Computationally Modeling User-Mediated Authentication Protocols

Britta Hale *

Naval Postgraduate School (NPS)
Monterey, California, USA
britta.hale@nps.edu

Abstract User interaction constitutes a largely unexplored field in protocol analysis, even in instances where the user takes an active role as a trusted third party, such as in the Internet of Things (IoT) device initialization protocols. Initializing the study of computational analysis of 3-party authentication protocols where one party is a physical user, this research introduces the 3-party possession user mediated authentication (3-PUMA) model. The 3-PUMA model addresses active user participation in a protocol which is designed to authenticate possession of a fixed data string – such as in IoT device commissioning. To demonstrate the 3-PUMA model in practice, we provide a computational analysis of the ISO/IEC 9798-6:2010 standard's Mechanism 7a authentication protocol which includes a user interface and interaction as well as a device-to-device channel. We show that the security of ISO/IEC 9798-6:2010 Mechanism 7a relies upon a non-standard MAC security notion, which we term existential unforgeability under key collision attacks (EUF-KCA). It is unknown if any standardized MAC algorithm achieves EUF-KCA security, indicating a potential vulnerability in the standard.

Keywords Authentication Protocols · key distribution · User Interface · MAC Security · Key-Collision Attacks

1 Introduction

While work has been done on modeling of 3-party – and more generally multi-party – key exchange protocols [9, 16, 17, 26, 35], 3-party authentication protocols are largely ignored. Analyses of many 3-party key exchange protocols handle the user as an out-of-band (OOB) information exchange [35, 17]. Indeed, this follows from standard practice where security is only considered device-to-device and identification of a device's user is considered irrelevant or external to the cryptographic model. However, in a user-mediated protocol, the user is an active participant relaying and confirming information and even generating nonces or keys, instead of a simple possessor of a device. It is thus possible to consider a user-to-device "'channel", e.g. a device keypad or display, as well as adversarial behavior on this channel. For example, an adversary may have *a priori* access to a device and may therefore be able to manipulate inputs/outputs.

* The views expressed in this document are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

ISO/IEC 9798–6:2010 Mechanism 7a Analysis. One such user-mediated protocol is ISO/IEC 9798–6:2010 Mechanism 7a [23] (abbreviated Mechanism 7a) authentication protocol, originally published in [21]. Unlike previously analyzed ISO/IEC 9798:2010 protocols, protocols within the ISO/IEC 9798–6 standard employ an active user interface. Furthermore, the goals of Mechanism 7a differ also from those expected from typical mutual authentication protocols; instead of entity authentication, Mechanism 7a achieves a form of data authentication, which can be seen as a version of "aliveness" [31]. In the absence of long-term keys and symmetric keys, etc., which could normally be used for entity authentication, the goal of Mechanism 7a is to provide a mutually authenticated data string D, such that both parties are assured that the protocol partner has also agreed to the string. Protocol participant identities are assumed to be known *a priori* but no verification takes place during the protocol. In fact, the only intended prevention mechanism against man-in-the-middle attacks is a user generated value, which again highlights the highly interactive nature of the protocol.

3-PUMA Model. This research introduces the 3-party possession user mediated authentication (3-PUMA) model, for protocols with an active user protocol participant and authentication of mutually held data. Application of the 3-PUMA model extends beyond the current analysis of ISO/IEC 9798–6 Mechanism 7a, and has implications for analysis of other common IoT protocols which demand user interaction for authentication. Furthermore, this extends previous research on other ISO/IEC 9798 authentication protocols which do not demand a user interface.

EUF-KCA MAC Security. The security goals of Mechanism 7a's underlying message authentication code (MAC) algorithm differ not only from the standard's currently assumed MAC goals, but also from all accepted MAC security assumptions (e.g. EUF-CMA and SUF-CMA). Mechanism 7a sends a one-time MAC key in the clear, before verifying the MAC tag. Thus the MAC must be secure against an adversary that can produce a different but valid MAC key – essentially producing a key forgery, given a message-tag pair. In order to address these demands, we formalize key collision attacks (EUF-KCA). We cross-compare EUF-KCA security with other standard security assumptions. The non-standard reliance on EUF-KCA security calls into questions the security of Mechanism 7a. It is currently unknown which, if any, standardized MAC algorithms achieve EUF-KCA security.

Related Work. Previous analyses of the ISO/IEC 9798 standard have addressed mechanisms in the standard which do not include a user interface [4, 22, 41]. These analyses include both formal modeling [4, 41] and computational modeling [22]. None of these works cover any of the protocols of ISO/IEC 9798–6, but demonstrate the importance of analyzing such standardized protocols. One exception to the above list undertakes capturing the user interface in ISO/IEC 9798–2 and –4 using symbolic analysis [19]. However symbolic and computational analysis differ considerably in the scope and detail of the analysis performed [18]. Computational analysis, as is used in this work, enables a detailed view of the underlying algorithms, such as the EUF-KCA security assumption discussed above.

Classically, we assume that an adversary cannot access an algorithm's secret key. This applies to MACs, digital signatures, etc.; if an adversary obtains the key and can generate new MACs or signatures with it, security is lost. However, this classical approach inherently assumes that changes to a MAC tag or digital signature would not be noticed. Naturally this is the case with document signatures, where generation of a new signature on an edited document yields an apparently valid message-signature pair. Yet Mechanism 7a contains a different scenario, one in which MAC tags are essentially "committed" to. Then, even if the key is revealed, the tag cannot be altered and an adversary must generate a message forgery that corresponds to the fixed tag, or produce an alternative key.

Known Key, Chosen Key, and Related Key attacks bear some similarities to the present adversarial MAC challenge of producing a new key given a known key; however it is important to note that these differ from the present case. Known Key attacks (KKAs) were introduced in [27] and cover the case of block ciphers where an adversary knows a key and aims to exhibit non-random behavior in the cipher. KKAs have been studied extensively [2, 14, 20, 33, 38, 39, 40]. Chosen Key attacks (CKAs) consider a similar situation, but where an adversary may choose the key in question [13, 30, 36]. In a Related Key attack (RKA) [11] an adversary chooses a relation between a pair of keys for a blockcipher, but not the keys themselves, before launching a chosen plaintext attack. Much research has been done on RKAs, both in the development of practical attacks and in the theoretical definitions [3, 5, 6, 7, 12, 15, 32, 34]. As KKAs, CKAs, and RKAs are aimed at block ciphers, they can be relevant for MAC security, and RKAs have already been considered in the context of MACs [10, 28]. However, the intrinsic goal of these attacks (e.g. non-random behavior in the cipher in the case of a KKA) is different from the goal of the MAC adversary exhibited in Mechanism 7a (generating a new key for a given message-tag pair).

Standard MAC security variants include EUF-CMA and SUF-CMA. In both cases the MAC key is fixed, unlike the MAC security demands of Mechanism 7a where an adversary may select the verification key to be used. The concept of adversarial key guessing has surfaced previously under terms such as *key spoofing* and *key-collision*. The concept of *key spoofing* was briefly discussed in [1], in terms of symmetric encryption, as a situation where an adversary's goal is to find a new key which produces a given message-ciphertext pair. Later, the idea was revived in [37] (not peer-reviewed) under the term *key-collision* for digital signatures. Unlike in key spoofing, key collision demands only a fixed ciphertext – the adversary must find a new key, and may additionally find a new message, which yields the given ciphertext. If the messages are indeed the same, the term *1^{st} order key collision* is employed, while *2^{nd} order key collision* is reserved for the case of different messages. [37] demonstrates key-collision in DSA and ECDSA. Still, these attacks do not consider a related-key case, where the adversary can exploit knowledge of the actual key. This leaves an open problem. How can we formulate MAC security for when the MAC key is intentionally provided to the adversary?

