Post-Session Authentication

Naveed Ahmed and Christian D. Jensen

Technical University of Denmark, Copenhagen
{naah,Christian.Jensen}@imm.dtu.dk

Abstract. Entity authentication provides confidence in the claimed identity of a peer entity, but the manner in which this goal is achieved results in different types of authentication. An important factor in this regard is the order between authentication and the execution of the associated session. In this paper, we consider the case of post-session authentication, where parties authenticate each other at the end of their interactive session. This use of authentication is different from session-less authentication (e.g., in RFID) and pre-session authentication (e.g., for access control.)

Post-session authentication, although a new term, is not a new concept; it is the basis of at least a few practical schemes. We, for the first time, systematically study it and present the underlying authentication model. Further, we show that an important class of problems is solvable using post-session authentication as the only setup assumption. We hope post-session authentication can be used to devise new strategies for building trust among strangers.

1 Introduction

Entity authentication is an important requirement for the security of interactive protocols, because if a party does not know with whom it is communicating then there is little left what one can achieve in terms of security. Whereas authentication may seem a simple concept, it is one of the most confusing goals in the security analysis [11]—even its operational definition[1] is not agreed upon.

Nevertheless, most security experts do agree that authentication does not correspond to one monolithic goal [8, 14]. To us, the term refers to a set of fine level authentication goals (FLAGs) [16, 9]. A few examples of FLAGs are identification, recognition, operativeness and willingness. For an entity A that authenticates a peer entity B, identification assures that A is able to compute the correct identity of B, while recognition makes sure that A is able to recognize B as the party with whom it has communicated before [23]. Similarly, the operativeness assures A that B is currently there at the far-end, and the

[1] A conceptual definition is often in a natural language capturing the meaning and the use of a concept. An operational definition represents a computational procedure that provides *yes* or *no* answer corresponding to the presence or absence of the concept in a given system.

willingness makes sure that B is aware that it is being authenticated. Since different protocols achieve different sets of these fine level goals, the interpretation of authentication varies.

A party always uses authentication as a service in an application. In Lowe's words [8], "the appropriate authentication requirement will depend upon the use to which the protocol is put." We distinguish between the three classes of use-cases corresponding to the execution order of an authentication protocol and the authentication-dependent interactive session. The first class represents session-less authentication, e.g., RFID [20] and simple entity authentication [12]. In this class, the result of authentication is used by a system to update its state, e.g., a back-end database. Although the authentication result is not used for the other types of interaction, the result may influence how authentication is carried out subsequently, e.g., see the synchronization approach [20].

The second class represents pre-session authentication, which is the most common use of authentication. Here, the result of authentication is used in a subsequent session. For example, when a person logs in on a computer, the operating system uses the authentication result, the person's identity, to launch his session, and all access control decisions in the session essentially depend on it.

The third class, which is relatively less common, is post-session authentication, where authentication is carried out at the end of the associated session. Authenticating the parties when a session is already over may not seem so useful, but the following observations make this case worth considering. Firstly, if an instance of post-session authentication fails then parties can always reject the output of the session. Secondly, post-session authentication allows parties to anonymously interact in the session and build a trust level before authenticating each other, e.g., two spies may want to engage in such a session before revealing their identities. For online shoppers, this type of authentication could be attractive because it provides a kind of assurance that vendors are not using any user-dependent pricing strategy. Similarly, mutually distrustful parties can anonymously engage in an auction for a precious item, while keeping the thieves among the bidders at bay.

In a general model of post-session authentication, parties engage in an arbitrary distributed computation, and at the end they authenticate each other in the context of this computation. Clearly, an adversary can trivially take part in the computation, but, at the end, the adversary can not authenticate himself as a legitimate participant if the authentication protocol is secure.

Because the execution of a post-session authentication protocol is session-dependent, its requirements are clearly more stringent than a session-less authentication protocol. On the other hand, in the pre-session case, we may also need to protect the confidentiality of some protocol terms (e.g., a session key), in order to protect the integrity of the subsequent session. Sometimes a hybrid form of authentication is used, e.g., continuous authentication in Auth-SL [26], which may depend on a previous session, can be used for authorizing the access to a protected resource in a later session.

The rest of the paper is arranged as follows. A few motivating examples are presented in § 2. In § 3, we present our authentication model, and then in § 4 we demonstrate a plausibility result, namely session-less authentication, in principle, can be used to compute any multi-party function. In § 5, we briefly discuss some other interesting aspects of post-session authentication, followed by a summary of the related work in § 6 and concluding remarks in § 7.

2 Examples of Post-session Authentication

In this section, we present four examples. The reader must note that a session does not necessarily include all of the messages exchanged between two parties. A session may represent a part of such interaction, such as the initial or middle part, depending on the interdependency of authentication and exchanged messages.

Probably, the first known application of post-session authentication is in PGPfone [1], which uses the method of numeric comparison for authentication. As we know, against an active attacker, Diffie-Hellman key-agreement (DHKA) [21] can only provide confidentiality of the key if the man-in-middle scenario can be rejected. PGPfone use DHKA to establish a call. A hash value of the transcript of the key-agreement phase is computed and converted to numeric values at the both ends. Then, the two parties authenticate each other by simply reading off their respective numeric values.

The second example is of the Cocaine Auction (CA) protocol [3]. Its setup uses anonymous broadcast to carry out an English style auction among untrustworthy parties. The classic allegory for the protocol is as follows.

