Cryptographic Binding Should Not Be Optional: A Formal-Methods Analysis of FIDO UAF Authentication

Enis Golaszewski[1], Alan T. Sherman[1], and Edward Zieglar[2]

[1] Cyber Defense Lab, University of Maryland, Baltimore County (UMBC), 1000 Hilltop Circle, Baltimore MD 21228, USA
golaszewski, sherman@umbc.edu
[2] National Security Agency, 9800 Savage Road, 20755 Fort George G. Meade, USA
evziegl@uwe.nsa.gov

Abstract. As a case study in cryptographic binding, we present a formal-methods analysis of the *Fast IDentity Online (FIDO) Universal Authentication Framework (UAF)* authentication protocol's cryptographic channel binding mechanisms. First, we show that UAF's channel bindings fail to mitigate protocol interaction by a *Dolev-Yao (DY)* adversary, enabling the adversary to transfer the server's authentication challenge to alternate sessions of the protocol. As a result, in some contexts, the adversary can masquerade as a client and establish an authenticated session with a server, which might be a bank server. Second, we implement a proof-of-concept man-in-the-middle attack against eBay's open source FIDO UAF implementation. Third, we propose and verify an improvement of UAF channel binding that better resists protocol interaction, in which the client and the server, rather than the client alone, bind the server's challenge to the session. The weakness we analyze is similar to the vulnerability discovered in the Needham-Schroeder protocol over 25 years ago. That this vulnerability appears in FIDO UAF highlights the strong need for protocol designers to bind messages properly and to analyze their designs with formal-methods tools. Our case study illustrates the importance of cryptographically binding context to protocol messages to prevent an adversary from misusing messages out of context.

Keywords: Authentication · Cryptographic binding · Cryptographic protocols · Cryptographic Protocol Shapes Analyzer (CPSA) · Cryptography · Cybersecurity · Fast Identity Online (FIDO) · Formal-methods analysis of protocols · Universal Authentication Framework (UAF).

1 Introduction

42 years apart, the 1978 *Needham-Schroeder (NS)* public-key protocol [34] and the 2020 *Fast IDentity Online (FIDO) Universal Authentication Framework (UAF)* 1.2 specification [5] share a common flaw: they both fail to cryptographically bind a sensitive value to its source. To protect against a *Dolev-Yao*

(DY) network intruder [16], appropriate cryptographic binding is not optional. A DY intruder can potentially exploit inadequate binding to discover protocol interactions and launch *man-in-the-middle (MitM)* attacks, which may compromise authentication, integrity, or other security goals. Protocol interactions are widespread and often challenging to discover without the aid of formal-methods protocol analysis tools. Due diligence requires a formal-methods analysis of cryptographic binding in emerging standards to identify and mitigate harmful protocol interaction.

We present a formal-methods analysis of UAF authentication as an instructive case study on cryptographic binding, emphasizing the protocol's optional and deficient binding of an important cryptographic challenge. Using the *Cryptographic Protocol Shapes Analyzer (CPSA)* [28], we enumerate all essentially different protocol interactions as "shapes," comprising message sequence charts that reflect the flow of messages across a network under the control of a DY network intruder. To our knowledge, our work is the first to perform a formal-methods analysis of UAF's channel bindings.

The standard refers to "*channel binding*," but this phrase is a misnomer because UAF binds to the endpoints of the channel and not to the session. This inadequate binding creates a vulnerability in which an adversary can transplant messages among different instances of the protocol between the same endpoints. For simplicity, we will nevertheless continue to use this phrase.

Our analysis reveals a MitM attack on UAF authentication that enables an adversary to masquerade as a client, for example, when establishing a session with a bank server. This attack results from the following weaknesses in UAF: (1) Even when performing channel binding, the server does not cryptographically bind the challenge adequately, preventing the client from authenticating the challenge. (2) The standard makes channel bindings optional, creating circumstances in which there is no cryptographic binding between the client's attestation and the server's protocol session. (3) The server selectively accepts incorrect channel bindings, potentially accepting attestations from malicious protocol sessions. Exploiting these weaknesses, an adversary can trick a legitimate client into acting as a confused deputy [21], producing attestations for a legitimate server's challenge that the legitimate server accepts. To our knowledge, we are first to provide details of a MitM attack against UAF authentication that exploits these weaknesses, and first to identify the structural weakness that results as a consequence of binding the challenge inadequately.

As a proof of concept, we implement and demonstrate our MitM attack against eBay's open-source UAF server [17], which does not implement channel binding (see Section 10). To carry out this attack, an adversary exploits inadequate binding of the server's challenge to specific sessions of the UAF authentication protocol, producing a protocol interaction in which the honest client-authenticator pair generates attestations for the adversary's malicious sessions. This attack demonstrates an example of a harmful protocol interaction resulting from inadequate cryptographic binding.

UAF is an attractive emerging standard and the subject of a growing number of studies, including formal-methods studies (see Section 6). The FIDO Alliance counts among its members many recognizable technology giants, financial institutions, retailers, government institutes, and standards bodies from nations all over the world [2], including Google, Apple, Microsoft, and Amazon, coming together in broad push to eliminate traditional password-based authentication. As of April 2023, there are 472 FIDO-certified deployed implementations of UAF [3]. With UAF replacing many traditional password-based systems, it is vital that we analyze and improve the standard's deficient cryptographic bindings.

Vulnerabilities we identify in UAF authentication highlight two serious deficiencies in how the FIDO Alliance developed the standard: (1) The FIDO Alliance did not systematically adopt an appropriate adversarial model, but instead considered only separate adversarial capabilities in response to various ad hoc threats, and (2) the FIDO Alliance did not perform formal-methods analyses of their proposals. Two vital lessons for protocol designers from our case study are not to repeat these errors.

UAF and studies of it reflect a broader failure to consider cryptographic binding carefully. In this paper we identify failure of UAF to bind the server's challenge properly. Existing studies of UAF incorporate UAF's channel bindings inconsistently and fail to consider weaknesses of the standard's available channel bindings. UAF attempts to mitigate protocol interaction by incorporating an optional channel binding from one of several existing and draft TLS channel-binding standards. To analyze these bindings, we specify several formal models of UAF authentication, omitting or including variations of channel binding, and interpret the resulting CPSA shapes to identify potentially harmful protocol interactions.

We present our analysis by (1) explaining a major structural flaw in FIDO UAF and giving three examples of resulting possible attacks, (2) conceptually introducing cryptographic binding and related background, (3) introducing the FIDO UAF protocol and its channel binding mechanisms, (4) modeling variations of the protocol in CPSA, (5) analyzing the resulting shapes for significant protocol interactions, (6) implementing an attack on UAF channel binding, and (7) recommending improvements to UAF's channel binding.

Our contributions include: (1) a formal-methods analysis of UAF 1.2 authentication's optional channel bindings, (2) a structural weakness and resulting attack against FIDO UAF authentication with channel binding, exploiting inadequate binding of the server's challenge, (3) details of a MitM attack in which an adversary, exploiting inadequate binding of an honest server's challenge, tricks a client and authenticator pair to act as confused deputies to authenticate the adversary, (4) a demonstration of the attack against eBay's open-source UAF server, and (5) an improvement to the UAF standard in which both the client and server bind the challenge. We include artifacts of our work, including all of our CPSA and attack sourcecode (available on GitHub [26]) and selected CPSA sourcecode (Appendix B).

2 A Structural Weakness of FIDO UAF

FIDO UAF has a fundamental structural flaw: with, or without channel binding, the client cannot verify if the server-generated challenge originated from the current session between the client and server. This flaw results from inadequate binding of the challenge to the session context.

As a result, in certain situations, a DY adversary is able to carry significant malicious actions by manipulating and transplanting parts of messages (e.g., containing the challenge) between different protocol sessions. We summarize three examples.

Example 1. when channel binding is not used, the adversary masquerades as a legitimate bank server. When the client initiates a session with the adversary, the adversary launches a parallel session with the legitimate server. The adversary obtains a challenge from the legitimate server and sends it to the client. Unable to verify the context of the challenge, the client returns a signed attestation of the challenge to the adversary, who passes it along to the server, thereby authenticating to the server as the client. In Section 10 we implement this MitM attack against eBay's open-source FIDO UAF server.