We consider a slightly different variant of the 1^{st} order key collision goal, where the adversary not only guesses a different valid key but is actually provided the correct key, thus enabling attacks based on related keys, etc. Providing the adversary with the correct validation key necessitates restriction of

Exhaustive comparison of MAC security experiments. MAC security experiments corresponding to verification inputs which are optionally generated by the adversary (*), required fresh for a win (✓), and fixed (-).			
Security Experiment	K	m	t
EUF-KCA	✓	*	-
EUF-CMA	-	✓	*
Forged tag	-	*	✓
SUF-KCA	✓	✓	-
SUF-CMA = EUF-CMA + Forged tag	-	✓	✓
Trivial (win)	✓	-	✓
Trivial (win)	✓	✓	✓
Trivial (impossibility)	-	-	-

Table 1. Security experiment against a MAC algorithm with corresponding new inputs into the verification oracle (key K, message m, or MAC tag t). If an adversary can generate a new input (✓) when others are fixed (-), it wins the corresponding experiment. Other inputs which the adversary may manipulate are denoted (*). For example, an adversary wins SUF-CMA if it can generate a new MAC tag, a new message or both. For visual completeness we include the trivial combinations where the adversary must generate a new key-tag pair for a message (whether given or of the adversary's choice). SUF-KCA and EUF-CMA experiments can be found in Appendix A.

its use, leading us to a one-time variant. We call this existential unforgeability under key collision attacks (EUF-KCA), as the adversary is essentially forging an alternative key with knowledge of a valid one. It is also possible to draw comparisons to EUF-CMA and SUF-CMA, based on information available to an adversary. Table 1 shows all possible forgeries an adversary can perform and the corresponding security game that captures such abilities. In all non-trivial cases, either the key or message tag is fixed. Note that classical SUF-CMA fixes the key while SUF-KCA fixes the tag.

Other real-world examples exist where a MAC tag, or even digital signature, is committed to in advance. One could consider implications of a signature variant of the EUF-KCA model for blockchain, for example, where a digital signature is publicly committed to and the adversary wins if it can produce a new message or key that matches the given signature.

Since we allow the adversary full control of the key between generation of a MAC tag and verification, the EUF-KCA security model falls in between the typical *secret-key* cryptographic assumption, where the adversary cannot access the key, and the *open-key* model (used in KKA and CKA). In the latter case, an adversary knows – or even is in control of – the secret key.

Contributions. This work extends previous research on ISO/IEC 9798, providing a model for user-mediated authentication and analyzing a previously untouched protocol. Moreover, and in response to these new authentication goals and challenges, the contributions in this paper include the following:

– We initiate the study of computational analysis for user-mediated protocols and introduce the 3-party Possession User-Mediated Authentication (3-PUMA) protocol security model.
– We introduce and provide security definitions for Existential Unforgeability under Key Collision Attacks (EUF-KCA) for MAC algorithms.
– We computationally analyze the ISO/IEC 9798–6:2010 authentication protocol under the 3-PUMA model. Ultimately, we demonstrate that the MAC requirements stated in Mechanism 7a are insufficient for the protocol's security.

2 Preliminaries

Here we introduce Mechanism 7a according to the ISO/IEC 9798–6:2010 standard's specification, as well as MAC definitions. The latter definitions include a standard security experiments for a MAC (SUF-CMA), as well as MAC security requirements per Mechanism 7a and the EUF-KCA security experiment.

2.1 ISO/IEC 9798–6:2010 Mechanism 7a Authentication Protocol Specification

Both devices possess a "simple output interface", e.g. red and green lights. They also possess "standard input" interfaces which allow a user to input a bit-string into the devices. Fig. 1 shows the Mechanism 7a protocol, with user-interface interaction displayed in green and device-to-device message flows displayed in blue.

Variables.

– R: A short random bit-string generated by the user. Per ISO/IEC 9798, R should be 16-20 bits.
– D: A data string. D is the agreed upon data at the termination of the protocol run.
– K_I: A short-term session key derived by identity I. Per ISO/IEC 9798, K_I should be 128-160 bits.
– mac_I: A message authentication code output by a MAC algorithm. Per ISO/IEC 9798, a *mac* should be 128-160 bits, with algorithms chosen from ISO 9797 [24, 25].
– `ready`: An indicating signal that the device is ready for the protocol to start.
– `start`: An initiation message.
– `OK`: An indicating signal that the protocol is completed successfully. Alternatively, a message `failed` is sent.

Note that Mechanism 7a does not define the distribution of the data string D, stating instead that it can be data transfered from one device to other other or concatenation of bilaterally transfered data. In either case, the mechanism does not begin until after the data has been distributed.

Mechanism 7a does not specify how each party obtains the other's identity, but makes it a specific requirement that identities are known prior to the start of the mechanism. Thus, we consider the identities to be transmitted out-of-band.

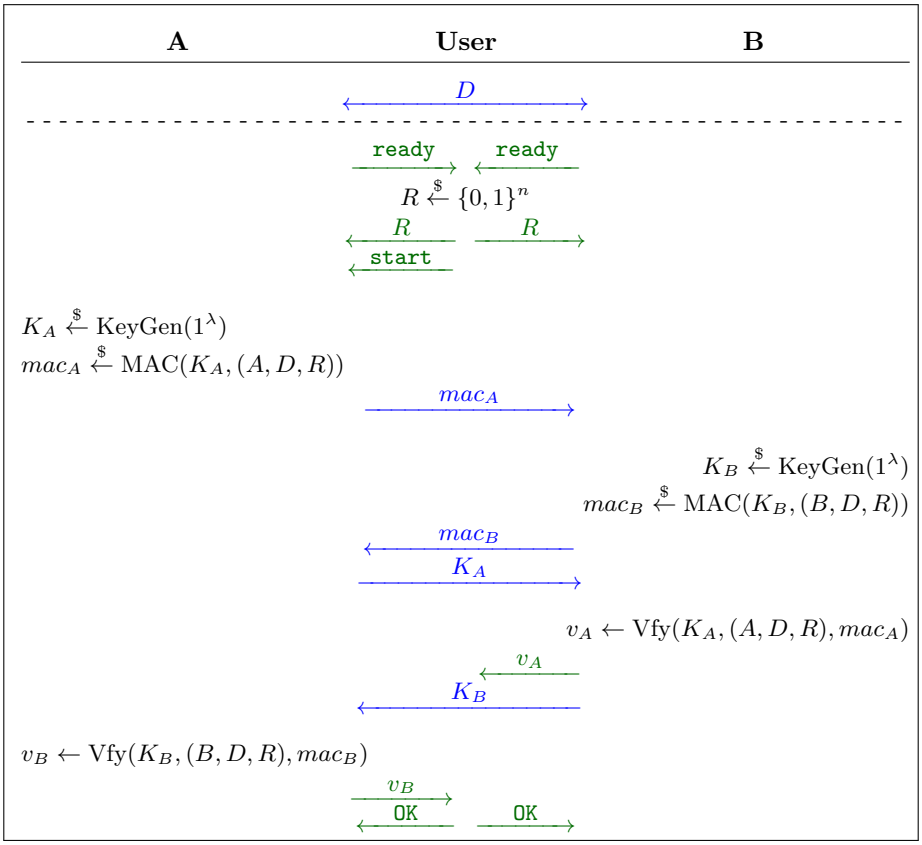

Figure 1. ISO/IEC 9798–6:2010 Protocol Mechanism 7a. Protocol flows are color-coded for the Device-to-Device channel and User-to-Device channel. Data string D is distributed before the start of the protocol and identities are pre-shared.

