This is the full version of the extended abstract which appeared in:
Proceedings of Information Security and Privacy, 13th Australasian Conference, ACISP 2008, Wollongong, Australia, July 7-9, 2008
Yi Mu, Willy Susilo, Jennifer Seberry (Eds.), Springer-Verlag, LNCS 5107, pp. 6--20.

Enforcing User-Aware Browser-Based Mutual Authentication with Strong Locked Same Origin Policy (Full Version)

Sebastian Gajek[1] and Mark Manulis[2] and Jörg Schwenk[1]

[1] Horst Görtz Institute for IT-Security, Germany
{sebastian.gajek|joerg.schwenk}@nds.rub.de
[2] UCL Crypto Group, Belgium
mark.manulis@uclouvain.be

Abstract. The standard solution for mutual authentication between human users and servers on the Internet is to execute a TLS handshake during which the server authenticates using a X.509 certificate followed by the authentication of the user either with own password or with some cookie stored within the user's browser. Unfortunately, this solution is susceptible to various impersonation attacks such as phishing as it turned out that average Internet users are unable to authenticate servers based on their certificates.
In this paper we address security of *cookie-based authentication* using the concept of *strong locked same origin* policy for browsers introduced at ACM CCS'07. We describe a cookie-based authentication protocol between human users and TLS-servers and prove its security in the extended formal model for *browser-based mutual authentication* introduced at ACM ASIACCS'08. It turns out that the small modification of the browser's security policy is sufficient to achieve provably secure cookie-based authentication protocols considering the ability of users to recognize images, video, or audio sequences.

1 Introduction

Motivation The browser plays an indispensable function as the user's interface to access the rich world of Web based services. In order to serve the purpose of an universal client, commodity browsers have been augmented with numerous functionalities. Examples include extensions of the HTTP header to control caching and transport cookies, or the HTML markup language to enable high-level scripting and supply technologies like AJAX, AFLEX or SOAP. By contrast, much effort to amend the browser security model and provide new cryptographic services has not been spent. Since its adaption more than a decade ago [9], the Transport Layer Security (TLS) framework is the main pillar of browser-based protocols to provide Web applications with a security layer. After the protocol framework has been peer-reviewed without finding any significant vulnerabilities [22, 24, 25, 28], it has been believed to be the holy grail for secure Web authentication. However, recent studies point out that average-skilled Internet users understand neither TLS nor its indication in commodity Web browsers at all [7, 27]. Users tend to ignore browser's warnings and prefer to identify Web sites on the basis of non-technical indicators (e.g., brands, logos). This attitude provides a wrong sense of security. An adversary may fake the site and disclose the user's password ("phishing attacks"). The advent of these large-scale fraud attacks has led to several modifications in the visualization of TLS. Unfortunately, it seems to turn out that the changes do not meet their high expectations either [16].

Another line of research addresses the design of authentication protocols that provide user-awareness. The essence of user-aware protocols is to relax the assumptions on user behavior and provide secure authentication ceremonies. Recently, the authors of this paper introduced a formal security model for *browser-based mutual authentication (BBMA)* between a human user and a server where the browser is modeled as the mediator of the communication [11]. Their model is an extension of the classical model for authentication from [3] towards consideration of user-awareness within the authentication protocols on the Internet whereby user-awareness is modeled via *human perceptible authenticators (HPAs)* that are implied by natural human senses, such as recognition of images, videos, and audio sequences. In addition to the model, [11] describes a protocol called BBMA (based on the ideas of the PassMark Security Inc.'s Two-Factor-Two-Way Authentication™) which can be implemented within the standard specification of the TLS protocol. In this protocol the human user authenticates via password which is typed into an HTML form only after the successful recognition of some expected HPA sent by the server. In order to protect the disclosure of this HPA to unauthorized parties, the TLS protocol uses client (possibly self-generated) certificates which serve as a cryptographic identifier for the corresponding HPA.

Extending this line of research, we deal with user-awareness in cookie-based authentication protocols. These protocols execute a server-only authenticated TLS session, where the user authenticates through a cookie that has been previously set by the server and stored in the browser's cache. The technique has the advantage that the user is refrained from retyping the password. Further, the cookie is taken from a sufficiently large random distribution. There is no need to expect a "security defect" due to the use of low-entropy passwords. These simplifications of user authentication have led to a wide adaption of cookie-based authenticated channels in browser-based protocols and there are many protocols that build upon this technique. Unfortunately, they have been shown to be vulnerable when taking the mature browser security model into account (see Section 2 for more discussions). The crux is that the browser decides on the basis of the server's domain name whether to reveal the cookie. The adversary is feasible to steal the cookie by spoofing the domain names and there are many attacks allowing the adversary to do this (e.g., dynamic pharming, DNS rebinding [15, 19].

To protect against the growing presence of these threats, Karlof et. al. propose refinements of the browser's cookie disclosure policy [19, 23]. Their contribution is to augment the browser with some additional functionality which uses cryptographic mechanisms to enforce restricted access policies without relying on DNS, dubbed the *strong locked same origin (SLSO)* policy. In the context of cookie-based authentication protocols over the TLS channel, the SLSO policy enforcement means that the browser sends a cookie to the server only after the server proves the possession of a valid cryptographic identifier, namely the server's public key, i.e., the server proves the knowledge of the corresponding private key.

Contributions In this paper we extend our model from [11] towards cookie-based authentication and consideration of the browser's SLSO policy. Using the extended model we analyze the security of the cookie-based version of `BBMA` from [11] re-engineered under the SLSO policy. We call the modified protocol `BBMA−SLSO`. It turns out that some minor changes of the browser security model to enforce the SLSO policy—which is a straightforward task compared to the large scale deployment of, say secure domain name resolution protocols (DNSSEC)—turns an insecure protocol into a provably secure one. Additionally, the use of SLSO policy allows us to eliminate the costly use of the client certificates, which are essential to prove security of `BBMA`. In addition to the formal security definition, `BBMA−SLSO` has additional advantages over previous cookie-based authentication protocols. The advantages include

1. `BBMA−SLSO` is *user-aware*. In order to authenticate, the server sends a HPA, which serves (i) as non-cryptographic identifier for the user to validate the server as in the physical world where identities are provided in an easily recognizable fashion and (ii) as fail-stop mechanism to hamper that she discloses private information on a faked site.
2. `BBMA−SLSO` fits into the standard TLS specification. There is no need to modify commodity server implementations. In fact, the necessary augmentations address browsers, more precisely their functionality to access cookies corresponding to the SLSO policy. See [23] for more details.

We remark that the enforcement of the SLSO policy is ineligible to protect against *cross-site scripting (XSS)* attacks. The anatomy of XSS attacks is to exploit weaknesses of application servers and inject malicious scripts into the communication that enable the adversary to invoke certain browser functionalities. Since the scripts are in the same security context the SLSO policy does not help. Consequently, the adversary would have access to the user's password typing, the cookie and HPA in `BBMA−SLSO`. Though we treat XSS attacks as (server) corruptions in our model and exclude them in the analysis, a work-around to make `BBMA−SLSO` resistant against the attacks is to completely isolate the named security critical information and prevent that they are accessible from the surrounding (potentially malicious) scripts. Such a feature is already available in the Internet Explorer for cookies [21]. The approach has to be extended for passwords and HPAs. Since the implementation of the SLSO policy requires the modification of the current browser's security policy anyway, we suggest to enrich this policy with the private/public tagging of elements. An element such as a password field tagged with a private value shall signal the browser that any script is prevented from access, regardless of its security context. See [10] for more details.

Organization The remainder sections are structured as follows. We review related work in Section 2. In Section 3, we describe the formal security model for cookie-based BBMA protocols under consideration of the SLSO policy. In Section 4 we specify a concrete protocol called `BBMA−SLSO` using the high level description of the TLS handshake in the key transport mode and prove that it is user-aware and satisfies the defined authentication requirement. Finally, we conclude the paper in Section 5.

2 Related Work

So far, few browser-based protocols have been subject to rigorous security analysis: Kormann and Rubin [20] show that Microsoft's .NET passport, a Web-based realization of the Kerberos protocol for single sign on, is susceptible to attacks where the adversary steals the ticket granting ticket cookie. Soghoian and Jakobsson [31] investigate the SiteKey-protocol that displays a previously negotiated image in addition to password forms in order to signal that the user is connected to the benign server. The authors show the feasibility of stealing the shared secret that is stored in a cookie. Groß [12] analyzes SAML, an alternative single sign on protocol, and shows that the protocol is vulnerable to adaptive attacks where the adversary intercepts the authentication token contained in the URL. By contrast, BBMA−SLSO has formal security arguments and is provably secure in a model which takes into account the adversarial control over the network and attacks against the classical browser's security policies that reveal weak identifiers, such as cookies.