Example 2. Many organizations install a perimeter TLS proxy for the purpose of monitoring traffic flows across their perimeters [35]. To support this practice, when channel binding is used, FIDO UAF permits the server to accept channel bindings to the proxy rather than to the server. In the DY model, however, the proxy might be malicious. Because the client cannot verify the source of the challenge, the client cannot distinguish whether the challenge is from a session with the server, a legitimate proxy, or a malicious proxy. In this sense, FIDO UAF supports an adversary to carry out a MitM attack between the client and the server.

Example 3. When channel binding is used with or without perimeter TLS proxies, a potential subtle vulnerability arises when the client carries out multiple sessions with the same server. Because "channel binding" in FIDO UAF binds only to the endpoints of a channel, and not to the session, potential threats arise in which the adversary manipulates messages among the multiple sessions. For example, suppose a client establishes two concurrent sessions with an investment bank for the purpose of making a stock transaction in each session. Although the server generates a different challenge for each session, the adversary might be able to manipulate the challenges, and the client's signed attestations of them, to change the order of the transactions. Changing the order of transactions can have significant consequences. Carrying out this attack would require dealing with other complexities, including the authenticator's signature counter, if the authenticator has a signature counter. Even if the authenticator has a signature counter, it might be possible for the adversary to send the client a policy that states that the adversary will accept only authenticators that do not use signature counters.

Another serious issue with FIDO UAF is that it fails to articulate a clear and well-defined security goal. The FIDO UAF specification [5, p.4–5] vaguely states: "The goal of this Universal Authentication Framework is to provide a

unified and extensible authentication mechanism that supplants passwords while avoiding the shortcomings of current alternative authentication approaches."

What FIDO UAF does is for the server to send a challenge to the client, the client to return a signed attestation of this challenge, and the server to verify signature and the contents of the attestation, including the channel binding if present. These actions, however, have serious limitations. The client cannot verify from what session the challenge originates; the client can choose not to use channel binding; the server may accept incorrect bindings; and channel binding binds only the endpoints of the channel and not the session. These limitations exist whether the authenticator is based on passwords or biometrics.

Although we point out vulnerabilities of and attacks against FIDO UAF, this paper is not intended narrowly as an analysis of FIDO UAF. Instead, our work is a case study in the crucial concept of cryptographic bindings in protocols. The vulnerabilities and resulting attacks we uncover stem from deficiencies in cryptographic bindings. These binding issues in FIDO UAF are the tip of a much larger serious pervasive problem that plagues many modern protocols.

Furthermore, our case study reveals disturbing underlying causes that gave rise to the binding issues: the FIDO UAF specification does not clearly and precisely state its security objectives; the specification does not define and follow an appropriate consistent adversarial model, such as the DY network adversary model; and the designers did not perform formal-methods analyses. This confused thinking, in combination with a desire to encourage adoption through permissive policies, resulted in a complex protocol for which its security implications are difficult to analyze, a protocol that is difficult to correct, and a protocol that is vulnerable to attack. Using CPSA, we provide a formal-methods analysis of FIDO UAF with and without channel binding.

3 Cryptographic Binding

In 1996, Abadi and Needham [1] presented informal guidelines for designing sound cryptographic protocols, including the need for explicit *context* and *cryptographic binding*. Cryptographic binding associates sensitive data with a specific context, complicating the malicious act of transplanting data from one protocol context to another. Digital certificates are a well-known example of binding: the certificate associates an entity's identifier with the entity's public key, bound by the digital signature of a trusted issuer.

Network protocols that fail to bind messages to a context are vulnerable to *protocol interaction* [25], in which an adversary uses messages between different protocols, or different sessions of the same protocol, to produce undesirable outcomes. Often, an adversary will produce interactions between two sessions of the same protocol, as in a man-in-the-middle attack. For example, Gavin Lowe's [30] 1995 famous attack on the NS public-key protocol [34] exploits a lack of binding between a random nonce and its owner, enabling an adversary to misrepresent another communicant's nonce as their own. A protocol that fails to bind messages may also permit an adversary to transplant data or entire messages to

attack a separate protocol. Protocol interaction can be mitigated by binding cryptographic values to a specific protocol session.

4 Background

We now present brief background for CPSA and formal-methods analysis of protocols, which we use in subsequent sections to perform a formal-methods analysis of FIDO UAF authentication.

4.1 Formal Methods for Protocol Analysis

Current tools for formal-methods analysis of cryptographic protocols include ProVerif [9], Tamarin Prover [32], Maude-NPA [18], and CPSA [28]. These tools build on ideas of legacy tools, including NPA [31], Interrogator [33], and Scyther [13], and efforts such as Lowe's analysis of NS using FDR, a refinement checker for formal *Communicating Sequential Processes* models [30].

ProVerif proves properties of Horn clauses modeling a protocol. Tamarin Prover proves properties about multiset rewriting rules. Maude-NPA searches backwards from attack states through unification. By contrast, CPSA searches for protocol interactions, and upon termination, provably enumerates all essentially different protocol interactions for a protocol model [15, 27].

It is possible to carry out inductive security analysis of protocols using high-order logic theorem provers such as Isabelle [37]. This approach uses general interactive theorem provers, rather than specialized tools for protocol analysis.

We chose CPSA for our analysis because the tool is ideal for discovering protocol interactions, including those resulting from inadequate binding, and we are familiar with CPSA. Because CPSA enumerates all essentially different possible protocol executions, CPSA will find all essentially different protocol interactions possible given the input models. To our knowledge, no other tool has this characteristic. By contrast, many other tools (e.g., Tamarin) require the user to state properties to be verified, creating a risk that a protocol interaction might be overlooked because the user did not explicitly state an important property.

4.2 CPSA

CPSA is an open-source tool that analyzes cryptographic network protocols for protocol interactions. In contrast to many formal-methods tools, CPSA is not a theorem-prover but a model-finder. For an input model, which comprises roles, messages, variables, and assumptions regarding those variables, CPSA outputs trees of graphical *shapes* that illustrate possible protocol executions. When CPSA terminates, it provably discovers all essentially different shapes for the input model, enabling users to inspect these shapes and identify all protocol interactions possible for the input model. Additionally, a model may specify goals, not unlike theorems, which each of the output shapes must satisfy.

Users define CPSA models using LISP-like s-expressions that implement a custom language. In these models, which superficially resemble (but are not) executable source code, users specify one or more roles, associated variables and messages, and *skeletons*. Skeletons specify one or more initial roles and impose assumptions on the role variables, such as forcing a value to originate uniquely each session or being unavailable to the adversary. When CPSA executes, it attempts to satisfy skeletons into shapes by repeatedly applying actions available to a DY intruder in *strand space theory* [19].

Shapes, which CPSA represents graphically, consist of *strands* that each represent a legitimate protocol role or an adversary's listener for key values. Each strand consists of sequential *nodes* that specify message transmission or reception events. Connecting these nodes between different strands are two types of arrows: solid arrows indicate a pair of message events, transmission and reception, for which CPSA can prove a casual relationship. Dashed arrows indicate a pair of message events for which CPSA, acting as a DY adversary, manipulates available information to satisfy each event. When analyzing CPSA shapes, users take special note of dashed arrows as these often suggest undesirable protocol interactions.

To analyze protocols using CPSA, users often follow a common workflow: (1) extract messages and variables from a protocol specification and define these in a model as roles, (2) specify security assumptions for key variables that each role creates, or originates—often, these assumptions are specific to certain protocol perspectives and ultimately reside in skeletons, (3) specify skeletons for different role perspectives or special scenarios, (4) execute CPSA to produce output shapes, (5) manually analyze the output shapes to identify protocol interactions. In this paper, we follow this workflow to analyze FIDO UAF.