Consider a number of dealers gathered around a table. One of the dealers, the seller, offers his next shipment of cocaine to the the highest bidder, and he starts the auction by proposing an initial bid. It is required that the bidders remain anonymous to each other as well as to the seller. Also, the winner anonymously arranges a secret appointment with the seller, to receive the goods and to pay the bid. In the scenario described above, none of the parties completely trusts in any other party. There is no party that can act as a trusted arbitrator. For anonymity, even the seller should not able to find out the identity of the winner before committing the sale. The way the protocol achieves these goals is a good example of post-session authentication.

As the third example, we describe a secure communication protocol. Alice wishes to securely and anonymously communicate with her friend Bob over the Internet, but no public key infrastructure (PKI) is available to them, and neither they posses a common secret key. They can use the Diffie-Hellman protocol [21] to compute a common secret, but the protocol does not provide any authentication. Similarly, sending each other their public keys is of no use in absence of a common certification authority. Authenticating each other using biometrics, e.g., voice or video, is not good for anonymity, because a man-in-middle can then easily identify the two parties. In such a restrictive scenario, they can use the following post-session authentication protocol.

Alice and bob start an unauthenticated session, while also realizing the possibility of a man-in-middle attack. In this session, they present each other with a series of challenges, such that answering these challenges require the knowledge of their private interactions in the past. During the session they do not reveal the answers to these challenges. For robustness, we can rely on the challenges with *yes* (binary 1) or *no* (binary 0) answers. At the end of the session, the shared secret is computed by concatenating the answers of these challenges.

Once the shared secret is computed at the both ends, Alice and Bob can use this secret as a cryptographic key to authenticate each other using a suitable symmetric-key authentication protocol [12]. This particular configuration of the initial session and the authentication protocol clearly fits in the post-session authentication class, because the success of the protocol execution inevitably depends on the output of the preceding session.

The fourth example is the ordering system on an Amazon web store [22]. When a new user visits the web store, the Amazon web server stores a cookie containing a session ID in the user's machine. Although the user is completely anonymous to the web store, the web store maintains the temporary database record for the user's session, which is addressable by the session ID in the cookie. As a result, the user can conveniently explore the store, compare prices, select vendors, and manage the shopping basket allocated to the session.

When the user opts for the check out, the web server asks her to sign up with Amazon using her email address, which works as a pseudonym for the user, and then the user is asked to provide a shipping address and a valid payment option. In this way, the web store allows to complete all of the shopping activity except the actual payment without requiring any prior authentication.

As in the above example, the authentication is not necessary for the initial interaction, as long as a web store is willing to commit resources and allocate a unique shopping basket to every potential buyer. At the same time, it is also required that if the buyer proceeds to check-out then the result of her initial interaction—the basket—can some how be linked to the subsequent authentication during the store's check-out phase. On the other hand, note that this case of post-session authentication can well be implemented in the manner of pre-session authentication by forcing each potential buyer to sign up first, however, doing so may not be a good business strategy.

3 Model of Authentication

Authentication is what an authentication protocol does is a dangerous approach[2]. Many security models for protocols are not expressive enough to capture the authentication requirements (see § 6). Often, authentication properties are expressed in an indirect way, e.g., in terms of protocol messages [15] or runs [8]. We use our binding sequence based framework [9, 17] to model post-session authentication.

[2] by Dieter Gollmann in an invited talk

Before going into the details of the actual model, we first motivate the reader by listing some of its advantages. The framework allows simple definitions of authentication goals, based on the notion of distinguishability. It is relatively straight forward to express all interesting authentication goals (which we refer to as FLAGs) in this framework. The framework allows to validate the security and the correctness goals of a protocol independently.

The framework was originally used to model the session-less scenario [9], but the extension of that model to post-session authentication turns out to be quite straight forward and only requires the inclusion of one additional clause to the operational definitions of authentication goals.

Summary of Session-less Authentication Model [9]

A set of FLAGs represents a possible set of correctness requirements for an authentication protocol. Two authentication protocols are functionally different for a calling routine (who may use an authentication protocol as a a service) if their sets of FLAGs are different. Of course, to achieve a certain FLAG, different protocols may employ different cryptographic techniques, e.g., public-key vs. symmetric-key ciphers, and nonces vs. time-stamps.

Let X_c represents the local entity for which a FLAG is being defined, and X_j and X_l are two other network entities, s.t. $c \neq j \neq l$. Let G be a variable on FLAGs.

- RCOG$[X_c \rhd X_j] \stackrel{\text{def}}{=}$ If X_c verifies that X_j is the same entity that once existed then X_c is said to achieve the goal *recognition* for X_j.
- IDNT$[X_c \rhd X_j] \stackrel{\text{def}}{=}$ If X_c verifies that X_j can be linked to a record in a pre-specified identification database then X_c is said to achieve the goal identification[3] for X_j.
- OPER$[X_c \rhd X_j] \stackrel{\text{def}}{=}$ If X_c verifies that X_j currently exists on the network then X_c is said to achieve the goal operativeness for X_j.
- WLNG$[X_c \rhd X_j] \stackrel{\text{def}}{=}$ If an entity X_c verifies that once X_j wanted to communicate to X_c then X_c is said to achieve willingness for X_j.
- PSATH$[X_c \rhd X_j] \stackrel{\text{def}}{=}$ Pseudo single-sided authentication is achieved if an entity X_c verifies that a peer entity X_j, with a pseudonym $pid(X_j)$, is currently ready to communicate with X_c.
- SATH$[X_c \rhd X_j] \stackrel{\text{def}}{=}$ Single-sided authentication is achieved if an entity X_c verifies that a peer entity X_j, with the identification j, is currently ready to communicate with X_c.
- CNFM$[X_c \rhd X_j, G] \stackrel{\text{def}}{=}$ If an entity X_c verifies that the peer entity X_j knows that X_c has achieved a FLAG G for X_l then X_c is said to achieve the goal confirmation on G from X_j.

