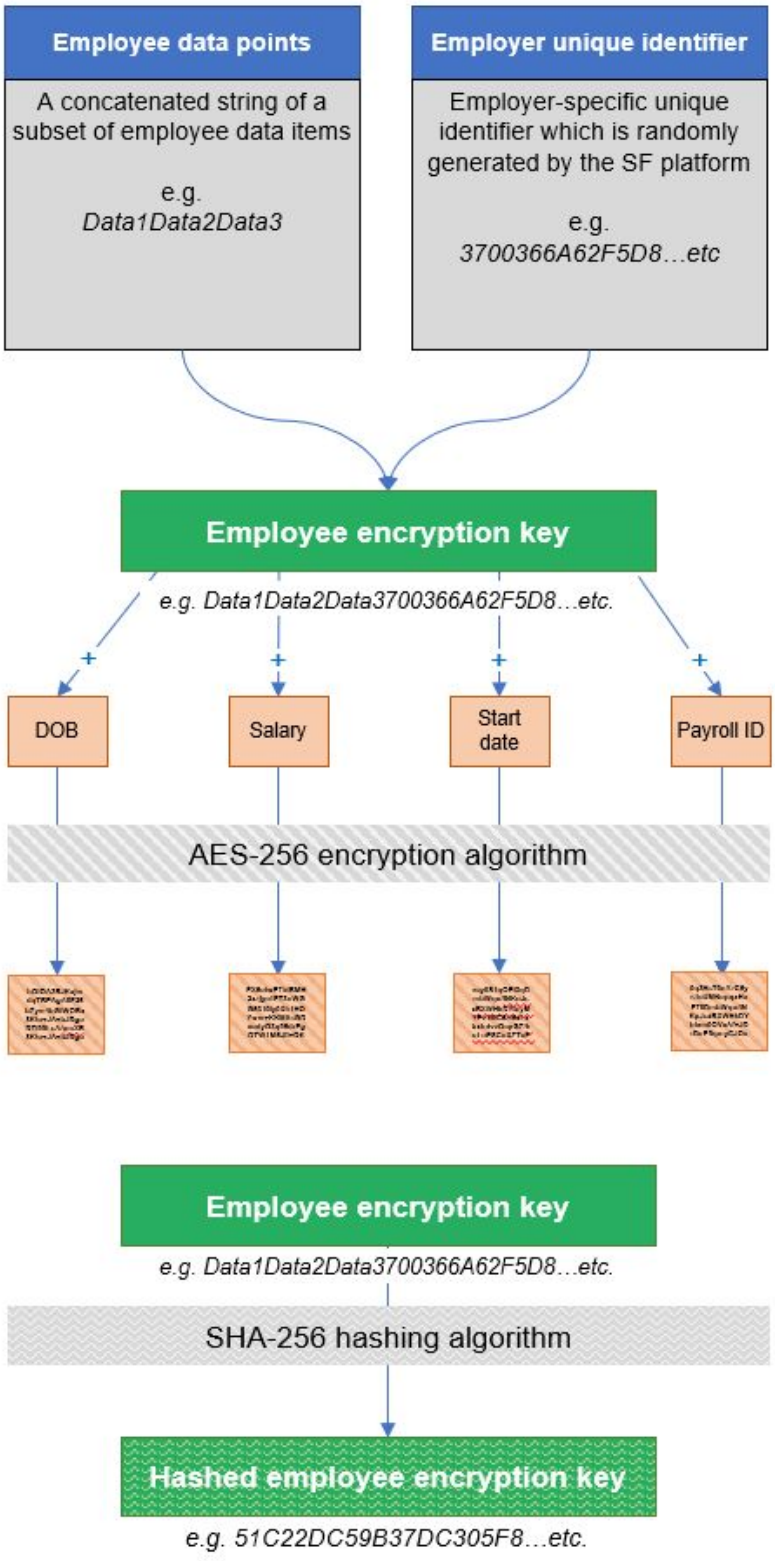
3 Hash encryption key

The encryption key created in step 1 is then hashed to create a long unique string which is used as part of our verification process.

The hash is based on the industry standard SHA-256 which creates an irreversible identifier for each row.

The hash is used during the application process – where the applicant enters the same original inputs to recreate the hashed value for verification.

Data points used to create the encryption key:



B Verifying employee

1 Recreates employee encryption key

During the application process, an applicant will provide the data points as used in part A.

