AUTOMATIC FORM FILLING WITH SECURE PAYMENT CREDENTIALS AND BIOMETRIC AUTHENTICATION

Introduction

Data breaches are increasingly common within financial systems and potentially impact billions of people globally. It can be costly for card issuers to handle fraud and inconvenient for users to handle the consequences of such fraud. While card networks have reduced in-store fraud with technologies such as chip and pin technology or tap and pay, networks and banks are still working to reduce fraud in e-commerce.

Tokenization is a feature that enables users to avoid sharing actual payment credentials (e.g., credit card number, debit card number) with third parties while making payment transactions. Normally, when a user makes a transaction, their entire card information including the Funding Primary Account Number (FPAN), expiry month and year, and Card Verification Value (CVV) needs to be shared. Using a FPAN, expiry and CVV, however, can expose security risks that enable fraudsters to access this information and use them for fraudulent transactions.

Device tokenization is a technique that replaces the FPAN, expiry, CVV combo with a Device Primary Account Number (DPAN), expiry and a cryptogram/CVV. A DPAN is a unique, per user, per device number assigned by a Token Service Provider (TSP, normally maintained by a Payment Network) and is mapped uniquely to the FPAN. The cryptogram is a field that gets computed based on a unique key and can be verified by the Payment network before connecting with the issuer for authorizing the transactions.

Device tokenization has made in-store transactions more secure since fraudsters cannot deploy replay attacks for DPAN-based transactions. However, fraud remains in e-commerce transactions where transactions are treated as card-not-present transactions and the authenticity
of the user is determined by verifying if they know the CVV of the card (along with the FPAN and expiry).

Summary

A payments platform is described with cloud tokenization and biometric authentication capabilities that are integrated with web browser autofill technology to protect payment instruments such as credit cards from being compromised. Cloud tokenization with autofill enables passing a tokenized card number, expiration date, and one-time use and dynamic CVV (dCVV) which expires after the first usage or predetermined time period, whichever comes first. These fields can be accepted by merchant websites with no merchant API integration. With cloud tokenization, a credit card Funding Primary Account Number is not shared online. If a hacker obtains a tokenized card number, a dynamic card verification value expires in a predetermined time period (e.g., 24 hours) after generation, rendering the tokenized card number and expiration date useless for malicious actors. Cloud tokens can be created for a user/FPAN combination and a dCVV can be used to protect the card from replay attacks while providing the same integration with payment gateways.

While logged into a browser, a cloud token can be generated for use in online shopping and other payment transactions. Users with eligible payment accounts and eligible credit or debit cards can have a cloud token available to them for filling in forms. In one click, a user can fill in the secure payment credential after passing a CVC or biometric authentication process, thereby enabling a secure payment credential for online shopping with no API integration. A browser can use this technology for auto-filling payment checkout forms. Instead of filling in forms with a
payment card number, forms can be filled with a cloud token which resembles a credit card number, expiration date, and CVV.

In some examples, a secure biometric authentication process can be provided for users prior to auto-filling payment credentials into a form, such as a merchant form on a checkout page. Payment credentials can be securely bound to a user’s account with a payment management computing system, and can additionally be securely bound to a user computing device by a successful check with an issuer computing system associated with the payment credentials.

**Detailed Description**

The present disclosure is directed to systems and methods for enabling a safe, easy, and reliable way to provide payments online. More particularly, a system is provided whereby payment credentials are tokenized to create a tokenized card number so that a Funding Primary Account Number (FPAN) is not shared online. A token can appear the same as or similar to a regular payment credential number such as a credit card number (e.g., FPAN), can behave like a FPAN, and may not functionally alter a traditional payment transaction processing flow. A token can include a 16-19 or other digit number (e.g., format preserving) and can appear similar to a PAN. A cloud token can include a tokenized card number such as a cloud primary account number (e.g., CPAN). A tokenized payment credential can utilize a dynamic card verification value (dCVV). A token can support the same operations as FPANS, including but not limited to, authorization, capture, refund, and chargeback. A transaction limit associated with a payment credential such as a credit card can be the same for a tokenized number. Payment processors and merchants can process tokens for transactions without having to know that an account number is
a tokenized card number. In some instances, such as in the case of subscriptions, merchants may setup a merchant initiated transaction framework (MIT) in order to continue to charge the tokenized payment credential for recurring subscriptions. In some examples, the payment platform enables adaptation of API’s in order to solve for autofill-specific use cases, including processing transactions without CVV and delayed charges with no authentication at time of card entry.

