EAP Method Update
Internet-Draft
Intended status: Standards Track
Expires: 4 September 2025

J.-F. Rieckers
DFN
S. Winter
RESTENA

3 March 2025

EAP-FIDO
draft-ietf-emu-eap-fido-01

Abstract

This document specifies an EAP method leveraging FIDO2 keys for authentication in EAP.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-emu-eap-fido/.

Discussion of this document takes place on the EAP Method Update Working Group mailing list (mailto:emu@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/emu/. Subscribe at https://www.ietf.org/mailman/listinfo/emu/.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 4 September 2025.

Copyright Notice

Table of Contents

1. Introduction

The Extensible Authentication Protocol (EAP) [RFC3748] is a widely used standard that allows a server to authenticate a client using different authentication methods. There is a huge variety of EAP methods available, that serve different purposes and have different security implications.

Two common EAP methods are EAP-PEAP and EAP-TTLS [RFC5281], that both use EAP-TLS [RFC5216] to provide confidentiality of the inner authentication. This inner authentication is most commonly password-based, meaning that an attacker that manages to compromise the TLS connection can eavesdrop on the authentication and observe the password. The authentication of the server to the client within the TLS handshake thus is a vital security function of these EAP methods.

Operational practice has shown that this is a common problem and possible flaw. The specification for EAP-TLS [RFC5216] does not include guidance on how to decide if a certificate is valid for this specific authentication. Instead it assumes the client has knowledge of the expected server name. This assumption is reasonable, if all devices are managed centrally and the administrators can easily know and configure all security-relevant parameters in advance. Devices in BYOD-environments like eduroam are not managed, so administrators cannot push the desired configuration on the devices, but instead have to rely on the users to configure their devices accordingly. This implies that the administrators are aware of all needed configuration items, provide the users with all this information, and the users must follow these guides. Failure to configure the parameters correctly will result in connection failure, and frustration, first on the user’s end, then on the side of the help desk, that has to deal with the users’ problems. If the user tries to omit these parameters and the implementation allows a fallback to just omitting the validation, the device will simply work, but now be dangerously susceptible to attacks, e.g. by rogue access points with the same SSID, to which the client will send their password to, regardless of the presented server certificate. The second outcome is especially dangerous, because the operator cannot easily check whether the device correctly performed the certificate check, and since the connection just works, the user will not suspect that their local configuration is faulty.

There are two major issues here, that this specification wants to address. Firstly, the use of passwords as authentication method implies that the password needs to be sent to the server. If an attacker observes this exchange, they can impersonate the user at any time. Therefore, this specification uses FIDO authentication, which is based on asymmetric cryptography. With this method, even if an attacker is able to compromise the TLS connection, the attacker cannot impersonate the user based on the observed data. Since the private key is never revealed, phishing attacks are impossible too.

The second major issue is the missing implicit derivation of the expected server name that is used to validate the server’s certificate. With EAP-FIDO, the supplicants now have a clear specification on how to decide whether or not a server certificate is

considered valid for the current authentication flow. This is achieved by using the trust anchors available on most devices and a method to determine the valid server name based on implicit information of the authentication configuration.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document uses terminology from EAP, as well as terminology from CTAP and WebAuthn. The FIDO specific terminology is defined in [FIDO-Glossary]. There is some terminology that is ambiguous in the different contexts, to disambiguate it, the following terminology will be used in this document. These terms will always be capitalized as shown here.

FIDO Authenticator: Authenticator as specified by [WebAuthn], Section 4: a cryptographic entity, existing in hardware or software, that can register and later assert possession of the registered key credential.

This is not the same as the EAP authenticator, which is the entity that initiates the EAP conversation, i.e. the Access Point in the case of Enterprise-WiFi.

Discoverable Credential: a public key credential source that is discoverable and usable in authentication ceremonies where the Relying Party does not provide any Credential IDs. See [WebAuthn], Section 4

Server-Side Credential: a public key credential source that is only usable in an authentication ceremony where the Relying Party supplies the Credential ID. This means that the Relying Party must manage the credential’s storage and discovery, as well as be able to first identify the user in order to discover the Credential IDs to supply this to the EAP Supplicant.

// We need a good term to define a "single authentication", meaning
// the single process of asking the FIDO Authenticator to sign the
// client data, with the option of previous discovery, in contrast to
// the overall authentication process. A complete EAP-FIDO flow may
// consist of multiple of these "single authentications", and we
// should have a clear terminology to say "this is just the single

```
// instance" and "this is the overall authentication"
//
// -- Janfred
```

3. Overview over the EAP-FIDO protocol

This section will cover both a rough overview of the protocol design, as well as the needed configuration parameters for the EAP-FIDO server and EAP-FIDO client.

3.1. Overall protocol design

The EAP-FIDO protocol comprises two phases: the TLS handshake phase and FIDO-exchange phase.

During the TLS handshake phase, TLS is used to authenticate the EAP-FIDO server to the client.

During the FIDO-exchange phase, the actual FIDO authentication is executed and the client authenticates itself to the server.

Once the FIDO exchange is completed successfully, the client and server can derive keying material from the TLS handshake phase implicitly.

3.1.1. TLS handshake phase

During the TLS handshake phase, the client and server establish a TLS tunnel. This is done using EAP-TLS [RFC5216], [RFC9190], [RFC9427] with the modifications described in Section 4.1.4. As part of the TLS handshake protocol, the EAP-FIDO server will send its certificate along with a chain of certificates leading to the certificate of a trusted CA. The client will check this certificate using the rules in Section 4.1.5.

Once the TLS tunnel is established, the client and server proceed to the FIDO-exchange phase to perform the authentication of the client.

3.1.2. FIDO-exchange phase

In this phase, the TLS record layer is used to securely tunnel information between the EAP-FIDO client and EAP-FIDO server.

For the FIDO-exchange phase, the client has two options, depending on the configuration and the capability of the FIDO token.

If the FIDO Authenticator supports Discoverable Credentials and EAP-FIDO is configured to use these for authentication, the client generates a challenge from the TLS keying material and triggers a FIDO challenge.

If the client is not configured to use Discoverable Credentials, the client first needs to send its username to the server. The server will answer with a list of FIDO Credential IDs and the client will attempt to use one of these keys to authenticate.