Our platform uses the applicants data points to generate an encryption key using the same process as described in part A.

2 Hash recreated employee encryption key

The encryption key is then hashed using SHA-256. This new hash is used to verify the data provided by the applicant.

3 Search for matching hash

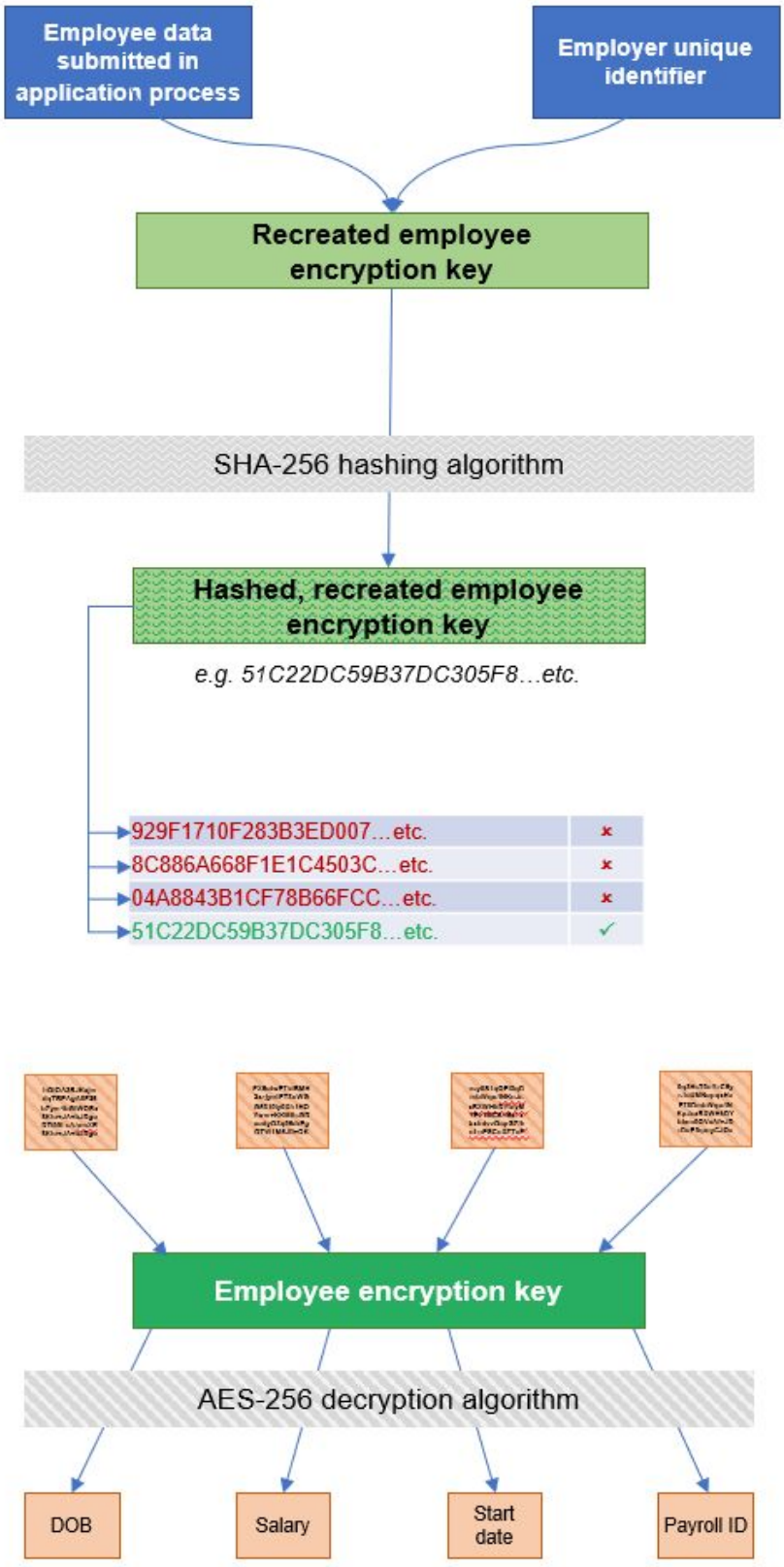
The platform checks that the hash provided by the applicant matches an existing hash provided by the employer.

Only if a matching hash is found, is the platform able to verify that the data points entered by the applicant match those provided by the employer.