In example embodiments, a user can have a payment credential such as a credit card including a FPAN stored at a payment management system. The user may be signed into an account with the payment management system (e.g., through a web browser). The user can arrive at a merchant checkout page that requires payment credentials such as a credit card number to be entered. The user can click on a form field to enter credit card information. In response, the system can provide the user with the option of using a virtual credit card. If the user chooses to use a virtual card, the user’s card token can be populated into the form with the expiration date and dynamic CVV. The user will receive confirmation from the merchant that the purchase was successful and the user should expect an emailed receipt that shows the CPAN’s last 4 digits.

In some instances, a user may switch between use of a FPAN and a CPAN. In example embodiments, the system can initiate user authentication when the user attempts to unmask or otherwise view the FPAN. When the user gets a CPAN, the system can initiate user authentication and a new dCVV can be generated.

A cloud token, sometimes referred to an ecommerce token, can include a token representation of card or other payment credential on file. The cloud token can be generated by network token service providers (TSPs). Cloud tokens can be used in remote transactions and be unique to each token requestor. For example, a first payment management computing system
operating as a token requestor will receive a different cloud token for the same FPAN than any other token requestor would receive for the same FPAN. Additionally, cloud tokens can be different if the same FPAN is added to different user accounts managed by the payment management computing system. Cloud tokens can be stored by the payment management computing system and can be accompanied by a dynamic element for each transaction (either a cryptogram or dCVV).

A cryptogram is a unique single use element generated by token service providers to authenticate tokens for each transaction. For device tokens, a cryptogram is typically required, but for cloud tokens, a single use dynamic CVV can be used instead of the cryptogram to authenticate the token. A dynamic CVV (dCVV) can be a single use CVV value generated for each transaction. A token service provider can use a dCVV in conjunction with an expiration date to verify a cloud token - instead of requiring a cryptogram. Merchant initiated transactions can include transactions that merchants are allowed to initiate using a cloud token without generating a new single use dynamic element (e.g., recurring subscription charges, hotel incidentals, additional ride sharing costs, etc.). Merchants can use cloud tokens without dynamic elements for transactions that fall within the scope of the network defined merchant-initiated transactions (MITs). For these transactions, merchants may share a reference ID for the original transaction during authorization.

FIG. 1 is a block diagram depicting a simplified view of an example of cloud tokenization, illustrating an example interaction between computing systems associated with various parties involved in a cloud tokenization process. FIG. 1 depicts an initial token generation process which can occur asynchronously and out of session in some instances. While an asynchronous flow is illustrated in FIG. 1, payment credentials such as credit cards can be
tokenized as soon as they are added to the payment management system(s) 104 in some examples. A payment management system (PMS) may include one or more computing devices such as web servers, application servers, etc. A user can access a user computing system 102 and provide payment credentials such as a FPAN and expiration date which are saved to the payment management system. For eligible payment credentials, the payment management system 104 can issue one or more calls to payment network system(s) 106 as shown at (2). A payment network system (PNS) may include one or more computing devices such as web servers, application servers, etc. The payment management system can request that the payment network system create a cloud token for the payment credential. The payment network system may be operated by a card network in example embodiments and may include or otherwise communicate with a token service provider (TSP). The TSP can be responsible for generating secure tokens for replacing FPANs of payment credentials. A TSP can also be responsible for lifecycle management of the tokens they generate. For example, a TSP may be able to suspend tokens or delete tokens.