Depending on the details of the first single FIDO authentication, the server MAY trigger a second authentication, to enforce token-specific policies.

3.2. Client and server configuration and preconditions

As the EAP-FIDO protocol aims to provide a means of authentication that is easy to setup and maintain for both users and server operators, there are only few configuration items required. However, if several setups require a different setup than the one that is outlined as the default here, additional configuration parameters can be set to modify the default behavior.

To better distinguish the server and client configuration items throughout this document, all server configuration options are prefixed with S_, all client options with C_.

3.2.1. Server configuration items and preconditions

The EAP-FIDO server configuration comprises of the following configuration items.

S_FIDO_RPID: The FIDO Relying Party ID to use for checking the FIDO assertion.

This value must be a valid domain string as specified in [WebAuthn]. The value of S_FIDO_RPID MAY be derived dynamically for each authentication flow from the realm part of the NAI. However, administrators SHOULD set this configuration option to an explicit value, because client and server MUST agree on the same RPID, otherwise the signature check will fail.

S_TLS_SERVER_CERT: The certificate used for the EAP-TLS layer.

The certificate SHOULD include a SubjectAltName:dnsName for the domain eap-fido-authentication.<S_FIDO_RPID> (See Section 4.1.5)

Additionally, the server needs access to a database of FIDO credentials. Depending on the usecase and other requirements, different fields are needed in the database.

DB Field	Description	Mandatory
username	A reference to the user	only for Server-Side Credentials (see remark 1)
PKID	The public key identifier of the stored public key	yes
PubKey	The public key used to verify the FIDO assertion	yes
signCount	The last observed signCount sent by the FIDO Authenticator	no (see remark 2)

Table 1: Database fields in the server’s authentication database

Depending on the use case, the database may have additional fields that save the last successful authentication with User Verification or User Presence, in order to implement time-dependent policies (see Appendix A.5 for an example)

Remarks:

1: The username may not be needed in cases where an identification of the individual user is not necessary and Discoverable Credentials are used. Server-Side Credentials need a hint to the user, since the server must send a list of possible PKIDs to the client. For Discoverable Credentials, this is not needed. If a server operator does not care about the identity of the user or has other means to identify a user based on the used PKID, i.e. because the mapping is stored in a separate database, the EAP-FIDO server does not need access to this information.

2: If present, the signCount field SHOULD be writable to the EAP-FIDO server. The signCount functionality is intended to have a means for the server to detect when a credential was cloned and two instances are used in parallel. The value is strictly monotonically increasing for each FIDO Authenticator, so if at some point a signCount is transmitted that is smaller than or equal to the saved signCount, a

second FIDO Authenticator is potentially using the same credential. How servers deal with such case is outside of the scope of this specification.

3.2.2. Client configuration items and preconditions

The client configuration of EAP-FIDO is intended to be as simple as typing in one string and being connected. However, the precondition is that the FIDO credential is already registered with the server. Details on this rationale can be found in Section 6.1.

// A future version of this draft may include some means for the
// server to signal a URL where to register the FIDO key. The flow
// could be the following: The user enters their realm, the server
// recognizes that there is no FIDO credential available, send a URL
// where the user can register their credential, the UI shows this
// URL, the user can click their, log in with their credentials (i.e.
// via SSO), register the FIDO token and then it just works. \o/ Of
// course we need to make sure that no one can hijack that process,
// so the EAP-TLS handshake must be performed, so the client knows it
// talks with the correct server.
//
// -- Janfred

The client has several configuration options. However, there is only one mandatory configuration option and user interfaces SHOULD present this option prominently. If the server relies on Server-Side Credentials, it needs an identity, this option SHOULD be presented in the initial configuration interface as well, marked as optional. The other configuration options SHOULD still be made available, i.e. behind an "Advanced" button, but SHOULD indicate the derived value based on the input in C_FIDO_RPID, to help the user understand what this value usually looks like and whether or not it is necessary to change the derived value.

C_FIDO_RPID: (Mandatory) The FIDO Relying Party ID to use. This is the basis for all derived configuration items.

The value must be a valid domain string as specified in [WebAuthn].

Implementations MAY offer the user a set of known RPIDs, if Discoverable Credentials are used and there are already Credentials registered.

C_IDENTITY: (Optional) The identity (username) of the user.

This configuration item is only needed if Server-Side Credentials are used. It is RECOMMENDED that the identity is just the username and does not contain a realm.

C_NAI: (Optional, Derived) The NAI to use in the EAP layer.

This option MUST be set to anonymous@<C_FIDO_RPID>, unless explicitly configured differently.

C_EXPECTED_SERVERNAME: (Optional, Derived) The expected server name to use to verify the server’s identity.

This option MUST be set to eap-fido-authentication.<C_FIDO_RPID> unless explicitly configured differently.
// The prefix value is not final yet. Might still change, depending
// on the result of the name discussion. Maybe we could also
// reference RFC9525 here, especially Section 6.3 on how to match the
// DNS Domain Name Portion against a certificate. In this case, the
// C_EXPECTED_SERVERNAME would be a DNS-ID according to RFC9525
// terminology.
//
// -- Janfred If manually configured, implementations MUST check that C_EXPECTED_SERVERNAME is either equal to or a subdomain (in any depth) of C_FIDO_RPID and MUST reject configurations that do not match this condition.

C_TLS_TRUST_ANCHORS: (Optional, with default) A set of trust anchors to use for checking the server certificate.

This option MUST default to the set of trust anchors already present on the device, i.e. the set of trust anchors used to verify server certificates for HTTPS, unless the device has no such set available. In this case, the implementation MUST enforce that this option is set. Implementations MUST NOT allow this option to be set to trust certificates from unknown trust anchors. Implementations SHOULD allow to select multiple trust anchors.

4. EAP-FIDO protocol flow and message format

This section describes the EAP-FIDO protocol flow and the message format.

The protocol starts with the TLS handshake phase, and, after the TLS tunnel is established, continues with the FIDO exchange.

4.1. TLS handshake phase

The packet format for EAP-FIDO messages follows the format specified in [RFC5216], Section 3 with the following modifications:

* The Type field is set to <insert EAP code here, Proof of Concept uses the method type 255 "Experimental", once we have a stable message format we'll ask for IANA early allocation> (EAP-FIDO)

* Within the Flags field, the Version bits are set to the major version of EAP-FIDO. For this specification, the major version is 0. Future EAP-FIDO versions MAY increase the version number.