Groß et al. prove in [14] the security of WS-Federation passive Requestor Profile—a browser-based protocol for federated identity management. The proof is carried out in the browser model [13] that builds on the Reactive Simulatability framework due to Pfitzmann and Waidner [26]. The model abstracts away the TLS-protected channel through an ideal functionality that captures the same cryptographic task and presupposes ideal users who are able to identify servers based on certificates. There exists no soundness proof that TLS is simulatable and realizes such functionality, especially with respect to the relaxed user behavior assumptions. BBMA−SLSO takes explicitly into account the TLS protocol and is shown to be provably secure in the Random Oracle Model when instantiated with the widely deployed key transport cipher suite in server authentication mode.

3 Modeling BBMA with SLSO Policy

In this section we extend our security model for browser-based mutual authentication from [11] towards consideration of cookie-based authentication and the SLSO policy implemented within the browser.

3.1 Protocol Participants and Communication Model

User, Browser, Server, and their Long-Lived Keys Let $\mathcal{U}$ denote a *human user* for whom we do not make any further assumptions except for the ability to use some naturally born senses. We assume that $\mathcal{U}$ remembers some (high-entropy) *human perceptible authenticator (HPA)* $w \in \mathcal{W}$ (e.g. an image or a video/audio sequence from some space $\mathcal{W}$) as its long-lived key $LL_{\mathcal{U}}$.

To the contrary, the *browser* $\mathcal{B}$ and the *server* $\mathcal{S}$ are modeled as PPT machines. $LL_{\mathcal{B}}$ is the browser's high-entropy long-lived key which contains $(\mathcal{S}, pk_{\mathcal{S}}, cky)$ where $\mathcal{S}$ is the identity (domain name) of the server, $pk_{\mathcal{S}} \in \{0,1\}^{p_1(\kappa)}$ its certified public key, and $cky \in \{0,1\}^{p_2(\kappa)}$ is the cookie set by $\mathcal{S}$ during the establishment of the security association with the *client* which is denoted by $\mathcal{C} = (\mathcal{U}, \mathcal{B})$. (Here and in the following,

$p_i : \mathbb{N} \rightarrow \mathbb{N}, i \in [1,5]$ is a polynomial and $\kappa \in \mathbb{N}$ the security parameter.) We assume that cky contains secret information (e.g. obfuscated or cryptographically processed password) which allows $\mathcal{S}$ to uniquely identify $\mathcal{U}$. Similarly, $LL_{\mathcal{S}}$ contains the private key $sk_{\mathcal{S}} \in \{0,1\}^{p_1(\kappa)}$ and the tuple $(\mathcal{U}, cky, w)$.

Additionally, by $\mathcal{C}$ we denote the traditional *client* given by a pair $(\mathcal{U}, \mathcal{B})$.

Communication between $\mathcal{B}$ and $\mathcal{U}$ via `render`-Function Let $\lambda_i : \mathbb{N} \rightarrow \mathbb{N}, i \in [1,2]$ be two polynomials. $\mathcal{B}$ communicates to $\mathcal{U}$ through the *visualization function* `render` : $\mathcal{M} \times \Psi \rightarrow \mathcal{M}^*$ where $\mathcal{M} \in \{0,1\}^{\lambda_1(\kappa)}$ is the message space (space of all HTML messages) and $\Psi \in \{0,1\}^{\lambda_2(\kappa)}$ is the browser's configuration for message processing that may be altered by querying the browser's DOM model.

Modeling User-Awareness via `recognize`-Function Similar to [11] we assume that $\mathcal{U}$ can recognize some previously remembered high-entropy HPA $w \in \mathcal{W}$. The recognition is handled by a boolean *human perception function* `recognize` : $\mathcal{M}^* \times \mathcal{W} \rightarrow \{0,1\}$ which on input a visualized message $m^* \in \mathcal{M}^*$ and w the `recognize` function outputs 1 if $\mathcal{U}$ recognizes w among the content of m^*; otherwise the output is 0. In this paper we assume that if m^* contains w (denoted as $m^*|w$) then `recognize` outputs 1, i.e., the ability of $\mathcal{U}$ to recognize w is *perfect*. On the other hand, we do *not* assume that w is the only HPA for which `recognize` outputs 1, i.e., we do *not* idealize $\mathcal{U}$ as there can be some set $\mathcal{W}^* \subseteq \mathcal{W}$ which contains HPAs that are *perfectly human-indistinguishable* from $\mathcal{U}$ according to the following definition.

Definition 1 (Perfect Human-Indistinguishability of HPAs). *Let $w \in \mathcal{W}$ be some given HPA. For any $m^* \in \mathcal{M}^*$ and any $w^* \in \mathcal{W}$, we say that w and w^* are perfectly human-indistinguishable, if for any human user $\mathcal{U}$*

$$\Big|\Pr[\mathcal{U}.\texttt{recognize}(m^*|w, w) = 1] - \Pr[\mathcal{U}.\texttt{recognize}(m^*|w^*, w) = 1]\Big| = 0$$

where the probabilities are computed over the choices of w^*. By $\mathcal{W}^* \subseteq \mathcal{W}$ we denote the set of all perfectly human-indistinguishable HPAs for some given $w \in \mathcal{W}$ assuming that $w \in \mathcal{W}^*$.

The main idea in designing user-aware security protocols based on HPAs is to opt for authenticators for which $\mathcal{W}^*$ is *sufficiently small* for most of the users. In this case the probability that an adversary chooses or guesses some HPA that cannot be distinguished from w by $\mathcal{U}$ can be kept low. The ideal case would be if $\mathcal{W}^*$ would consist only of w. We call w a *good* HPA if the size of the set $\mathcal{W}^*$ is sufficiently small such that the term $|\mathcal{W}^*|/|\mathcal{W}|$ which is used in our proof beside other cryptography-related terms to compute the overall probability of a successful attack is negligible.

For our protocol we assume that the HPA used by $\mathcal{U}$ in the execution of our protocol is good. We stress that in order to identify good HPAs extensive user experiments, possibly under consideration of specific statistic models, have to be conducted. We conjecture that good HPAs may be found from the personal digital images, audio and even video sequences.

Protocol Sessions and Participating Instances Participation of $\mathcal{C} = (\mathcal{U}, \mathcal{B})$ and $\mathcal{S}$ in distinct executions of Π is modeled via *instances* $[\mathcal{C}, sid_{\mathcal{C}}]$ and $[\mathcal{S}, sid_{\mathcal{S}}]$ where $sid_{\mathcal{C}}, sid_{\mathcal{S}} \in \mathbb{N}$ are respective *session ids* and if $sid_{\mathcal{C}} = sid_{\mathcal{S}}$ then the instances are *partnered* – belong to the same session. We sometimes write $\mathcal{C}$ and $\mathcal{S}$ instead of their instances when the difference is visible from the context.

Execution Stages Once initialized with the corresponding long-lived key an instance $[\mathcal{C}, sid_{\mathcal{C}}]$ or $[\mathcal{S}, sid_{\mathcal{S}}]$ is marked as *used* and turns into the *stand-by* stage where it waits for an invocation to execute the protocol. Upon receiving such invocation the instance turns into a *processing* stage where it proceeds according to the protocol specification until it collects enough information to decide whether the execution was successful or not, and to *terminate* then. If the execution is successful then we say that the instance *accepts* before it terminates; otherwise we say it *aborts*. The acceptance of $[\mathcal{C}, sid_{\mathcal{C}}]$ with $\mathcal{C} = (\mathcal{U}, \mathcal{B})$ is implied by the acceptance of $\mathcal{U}$ regardless of $\mathcal{B}$, as $\mathcal{U}$ is the ultimate endpoint of the communication and controls the browser. However, $[\mathcal{C}, sid_{\mathcal{C}}]$ aborts if either $\mathcal{U}$ or $\mathcal{B}$ does so.

3.2 Security Model

In the following we specify attacks and security goals for BBMA protocols from the perspective of fixed identities $\mathcal{S}$ and $(\mathcal{U}, \mathcal{B})$.

Assumptions on the Initialization We assume that the establishment of the security association between $\mathcal{S}$ and $(\mathcal{U}, \mathcal{B})$ during which $\mathcal{B}$ receives (certified) $pk_{\mathcal{S}}$ and cky, and $\mathcal{S}$ receives w is *trusted*. In practice, this can be done through the execution of the very first TLS handshake in the key transport mode under the assumption that this first session is not compromised. We remark that this assumption has practical substantiation. For example, assume that the protocol should be deployed for the login access to the online banking service of some bank UFB (for User Friendly Bank). If some $\mathcal{U}$ who does not have any online banking account at UFB receives phishing emails with the invitation to access some fake website of UFB there will be no damage even if $\mathcal{U}$ accepts. However, after $\mathcal{U}$ subscribes for the corresponding online service of UFB and receives the user guide that usually includes information on the connection establishment, it is likely that $\mathcal{U}$, especially if $\mathcal{U}$ is technology-unaware and has no experience in online banking, will follow the guidelines, at least for the very first session in which the required security association through the upload of w will be established. Thus, for a successful attack the phishing email should be received by $\mathcal{U}$ in the time period between the subscription and the registration on the site.