As shown at (3) in FIG. 1, the payment network system 106 checks for tokenization eligibility of the payment credential with an issuer computing system(s) 108. The issuer computing system 108 responds at (4) with an indication (e.g., Yes/No) that the payment credential is eligible for tokenization. If the issuer computing system 108 indicates eligibility for tokenization of the payment credentials, the payment network system 106 generates a cloud token at (5). The payment network system then passes the cloud token and the expiration date to the payment management system. The payment management system can store the cloud token and expiration date for use in payment transactions for the payment credentials supplied by the user (e.g., the FPAN and expiration date).
In some examples, a payment management system can use a cloud token when available and use an FPAN for eligible merchants when there are no cloud tokens available for a given FPAN. Dynamic elements can be requested for every facilitated transaction in some instances. A dynamic element can be requested either in advance (e.g., with a small grace period for validity of dCVV such as 24 hours) or during transaction time. Tokens may be long lived in a token requestor’s servers, but dynamic elements that accompany the tokens can be single use.

FIG. 2 is a block diagram depicting a simplified view of a payment transaction, illustrating an example interaction between computing systems associated with parties utilizing cloud tokenization. FIG. 2 depicts an example of a payment transaction whereby a user is able to fill in forms automatically with cloud token credentials. A user can access user computing system 102 and interact with various merchants or other entities that host pages having form fields for providing payment information such as credit card credentials for completing a payment transaction. At (1), a user can access a checkout or other page associated with a merchant computing system that is able to receive payment credentials. The user downstreams an autofill component from the payment management system 104 to the user computing system 102. In response to the user request to downstream autofill of payment credentials, the user may be prompted by the user computing system 102 to provide authentication credentials at (2). For example, the user may pass authentication credentials to the payment management system 104, such as a CVC check with issuer (e.g., bank) or biometric authentication as hereinafter described. If the user passes successful verification, the payment management system 104 requests a dynamic CVV from the payment network system 106 at (3). In response to the
request, the payment network system 106 generates a one-time use dynamic CVV at (4). The dCVV expires after a predetermined amount of time (e.g., 24 hours) from generation, or after its first user, whichever comes first. The PNS passes the one-time use dCVV to the payment management system. The payment network system then passes the cloud token and the dCVV to the user computing system 102. The client browser 110 places the cloud token and the dCVV into the merchant’s form. In this manner, the merchant’s form is auto-filled or auto-populated with the cloud token and dCVV to enable a payment transaction without requiring the user to manually enter this information. The client browser 110 can then pass the cloud token and the dCVV to the payment network system. For example, the user can submit the cloud token and the dCVV by providing input at the browser resulting in transmission of one or more packets including the cloud token and dCVV to the merchant’s payment processor. The PNS can then send a request to the issuer system 106. The PNS can send one or more messages including the FPAN corresponding to the cloud token and a request that the issuer system 108 authorize the FPAN. The issuer system 108 can then respond (not shown) with a message indicating that the transaction is authorized or declined.

In accordance with some example embodiments, a secure biometric authentication process can be provided for users, prior to auto-filling payment credentials into a form, such as a merchant form on a checkout page. Payment credentials can be securely bound a user’s account with a payment management system and can be additionally be securely bound to a user computing device by a successful CVC check with an issuer computing system associated with the payment credentials. In some examples, a payment management system can automatically enroll a user’s security key.
Example biometric authentication processes described herein can provide less user friction than a traditional CVV check with an issuer system. Furthermore, these processes can be more secure than a numeric PIN code or a client-side biometric authentication which are more easily hacked or spoofed. In accordance with example embodiments, cryptographic checks can be performed server side (e.g., by a payment management computing system), which can provide higher security than purely client-side checks which can be easier to circumvent. In some examples, a device can be bound to a card or other payment credential using a successful CVC authorization with an issuer computing system on a per-card basis. A device can be bound to a card or other payment credential in order to access the card or payment credential.