[3] Further, if that record cannot be used to feasibly recover the identity j then it is qualified as anonymous identification. For brevity, we do not include the anonymity aspect in this exposition, but it is trivial to write the anonymous versions of FLAGs.

SSATH$[X_c \rhd X_j] \stackrel{\text{def}}{=}$ Strong single-sided authentication is achieved by X_c for X_j if X_c has the confirmation on the *single-sided authentication* for X_j from X_j.

MATH$[X_c \rhd X_j] \stackrel{\text{def}}{=}$ If an entity X_c verifies that both parties (X_c and the peer entity X_j) currently want to communicate with each other, then X_c is said to achieve mutual authentication.

The above list of FLAGs is based on our experience. The FLAGs as presented above are independent of any protocol or any security model [5, 14] and only capture the natural use of these terms. Now, we turn to the operational definitions of FLAGs, in order to provide computational procedures corresponding to whether or not certain FLAGs are achieved in the operational settings of an authentication protocol Π. A central concept in this regard is of a *binding sequence*.

Binding Sequence: A binding sequence β_{X_c} is a list of received messages in the protocol transcript of an entity X_c, such that the messages are guaranteed to be sent by honest parties.

A *binding sequence* can be replayed; only an unauthorized change in the list is not possible without being detected by X_c. For example, the list of received encrypted messages $[\{N_c\}_{K_c}, \{N_c + 1\}_{K_c}]$, where K_c is the public key of X_c, cannot be changed[4] by a man-in-middle without the possibility of being detected by X_c, although X_c may not know who is at the far-end and whether the list is being replayed. In the literature, sometimes such a property for an individual message is called the *message integrity*.

In the following, X_c can distinguish between two instances of a *binding sequence* if a distinguisher algorithm $\mathcal{D}(C_b, \lambda)$ (which runs in a polynomial time in the length of its input) can be constructed on X_c. Here, C_b is a challenge picked by X_c and is either C_0 or C_1; and λ is an auxiliary input, such as a decryption key. The distinguisher correctly outputs 0 or 1 corresponding to C_0 and C_1, with a high probability.

RCOG$(X_c \rhd X_j, \beta_{X_c}(i)) \stackrel{\text{def}}{=}$ Let $\beta_{X_c}(i)$, $\beta_{X_c}(i')$ and $\beta_{X_c}(i'')$ be generated when X_c executes Π with X_j, X_l and X_j respectively, as shown in Fig. 1. Let the two challenges be $C_0 = (\beta_{X_c}(i), \beta_{X_c}(i'))$ and $C_1 = (\beta_{X_c}(i), \beta_{X_c}(i''))$. If there exists $\mathcal{D}^{rcog}(C_b, \lambda)$ on X_c for all choices of j and l then X_c is said to achieve the goal *recognition* of X_j from $\beta_{X_c}(i)$.

IDNT$(X_c \rhd X_j, \beta_{X_c}(i)) \stackrel{\text{def}}{=}$ Same as RCOG$(X_c \rhd X_j, \beta_{X_c}(i))$ except the distinguisher $\mathcal{D}^{idnt}(C_b, \lambda)$ gets a read-only access to an identification database containing the identification records of all network entities, as a part of its auxiliary input λ.

OPER$(X_c \rhd X_j, \beta_{X_c}(i)) \stackrel{\text{def}}{=}$ Let $\beta_{X_c}(i)$ and $\beta_{X_c}(i')$ be generated when X_c executes Π twice with X_j, as shown in Fig. 1. Let the two challenges be

[4] Standard assumptions apply: the public-key encryption scheme is secure, and the private key is only known to X_c.

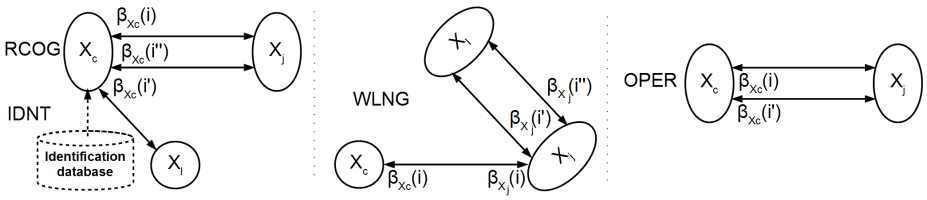

Fig. 1. Distinguishability Setups for FLAGs

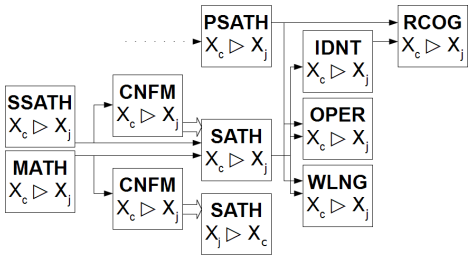

Fig. 2. Relations among FLAGs

$C_0 = \beta_{X_c}(i)$ and $C_1 = \beta_{X_c}(i')$. If there exists $\mathcal{D}^{oper}(C_b, \lambda)$ on X_c for all runs with X_j then X_c is said to achieve the goal *operativeness* for X_j.