Mechanism 7a does not specify how the random bit-string R is generated, but does require it to be kept secret, as well as recommending caution during the user-to-device transfer of R. It should be noted that an improperly behaving user could re-use R in different protocol runs and, without checks on the reuse of R, this could lead to potential attacks. Following the specification, we assume that R is generated randomly. Additionally, we do not consider "shoulder-surfing" attacks, where an adversary may observe R on input, in accordance with the standard's strict specification on the secrecy of R to prevent man-in-the-middle attacks.

Device A, resp. B, outputs an indication of success/failure to the user based on the MAC verification step – we indicate this as $v_B = 1/0$, resp. $v_A = 1/0$. If both devices output an indication of success to the user ($v_B = v_A = 1$), then the user enters a confirmation of success (`OK`) into both devices. If either $v_B = 0$ or $v_A = 0$, the user enters an indication of failure `failed` into both devices; absence of a user response (`OK`/`failed`) within a specified time interval is interpreted as `failed` by the device.

Bit-length of keys, MACs, etc., clearly affects the security of the protocol. However, this also linked to the threat landscape and device capability. In this

analysis we will forgo specific consideration of bit security levels. Mechanism 7a also requires the implementation of time-out procedures (e.g. when a user fails to enter a confirmation `OK/failed` within a specified time interval); we assume that no such significant delays occur in the protocol execution.

We say that a protocol instance of Mechanism 7a *accepts* if:

- it has received a value R from the user, and
- it outputs a verification bit $v_{\text{partner identity}} = 1$, and
- the last message received from the user is `OK`.

2.2 MAC Security

Definition 1. *A message authentication code (MAC) algorithm for a message space $\mathcal{M}$, a key space $\mathcal{K}$, and an output message authentication code space $\mathcal{MAC}$ is a tuple of algorithms:*

- $\text{Kgn}(1^\lambda) \xrightarrow{\$} K$*: A probabilistic key generation algorithm that takes as input a security parameter λ and outputs a key $K \in \mathcal{K}$.*
- $\text{MAC}(K, m) \xrightarrow{\$} mac$*: A probabilistic message authentication algorithm that takes as input a key $K \in \mathcal{K}$ and a message $m \in \mathcal{M}$, and outputs a MAC tag $mac \in \mathcal{MAC}$.*
- $\text{Vfy}(K, m, mac) \to v$*: A deterministic verification algorithm that takes as input a key $K \in \mathcal{K}$, a message $m \in \mathcal{M}$, and a message authentication code $mac \in \mathcal{MAC}$, and output is a verification bit $v \in \{0, 1\}$.*

The authenticated message-tag pair *is the pair (m, mac), and mac is called the* MAC *or* tag *on m.*

Correctness: It is required for all $K \in \mathcal{K}$ and $m \in \mathcal{M}$, that $\text{Vfy}(K, m, \text{MAC}(K, m)) = 1$.

Mechanism 7a refers to the ISO/IEC 9797-1 standard for MACs for use. Particularly, Mechanism 7a uses the following definition for a MAC, which we present here for comparison and later discussion.

Definition 2 (Mechanism 7a MAC [23]). *A MAC algorithm is an algorithm for computing a function which maps strings of bits and a secret key to fixed-length strings of bits, satisfying the following properties:*

- *for any key and any input string the function can be computed efficiently;*
- *for any fixed key, and given no prior knowledge of the key, it is computationally infeasible to compute the function value on any new input string, even given knowledge of the set of input strings and corresponding function values, where the value of the ith input string may have been chosen after observing the value of the first i-1 function values.*

In juxtaposition to the above definition, we present the following security games for a MAC, which will be used in the analysis of Mechanism 7a.

$\underline{\mathsf{Exp}^{\mathsf{SUF\text{-}CMA}}_{\mathrm{MAC},\mathcal{A}}(\lambda)}$:

1: $K \xleftarrow{\$} \mathrm{Kgn}(1^\lambda)$
2: $S \leftarrow \emptyset$
3: $\mathcal{A}^{\mathsf{MAC}(\cdot),\mathsf{MAC.Vfy}(\cdot)}()$
4: **return** `phase`

$\underline{\mathsf{MAC}(m)}$:

1: $t \leftarrow \mathrm{MAC}(K, m)$
2: $S \leftarrow S \cup \{(m, t)\}$
3: **return** t

$\underline{\mathsf{MAC.Vfy}(m, t)}$:

1: $v \leftarrow \mathrm{Vfy}(K, m, t)$
2: **if** $(v = 1) \wedge ((m, t) \notin S)$ **then**
3: `phase` $\leftarrow 1$
4: **return** `phase` from experiment
5: **return** v

Figure 2. Security experiment for strong unforgeability (SUF-CMA) of a message authentication code algorithm $\mathrm{MAC} = (\mathrm{Kgn}, \mathrm{MAC}, \mathrm{Vfy})$ and adversary $\mathcal{A}$.

Definition 3. *Let $\mathcal{A}$ be a PPT adversarial algorithm against the MAC. The one-time strong unforgeability (OT-SUF-CMA) experiment for MAC, $\mathsf{Exp}^{\mathsf{OT\text{-}SUF\text{-}CMA}}_{\mathrm{MAC},\mathcal{A}}$ is given by the strong unforgeability (SUF-CMA) experiment $\mathsf{Exp}^{\mathsf{SUF\text{-}CMA}}_{\mathrm{MAC},\mathcal{A}}$ per Fig. 2, with the additional restriction that an adversary may only query* MAC *and* MAC.Vfy *once each. We define*

$$\mathbf{Adv}^{\mathsf{OT\text{-}SUF\text{-}CMA}}_{\mathrm{MAC},\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{OT\text{-}SUF\text{-}CMA}}_{\mathrm{MAC},\mathcal{A}}(\lambda) = 1] \ .$$

Definition 4 (One-Time Strong Unforgeability MAC Security). *We say that a MAC scheme is OT-SUF-CMA secure if there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all PPT adversaries $\mathcal{A}$ interacting according to the experiment $\mathsf{Exp}^{\mathsf{OT\text{-}SUF\text{-}CMA}}_{\mathrm{MAC},\mathcal{A}}$ it holds that*

$$\mathbf{Adv}^{\mathsf{OT\text{-}SUF\text{-}CMA}}_{\mathrm{MAC},\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda) \quad .$$

As discussed in Section 1, unforgeability under key collision attacks are intrinsically different from Known-Key, Chosen-Key, and Related-Key attacks. Since the MAC key is provided to the adversary following generation of a MAC tag, EUF-KCA security is naturally a one-time security game.

Definition 5. *Let $\mathcal{A}$ be a PPT adversarial algorithm against the MAC. The existential unforgeability under key collision attacks (EUF-KCA) experiment for MAC, $\mathsf{Exp}^{\mathsf{EUF\text{-}KCA}}_{\mathrm{MAC},\mathcal{A}}$ is given in Fig. 3. We define*

$$\mathbf{Adv}^{\mathsf{EUF\text{-}KCA}}_{\mathrm{MAC},\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{EUF\text{-}KCA}}_{\mathrm{MAC},\mathcal{A}}(\lambda) = 1] \quad .$$

Definition 6 (Existential Unforgeability under Key Collision Attacks MAC Security). *We say that a MAC scheme is EUF-KCA secure if there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all PPT adversaries $\mathcal{A}$ interacting according to the experiment $\mathsf{Exp}^{\mathsf{EUF\text{-}KCA}}_{\mathrm{MAC},\mathcal{A}}$ it holds that*

$$\mathbf{Adv}^{\mathsf{EUF\text{-}KCA}}_{\mathrm{MAC},\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda) \quad .$$

Since it addresses key guessing, it is perhaps natural to ask how EUF-KCA differs from a brute-force key search, and why it is necessary. Unlike in a brute-force search, the adversary in an unforgeability key collision attack actually has

