Frictionless Web Payments with Cryptographic Cardholder Authentication[*]

Francisco Corella and Karen Lewison

Revised July 21, 2019

Abstract

The 3-D Secure protocol, introduced 20 years ago, aims at reducing online credit card fraud by authenticating the cardholder. Version 2 of the protocol, not yet deployed, addresses usability problems that have hindered the deployment of version 1 by introducing a frictionless flow for low risk transactions. But the frictionless flow does not authenticate the cardholder. Instead, it requires the merchant to send information to the issuer through a back channel, potentially violating the cardholder's privacy. The paper analyzes the usability, privacy and security provided by 3-D Secure 2 and proposes an alternative protocol, simpler and less expensive to implement, where the cardholder is authenticated with a cryptographic credential stored in the cardholder's browser with zero friction. The scheme can take advantage of a native bank app in the cardholder's device to further authenticate the cardholder by fingerprint scanning or face recognition as made available by the device, and can be used for credit card purchases made on the merchant's web site or on a merchant app.

Contents

[*]This is the authors' version of a paper presented at HCII 2019. The publisher's version can be found in the Late Breaking Papers volume of the proceedings.

List of Figures

1 Introduction

The addition of EMV chips to credit cards has reduced credit card fraud for in-store transactions, but fraudsters have shifted their efforts to online transactions, and the rate of online credit card fraud has increased.

Providing serious security for online credit card purchases requires authentication of the cardholder. This was recognized by the credit card industry and addressed, first by the specification of the Security Electronic Transactions (SET) protocol [?, 1], which was never deployed, then by the introduction in 1999 of version 1 of the *3-D Secure* protocol [2], created by Visa and marketed under different names by different credit card networks (Verified by Visa, MasterCard SecureCode, American Express SafeKey, etc.). In 3-D Secure 1.0 the merchant's web site redirects the cardholder's browser to the web site of the issuing bank, which authenticates the cardholder and redirects the browser back to the merchant's site.

But two decades after its introduction 3-D Secure 1.0 is still unevenly used in European countries and rarely used in the United States.

One reason for the limited deployment of 3-D Secure 1.0 is the friction that it causes. Traditionally, the bank authenticated the cardholder by asking for a password. The cardholder had to wait for the redirection to the issuing bank, enter the password, and wait for the redirection back to the merchant's site. Today some banks use two-factor authentication, adding to the friction by requiring a one-time password in addition to a static password. Consumers "hate" 3-D Secure [3], and merchants are wary of transaction abandonment.

Another reason for the limited deployment of 3-D Secure is the withering criticism with which it was met for prompting for the password in an inline frame within a window of the merchants web site, which facilitates phishing attacks [4, 5, 6]. Yet another reason may be its cost to merchants and banks. Merchants who want to use 3-D Secure are required by the credit card networks to use a "merchant plug-in (MPI)" provided by a licensed provider, and banks must use an access control server (ACS).

To address the limited deployment of 3-D Secure, the credit card networks have tasked EMVco with the specification of version 2 of the protocol [7]. But the new version has flaws that make it less likely to be successfully deployed than the original version.

3-D Secure version 2 (3DS2) reduces friction for low risk transactions by introducing a *frictionless flow*. But friction is reduced by eliminating cardholder authentication altogether for those transactions. Instead of authenticating the cardholder, the frictionless flow creates a back channel through which the merchant sends transaction, device and cardholder information to the issuing bank without going through the cardholder's browser. The bank uses the information to decide whether to require a subsequent *challenge flow* that does authenticate the cardholder.

The frictionless flow is depicted in a Visa infographic [8], which shows many little icons traveling from the merchant to the issuer and states that "3-D Secure 2.0 delivers 10 times more data, such as device channel and payment history, than a previous version". This should raise a red flag among privacy advocates and government regulators. The issuing bank does not need a history of payments made with cards issued by the same bank, and sharing payments made with cards issued by other banks without cardholder's consent could be deemed a violation of the cardholder's privacy by the merchant in some jurisdictions. As further discussed below in Section 2.3, the current version of 3-D Secure (version 2.2.0) does not actually specify any payment history data to be sent in the frictionless flow, but the specification is still incomplete, and it is not clear if such data will be specified in a future version.

An important selling point of 3DS2 has been that it supposedly features biometric authentication of the cardholder by fingerprint scanning or facial recognition. Such authentication would be implemented by a bank app, but how the bank app would be used by 3-D Secure is declared to be outside the scope of the specification. As further discussed below in Section 2.1, figures showing user interface templates suggest that, after being taken to the bank's web site, the cardholder would be shown written instructions asking him/her to manually open the bank's native app in order to be authenticated. It is hard to imagine the cardholder reading and following the instructions instead of abandoning the transaction. Such an awkward user experience would create at least as much friction as version 1 of 3-D Secure.