Assumptions on the Adversary The PPT adversary $\mathcal{A}$ controls all communication between the protocol parties. This implies:

- $\mathcal{A}$ controls the domain name resolution. This also allows $\mathcal{A}$ to mount phishing and pharming attacks. Due to the SLSO policy we assume that the adversary can establish security association $(\mathcal{S}', pk_{\mathcal{S}'}, cky')$ with the client $(\mathcal{U}, \mathcal{B})$ for any server identity $\mathcal{S}'$ as long as it can prove the knowledge of the corresponding private key

$sk_{\mathcal{S}'}$.[3] Upon sending forged domain resolution responses, the adversary obtains access to the parts of the browser's DOM model which are not protected by the policy. Note also that since the human recognizable authenticator is not cached, it can not be accessed using the DOM model.

- $\mathcal{A}$ can issue public keys which $\mathcal{B}$ accepts. There is *no* trusted third party in the sense of a trusted CA. Hence, a certified public key in a X.509 server certificate is treated as a public key that can be identified by a unique identifier (i.e., hash value of the public key).
- $\mathcal{A}$ is unable to corrupt $\mathcal{B}$. Note that in this model we do not deal with malware[4] attacks against $\mathcal{B}$ and $\mathcal{S}$, therefore, do not consider the case where $\mathcal{A}$ reveals the ephemeral and long-lived secrets stored inside $\mathcal{B}$. In particular this implies that the adversary is not able to access the secure cookie cky unless its request is successfully verified by $\mathcal{B}$ based on the SLSO policy. By the same token we do not consider attacks resulting from the physical access of the adversary to the user's digital device running $\mathcal{B}$.
- $\mathcal{A}$ is unable to corrupt $\mathcal{S}$. Note also that in this model we do not deal with malware attacks against the server. This means that the adversary is excluded from revealing the ephemeral and long-lived secrets stored inside $\mathcal{S}$.

Adversarial Queries $\mathcal{A}$ can participate in the actual protocol execution via the following queries:

- $\mathsf{Execute}(\mathcal{C}, \mathcal{S})$: $\mathcal{A}$ eavesdrops the execution of the new protocol session between $\mathcal{C}$ and $\mathcal{S}$ and receives its transcript.
- $\mathsf{Invoke}(\mathcal{C}, \mathcal{S})$: $\mathcal{U}$ starts the protocol execution with the new instance of $\mathcal{S}$ using the associated instance of browser $\mathcal{B}$ and $\mathcal{A}$ obtains the first protocol message returned by $\mathcal{B}$ (which is usually generated on some input received from $\mathcal{U}$, e.g., the entered URL).
- $\mathsf{Send}(P, m)$: In an active attack $\mathcal{A}$ can send a message to some (instance) of $P \in \{\mathcal{U}, \mathcal{B}, \mathcal{S}\}$ whereby messages addressed to $\mathcal{U}$ are implicitly handled as messages addressed to the associated browser $\mathcal{B}$ with the subsequent execution of $\texttt{render}(m, \Psi)$ and visualization of its output to $\mathcal{U}$. $\mathcal{A}$ receives the response which P generates after having processed m according to the specification of Π (or an empty string if m is unexpected).
- $\mathsf{RevealState}(\mathcal{B})$: $\mathcal{A}$ receives information stored within the browser's state Ψ and which is not protected via the SLSO policy. Additionally, it returns $(\mathcal{S}, pk_{\mathcal{S}})$, i.e., $\mathcal{A}$ may learn which servers have security associations with the client, without learning their secure cookies.

[3] Assuming that the initialization process is done during the trusted TLS key transport session between $(\mathcal{U}, \mathcal{B})$ and $\mathcal{S}$, the adversary must be able to decrypt messages encrypted with $pk_{\mathcal{S}'}$. Under the assumption that the deployed asymmetric encryption scheme is sufficiently secure the decryption operation can be seen as the required proof of possession.

[4] Consideration of malware attacks and augmentation of the proposed model with Trusted Computing functionalities to model resistance against malware attacks is surely an interesting aspect for the future work on security of browser-based protocols.

- $\mathsf{SetCKY}(\mathcal{B}, (\mathcal{S}', pk_{\mathcal{S}'}, cky'))$: With this query (which is new in comparison to [11]) $\mathcal{A}$ sets up a new security association with $(\mathcal{U}, \mathcal{B})$ on behalf of some server $\mathcal{S}'$ as long as $pk_{\mathcal{S}'} \neq pk_{\mathcal{S}}$ (note that due to our assumptions that $\mathcal{A}$ controls the domain name resolution and can issue certificates that $\mathcal{B}$ will accept we explicitly allow $\mathcal{S}'$ to be equal to $\mathcal{S}$.) $\mathcal{A}$ receives the HPA $w' \in \mathcal{W}$ chosen by $\mathcal{U}$ such that it is distinguishable from w, i.e., $w' \notin \mathcal{W}^*$ according to the Definition 1.[5]

Correctness and Browser-Based Mutual Authentication The following definition specifies the correctness requirement for BBMA protocols.

Definition 2 (Correctness). *A BBMA protocol Π is* correct *if each* $\mathsf{Execute}(\mathcal{C}, \mathcal{S})$ *query results in two instances,* $[\mathcal{C}, sid_{\mathcal{C}}]$ *and* $[\mathcal{S}, sid_{\mathcal{S}}]$ *which are partnered* ($sid_{\mathcal{C}} = sid_{\mathcal{S}}$) *and accept prior to termination.*

In the following we define the main security requirement of browser-based mutual authentication between participating $\mathcal{U}$ and $\mathcal{S}$ with $\mathcal{B}$ acting as a mediator of the communication.

Definition 3 (Browser-Based Mutual Authentication). *Let Π be a correct protocol according to Definition 2 and* $\mathsf{Game}_{\Pi}^{bbma}(\mathcal{A}, \kappa)$ *the interaction between the instances of* $\mathcal{C} = (\mathcal{U}, \mathcal{B})$ *and $\mathcal{S}$ with a PPT adversary $\mathcal{A}$ who is allowed to query* $\mathsf{Execute}$, Invoke, Send, $\mathsf{RevealState}$, *and* SetCKY. *We say that $\mathcal{A}$* wins *if at some point during the interaction:*

1. *An instance $[\mathcal{C}, sid_{\mathcal{C}}]$ accepts but there is* ***no*** *partnered instance $[\mathcal{S}, sid_{\mathcal{S}}]$,* ***or***
2. *An instance $[\mathcal{S}, sid_{\mathcal{S}}]$ accepts but there is* ***no*** *partnered instance $[\mathcal{C}, sid_{\mathcal{C}}]$.*

The maximum probability of this event (over all adversaries running in time κ) is denoted $\mathsf{Succ}_{\Pi}^{bbma}(\mathcal{A}, \kappa) = \max_{\mathcal{A}} |\Pr[\mathcal{A}$ *wins in* $\mathsf{Game}_{\Pi}^{bbma}(\mathcal{A}, \kappa)]|$. *We say that Π provides* browser-based mutual authentication *if this probability is a negligible function of κ.*

The first requirement ensures that $\mathcal{U}$ authenticates to the matching server $\mathcal{S}$. Since the acceptance of $[\mathcal{C}, sid_{\mathcal{C}}]$ with $\mathcal{C} = (\mathcal{U}, \mathcal{B})$ is implied by the acceptance of $\mathcal{U}$ the second requirement ensures that $\mathcal{S}$ authenticates to the matching user $\mathcal{U}$. In both cases $\mathcal{B}$ plays the role of the mediator of the communication and can be queried by $\mathcal{A}$; thus, not mentioning $\mathcal{B}$ in the above definition would be incorrect from the formal point of view.

4 User-Aware BBMA over TLS with the SLSO Policy

In this section we specify the `BBMA-SLSO` protocol which can be seen as the modification of the `BBMA` protocol from [11] towards cookie-based authentication and SLSO policy.

4.1 Building Blocks of `BBMA-SLSO`

TLS Protocol The main pillar of `BBMA-SLSO` is the server authenticated *key transport*, where the server's identity is a cryptographic value independent from the Internet infrastructure. This complies with RSA-based ciphersuites as specified in [1]. These suites are preferentially negotiated between standard browsers and servers.