5 FIDO

We now introduce the FIDO UAF protocol and discuss the protocol's cryptographic channel binding. As we will show in Section 9, the specification has a structural weakness resulting from the server's inadequate binding of the challenge. Additional vulnerabilities result because channel binding is optional and the server may accept incorrect channel bindings.

5.1 FIDO UAF

In 2013, the FIDO Alliance proposed the UAF [5], an open standard forgoing passwords in favor of devices (e.g., cell phones) with local authentication mechanisms (e.g., biometric, PIN). There are many protocols and variations thereof within the UAF standard described in at least 11 documents. Initially, a client registers one or more of these devices, known as *authenticators*, with a server. To authenticate itself to a server, the client (1) receives a unique random challenge from the server, (2) for each authenticator, forwards a hash of the challenge parameters and satisfies each of their local authentication mechanisms, (3) receives

a signed attestation from each authenticator, and (4) sends the attestations to the server. Using knowledge of the authenticators and their public keys, the server verifies the signatures and authenticates the client.

UAF's design seeks to reduce the use of traditional passwords, which FIDO indicates as problematic due to common issues: weak password choices, password reuse, management of many passwords, and phishing attacks. FIDO identifies passwords as responsible for over 80% of all data breaches, and promotes UAF as a potential mitigation. To encourage adoption of UAF and maximize compatibility with existing systems, FIDO makes several features optional, most notably the protocol's cryptographic channel binding.

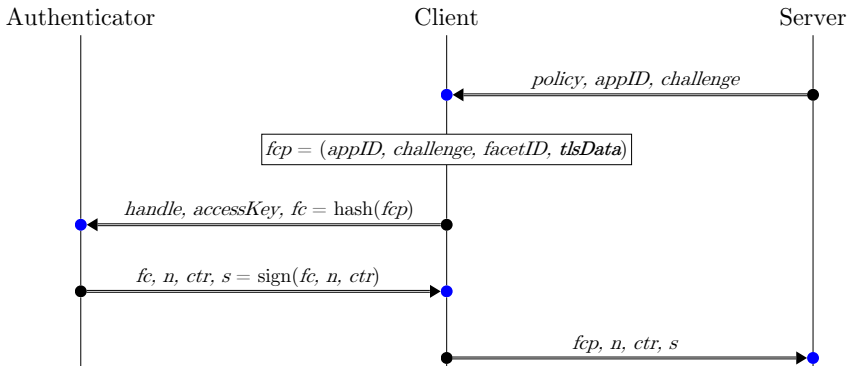

Fig. 1. Idealized message sequence diagram of UAF authentication cryptographic flow featuring a single authenticator. Vertical lines correspond to protocol roles; arrows indicate sending or receiving a message; and arrow labels specify message content. The boxed note on the client role specifies how the client generates *fcp* (final challenge parameters), which it sends to the authenticator. When generating *fcp*, the client has the option of including zero or more channel bindings in the tlsData.

Figure 1 illustrates a simplified example of UAF authentication with a single authenticator based on the cryptographic data flow in the UAF specification. The server initiates the protocol by generating a unique, random *challenge*, which it sends to the client together with a *policy* dictionary, specifying a list of acceptable authenticators, and the service's *application identifier (appID)*. Upon receiving the challenge, the client generates *final challenge parameters (fcp)* that comprise four values: (1) the *appID*, (2) the challenge, (3) the *facetID*, which is a URL of the request triggering the protocol, and (4) an optional *Transport Layer Security (TLS)* channel binding [8, 39, 4]. The client then sends a hash of the *fcp* to the authenticator, including a lookup handle and access key corresponding to the server. Using the handle, the authenticator verifies the access key, increments a counter, generates a nonce, and signs a concatenation of the nonce, counter, and the hash of *fcp* to produce an attestation *s*. From the authenticator, the client receives the nonce, counter, and *s*, and sends these values to the server, including

the original *fcp*, which the server verifies to authenticate the client and complete the protocol.

5.2 Channel Binding

UAF specifies four channel binding mechanisms: (1) server endpoint, (2) server certificate, (3) channel ID, and (4) token binding. In (1) and (2), the client binds the challenge to a hash of the server's certificate and to the server's certificate, respectively. In (3) and (4), the client binds the challenge to a public key corresponding to a private key the client holds—through the use of extensions, this public key associates itself with the TLS channel between the client and the server.

The FIDO Alliance purposefully specifies these bindings as an optional feature of UAF, citing inadequate support for channel bindings and challenges facing perimeter proxies in 2018 [7]: "... the addition of channel-binding information in FIDO assertions is optional: not all client platforms support Channel ID or Token Binding, and even if a client has all the necessary support for channel-binding, it might make sense not to enforce channel-binding."

Notably, even when the server requires channel binding, the specification [5, p. 52] allows for a circumstance in which the server has discretion whether to accept a certificate-based channel binding from the client (e.g., when the client communicates to the server through a perimeter proxy, as some organizations deploy to monitor communications passing through their firewall). This policy creates a potential vulnerability and single point of failure (only the server verifies the client's channel binding), since the server might accept a channel binding to the adversary's certificate. Because the client depends on the server to verify the channel binding, and cannot verify the origin of a challenge, the client might be tricked into generating a channel binding for an adversary. Our improved binding enhances the client's assurance of the challenge's legitimacy by enabling the client to authenticate the server's challenge (see Section 11).

The specification hints that a MitM attack may be possible when omitting channel bindings [7]: "... to deny man-in-the-middle attackers the ability to re-play credentials obtained from eavesdropping on their victims, sites on the internet should use FIDO ... and channel-binding techniques such as Token Binding (to eliminate the risk of stolen cookies)." We confirm the presence of such an attack in Section 9 and implement it against an open-source server in Section 10.

In their threat analysis [6], the FIDO Alliance does not carefully define an adversarial model and does not consider a DY adversary. Instead, they informally consider various adversarial capabilities separately for each threat, including MitM. As a result, they overstate the protections provided by their channel bindings. The protections do not enable a client to verify a challenge in a DY network (in which the client may communicate with a malicious server), allowing the adversary to invoke the client as a confused deputy. In the DY model, it is a realistic threat that the adversary operates a legitimate server. Additionally, FIDO's threat model does not address the circumstance in which a server accepts

a channel binding from a client's TLS session with a separate entity (e.g., a proxy).

6 Previous Work

In contrast with our work, existing formal-methods studies of the FIDO UAF protocol do not explicitly address cryptographic binding, and channel binding in particular, possibly because channel binding is optional.

In 2016, Feng et al. [24] presented a formal-methods analysis of the UAF registration and authentication protocols using ProVerif. Formalizing and analyzing the standard's security goals, they describe four attacks: (1) an authenticator rebinding attack, including implementations against real finance applications, (2) a parallel session attack, (3) a privacy disclosure attack, and (4) a denial-of-service attack. Their work grants additional capabilities to the adversary beyond DY and does not address security implications of the standard's optional binding, nor the weaknesses of the standard's specific binding mechanisms and client-side binding strategy.

Pereira et al. [38] also present a formal-methods analysis of UAF using ProVerif, concluding that if the client correctly verifies the *appId* it receives from the server, the protocol resists a DY-style network adversary. Notably, their analysis completely omits channel binding, focusing exclusively on the *appId*'s role in helping the client authenticate the server's challenge, and they do not present any analysis of UAF with channel binding. We consider the *appId* a poor value to bind because it is a public, not a cryptographic, value, and it is not unique to a specific protocol session.

Hu and Zhang [24] present an informal analysis of the UAF protocols, identifying three attacks: (1) a misbinding attack, (2) a parallel session attack, and (3) a multi-user attack. These attacks do not consider the impact of channel binding and require a stronger adversarial model than DY: the adversary corrupts the client, corrupts the authenticator, or exploits multiple users sharing an authenticator.

Panos et al. [36] perform an informal analysis of UAF, presenting several high-level attack vectors. Büttner and Gruschka [10] present MitM attacks on FIDO extensions, design a protocol to protect the extensions, and analyze the protocol using ProVerif. Neither of these works considers channel binding.

Additional studies of FIDO UAF assess social-engineering attacks [40], biometric authenticators [12], informal trust requirements [29], and feasibility [11].