3DS2 is very complex, and requires a lot of infrastucture. While version 1 of 3-D secure required an ACS server, version 2 now requires three servers (an ACS server, a Directory Server (DS), and a 3DS Server). The cost of this infrastructure will have to be borne by merchants and banks.

Thus 3-D Secure 2 raises privacy issues and adds complexity and cost for merchants and banks without achieving the goal of frictionless authentication of the cardholder.

Yet it is possible to provide strong cardholder authentication for all transactions with zero friction and negligible infrastructure cost without raising privacy issues, using a cardholder authentication scheme that uses a cryptographic credential and optional biometric authentication as we propose in this paper.

In brief, the cardholder authenticates with a cryptographic credit card credential stored in the cardholder's browser, where it is protected by the *same-origin policy* of the web and controlled by a *service worker* registered with the browser by the issuing bank. The credential consists of a private key and a credit card certificate that binds the associated public key to a hash of credit card data. A hash of the data is used instead of the data itself to protect the data against an attacker who gains physical possession of the cardholder's device. At checkout, after the cardholder enters the credit card data, the merchant's site redirects the browser to the issuer's site with the data and a description of the transaction. But the redirected request never reaches the issuer's site. It is intercepted by the service worker and handled within the browser. The service worker checks the data against the hash, asks the cardholder to confirm the transaction, signs the transaction with the private key and sends the signature and the certificate to the merchant, which retains the signature on the transaction for non-repudiation purposes.

If the bank has a native app in the cardholder's device, the credit card credential is kept in storage controlled by the app rather than in the browser, and the app authenticates the cardholder using the credential in the same way as the service worker would in the absence of a bank app. Optionally, if the cardholder's device provides a biometric API for native apps, the bank app may further authenticate the cardholder by means of a fingerprint scan or face recognition. The role of the service worker when the bank has a native app is just to further redirect the merchant's authentication request to the bank API, then redirect the authentication result back to the merchant's site.

In the cardholder authentication scheme that we propose in this paper the cardholder is authenticated with a strong cryptographic factor, possession of the private key, in addition to the weak factor used today as the only factor in most online credit card transactions, knowledge of the credit card and cardholder data. We refer to this combination of a strong factor and a weak factor as $1\frac{1}{2}$-factor authentication ($1\frac{1}{2}$FA). If the issuing bank has a native app in the cardholder's device the cardholder may be further authenticated with an additional biometric factor, in what we call $2\frac{1}{2}$-factor authentication ($2\frac{1}{2}$FA). In either case the cardholder is strongly authenticated without having to enter a static password or a one-time password, and with no password exposure to back-end breaches or reuse at malicious sites.

The rest of the paper is organized as follows. Section 2 provides a brief analysis of the 3DS2 specification, including a description of the frictionless and challenge flows in Subsection 2.1 and a discussion of usability, privacy, security, and cost in Subsections 2.2–2.5. Section 3 describes the proposed cardholder authentication scheme. Subsection 3.1 describes the infrastructure required to implement the scheme. Subsections 3.2 and 3.3 describe the credit card credential and how it is stored in the browser. Subsection 3.4 explains how the credential is provisioned to the cardholder by the issuing bank. Subsection 3.5 describes the protocol for cryptographic authentication of the cardholder, Subsection 3.6 explains how a native app provided by the issuing bank can be used to add biometric authentication, and Subection 3.7 describes the case where the cardholder uses a merchant app instead of accessing the merchant's site through a web browser. Subsections 3.8–3.11, evaluate the usability, security, privacy and cost of the proposed scheme. Section 4 concludes with a comparison between 3-D Secure 2 and the proposed cardholder authentication scheme.

2 Brief Analysis of 3-D Secure 2

The version of the specification referenced in this section is version 2.2.0, published December 13, 2018 [7]. It consists of two documents, the Protocol and Core Functions specification, more briefly called the *Protocol Specification*, and the *SDK Specification*.

2.1 Frictionless and Challenge Flows

In a traditional web payment by credit card, the cardholder enters credit card and cardholder data and clicks a button to submit the payment. Then the merchant's web site sends a payment authorization request that goes through the merchant's acquirer bank and the card's network to the card's issuing bank. The issuing bank's response to the authorization request takes the same route in reverse to reach the merchant. We shall refer the authorization request/response roundtrip as the *authorization processing*.

When 3-D Secure is used, it takes place between the payment submission by the cardholder and the authorization processing. 3-D Secure causes the authorization processing to be omitted if cardholder authentication fails, but is otherwise entirely separate and independent from the authorization processing.

3-D Secure 1.0 has a simple flow where the cardholder's browser is redirected to the issuer's ACS, which typically prompts the cardholder for a static password, plus possibly a second factor such as a one-time password, and redirects back to the merchant's site with the result of the authentication.