[5] Thus, we assume that users do not use same HPAs with different servers.

Cryptographic Primitives $\texttt{BBMA-SLSO}$ uses (well-known) cryptographic primitives that are deployed in the cryptographic key transport suites of the TLS protocol, namely:

- A *pseudo-random function* $\texttt{PRF} : \{0,1\}^{p_3(\kappa)} \times \{0,1\}^* \rightarrow \{0,1\}^*$. Note that TLS defines $\texttt{PRF}$ with data expansion s.t. it can be used to obtain outputs of a variable length which becomes useful for the key extraction phase. We refer to [8] for the proof that the key extraction function in TLS is indeed pseudo-random. By $\mathsf{Adv}_{\texttt{PRF}}^{prf}(\kappa)$ we denote the maximum advantage over all PPT adversaries (running within security parameter κ) in distinguishing the outputs of $\texttt{PRF}$ from those of a random function better than by a random guess.
- A *symmetric encryption scheme* which provides indistinguishability under chosen plaintext attacks (IND-CPA). The symmetric encryption operation is denoted Enc and the corresponding decryption operation Dec. By $\mathsf{Adv}_{(Enc,Dec)}^{ind-cpa}(\kappa)$ we denote the maximum advantage over all PPT adversaries (running within security parameter κ) in breaking the IND-CPA property of (Enc, Dec) better than by a random guess;
- An IND-CPA secure *asymmetric encryption scheme* whose encryption operation is denoted $\mathcal{E}$ and the corresponding decryption operation $\mathcal{D}$. By $\mathsf{Adv}_{(\mathcal{E},\mathcal{D})}^{ind-cpa}(\kappa)$ we denote the maximum advantage over all PPT adversaries (running within security parameter κ) in breaking the IND-CPA property of $(\mathcal{E}, \mathcal{D})$ better than by a random guess; Note that the general case of RSA-OAEP encryption which is used in the TLS key transport mode has been proven in [29] based on the assumptions of the Random Oracle Model [4] to satisfy indistinguishability under *adaptive* chosen ciphertext attacks (IND-CCA2), which is stronger than IND-CPA. Also [18] provides such proof which is tailored specifically to the construction used in the TLS protocol. Still, we emphasize that for the security of $\texttt{BBMA-SLSO}$ the weaker requirement of IND-CPA which is implied by IND-CCA2 is fully sufficient.
- A cryptographic *collision-resistant hash function* $\texttt{Hash} : \{0,1\}^* \rightarrow \{0,1\}^{p_4(\kappa)}$. By $\mathsf{Succ}_{\texttt{Hash}}^{coll}(\kappa)$ we denote the maximum success probability over all PPT adversaries (running within security parameter κ) in finding a collision, i.e., a pair $(m, m') \in \{0,1\}^* \times \{0,1\}^*$ s.t. $\texttt{Hash}(m) = \texttt{Hash}(m')$.
- A *digital signature scheme* which provides existential unforgeability under chosen message attacks (EUF-CMA). The signing operation is denoted Sig and the corresponding verification operation Ver. By $\mathsf{Succ}_{(Sig,Ver)}^{euf-cma}(\kappa)$ we denote the maximum success probability over all PPT adversaries (running within security parameter κ) given access to the signing oracle in finding a forgery;
- The well-known *message authentication code* function $\texttt{HMAC}$ which is believed to satisfy *weak unforgeability under chosen message attacks* (WUF-CMA) [2]. Here we remark that security of $\texttt{HMAC}$ is not relevant for the security analysis of $\texttt{BBMA-SLSO}$. A detailed look on the protocol from the formal perspective shows that using $\texttt{HMAC}$ is redundant since all $\texttt{HMAC}$ values are encrypted prior to the transmission. Nevertheless, we do not omit protocol parts where $\texttt{HMAC}$ is computed from our description since this is what happens in the today's execution of TLS.

SLSO Policy in $\texttt{BBMA-SLSO}$ During the initialization procedure which is assumed to be trusted server $\mathcal{S}$ establishes a security association with the client $(\mathcal{U}, \mathcal{B})$ using the TLS protocol in key transport with its (certified) public key. For the successful verification of the SLSO policy in subsequent connections $\mathcal{B}$ stores $pk_{\mathcal{S}}$ and the http cookie provided by $\mathcal{S}$. This cookie contains information which allows $\mathcal{S}$ to authenticate $\mathcal{U}$. On each connection with $\mathcal{S}$, $\mathcal{B}$ has to make a decision whether to send cky or not. Following the definition of the SLSO policy in [19], $\mathcal{B}$ decides by comparing the public key used by the candidate server during that particular TLS handshake to the stored $pk_{\mathcal{S}}$. If the keys are equal then cky is transmitted, otherwise not. However, since the browser is a very general piece of software that must be able to communicate with any http server on the Internet, we design BBMA−SLSO in such a way that it does not abort the communication if this verification fails; otherwise this would pose a lot of compatibility problems and could be seen as an impractical solution. Instead, if the verification fails, the browser will simply continue with the protocol, by sending the *empty cookie* which we consider as some constant publicly known value $\zeta \in \{0,1\}^{p_2(\kappa)}$. In this way the decision on whether the communication should be continued or not is mitigated to $\mathcal{S}$, which will normally abort the communication since otherwise $\mathcal{U}$ remains unauthenticated.

4.2 Protocol Description

In the following we describe the execution of the BBMA−SLSO protocol specified in Figure 1. Let l_1, l_2, l_3 and l_4 denote the publicly known *labels* specified in TLS for the instantiation of PRF. (We write in parenthesis the corresponding standard TLS messages.)

Initiate the Protocol. The user $\mathcal{U}$ initiates the protocol by communicating server's URL to the own browser $\mathcal{B}$. Upon resolving the corresponding address $\mathcal{B}$ chooses his own *nonce* $r_{\mathcal{C}}$ of length $p_5(\kappa)$ at random and forwards it to $\mathcal{S}$ (ClientHello). In response $\mathcal{S}$ chooses own random *nonce* $r_{\mathcal{S}}$ and a *TLS session identifier* sid of length $p_5(\kappa)$ and appends it to the own certificate $cert_{\mathcal{S}}$ (ServerHello). We stress that sid chosen by $\mathcal{S}$ is not the session identifier $sid_{\mathcal{S}}$ used in our security model but a value specified in TLS.

Negotiate Key Material. $\mathcal{B}$ chooses a *pre-master secret* k_p of length $p_3(\kappa)$ at random and sends it to $\mathcal{S}$ encrypted with the received public key $pk_{\mathcal{S}}$ (ClientKeyExchange) taken from the servers certificate $Cert_{\mathcal{S}}$. The pre-master secret k_p is used to derive the *master secret* k_m through a pseudo-random function PRF on input $(l_1, r_{\mathcal{C}}|r_{\mathcal{S}})$ with k_p as the secret seed. This key derivation is performed based on the standard TLS pseudo-random function PRF (see [1, Sect. 5]). The master secret is then used as a secret seed for the instantiation of the pseudo-random function PRF on input $(l_2, r_{\mathcal{C}}|r_{\mathcal{S}})$ to derive the *session keys* $k_1|k_2$ used to encrypt and authenticate session messages exchanged between $\mathcal{B}$ and $\mathcal{S}$. TLS specifies the generation of six session keys: A symmetric encryption key, a MAC key, and an IV for block ciphers only (either for client and server). For simplicity, we denote k_1 as the encryption key and k_2 as the authentication key which are the same for $\mathcal{B}$ and $\mathcal{S}$. Here we remark that as shown later in our security analysis the use of different keys for encryption and authentication in TLS is redundant from the formal point of view. The reason is that each computed HMAC value is encrypted using k_1 prior to its transmission over the network. Since the computed value $k_1|k_2$ can be seen as a single output of PRF the security of the applied encryption scheme is already sufficient to achieve symmetric authentication of the encrypted message.

Session Key Confirmation. $\mathcal{B}$ confirms the session key generation, i.e., $F_{\mathcal{C}}$ is the first message that is authenticated via HMAC computed with k_2 and encrypted via the symmetric encryption scheme computed with k_1. $F_{\mathcal{C}}$ is computed as output of PRF on input (l_3, h_1) with k_m as the secret seed; whereby h_1 denotes the hash value computed over all messages previously processed by $\mathcal{B}$ (ClientFinished). Further, $\mathcal{S}$ generates k_m and derives the session keys (k_1, k_2) in a similar way. $\mathcal{S}$ uses the own session keys (k_1, k_2) to ensure that it communicates with $\mathcal{B}$ through the verification of $F_{\mathcal{C}}$. If the verification fails, $\mathcal{S}$ aborts the protocol. Otherwise, it confirms the negotiated session parameters, using PRF on input (l_4, h_2) with k_m as secret seed; whereby h_2 denotes the hash value over the received messages. The output of PRF is first authenticated via HMAC computed with k_2 and then encrypted via the symmetric encryption scheme computed with k_1 (ServerFinished). The client $\mathcal{C}$ checks this message analogously.

