August 2020

# Browser-native Static Virtual Card For Online Transaction Security

Aneesh Ali Nainamvalappil Cheriyakath

Ling Lin

# Browser-native Static Virtual Card For Online Transaction Security

ABSTRACT

Online transactions using credit cards are susceptible to fraud. This disclosure describes techniques to provide a browser-native implementation to dynamically bind a static virtual card (SVC) with a merchant server for an authorized period. When a user provides SVC credentials to an e-commerce merchant for completing a transaction, the SVC is authenticated at the merchant server by obtaining an authorization ID from the card issuer. The authorization ID has an associated expiration date. The browser native implementation allows accurate determination of the merchant domain and establishing binding between the SVC, the merchant ID, and the merchant domain for a specific period after which the binding is automatically removed. In this manner, a user has only one SVC associated with a payment card at any point in time that can be used with multiple merchants. The techniques improve credit card security while providing a simple user experience.

KEYWORDS

- Static virtual card
- Payment card
- Credit card
- Card fraud
- Online transaction
- Browser native
- Dynamic binding

BACKGROUND

Data breaches are increasingly common. While payment card networks have reduced in-store fraud with chip and pin technology or tap and pay, there is no efficient solution for e-commerce, without the merchant doing some integration work with the card processors. When a user shops online, the user needs to give out credit card credentials. Such credentials can be accessed and abused by fraudsters who hack into the merchant's servers and steal payment card numbers. It is important that users be provided security but in a non-disruptive and easy to use manner.

Current e-commerce transactions can be classified into two categories:

1. **Transactions using tokens:** A token is used for an e-commerce transaction instead of an actual credit card. This token has static as well as dynamic elements that make the transaction secure. For example, merchants can integrate with online payment or wallet systems to reduce credit card fraud by using tokenized cards.
2. **Transactions using actual card details:** The actual credit card details, including the card number, expiry, and CVN, are used for the transaction.

To deal with frauds associated with the second category above, some financial institutions provide their customers with a Virtual Card Number (VCN) that uses new technologies, e.g., a browser extension, for security. However, a VCN is associated with a single merchant, thereby requiring users to manage VCNs associated with different merchants. Some financial institutions use cloud tokens, where a dynamic Card Verification Number (CVN) is generated each time the user attempts a transaction. However, merchants need additional technological capabilities to handle tokens correctly each time and may end up declining purchases.

DESCRIPTION

This disclosure describes mechanisms of generating a single Static Virtual Card (SVC) corresponding to a payment card and providing dynamic binding of the card to multiple merchants. A virtual card is a randomly-generated credit card credential delivered for payment on behalf of a user's static credit card. The SVC is provided via browser native integration.