3-D Secure 2 has a *frictionless flow* that may or may not be followed by a *challenge flow*. In the frictionless flow the merchant's site sends transaction, device and cardholder information to the issuer, using a back channel to transmit the information instead of a redirection through the cardholder's browser. The back channel is not a direct connection. The information goes from the merchant's site to the 3DS server, then to the Directory Server, then to the issuer's ACS, before finally reaching the issuer.

Based on the information provided by the frictionless flow the issuer decides whether cardholder authentication is required. If it is not required the merchant initiates authorization processing right away. If it is required, the merchant responds to the payment submission with a redirection to the issuer's ACS, which initiates the challenge flow. As in 3-D Secure 1.0, the issuer's ACS authenticates the cardholder and redirects back to the merchant's site, which initiates the authorization processing if the authentication succeeded, or displays an error message to the cardholder otherwise.

Version 1 of 3-D Secure, released 20 years ago, was only concerned with web apps, but 3DS2 is concerned with native apps as well.

The SDK Specification is concerned with how a native app downloaded from the merchant's site into the cardholder's device can use a *3DS SDK*. The SDK implements functionality that includes gathering device information to be sent to the issuer in the frictionless flow, and interacting with the issuer's ACS in the challenge flow.

After the cardholder submits payment information on the merchant's app, the app sends transaction and device information to the merchant's site, which communicates with the issuer to execute the frictionless flow, implemented as in the case where the cardholder uses a web app.

The SDK implements the challenge flow without using a web browser. It sends a message to the issuer's ACS asking for a "challenge". The ACS responds with a message asking the cardholder to enter authentication data such as a password. After the SDK sends the requested data, the ACS may ask for additional authentication data, such as a one-time password for two-factor authentication. Eventually the issuer's ACS tells the SDK whether the cardholder authentication has succeeded or failed.

3-D Secure 2 is also concerned with the case where the issuing bank has a native app on a device controlled by the cardholder. (The specification does not make it clear whether this must be the same device used for payment submission or may be a different device.) The issuer's app may be used in the challenge flow to authenticate the cardholder, but the issuer's ACS is still involved in the flow. The ACS is asked to authenticate the cardholder as described above, but it delegates authentication to the native app instead of asking the cardholder for authentication data such as a static password or a static password plus a one-time password.

The specification does not specify how the ACS uses the native app, but it makes two suggestions. One suggestion can be found in section 3.2 of the Protocol Specification, which states that the issuer's ACS may send "a push notification to a banking app that completes authentication and then sends the results to the ACS". Another suggestion can be found in user interface templates depicted in Figures 4.6, 4.12 and 4.17, which contain the following instructions to the cardholder: "For added security you will be authenticated with YourBank application. Step 1 — Open your YourBank application directly from your phone and verify this payment. Step 2 — Tap continue after you have completed authentication with your YourBank application."

2.2 Usability Analysis

The frictionless flow, when not followed by the challenge flow, requires no cardholder interaction. This improves the user experience for low risk transactions when compared to 3-D Secure 1.0, which requires the cardholder to authenticate to the issuer's ACS for all transactions. On the other hand the frictionless flow adds the latency of a roundtrip through three intermediate servers for all transactions, which reduces usability for high risk transactions.

3DS2 allows the cardholder to be authenticated biometrically using a native app provided by the issuing bank. However, the user experience for doing so is extremely awkward. The cardholder has to read instructions and manually locate and open the bank's native app before authenticating. This cannot compete with payment methods such as Paypal or Apple Pay that feature tight integration of biometric authentication with the checkout process.

2.3 Privacy Analysis

In the frictionless flow the merchant shares cardholder information with the issuing bank. This may or may not violate the cardholder's privacy, depending on what information is shared, and whether the cardholder is notified and asked for consent.

A Visa infographic [8] states that the information sent to the issuer includes the cardholder's payment history, which should raise a red flag. Annex A.7 of the Protocol Specification specifies data elements that the merchant sends to the issuer in the frictionless flow.

Sections A.7.1–A.7.4 include 32 data elements, none of which are payment history entries. (Sections A.7.5–A.7.10 are unrelated to the frictionless flow and may have been included in Annex A.7 by mistake.) However some of the data elements are not defined in detail yet, suggesting that the specification of the frictionless flow has not been finalized and could include a payment history in the future. One checkout software provider states that its implementation of 3DS2 sends over 100 data elements to the issuing bank, including the payment history [9].

2.4 Security Analysis

The frictionless flow does not authenticate the cardholder. Therefore 3DS2 provides less security for low risk transactions than 3-D Secure 1.0.