$\mathrm{WLNG}(X_c \rhd X_j, \beta_{X_c}(i)) \stackrel{\text{def}}{=}$ If $\beta_{X_c}(i)$ is generated on X_c in a *run* involving X_c and X_j, as shown in Fig. 1, then $\mathrm{IDNT}(X_j \rhd X_c, \beta_{X_j}(i)) \Rightarrow \mathrm{WLNG}(X_c \rhd X_j, \beta_{X_c}(i))$, where $\beta_{X_j}(i)$ consists of all those messages from $\beta_{X_c}(i)$ in which X_j is a peer entity.

$\mathrm{PSATH}(X_c \rhd X_j, \beta_{X_c}(i)) \stackrel{\text{def}}{=} \mathrm{WLNG}(X_c \rhd X_j, \beta_{X_c}(i)) \wedge \mathrm{OPER}(X_c \rhd X_j, \beta_{X_c}(i)) \wedge \mathrm{RCOG}(X_c \rhd X_j, \beta_{X_c}(i))$

$\mathrm{SATH}(X_c \rhd X_j, \beta_{X_c}(i)) \stackrel{\text{def}}{=}$
$\mathrm{WLNG}(X_c \rhd X_j, \beta_{X_c}(i)) \wedge \mathrm{OPER}(X_c \rhd X_j, \beta_{X_c}(i)) \wedge \mathrm{IDNT}(X_c \rhd X_j, \beta_{X_c}(i))$

$\mathrm{CNFM}(X_c \rhd X_j, \beta_{X_c}(i), G) \stackrel{\text{def}}{=} \mathrm{RCOG}(X_c \rhd X_j, \beta''_{X_c}(i)) \wedge \mathrm{OPER}(X_c \rhd X_j, \beta''_{X_c}(i)) \wedge \mathrm{G}(X_c \rhd X_j, \beta'_{X_c}(i))$,
where $\beta_{X_c}(i) = \beta'_{X_c}(i) || \beta''_{X_c}(i)$ (|| stands for concatenation).

$\mathrm{SSATH}(X_c \rhd X_j, \beta_{X_c}(i)) \stackrel{\text{def}}{=} G \wedge \mathrm{CNFM}(X_c \rhd X_j, \beta_{X_c}(i), G)$,
where $G = \mathrm{SATH}(X_c \rhd X_j, \beta_{X_c}(i))$

$\mathrm{MATH}(X_c \rhd X_j, \beta_{X_c}(i)) \stackrel{\text{def}}{=} \mathrm{SATH}(X_c \rhd X_j, \beta_{X_c}(i)) \wedge \mathrm{CNFM}(X_c \rhd X_j, \beta_{X_c}(i), G)$,
where $G = \mathrm{SATH}(X_j \rhd X_c, \beta_{X_j}(i))$

The hierarchical relations between FLAGs that are valid (by definition) are shown in Fig. 2. *Identification*, *willingness* and *recognition* do not have any timeliness property. *Operativeness* and *willingness* do not require the knowledge of the identity (or pseudo identity) of a peer entity. The goal *confirmation* can be applied to any other goal, e.g., a confirmation on MATH may be regarded as a

stronger form of *mutual authentication*. *Identification*, *operativeness*, and *single-sided authentication* are respectively comparable to *aliveness*, *recent aliveness* [8], and *strong entity authentication* [12].

Extension to Post-Session Class

For the post-session class, the definition of security—the binding sequence—remains the same. We extend the correctness requirements, by including a post-session clause in the operational definitions of FLAGs, to meet the requirement that authentication should only succeeds in the context of a session. Before introducing the new clause, we first elaborate the notion of a session itself.

In our model, a session only refers to the interactive part of a distributed computation, in which a number of parties interact with each other by passing messages over unreliable channels. Whether the computation is secure if some of the parties are dishonest [19], and whether the computation meets its functional requirements, are the concerns that are beyond the scope of our notion of (interactive) session[5]. Similarly, in the part of computation that is carried out locally by a party, all components and communication between the components are assumed to be trusted (cf. secure information flow.)

Next, we consider the specification of a session, so that it can be used in our computational model. On the one hand, one may need to specify a session at the level of primitive communication steps of interactive Turing machines. On the other hand, specifying a session by the end result of a computation may suffice. The right level of specification is certainly application dependent. We abstract away from this decision by defining the computational interpretation of a session in the following way.

Session: An i_{th} session $\Psi_i = f(\tau_i, .)$ is a set of terms computed by a party from the transcript of its interaction τ_i before the i_{th} execution of an authentication protocol, such that each Ψ_i is unique among all q sessions in the network, i.e., $\{\Psi_i : 1 \leq i \leq q\}$ is necessarily a q_{th} order set.

Depending on the required level of granularity, Ψ_i may well consists of a complete transcript of communication with the time-stamps (thus making each session trivially unique), a hash of the transcript, or a binary value distinguishing only between uncorrupted and corrupted sessions (modified by an adversary.) For our purpose, all sessions are in the set $\{\Psi_i : 1 \leq i \leq q\}$. This set will always be empty for session-less authentication.

Claim 1: An authentication protocol that achieves G is vulnerable to session hijacking if G does not depend on the session.

***proof*:** Assuming G is not a valid post-session FLAG, a generic attack is possible. Let X_1 and X_2 be the two honest parties and $\mathcal{A}$ be an adversary. Now, $\mathcal{A}$ plays a man-in-middle role while executing Ψ_1 with X_1 and Ψ_2 with X_2. At the end of these sessions, $\mathcal{A}$ simply authenticate itself to X_1 and X_2.

[5] This is why a seamless interaction with an MPC protocol is possible in § 4.