Building on our initial work, Fuchs et al. [20] perform a formal-methods analysis of FIDO UAF registration.

7 Adversarial Model

We analyze the FIDO UAF authentication protocol in the DY adversarial model [16], a well known formal model which CPSA incorporates. DY specifies a network

intruder that carries all messages on the network, capable of arbitrarily manipulating messages across any number of protocol instances. This network intruder cannot perform any cryptanalysis, but will apply knowledge of secret keys to encrypt and decrypt messages and is capable of generating new keys. A DY adversary hunts for protocol interaction, exploiting the lack of cryptographic binding to exploit structural flaws in protocols.

In our analysis, we assume that the adversary is unable to compromise the private keys of any legitimate entities, including the authenticator and TLS *certificate authority (CA)*. As a result, the adversary is unable to sign their own attestations or forge valid TLS certificates for honest entities. However, we assume the adversary is in possession of their own valid certificate and capable of appearing as a legitimate client or server, consistent with the DY model. Additionally, we assume that the adversary is unable to manipulate messages on the secure channel between the authenticator and the client.

8 CPSA Model

As detailed in Appendix B and GitHub [26], we specify protocol roles, including messages and variables, and assumptions to model five variations of UAF authentication's cryptographic message flow in CPSA: (1) unbound, (2) server endpoint binding, (3) server certificate binding, (4) channel ID binding, and (5) token binding. Each model derives from a common base model, which we illustrate in Figure 2. To authenticate the server to the client and establish an encrypted channel, the base model incorporates an existing model of TLS 1.2.

Our model specifies three roles: *client*, *server*, and *authenticator*. Prior to the start of the protocol, the client and the server complete an instance of TLS [14]. We encrypt communication between the client and the authenticator using a *long-term key*, modeling a secure channel. The server initiates the protocol, sending to the client a freshly generated challenge that it encrypts using the TLS *serverWriteKey*. The client receives this challenge, computes the final challenge $fc = hash(challenge, tlsData)$, where *tlsData* contains an optional channel binding, and transmits it to the authenticator. The authenticator now signs fc using its private key and sends the resulting attestation $(fc, s = sign(fc))$ to the client. Finally, the client transmits the attestation (fc, s) to the server, encrypting it using the TLS *clientWriteKey*.

For simplicity, we omit the *appID*, *facetID*, *policy*, and *handle* from our messages. The *appID* identifies the server endpoint. The *facetID* identifies the address of the web page or application triggering the FIDO UAF protocol. The *policy* specifies a list of authenticators that the server accepts. The *handle* notifies the authenticator which key pair to use for the session.

We assume the adversary has knowledge of the *appID*, *facetID*, and *policy* because the honest server shares these values in any session. Because we model the protocol with one authenticator that registers with a single server, it is unnecessary to model the *handle*, which enables an authenticator to identify the private key with which to sign its attestation.

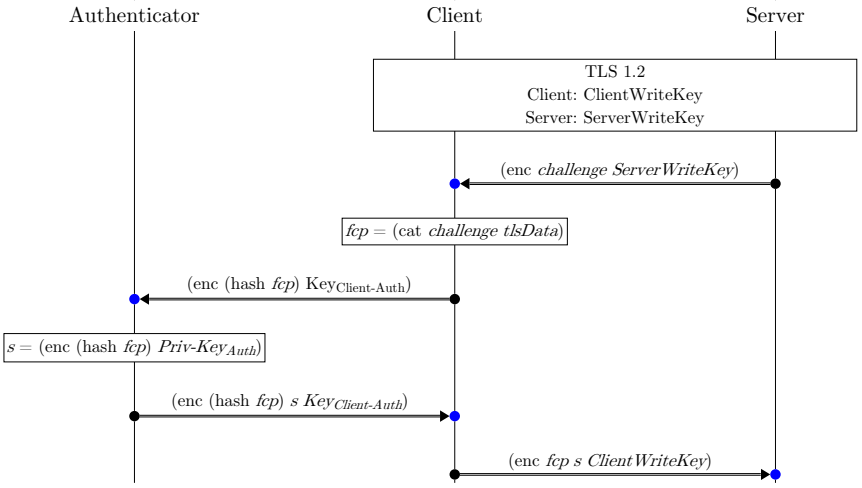

Fig. 2. Base CPSA model for FIDO UAF authentication that represents messages as *s-expressions* consistent with CPSA's modeling language. This model incorporates an existing CPSA model of TLS 1.2 to negotiate the server and client write keys. The client incorporates channel bindings into the tuple *tlsData*, if available. The client and authenticator encrypt using a long-term, symmetric key unavailable to the adversary.

We also omit the nonce n and the signature counter *ctr* from the authenticator's attestation. If the authenticator does not support a signature counter, the authenticator sets the counter to zero. The standard specifies these values for mitigating authenticator cloning, in which the adversary steals the authenticator's private key to sign their own attestations, and attestation replay attacks, in which the adversary replays old attestations. Our adversarial model does not permit the adversary to compromise the authenticator's private key and assumes a secure channel between the client and the authenticator. We assume that the server will never generate the same challenge twice, thereby enabling it to detect any replayed attestation from the authenticator. In light of these assumptions, omitting the nonce and signature counter does not permit any attacks within our scope. Additionally, the specification provides insufficient details about the many difficult issues, including synchronization, that can arise with counters.

9 CPSA Analysis

We now analyze significant CPSA shapes resulting from our models, evaluating each of UAF's four channel bindings and identifying a potential attack when omitting channel binding. Our analysis considers three honest viewpoints (client, server, authenticator) and discusses channel binding from each of these perspectives. It is essential to consider these different perspectives because each role is restricted in what it can verify or deduce based the cryptographic values it

knows. For example, the server does not know whether the client's TLS pre-master secret is fresh.

9.1 Server's Perspective

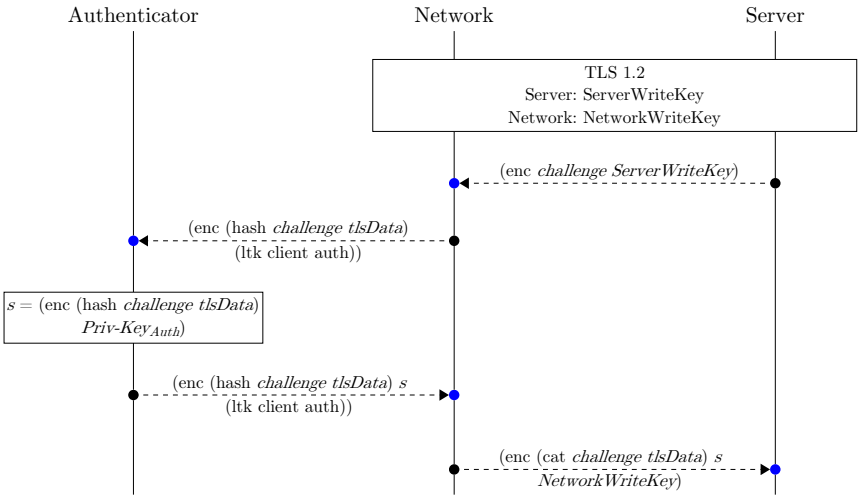

Fig. 3. CPSA shape capturing the server's perspective, including an optional channel binding as part of *tlsData*. The server does not authenticate the client and does not know the status of the client-authenticator access key, and subsequently receives an attestation from an unknown session involving the honest authenticator via the network (adversary). When channel binding is present, the server can verify the binding present in the attestation and terminate the protocol if this binding specifies incorrect endpoints. Even when binding is present, the server cannot identify to which specific protocol session the attestation corresponds.

For each variation of channel binding, from the server's perspective, CPSA produces a single shape (see Figure 3). In this shape, the protocol completes without involving an honest client, suggesting two potential issues: (1) due to the limitations of most common applications of TLS, the server does not authenticate the client and does not know with whom it is communicating, and (2) in the absence of channel binding, the server does not know if an attestation results from its UAF authentication session or protocol interaction with a potentially malicious session.