In the frictionless flow the merchant sends transaction, device and cardholder information to the issuer, which the issuer uses to assess transaction risk. Today, without 3-D Secure, merchants or merchant processors make their own risk assessment before submitting a transaction for authorization processing, and issuers make their own risk assessment when they receive the payment authorization request. 3DS2 will not increase the sum total of information that is used today for risk assessment, but there may be a security advantage at allowing the issuer to combine its own information with information received from the merchant.

Both in 3-D Secure 1.0 and 3DS2, the issuer is free to use any method it wants to authenticate the cardholder. Password authentication was used traditionally in 3-D Secure 1.0, and two-factor authentication has begun to be used recently. The word “challenge” in “challenge flow” might suggest that 3DS2 has replaced password authentication with a cryptographic challenge-response protocol, but this is not the case. The glossary in the Protocol Specification defines “challenge” as “the process where the ACS is in communication with the 3DS Client to obtain additional information through Cardholder interaction” and “challenge flow” as “a 3-D Secure flow that involves Cardholder interaction as defined in Section 2.5.2”. There is no reason to believe that passwords will not continue to be used in version 2 of 3-D Secure, with their well-known vulnerabilities to phishing attacks, back-end breaches and password reuse.

EMVCo and the FIDO Alliance announced collaborations in July 2016 [10], June 2018 [11] and April 2019 [12], with the latest collaboration also involving the W3C. The FIDO Alliance has developed standards for web authentication by means of a cryptographic key pair (UAF, U2F and FIDO2) [13]. The 3-D Secure Protocol Specification mentions FIDO authentication, but only in connection with authentication of the cardholder to the merchant prior to the credit card transaction. The Protocol and SDK specifications make no other mention of cryptographic authentication of the cardholder.

Biometric authentication of the cardholder by a native app provided by the issuing bank, as described above, would be more secure than password-based or two-factor authentication methods currently used in 3-D Secure 1.0.

2.5 Cost

3DS2 is very complex, specially for the merchant, which is called the Requestor in the Protocol and SDK specifications. Section 2.1.1 of the Protocol Specification mentions the following components or entities involved in the 3DS Requestor Environment: 3DS Requestor, 3DS Client, 3DS Server, 3DS Requestor App, 3DS Method, 3DS SDK, 3DS Integrator. The app-based authentication flow comprises 25 steps and 90 "requirements". The browser-based authentication flow comprises 22 steps and 71 "requirements". This complexity will result in high development, outsourcing and licensing costs for merchants, merchant processors and issuing banks. 3DS2 also requires 3DS Servers, Directory Servers and Access Control Servers, whose cost will have to be borne by merchants and issuing banks.

3 Frictionless Cardholder Authentication

In this section we describe the proposed scheme for authenticating the cardholder by means of a cryptographic credit card credential, plus optional biometric authentication if the issuing bank has a native app on the cardholder's device.

All communications over the Internet take place over secure TLS connections.

3.1 Infrastructure

The infrastructure required to implement the proposed scheme consists of a public database of issuing banks that support the scheme (the *scheme database*). Each issuer is identified in the database by its issuer identification number (IIN), which consists of the first six digits of the credit card number. (Eight digits will be used in the future.) The database maps the IIN of each issuer to two data items used by the scheme:

1. The URL of a cardholder authentication endpoint, to which the merchant redirects the cardholder's browser to authenticate the cardholder and obtain a signature on the transaction.

2. The public key associated with the private key that the issuer uses to sign credit card certificates.

Merchants can download the scheme database or use an HTTP API to query the database. The database, downloads of the database, and queries to the HTTP API must be integrity-protected but need not be confidentiality-protected.

3.2 Credit Card Credential

The credit card credential consists of a private key and a certificate that binds the associated public key to a cryptographic hash of credit card data. A hash of the data is used instead of the data itself to protect the data against an attacker who gains physical possession of the cardholder's device. The data is entered as usual by the cardholder when submitting a payment with the credit card. A hash of the data is then computed and compared with the hash in the certificate. If credentials for multiple credit cards from the same issuer are stored in the browser, the hash also serves to uniquely identify the one to be used for the current transaction. The data hashed into the certificate is the data printed on the physical credit card, viz. the credit card number, the expiration date, the security code, and the cardholder name. Cardholder data such as address, telephone number, email address, etc., is not included in the hash, but it may be required as usual by the merchant, and if required it may be used by the merchant to accept or reject the transaction before asking for authorization, and by the issuer when processing the authorization.

The credit card certificate is an X.509 certificate as profiled by the IETF [14] with an empty subject field, containing the hash of the credit card data in a subjectAltName extension. It is signed by the issuer and the signature can be verified with the public key associated with issuer's IIN in the scheme database. By issuing the certificate, the issuer plays the role of a certificate authority (CA), but there is no CA hierarchy and no certificate chain to be verified. The certificate becomes useless when the card expires or is invalidated. Hence there is no need to revoke it or assign a meaningful expiration time to it. The expiration time of the certificate must not coincide with the card expiration date, to avoid revealing the latter to an attacker who steals the cardholder's device. A bogus validity period is used instead for the certificate with a "notAfter" field that is far enough in the future and a "notBefore" field that is far enough in the past, so that the same bogus validity period can be used for all certificates from the same issuer.