Mutual Authentication between Browser and Server. The browser $\mathcal{B}$ now exploits the fact that the server $\mathcal{S}$ has been authenticated in the previous step by showing that he knows the private key associated with $pk_{\mathcal{S}}$. This value is used as a key to the credential store of the browser, and the corresponding cookie cky is retrieved and sent to the server, encrypted with k_1 together with the attached message authentication code computed using k_2.

Human Perceptible Server Authentication. The server selects the HPA w associated with cky, and sends it (encrypted with k_1 together with the attached message authentication code computed using k_2) for display to the browser. We call the message in a high-level description the HumanAuth message. $\mathcal{B}$ communicates the decrypted authenticator to $\mathcal{U}$ through execution of the render function which takes as input the authenticator w and state Ψ and outputs the visualization of w named w^*. The abstract human perception function recognize is used to model the ability of $\mathcal{U}$ to decide whether the authenticator w^* matches the original authenticator w which is shared with $\mathcal{S}$ after the initialization stage.

Before we continue with the security analysis we reemphasize the triangular model of authentication in BBMA-SLSO. When verifying $F_{\mathcal{S}}$, $\mathcal{B}$ knows the identity of $\mathcal{S}$. $\mathcal{B}$ resolves $pk_{\mathcal{S}}$ to look up for the corresponding cookie cky. If no matching triple ($\mathcal{S}$,$pk_{\mathcal{S}}$, cky) exists, $\mathcal{B}$ sends an empty cookie ζ and continues with the protocol (it is now in responsibility of the server to abort); otherwise, $\mathcal{B}$ continues by sending cky confidentially to $\mathcal{S}$.

However, TLS in server authentication mode does not prevent $\mathcal{U}$ from contacting to a rogue server in order to disclose sensitive information. When verifying w^* through the execution of recognize, $\mathcal{U}$ is sure to be communicating to $\mathcal{S}$ through $\mathcal{B}$, since $\mathcal{S}$ is the only owner of w apart from $\mathcal{U}$. Upon this stage, the protocol ensures that $\mathcal{S}$ is authenticated to $\mathcal{U}$.

Client $(\mathcal{U}, \mathcal{B})$ | Server $\mathcal{S}$

$\{LL_{\mathcal{U}} := w,\ LL_{\mathcal{B}} := (\mathcal{S}, pk_{\mathcal{S}}, cky)\}$ | $\{LL_{\mathcal{S}} := (cky, w, sk_{\mathcal{S}}, cert_{\mathcal{S}})\}$

```
get URL of $\mathcal{S}$ from $\mathcal{U}$
$r_{\mathcal{C}} \in_r \{0,1\}^{p_5(\kappa)}$
$A := r_{\mathcal{C}}$
```

$$\xrightarrow{\boxed{A}}$$

```
$r_{\mathcal{S}}, sid \in_r \{0,1\}^{p_5(\kappa)}$
$sid_{\mathcal{S}} := r_{\mathcal{C}} | r_{\mathcal{S}}$
$B := r_{\mathcal{S}} | sid | cert_{\mathcal{S}}$
```

$$\xleftarrow{\boxed{B}}$$

```
$sid_{\mathcal{C}} := r_{\mathcal{C}} | r_{\mathcal{S}}$
$k_p \in_r \{0,1\}^{p_3(\kappa)}$
$k_m := \mathtt{PRF}_{k_p}(l_1, sid_{\mathcal{C}})$
[validate $cert_{\mathcal{S}}$]
get $pk'_{\mathcal{S}}$ from $cert_{\mathcal{S}}$
$C := \mathcal{E}_{pk'_{\mathcal{S}}}(k_p)$
$k_1 | k_2 := \mathtt{PRF}_{k_m}(l_2, sid_{\mathcal{C}})$
$h_1 := \mathtt{Hash}(A|B|C)$
$F_{\mathcal{C}} := \mathtt{PRF}_{k_m}(l_3, h_1)$
$D := Enc_{k_1}(F_{\mathcal{C}} | \mathtt{HMAC}_{k_2}(F_{\mathcal{C}}))$
```

$$\xrightarrow{\boxed{C|D}}$$

```
$k_p := \mathcal{D}_{sk_{\mathcal{S}}}(C)$
$k_m := \mathtt{PRF}_{k_p}(l_1, sid_{\mathcal{S}})$
$k_1 | k_2 := \mathtt{PRF}_{k_m}(l_2, sid_{\mathcal{S}})$
$h_1 := \mathtt{Hash}(A|B|C)$
$F_{\mathcal{C}} | \mu_D := Dec_{k_1}(D)$
**if** $F_{\mathcal{C}} \neq \mathtt{PRF}_{k_m}(l_3, h_1)$
**or** $\mu_D \neq \mathtt{HMAC}_{k_2}(F_{\mathcal{C}})$
**then** ABORT **else**
$h_2 := \mathtt{Hash}(A|B|C|F_{\mathcal{C}})$
$F_{\mathcal{S}} := \mathtt{PRF}_{k_m}(l_4, h_2)$
$E := Enc_{k_1}(F_{\mathcal{S}} | \mathtt{HMAC}_{k_2}(F_{\mathcal{S}}))$
```

$$\xleftarrow{\boxed{E}}$$

```
$F_{\mathcal{S}} | \mu_E := Dec_{k_1}(E)$
$h_2 := \mathtt{Hash}(A|B|C|F_{\mathcal{C}})$
**if** $F_{\mathcal{S}} \neq \mathtt{PRF}_{k_m}(l_4, h_2)$
**or** $\mu_E \neq \mathtt{HMAC}_{k_2}(F_{\mathcal{S}})$
**then** ABORT **else**
[SLSO policy test]
**if** $pk'_{\mathcal{S}} = pk_{\mathcal{S}}$
**then** $F := Enc_{k_1}(cky, \mathtt{HMAC}_{k_2}(cky))$
**else** $F := Enc_{k_1}(\zeta, \mathtt{HMAC}_{k_2}(\zeta))$
```

$$\xrightarrow{F}$$

```
$cky' | \mu_F := Dec_{k_1}(F)$
**if** $\mu_F \neq \mathtt{HMAC}_{k_2}(cky')$
**or** $cky' \neq cky$
**then** ABORT **else**
$G := Enc_{k_1}(w | \mathtt{HMAC}_{k_2}(w))$
ACCEPT
```

$$\xleftarrow{G}$$

```
$w | \mu_G := Dec_{k_1}(G)$
**if** $\mu_G \neq \mathtt{HMAC}_{k_2}(w)$
**then** ABORT **else**
visualize $w^* := \mathtt{render}(w, \Psi)$ to $\mathcal{U}$
```

Fig. 1. Sketch of the BBMA–SLSO protocol between $(\mathcal{U}, \mathcal{B})$ and $\mathcal{S}$ based on the SLSO policy of $\mathcal{B}$. Boxed messages denote the standard TLS handshake. $\mathcal{U}$ accepts in the protocol execution only if $\mathcal{U}.\mathtt{recognize}(w^*, w) = 1$.

4.3 Security Analysis

In the following we argue on the security of the proposed BBMA−SLSO protocol. We recall that the goal of the protocol is to provide mutual authentication between $\mathcal{U}$ and $\mathcal{S}$ communicating via $\mathcal{B}$ according to Definition 3.

Theorem 1 (BBMA-Security). *Let q denote the total number of executed protocol sessions during the interaction with an adversary $\mathcal{A}$ participating in* $\mathsf{Game}^{bbma}_{\mathtt{BBMA-SLSO}}(\mathcal{A},\kappa)$. *If* PRF *is pseudo-random,* Hash *is collision-resistant, (Enc, Dec) and $(\mathcal{E},\mathcal{D})$ are IND-CPA secure and $\mathcal{W}^*$ is sufficiently small, then* BBMA−SLSO *provides browser-based mutual authentication in the sense of Definition 3, and*

$$\mathsf{Succ}^{bbma}_{\mathtt{BBMA-SLSO}}(\kappa) \leq \frac{q|\mathcal{W}^*|}{|\mathcal{W}|} + \frac{3q^2}{2^{p_5(\kappa)}} + \frac{q^2}{2^{p_3(\kappa)}} + \frac{q}{2^{p_2(\kappa)}} + q\mathsf{Adv}^{ind-cpa}_{(\mathcal{E},\mathcal{D})}(\kappa) + \\ 4q\mathsf{Adv}^{ind-cpa}_{(Enc,Dec)}(\kappa) + 4q\mathsf{Adv}^{prf}_{\mathtt{PRF}}(\kappa) + 2q\mathsf{Succ}^{coll}_{\mathtt{Hash}}(\kappa).$$

Proof. (Sketch) In this proof we apply the meanwhile classical proving technique from [30]. We construct a *sequence of games* $\mathbf{G}_i$, $i = 0,\ldots,14$ and denote by Win_i the event that adversary $\mathcal{A}$ breaks the mutual authentication of the protocol in game $\mathbf{G}_i$, i.e., wins in the corresponding interaction as described in Definition 3.