$\underline{\mathsf{Exp}^{\mathsf{EUF\text{-}KCA}}_{\mathrm{MAC},\mathcal{A}}(\lambda)}$:

1: $K \xleftarrow{\$} \mathrm{Kgn}(1^\lambda)$
2: $\mathtt{K} \leftarrow \bot, \mathtt{t} \leftarrow \bot$
3: $\mathcal{A}^{\mathsf{MAC}(\cdot),\mathsf{MAC.Vfy}(\cdot)}()$
4: **return** `phase`

$\underline{\mathsf{MAC}(m)}$:

1: **if** $(\mathtt{K}, \mathtt{t}) \neq (\bot, \bot)$ **then**
2: **return** $\bot$
3: $t \leftarrow \mathrm{MAC}(K, m)$
4: $\mathtt{K} \leftarrow K, \mathtt{t} \leftarrow t$
5: **return** (K, t)

$\underline{\mathsf{MAC.Vfy}(k, m)}$:

1: $v \leftarrow \mathrm{Vfy}(k, m, \mathtt{t})$
2: **if** $(v = 1) \wedge (k \neq \mathtt{K})$ **then**
3: `phase` $\leftarrow 1$
4: **return** `phase` from experiment
5: **return** v

Figure 3. Security experiment for EUF-KCA of a message authentication code algorithm $\mathrm{MAC} = (\mathrm{Kgn}, \mathrm{MAC}, \mathrm{Vfy})$ and adversary $\mathcal{A}$.

a valid key at its disposal. It also possesses a complete message triple (K, m, t), with the goal of finding an alternative key. Improper handling of keys in a MAC algorithm could enable an adversary to find a related key which satisfies the message-tag pair.

3 3-PUMA Model for Simple Output Devices

Introducing a third-party user interface to a two-party protocol creates a unique modeling challenge. Not only must we consider the communication channel between devices (wired or wireless), but also the third-party user interaction and channels between the user and devices. The 3-PUMA model captures protocols for simple-output devices (i.e. such as a red and green lights or other such success/failure indication mechanisms [23]), and is suitable for protocols which do not possess long-term keys. We leverage partnering via session IDs to capture the agreement between authenticating devices instead of handling matching of all three-party transcripts.

ISO/IEC 9798–6:2010 does not specify how each party obtains the other's identity, but requires that identities are known prior to the start of the mechanism. In this model, we let the adversary choose partner identities.

3.1 Protocol Participants

Each device possesses a *simple output* interface, e.g. binary success/failure indication, as well as a *standard input* interface which allows a user to input a bit-string into the device. A participant in a 3-PUMA protocol is either a device $I \in \mathcal{ID}$ or a user U. As there is only one user interface, we do not model multiple users.[1] The set of all participants is the union $\mathcal{ID} \cup \{U\}$. We refer to elements of $\mathcal{ID}$ alternatively as *devices* or *identities*.

[1] Naturally, it is possible that two users each possess one of the devices participating in the authentication protocol. However, by requiring the user to behave honestly and reliably perform protocol steps, there is no conceptual difference between multiple users and a single user in possession of all devices.

We model participants via *sessions*, such that π_i^P is the i-th session at P. There may be multiple and simultaneous sessions at each participant, e.g. the data D may be transferred and shared between devices before a protocol run begins.

Devices. Each device $I \in \mathcal{ID}$ is modeled via session oracles, where each session maintains a list of the following variables:

- $K \in \mathcal{K}$: a variable for storing an ephemeral key, where $\mathcal{K}$ is the key space of the protocol.
- $D \in \{0,1\}^*$: a variable for storing the possession data represented by a finite binary string, as defined in Section 2.1.
- $\mathsf{role} \in \{\mathsf{initiator}, \mathsf{responder}\}$: a variable indicating the role of I in the session.
- $\mathtt{pid} \in \mathcal{ID}/\{I\}$: a variable for storing the partner identity for the session.
- $\delta \in \{\mathsf{accept}, \mathsf{reject}, *\}$: a variable indicating if the session accepts, rejects, or has not yet reached a decision.
- $\mathtt{sid}$: a variable for storing the session ID.

The internal state of each session oracle at identity I is initialized to $(K, D, \mathsf{role}, \mathtt{pid}, \mathtt{sid}) = (\emptyset, *, \emptyset, \emptyset, \emptyset)$, where $V = \emptyset$ indicates that the variable V is undefined and $*$ indicates that the variable value may or may not be defined. In the case of out-of-band exchange of D, D is initialized to the agreed value; otherwise, D is initialized to $\emptyset$. In general, the possession data D may be exchanged out-of-band or during the protocol, and it is the explicit goal of the 3-PUMA protocol to authenticate possession of D at two sessions π_s^I and $\pi_{\bar{s}}^{I'}$, versus mutually authenticating parties I and I'. Rejection of the protocol run may occur at any time, but acceptance does not usually occur until the protocol is complete. We disallow $\mathtt{pid}_I = I$, such that devices do not authenticate themselves. Since a 3-PUMA protocol aims to authenticate possession of D, we require that D be included in the $\mathtt{sid}$, in addition to any other protocol elements.

User. U is modeled via session oracles, where each session maintains the following variables:

- $\mathtt{init} \in \mathcal{ID}$: a variable indicating the initiating identity.
- $\mathtt{resp} \in \mathcal{ID}$: a variable indicating the responding identity.

The internal state of each session oracle at U is initialized to $(\mathtt{init}, \mathtt{resp}) = (\emptyset, \emptyset)$.

For devices we use a notion of partnering based on session IDs ($\mathtt{sid}$). Note that Device-to-Device (DtD) messages occur on a different channel than User-to-Device (UtD) messages, and partnering is defined on the DtD channel only.

Definition 7 (Matching Session ID). *We say that identities I and I' possess* matching session IDs *if* $\mathtt{sid}_I = \mathtt{sid}_{I'}$.

Remark 1. In practice for analysis of Mechanism 7a, we will use $\mathtt{sid} = (D, R, mac_A, mac_B, K_A, K_B)$, i.e. the transcript shared by device protocol participants I and I' inclusive of the pre-exchanged data string D, and R. Selection of a session ID is non-trivial, especially in a three-party case. We emphasize that the selection above is made using the full transcript between I and I', inclusive

of out-of-band data D, but also includes elements *sent* to respective identities by the user, on the UtD channel. Results of the MAC verifications are sent to the user, and therefore are not mutually held by I and I', as are the `ready` messages, but the user's selection R is sent from the user to both devices and is thus mutually held by the identities. While the confirmation message `OK` is also sent by the user, as the final protocol message a recipient cannot be guaranteed that the protocol partner has also received the message (i.e. it may be dropped), and therefore we only include message flows up until the confirmation message. In general, we expect the user to be an active protocol participant generating such confirmations. Notably, this selection of session ID can be observed from the end-goal. The identities authenticate possession of D, but no authentication is considered with regards to user interaction. Thus, it is natural to include messages between identities, as well as messages *received* by both identities from the user.

Definition 8 (Partnering Device to Device). *We say that two sessions* $\pi_s^I, \pi_{\tilde{s}}^{I'}$*, for* $I, I' \in \mathcal{ID}$*, are* partnered *if they both accept, and possess, respectively,* $(\mathtt{pid}_I, \mathtt{sid}_I)$ *and* $(\mathtt{pid}_{I'}, \mathtt{sid}_{I'})$*, where* $\mathtt{pid}_I = I'$*,* $\mathtt{pid}_{I'} = I$*, and* $\mathtt{sid}_I = \mathtt{sid}_{I'}$*.*

Note that the above definition is Device-to-Device only. We do not define a notion of partnering between the user and devices as there is a single user for all devices. Note that D is a required element of `sid`, therefore demanding agreement between identities on D.