3.3 Credential Storage

The credit card credential is stored in the browser, where it is protected by the same origin policy of the web [15, 16] and used by a service worker [17, 18]. The credential can be stored in an "object store" of an IndexedDB database [19, 20], as an object having the private key and the certificate as properties. The hash is included in the certificate, but it is also stored as a third property used as a database primary key for fast retrieval of the object. If the Web Cryptography API [21] is used for cryptographic computations, the private key can be wrapped in a CryptoKey object, from which it can be made non-extractable.

3.4 Credential Provisioning

The cardholder may obtain the credit card credential on demand, by asking for it on the issuer's web site, or on the fly, when a merchant asks for cardholder authentication.

In on-demand provisioning, the cardholder logs in to the issuer's site and requests a credential for a particular credit card, possibly among multiple credit cards issued by the same issuer to the same cardholder. The issuer's site computes the hash of the credit card data and responds to the request with a script (i.e. JavaScript code embedded in a web page) that contains the hash. The script displays an error message if it finds a credential with the same hash already stored in the browser. If no credential is found, the script creates a key pair and sends a certificate signing request to the issuer's site, containing the hash, the public key and a proof of possession of the private key. The site verifies that the cardholder is still logged in and may ask for a refresh of the login session or additional authentication. The site verifies that the hash is the same one that was computed earlier and downloads a signed certificate binding the hash to the public key, with the above-mentioned bogus validity period. The script creates a credential comprising the certificate and the private key and stores it in the browser. There may already be a service worker configured to intercept requests that target the issuer's cardholder authentication endpoint; if not, the script registers one.

On-the-fly provisioning takes place when the merchant's site redirects the cardholder's browser to the cardholder authentication endpoint but there is no service worker to intercept the request, or there is a service worker that cannot find a credential with the hash of the credit card data. If there is no service worker, the redirected request reaches the issuer's site. If a credential cannot be found, the service worker does a further redirection to a secondary cardholder authentication endpoint. In either case the issuer's site receives the cardholder authentication request, which contains the credit card data, a description of the transaction and a callback URL that targets the merchant's site.

The issuer's site asks the cardholder to log in if a login session is not already in progress. Then it responds to the request by displaying a transaction confirmation page, augmented with a check box that the user can check to ask for the provisioning of a credit card credential to the browser. Additional authentication may be required if the cardholder checks the box.

If the cardholder confirms the transaction and checks the box, a script in the transaction confirmation page provisions a credit card credential to the cardholder's browser and registers a service worker if there is not one already that is configured to intercept requests to the cardholder authentication endpoint. Then it signs the transaction with the private key and sends the signed transaction and the credit card certificate to the callback URL.

If the cardholder confirms the transaction but does not check the box, the script creates a one-time credit card credential, uses it to sign the transaction, and sends the signed transaction and the credit card certificate to the merchant as if the cardholder had checked the box. But it does not save the credential and it does not register a service worker.

The cardholder may repeatedly confirm transactions without checking the credential-creation box. In that case the cardholder is repeatedly authenticated by logging in or having logged in to the issuer's site. The merchant does not know whether the transaction has been signed by the issuer's site with a one-time credential, or by a service worker with a credential stored in the cardholder's browser, unless explicitly told by the issuer.

3.5 Cryptographic Authentication with a Credit Card Credential

The protocol for cryptographic authentication of the cardholder comprises the following steps, illustrated in Fig. 1.

1. In the checkout page of a purchase transaction, the merchant's site asks the cardholder to enter credit card and cardholder data as usual. Knowledge of this data is the weak authentication factor that is used today for authentication of web transactions. Combined with strong authentication by means of the credit card credential it provides what we call $1\frac{1}{2}$-FA. The cardholder enters the data and clicks "continue" to go a confirmation page.

2. The merchant looks up the IIN prefix of the credit card number in the scheme database. If there is no entry for the IIN in the database, the merchant may proceed without