Because the server does not authenticate the client when establishing the TLS channel, and does not provide the client with any means to authenticate the challenge, we are unable to verify that the authenticator's assertion originates from the legitimate server's session. In subsequent perspectives, we observe that the client and authenticator must act as a pair to attest a challenge. Without

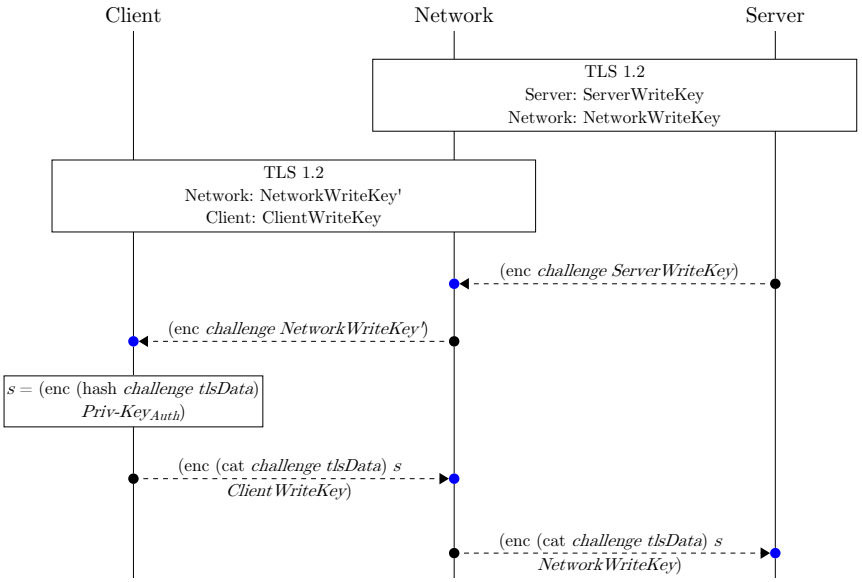

Fig. 4. CPSA shape capturing the server's perspective, where for clarity, we integrate the client and authenticator into one client role. Whether or not the client includes channel bindings in the *tlsData*, the client acts as a confused deputy and produces an attestation for the adversary. This shape reveals the following structural weakness: in three scenarios, the adversary can masquerade as the client to the server, or transplant messages from one session to another between the same endpoints. In particular, the adversary can masquerade as the client when the client omits channel binding in the *tlsData*, or the server fails to verify this binding. The adversary can transplant messages between such sessions because the client binds only to the endpoints of the channel and not to the session.

channel binding, an adversary that masquerades as a client to the honest server, and as a server to the honest client-authenticator pair, potentially exploits the client's inability to authenticate the challenge. Thereby, the adversary tricks the pair to act as confused deputies to attest a challenge on the adversary's behalf. Figure 3 illustrates this attack when channel binding is not present. For clarity, we also show this attack when the client and authenticator are integrated into one role, as allowed by the specification—e.g., a client might run the protocol on their laptop which also serves as the authenticator (see Figure 4).

When one of the channel bindings is present, we verify that the authenticator's attestation must include the appropriate binding parameters. Consequently, an adversary that tricks a client-authenticator pair into generating an attestation will receive an attestation that binds to the incorrect communication channel, enabling the legitimate server to identify this discrepancy and terminate the protocol. Under some circumstances, the standard permits a server to accept an incorrect channel binding, for example, when communicating with a client through a proxy. If the server accepts such a binding, it enables a similar attack to the one that exists when the protocol does not channel bind.

Even when the protocol channel binds, the server is unable to verify to which session of the protocol an attestation binds. This weakness results from TLS channel binding only to the communication endpoints rather than to the session. Consequently, the server may accept a channel-bound attestation from a client that corresponds to another session with the same client. In Figure 4, the server cannot determine to which session an attestation binds.

From the server's perspective, we note the following weaknesses of channel binding in UAF: (1) the client cannot authenticate the challenge prior to binding it, nor identify from what session the challenge originated; (2) the authenticator does not possess information to verify a binding that the client applies, producing an attestation regardless of the binding's legitimacy; (3) the channel bindings bind only to the endpoints, not to the protocol session; and (4) the server is the sole entity, and thus a single point of failure, that verifies the channel binding.

9.2 Client's Perspective

From the client's perspective, CPSA produces several shapes from which we observe that the client, following the TLS handshake, is certain it is communicating with a legitimate server, assuming (1) the adversary does not compromise the TLS CA or possess a legitimate certificate, and (2) there is a legitimate, registered authenticator to complete the protocol. Because many of the shapes feature the client communicating with different instances of the legitimate server concurrently, we assume correct behavior from this server and consolidate these shapes into a single shape.

Despite authenticating the server via the certificate, the client cannot authenticate the challenge directly, potentially acting on a challenge resulting from protocol interaction. This issue remains when applying one of the four channel-binding mechanisms, because the client is solely responsible for applying these bindings.

9.3 Authenticator's Perspective

In the absence of channel binding, the authenticator's perspective comprises a single shape: the authenticator receives a challenge from the honest client and replies with an attestation. This shape confirms that, when the adversary lacks knowledge of the access key, only the legitimate client can issue a challenge for the authenticator to attest. Due to a lack of information, the authenticator relies on the client to authenticate the challenge and the server's *appID*, potentially enabling the authenticator to attest malicious challenges that the client fails to authenticate—the server's perspective illustrates one such scenario when the client fails to bind the challenge.

With channel binding, CPSA generates an additional shape in which an authenticator acts as the TLS CA, signing an attestation comprising a server's certificate with the CA's secret key. We do not consider this scenario plausible because our threat model does not consider a compromised CA.

10 Attack Implementation

We exploit missing channel binding to carry out a MitM attack against eBay's popular open-source UAF server [17], which implements a subset of UAF 1.2, enabling us to authenticate as an honest user without access to the user's authenticators.

To carry out the protocol, we implement a basic client in the Python programming language. Our client registers dummy authenticators with the server and responds to authentication requests with valid assertions from these authenticators. We implement a malicious server as a process that passes messages between the client and the UAF server. Our attack sourcecode is available on GitHub [26].

We assume: (1) an honest user wishes to communicate with a DY adversary acting as a server; (2) there is a legitimate server on the network; (3) the adversary controls a subdomain under the server's URL (e.g., the adversary compromises one of the server's trusted facets [22]); and (4) the client does not implement channel binding. The adversary wishes to masquerade as the honest client to a legitimate server.

In addition to not supporting channel binding, the eBay FIDO UAF implementation fails to inspect the contents of an assertion's final challenge parameters, verifying only an authenticator's signature. This egregious omission likely results in additional potential vulnerabilities beyond the scope of our work: an adversary can freely substitute challenge parameters without the server's knowledge. For our attack, we assume only that the server does not enforce any channel binding.

The attack proceeds in two steps, registration and authentication, which both exploit lack of binding of the UAF challenge.

10.1 Step 1: Registration

In our attack's first phase, the adversary exploits missing binding to register the honest user's authenticators with the legitimate server, claiming to own these authenticators.

First, the honest user intentionally attempts to registers this authenticator with the adversary, who masquerades as an honest server on the network. The adversary now engages in two concurrent UAF registration protocols: one between itself and the user, and one between itself and an honest server, presenting the user with the honest server's challenge. Subsequently, the user issues the challenge to their authenticator, which generates a *key registration data (KRD)* object, including an *authenticator attestation ID (AAID)*, a new public-private key pair, an attestation certificate, a pair of counters (registration and signature), and a hash of these parameters including the malicious challenge. Using the private key, the authenticator signs the KRD and returns an assertion to the user, who forwards it to the adversary. The adversary now claims the assertion to the legitimate server, thereby registering the user's authenticator without their knowledge.

10.2 Step 2: Authentication

The adversary engages in parallel UAF authentication protocols, transplanting the legitimate server's challenge and policy, which includes the user's authenticator maliciously registered by the adversary, into the session between the adversary and the user. The user issues the challenge to the authenticator, which builds an authentication assertion that the adversary presents to the legitimate server as their own to complete the protocol and authenticate. Claiming the user's identity, the adversary is now free to engage in malicious behaviors under the user's name.