Game $\mathbf{G}_0$. [*Real protocol*] This is the real $\mathsf{Game}^{bbma}_{\mathtt{BBMA-SLSO}}(\kappa)$ played between a simulator Δ and a PPT adversary $\mathcal{A}$. Δ simulates the actions of the server $\mathcal{S}$ and the browser $\mathcal{B}$ according to the natural protocol specification and answers all queries of $\mathcal{A}$. Although we treat the human user $\mathcal{U}$ as part of Δ we explicitly assume that $\mathcal{U}$ performs the simulated tasks on its own.

Game $\mathbf{G}_1$. [*Same TLS Session Id*] In this game the simulation aborts if during the interaction the simulator chooses the same TLS session id sid on behalf of $\mathcal{B}$ in two different protocol sessions. Considering the probability for the collision of two random choices we obtain

$$|\Pr[\mathsf{Win}_1] - \Pr[\mathsf{Win}_0]| \leq \frac{q^2}{2^{p_5(\kappa)}}. \quad (1)$$

Game $\mathbf{G}_2$. [*Same Nonces*] In this game the simulation aborts if during the interaction the simulator chooses the same random nonce $r_\mathcal{C}$ resp. $r_\mathcal{S}$ on behalf of $\mathcal{B}$ resp. $\mathcal{S}$ in two different protocol sessions. Similar to Game $\mathbf{G}_1$ we obtain

$$|\Pr[\mathsf{Win}_2] - \Pr[\mathsf{Win}_1]| \leq \frac{2q^2}{2^{p_5(\kappa)}}. \quad (2)$$

Note that since in our protocol both session ids – $sid_\mathcal{C}$ and $sid_\mathcal{S}$ – are computed as concatenation $r_\mathcal{C}|r_\mathcal{S}$ this game rules out the occurrence of different (uncorrupted) client or server instances having the same session id, i.e., for the honest party each new session is associated with a different session id.

Game $\mathbf{G}_3$. [*Same Pre-master Secret*] In this game the simulation aborts if during the interaction the simulator chooses the same pre-master secret k_p on behalf of $\mathcal{B}$ in two different protocol sessions. Thus,

$$|\Pr[\mathsf{Win}_3] - \Pr[\mathsf{Win}_2]| \leq \frac{q^2}{2^{p_3(\kappa)}}. \quad (3)$$

Game $\mathbf{G}_4$. [*Indistinguishability of C*] This game proceeds exactly as Game $\mathbf{G}_3$ except for the following actions of Δ: if Δ receives a message $B = A|r_{\mathcal{S}}|cert_{\mathcal{S}}$ such that $cert_{\mathcal{S}}$ contains the public key $pk_{\mathcal{S}}$ stored by $\mathcal{B}$ as part of the adversarial Send query then Δ computes $C := \mathcal{E}_{pk_{\mathcal{S}}}(\alpha)$ for some additionally randomly chosen $\alpha \in \{0,1\}^{p_3(\kappa)}$. Otherwise, if $B = A|r_{\mathcal{S}}|cert'_{\mathcal{S}}$ such that $cert'_{\mathcal{S}}$ contains a different public key $pk'_{\mathcal{S}} \neq pk_{\mathcal{S}}$ then Δ computes $C := \mathcal{E}_{pk'_{\mathcal{S}}}(k_p)$, i.e., exactly as specified in the protocol. We denote this public key injection event by InjPK. With the above modification we consider in our proof attacks against users that do not properly verify the validity of server's certificate. Note that one of our assumptions on $\mathcal{A}$ is that it can issue certificates that $\mathcal{B}$ accepts. That is, the protocol proceeds in a natural way even if the user/browser accepts some forged or invalid certificate. On the other hand, if the real server's public key $pk_{\mathcal{S}}$ is part of the received certificate (no InjPK occurred) then the purpose of this game is to show that the security of the used asymmetric encryption scheme has an impact on the secrecy of the transmitted pre-master secret k_p. Due to the IND-CPA property of $(\mathcal{E}, \mathcal{D})$ we get

$$|\Pr[\mathsf{Win}_4] - \Pr[\mathsf{Win}_3]| \leq q\mathsf{Adv}^{ind-cpa}_{(\mathcal{E},\mathcal{D})}(\kappa). \tag{4}$$

Note that the specification of TLS prescribes the use of the RSA encryption according to PKCS#1 (a.k.a. RSA-OAEP) which in turn is known to provide IND-CPA security in ROM (see [29] for the proof).

Game $\mathbf{G}_5$. [*Pseudo-randomness of k_m*] This game proceeds exactly as Game $\mathbf{G}_4$ except that if no InjPK occurred then the simulator chooses the master secret k_m at random instead of computing it using the pseudo-random function $\mathtt{PRF}$. This can be done since the secret seed (given by the pre-master secret k_p) used in the computation is uniformly distributed. Note that if InjPK has occurred then these modifications are not applied, i.e., the simulator proceeds as specified in the protocol. Due to the pseudo-randomness of $\mathtt{PRF}$ we obtain

$$|\Pr[\mathsf{Win}_5] - \Pr[\mathsf{Win}_4]| \leq q\mathsf{Adv}^{prf}_{\mathtt{PRF}}(\kappa). \tag{5}$$

Game $\mathbf{G}_6$. [*Pseudo-randomness of k_1 and k_2*] This game proceeds exactly as Game $\mathbf{G}_5$ except that if no InjPK occurred then the simulator chooses $k_1|k_2$ at random instead of computing it using $\mathtt{PRF}$. Note that the master secret k_m is already uniform if InjPK has not occurred (according to Game $\mathbf{G}_5$). On the other hand, if InjPK has occurred then $k_1|k_2$ are computed as specified in the protocol. Due to the pseudo-randomness of $\mathtt{PRF}$ we obtain

$$|\Pr[\mathsf{Win}_6] - \Pr[\mathsf{Win}_5]| \leq q\mathsf{Adv}^{prf}_{\mathtt{PRF}}(\kappa). \tag{6}$$

This game implies that k_m as well as $k_1|k_2$ is fresh for every new session.

Game $\mathbf{G}_7$. [*Hash collision of h_1*] This game proceeds exactly as Game $\mathbf{G}_6$ except that the simulation aborts if during the interaction Δ computes the same hash value $h_1 := \mathtt{Hash}(A|B|C)$ in two different sessions. Note that Games $\mathbf{G}_1$ and $\mathbf{G}_3$ ensure that values A and B are fresh in different protocol sessions. Hence, the simulation aborts in the current game if Δ computes a hash collision. Due to the collision-resistance of $\mathtt{Hash}$ we obtain

$$|\Pr[\mathsf{Win}_7] - \Pr[\mathsf{Win}_6]| \leq q\mathsf{Succ}^{coll}_{\mathtt{Hash}}(\kappa). \tag{7}$$

Note, this game implies that h_1 is fresh for each new session.

Game $\mathbf{G}_8$. [*Pseudo-randomness of $F_{\mathcal{C}}$ and Indistinguishability of D*] This game proceeds exactly as Game $\mathbf{G}_7$ except that if no InjPK occurred then the simulator chooses $F_{\mathcal{C}}$ at random instead of computing it using PRF. This substitution implies also the randomization of the message $F_{\mathcal{C}}|\mathtt{HMAC}_{k_2}(F_{\mathcal{C}})$ encrypted in D. Due to the pseudo-randomness of PRF and IND-CPA security of (Enc, Dec) we obtain

$$|\Pr[\mathsf{Win}_8] - \Pr[\mathsf{Win}_7]| \leq q\mathsf{Adv}^{prf}_{\mathtt{PRF}}(\kappa) + q\mathsf{Adv}^{ind-cpa}_{(Enc,Dec)}(\kappa). \quad (8)$$

Note, this game implies that $F_{\mathcal{C}}$ does not leak any information about k_m and that D does not leak any information about k_1. Since $k_1|k_2$ is treated as the single output of PRF and by definitions of the security model the adversary is not allowed to reveal either of these keys there is no need to consider secrecy of k_2 explicitly. Thus, our proof omits security of HMAC. This is not surprising since the protocol would provide the same level of security even in case where HMAC is not used.