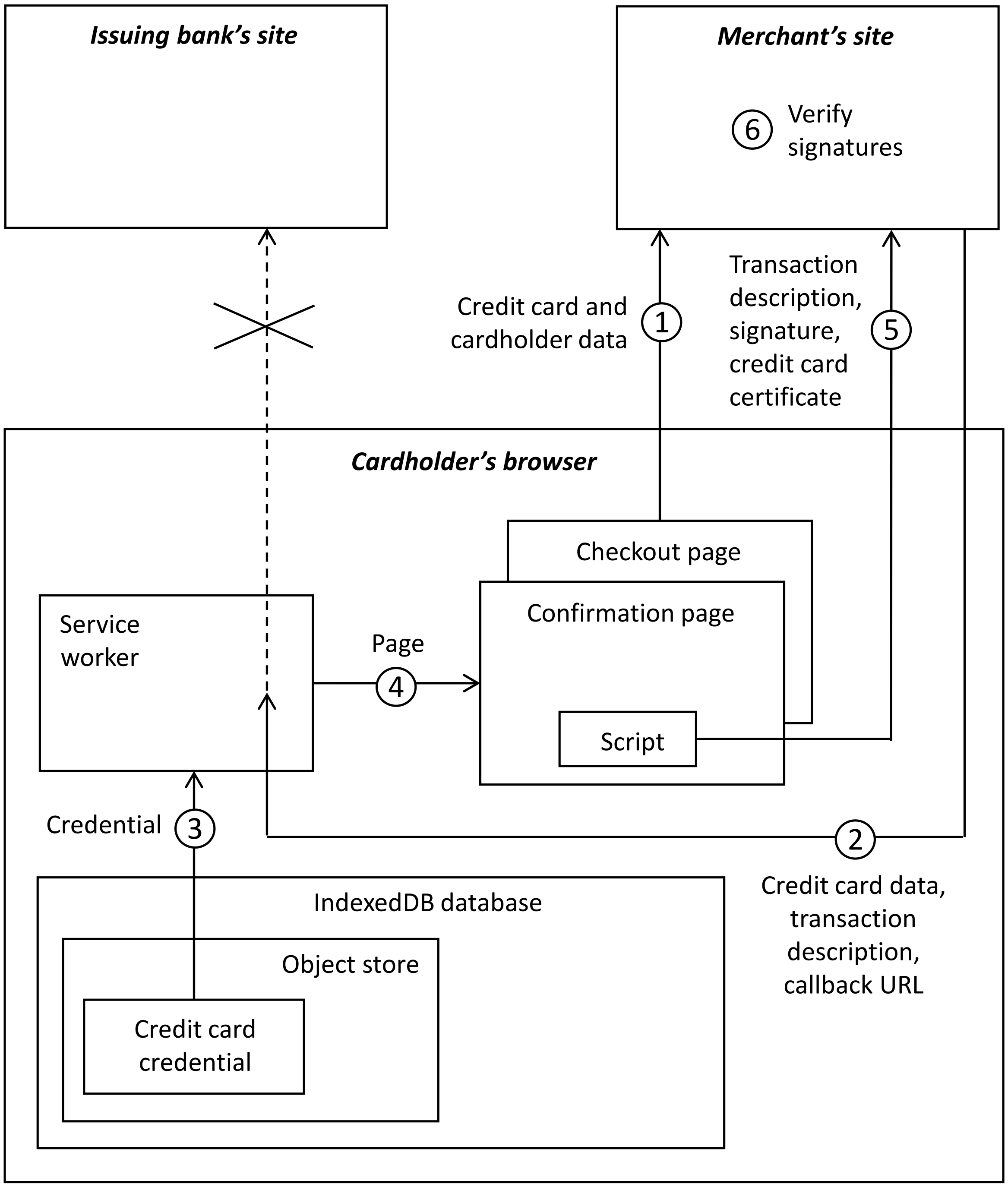

Figure 1: Cryptographic cardholder authentication

cardholder authentication or use a different method of cardholder authentication such as 3-D Secure. If there is an entry for the IIN prefix, the merchant redirects the cardholder's browser to the cardholder authentication endpoint of the issuer, using a script in the checkout page to submit a POST request that conveys the credit card data, a description of the transaction, and a callback URL that targets the merchant's site. If there is no service worker in the cardholder's browser configured to intercept requests to the authentication endpoint, the request reaches the issuer's site, which attempts on-the-fly credential provisioning as described above in Section 3.4.

3. If there is a such a service worker, it intercepts the request, hashes the credit card data, and looks for a credential containing the hash in the object storage of the IndexedDB database where the issuer stores credit card credentials. If no credential is found there, the service worker forwards the request received from the merchant to a secondary authentication endpoint of the issuer's site, and the issuer's site attempts on-the-fly credential provisioning as described above in Section 3.4.

4. If the service worker finds a credential, it creates a transaction confirmation page and delivers it to the browser in response to the intercepted request. The browser renders the page as if it came from the issuer's site, showing the URL of the cardholder authentication endpoint in the address bar. The page describes the transaction using the description received from the merchant and has buttons that the cardholder can use to confirm or cancel the transaction. The page contains a script that includes the hash of the credit card data and the description of the transaction as JavaScript literals.