3.2 Adversarial Model

We consider a probabilistic polynomial-time (PPT) adversarial algorithm $\mathcal{A}$ against authentication. We define the following abilities of $\mathcal{A}$ in the 3-party possession user-mediated authentication experiment $\mathsf{Exp}_{\mathcal{A}}^{\text{3-PUMA}}$, including allowed queries.

Device-to-Device (DtD). For messages between participants I and I', such that $I, I' \in \mathcal{ID}$, the adversary is allowed to read, modify, replay, and delete messages.

User-to-Device (UtD). For messages sent between identities $I \in \mathcal{ID}$ and the user U, the adversary may not modify a message's sender/recipient. We present three variants of adversarial behavior allowed on the UtD channel, and use the notation 3-PUMA$_i$ for the i-th variant:

1. Before the first DtD message, the adversary is allowed to read, modify, replay, or delete UtD messages.
   Once the first DtD message is sent, the adversary is allowed to read, replay, and delete messages, but may not modify UtD messages.
2. The adversary is allowed to read, replay, and delete messages, but may not modify UtD messages.
3. Before the first DtD message, the adversary is allowed to replay or delete messages sent from a user to a device, but may not read or modify messages.
   The adversary is allowed to replay, delete or read messages sent from a device to a user, but may not modify messages.
   Once the first DtD message is sent, the adversary is allowed to read, replay, and delete UtD messages, but may not modify messages.

We model the user as an honest, benign, and unauthenticated third party, i.e. which behaves according to the protocol specification and may not be compromised. However, we capture a CCA1 variant (3-PUMA$_1$) by allowing adversary access to a device before the first DtD message. The 3-PUMA$_1$ model may be suitable for protocols that expect the user to behave as a confirmation source (e.g. assuring that both devices output a confirmation message), but where the user does not generate or control any secrets. 3-PUMA$_2$ is likely the most typical user-mediated variant, capturing shoulder surfing attacks, but forbidding active modification of messages. 3-PUMA$_3$ provides the most restricted view, where the adversary may not even view messages being sent in the initial phase. It is expected that user-interaction involving secrets (keys, random nonces, etc.) occurs before communication on the DtD channel, thus we allow the adversary to read messages that do not involve protocol secrets.

We distinguish between an adversary's ability to read user and device messages in variant 3 based on typically required user behavior. A user may be asked to conceal input of secret information (e.g. pin codes into a card reader, or R in the case of Mechanism 7a), but is rarely required to conceal information displayed by a device or machine.

Remark 2. Note that this adversarial behavior is as expected for a UtD channel, due to the physicality of the channel (i.e. user interface). While we may consider message deletion, re-routing/modification of the sender or receiver of message is not realistic. We distinguish between the memory/card on the device (used in the generation of K_I) and device input/output mechanisms. Excluding compromise of the user or device card, we have the following channel adversarial control scenarios:

- An adversary can observe inputs to devices (i.e. shoulder surfing attacks).
- An adversary can observer outputs from a device to a user (e.g. simple output screen or red/green lights).
- An adversary may delete or replay messages (e.g. faulty hardware in a device).
- An adversary can modify messages output from a device to a user (implies control of e.g. a display or red/green lights).
- An adversary can modify messages input into a device (implies control of e.g. touchpad or touchscreen).

Clearly the last two items are only viable attack vectors in certain sophisticated devices where control of e.g. display output does not imply compromise of the device secret key. This model does not consider a manipulated or dishonest user; more particularly, the user does not relinquish control of the devices throughout the protocol run.

Mechanism 7a specifies that partner identities are already known to respective devices before a protocol run, but does not specify how they are distributed. To enable maximum flexibility in modeling, we allow the adversary to choose partner identities when initiating devices via the following queries.

Queries. The adversary may use the following queries:

– $\mathsf{SendDevice}(\pi_s^I, m)$. Using this query, the adversary sends a message m to a session oracle of his choice, where π_s^I is an oracle for session s at a participant $I \in \mathcal{ID}$. The message is processed according to the protocol and any response is returned to the adversary.
If a session oracle π_s^I, where $I \in \mathcal{ID}$, receives m as a first message, then the oracle checks if m consists of a special initiation message ($m = (\texttt{init}, I')$), for $I' \in \mathcal{ID}$, to which it responds by setting $\texttt{pid} = I'$ and outputting the first protocol message. Else it outputs $\perp$.
If at any point a session oracle π_s^I, where $I \in \mathcal{ID}$, receives a message m from U during a protocol run, such that m consists of a special role-setting message $m = \texttt{start}$ and π_s^I has not received a message from another identity $I' \in \mathcal{ID}$, then π_s^I sets $\mathsf{role} = \mathsf{initiator}$ and responds according to the initiator role in the protocol. Else, if π_s^I receives a message from another identity $I' \in \mathcal{ID}$ according to the protocol without having received such a message $m = \texttt{start}$ from U, it sets $\mathsf{role} = \mathsf{responder}$ and responds according to the responder role in the protocol.
– $\mathsf{SendUser}(\pi_s^U, m)$. Using this query, the adversary sends a message m to a session oracle of his choice, where π_s^U is an oracle for session s at user U. The message is processed according to the protocol and any response is returned to the adversary.
If a session oracle π_s^U, receives m as a first message, then the oracle checks of m consists of a special initiation message ($m = (\texttt{init}, (I, I'))$), for $I, I' \in \mathcal{ID}$, to which it responds by setting $\texttt{init} = I$ and $\texttt{resp} = I'$. Else it outputs $\perp$.
– $\mathsf{RevealEphKey}(\pi_s^I)$. This query returns the ephemeral key K_{I_s} of the s-th session for the identity $I \in \mathcal{ID}$. If $K_{I_s} = \emptyset$, $\mathsf{RevealEphKey}$ returns $\perp$.

In brief, the above queries are used by the adversary to send messages to devices ($\mathsf{SendDevice}$), to the user ($\mathsf{SendUser}$), and to acquire the ephemeral key used at a device ($\mathsf{RevealEphKey}$). The first message used in a $\mathsf{SendUser}$ query *must* consist of an initiation message with an identity I; this sets the initiating identity for the protocol. For a user-mediated protocol, the user selects the device that will start the protocol run. We specifically allow the adversary to choose the initiating device using this message.

Definition 9 (Freshness). *A session oracle π_s^I for an identity $I \in \mathcal{ID}$ is called* fresh *unless*

– *a* $\mathsf{RevealEphKey}$ *query on π_s^I occurs before the last DtD message is sent/received by π_s^I, or*
– *a* $\mathsf{RevealEphKey}$ *query on $\pi_{\bar{s}}^{I'}$ occurs before the last DtD message is sent/received by $\pi_{\bar{s}}^{I'}$, where $\pi_{\bar{s}}^{I'}$ is the partner of π_s^I.*

3.3 Security

Here we define authentication security, building on previous authentication security definitions [8], with inclusion of ephemeral key reveal and session IDs per the eCK model [29].

We rely on session IDs due to the parallel-channel structure of the protocol. Even in an honest protocol run, participants are not expected to possess matching transcripts at the end of the protocol, due to the three-party interaction.

Thus session IDs can be used to define the expected agreement between devices. If matching session transcripts were used, it would be necessary to extract and distinguish between the transcripts in the DtD and UtD channels within the protocol. In practice, the session ID may be the DtD transcript in addition to any critical information provided by the user U.