Consequently, X_1 and X_2 conclude that the prior session took place with $\mathcal{A}$. Hence, $\mathcal{A}$ is able to hijack (and claim the credit of) Ψ_1 and Ψ_2 □

To prevent session hijacking and similar problems, the distinction between *responsibility* and *credit* [27] is important. To summarize, in some applications a claimant of a session can be held responsible for the messages in the session, e.g., to make payment for an order in the last example of § 2, or the session may represent an access control policy that is to be enforced on the behalf of the claimant. In the other applications, the claimant may expect credit for the session, e.g., winning bid in an auction, monetary reward for the session containing the solution to a puzzle, or an increase in the reputation score.

There is no incentive in hijacking a session if the hijacking only implies the responsibility at a later stage, however, making some honest party responsible for an adversary's generated session can be a real threat. On the other hand, if the claim on a session means some credit then certainly hijacking is well motivated.

Therefore, we further qualify the session computing function $f(\tau_i, .)$ in the definition of a session: if a session implies some credit (possibly in combination with responsibility) then we require that the inverse function $\tau_i = f^{-1}(\Psi_i, .)$ is a one-way function. There are several ways to meet this requirement, e.g., by employing a Diffie-Hellman type construction [1] or a nonce based commitment [3]. In the following, we assume that this requirement is always met. For a session that only leads to the responsibility, there is no such requirement.
Now, we extend the operational definitions of FLAGs to express the requirement of post-session authentication.

Post-Session Clause: G is a valid post-session FLAG *if* finding a pair of sessions Ψ_i and Ψ_j is infeasible such that, in a given run of authentication protocol, G can be validated in the run for both Ψ_i and Ψ_j.

Intuitively, if a party can derive a FLAG from its binding sequence after Ψ_i but the same FLAG can not be validated independent of Ψ_i, then this FLAG is a valid post-session FLAG for the party. To illustrate how the proposed extension works, let us once again consider the third example from § 2; the abstract narrations of the protocol are as follows.

1. Interactive Session:
   $X_{alice} \rightarrow X_{bob} : [c_i : 1 \leq i \leq |K|/2]$ (Alice sends her set of challenges.)
   $X_{bob} \rightarrow X_{alice} : [c_i : |K|/2 < i \leq |K|]$ (Bob sends his set of challenges.)
2. Computation of Session:
   on $X_{bob} : \Psi_{bob} = K_{bob} \leftarrow [c_i : 1 \leq i \leq |K|]$ (Compute Bob's version of key.)
   on $X_{alice} : \Psi_{alice} = K_{alice} \leftarrow [c_i : 1 \leq i \leq |K|]$ (Compute Alice's version of key.)
2. Authentication :
   $X_{alice} \rightarrow X_{bob} : N_{alice}$
   $X_{bob} \rightarrow X_{alice} : N_{bob}, \{N_{alice}, N_{bob}, Alice\}_{K_{bob}}$
   $X_{alice} \rightarrow X_{bob} : \{N_{bob}, N_{alice}\}_{K_{alice}}$

In this example, the authentication protocol (ISO/IEC 9798-2) can not succeed without the same session at both ends, i.e., the authentication succeeds only if Alice's key K_{alice} and Bob's key K_{bob} are equal. When the same session is used, Alice and Bob achieve a certain set of FLAGs; this set can be computed using the operational definitions of FLAGs.

Consider the operativeness of Bob: OPER($X_{alice} \triangleright X_{bob}, \beta_{alice}$), where $\beta_{alice} = [\{N_{alice}, N_{bob}, Alice\}_{K_{bob}}]$. As per the operational definition, we need to consider two instances of the binding sequence in different runs of the protocol: $\beta_{alice}(0) = [\{N^0_{alice}, N^0_{bob}, Alice\}_{K_{bob}}]$ and $\beta_{alice}(1) = [\{N^1_{alice}, N^1_{bob}, Alice\}_{K_{bob}}]$.

Claim 2: Alice achieves a valid post-session FLAG for Bob:
OPER($X_{alice} \triangleright X_{bob}, [\{N_{alice}, N_{bob}, Alice\}_{K_{bob}}]$).

***proof*:** The two operativeness challenges on X_{alice} are $C_0 = \beta_{alice}(0)$ and $C_1 = \beta_{alice}(1)$. On X_{alice}, we use $\lambda = [K_{alice}, N^0_{alice}, N^1_{alice}]$ as the auxiliary input for the operativeness distinguisher $\mathcal{D}^{oper}$. We select a random bit b and invoke the distinguisher: $b' \leftarrow \mathcal{D}^{oper}(C_b, \lambda)$, where b' is the distinguisher's output. The distinguisher construction is as follows.

$\mathcal{D}^{oper}(C_{b'}, \lambda)$:

(1) Decrypt $C_{b'}$ using $\lambda[0]$ to compute $\{x, ..\}$.

(2) If $\lambda[2] = x$ then return $b' = 1$ else return $b' = 0$.

As per the operational requirement, if $b = b'$ then our distinguisher has done a good job. For a key of size $s = |K_{alice}|$ and a nonce of size $t = |N_{alice}|$, and assuming uniform distribution for the key and the nonce, an upper bound on the probability of failure for the distinguisher ($b \neq b'$) is $p.2^{-s} + p.2^{-t}$, where p is the number of protocol instances using the same key. Clearly, for sufficiently large s and t, the upper bound is negligible. Also, trivially, finding $\Psi_{bob} = K_{bob}$ and $\Psi_{alice} = K_{alice}$, such that $K_{alice} \neq K_{bob}$ and $\{N^0_{alice}, N^0_{bob}, Alice\}_{K_{bob}} = \{N^0_{alice}, N^0_{bob}, Alice\}_{K_{alice}}$ is infeasible. Hence, Alice can achieve the operativeness of Bob by running the protocol. □

A similar, analysis can be done for Bob, which we leave out due to space constraints. Also the identification and willingness goals are trivially achieved because there are only two legitimate parties, e.g., for the identification case, the distinguisher can simply return the name of a far-end party after the successful decryption of the received messages.