5. If the cardholder cancels the transaction, a script in the confirmation page sends a POST request to the callback URL notifying the merchant that the transaction has been canceled. If the cardholder confirms the transaction, the script uses the hash to retrieve the credential, signs the description of the transaction with the private key component of the credential, and sends a POST request to the callback URL containing the description of the transaction, the signature, and the credit card certificate.

6. When the merchant's site receives the POST request sent upon confirmation, it verifies the signature on the transaction using the public key in the certificate, and the signature in the certificate using the public key of the issuer found in the scheme database.

3.6 Biometric Authentication with a Bank App

If the issuing bank has a native app in the cardholder's device, the app can be used to provide biometric authentication for transactions that are deemed exceptionally risky by the merchant or the bank. Biometric authentication is used in addition to, rather than instead of cryptographic authentication because cryptographic authentication creates zero friction. Cryptographic and biometric authentication in combination with the knowledge of the credit card and cardholder data amounts to what we call $2\frac{1}{2}$-FA.

If there is a bank app in the cardholder's device, the credit card credential is stored by the app, using any secure storage made available by the platform. When the service worker intercepts the POST request that carries credit card data, the transaction description and the callback URL, it relays the request to the bank's app, using a URL with a custom scheme registered by the app. The app asks the cardholder for confirmation and may authenticate the cardholder biometrically by means of a fingerprint scan or face recognition, using an API made available by the platform. If all goes well the app signs the transaction and causes the browser to open a javascript URL (i.e. a URL where the scheme is "javascript" instead of "http" or "https") where the script in the URL (i.e. the JavaScript code that follows "javascript:") sends a POST request to the callback URL with the signed transaction and the credit card certificate.

3.7 Using a Merchant App

The case where the cardholder uses a native app provided by the merchant is handled in essentially the same way as the case where the cardholder uses a web app, using either the default browser or a bank app to store the credit card credential.

After the cardholder enters the credit card and cardholder information, the merchant's app asks the default browser to open a javascript URL where the script in the URL sends a POST request with the credit card data, the transaction description and a callback URL to the cardholder authentication endpoint of the issuing bank. The POST request is intercepted by a service worker as described above, the cardholder is asked to confirm the transaction by the browser or a bank app, the description of the transaction is signed with the private key component of the credit card credential, and a POST request conveying the signed transaction is sent to the callback URL. In this case, however, the callback URL does not target the merchant's site. It is instead a URL with a custom scheme registered by the merchant's app, so that the cardholder is returned to the merchant's app after confirming the transaction in the browser or the bank's app.

3.8 Usability Analysis

The proposed scheme allows the cardholder to be strongly authenticated by possession of a private key with zero friction and zero latency.

The only action that the cardholder has to take to be authenticated cryptographically is to click a button to confirm the transaction. The cardholder's user experience is the same as in an ordinary online credit card transaction with a separate confirmation page, except that the page is a bank page rather than a merchant page. The bank page is generated in the browser itself by a service worker, without a roundtrip to the bank. It may thus be displayed faster than a confirmation page retrieved by the browser from the merchant's site.

If there is a native app supplied by the issuing bank in the cardholder's device, it can be used to further authenticate the cardholder using a biometric facility made available by the device to the app through an API, such as fingerprint scanning or face recognition. The biometric facility that a device makes available to native apps is typically the same one that is used for unlocking the device. It should therefore be highly usable and familiar to the cardholder.

3.9 Privacy Analysis

In the proposed scheme, the merchant does not share any information with the issuing bank beyond what it already shares today through the payment authorization process.

The credit card certificate that the cardholder sends to the merchant together with the signed description of the transaction only contains data printed on the card. Such data is required to be provided by the cardholder in most online credit card transactions.

3.10 Security Analysis

Since cryptographic authentication causes no friction, the cardholder can be strongly authenticated for all transactions, not only for some transactions deemed to be riskier than others.

No passwords are used in the proposed scheme. The well-known vulnerabilities of passwords are therefore avoided.

No shared secrets are used in the proposed scheme. Therefore the scheme is not vulnerable to backend database breaches.

3.11 Cost

The proposed scheme is simple, easy to describe and specify as can be seen by the length of this paper, relatively easy to implement for the issuer, and trivial to implement for the merchant or merchant processor. This simplicity will translate into low implementation costs.

The only infrastructure required by the scheme is the scheme database, which merchants may download or access online. The contents of the database must be integrity-protected but need not be kept secret. There will thus be very low infrastructure costs to be borne by merchants and banks.

4 Conclusion

The addition of EMV chips to credit cards has reduced credit card fraud for in-store transactions. But the rate of online credit card fraud has increased, because cardholders are authenticated online by their knowledge of credit card and cardholder data, which is a weak secret.

Reducing online fraud will require strong authentication of the cardholder. Version 1 of the 3-D Secure protocol, introduced 20 years ago, provides authentication of the cardholder, typically by password or two-factor authentication. But it is rarely used in the US abd unevenly used in other countries because merchants fear transaction abandonment caused by poor usability. It is also expensive to implement for merchants and card issuing banks.