In example embodiments, a web authentication application programming interface (API), sometimes referred to as WebAuthn, is leveraged. A WebAuthn specification can define an API enabling the creation and use of strong, attested, scoped, public key-based credentials by web applications, for the purpose of strongly authenticating users. WebAuthn can be based on a set of specifications provided by one or more parties. In example embodiments, a payment management computing system can provide a user’s sensitive payment information to a user via a web browser, and a WebAuthn API can be used as a trusted mediator with the payment management computing system as a relying party ID. WebAuthn can include registration and authentication. An example registration flow is shown in FIG. 3 where a user opts into WebAuthn-based authentication. Once this flow is successful, the user is registered with the relying party for future authentication. In the case of autofill using a payment management system, the relying party ID can be a domain of one or more computing devices associated with the payment management computing system. Registration can happen at various times in autofill
- on upstream, on downstream, via browser settings, via payment system settings, at browser start time, at device or operating system start time, etc.

Once a user is registered on a device with the payment management system domain WebAuthn and a payment credential card is already CVN verified on that device, a WebAuthn authentication flow can be provided. FIG. 4 depicts an authentication flow diagram illustrating interactions between a relying party, WebAuthn API, and an authenticator. The relying party ID can be the domain of a payment management computing system.

WebAuthn credential management can be created after a CVC check in example embodiments. A user can have the option to opt-in and opt-out from a settings page. The payment management system can reuse credentials, and credentials created by the system can be reused by other services.

FIG. 5 depicts a detailed flow of a registration process associated with autofill of payment information in accordance with example embodiments. FIG. 6 depicts a detailed flow of an authentication process associated with autofill of payment information in accordance with example embodiments. In example embodiments, an autofill client can be provided as an internal component of a web browser. For instance, the autofill client can hook directly into the WebAuthn internal implementation instead of calling into a Javascript API. Moreover, server calls can be made through a payment management system client, and an orchestration server can interact with Cryptauth servers of the payment management system on behalf of the client.

Browser client code can provide HTTPS requests to Orchestration servers of the payment management system, as opposed to making direct calls to CryptAuth servers in some examples. The Orchestration servers can make calls to the CryptAuth server, processing the response and responding to the browser client.
The browser client can be configured not to make separate calls to Orchestration and CryptAuth servers in some examples because the payment management computing system may want cryptographic proof that the authenticator signed the dynamic nonce correctly. Since the CryptAuth server is making this check, the Orchestration server may desire a direct response from the CryptAuth server — as opposed to trusting the browser client about the Cryptauth server’s verification response.

The WebAuthn API can compare the web page’s origin to the “Relying Party ID” (rpID) of a credential to ensure the page has permission to access the credential. Since autofill can happen on third-party merchant checkout forms, the web origin may not be related to autofill’s rpID. An internal WebAuthn implementation can be created that bypasses this origin check in some examples. This bypass can happen in the browser process and may not be accessible from the renderer (and therefore not accessible by any web-based attack). Furthermore, by setting the file visibility to be restricted to certain classes, it may be ensured that no other components in the browser can access this.
Figures

FIG. 1

FIG. 2
FIG. 3
FIG. 4
Abstract

A payments platform includes cloud tokenization and user authentication capabilities that are integrated with browser autofill technology to protect payment instruments such as credit cards from being compromised. A tokenized card number, expiration date, and a one-time use and dynamic value can be passed as part of a payment transaction. These fields can be accepted by merchant websites with no merchant API integration. With cloud tokenization, a credit card Funding Primary Account Number (FPAN) is not shared online. Cloud tokens can be created for a user/FPAN combination and a dynamic value can be used to protect the card from replay attacks while providing the same integration with payment gateways. In example embodiments, a secure biometric authentication process can be provided for users prior to auto-filling payment credentials into a form, such as a merchant form on a checkout page. Payment credentials can be securely bound to a user’s account with a payment management computing system and can additionally be securely bound to a user computing device by a successful check with an issuer computing system associated with the payment credentials.