About Security Analysis

As mentioned earlier, one advantage of the binding sequence based model is that the correctness analysis (for FLAGs) and the security analysis (for the binding sequence) are independent. In fact, the security analysis is no more than verifying the validity of the binding sequence of an authentication protocol. Since security analysis is not the main focus of this paper, we briefly discuss how the validation of a binding sequence can be done in complexity theoretic models [5] and in formal security models [24, 25].

For the former case, let us consider the binding sequence of Bob corresponding to the last message he received: $\beta_{X_{bob}} = [\{N_{bob}, N_{alice}\}]$. There are three different ways in which this sequence can be modified: $[\{N_{bob}, N'_{alice}\}]$, $[\{N'_{bob}, N_{alice}\}]$ and $[\{N'_{bob}, N'_{alice}\}]$, where a primed term represents a modified message. Now, for each of these modified sequence, we calculate an upper bound of accepting the modified sequence by Bob. For a valid *binding sequence* these upper bounds should represent a negligible probability. Generalizing this method results in a security analysis that involves verifying $2^{|\beta_{X_c}|} - 1$ cases of modified sequences. Interested readers are referred to the appendixes of our technical report [17].

In an automated tool based on symbolic models, such as OFMC [24] or LYSA [25], one can easily verify the validity of a binding sequence by verifying the authenticity of each message in the binding sequence. Of course, this is an over-approximation of the actual requirements of the binding sequence, because a binding sequence can be replayed. We are currently investigating how to accurately specify the actual requirement that allows such a replay but forbids the replay of the individual messages in a binding sequence. For now, we can rely on an ad-hoc solution: ignore all those attack traces in which the whole binding sequence is being replayed.

4 Plausibility Result: Computable Class of Problems

In the classic problem of multi party computation (MPC) [19], a set of parties want to compute an arbitrary function, such that the computation preserves certain security properties, e.g., the correctness of the result and the privacy of the inputs. The set of MPC parties consists of both honest and dishonest parties. Most of the work on secure MPC, however, assumes the availability of authenticated communication channels between honest parties.

In reality, authenticated channels may not always available, and therefore it is interesting to consider the MPC security problem without this assumption. Clearly, if the channels are not authentic then an adversary can even disconnect the MPC parties and run the protocol with any one of them without the possibility of being detected. Therefore, one needs some weak assumption to achieve a useful security guarantee. For example, Barak et al. [10] introduce the assumption of *independent execution*: roughly speaking, if an adversary plays man-in-middle then he must engage in independent executions of a protocol with each of the protocol parties.

The post-session authentication is another such assumption, but this is strong enough that it suffices to realize any MPC functionality correctly, assuming that there exists an MPC protocol that computes this functionality on authenticated channels. Note that the privacy of the inputs may not be protected, but the correctness of the output is guaranteed. The reader may wonder that if the parties have the capability to authenticate each other after a session then why not they do so at the start and establish an authentic channel instead, however, this is not always possible; some of the possible factors are listed below.

– PKI may be off-line or only accessible for a short duration at regular intervals. In this situation, immediate authentication is not always possible.
– The honest parties of MPC may not necessarily trust each other. Therefore, their decision to reveal their identities should depend on the observed behaviour in the session.
– The authentication may require a long time, e.g., in using physical authentication or postal mail to deliver PIN codes. Therefore, instead of waiting, parties may decide to start a session based on a general trust level of their community.

Usually, the proof for a theoretical plausibility of MPC is based on the simulation paradigm [19], in which one shows the equivalence between an actual model and an appropriately constructed ideal model. For our post-session authentication problem, this means constructing an ideal model that is similar to the standard authenticated channel model ($\mathcal{F}_{auth}$ [10]) except it reveals all the inputs to an adversary; then, we need to show that the adversary gain is negligible in the post-session authentication case.

Instead of the simulation based approach, we employ an indirect and simpler method. We construct, which we call, the Tabular scheme that interacts with an arbitrary MPC protocol in a black-box manner to achieve the correct result, while running on unauthenticated channels. In this way, this scheme serves as a constructive proof of the correct computation of any MPC protocol.

Tabular Scheme: Consider n parties that take part in an MPC protocol, using unauthenticated channels. Each party P_i, where $1 \leq i \leq n$, maintains two tables: T_t and T_r, each having n rows[6] . In T_t, the j_{th} row, where $1 \leq j \leq n$, represents the list of messages sent to P_j. Similarly, in T_r, the j_{th} row represents the list of messages received supposedly (as the connections are not authentic) from P_j. When the MPC protocol terminates, we execute a post-session authentication protocol between each P_i and P_j pair, such that P_i authenticates P_j using the j_{th} row of T_t as its session, while P_j participate in the authentication using her i_{th} row of T_r as a session.

Claim 3: Consider a protocol Π_{MPC} between n parties communicating over authenticated channels to compute a probabilistic functionality $\mathcal{F}_{\text{MPC}}$ within m interactions. If the inputs of n parties are not private then parties can also compute $\mathcal{F}_{\text{MPC}}$ while communicating over unauthenticated channels and using an n-party post-session authentication protocol.