Version 2 of 3-D Secure, not yet deployed, addresses the usability problem of version 1 by introducing a frictionless flow to be used for low risk transactions. But the frictionless flow does not actually authenticate the cardholder, and may violate the cardholder's privacy by requiring the merchant to share cardholder information with the issuer through a back channel. Furthermore, version 2 is very complex and requires a heavy infrastructure whose cost will have to be borne by merchants and banks.

It is possible to do much better. We have proposed a cardholder authentication scheme that provides strong authentication of the cardholder without causing any friction. The cardholder is authenticated with a cryptographic credential consisting of a private key and a certificate that binds a cryptographic hash of credit card data to the associated public key. The credential is stored in the cardholder's browser, or in a bank app if available on the cardholder's device, and used automatically to sign a description of the transaction upon confirmation of the transaction by the cardholder.

Since the scheme introduces no friction it can be used for all transactions, not only some transactions deemed to be riskier than others. Cryptographic authentication can be combined with biometric authentication provided by a bank app and can be used for credit card purchases made on the merchant's web site or using a merchant app.

The proposed scheme is trivial to implement for merchants, and only requires a lightweight infrastructre consisting of a database that maps the IIN of each participating bank to the URL of a cardholder authentication endpoint and to a public key used for verifying the certificates issued by the bank. It can therefore be deployed inexpensively.

References

[1] Mastercard and VISA. SET Secure Electronic Transaction Specification. Version 1.0, May 1997. http://www.maithean.com/docs/set_bk1.pdf, http://www.maithean.com/docs/set_bk2.pdf, http://www.maithean.com/docs/set_bk3.pdf.

[2] Visa. Visa 3-D Secure 1.0. https://technologypartner.visa.com/Library/3DSecure.aspx.

[3] Miles Brignall. MasterCard and Visa to simplify hated verification systems. The Guardian, 13 Nov 2014. https://www.theguardian.com/money/2014/nov/13/mastercard-visa-kill-off-verification-systems.

[4] Ben Laurie. More Banking Stupidity: Phished by Visa. March 28, 2009. https://www.links.org/?p=591.

[5] Jonathan Baker-Bates. Phishing with 3-D Secure. 13 August 2009. https://webtorque.org/phishing-with-3-d-secure/.

[6] Steven J. Murdoch and Ross Anderson. Verified by Visa and MasterCard SecureCode: or, How Not to Design Authentication, 2010.

[7] EMVCo™. EMV® 3-D Secure. https://www.emvco.com/emv-technologies/3d-secure/.

[8] Visa. New and improved 3-D Secure. Originally retrievable from https://www.visaeurope.com/media/pdf/visa-infographic.pdf. Retrieved on June 8, 2019 from https://usa.visa.com/dam/VCOM/global/visa-everywhere/documents/visa-3d-secure-2-program-infographic.pdf. Archived at https://web.archive.org/web/20190608194449/https://usa.visa.com/dam/VCOM/global/visa-everywhere/documents/visa-3d-secure-2-program-infographic.pdf.

[9] Checkout.com. 3-D Secure payments. https://docs.checkout.com/docs/3d-secure-payments.

[10] FIDO Alliance. EMVCo and the FIDO Alliance Collaborate on Mobile Payment Authentication. July 2016. https://fidoalliance.org/fido-emvco-mou/.

[11] FIDO Alliance. EMVCo and the FIDO Alliance to Address FIDO Authentication in EMV 3-D Secure Use Cases. June 2018. https://fidoalliance.org/emvco-and-the-fido-alliance-to-address-fido-authentication-inemv-3-d-secure-use-cases/.

[12] FIDO Alliance. EMVCo, FIDO Alliance, and W3C Form Interest Group to Enhance Security and Interoperability of Web Payments. April 2019. https://fidoalliance.org/emvco-fido-alliance-and-w3c-form-interest-group-to-enhance-security-and-interoperability-of-web-payments/.

[13] FIDO Alliance. Specifications Overview. https://fidoalliance.org/specifications/.

[14] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and CRL Profile, May 2008. http://datatracker.ietf.org/doc/rfc5280/.

[15] Mozilla. Origin. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Origin.

[16] A. Barth. The Web Origin Concept. IETF RFC 6454. https://tools.ietf.org/html/rfc6454.

[17] Mozilla. Service Worker API. https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API.

[18] W3C. Service Workers. https://www.w3.org/TR/service-workers/.

[19] Mozilla. IndexedDB API. https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API.

[20] W3C. Indexed Database API. https://www.w3.org/TR/IndexedDB/.

[21] W3C. Web Cryptography API. https://www.w3.org/TR/WebCryptoAPI/.