4.1.1. EAP-FIDO Start packet

In the first packet from the server to the client, the S-bit of the Flags MUST be set, indicating the start of the EAP-FIDO protocol. The S-bit MUST NOT be set in any subsequent packet. This packet contains only the EAP header (Code, Identifier, Length, Type) and the EAP-TLS flags.

Future versions of EAP-FIDO may send additional data outside the TLS tunnel with the first EAP message, supplicants SHOULD ignore any additional data sent with this packet.

4.1.2. Version negotiation

The major version of EAP-FIDO is negotiated in the first exchange between server and client. The server sets the highest major version number of EAP-FIDO that it supports in the V field of the flags in its Start message. In the case of this specification, this is 0. In its first EAP message in response, the client sets the V field to the highest major version number that it supports that is no higher than the version number offered by the server. If the client version is not acceptable to the server, it sends an EAP-Failure to terminate the EAP session. Otherwise, the version sent by the client in its response is the version of EAP-FIDO that MUST be used, and both server and client MUST set the V field to that version number in all subsequent EAP-TLS messages.

Given the limited range of the V field (values 0-7), future EAP-FIDO versions MUST NOT increase the major version if there are no changes to the outer message format. Minor version updates that only affect the protocol flow within the TLS tunnel MUST be done with means available during the TLS handshake, i.e. using Application Layer Protocol Negotiation (ALPN).

4.1.3. Fragmentation

Each EAP-FIDO message contains a single leg of a half-duplex conversation. Since EAP carrier protocols may constrain the length of an EAP message, it may be necessary to fragment an EAP-FIDO message across multiple EAP messages.

This is done through the fragmentation mechanism within EAP-TLS. This method is described in [RFC5216], Section 2.1.5.

4.1.4. TLS Handshake Requirements

The client and server perform a TLS handshake following the specification in [RFC5216], Section 2 and [RFC9190], Section 2 with the following modifications:

* TLS version 1.3 or higher MUST be negotiated.

* Mutual authentication is not required. Implementations MUST support EAP-FIDO without TLS client authentication, but MAY allow it, i.e. if EAP-FIDO is used as a 2-Factor authentication method where TLS client certificates are the first factor and the FIDO authentication is the second.

* The certificate of the server MUST be validated according to the verification steps described in the next section.

4.1.5. TLS Server Certificate Verification

Clients MUST validate the certificate sent by the server. For this the client must first check that C_EXPECTED_SERVERNAME is set to the exact domain or a subdomain of C_FIDO_RPID. This ensures that a misconfiguration cannot be used for cross-domain or even cross-protocol attacks. The client then MUST validate that the server certificate is valid for the domain saved in the configuration item C_EXPECTED_SERVERNAME and the certificate chain leads back to a trust anchor listed in C_TLS_TRUST_ANCHORS.
// Again, here we could look if we can reference RFC9525?
//
// -- Janfred

* TODO: OCSP Stapling? Mandatory or not?

4.2. FIDO-exchange

After the TLS handshake is completed, the client and server perform the FIDO-exchange to authenticate the client inside the TLS tunnel.

This section describes the message format and the protocol flow.

4.2.1. Message format

All EAP-FIDO messages in the inner authentication consist of a CBOR sequence with one or two elements. Each message of the inner authentication MUST be sent in a single TLS record. The length of the TLS record MUST match the length of the CBOR sequence.

The elements are:

type: integer to indicate the message type. Table 2 contains a list of the different message types.

attributes: a CBOR encoded map with attributes. A list of the different attributes, their assigned mapkey and the type are listed in Table 3. This element is omitted in the Success indicator message.

Type	Description	Sent by
-2	Error	Both
-1	Failure indicator	Both
0	Success indicator	Both
1	Authentication Request	Server
2	Authentication Response	Client
3	Information Request	Client
4	Information Response	Server

Table 2: Message types

Mapkey	Type	human-readable Label	Description
0	Text String	Identity	User Identity (usually username)
1	Byte String	Additional Client Data	Additional Data to be signed by the FIDO Authenticator
2	Array of Byte Strings	PKIDs	List of acceptable Credential IDs
3	Byte String	Auth Data	Authdata according to [WebAuthn], Section 6.1
4	Byte String	FIDO Signature	
5	Array of Integers or Text Strings	Authentication requirements	Sent by the server to indicate the current authentication requirements, i.e. if user presence or user verification is required
6	Byte String	PKID	Needed to identify the credential
7	Integer	Error Code	A code describing the error, see Section 4.4 for a list of error codes
8	Text String	Error Description	An optional human-readable error description

Table 3: Mapkeys for the attributes

We will now describe the meaning, format and required attributes for each message type.

4.2.1.1. Success indicator

This message is the protected success indicator, as required by [RFC9427], Section 5.2. It is sent by the server to indicate a successful authentication. Since EAP is a strict request-response based protocol, the client needs to reply to a success indicator sent by the server, so the server can send an EAP-Success message. The client will acknowledge the reception of this packet through the acknowledgement mechanism in EAP-TLS with an EAP-TLS acknowledgement packet.

To achieve the compatibility with the protected success indication mechanism of other EAP methods, the attributes field of the message MUST be omitted, that is, this message is only one byte with the value of 0x00.

4.2.1.2. Failure indicator

A failure indicator message signals a non-recoverable error condition for the current authentication exchange.

The attributes field of the message MUST contain at least the Error Code attribute with an error code describing the error and MAY contain the Error Description attribute with a human-readable error description.

4.2.1.3. Error

The Error message signals an error condition and can be sent both from client to server and vice versa. This error condition does not necessarily lead to an authentication failure, since the EAP-FIDO server may decide that the previous authentication is sufficient. (See Appendix A.6 for an example for this use case)

The attributes field MUST contain at least the Error Code attribute with an error code describing the error and MAY contain an Error Description attribute with a human-readable error description.

4.2.1.4. Authentication Request

An authentication request is sent by the server to initialize a new authentication request. With this request, the server sends along information that the client needs to perform the FIDO authentication.