Definition 10 (3-PUMA Experiment). *Let $\mathcal{A}$ be a PPT adversarial algorithm against 3-party possession user-mediated authentication, interacting with a challenger in the experiment $\mathsf{Exp}_{\mathcal{A}}^{\text{3-PUMA}}$ via the queries defined above. We say that the challenger outputs 1, denoted $\mathsf{Exp}_{\mathcal{A}}^{\text{3-PUMA}}(\lambda) = 1$, if either of the following conditions hold:*

1. Failure of (matching `sid` $\rightarrow$ acceptance).
   - *Oracles π_s^I and $\pi_{\bar{s}}^{I'}$ have matching* `sid` *and*
   - *either π_s^I or $\pi_{\bar{s}}^{I'}$ does not accept.*
2. Failure of (acceptance $\rightarrow$ matching `sid`).
   - *There exists a fresh oracle π_s^I which has accepted and*
   - *there is no partner oracle $\pi_{\bar{s}}^{I'}$ which is fresh.*

Otherwise the experiment outputs a random bit. We define the advantage of the adversary $\mathcal{A}$ in the experiment $\mathsf{Exp}_{\mathcal{A}}^{\text{3-PUMA}}(\lambda)$ as

$$\mathbf{Adv}_{\mathcal{A}}^{\text{3-PUMA}}(\lambda) := \Pr[\mathsf{Exp}_{\mathcal{A}}^{\text{3-PUMA}}(\lambda) = 1] \quad .$$

Note that, as an adversary cannot corrupt the user U, modification of messages on the UtD channel require a $\mathsf{RevealEphKey}$ query on the device. The above definition therefore allows for maximum flexibility for the adversary.

Definition 11 (Security of 3-PUMA). *We say that a 3-party possession user-mediated authentication protocol is* secure *if there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all PPT adversaries $\mathcal{A}$ interacting according to the experiment $\mathsf{Exp}_{\mathcal{A}}^{\text{3-PUMA}}(\lambda)$, it holds that*

$$\mathbf{Adv}_{\mathcal{A}}^{\text{3-PUMA}}(\lambda) \leq \mathsf{negl}(\lambda) \quad .$$

4 Security Analysis

As stated in Section 3.2, we define $\texttt{sid} = (D, R, mac_A, mac_B, K_A, K_B)$ for Mechanism 7a. Given the protocol, as presented in Fig. 1, we now consider security with respect to the MAC requirements presented in Section 2.2. As ISO/IEC 9798–6:2010 Mechanism 7a requires strict privacy with regards to R, we use the 3-PUMA$_3$ model. In contrast, ISO/IEC 9798–6:2010 Mechanisms 1–6 do not require the same form of secrecy; 3-PUMA$_2$ would be a more suitable variant in such cases.

Theorem 1 (Security of ISO/IEC 9798–6:2010 Mechanism 7a). *Let* ISO *be the Mechanism 7a protocol and let $\mathcal{A}$ be a PPT adversarial algorithm against the* 3-PUMA$_3$*. Let q be a polynomial bound on the number of queries allowed to $\mathcal{A}$ and let $p = |\mathcal{ID}|$. Then we can construct adversaries $\mathcal{B}_0$ and $\mathcal{B}_1$*

against the OT-SUF-CMA and EUF-KCA security of the MAC, respectively, such that

$$\begin{aligned}\mathbf{Adv}^{\text{3-PUMA}_3}_{\text{ISO},\mathcal{A}}(\lambda) \leq & (2p^2+1)\cdot\mathbf{Adv}^{\mathsf{OT\text{-}SUF\text{-}CMA}}_{\text{MAC},\mathcal{B}_0}(\lambda) + 2p^2\cdot\mathbf{Adv}^{\mathsf{EUF\text{-}KCA}}_{\text{MAC},\mathcal{B}_1}(\lambda) \\ & + q^2/2^n \quad .\end{aligned}$$

where n is the prescribed bit-length of R.

Proof. To prove security it is necessary to address both cases in Definition 10. Checking the first case is trivial. Therefore this proof will focus on the second case.

In this proof we will follow a series of game hops between an attacker $\mathcal{A}$ and a challenger, where the adversarial advantage in Game i is denoted $\mathbf{Adv}_i$. Let q be a polynomial bound on the number of queries allowed to $\mathcal{A}$. The challenger generates a set of identities for potential protocol participants $\mathcal{ID}$, where $p = |\mathcal{ID}|$.

Game 0. This is the same as the original security experiment, thus

$$\mathbf{Adv}_0 = \mathbf{Adv}^{\text{3-PUMA}_3}_{\text{ISO},\mathcal{A}}(\lambda) \quad .$$

Game 1. This game is identical to Game 0 except for the addition of an abort condition. We raise the event `abort` and abort the experiment, outputting a random bit, if there ever exists two session oracles which generate the same MAC key. Thus,

$$\mathbf{Adv}_1 \geq \mathbf{Adv}_0 - \Pr[\texttt{abort}] \quad .$$

By a straightforward reduction to the security of the MAC algorithm, we can construct an adversary $\mathcal{B}_0$ against the OT-SUF-CMA security of the MAC. $\mathcal{B}_0$ possesses a MAC oracle which it uses to compute message authentication codes for identity A, per the $\mathsf{Exp}^{\mathsf{OT\text{-}SUF\text{-}CMA}}_{\text{MAC}}$ experiment, run on a random key $K_A \xleftarrow{\$} \text{Kgn}()$, and a $\mathsf{MAC.Vfy}$ oracle. When identity A sends a message (A, D, R), $\mathcal{B}_0$ calls $\mathsf{MAC}(A, D, R)$ and returns $t = mac_A$. $\mathcal{B}_0$ then generates an additional $p-1$ keys $K_i \xleftarrow{\$} \text{Kgn}()$ and computes $\text{Vfy}(K_i, (A, D, R), mac_A)$ for each $i \in [p-l]$, using the Vfy algorithm. If there exists i such that $1 \leftarrow \text{Vfy}(K_i, (A, D, R), mac_A)$, then $\mathcal{B}_0$ can trivially compute $mac_{win} \xleftarrow{\$} \text{MAC}(K_i, (A, D, R))$, a new MAC on A's message. Since $mac_A = mac_{win}$, the new MAC will verify correctly in $\mathsf{MAC.Vfy}$, despite never being generated by MAC (as MAC is probabilistic), and therefore $\mathcal{B}_0$ wins the $\mathsf{Exp}^{\mathsf{OT\text{-}SUF\text{-}CMA}}_{\text{MAC}}$ experiment. Thus we have,

$$\Pr[\texttt{abort}] \leq \mathbf{Adv}^{\mathsf{OT\text{-}SUF\text{-}CMA}}_{\text{MAC},\mathcal{B}_0}(\lambda) \quad ,$$

and

$$\mathbf{Adv}_1 \geq \mathbf{Adv}_0 - \mathbf{Adv}^{\mathsf{OT\text{-}SUF\text{-}CMA}}_{\text{MAC},\mathcal{B}}(\lambda) \quad .$$

Game 2. This game is identical to Game 1 except for the addition of an abort condition. We raise the event `abort` and abort the experiment, outputting a random bit, if there ever exists two session pairs for which the user generates the same R value. Thus,

$$\mathbf{Adv}_2 \geq \mathbf{Adv}_1 - \Pr[\texttt{abort}] \quad ,$$

and

$$\mathbf{Adv}_2 \geq \mathbf{Adv}_1 - q^2/2^n \quad .$$

Game 3. This game is identical to Game 2 except that we guess the pair of authenticating devices uniformly at random and abort if $\mathcal{A}$ does not try to win the authentication experiment against the guessed pair. From the birthday paradox we have,

$$\mathbf{Adv}_3 \geq 1/p^2 \cdot \mathbf{Adv}_2 \quad .$$

The challenger starts the protocol run with guessed identities A and B. We replace the generated ephemeral keys with random keys which will be used in MAC oracles. If the adversary calls a $\mathsf{RevealEphKey}$ query on either A or B, the challenger aborts.