11 Dual Channel Binding: Client and Server

We propose a variation of UAF authentication, which we call "*dual channel binding*," in which both the server and the client bind the challenge, rather than channel binding solely at the client. As discussed in Section 9, even when we incorporate the optional channel binding present in UAF, the client cannot authenticate the challenge and therefore must rely on the server to verify the client's channel binding correctly. The client's inability to authenticate the challenge puts them at risk to act as a confused deputy and generate attestations for malicious challenges. There is no guarantee that an implementation of the FIDO server verifies the client's attestation correctly, and the standard permits a server to accept incorrect channel bindings. See Section 10 for examples of implementations that fail to perform this step. In this section, we present a meaningful improvement of UAF that expands channel binding to both the client and the server, enabling the client to identify which server generated a challenge.

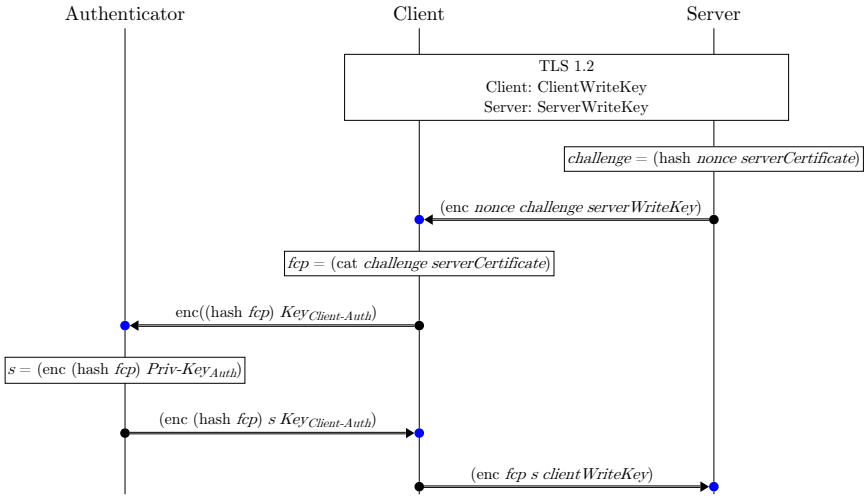

Fig. 5. Idealized message sequence diagram of a CPSA model for our proposed UAF dual binding, in which both the client and the server apply cryptographic channel binding to the challenge. In UAF dual binding, the server's challenge comprises a hash of (a) a random, fresh nonce that the server chooses, and (b) the server's certificate. The client hashes the nonce together with the server's certificate to authenticate the challenge, enabling the client to terminate the protocol when receiving a challenge bound to an incorrect certificate.

Figure 5 illustrates our idealized message sequence for UAF dual binding, in which both the client and the server apply channel binding to the challenge. In our analysis of this model, we again consider three perspectives: client, server, and authenticator.

From the client's perspective, we produce shapes representing the ideal execution of the protocol, varying only to illustrate that we cannot establish that a certificate-based binding binds to a specific TLS session between an honest client and honest server.

The authenticator's perspective yields a single shape in which the authenticator is certain they are attesting a challenge from the legitimate client.

From the server's perspective, because most applications of TLS authenticate only the server to the client, we observe several points of interest: (1) the server is unable to authenticate the client, because the client produces no certificate; (2) due to the inability of the server to determine the freshness of the client's TLS 1.2 pre-master secret, it cannot assume a confidential TLS channel between itself and a client; and (3) because the TLS channel bindings do not incorporate session-specific data, the client cannot verify from which server session a challenge originates.

For simplicity, our dual binding proposal also binds to the endpoints, not to the channel. We conjecture that dual binding could be further improved by binding also to the TLS channel's master secret.

We note three limitations of our proposal: (1) As is true for UAF channel binding, as a result of a limitation of the most common form of TLS, the server cannot authenticate the client. (2) To verify the server's binding, the client must support certificate channel binding. (3) When communicating through a perimeter proxy, communicants cannot verify the binding of the challenge.

From the shapes we analyzed, we conclude that dual-binding UAF authentication enables the client to verify the origin, but not the specific session, of the challenge while mitigating the man-in-the-middle attack (when binding is absent) discussed in Section 9. With dual binding the client can terminate the session if their verification of the challenge fails. For these reasons, our proposal presents an improvement over the standard's existing client-only channel binding.

12 Discussion

We now discuss several issues that arose in our work: (1) the necessity of binding cryptographic values at their origin, (2) implications of making channel binding optional in UAF, and (3) open problems.

12.1 Binding Values at their Origin

In a prudent protocol design, every protocol role must bind cryptographic values to a session's context explicitly. Binding session data as early as possible—ideally at the origin of the data—reduces an adversary's opportunities to produce protocol interactions. To build a session context, we suggest incorporating an underlying cryptographic session, incorporating additional attributes such as the protocol's name, message sequence numbers, unique session identifiers, communicant identifiers, and any other values unique to the session. Many protocol interactions result from an inability to authenticate sensitive cryptographic values, resulting from a lack of binding. Often, the vulnerability can be mitigated by binding to a context appropriately.

12.2 UAF's Optional Channel Binding

,

To promote adoption of the UAF standard, FIDO specified channel binding as an optional feature. It is possible that requiring this binding would result in developers preferring older, password-based solutions over UAF, due to the challenge of implementing the channel-binding requirement. When both the client and the server support channel binding, the client applies this binding to the challenge parameters prior to forwarding them to their authenticators. We now discuss UAF's optional channel binding from a technical and a policy perspective.

From a cryptographer's perspective, without channel binding, the UAF authentication protocol is vulnerable to a DY adversary: the adversary can exploit

the lack of channel binding to launch a MitM attack, masquerading as an honest client to a server, without access to the client's authenticator. To mitigate this attack, the protocol must require the client, and potentially the server, to bind the challenge (see Section 11).

From a policy perspective, requiring channel binding in the UAF standard presents practical problems: entities unable to support the channel binding standard may favor a different system over UAF, and some of the TLS channel bindings are still in draft form. Optional channel binding is a conscious trade-off between security and ease of adoption, which supports the goal of reducing and eliminating traditional, password-based security. Policy makers, adopters, and users, however, should be aware that optional channel binding enables an adversary to create potentially serious protocol-interactions in FIDO UAF authentication.

Additionally, the specification permits a server to disregard incorrect bindings to facilitate clients that bind to other channels (e.g., when communicating through a perimeter proxy, as explained in Section 5.2). From a cryptographer's perspective, a server that fails to verify a binding enables harmful protocol interactions. From a policy perspective, supporting clients that communicate through proxies facilitates the adoption of UAF but potentially enables hostile proxies (adversaries) to attack the protocol. Developing a mechanism for the server to verify proxy bindings, such as cryptographically binding the client's channel with the proxy to the server's session, may mitigate the weakness enabled by this policy.

12.3 Open Problems and Future Work

Our work on FIDO UAF authentication raises four open problems: (1) Further analyze the UAF registration protocol, which similarly binds server-generated challenges at the client. (2) Search for potential protocol interactions between the authentication and registration protocols, exploiting similarities between these two protocols. (3) Develop channel bindings that bind to protocol endpoints and sessions, and encapsulate additional bindings to address cases where the client or server communicate with a proxy. (4) Explore channel binding in FIDO2 [23], a new version of FIDO building on UAF that expands the available types of authenticators and improves interoperability with existing standards.

As future work, we are developing automatic tools for cryptographically binding protocol messages to their context in a manner that provably eliminates the possibility of any protocol interaction in the DY model.

13 Recommendations

We recommend: (1) For applications where it is critical to mitigate protocol interactions, the standard require both the server and the client to apply channel bindings to the challenge, as we explain in Section 11. (2) The server should not accept attestations that bind to channels it cannot verify. (3) Applications should

consider limitations of TLS channel bindings in UAF: these bindings bind only to endpoints of the session; they do not bind to specific protocol sessions. (4) The FIDO Alliance should adopt a more formal adversarial model, including a DY adversary, and perform formal-methods analysis of the UAF specification.