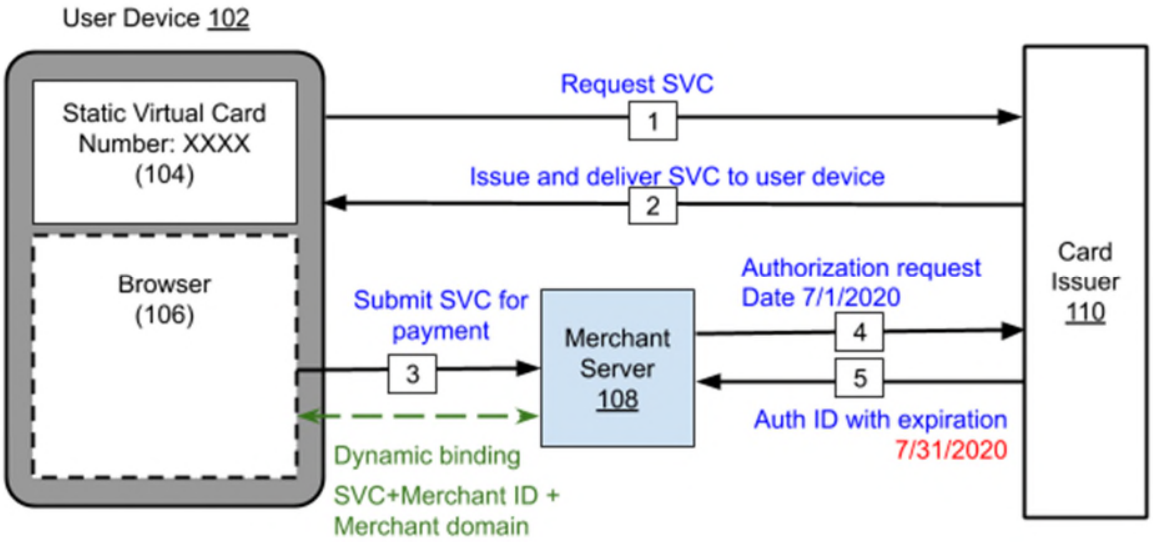


**Fig. 1: Example process for dynamic binding of a static virtual card with a merchant**

Fig. 1 illustrates an example of the dynamic binding of an SVC with a merchant. A user that wishes to perform an online payment via a user device (102) requests [1] a static virtual card (SVC) number (104) from a card issuer (110). The SVC number, linked to the user, is delivered [2] to the user device. The user then visits a merchant's website using the device browser (106). To purchase a product, the user submits [3] the SVC credentials to the merchant server (108).

The merchant server requests authorization [4] from the card issuer. The card issuer validates the SVC and returns an authorization ID [5] to the merchant server. The authorization request has a request date (“7/1/2020” in the example of Fig. 1). The authorization ID has an expiry date associated with it (“7/31/2020” in the example of Fig. 1). In this manner, a merchant

binding is established between the SVC, merchant ID, and merchant domain. The expiry date of the merchant binding can be different from the expiry of the SVC itself and is determined based on the expiry associated with the authorization code when the merchant first initiates a transaction authorization request.

The authorization process aligns with and mimics the behavior of a standard card pre-authorization request, where the merchant gets a certain number of days to initiate a capture request. For example, for a debit card, the expiry date of merchant binding is seven days. Similarly, for a credit card, the expiry date of merchant binding maybe thirty days. However, the merchant binding's expiry date may also depend on the card issuer and on merchant types.

The user can use the same SVC credentials for e-commerce transactions with other merchants as well. A second merchant can go through the same steps as the first merchant, as described above. The second merchant server establishes a dynamic binding with the SVC with expiration determined by the expiry date associated with the authorization provided by the card issuer to the second merchant. Thus, the user does not require new SVC credentials for transactions with different merchants.

The merchant binding is removed as soon as the expiry date arrives. When the merchant binding is about to expire, the user can be notified about the expiry. The expiry notification gives the user a chance to extend the expiry of binding in specific cases where the merchant needs more time to make additional transactions related to the purchase. To address split shipment cases, the techniques of this disclosure provide additional functionality to extend the expiry of the binding every time the merchant for the same SVC initiates a new pre-authorization request. The expiry in such cases can be extended to the new expiry date associated with the most recent authorization request.

The described techniques provide several advantages in preventing credit card fraud. Browser native implementation helps in determining the merchant domain accurately, even if the user utilizes a payment gateway in a different domain. Correctly identifying the merchant domain helps in having a precise binding of the SVC, merchant ID, and the merchant domain. Since there is only one SVC associated with a card at any point in time, the techniques discussed herein make it easier for users to manage SVCs.

Dynamic expiry of merchant binding makes things stricter for the merchants where they can use an SVC only for the allotted period. Automatic expiry reduces the risk associated with the compromise of payment card data from the merchant's database. A fraudster can only make successful transactions using SVCs that haven't expired. If a fraudster attempts a purchase on an expired SVC, the issuer can automatically decline the request. In case an SVC credential and merchant binding is compromised, a new SVC can be issued, invalidating all prior merchant bindings.

## CONCLUSION

Online transactions using credit cards are susceptible to fraud. This disclosure describes techniques to provide a browser-native implementation to dynamically bind a static virtual card (SVC) with a merchant server for an authorized period. When a user provides SVC credentials to an e-commerce merchant for completing a transaction, the SVC is authenticated at the merchant server by obtaining an authorization ID from the card issuer. The authorization ID has an associated expiration date. The browser native implementation allows accurate determination of the merchant domain and establishing binding between the SVC, the merchant ID, and the merchant domain for a specific period after which the binding is automatically removed. In this manner, a user has only one SVC associated with a payment card at any point in time that can be

used with multiple merchants. The techniques improve credit card security while providing a simple user experience.

REFERENCES

1. https://www.capitalone.com/applications/eno/