4 Decrypt applicant's data

Having been matched, the employee encryption key can now be used to decrypt the employee data as it has been provided by the employee during the application process.

Data points required to recreate encryption key:



How does EVA work?

EVA validates the data of only those employees who have set up a Salary Finance account, applied for a product and have therefore signed up to our Terms & Conditions. By agreeing to our T&Cs, employees consent to their employer sharing data with Salary Finance.

Below we set out the high-level process for completing the validation process:

1. The Payroll Administrator opens EVA and signs in with username and password
2. The Payroll Administrator loads a Payroll File into EVA
3. The Payroll Administrator encrypts and hashes the Payroll File in EVA
4. The Payroll Administrator uploads the encrypted values and hashes to the Salary Finance platform

During the application process, the employee will provide the following information as part of their Loan application:

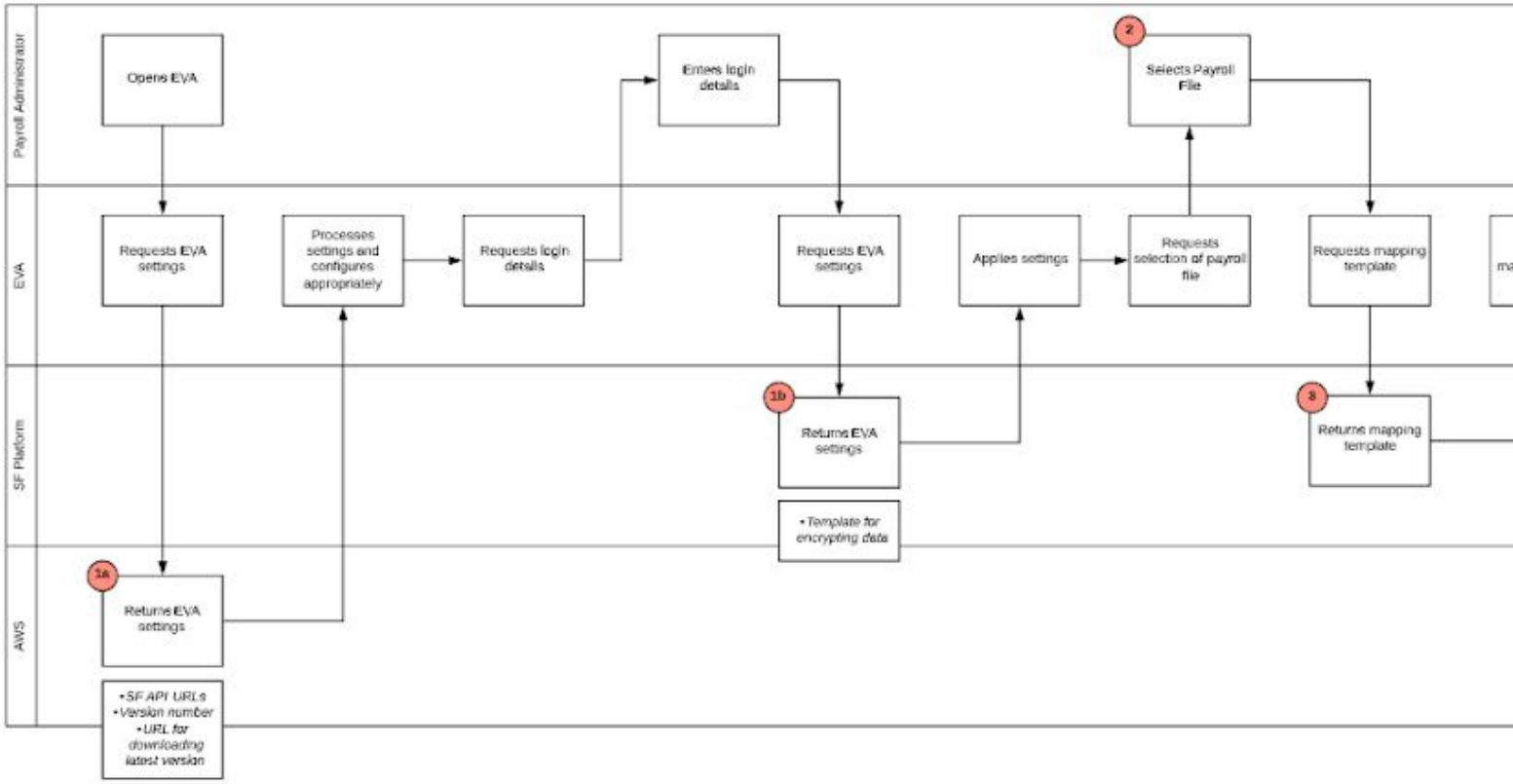
- Payroll ID
- Date of Birth
- Salary
- Start Date

The data listed above, along with the employer key, is encrypted and hashed to create an applicant hash. The Salary Finance platform searches for an employee hash that was previously uploaded via EVA which exactly matches the applicant's hash. If EVA can find a match, we can confirm that the applicant does indeed work for the employer. If EVA can't find a match, then the applicant fails validation.