***proof*:** We augment each of the n parties of Π_{MPC} with our Tabular Scheme as specified above and use SATH (see § 3) as the definition of authentication in the scheme. For each party, the memory requirement of the tables is $|T_t| + |T_r| = 2 \times m \times n \times |M|$, where $|M|$ is the maximum size of any individual message in the protocol.

The authentication protocol in the Tabular Scheme succeeds between P_i and P_j only if j_{th} row of T_t (on P_i) and i_{th} row of P_r (on P_j) are exactly same and the two parties possess legitimate credentials. These two rows can be

[6] Actually one needs $n-1$ rows, but we use n to simplify the indexing.

considered as the session footprint for the communication from P_i to P_j. On the other hand, if both of these rows are same then this guarantees that the adversary has not modified any message in these rows.
Next, we rerun the authentication protocol of the Tabular scheme to achieve SATH between every pair of the protocol parties, which requires running $n(n-1)$ instances[7] of two-party SATH protocol. If all these instances succeed then this guarantees that all parties agree on the messages that were exchanged in the session and the adversary has not fabricated, modified or deleted any message in the session. Hence, the output of the protocol Π_{MPC} is necessarily correct, i.e., $\mathcal{F}_{\text{MPC}}$. □

Clearly, the Tabular scheme interacts with Π_{MPC} in a black-box manner, which implies that we can deploy an arbitrary MPC protocol given that the protocol does not require input privacy. Also note that the overhead, in terms of memory ($2 \times n \times m \times |M|$ bits) and time ($n^2 - n$ instances of authentication), is polynomial in the size of a protocol.

We can optimize the Tabular scheme by using a hash function, i.e., instead of using a complete row we may use the hash value of the row as the representation of a session, which, in many cases, can be encoded as a single message in an authentication protocol. Depending on the requirements of an MPC protocol, the definition of authentication can be relaxed from SATH, e.g., if timeliness is not important then the operativeness goal (OPER) is not required.

5 Discussion and Future Directions

One may argue that the additional requirements in pre-session or post-session authentication are not the "real" authentication requirements. A good illustrative example is of a two-party secure communication protocol, in which a secret session key is computed to establish a secure channel between the parties. Here, the confidentiality of the key and authentication of the parties appear to be completely independent protocol goals.

This view, however, manifests its limitation as soon as we consider the goal of establishing two parallel secure channels between same two parties. Now, there are two authentication results and two secret session keys, and the associations between the keys (or the subsequent sessions) and the authentication results are indeed essential requirements. Such a situation is even more dangerous for post-session authentication, e.g., it will allow a session hijacking attack, in which an honest party does all the hard work in a session and then a dishonest party simply claims the ownership of the session at the end.[8]

The reader may have realized that not all the problems that are solvable using pre-session authentication can be solved using post-session authentication, partially because post-session authentication can not guarantee the confidentiality of the inputs. Another factor is that if the session involves some access to a

[7] The number of permutation pairs on a set of order n

[8] The same attack is also described in the auction protocol [3].

protected resource, which only an authorized entity is allowed to do, then post-session authentication can not help, because an adversary can easily pretend to be an authorized party. Nevertheless, in many applications the effect of a session on a system can be reversed, e.g., cancelling the purchasing order (if the customer's credit card payment is later denied by the issuing bank) and redoing an auction.

The separation of correctness and security requirements as detailed in our earlier work [9] is not affected with the post-session extension. In particular, the validity of a binding sequence is the only required security property; all authentication properties of practical significance (FLAGs) can be derived from the binding sequence. We believe that the job of a security analyst (human/automated tool) would be less strenuous if security requirements are fewer and pure, considering the security analysis is an undecidable problem in general.

For the future work, an immediate challenge is to find a general method that can be used to specify the session computing function from a given set of application requirements. In this regard, the notions of credit and responsibility are critical and somehow needs to be specified formally. The notion of a session Ψ_i can be interpreted in a probabilistic sense to obtain precise security bounds especially when Ψ_i is a digest of a complete transcript. This will imply that the unique identification of sessions using their Ψ_i occurs with a certain (high) probability. More research on these issues will help to integrate post-session authentication into existing tools that automatically analyse authentication protocols or provide a provable security assurance.

6 Related Work on Authentication Models

We only cover some highlights in the area that concerns with the modelling aspect of authentication. Although the current models do not consider the post-session scenario, we believe they can be extended for this purpose.

Probably, the first attempt to model authentication is in BAN logic [4], which formalizes the authentication goals in terms of beliefs held by peer entities, however, this line of work has some limitations [13]. In cryptographic models, authentication in terms of matching conversation [5] is among the first, but still popular, approach. This requirement is too strong [12], but it can be extended to include a session to capture the post-session requirements.

Gollmann [7] presents an in-depth analysis of authentication. Roscoe [15] distinguishes between *intensional* and *extensional* style of authentication goals. Boyd and Mathuria [12] consider intensional specifications to be restrictive, and Gollman [11] even discourages such formal specifications. The underlying cause of this puzzle is that it is often not clear how an intensional property is related to an extensional property. In our model, this problem is resolved as FLAGs (extensional goals) are derived from the binding sequence (an intensional property).

Some other proposals for authentication goals [6, 18, 2] are not satisfactory [12]. Lowe [8] identifies four requirements of authentication with varying strength

and formalize them using process algebra. Boyd and Mathuria [12] provide only two goals related to entity authentication. Cremer [14] introduces an hierarchy of authentication levels. In many formal methods of security analysis [25, 24], the focus is on message authentication. Nevertheless, these tools enable an automatic validation of binding sequences, as indicated in § 3.