14 Conclusion

Using CPSA, we performed a formal-methods analysis of FIDO UAF authentication channel bindings. From this analysis, we: (1) showed that FIDO UAF's channel bindings fail to mitigate protocol interaction, resulting in a MitM attack from the server's failure to bind its challenge adequately. (2) implemented the attack against eBay's open-source FIDO UAF server, allowing an adversary to masquerade as a legitimate client. (3) proposed and verified an improved version in which both the client and server perform channel binding, allowing the client and server each to verify the binding, providing some protection even if the server fails to verify the binding correctly, and performing the binding at the server where the challenge is created.

Our attack exploits four limitations of channel binding in FIDO UAF: (1) channel binding is optional; (2) the server may accept attestations that bind to incorrect channels; (3) the client cannot verify the origin of the server's challenge; and (4) UAF binds only to the protocol's endpoints, not to the session. In addition, the most common application of TLS performs only one-way authentication, preventing the server from authenticating the client.

Although the FIDO UAF specification suggests that omitting channel binding may create a vulnerability (see Section 5), to our knowledge, we are first to carry out a formal-methods analysis of channel binding in FIDO UAF and first to exhibit details of an attack on FIDO UAF that exploits the weaknesses of UAF's channel binding. Previous studies of FIDO UAF inadequately analyzed the role of channel binding. The problems we uncover with FIDO UAF result significantly from the failure of FIDO designers to adopt an appropriate adversarial model. Policy makers should be aware that omitting channel binding, or accepting attestations that bind to the incorrect channel, creates a serious vulnerability in which the adversary can trick the client and authenticator to act as confused deputies to sign an authentication challenge for the adversary.

Despite decades of progress in protocol design, many current protocols—including FIDO UAF—fail to apply cryptographic binding consistently and correctly, allowing potential harmful protocol interactions. Our case study of FIDO UAF illustrates the value of adopting a well-defined adversarial model and using formal-methods tools, such as CPSA, in protocol analysis—including analysis in the design process—and the need to develop and apply rigorous techniques to ensure that all protocol messages are always properly cryptographically bound to their context.

Acknowledgements

We thank Ian Blumenfeld and Joshua Guttman for helpful comments. Alan Sherman was supported in part by the National Science Foundation under DGE grants 1753681 (SFS), 1819521 (SFS Capacity), and 2138921 (SaTC).

References

1. Abadi, M., Needham, R.: Prudent engineering practice for cryptographic protocols. IEEE Transactions on Software Engineering **22**(1), 6–15 (1996)
2. Alliance, F.: FIDO alliance member companies and organizations (Jan 2023), https://fidoalliance.org/members/
3. Alliance, F.: FIDO certified products (Jan 2023), https://fidoalliance.org/certification/fido-certified-products/
4. Altman, J.E., Zhu, L., Williams, N.: Channel Bindings for TLS. RFC 5929 (Jul 2010). https://doi.org/10.17487/RFC5929, https://rfc-editor.org/rfc/rfc5929.txt
5. Baghdasaryan, D., Balfanz, D., Hill, B., Hodges, J., Yang, K.: FIDO UAF protocol specification v1.2. Tech. rep., FIDO Alliance (2020)
6. Baghdasaryan, D., Hill, B., Hill, J., Biggs, D.: FIDO security reference. Tech. rep., FIDO Alliance (2018)
7. Balfanz, D.: FIDO TechNote: The growing role of token binding. https://fidoalliance.org/fido-technotes-channel-binding-and-fido/ (2016), accessed: 2023-4-8
8. Balfanz, D., Hamilton, R.: Transport Layer Security (TLS) Channel IDs (Jun 2013), https://datatracker.ietf.org/doc/draft-balfanz-tls-channelid/01/, work in Progress
9. Blanchet, B.: Automatic verification of security protocols in the symbolic model: The verifier ProVerif. In: Foundations of Security Analysis and Design VII, pp. 54–87. Springer, Berlin, Germany (2013)
10. Büttner, A., Gruschka, N.: Protecting FIDO extensions against man-in-the-middle attacks. In: Emerging Technologies for Authorization and Authentication: 5th International Workshop, ETAA 2022, Copenhagen, Denmark, September 30, 2022, Revised Selected Papers. pp. 70–87. Springer, Berlin, Germany (2023)
11. Chadwick, D.W., Laborde, R., Oglaza, A., Venant, R., Wazan, S., Nijjar, M.: Improved identity management with verifiable credentials and FIDO. IEEE Communications Standards Magazine **3**(4), 14–20 (2019)
12. Chae, C.J., Cho, H.J., Jung, H.M.: Authentication method using multiple biometric information in FIDO environment. Journal of Digital Convergence **16**(1), 159–164 (2018)
13. Cremers, C.J.: The Scyther Tool: Verification, falsification, and analysis of security protocols. In: International Conference on Computer Aided Verification. pp. 414–418. Springer, Berlin, Germany (2008)
14. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Aug 2008), http://www.ietf.org/rfc/rfc5246.txt
15. Doghmi, S., Guttman, J., Thayer, J.: Searching for shapes in cryptographic protocols. In: Tools and Algorithms for the Construction and Analysis of Systems: 13th International Conference. pp. 523–537. Springer, Berlin, Germany (2007)

16. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Transactions on Information Theory **29**(2), 198–208 (1983). https://doi.org/10.1109/TIT.1983.1056650
17. eBay: eBay FIDO UAF Implementation. `https://github.com/eBay/UAF` (2022), accessed: 2023-03-31
18. Escobar, S., Meadows, C., Meseguer, J.: Maude-NPA: Cryptographic protocol analysis modulo equational properties. In: Foundations of Security Analysis and Design V, pp. 1–50. Springer, Berlin, Germany (2009)
19. Fábrega, F.J.T., Herzog, J.C., Guttman, J.D.: Strand spaces: Proving security protocols correct. Journal of Computer Security **7**(2/3), 191–230 (1999)
20. Fuchs, J., Hamer, S., Liu, D.: A man-in-the-middle attack on the FIDO UAF registration protocol. CSEE Dept, UMBC, unpublished manuscript (2022)
21. Hardy, N.: The confused deputy: (or why capabilities might have been invented). ACM SIGOPS Operating Systems Review **22**(4), 36–38 (1988)
22. Hill, B., Balfanz, D., Baghdasaryan, D.: FIDO AppID and facet specification. Tech. rep., FIDO Alliance (2018)
23. Hodges, J., Jones, J.C., Jones, M.B., Kumar, A., Lundberg, E.: Web authentication: An api for accessing public key credentials level 2. `https://www.w3.org/TR/webauthn/` (2021)
24. Hu, K., Zhang, Z.: Security analysis of an attractive online authentication standard: FIDO UAF protocol. China Communications **13**(12), 189–198 (2016)
25. Kelsey, J., Schneier, B., Wagner, D.: Protocol interactions and the chosen protocol attack. In: International Workshop on Security Protocols. pp. 91–104. Springer, Berlin, Germany (1997)
26. Lab, U.P.A.: PAL GitHub repository. `https://tinyurl.com/3d2wnhuf` (April 2023)
27. Liskov, M., Rowe, P., Thayer, J.: Completeness of cpsa. Tech. rep., MITRE (2011), `https://www.mitre.org/sites/default/files/pdf/12_0038.pdf`
28. Liskov, M.D., Ramsdell, J.D., Guttman, J.D., Rowe, P.D.: The Cryptographic Protocol Shapes Analyzer: A Manual. The MITRE Corporation (2016)
29. Loutfi, I., Jøsang, A.: FIDO trust requirements. In: Nordic Conference on Secure IT Systems. pp. 139–155. Springer, Berlin, Germany (2015)
30. Lowe, G.: An attack on the needham-schroeder public-key authentication protocol. Information Processing Letters **56**(3), 131–133 (1995), `http://www.sciencedirect.com/science/article/pii/0020019095001442`
31. Meadows, C.: The NRL protocol analyzer: An overview. The Journal of Logic Programming **26**(2), 113–131 (1996)
32. Meier, S., Schmidt, B., Cremers, C., Basin, D.: The TAMARIN prover for the symbolic analysis of security protocols. In: International Conference on Computer Aided Verification. pp. 696–701. Springer, Berlin, Germany (2013)
33. Millen, J., Clark, S., Freedman, S.: The interrogator: Protocol security analysis. IEEE Transactions on Software Engineering **SE-13**(2), 274–288 (1987). https://doi.org/10.1109/TSE.1987.233151
34. Needham, R.M., Schroeder, M.D.: Using Encryption for Authentication in Large Networks of Computers. Commun. ACM **21**(12), 993–999 (Dec 1978). https://doi.org/10.1145/359657.359659, `http://doi.acm.org/10.1145/359657.359659`
35. O'Neill, M., Ruoti, S., Seamons, K., Zappala, D.: TLS proxies: Friend or foe? In: Proceedings of the 2016 Internet Measurement Conference. pp. 551–557. ACM, New York, NY (2016)