Game $\mathbf{G}_9$. [*Hash collision of h_2*] This game proceeds exactly as Game $\mathbf{G}_8$ except that the simulation aborts if during the interaction Δ computes the same hash value $h_2 := \mathtt{Hash}(A|B|C|F_{\mathcal{C}})$ in two different sessions. As observed in Games $\mathbf{G}_1$, $\mathbf{G}_2$, and $\mathbf{G}_8$ the input to Hash is fresh for each new session the probability of such collision in this game is given by the collision-resistance of Hash, that is

$$|\Pr[\mathsf{Win}_9] - \Pr[\mathsf{Win}_8]| \leq q\mathsf{Succ}^{coll}_{\mathtt{Hash}}(\kappa). \quad (9)$$

Game $\mathbf{G}_{10}$. [*Pseudo-randomness of $F_{\mathcal{S}}$ and Indistinguishability of E*] This game proceeds exactly as Game $\mathbf{G}_9$ except that if no InjPK occurred then the simulator chooses $F_{\mathcal{S}}$ at random instead of computing it using PRF. By the same argument as in Game $\mathbf{G}_8$ we obtain

$$|\Pr[\mathsf{Win}_{10}] - \Pr[\mathsf{Win}_9]| \leq q\mathsf{Adv}^{prf}_{\mathtt{PRF}}(\kappa) + q\mathsf{Adv}^{ind-cpa}_{(Enc,Dec)}(\kappa). \quad (10)$$

Note, this game implies that also server messages do not leak any information about k_m and $k_1|k_2$.

Game $\mathbf{G}_{11}$. [*Indistinguishability of F*] This game proceeds exactly as Game $\mathbf{G}_{10}$ except that if no InjPK occurred then the simulator chooses a random value $\beta \in \{0,1\}^{p_2(\kappa)}$ and computes $F := Enc_{k_1}(\beta, \mathtt{HMAC}_{k_2}(\beta))$. Considering IND-CPA security of (Enc, Dec) we obtain

$$|\Pr[\mathsf{Win}_{11}] - \Pr[\mathsf{Win}_{10}]| \leq q\mathsf{Adv}^{ind-cpa}_{(Enc,Dec)}(\kappa). \quad (11)$$

Note, this game implies that message F does not leak any information about the cookie cky to an outsider adversary if cky is revealed to $\mathcal{S}$ upon the successful verification of the SLSO policy.

Game $\mathbf{G}_{12}$. [*Impersonation of $(\mathcal{U}, \mathcal{B})$*] This game proceeds exactly as Game $\mathbf{G}_{11}$ except that the simulation aborts if no InjPK occurred *and* there has been a Send query addressed to $\mathcal{S}$ containing some F which has not been previously returned by Δ on behalf of $\mathcal{B}$ and which leads to the acceptance by $\mathcal{S}$ (i.e., if $\mathcal{A}$ impersonating the client causes the server to accept after the successful verification of cky). As noticed in Game

$\mathbf{G}_{11}$ the encryption F does not leak any information about cky if revealed to an authenticated server; moreover, as implied previously by Game $\mathbf{G}_6$ the keys $k_1|k_2$ computed by the browser $\mathcal{B}$ are fresh for each new session. This also excludes replay attacks on the ciphertext F. Therefore, the occurrence of the mentioned Send query addressed to $\mathcal{S}$ is upper-bounded by the probability of forgery of F which in turn is upper-bounded by the probability of a random guess from $\{0,1\}^{p_2(\kappa)}$. Thus, we have

$$|\Pr[\mathsf{Win}_{12}] - \Pr[\mathsf{Win}_{11}]| \leq \frac{q}{2^{p_2(\kappa)}}. \tag{12}$$

In fact this game implies that if a server instance accepts in the execution of `BBMA-SLSO` then there exists a corresponding partnered client instance, i.e., `BBMA-SLSO` satisfies the second requirement of Definition 3.

Game $\mathbf{G}_{13}$. [*Indistinguishability of G*] This game proceeds exactly as Game $\mathbf{G}_{12}$ except that if no InjPK occurred then the simulator chooses a (random) value $\gamma \in \mathcal{W} \setminus \mathcal{W}^*$ and computes $G := Enc_{k_1}(\gamma, \mathtt{HMAC}_{k_2}(\gamma))$. We assume that this choice is done in cooperation with the user $\mathcal{U}$, i.e., $\mathcal{U}$ advises Δ which γ to choose so that $\mathcal{U}$ knows which HPA should be accepted in the simulated protocol run. Note that the goal of this game is to ensure that message G does not leak any information about the original HPA w used in `BBMA-SLSO` to an outsider adversary if w is sent over the TLS channel to the authenticated client $(\mathcal{U}, \mathcal{B})$. Therefore, it is important that γ is chosen such that $\mathcal{U}$ can distinguish it from w, i.e., from the set $\mathcal{W} \setminus \mathcal{W}^*$. Considering IND-CPA security of (Enc, Dec) we obtain

$$|\Pr[\mathsf{Win}_{13}] - \Pr[\mathsf{Win}_{12}]| \leq q\mathsf{Adv}^{ind-cpa}_{(Enc,Dec)}(\kappa). \tag{13}$$

Game $\mathbf{G}_{14}$. [*Impersonation of $\mathcal{S}$*] This game proceeds exactly as Game $\mathbf{G}_{13}$ except that the simulation aborts if during the interaction there is a client instance which accepts but there exists no partnered server instance, i.e., the first condition for the adversarial success from Definition 3 is satisfied. Note that according to the protocol specification $\mathcal{U}$ accepts after having recognized the authenticator w. Obviously, the adversary succeeds only if $\mathcal{B}$ has received a Send query containing a message of the form G which has not been previously returned by (the simulator on behalf of) $\mathcal{S}$ such that the decryption of G to $(w'|\mu_G)$ and rendering of w' results in the acceptance by $\mathcal{U}$. Note that in `BBMA-SLSO` browser $\mathcal{B}$ does not abort if the verification of the SLSO policy fails, or if no cookie has been previously set. This in turn allows the malicious server to proceed with the protocol run trying to influence $\mathcal{U}$ to accept the communication. Having excluded leakage of information concerning $k_1|k_2$ in previous games we follow that the success probability of $\mathcal{A}$ is conditioned by the occurrence of InjPK (by which $\mathcal{A}$ learns k_1 and k_2) and is given by the probability of $\mathcal{A}$ to find some perfectly human-indistinguishable authenticator $w^* \in \mathcal{W}^*$ which will then be visualized to $\mathcal{U}$. Obviously, the success probability of $\mathcal{A}$ in finding such HPA depends on the size of $\mathcal{W}^*$. Thus, this is precisely the point in our security analysis of `BBMA-SLSO` where human skills to distinguish the authenticators become important. We get

$$|\Pr[\mathsf{Win}_{14}] - \Pr[\mathsf{Win}_{13}]| \leq \frac{q|\mathcal{W}^*|}{|\mathcal{W}|}. \tag{14}$$

Note that for *sufficiently small* set $\mathcal{W}^*$, i.e., better ability of the human user to distinguish between two different authenticators this excludes attacks resulting in the impersonation of the server towards $\mathcal{U}$. Since Game $\mathbf{G}_{12}$ has excluded the adversarial impersonation of $(\mathcal{U},\mathcal{B})$ towards $\mathcal{S}$, and since this game excludes the impersonation of malicious $\mathcal{S}$ towards $(\mathcal{U},\mathcal{B})$ we follow that the probability of $\mathcal{A}$ to win in this game is given by $\Pr[\mathsf{Win}_{14}] = 0$.

Considering Equations (1) to (14) we get the desired inequality

$$\begin{aligned}\mathsf{Succ}^{bbma}_{\mathtt{BBMA-SLSO}}(\kappa) &= \Pr[\mathsf{Win}_0] \\ &\leq \frac{q|\mathcal{W}^*|}{|\mathcal{W}|} + \frac{3q^2}{2^{p_5(\kappa)}} + \frac{q^2}{2^{p_3(\kappa)}} + \frac{q}{2^{p_2(\kappa)}} + \mathsf{Adv}^{ind-cpa}_{(\mathcal{E},\mathcal{D})}(\kappa) + \\ &\qquad 4q\mathsf{Adv}^{ind-cpa}_{(Enc,Dec)}(\kappa) + 4q\mathsf{Adv}^{prf}_{\mathtt{PRF}}(\kappa) + 2q\mathsf{Succ}^{coll}_{\mathtt{Hash}}(\kappa).\end{aligned}$$

Remark 1. Although not stated in Theorem 1 explicitly, the security proof of `BBMA-SLSO` based on the current TLS standard is valid in the Random Oracle Model (ROM) [4]. The reason is that the specification of TLS prescribes the use of the RSA encryption according to PKCS#1 (a.k.a. RSA-OAEP) which in turn is known to provide IND-CPA security in ROM (see [29] and [18] for the proof of the general construction and the TLS-specific construction, respectively). On the other hand, Theorem 1 assumes $(\mathcal{E},\mathcal{D})$ to be IND-CPA secure (independent of ROM). Hence, whether `BBMA-SLSO` is secure under standard assumptions or not heavily relies on the assumptions underlying the security of $(\mathcal{E},\mathcal{D})$.