Process flow

You can access a process flow, describing how EVA works and outlining the data flows by clicking on the image below. The red numbers on the process flow relate to the table of data handled by EVA which can be found on the page below

EVA Process Flow Diagram - Proactive



Data handled by EVA

Fig 1 - Illustration of EVA data flows

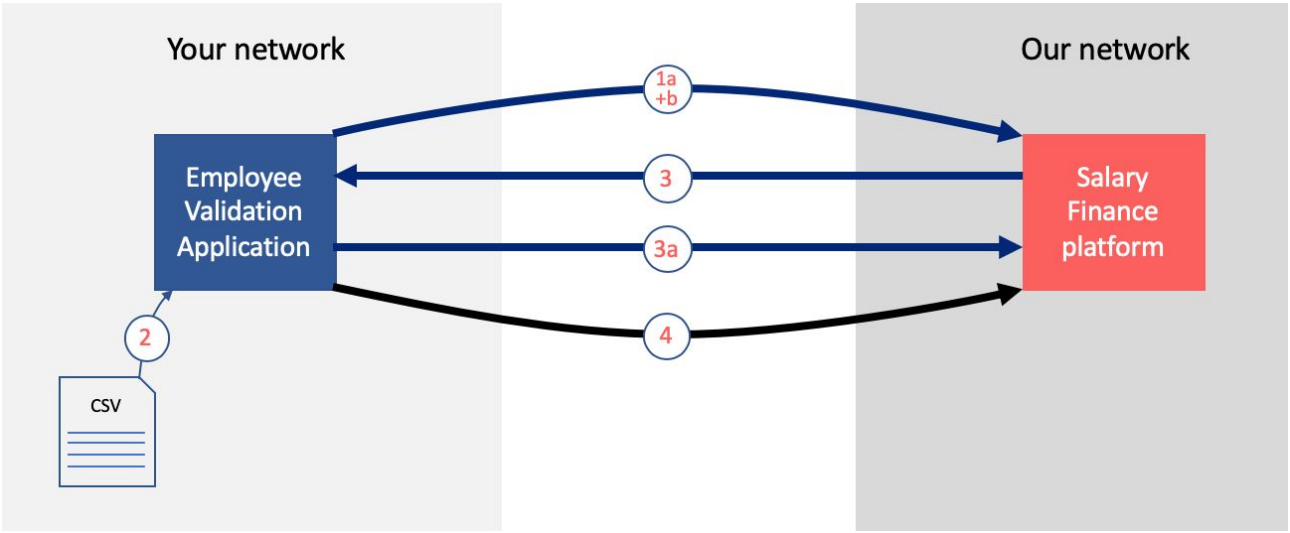


Table 1 - Description of EVA data flows, depicted in Fig 1

Data flow	Description	Data shared	Reason for data sharing
1a	Settings for EVA	SF API URLs	So that EVA can exchange data with the relevant SF services
		Latest version number	To compare vs. current version number - so that we can make sure you have the latest version
		URL for downloading latest version	To show the user where they can download the latest version
1b	Settings for EVA	Template for generating hash	So that EVA can encrypt and hash the data appropriately
2	Payroll File	Employee records	This is the payroll file to be used for validating applications
3	Mapping template	JSON object which lists data items required by SF and the associated payroll file column headers	To enable EVA to understand the payroll file - and to save Payroll Admin from having to map the file every time they use EVA

3a	Mapping template*	JSON object which maps data items required by SF to the associated payroll file column headers	*only shared if file has not previously been mapped or has been re-mapped
4	Encrypted and hashed records	The encrypted and hashed records generated from the payroll file data	So that we can validate employees as they apply (see “Why does Salary Finance validate employee data?”)

Validating bank details with EVA

Overview

So that we can provide our Advance product to your team members, we need to ensure that the bank account details shared by a team member belong to the account their salary is paid into.