Gorrieri et al. [28] formalize the informal notions of credit and responsibility [27], which can be extended to formalize the session computation function. Squicciarini et al. [26] propose an authentication framework that supports an authentication decision based on the previous events that occurred in the system. Such a framework can be used to support post-authentication, e.g., by defining an authentication policy that cryptographically connects a session to the success of a subsequent authentication event.

7 Conclusion

In this paper, we specify the requirements of post-session authentication and show that it can be used to solve any MPC problem that is solvable on authenticated channels and does not require the input privacy. Authenticating after a session, if possible, indeed offers some advantages, such as anonymity and less dependency on the availability of PKI. When the choice is available between post-session and pre-session authentication, relative pros and cons are normally application dependent. Although the use of post-session authentication is currently less common, we hope our work will be useful in recognizing its advantages, as well as its limitations, and building more innovative secure systems.

References

1. Zimmermann, P.R.: Pgpfone: Pretty good privacy phone owner's manual, version 1.0(5), http://web.mit.edu/network/pgpfone/manual/#PGP000057, 1996
2. ISO standard: Entity Authentication Mechanisms; Part 1: General Model. ISO/IEC 9798-1, Second Edition, September 1991
3. Stajano, F., Anderson, R.: The Cocaine Auction Protocol: On the Power of Anonymous Broadcast. Information Hiding, pp.434–447, pub. Springer, 2000
4. Burrows, M. and Abadi, M. and Needham, R.M.: A logic of Authentication. DEC System Research Center, Report 39, revised Feb 22, 1990
5. Bellare, M. and Rogaway, P.: Entity authentication and key distribution. Crypto'93, pp.232–249, Springer-Verlag LNCS, Vol 773, 1993
6. Syverson, P.F. and Van Oorschot, P.C.: On Unifying Some Cryptographic Protocol Logics. In Proc.: S&P, pub. IEEE, ISSN:1063-7109, 1994
7. Gollmann D.: What do we mean by entity authentication?. In Proc.: Symposium on Security and Privacy, pub. IEEE, pp.46–54, 1996
8. Lowe, G.: A Hierarchy of Authentication Specifications. In Proc.: 10th Computer Security Foundations Workshop (CSFW '97), 1997
9. Ahmed, N. and Jensen, C.D.: Demarcation of Security in Authentication Protocols. In Proc.: 1st SysSec Workshop, pub. IEEE Computer Society, pp. 43–50, 2011
10. Barak, B., Canetti, R., Lindell, Y., Pass, R., and Rabin, T.: Secure computation without authentication. In: CRYPTO, pp.361–377, pub. Springer, 2005

11. Gollmann, D.: Authentication—myths and misconception. Cryptography and Computational Number Theory, pub. Birkhauser, pp.203–225, 2001
12. Boyd, C., Mathuria, A.: Protocols for Authentication and Key Establishment. pub. Springer, ISBN: 978-3-540-43107-7, 2003
13. Kurkowski, M., Srebrny, M.: A Quantifier-free First-order Knowledge Logic of Authentication. Fundamenta Informaticae 72(1-3), IOS, ISSN 1875-8681, 2006
14. Cremers, C.J.F.: Scyther: Semantics and Verification of Security Protocols. IPA Dissertation Series 2006-20, Eindhoven, 2006
15. Roscoe, A.W.: Intensional specifications of security protocols. In Proc.: Computer Security Foundations Workshop, pub. IEEE, pp.28–38, 1996
16. Ahmed, N. and Jensen, C.D.: Definition of Entity Authentication. In Proc.: 2nd IWSCN, pub. IEEE, pp.1–7, 2010
17. Ahmed, N. and Jensen, C.D.: Adaptable authentication model. In Proc.: ESSoS, pub. Springer, pp.234–247, (Technical Report: IMM-TR-2010-17), 2011
18. Menezes, A.J. and Van Oorschot, P.C. and Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, 1997
19. Goldreich, O.: Foundations of cryptography: Basic applications. Cambridge University Press, 2004
20. Juels, A.: RFID security and privacy: A research survey. In J.: Selected Areas in Communications 24(2), pub. IEEE, pp.381–394, 2006
21. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory, 22(6), pp.644–654, 1976
22. Amazon UK web store, `http://www.amazon.co.uk`
23. Lucks, S., Zenner, E., Weimerskirch, A., Westhoff, D.: Concrete security for entity recognition: The Jane Doe protocol. IndoCrypt, pp.158–171, pub. Springer, 2008
24. Basin, D., Mödersheim, S., and Vigano, L.: OFMC: A symbolic model checker for security protocols. In International J. of Information Security, pp.181–208, 2005
25. Bodei, C., Buchholtz, M., Degano, P., Nielson, F., and Nielson, H.R.: Static validation of security protocols. Journal of Computer Security, pp.347–390, 2005
26. Squicciarini, A.C., Bhargav-Spantzel, A., Bertino, E., Czeksis, A.B.: Auth-SL: a system for the specification and enforcement of quality-based authentication policies. In Proc.: ICICS'07, pp.386–397, 2007
27. Abadi, M.: Two facets of authentication. In Proc.: Computer Security Foundations Workshop, pub. IEEE, pp.27–32, 1998
28. Gorrieri, R., Martinelli, F. and Petrocchi, M.: A formalization of credit and responsibility within the gndc schema. ENTCS, vol. 157(3), pub. Elsevier, pp.61–78, 2006