In order to win, the adversary must get A or B to accept, such the identity does not possess a matching $\mathtt{sid} = (D, R, mac_A, mac_B, K_A, K_B)$ with the other.

Case 1. We assume that the adversary gets A to accept incorrectly. Thus, the adversary must produce a different MAC tag or key for which the MAC algorithm to verifies correctly. We can therefore construct an efficient attacker $\mathcal{B}_0$ against the OT-SUF-CMA security of the MAC, or an efficient attacker $\mathcal{B}_1$ against the EUF-KCA security of the MAC, such that

$$\mathbf{Adv}_3 \leq \mathbf{Adv}^{\mathsf{OT\text{-}SUF\text{-}CMA}}_{\mathrm{MAC},\mathcal{B}_0}(\lambda) + \mathbf{Adv}^{\mathsf{EUF\text{-}KCA}}_{\mathrm{MAC},\mathcal{B}_1}(\lambda) \quad .$$

Case 2. We assume that the adversary gets B to accept incorrectly. This case follows similarly to Case 1. □

Remark 3. The reliance of Mechanism 7a security on the EUF-KCA security of the MAC presents a significant issue. EUF-KCA security is not well understood and is non-standard. Consequently, it is unknown whether or not basic MAC primitives, such as are recommended for use in Mechanism 7a, satisfy this security requirement.

One can perhaps ask why the EUF-KCA security game is of interest if we allow the 3-PUMA attacker to reveal keys part-way through the protocol. Such an action mixes expected protocol behavior with adversarial abilities. Moreover, it is important to note the difference between the security goals of the protocol and those of the MAC algorithm. It may not be the case that all 3-PUMA protocols send ephemeral keys in the clear mid-protocol. The MAC in ISO/IEC 9798–6:2010 Mechanism 7a is expected to be secure even when the key is revealed before MAC verification takes place. Intrinsically, the MAC should thus provide the security guarantees against a key collision or related key generation based on a known key.

As stated in Section 3.2, the analysis of ISO/IEC 9798–6:2010 Mechanism 7a assumes an honest and benign user U. However, this may not adequately represent real-world scenarios. For example, even if the MAC verification fails on one device during the protocol run shown in Fig. 1, a distracted user or physically present adversary may enter an $\mathtt{OK}$ acceptance message on each device. Consequently, satisfaction of the model and the honest and benign nature of the user, cannot be over-emphasized for achieving the security analysis result.

5 Conclusion

User mediated protocols are used every day real-world scenarios, including IoT device commissioning, decommissioning, and pairing protocols. Modern users even accept such protocols as part of their daily life, such as with mobile phone pairing with car speaker systems. Despite this, user modeling remains largely overlooked in protocol analysis, particular within computational analysis approaches. As a result subtle security weaknesses, such as the EUF-KCA assumption of ISO/IEC 9798–6:2010 Mechanism 7a, have gone unnoticed. This work initiates the study of computational modeling of user mediated protocols as well as the computational analysis of other non-standard protocol interactions.

Bibliography

[1] Ross J. Anderson and Roger M. Needham. Robustness principles for public key protocols. In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 236–247. Springer, Heidelberg, August 1995.

[2] Elena Andreeva, Andrey Bogdanov, and Bart Mennink. Towards understanding the known-key security of block ciphers. In Shiho Moriai, editor, *FSE 2013*, volume 8424 of *LNCS*, pages 348–366. Springer, Heidelberg, March 2014.

[3] Benny Applebaum, Danny Harnik, and Yuval Ishai. Semantic security under related-key attacks and applications. In Bernard Chazelle, editor, *ICS 2011*, pages 45–60. Tsinghua University Press, January 2011.

[4] David Basin, Cas Cremers, and Simon Meier. Provably repairing the iso/iec 9798 standard for entity authentication. In *Principles of Security and Trust*, pages 129–148. Springer Berlin Heidelberg, 2012.

[5] Mihir Bellare and David Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 666–684. Springer, Heidelberg, August 2010.

[6] Mihir Bellare, David Cash, and Rachel Miller. Cryptography secure against related-key attacks and tampering. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 486–503. Springer, Heidelberg, December 2011.

[7] Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 491–506. Springer, Heidelberg, May 2003.

[8] Mihir Bellare and Phillip Rogaway. Entity Authentication and Key Distribution. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 232–249. Springer, Heidelberg, August 1993.

[9] Mihir Bellare and Phillip Rogaway. Provably Secure Session Key Distribution: The Three Party Case. In *27th ACM STOC*, pages 57–66. ACM Press, May / June 1995.

[10] Rishiraj Bhattacharyya and Arnab Roy. Secure message authentication against related-key attack. In Shiho Moriai, editor, *FSE 2013*, volume 8424 of *LNCS*, pages 305–324. Springer, Heidelberg, March 2014.

[11] Eli Biham. New types of cryptanalytic attacks using related keys. *Journal of Cryptology*, 7(4):229–246, 1994.

[12] Eli Biham, Orr Dunkelman, and Nathan Keller. A unified approach to related-key attacks. In Kaisa Nyberg, editor, *FSE 2008*, volume 5086 of *LNCS*, pages 73–96. Springer, Heidelberg, February 2008.

[13] Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolic. Distinguisher and related-key attack on the full AES-256. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 231–249. Springer, Heidelberg, August 2009.

[14] Céline Blondeau, Thomas Peyrin, and Lei Wang. Known-key distinguisher on full PRESENT. In Rosario Gennaro and Matthew J. B. Robshaw,

editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 455–474. Springer, Heidelberg, August 2015.

[15] Florian Böhl, Gareth T. Davies, and Dennis Hofheinz. Encryption schemes secure under related-key and key-dependent message attacks. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 483–500. Springer, Heidelberg, March 2014.

[16] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 480–499. Springer, Heidelberg, August 2014.

[17] Richard Chang and Vitaly Shmatikov. Formal Analysis of Authentication in Bluetooth Device Pairing. `https://www.cs.cornell.edu/~shmat/shmat_fcs07.pdf`, 2018.

[18] Véronique Cortier, Steve Kremer, and Bogdan Warinschi. A survey of symbolic methods in computational analysis of cryptographic systems. *Journal of Automated Reasoning*, 46(3):225–259, Apr 2011.

[19] Stephanie Delaune, Steve Kremer, and Ludovic Robin. Formal verification of protocols based on short authenticated strings. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 130–143, Aug 2017.

[20] Le Dong, Wenling Wu, Shuang Wu, and Jian Zou. Known-key distinguisher on round-reduced 3D block cipher. In Souhwan Jung and Moti Yung, editors, *WISA 11*, volume 7115 of *LNCS*, pages 55–69. Springer, Heidelberg, August 2012.

[21] Christian Gehrmann and Kaisa Nyberg. *Security in Personal Area Networks*.

[22] Britta Hale and Colin Boyd. Computationally Analyzing the ISO 9798-2.4 Authentication Protocol. In *Security Standardisation Research, SSR 2014. Proceedings*, pages 236–255, 2014.

[23] ISO. Information technology – Security techniques – Entity Authentication – Part 6: Mechanisms using manual data transfer. ISO ISO/IEC 9798-6:2010, International Organization for Standardization, Geneva, Switzerland, 2010.

[24] ISO. Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher. ISO ISO/IEC 9797-1:2011, International Organization for Standardization, Geneva, Switzerland, 2011.

[25] ISO. Information technology – Security techniques – Message Authentication Codes (MACs) – Part 2: Mechanisms using a dedicated hash-function. ISO ISO/IEC 9797-2:2011, International Organization for Standardization, Geneva, Switzerland, 2011.

[26] Mike Just and Serge Vaudenay. Authenticated Multi-Party Key Agreement. In *Advances in Cryptology — ASIACRYPT '96*, pages 36–49, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[27] Lars R. Knudsen and Vincent Rijmen. Known-key distinguishers for some block ciphers. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 315–324. Springer, Heidelberg, December 2007.

[28] Tadayoshi Kohno. Related-key and key-collision attacks against RMAC. Cryptology ePrint Archive, Report 2002/159, 2002. `http://eprint.iacr.org/2002/159`.

[29] B. LaMacchia, K. Lauter, and A. Mityagin. Stronger Security of Authenticated Key Exchange. In *ProvSec 2007*, pages 1–16. LNCS vol. 4784, Springer, 2007.

[30] Mario Lamberger, Florian Mendel, Christian Rechberger, Vincent Rijmen, and Martin Schläffer. Rebound distinguishers: Results on the full Whirlpool compression function. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 126–143. Springer, Heidelberg, December 2009.

[31] Gavin Lowe. A Hierarchy of Authentication Specifications. In *Proceedings of the 10th IEEE Workshop on Computer Security Foundations*, CSFW '97, pages 31–43. IEEE Computer Society, 1997.

[32] Stefan Lucks. Ciphers secure against related-key attacks. In Bimal K. Roy and Willi Meier, editors, *FSE 2004*, volume 3017 of *LNCS*, pages 359–370. Springer, Heidelberg, February 2004.

[33] Bart Mennink and Bart Preneel. On the impact of known-key attacks on hash functions. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 59–84. Springer, Heidelberg, November / December 2015.

[34] Phuong Ha Nguyen, Matthew J. B. Robshaw, and Huaxiong Wang. On related-key attacks and KASUMI: The case of A5/3. In Daniel J. Bernstein and Sanjit Chatterjee, editors, *INDOCRYPT 2011*, volume 7107 of *LNCS*, pages 146–159. Springer, Heidelberg, December 2011.

[35] Trung Nguyen and Jean Leneutre. Formal Analysis of Secure Device Pairing Protocols. *2014 IEEE 13th International Symposium on Network Computing and Applications*, pages 291–295, 2014.

[36] Ivica Nikolic, Josef Pieprzyk, Przemyslaw Sokolowski, and Ron Steinfeld. Known and chosen key differential distinguishers for block ciphers. In Kyung Hyune Rhee and DaeHun Nyang, editors, *ICISC 10*, volume 6829 of *LNCS*, pages 29–48. Springer, Heidelberg, December 2011.

[37] Tomas Rosa. Key-collisions in (EC)DSA: Attacking non-repudiation. Cryptology ePrint Archive, Report 2002/129, 2002. `http://eprint.iacr.org/2002/129`.

[38] Yu Sasaki. Known-key attacks on Rijndael with large blocks and strengthening ShiftRow parameter. In Isao Echizen, Noboru Kunihiro, and Ryôichi Sasaki, editors, *IWSEC 10*, volume 6434 of *LNCS*, pages 301–315. Springer, Heidelberg, November 2010.

[39] Yu Sasaki, Sareh Emami, Deukjo Hong, and Ashish Kumar. Improved known-key distinguishers on Feistel-SP ciphers and application to Camellia. In Willy Susilo, Yi Mu, and Jennifer Seberry, editors, *ACISP 12*, volume 7372 of *LNCS*, pages 87–100. Springer, Heidelberg, July 2012.

[40] Yu Sasaki and Kan Yasuda. Known-key distinguishers on 11-round Feistel and collision attacks on its hashing modes. In Antoine Joux, editor, *FSE 2011*, volume 6733 of *LNCS*, pages 397–415. Springer, Heidelberg, February 2011.

[41] Sheikh Ziauddin and Bruno Martin. Formal Analysis of ISO/IEC 9798-2 Authentication Standard Using AVISPA, 07 2013.

A Appendix: Strong Unforgeability under Key Forgery Attacks and Applications

Section 2.2 introduces EUF-KCA security for a MAC, which is particularly necessary in Mechanism 7a. However, other key collision variants could prove useful in other environments. Here we describe the security experiment for strong unforgeability under a key forgery attack (SUF-KCA). EUF-KCA allows an adversary to win only if it produces a valid key corresponding to a given MAC tag – regardless of whether or not a new message is used. In comparison, SUF-KCA allows an adversary to win if it produces either a new key or a new message corresponding to a given authentication tag. Table 1 shows this comparison. The security experiments for SUF-KCA and EUF-CMA are presented in Fig. 4 and Fig. 5, respectively.

From Table 1 it appears that SUF-KCA is a combination of EUF-CMA and EUF-KCA. However, note that this is not the case. To win EUF-CMA, an adversary may manipulate the message tag in addition to the message, although the winning requirement is on the message only. Therefore, an adversary that can forge a new message-tag pair (m', t'), which validates correctly and where both m' and t' are new, may win the EUF-CMA experiment, but does not win the SUF-KCA experiment as the tag differs from those previously output by the MAC oracle.

$\underline{\mathsf{Exp}^{\text{SUF-KCA}}_{\text{MAC},\mathcal{A}}(\lambda)}$:
1: $K \xleftarrow{\$} \text{Kgn}()$
2: $\mathtt{K} \leftarrow \bot, \mathtt{m} \leftarrow \bot, \mathtt{t} \leftarrow \bot$
3: $\mathcal{A}^{\mathsf{MAC}(\cdot),\mathsf{MAC.Vfy}(\cdot)}()$
4: **return** `phase`

$\underline{\mathsf{MAC}(m)}$:
1: **if** $(\mathtt{K}, \mathtt{t}) \neq (\bot, \bot)$ **then**
2: **return** $\bot$
3: $t \leftarrow \text{MAC}(K, m)$
4: $\mathtt{K} \leftarrow K, \mathtt{m} \leftarrow m, \mathtt{t} \leftarrow t$
5: **return** (K, t)

$\underline{\mathsf{MAC.Vfy}(K, m)}$:
1: $v \leftarrow \text{Vfy}(K, m, \mathtt{t})$
2: **if** $(v = 1) \wedge ((K, m) \neq (\mathtt{K}, \mathtt{m})$ **then**
3: `phase` $\leftarrow 1$
4: **else**
5: `phase` $\leftarrow 0$
6: **return** `phase` from experiment
7: **return** v

Figure 4. SUF-KCA security experiment for a message authentication code algorithm $\text{MAC} = (\text{Kgn}, \text{MAC}, \text{Vfy})$ and adversary $\mathcal{A}$.

$\underline{\mathsf{Exp}^{\mathsf{EUF\text{-}CMA}}_{\mathrm{MAC},\mathcal{A}}(\lambda)}$:

1: $K \xleftarrow{\$} \mathrm{Kgn}()$
2: $S \leftarrow \emptyset$
3: $\mathcal{A}^{\mathsf{MAC}(\cdot),\mathsf{MAC.Vfy}(\cdot)}()$
4: **return** `phase`

$\underline{\mathsf{MAC}(m)}$:

1: $t \leftarrow \mathrm{MAC}(K, m)$
2: $S \leftarrow S \cup \{m\}$
3: **return** t

$\underline{\mathsf{MAC.Vfy}(m, t)}$:

1: $v \leftarrow \mathrm{Vfy}(K, m, t)$
2: **if** $(v = 1) \wedge (m \notin S)$ **then**
3: `phase` $\leftarrow 1$
4: **return** `phase` from experiment
5: **return** v

Figure 5. EUF-CMA security experiment for a message authentication code algorithm $\mathrm{MAC} = (\mathrm{Kgn}, \mathrm{MAC}, \mathrm{Vfy})$ and adversary $\mathcal{A}$.