

Card encryption - verifone.js

Overview

The verifone.js library is a quick and secure way to collect sensitive credit card data. This allows users full control over the checkout experience while maintaining a minimum SAQ A-EP level. The user receives an encrypted card block, which can be used to process a payment.

Availability

verifone.js is available for all Verifone merchants who use the Payments API of Verifone or process their payments using an external party.

Benefits

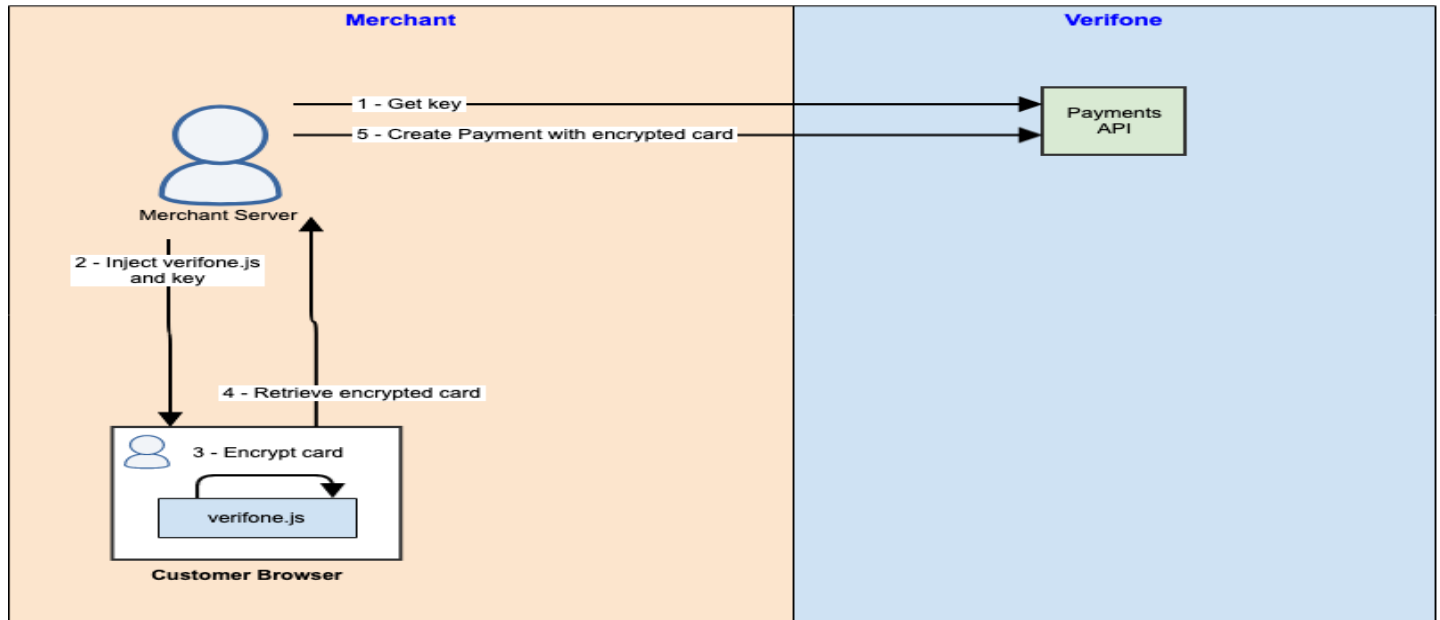
The verifone.js solution offers:

- Full ownership over merchant payment flow
- Reduced complexity for PCI DSS compliance
- Seamless integration into an existing checkout page, regardless of the stack used

Supported browsers

- Chrome
- Firefox
- Edge
- Safari
- Opera
- Brave

How it works



Integration steps

An account is created in Verifone Central by a Verifone employee.

The onboarding is done with a [Secure Card Capture Key on Verifone Central](#).

To integrate verifone.js in your Checkout page, follow these steps:

1. Load verifone.js in the <head> of your Checkout page, where the card form is shown.

```
<script src="https://cst.jsclient.vficloud.net/verifone.js"></script>
```

Sandbox environment:

- EMEA : <https://cst.jsclient.vficloud.net/verifone.js>
- US : <https://uscst.jsclient.vficloud.net/verifone.js>

Production environment:

- EMEA : <https://emea.jsclient.verifone.cloud/verifone.js>
- US : <https://us.jsclient.verifone.cloud/verifone.js>
- NZ : <https://nz.jsclient.verifone.cloud/verifone.js>

2. Define the [Secure Card Capture Key](#).

```
var encryptionKey = "SecureCardCaptureKey"
```

3. Collect the card data and place it into a JSON object.

Card schema

Parameter name	Type	Required/Optional	Description
cardNumber	String	Required*	Numeric value with no spaces or separators allowed between the digits
expiryMonth	String	Optional	Numeric value with length 2. e.g., March -> 03

Parameter name	Type	Required/Optional	Description
expiryYear	String	Optional	Numeric value with length 2, e.g., 2028 -> 28
cvv	String	Optional	Numeric value with length 3 or 4
svcAccessCode	String	Optional	Numeric value with length up to 8 PIN number of non-PCI cards (e.g.: Gift Cards)

***cardNumber** is required for getEncryptedCardDetails method and optional for encryptCard

```
var card = {
  "cardNumber": "4000000000001091",
  "expiryMonth": "03",
  "expiryYear": "28",
  "cvv": "123"
}
```

4. Invoke verifone.js to encrypt the card using `encryptionKey`

```
var encryptedCard = await verifone.encryptCard(card, encryptionkey)
```

verifone.encryptCard returns a Promise. Not all browsers support using async-await. Consider this in your integration.

Alternatively, if you would like to receive additional data regarding the card, such as expiration date and limited card digits use the function below:

```
var encryptedCard = await verifone.getEncryptedCardDetails(card, encryptionkey)
```

5. If you are using `getEncryptedCardDetails` you can use the BIN in a binlookup call to receive the card_brand for grater accuracy in initiating payments.

```
POST /v2/card/card-details
{
  "prefix": "500000"
}
```

If this returns multiple results, you should ask the shopper to select one of the brands, corresponding to their card.

```
{
  "bin_details": [
    {
      "bin": "52",
      "card_brand": "Maestro",
      "issuer_country": null,
      "issuer_name": "Mastercard",
      "type": "DEBIT",
      "funding_source": "DEBIT"
    },
    {
      "bin": "52",
      "card_brand": "MasterCard",
      "issuer_country": null,
      "issuer_name": "Mastercard",
      "type": "CREDIT",
      "funding_source": "CREDIT"
    }
  ]
}
```

6. [Optional] To initiate a payment using the Payments API, use the `encryptedCard` parameter alongside the public key alias of the encryption key:

```
POST /v2/transactions/card
{
  "amount": 3252,
  "encrypted_card": "encryptedCard", //encrypted card value from verifone.js
  "public_key_alias": "K9",
  "card_brand": "MasterCard" //optional
  ...
}
```