36. Panos, C., Malliaros, S., Ntantogian, C., Panou, A., Xenakis, C.: A security evaluation of FIDO's UAF protocol in mobile and embedded devices. In: International Tyrrhenian Workshop on Digital Communication. pp. 127–142. Springer, Berlin, Germany (2017)
37. Paulson, L.: Inductive analysis of the internet protocol TLS: Transcript of discussion. In: Security Protocols: 6th International Workshop Cambridge, UK, April 15–17, 1998 Proceedings 6. pp. 13–23. Springer, Berlin, Germany (1999)
38. Pereira, O., Rochet, F., Wiedling, C.: Formal analysis of the FIDO 1.x protocol. In: Foundations and Practice of Security: 10th International Symposium, FPS 2017, Nancy, France, October 23-25, 2017, Revised Selected Papers 10. pp. 68–82. Springer, Berlin, Germany (2018)
39. Popov, A., Nystrom, M., Balfanz, D., Langley, A., Hodges, J.: The Token Binding Protocol Version 1.0. RFC 8471 (Oct 2018). https://doi.org/10.17487/RFC8471, `https://rfc-editor.org/rfc/rfc8471.txt`
40. Ulqinaku, E., Assal, H., Abdou, A., Chiasson, S., Capkun, S.: Is real-time phishing eliminated with FIDO? social engineering downgrade attacks against FIDO protocols. In: 30th USENIX Security Symposium (USENIX Security 21). pp. 3811–3828. USENIX Association, Berkley, CA (Aug 2021), `https://www.usenix.org/conference/usenixsecurity21/presentation/ulqinaku`

A Acronyms and Abbreviations

AAID	Authenticator Attestation ID
CA	Certificate Authority
CPSA	Cryptographic Protocol Shapes Analyzer
DY	Dolev-Yao
FIDO	Fast Identity Online
KRD	Key Registration Data
MitM	Man-in-the-middle
NS	Needham-Schroeder
TLS	Transport Layer Security
UAF	Universal Authentication Framework

B CPSA Model Code (Selected Examples)

We provide key CPSA model code snippets from our FIDO CPSA models, organized as roles and skeletons. For complete models, see Github [26].

```
(defrole client
(vars
 (auth client server ca name)
 (n1 n2 pms challenge text)
 (pks akey)
 )
(trace
 (TLSclient_nocerts n1 n2 pms server pks ca)
 (recv (enc challenge (ServerWriteKey (MasterSecret pms n1 n2))))
 (send (enc (hash challenge <tlsData>) (ltk client auth)))
 (recv (enc (signed (hash challenge <tlsData>) auth) (ltk client auth)))
 (send (enc (signed (hash challenge <tlsData>) auth)
    (ClientWriteKey (MasterSecret pms n1 n2))))
 )
)
```

Fig. 6. CPSA specification of a client in UAF authentication. The *TLSclient_nocerts* macro generates client messages to establish a TLS session with the server, reflecting the common case of omitting a client certificate. When hashing the challenge, the client has the option of incorporating one of several channel bindings via macros to substitute *¡tlsData¿*, or can leave this value blank to model unbound UAF authentication. The value *pks* represents the server's public key.

```
(defrole server
(vars
 (auth server ca name)
 (n1 n2 pms challenge text)
 (pks akey)
 )
(trace
 (TLSserver_nocertreq n1 n2 pms server pks ca)
 (send (enc challenge (ServerWriteKey (MasterSecret pms n1 n2))))
 (recv (enc (signed (hash challenge <tlsData>) auth)
    (ClientWriteKey (MasterSecret pms n1 n2))))
 )
)
```

Fig. 7. CPSA specification of a server in UAF authentication The *TLSserver_nocertreq* macro generates server messages to establish a TLS session with the client without requiring a client certificate. In our model, the server must reflect the client's choice for the *¡tlsData¿* channel binding by incorporating the appropriate macro (see Figure 9). The server specifies the challenge as uniquely originating for each session and assumes that the adversary does not possess the legitimate CA's private key.

```
;;; Sign a message and append the signature.
(defmacro (signed m i)
    (cat m (enc m (privk i)))
)

(defrole auth
(vars
 (auth client name)
 (fcp mesg)
 )
(trace
 (recv (enc fcp (ltk client auth)))
 (send (enc (signed fcp auth) (ltk client auth)))
 )
)
```

Fig. 8. CPSA specification of an authenticator in UAF authentication. Relying on a long-term key with the client, the authenticator receives the final challenge parameters *fcp*, which may incorporate the client's optional channel binding, and generates a signed attestation using its private key.

```
(defmacro (channel_binding_endpoint challenge server pks ca)
  (hash challenge (hash (Certificate server pks ca)))
)

(defmacro (channel_binding_servercert challenge server pks ca)
  (hash challenge (Certificate server pks ca))
)

(defmacro (channel_binding_channel_id challenge client)
  (hash challenge (pubk client) (enc (pubk client) (privk client)))
)

(defmacro (channel_binding_token challenge client)
  (hash challenge (cat (pubk client) (enc (pubk client) (privk client))))
)
```

Fig. 9. CPSA macros for incorporating different channel bindings in *tlsData*. Each macro, which we name after the corresponding channel binding in the UAF standard, implements a cryptographic abstraction of that binding in CPSA.

```
;;; Client perspective
(defskeleton fido
  (vars (auth client server ca name) (n1 pms text) (pks akey))
  (defstrandmax client
    (auth auth)
    (client client)
    (server server)
        (ca ca)
    (n1 n1)
    (pms pms)
        (pks pks))
  (non-orig
   (privk auth)
   (privk ca)
   (ltk client auth)
   (invk pks))
  (uniq-orig n1 pms)
)
```

Fig. 10. CPSA skeleton specifying a client's perspective. In CPSA, we state assumptions that it is impossible for the adversary to obtain the private keys of the legitimate authenticator, server, or certificate authority. We also assume that the adversary cannot compromise the channel between the authenticator and the client. The client generates fresh values for the TLS session with the server and the server generates a fresh challenge for each session.

```
;;; Server perspective
(defskeleton fido
  (vars (auth server ca name) (n2 challenge text) (pks akey))
  (defstrandmax server
    (auth auth)
    (server server)
        (ca ca)
    (n2 n2)
        (challenge challenge)
        (pks pks))
  (non-orig
   (privk auth)
   (privk server)
   (privk ca)
   (invk pks))
  (uniq-orig n2 challenge)
)
```

Fig. 11. CPSA skeleton specifying a server's perspective. Similarly with the client, the server assumes that the adversary cannot access the server nor the authenticator's private keys, and the server generates a fresh nonce for each TLS session.

```
;;; Authenticator's perspective
(defskeleton fido
  (vars (auth client name))
  (defstrand auth 2
    (auth auth)
    (client client))
  (non-orig
   (privk auth)
   (ltk client auth))
)
```

Fig. 12. CPSA skeleton specifying an authenticator's perspective. The authenticator assumes confidentiality of its own key and of the long-term key by which it communicates with the client.