Remark 2. Another look on Theorem 1 reveals that the success probability of the adversary strongly depends on the size of $\mathcal{W}^*$, i.e., the set of authenticators that are perfectly human-indistinguishable from the HPA w used in the `BBMA-SLSO` protocol. In fact the protocol is secure if the size of $\mathcal{W}^*$ is *sufficiently small* such that the factor $q|\mathcal{W}^*|/|\mathcal{W}|$ can be seen as negligible. This happens in case that the chosen HPA is good and is precisely what makes `BBMA-SLSO` user-aware.

Remark 3. As already mentioned during the description of `BBMA-SLSO` the `HMAC` construction used in the standard specification of the TLS protocol, formally, does not play any role for the security of the protocol. This is not surprisingly since every output of `HMAC` is encrypted using session key k_1 before being sent over the network. Since $k_1|k_2$ is treated as a single output of `PRF` the separation into k_1 and k_2 can be seen as redundant from the theoretical point of view. Note also that Krawczyk has proved the MAC-then-encrypt construction as secure in [22]. Though he mentions some problems in the general construction he shows that they do not apply to TLS.

5 Conclusion

Authenticating the user on the Web is an essential primitive and target to various attacks. We have introduced and analyzed a cookie-based authentication protocol `BBMA-SLSO` that makes very weak assumptions on user's skills and requires little modifications of the browser security model to enforce the SLSO policy in order to be provably secure.

The protocol is specifically designed for security-unaware users who wish to identify Web sites through some easy-to-recognize indicators. The main assumption underlying the protocol security is that good HPAs w exist for which the size of their perfectly human-indistinguishable set $\mathcal{W}^*$ remains sufficiently small (for most of the users). It remains an open question, to find such HPAs. We have conjectured that good HPAs might be found among images, audio and video sequences. For example, a personal image taken during own summer vacation and extended with some additional personal text using graphic editor might be better recognizable than an image without such text. However, extensive usability experiments in this interesting research direction have still to be conducted. Nevertheless, the presented protocol is another step towards bridging the gap between protocols that are provably secure, interfaced to users who are prone to errors, and implementable within the design constraints of standard browsers.

References

1. C. Allen and T. Dierks. The TLS Protocol — Version 1.1. Internet proposed standard RFC 4346, 2006.
2. M. Bellare and C. Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In *ASIACRYPT'00*, LNCS 1976, pp. 531–545. Springer, 2000.
3. M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *CRYPTO'93*, LNCS 773, pp. 232–249. Springer, 1993.
4. M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM CCS'93)*, pp. 62–73. ACM Press, 1993.
5. S. Chiasson, P. C. van Oorschot, and R. Biddle. Graphical Password Authentication Using Cued Click Points. In *ESORICS'07*, LNCS 4734, pp. 359–374. Springer, 2007.
6. R. Dhamija and J. D. Tygar. The Battle against Phishing: Dynamic Security Skins. In *SOUPS'05*, pp. 77–88. ACM Press, 2005.
7. R. Dhamija, J. D. Tygar, and M. A. Hearst. Why Phishing Works? In *CHI'06*, pp. 581–590. ACM Press, 2006.
8. P.-A. Fouque, D. Pointcheval, and S. Zimmer. HMAC is a Randomness Extractor and Applications to TLS. In *ACM ASIACCS'08*, pp. 21-32. ACM Press, 2008.
9. A. O. Freier, P. Kariton, and P. C. Kocher. The SSL Protocol: Version 3.0. Internet draft, Netscape Communications, 1996.
10. S. Gajek, S. Schwenk. Revising the Mature Browser Security Model. Technical Report, HGI TR-2008-003, 2008.
11. S. Gajek, M. Manulis, A.-R. Sadeghi, J. Schwenk. Provably Secure Browser-Based User-Aware Mutual Authentication over TLS. In *ACM ASIACCS'08*, pp. 300–311. ACM Press, 2008.
12. T. Groß. Security Analysis of the SAML Single Sign-on Browser/Artifact Profile. In *ACSAC'03*, pp. 298–307. IEEE CS, 2003.
13. T. Groß, B. Pfitzmann, and A.-R. Sadeghi. Browser Model for Security Analysis of Browser-Based Protocols. In *ESORICS'05*, LNCS 3679, pp. 489–508. Springer, 2005.
14. T. Groß, B. Pfitzmann, and A.-R. Sadeghi. Proving a WS-Federation Passive Requestor Profile with a Browser Model. In *SWS'05*, pp. 54–64. ACM Press, 2005.
15. C. Jackson, A. Barth, A. Bortz, W. Shao, and D. Boneh. Protecting Browsers from DNS Rebinding Attacks. In *CCS'07*, pp. 421–431. ACM Press, 2007.

16. C. Jackson, D. R. Simon, D. S. Tan, and A. Barth. An Evaluation of Extended Validation and Picture-in-Picture Phishing Attacks. In *FC'07/USEC'07*, LNCS 4886, pp. 281–293. Springer, 2008.
17. M. Jakobsson and S. Myers. Delayed Password Disclosure. In *IJACT*, 1(1):47–59, 2008.
18. J. Jonsson and B. S. Kaliski. On the Security of RSA Encryption in TLS. In *CRYPTO'02*, LNCS 2442, pp. 127–142. Springer, 2002.
19. C. Karlof, U. Shankar, J. D. Tygar, and D. Wagner. Dynamic Pharming Attacks and Locked Same-Origin Policies for Web Browsers. In *ACM CCS'07*, pp. 58–71. ACM Press, 2007.
20. D. Kormann and A. Rubin. Risks of the Passport Single SignOn Protocol. *Computer Networks*, 33(1–6):51–58, 2000.
21. Microsoft Corporation. Mitigating Cross-Site Scripting with HTTP-only Cookies. `http://msdn2.microsoft.com/en-us/library/ms533046.aspx`, 2008
22. H. Krawczyk. The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?). In *CRYPTO'00*, LNCS 2139, pp. 310–331. Springer, 2000.
23. C. Mason, K.-H. Baek, and S. Smith. WSKE: Web Server Key Enabled Cookies. In *FC'07/USEC'07*, LNCS 4886, pp. 294–306. Springer 2008.
24. J. C. Mitchell, V. Shmatikov, and U. Stern. Finite-State Analysis of SSL 3.0. In *USENIX Security Symp.*, pp. 201–216, 1998.
25. L. C. Paulson. Inductive Analysis of the Internet protocol TLS. *ACM Trans. on Comp. and Syst. Sec.*, (3):332–351, 1999.
26. B. Pfitzmann and M. Waidner. A Model for Asynchronous Reactive Systems and its Application to Secure Message Transmission. In *IEEE S&P'01*, pp. 184–200. IEEE CS, 2001.
27. S. E. Schechter, R. Dhamija, A. Ozment, and I. Fischer. The Emperor's New Security Indicators. In *IEEE S&P'07*, pp. 51–65. IEEE CS, 2007.
28. B. Schneier and D. Wagner. Analysis of the SSL 3.0 protocol. In *USENIX Workshop on Electronic Commerce*, 1996.
29. V. Shoup. OAEP Reconsidered. *Journal of Cryptology*, 15(4):223–249, 2002.
30. V. Shoup. Sequences of Games: A Tool for Taming Complexity in Security Proofs. Cryptology ePrint Archive, Report 2004/332, 2006. `http://eprint.iacr.org/2004/332.pdf`.
31. C. Soghoian and M. Jakobsson. A Deceit-Augmented Man In The Middle Attack Against Bank of America's SiteKey Service. 2007. `http://paranoia.dubfire.net/2007/04/deceit-augmented-man-in-middle-attack.html`.
32. X. Suo, Y. Zhu, and G. S. Owen. Graphical Passwords: A Survey. In *Ann. Comp. Sec. Applic. Conf.*. IEEE CS, 2005.
33. W3C. Document Object Model (DOM), 2005. `http://www.w3.org/DOM`.