The attributes field in the authentication request message contain the following attributes:

PKIDs: (Optional) A list of acceptable Credential IDs. This can be

used to trigger a re-authentication of a specific credential or to provide a list of the Credential IDs for a specific user, if Server-Side Credentials are used.

Authentication Requirements: (Optional) A list of requirements for the FIDO authentication. See {:auth_requirements} for details.

Additional Client Data: (Optional) Additional data to be signed by the FIDO Authenticator.

If no attributes are transmitted, the attributes field MUST be set to an empty map, instead of omitting it completely.

Since this packet signals the start of a new authentication, the client MUST initialize a new authentication and MUST NOT reuse information from any previous authentication attempt, even if the previous authentication exchange was not completed. It MAY cache some data to perform sanity checks, i.e. to protect itself against misbehaving servers that try to re-initialize an authentication with the same parameters multiple times.

4.2.1.5. Authentication Response

If a client has sufficient information to perform a FIDO authentication, the client sends an authentication response. The authentication response signifies the completion of one authentication. This message can be sent in response to either an Authentication Request or an Information Response.

The attributes field in the authentication response message contain the following attributes:

PKID: The Credential ID of the FIDO Credential used to generate the signature.

Auth Data: The signed auth data as returned from the FIDO Authenticator (see [FIDO-CTAP2], Section 6.2)

FIDO Signature: The signature as returned from the FIDO Authenticator (see [FIDO-CTAP2], Section 6.2)

All three attributes MUST be present in the authentication response message.

4.2.1.6. Information Request

If a client does not have sufficient information to perform the FIDO authentication, the client can send an information request message to the server.

This is the case if Server-Side Credentials are used, since the FIDO Authenticator needs the list of acceptable Credential IDs to access the actual credentials on the FIDO Authenticator.

With the information request the client can transmit additional information that help the server to compile this information.

The attributes field in the information request contains the following attributes:

Identity: The identity of the user (usually a username, configured in C_IDENTITY)

A client MUST NOT send an Information Request packet twice for one authentication. If a client does not get sufficient information to perform the FIDO authentication after the first Information Request and the subsequent Information Response, the client will not get more information by asking the server a second time. In this case, a client MUST respond with an Error message, indicating Insufficient Information.

A server MUST respond with a Failure Indicator message if it receives an Information Request packet when it does not expect one.

4.2.1.7. Information Response

The server answers to an Information Request from the client with an Information Response.

This packet is used to transmit additional information to the client.

The attributes field in the information response can contain any attribute that is also allowed in the Authentication Request packet. If an attribute was both present in the Authentication Request and the Information Response packet, the client MUST discard the previous value(s) of this attribute that were sent in the Authentication Request and use the value(s) in the Information Response packet.

A server MUST NOT send an Information Response packet twice for one authentication. A client MUST respond with a Failure Indicator message if it receives an Information Response packet when it does not expect one.

4.2.2. Protocol Sequence

The FIDO exchange phase starts with the server sending an authentication request to the client. This message is sent along with the last message of the server's TLS handshake.

The Authentication Request can include authentication requirements, additional client data and a list of Credential IDs.

The client then decides if it has sufficient information to perform the FIDO authentication. This can be done by probing the FIDO Authenticator with all information given in the Authentication Request message.

If the FIDO authentication is already possible at this point, the client performs the FIDO authentication process and sends an Authentication Response message with the results from the FIDO authentication to the server. This authentication flow can be used if the FIDO Authenticator has a Discoverable Credential registered for the given Relying Party ID.

If the client needs additional information, i.e. because it uses Server-Side Credentials and therefore needs a list of Credential IDs, the client sends an information request to the server, which includes additional information from client to help the server to fulfill the information request. In the current specification, this is namely an identifier, from which the EAP-FIDO server can perform a lookup for all registered FIDO credentials registered to this identifier.

Upon reception of the Information Request message from the client, the server looks up the registered Credential IDs for the given identity. Depending on the lookup, the requirements for user presence or user verification may change from the previous assumption. The found Credential IDs, and optionally also the updated authentication requirements, are then sent with the Information Response back to the client.

The client can now, with the additional information in the Information Response message, perform the FIDO authentication. The result of the FIDO authentication is then sent to the Server in an Authentication Response message, which includes the PKID, Auth Data and FIDO Signature from the FIDO authentication result.

When a server receives an Authentication Response message, it validates the FIDO data. If the FIDO authentication is successful and the FIDO key has sufficient authorization, the server sends a Success indication message to indicate the Success of the FIDO exchange phase. The client will acknowledge this packet using the EAP-TLS acknowledgement mechanism and the server sends an EAP-Success message.

Depending on the result of the FIDO authentication, the user presence or user verification assertions and the policy for a specific FIDO credential, the server MAY choose to trigger a second FIDO authentication with a different set of authentication requirements. This is done by sending a new Authentication Request message to the client. This message MUST include a PKIDs attribute with only the PKID of the credential used in the previous FIDO authentication process.

The client then triggers a new FIDO authentication process and answers with an Authentication Response message.

The server MUST NOT trigger a challenge with the same Credential ID and Authentication Requirements twice.

4.3. FIDO authentication

This section will describe the actual FIDO authentication process, that is performed between the EAP-FIDO client and the FIDO Authenticator.

(currently mainly a sub, more text is TODO)

The client will use CTAP version 2.0 or above [FIDO-CTAP2] to communicate with the FIDO Authenticator.

The Relying Party ID (RPID) is explicitly configured in C_FIDO_RPID

The client data is a concatenation of three items.

The first item are 8 bytes with the values 0x45, 0x41, 0x50, 0x2d, 0x46, 0x49, 0x44, 0x4F (ASCII: "EAP-FIDO")

The second item is derived from the TLS keying material:

FIDO_CHALLENGE_TLS = TLS-Exporter("fido challenge", NULL, 32)

The third item is the optional additional client data sent by the server. If the server did not send additional client data, this is omitted.

All three items are concatenated and hashed using SHA-256.
// This has no crypto agility, as was correctly pointed out. This
// part comes from the WebAuthn spec, where SHA-256 is fixed. For
// EAP-FIDO we could opt for crypto agility, since the hashing
// algorithm is not neccesarily fixed.
//
// -- Janfred The result is the clientDataHash for the FIDO
authentication.

TODO: format of Authentication Requirements and how to send them using CTAP

Completely TODO: Server side. How to validate. If up/uv was provided, even if not required, the server should update the "last up/uv seen" field for example.

4.3.1. Authentication requirements

CTAP allows different authentication requirements to be requested. The server can request those by sending values in the authentication requirements attribute with either the Authentication Request or the Information Response message.

For standardized options, numerical values are used. For experimental use cases, implementations can use text strings. Implementations MUST ignore text strings they do not recognize.

At the time of writing, the CTAP protocol has two authentication options relevant to this specification: User Presence and User Verification.

If the server includes a value of 1 in the Authentication Requirements attribute array, the supplicant MUST require user presence from the FIDO authenticator. If the server includes a value of 2, the supplicant MUST require user verification from the FIDO authenticator.

4.4. Error conditions

TODO, only a stub, not yet finished.

Errors can be non-recoverable, or recoverable. If an error is non-recoverable, then it MUST only appear in a Failure Indication message. If an error is recoverable, the peer may decide whether or not the current error condition is non-recoverable or not. These errors can be included in both the Failure Indication message and the Error message.

An example of a non-recoverable error is the Unexpected Message error. In this case either the other peer is misbehaving or they have a different state, in both cases, a successful authentication is not possible.

An example of a recoverable error, that may be sent with a Failure Indication message is the No Username configured error. If the client receives the authentication request and there is no FIDO authenticator available with a Discoverable Credential, and there is no username configured, then it is unlikely that the server has more information. In this case the client MAY send a Failure Indication message, signaling its unwillingness to try again. A client MAY also choose to send this error within the Error message, in which case the decision on trying again lies with the server.

In this table are some error conditions, the list is not yet complete and not ordered.

Error code	non-recoverable	human-readable Label	Description
?	no	No username configured	The client configuration had no username configured and no Discoverable Credential was available.
?	yes	Unexpected Message	The client or server received an unexpected message. Either one of the peers is misbehaving or they use incompatible versions. Either way, a successful authentication is not possible under these circumstances.
?	no	Insufficient Information	The client did not have sufficient information to perform a FIDO authentication.

Table 4

5. Implementation Guidelines

TODO

6. Design decisions

This section documents several design decisions for the EAP-FIDO protocol

6.1. Registration of FIDO2 keys is out of scope

The FIDO CTAP protocol has distinct primitives for the registration and the usage of a FIDO2 credential. This specification requires that the registration of the security token has been done out-of-band, for example using the WebAuthn protocol in a browser context.

There are multiple degrees of freedom when registering a token with CTAP version 2. This specification recognizes the following choices at registration time, and defines how to effectuate an authentication transaction for any combination of these choices.

6.1.1. Discoverable Credentials vs. Server-Side Credentials

FIDO2 tokens contain a master key which never leaves the security perimeter of the token. FIDO2 tokens transact by generating asymmetric keypairs which are bound to a scope (often: a domain name, a RADIUS realm). The scoped keying material can be accessed using two different methods:

* Server-Side Credentials: The keying material is not accessible directly, but only by providing a Credential ID that was generated during registration. Depending on the actual implementation inside the FIDO Authenticator, the Credential ID may be used as a seed to re-generate the specific key pair with the help of the FIDO Authenticator's master key. Other FIDO Authenticator implementations may store the keypair and generate a random Credential ID by which the key can be referenced. To trigger an authentication, the client must provide the correct Credential ID to the FIDO Authenticator. It is fair to assume that the number of Server-Side Credentials that a FIDO Authenticator can generate is not significantly limited, so the number of keys should not weigh in to the consideration whether or not it is worth to register a new Server-Side Credential for a given scope.

* Discoverable Credentials: The keying material is stored on the security token itself, along with the scope for which the keypair was generated. During authentication transactions, only the scope (as configured, or as sent by the server) determines which keypair is to be used in the transaction. The key can store multiple keys for the same scope. The number of slots for Discoverable Credentials is limited, and this limit needs to be considered when deciding whether or not a new Discoverable Credential should be registered for a specific use case.

EAP-FIDO supports both Discoverable and Server-Side Credentials.

Registering a Discoverable Credential has several advantages and one disadvantage:

* No username needs to be sent during the authentication transaction

* The transaction requires less round-trips due to skipping the username transmission process

* The amount of data sent from EAP server to EAP peer is significantly smaller, reducing the probability of extra roundtrips or packet fragmentation

* The scopes of the stored credentials can be seen and enumerated by the EAP supplicant, helping the user in finding the right value for configuring the EAP realm

* The Discoverable Credential consumes space on the FIDO Authenticator, which might be limited

6.1.2. User involvement during registration

Token registration can involve one of two levels of asserting the user presence.

* UP (userPresence): the registration ceremony ensures that a person is present at the token while registering the device (e.g. human tissue needs to touch a physical security key while the registration transaction executes).

* UV (userVerification): the security token registers a unique property of the user during the registration ceremony, such that it is asserted that only the exact same person can interact with the token in the future (e.g. by registering a fingerprint or facial recognition)

During authentication transactions, an EAP-FIDO server can request one of three levels of asserting user presence.

* Silent (interaction with a human is not required)

* UP (physical interaction with a person is required)

* UV (physical interaction with the registered user is required).

An authentication transaction can not request a higher level than was set at registration time; i.e. a token registered in UP mode can not transact in UV mode.

EAP-FIDO supports all three transaction modes, and the server can signal its required minimum assertion level for each individual authentication transaction.

6.2. FIDO2 key scopes and certificate name considerations

The scope of a FIDO2 key as set during the registration transaction determines the contexts in which it can be used. In EAP-FIDO, the following three notions interplay:

* the realm of username as used in the EAP-Identity exchange ("outer ID")
* the servername as presented during the EAP-TLS exchange by the EAP-FIDO server
* the relyingPartyIdentifier (rpId) that is used during the FIDO CTAP client authentication phase

Since FIDO keys may be used in other contexts, such as [WebAuthn], we have to ensure that the security parameters of EAP-FIDO and other FIDO-based protocols align.

The most important restriction regards the relation between domain name and Relying Party ID. This relation ensures that a FIDO credential is linked to a specific scope and cannot be used outside of that scope. Thus, in EAP-FIDO we must a equivalent dependency, that prevents the usage of a FIDO credential outside its intended scope, which could serve as basis for a cross-protocol attack.

Therefore, we require that the configuration item C_EXPECTED_SERVERNAME, which is used to verify the server certificate within the EAP-TLS handshake, is the same as or a subdomain of C_FIDO_RPID. This ensures that only an entity with a valid certificate within the scope the FIDO credential is registered to can also use the FIDO credential for EAP-FIDO.

This check MUST be done within the supplicant, and the server has to trust the supplicant’s implementation to actually check this.

6.3. EAP-Method with EAP-TLS vs standalone EAP method to be used in tunnels

Since there already exist EAP methods that provide a TLS tunnel and are capable of encapsulating further EAP methods, e.g. EAP-PEAP, EAP-TTLS or EAP-TEAP, the question arises, why this specification does not focus solely on the FIDO exchange as a standalone EAP method instead of re-specifying a new EAP-method that again makes use of EAP-TLS.

The main reason for a decision against this is the potential for misconfiguration. One of the goals for this EAP method is to provide a means to validate the server certificate using implicit configuration options. Using EAP-TTLS or EAP-PEAP would counteract this goal, since in most supplicants the configuration for the different phases of the tunneled TLS methods is done separately, so the users would have to configure the certificate check parameters manually again. Additionally, not every supplicant application may allow the code for the phase 2 exchange to access information about the phase 1 exchange, namely the server certificate parameters, which is necessary for the security of the EAP-FIDO exchange. Specifying EAP-FIDO as standalone EAP methods could therefore require modifying the EAP stack. Implementers might be tempted to re-use the insecure and error-prone configuration interfaces. To prevent this from the start, EAP-FIDO specifies an EAP-TLS based EAP method that cannot be used standalone.

Although this requires potentially duplicate code for supplicants that support multiple EAP-TLS based methods, the authors believe this means of specification to be more resistant against implementation errors and prevent error-prone user interfaces.

7. Implementation Status

Note to RFC editor: Remove this section, as well as the reference to [RFC7942] before publication

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

There is one early prototype proof-of-concept implementation of EAP-FIDO into hostap (hostapd for the server side, wpa_supplicant on the client side) available. The implementation was done before the specification of this draft was finished and is therefore not compatible with any draft version (different message format, simplified message flow, missing security checks), but serves as a proof-of-concept for the overall principle of using FIDO to perform an eduroam login. The source code can be found under https://git.rieckers.it/rieckers/hostap/-/tree/eap_fido_poc_tnc23 (https://git.rieckers.it/rieckers/hostap/-/tree/eap_fido_poc_tnc23)

8. Security Considerations

TODO Security

The security properties of FIDO are described in [FIDO-SecRef]

9. Deployment Considerations

This section will list considerations for deploying EAP-FIDO. It is intended to help local administrators to decide which parameters should be used in their deployment and give a brief overview over our suggested best practices.

9.1. Token Registration with User Presence vs. User Verification

In many use cases, it is desirable for a deployer to attribute an authentication transaction to a specific individual on an ongoing basis. If this ongoing attribution is important, tokens need to be registered in User Verification (UV) mode.

A token which is registered with User Presence (UP) only does not allow to ascertain the binding to a specific individual on an ongoing basis. The registration process makes sure that the token belongs to an authorized user initially at registration time - but this individual may transfer the token to other individuals post-registration, either by conscious decision or by accident. The new owner will be trivially able to complete an eventual UP challenge in the future, because UP challenges do not involve a personal authentication factor. Examples of such transfers include a physical hand-over of a USB Security Token, sharing the credential of a platform authenticator using AirDrop or theft of a device.

A token which is registered with User Verification (UV) on the contrary can be issued a UV challenge, which will require the personal authentication factor used during registration (e.g. PIN, biometric). While it may still be possible to transfer the token along with the authentication factor (say, USB Security Token and

associated PIN), this behavior is then equivalent to directly sharing the password in password-based EAP types or the private key of a Client Certificate for certificate based login. This has a higher psychological barrier, is a known problem, and can be sanctioned by the deployer in the same way as traditional password sharing is. Additionally, this prevents an attacker that came into possession of the FIDO token without consent/knowledge of the original owner from completing UV challenges, since they are missing the required verification credential (i.e. PIN or biometric).

We want to emphasize that preventing users from transferring ownership is only possible if the used FIDO keys require biometric authentication and this can not be circumvented (i.e. by a backup-PIN or a PIN-based mechanism to add new fingerprints).

10. IANA Considerations

This document has IANA actions:

* EAP type code point for EAP-FIDO

* EAP-FIDO registry

  - Message types, should probably be of policy "Specification Required"

  - Attributes, should be split with "Specification Required" and "Private use"

  - Error codes, should be split with "Specification Required" and "Private use"

  - Authentication requirements, with ints as "Specification Required" and text strings as "Private use" or "Experimental"

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC5216] Simon, D., Aboba, B., and R. Hurst, "The EAP-TLS Authentication Protocol", RFC 5216, DOI 10.17487/RFC5216, March 2008, <https://www.rfc-editor.org/rfc/rfc5216>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC9190] Preu Mattsson, J. and M. Sethi, "EAP-TLS 1.3: Using the Extensible Authentication Protocol with TLS 1.3", RFC 9190, DOI 10.17487/RFC9190, February 2022, <https://www.rfc-editor.org/rfc/rfc9190>.

[RFC9427] DeKok, A., "TLS-Based Extensible Authentication Protocol (EAP) Types for Use with TLS 1.3", RFC 9427, DOI 10.17487/RFC9427, June 2023, <https://www.rfc-editor.org/rfc/rfc9427>.

11.2. Informative References

[FIDO-CTAP2]
FIDO Alliance, "Client to Authenticator Protocol (CTAP)", 21 June 2022, <https://fidoalliance.org/specs/fido-v2.1-ps-20210615/fido-client-to-authenticator-protocol-v2.1-ps-errata-20220621.html>.

[FIDO-Glossary]
FIDO Alliance, "FIDO Technical Glossary", 23 May 2022, <https://fidoalliance.org/specs/common-specs/fido-glossary-v2.1-ps-20220523.html>.

[FIDO-SecRef]
FIDO Alliance, "FIDO Security Reference", 23 May 2022, <https://fidoalliance.org/specs/common-specs/fido-security-ref-v2.1-ps-20220523.html>.

[IETF115-emu-minutes]
IETF, "EMU @ IETF 115, Minutes", 7 November 2022, <https://datatracker.ietf.org/meeting/115/materials/minutes-115-emu-202211071530-00>.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <https://www.rfc-editor.org/rfc/rfc3748>.

[RFC5281] Funk, P. and S. Blake-Wilson, "Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)", RFC 5281, DOI 10.17487/RFC5281, August 2008, <https://www.rfc-editor.org/rfc/rfc5281>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/rfc/rfc7942>.

[WebAuthn] World Wide Web Consortium, "Web Authentication: An API for accessing Public Key Credentials Level 2", 8 April 2021, <https://www.w3.org/TR/2021/REC-webauthn-2-20210408/>.

Appendix A. Example use cases and protocol flows

A.1. Authentication using Discoverable Credentials with silent authentication

With this use case, the server will send an Authentication Request containing only the Relying Party Id attribute.

The client can trigger the silent authentication with the Discoverable Credential stored on the FIDO Authenticator and includes the response from the FIDO Authenticator into the Authentication Response message.

The server can look up the Credential IDs in its database, verify the FIDO signature with the stored public key and, if the signature was valid, send a protected success indicator to the client. The client responds with an acknowledgement and the server sends an EAP-Success

A.2. Authentication using Discoverable Credentials with silent auth for WiFi and uv auth for VPN

With this use case, the server will modify the Authentication Request based on which RADIUS client (the Wifi Controller or the VPN appliance) sent the request.

If the WiFi appliance sent the request, silent auth is used, and the flow is identical with the previous use case.

If the request came from the VPN appliance (e.g. because the VPN is done via IPSEC with EAP), the server adds "uv" in the Authentication Requirements attribute, which triggers an UV action on the client’s side.

The client triggers the authentication with the Discoverable Credential with user verification and responds to the server with the authentication data. The server verifies the signature, sends a success indication, the client acknowledges and the server sends an EAP-Success.

A.3. Authentication with Server-Side Credentials

In this use case, the FIDO Authenticator does not have a Discoverable Credential for the Relying Party ID. Instead, the server has a list of Credential IDs stored for each username.

After the initial Authentication Request from the server, the client does not have enough data to trigger the FIDO Authentication, so it needs additional information.

The client then sends an Information Request message with its username.

The server looks up the username in its database and sends back a list of Credential IDs it has stored for this user with an Information Response message.

The client can now trigger the FIDO authentication with this list and responds with an Authentication Response, that includes the Credential ID that was actually used for this FIDO authentication.

The server can now verify the signature and client and server finalize the EAP method.

A.4. Authentication with Server-Side Credentials and user-specific authentication policies

In this use case, different users have different authentication policies, i.e. employees are allowed to use silent authentication, but administrators need an authentication with user presence.

The server starts with an AuthenticationRequest and no authentication requirements or an empty array as authentication requirements. The client transmits its identity to the server in the Information Request message. Now the server looks up the user and their registered Credential IDs, and checks whether or not user presence verification is necessary.

If it is necessary, the server includes an authentication requirements attribute with the "up" value along with the PKIDs in the Information Response message. Since the client discards any previous attribute values, it now performs a FIDO authentication with user presence, and responds to the server.

A.5. Authentication with mandatory verification after a timespan

It may be desired to force a user verification after a timespan, to ensure that the FIDO token is still in possession of the user. This timespan is specific for each token, so the check whether or not the token is still allowed to perform only silent authentication can only be done after the authentication has happened.

Given, a user has two registered token, one was verified recently and the other exceeded the verification timespan.

The server start with an authentication request, the client answers with an Information Request and the Identity, the server transmits the Information Response with the two PKIDs and the client performs the silent FIDO authentication with the "expired" FIDO token. After sending the Authentication Response, the server verifies the FIDO authentication and recognizes the expired FIDO token. Now the server sends a new Authentication Request with the PKID of the expired token and "uv" as Authentication Requirement. The client now performs the FIDO authentication again, this time with user verification and sends the Authentication Response to the server. The server stores the timestamp of the successful user verification in its database and sends the success indicator.

A.6. Authentication with mandatory verification after a timespan with a grace period

As with the previous example, in this case the FIDO token again have a token-specific timeout to allow silent authentication for a period of time after a successful user verification. Unlike in the previous example, this time there is a grace period that still allows a silent authentication for a while after the timer expired. The intention is to not kick out the user in the moment the timer expires, i.e. a user is currently not in the vicinity of the device at that time, but one would not want to kick out the user if it needs to reconnect due to poor WiFi performance. Instead, the user may choose a convenient time to verify their identity/presence within this grace period.

The protocol flow is the same as the previous example, but this time the second Authentication Request from the server cannot be answered from the client, since the user is not performing the user verification. Instead, the client will send an Error message with error code TODO (FIDO authentication Timeout).

The server can now decide whether or not the silent authentication is still acceptable. If it is, meaning that the timeout for silent authentication has expired, but it is still in the grace period, it answers with a Success Indicator. If not, meaning that the grace period has expired too, the server will send a Failure Indicator with the appropriate error code.

A.7. 2FA-Authentication with client certificate on TLS layer and FIDO in the inner authentication

Since EAP-FIDO uses TLS, it is possible to perform two-factor authentication directly with only EAP-FIDO. In this case, the client and server perform mutual authentication at the TLS layer.

The server can now determine the identity of the user based on the certificate and already look up the stored Credential IDs. With this lookup, the server can already include the PKIDs in the Authentication Request. The client doesn't need to send an Information Request, since it already has all information. It can immediately perform the FIDO authentication process and send the Authentication Response to the server.

Appendix B. Open Questions regarding Protocol design

Note to RFC Editor: Remove this section and all references from this section before publication.

Since this specification is an early draft, there are a lot of open questions that we want to get community feedback on.

B.1. How to determine the FIDO Relying Party ID?

FIDO needs a relying party ID to function. The question is how this RPID is determined and verified, there are several options that all have pros and cons.

The main thing is to have in mind, that there are three relevant parameters, that need to be put into a certain relationship:

* the RADIUS realm (i.e. 'anonymous@dfn.de')
* the RPID (i.e. 'eduroam.dfn.de')
* the Server Certificate Name (usually subjectAltName:DNS, i.e. 'radius.eduroam.dfn.de', in the following abbreviated simply with SAN)

All these three parameters need to be in a pre-defined relationship to allow a simple and hard-to-mess-up-and-still-secure configuration. Both the client and the server have to agree on what the RPID is, in order for the FIDO authentication to succeed. If there is a defined relationship between the RPID and the certificate name (i.e. SAN needs to be a subrealm of the RPID), then the client needs to verify the certificate against exactly that. When does the client do that? What security implications does that bring? All these options need some thought.

B.1.1. Option 1: Configuration

The first option would be to just have the RPID as a configuration item, maybe with a default on the realm of the outer username. Adding a configuration option complicates the setup of the EAP method, but hopefully not too much. A misconfiguration of the RPID is also not that critical from a security standpoint. The effects of a misconfigured RPID are only a problem if the used FIDO key is also registered with a third party, in which case the third party could trick the client to connect to a bogus network.

If the RPID deviates from the realm, the client could send the requested RPID using Server Name Indication.

B.1.2. Option 2: Mandate RPID to equal Realm of the Username

The second option would be to mandate that the RPID is equal to the realm portion of the username. This restricts options on how to use EAP-FIDO and may cause unnecessary difficulties in routing, if the convenient routing domain (e.g. the registered domain for a company) should not be used as RPID due to security concerns, or different RPIDs should be used under the same routing realm.

B.1.3. Option 3: RPID is determined by the server and sent before the TLS handshake

Since the RPID plays an important role in the decision whether or not the certificate sent by the server is to be trusted, the RPID should be determined before the TLS handshake. The server could determine the RPID based on the outer username and send it as payload in the EAP-TLS Start packet. This way, the client has a clear indication as to whether or not to trust the server certificate sent in the subsequent TLS handshake.

However, this opens up some security issues that are yet to be investigated, since the RPID could be modified by an on-path attacker.

B.1.4. Option 4: RPID is determined by the server and sent after the TLS handshake

With this option, the problem is that the client needs to cache the server certificate in order to determine if the RPID is valid. for the given certificate, unless the rules for certificate verification and RPID determination specify it otherwise. One possibility to circumvent this would be to allow the server certificate names and the RPID to deviate, but validate both against the realm of the outer username, e.g. a realm of example.com with a server certificate for radius.example.com and the FIDO RPID fido.example.com.

This, however, adds a whole lot more of security concerns, especially in environments with different independent divisions under the same domain suffix.

B.1.5. CURRENT DECISION: Option 1

With draft-janfred-eap-fido-02 we introduced wording that the Relying Party ID is the origin of all configuration items. All other configuration items on the client side are derived from that. This also means that we have a default DNS name that we check the server certificate against: eap-fido-authentication.<RPID>. The name is not final yet and should not be hardcoded in any code other than proof-of-concept code. That it is ok to just define a specific subdomain and mandate it was picked from RFC 8461 (MTA-STS), where mta-sts.<mail domain part> is mandated as host that will serve the MTA-STS file.

B.2. Missing Features

There are a lot of features, that have been brought up by several people at several occasions. This EAP method could include spec to address these problems. Also there may be FIDO-specific things that are not part of this specification.

B.2.1. Deprovisioning of EAP configuration

It may be useful to directly include a way to signal in-band that an EAP configuration should be deleted.

The idea stems from the discussion at IETF 115 about EAP-DIE [IETF115-emu-minutes].

With EAP-FIDO it may be desirable to also allow for deletion of Discoverable Credentials. (Maybe residential Server-Side Credentials too?

Input on the need and specific way to achieve this is welcome.

B.3. Open questions regarding security

B.3.1. Multiple signatures with the same parameters a problem?

The current specification allows the server to send additional data for the FIDO client-data-hash, but if the server omits that, the client-data-hash is only comprised of the exported key material from TLS. Now the interesting question is: Is this a problem? Could a malicious server, that came in possession of a certificate that the client trusts, perform multiple authentication runs and observe the client's behavior and analyze the different signatures? Basically, this could be a chosen-plaintext attack, where the attacker could control at least some portions of the plaintext. One way around this would be to include a nonce in the TLS-Exporter for the FIDO challenge, so the client selects this nonce and sends it to the server, this way the server can not predict the plaintext used for signing. The possibility of a double-signature with the same data is prevented by the counter increasing with every signature, but maybe some authenticators do not implement this counter. The research whether or not this is a problem is still TODO, if FIDO experts have an opinion about this, please contact the authors.

Appendix C. Document Status

Note to RFC Editor: Remove this section before publication.

C.1. Change History

draft-janfred-eap-fido-00:

- Initial draft version

draft-janfred-eap-fido-01:

- Updated terminology to align with the FIDO/WebAuthn terminology
- Reword introduction

draft-janfred-eap-fido-02:

- Rewording of introduction - better description under which circumstances the current practice is flawed.
- Add Section describing client and server configuration
- Add eap-fido-authentication.<RPID> as default server identity.

- Refine wording around server name verification

- Adjust message formats and attribute mapkeys to not transmit the RPID any more.

- Add first few paragraphs on error handling

draft-ietf-emu-eap-fido-00:

- First WG draft

- Update way FIDO client data is constructed (include protocol binding at the very beginning, before exported key material from TLS)

- Change auth requirements attribute to array of ints or text string, with text strings used for experimental features

- Update IANA section for registry of auth requirement ints

C.2. Missing Specs

* Error codes and Error handling

* Key derivation for i.e. WPA2

  - Will be exported from TLS layer, maybe include some information from the FIDO exchange to bind it to the FIDO exchange?

Acknowledgements

The document authors want to thank Alexander Clouter for the idea of a default authentication server name.

Authors' Addresses

Jan-Frederik Rieckers
Deutsches Forschungsnetz | German National Research and Education Network
Alexanderplatz 1
10178 Berlin
Germany
Email: rieckers@dfn.de
URI: www.dfn.de

Stefan Winter
Fondation Restena | Restena Foundation
2, avenue de l’Universit
L-4365 Esch-sur-Alzette
Luxembourg
Email: stefan.winter@restena.lu
URI: www.restena.lu