We are aware that banking data is particularly sensitive, and we therefore process it in a slightly different way compared to the other data items in the Payroll File. The process has been designed in such a way that it is impossible to work back from the datapoint that you share with Salary Finance to the unique bank account and sort code of the team member.

How the hash enables safe data validation

The crucial element of sharing bank account details via EVA is the creation of a four-character hash. EVA takes the account number and sort code, concatenates them, passes them through SHA-256 and selects the last four characters of the hash to produce the four-character hash.

We create a four-character hash because even if an individual had access to the four-character hash and the hashing algorithm, it would be impossible to work back to a *unique* bank account and sort code combination. This is because, simplistically speaking[2], there are 100,000,000,000 (100 trillion) possible account number and sort code combinations (14 digits) and only 65,000 possible hashes (4 hexadecimal characters = 16^4 = 65k). This means that for each possible four-character hash there are c.2 billion possible account number and sort code combinations - making it impossible to work back from a four-character hash to a *unique* account number and sort code.

Once created by EVA, the hash is passed to our platform with the rest of the data (step 4, above) and stored on our platform. During the application process, when the team member enters their

[2] The way that account numbers and sort codes are created (using a modulus check) and the dominance of four major high-street banks makes the calculation more nuanced than we present above. We estimate that it’s closer to 100k potential account number and sort code combinations that will produce a given four-character hash. Nonetheless, it’s still impossible to work back to a unique account number and sort code combination from the hash.

bank account and sort code, we pass the bank details through SHA-256 and compare the last four characters of the hash against the hash stored in our database. If the four characters match, then we know we have the team member's salaried account[3] and are able to validate the account.

[3] There is a 1 in 65k chance that the team member will mis-type their account details and yet still pass the check

C Bank-details hash

1 Create four-character hash

Due to the sensitive nature of the data points bank details do not follow the same process as previously mentioned in A and B.

EVA creates a four-character hash by:

i) concatenating the account number and sort code
ii) processing the resulting value through the SHA-256 hashing algorithm

Only the last four digits of the hash are then sent as part of the data file.

2 Store hash

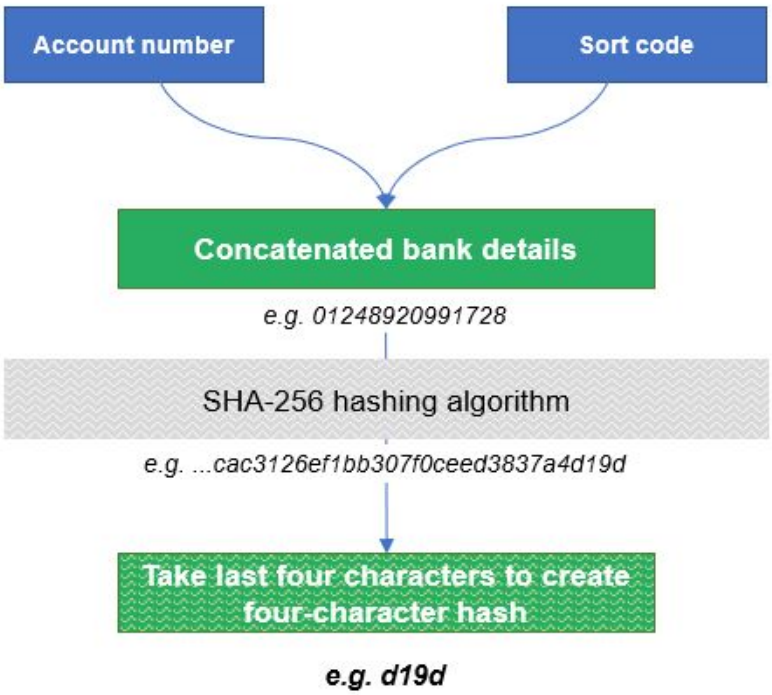
EVA then transmits the four digit hash to the SF platform via HTTPS/TLS API along with the encrypted and hashed dataset mentioned in A and B.

3 Recreate & compare four-character hash

An applicant enters their bank account number and sort code during the application. We process it through the same concatenation and hashing algorithms to produce a four-character hash.

This hash is compared against the four-character hash in step 1. If the hashes match then we are able to validate that the bank account details provided during the application are the applicant's salaried bank account.

Data points to generate the hash:



Recreate the four-character hash: