Security Analysis of Olvid's SAS-based Trust Establishment Protocol

Michel Abdalla

Security Analysis of Olvid's SAS-based Trust Establishment Protocol

Michel Abdalla[1,2]

[1] DIENS, École normale supérieure, CNRS, PSL University, Paris, France
michel.abdalla@ens.fr
[2] INRIA, Paris, France

Abstract. In this report, we analyze the security of the trust establishment protocol used in the Olvid messaging protocol. The latter relies on the PV-SAS-MCA message cross-authentication protocol by Pasini an Vaudenay based on short authenticated strings (SAS). In order to make the implementation portable across different platforms, Olvid proposed particular instantiations of the underlying primitives used in PV-SAS-MCA in addition to some other minor modifications. Here, we show that these changes have no impact on the security of the scheme. More precisely, we formally prove that the trust establishment protocol used in Olvid is a secure message cross-authentication protocol. The proof of security is in the random-oracle model and relies on the security of the underlying pseudorandom generator. It also assumes users know each other and have an authentic channel between them.

1 Introduction

Message cross-authentication (MCA) protocols allow users to identify each other and to simultaneously exchange messages in an authenticated manner. In [Vau05], Vaudenay proposed a variant of the above notion in which one assumes the existence of an extra channel which can authenticate short strings. The new notion allows for the construction of efficient authentication protocols based on short authenticated strings (SAS) and is known as SAS-based authentication.

Olvid[3] is a secure instant messaging application which, contrary to other applications like WhatsApp or Signal, does not impose its users to trust a central directory of users for the key distribution. Instead, Olvid offers multiple trust establishment protocols relying on the natural relationship between users so as to retain the same "security level" in the digital world as in real life. The absence of a central directory has two main advantages: (1) users do not need to disclose any personal information to the server, and (2) the key distribution is completely decentralized without any need for a trusted third party.

---

[3] https://olvid.io/

In this report, we analyze the security of the first trust establishment protocol used in Olvid, which assumes users know each other and have an authentic channel between them (e.g, phone calls, face-to-face meetings, voice chats). This protocol uses the SAS-based message cross-authentication (MCA) protocol PV-SAS-MCA by Pasini and Vaudenay [PV06, Pas09]. In Olvid, the SAS-based MCA protocol is used to allow users not only to identify each other but also to exchange their public keys in authenticated manner.

In the remainder of this report, we analyze the security of the particular instantiation of the PV-SAS-MCA protocol used in Olvid and show that it meets the security requirements listed in the security theorem for PV-SAS-MCA in [PV06, Pas09]. For completeness, we also provide a formal security proof that shows that the trust establishment protocol used by Olvid is a secure message cross-authentication protocol.

Organization. Section 2 recalls some basic notation used in this report and the cryptographic primitives used in the PV-SAS-MCA protocol. Section 3 recalls the description of the PV-SAS-MCA protocol by Pasini and Vaudenay [PV06, Pas09]. Section 4 describes the variant of the PV-SAS-MCA protocol used by Olvid together with a detailed security analysis. Finally, Section 5 concludes this report.

2   Preliminaries

Notation. We write $x \leftarrow y$ for the action of assigning the value y to the variable x. We write $x_1, \ldots, x_n \leftarrow_{\$} \mathsf{X}$ for sampling $x_1, \ldots, x_n$ from a finite set X uniformly at random. If A is a probabilistic algorithm we write $y_1, \ldots, y_n \leftarrow_{\$} A(x_1, \ldots, x_n)$ for the action of running A on inputs $x_1, \ldots, x_n$ with independently chosen coins, and assigning the result to $y_1, \ldots, y_n$. PPT as usual abbreviates probabilistic polynomial-time.

Commitment Schemes. A keyed tag-based commitment scheme is defined by a triple of PPT algorithms (setup, commit, open). Via $(K_p, K_s) \leftarrow_{\$} \mathsf{setup}(1^\lambda)$, the setup algorithm generates a public-secret key pair (K_p, K_s). For any message m, tag t, and public key K_p, the commitment algorithm outputs a pair (c, d) corresponding to the commit and decommit values via $(c, d) \leftarrow_{\$} \mathsf{commit}(K_p, t, m)$. Given (c, d) and a tag t, the original message can be recovered using the opening algorithm open via $m \leftarrow \mathsf{open}(K_p, t, c, d)$. As noted in [Pas09], K_p may be empty.

For the security of the SAS-based message cross-authentication protocol used in Olvid and described in Section 3, the commitment scheme needs to satisfy the following properties:

- **Completeness**: For any key pair $(K_p, K_s) \leftarrow_{\$} \mathsf{setup}(1^\lambda)$, any message m, any tag t, and any $(c, d) \leftarrow_{\$} \mathsf{commit}(K_p, t, m)$, we have $m = \mathsf{open}(K_p, t, c, d)$.
- **Perfectly Hiding**: For any key pair $(K_p, K_s) \leftarrow_{\$} \mathsf{setup}(1^\lambda)$, any tag t, any values m_0, m_1, any random bit b, and any $(c, d) \leftarrow_{\$} \mathsf{commit}(K_p, t, m_b)$, c gives no information on b.
- **Computationally Binding**: For any key pair $(K_p, K_s) \leftarrow_{\$} \mathsf{setup}(1^\lambda)$, it is hard to find (m, t, c, d_0, d_1) such that $m_b \leftarrow \mathsf{open}(K_p, t, c, d_b) \neq \perp$ for $b = 0, 1$ and $m_0 \neq m_1$.
- **Equivocable**: There exist algorithms simcommit and equivocate capable of generating fake commitments that can opened to arbitrary values. That is, given the secret key K_s and tag t, $(c, \xi) \leftarrow_{\$} \mathsf{simcommit}(K_s, t)$ yields a fake commit value c and an additional information value ξ. Then, given t, K_s, and the output (c, ξ) of $\mathsf{simcommit}(K_s, t)$, $\mathsf{equivocate}(K_s, c, \xi, m)$ can output a decommit value d such that $m = \mathsf{open}(K_p, c, d)$ for any message m.

Message Cross-Authentication. Message Cross-Authentication (MCA) protocols allow users to exchange messages in an authenticated manner and are often used to exchange public keys.

Definition 2.1 (Message Cross-Authentication [Pas09]). *A Message Cross-Authentication (*MCA*) protocol between* Alice *and* Bob *starts with inputs m_A and m_B and ends with outputs $(\hat{\mathsf{id}}_B, \hat{m}_B)$ and $(\hat{\mathsf{id}}_A, \hat{m}_A)$, respectively.*

– *An honest run should lead to* $(\hat{\mathsf{id}}_A, \hat{m}_A) = (\mathsf{id}_A, m_A)$ *and* $(\hat{\mathsf{id}}_B, \hat{m}_B) = (\mathsf{id}_B, m_B)$.
– *An adversary is successful if some instance of an uncorrupted node ended with a pair* (m, id), *but no instance of the node of identity* id *was launched with input* m.

3   SAS-based MCA protocol

The PV-SAS-MCA protocol by Pasini and Vaudenay [PV06, Pas09] is recalled in Fig. 1 and works as follows. First, users Alice and Bob initiate the protocol with their respective inputs m_A and m_B and then each picks a random key, denoted by K and R. Next, Alice computes a commitment c of her key K and sends both m_A and c to Bob, who replies with m_B and R. After receiving m_B and R, Alice reveals her hidden key K by sending the decommitment value d to Bob. Finally, both Alice and Bob authenticate the string $\mathsf{SAS} = R \oplus h(m_B, K)$ using the extra authenticated channel, where h is an almost strongly universal hash function.

<div align="center">

Alice   Bob
input : m_A input : m_B

$K \leftarrow_\$ \{0,1\}^\kappa$ $R \leftarrow_\$ \{0,1\}^\rho$
$(c,d) \leftarrow \mathsf{commit}(\mathsf{id}_A, m_A, K)$ $\xrightarrow{\;m_A, c\;}$
$\xleftarrow{\;m_B, R\;}$

$\xrightarrow{\quad d \quad}$ $\hat{K} \leftarrow \mathsf{open}(\hat{\mathsf{id}}_A, \hat{m}_A, \hat{c}, \hat{d})$

$\mathsf{SAS} \leftarrow \hat{R} \oplus h(\hat{m}_B, K)$ $\xrightarrow{\;\mathsf{auth}_{\mathsf{id}_A}(\mathsf{SAS})\;}$ $\mathsf{SAS} \overset{?}{=} R \oplus h(m_B, \hat{K})$

Check SAS is the same $\xleftarrow{\;\mathsf{auth}_{\mathsf{id}_B}(\mathsf{SAS})\;}$
Check $\mathsf{id}_A \neq \mathsf{id}_B$

output : $\mathsf{id}_B, \hat{m}_B$ output : $\mathsf{id}_A, \hat{m}_A$
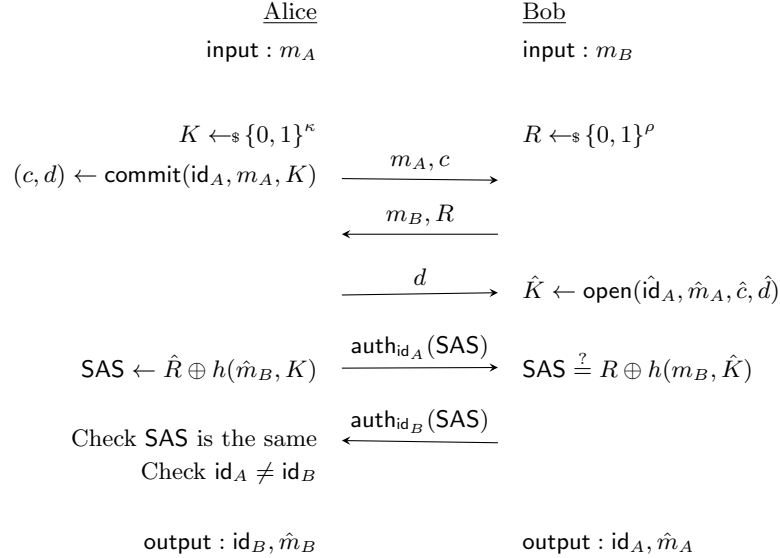
</div>

Fig. 1. The SAS-based MCA protocol PV-SAS-MCA [PV06, Pas09].

As stated in [Pas09, Section 8.4], the SAS-based MCA protocol PV-SAS-MCA directly authenticates m_B through the SAS and indirectly authenticates m_A via the direct authentication of K and the tagged commitment scheme, which binds m_A to K. Moreover, as discussed in [Pas09], h can be instantiated as $h(x, K) = \mathsf{trunc}(\mathsf{hash}(K, x))$ with a collision-resistant hash function hash such as SHA-256 [SHA04] and trunc is a function that truncates its output to the leading bits.

4   Olvid's SAS-based MCA protocol

The SAS-based MCA protocol used in Olvid is described in Fig. 2 and differs slightly from the PV-SAS-MCA protocol in Fig. 1. In particular, the values m_A and m_B being exchanged are public keys associated with Alice and Bob, respectively. In addition to this, there are a few other notable changes to the original protocol:

– Bob now sends m_B to Alice at the beginning of the protocol;

- Every message from Alice to Bob is encrypted under Bob's public key m_B;
- With the exception of the first message, every message from Bob to Alice is encrypted under Alice's public key m_A;
- The function h is instantiated with a hash function H, modeled as a random oracle in the security analysis;
- The random value R is computed as $\mathsf{H}(k_B, c)$ where H is the SHA-256 hash function [SHA04] and k_B is a secret key held by Bob; and
- The SAS value is computed as $\mathsf{SAS} = \mathsf{prng}(R \oplus \mathsf{H}(\hat{m}_B, K))$.

<table>
<tr><td align="center"><u>Alice</u>
input : m_A</td><td></td><td align="center"><u>Bob</u>
input : m_B</td></tr>
<tr><td align="right">$K \leftarrow_\$ \{0,1\}^\kappa$</td><td align="center">$\xleftarrow{\quad \mathsf{id}_B, m_B \quad}$</td><td></td></tr>
<tr><td align="right">$(c,d) \leftarrow \mathsf{commit}(m_A, K)$</td><td align="center">$\xrightarrow{\quad \{\mathsf{id}_A, m_A, c\}_{m_B} \quad}$</td><td></td></tr>
<tr><td></td><td align="center">$\xleftarrow{\quad \{R\}_{m_A} \quad}$</td><td align="left">$R \leftarrow \mathsf{H}(k_{B,m_B}, \hat{c})$</td></tr>
<tr><td></td><td align="center">$\xrightarrow{\quad \{d\}_{m_B} \quad}$</td><td align="left">$\hat{K} \leftarrow \mathsf{open}(\hat{m}_A, \hat{c}, \hat{d})$</td></tr>
<tr><td align="right">$\mathsf{SAS} \leftarrow \mathsf{prng}(\hat{R} \oplus \mathsf{H}(\hat{m}_B, K))$</td><td align="center">$\xrightarrow{\quad \mathsf{auth}_{\mathsf{id}_A}(\mathsf{SAS}) \quad}$</td><td align="left">$\mathsf{SAS} \overset{?}{=} \mathsf{prng}(R \oplus \mathsf{H}(m_B, \hat{K}))$</td></tr>
<tr><td align="right">Check SAS is the same
Check $\mathsf{id}_A \neq \mathsf{id}_B$</td><td align="center">$\xleftarrow{\quad \mathsf{auth}_{\mathsf{id}_B}(\mathsf{SAS}) \quad}$</td><td></td></tr>
<tr><td align="center">output : $\mathsf{id}_B, \hat{m}_B$</td><td></td><td align="center">output : $\mathsf{id}_A, \hat{m}_A$</td></tr>
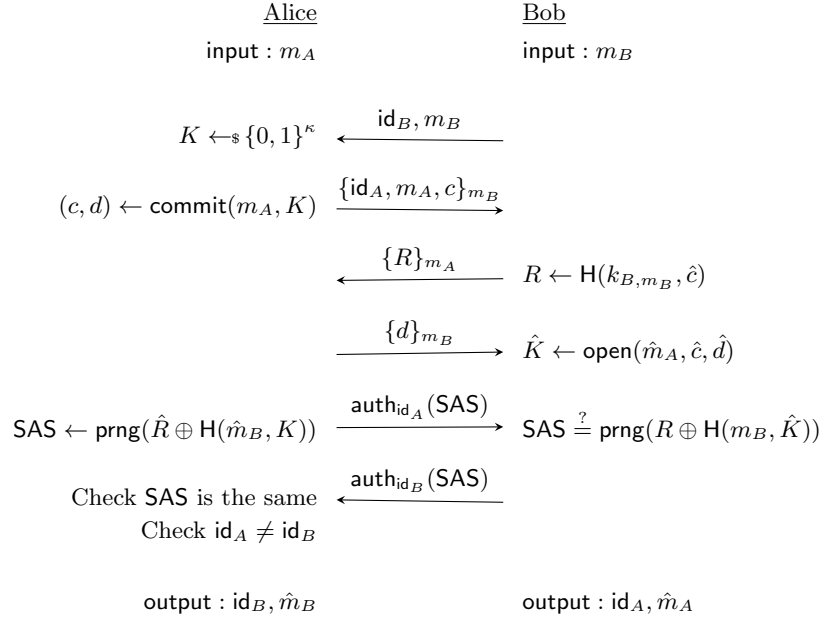</table>

Fig. 2. Olvid's SAS-based MCA protocol. The notation $\{a\}_m$ indicates the public-key encryption of the plaintext a using m as the public key.

4.1   Instantiations

Equivocable Commitments. In order to instantiate the commitment scheme in PV-SAS-MCA protocol in Fig. 1, Olvid uses the following random-oracle-based commitment scheme given in [Pas09, Section 3.5.9.2].

Let ℓ_c, ℓ_e, and ℓ_h be three integers and let H be a hash function from $\{0,1\}^{\ell_h + \ell_e}$ to $\{0,1\}^{\ell_c}$ modeled as a random oracle. The equivocable commitment works as follows:

- setup: There is no setup algorithm. Hence, $K_p = K_s = \varepsilon$.
- commit(t, m): Sample $e \leftarrow_\$ \{0,1\}^{\ell_e}$, set $c = \mathsf{H}(t, m, e)$ and $d = (m, e)$, and output (c, d).
- open(t, c, d): Parse d as (m, e), check that $c = \mathsf{H}(t, m, e)$. If true, return m, else return $\bot$.

As noted in [Pas09, Section 3.5.9.2], one can easily build the simcommit and equivocate algorithms using the fact that H is modeled as a random oracle.

As shown in [Pas09], the commitment scheme above is perfectly hiding and statistically binding, In Olvid, the hash function H is instantiated with SHA-256 [SHA04].

SAS Computation. The implementation of the SAS differs from the original protocol in [Pas09] as follows:

– The SAS computation outputs a random 8-digit numbers, where the first and last 4 digits are displayed to Alice and Bob, respectively. To complete the authentication, both parties must type the 4 digits associated with the other party.

Pseudorandom Number Generator. Olvid instantiates the PRNG using the NIST SP 800-90A [BK15].

4.2   Changes to the Security Analysis of PV-SAS-MCA

Despite the numerous modifications, the proof of security for the PV-SAS-MCA protocol in [Pas09] also applies to the Olvid SAS-based MCA protocol described in Fig. 2. In the following, we describe the main changes to the proof.

Encryption of protocol messages. In Fig. 2, with the exception of the very first message from Bob to Alice, all messages are encrypted using the ECIES encryption scheme [ABR01] under the public-key of the recipient. Since the security of the PV-SAS-MCA protocol in [Pas09] does not require privacy for the messages being exchanged and given that the public-key encryption scheme is perfectly binding, this change has no impact on the security of Olvid's protocol. In particular, the security analysis of Olvid's protocol makes no assumption about the security of the underlying public-key encryption scheme and simply assumes that these messages are sent in the clear.

Computation of R. Unlike in the PV-SAS-MCA protocol where R is chosen uniformly at random in each session, the random value R in Olvid's version of the protocol is computed as $\mathsf{H}(k_{B,m_B}, c)$ where H is the SHA256 hash function [SHA04], k_{B,m_B} is a secret key held by Bob which is chosen uniformly at random for each different message m_B, and c is Alice's commitment. Since the commitment scheme being used is statistically binding and H is a random oracle, this has no impact on the security of the protocol given that each execution of protocol between Alice and Bob yields a random and uniformly distributed R value.

PRNG used in the SAS computation. In the PV-SAS-MCA protocol, the short authenticated string used for the SAS authentication is equal to $R \oplus \mathsf{H}(m_B, K)$. In Olvid's protocol, the value $R \oplus \mathsf{H}(m_B, K)$ is used to derive the short authenticated string via a PRNG function prng. Since the security analysis of the PV-SAS-MCA protocol shows that the string $R \oplus \mathsf{H}(m_B, K)$ is random and uniformly distributed, the same is true for $\mathsf{prng}(R \oplus \mathsf{H}(m_B, K))$ due to the security of the PRNG function prng. As a result, the probability that an adversary can impersonate either Alice or Bob on a given interaction is $1/10^4$.

Bob's first message. Unlike the PV-SAS-MCA protocol, Bob now sends message m_B to Alice at the beginning of the protocol. Since m_B is directly authenticated through the SAS authentication and given that the value R computed by Bob remains random and uniformly distributed even when Alice chooses its first flow message as a function of m_B, this change has no impact on the security of the protocol.

Universal hash function. In the PV-SAS-MCA protocol, the function h is assumed to be an almost strongly universal hash function. In Olvid's protocol, the function h is is simply instantiated with the SHA-256 hash function, which we assume behaves as a random oracle.

4.3   Security Proof

As indicated in Section 4.2, the modifications made to the PV-SAS-MCA protocol do not affect the security of the scheme. To prove this formally, we prove the security of the Olvid SAS-based MCA protocol described in Fig. 2 against one-shot adversaries, in which adversaries can launch at most one instance of each party. As shown by Vaudenay [Vau05], this result already implies security with respect to adversaries that can launch polynomially many instances of each party.

Security model. As indicated in [Vau05], adversaries are assumed to have full control of the communication network, except for the authenticated channel. in particular, adversaries are allowed to launch new instances of a protocol party (via queries to a launch oracle) or to send messages to protocol instances (via queries to a send oracle). The adversary is also allowed to specify the input of a party when querying the launch oracle.

- launch(n, r, x) denotes a query to the launch oracle where n is a node of the network, r is the role of this node, and x is the input. It outputs a unique instance tag π_n^i.
- send(π_n, m) denotes a query to the send oracle through which the adversary can send a message m to an instance π_n. It outputs a message m' that is meant to be sent to the other protocol party.

In a SAS-based MCA protocol, there are two parties, called Alice and Bob. On Alice's side, the protocol input is (m_A, id_A) and its output is $(\hat{m}_B, \hat{\mathsf{id}}_B)$. On Bob's side, the protocol input is (m_B, id_B) and its output is $(\hat{m}_A, \hat{\mathsf{id}}_A)$. Finally, the adversary is considered successful in attacking the authentication protocol if there exists at least one instance of Alice (resp. Bob) that terminates and outputs $(\hat{m}_B, \hat{\mathsf{id}}_B)$ (resp. $(\hat{m}_A, \hat{\mathsf{id}}_A)$) such that no instance of Bob (resp. Alice) exists with input $(m_B, \mathsf{id}_B) = (\hat{m}_B, \hat{\mathsf{id}}_B)$ (resp. $(m_A, \mathsf{id}_A) = (\hat{m}_A, \hat{\mathsf{id}}_A)$).

Security theorem. As the following theorem shows, Olvid's SAS-based MCA protocol is secure against one-shot adversaries, if H is modeled as a random oracle, prng is a secure pseudorandom generator, and the underlying commitment scheme is the one described in Section 4.1. The proof is based on the proof of security for the original SAS-based MCA protocol PV-SAS-MCA [PV06, Pas09].

Theorem 4.1. *Let* H *be a random oracle, let* commit *be the random-oracle-based commitment scheme in Section 4.1, and let* prng *be a secure pseudorandom generator. Then any one-shot adversary against the Olvid's SAS-based MCA protocol is successful with probability negligibly close to* $1/10^4$.

Proof. We prove the security of Olvid's SAS-based MCA protocol using a sequence of games, starting with the original protocol and ending with a protocol in which the success probability of the adversary is 0.

Game G_1. This is the initial security game, so we have:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{OLVID\text{-}SAS}}(\ ) = \Pr[\,\mathsf{G}_1 \Rightarrow \mathsf{T}\,]$$

Game G_2. In this game, we change the simulation of the protocol so that Bob chooses the value R uniformly at random in the range of H.

Since H is a random oracle. the advantage of the adversary remains the same unless it finds a collision on the output of the random oracle, which is negligible for any polynomial number of queries to the latter. As a result, $|\Pr[\,\mathsf{G}_2 \Rightarrow \mathsf{T}\,] - \Pr[\,\mathsf{G}_1 \Rightarrow \mathsf{T}\,]\,|$ is negligible.

Game G_3. In this game, we extract the value K committed in $\hat{c}$ and abort if extraction fails.

Since the random-oracle-based commitment scheme described in Section 4.1 is statistically binding, the probability of aborting is negligible, This is due to the fact that an adversary can only break the binding property if it finds a collision in the random oracle used by the commitment scheme, which is negligible for any polynomial number of queries to the latter. Hence, $|\Pr[\,\mathsf{G}_3 \Rightarrow \mathsf{T}\,] - \Pr[\,\mathsf{G}_2 \Rightarrow \mathsf{T}\,]\,|$ is negligible.

Game G_4. If the two instances created by the one-shot adversary are Alice's instances, then the seed used by the pseudorandom generator prng of Alice's instance that terminates last is chosen at random.

Let $K^{(i)}, \hat{m}_B^{(i)}, \hat{R}^{(i)}$ be the values seen or chosen by Alice's instance $i \in \{1, 2\}$, where the index 2 indicates the instance that terminates last. Since the commitment scheme is perfectly hiding, the value $K^{(2)}$ is uniformly distributed and independent from the rest. As a result, the adversary can

only detect this change if $\hat{R}^{(1)} \oplus \mathsf{H}(\hat{m}_B^{(1)}, K^{(1)}) = \hat{R}^{(2)} \oplus \mathsf{H}(\hat{m}_B^{(2)}, K^{(2)})$, which happens with negligible probability. Hence, it follows that $|\Pr[\, \mathsf{G}_4 \Rightarrow \mathsf{T}\,] - \Pr[\, \mathsf{G}_3 \Rightarrow \mathsf{T}\,]|$ is negligible.

Game G_5. If the two instances created by the one-shot adversary are Bob's instances, then the seed used by the pseudorandom generator prng of the instance of Bob that sends its R value last is chosen at random.

Let $\hat{K}^{(i)}, m_B^{(i)}, R^{(i)}$ be the values seen or chosen by instance $i \in \{1, 2\}$ of Bob, where the index 2 indicates the instance that sends its R value last. Since the value $R^{(2)}$ is uniformly distributed and independent from the rest, the adversary can only detect this change if $R^{(1)} \oplus \mathsf{H}(m_B^{(1)}, \hat{K}^{(1)}) = R^{(2)} \oplus \mathsf{H}(m_B^{(2)}, \hat{K}^{(2)})$, which happens with negligible probability. Thus, $|\Pr[\, \mathsf{G}_5 \Rightarrow \mathsf{T}\,] - \Pr[\, \mathsf{G}_4 \Rightarrow \mathsf{T}\,]|$ is negligible.

In Games G_6–G_8, we consider the cases in which the one-shot adversary creates one instance of Alice and one of Bob. Without loss of generality, we will assume that Bob's instance terminates last since the other cases can be dealt with in a similar way.

Game G_6. If the one-shot adversary creates one instance of Alice and one of Bob *and* Alice's instance's terminates before Bob's instance sends its R value, then the seed used by the pseudorandom generator prng of Bob's instance is chosen at random.

Since the value R is uniformly distributed and independent from the rest, the adversary can only detect this change if $\hat{R} \oplus \mathsf{H}(\hat{m}_B, K) = R \oplus \mathsf{H}(m_B, \hat{K})$, which happens with negligible probability. Thus, $|\Pr[\, \mathsf{G}_6 \Rightarrow \mathsf{T}\,] - \Pr[\, \mathsf{G}_5 \Rightarrow \mathsf{T}\,]|$ is negligible.

Game G_7. If the one-shot adversary creates one instance of Alice and one of Bob *and* Alice's instance does not terminate before Bob's instance sends its R value *and* $\hat{c} \neq c$, then the seed used by the pseudorandom generator prng of Alice's instance is chosen at random.

Since the value K remains hidden until Alice's instance sends the decommitment value d, its value is uniformly distributed and independent from the other values used in the seed computation of both instances. As a result, the probability that the adversary can only detect the change in the game is negligible. Hence, $|\Pr[\, \mathsf{G}_7 \Rightarrow \mathsf{T}\,] - \Pr[\, \mathsf{G}_6 \Rightarrow \mathsf{T}\,]|$ is negligible.

Game G_8. If the one-shot adversary creates one instance of Alice and one of Bob *and* Alice's instance does not terminate before Bob's instance sends its R value *and* $\hat{c} = c$, then the seed used by the pseudorandom generator prng of Alice's instance is chosen at random.

Since the commitment scheme is statistically binding, it follows that $\hat{m}_A = m_A$. Hence, we can assume that $\hat{m}_B \neq m_B$ as otherwise the adversary would not be considered successful. Moreover, as in Game G_7, the value K remains hidden until Alice's instance sends the decommitment value d and its value is uniformly distributed and independent from the other values used in the seed computation of both instances. As a result, the probability that the adversary can only detect the change in the game is negligible. Hence, $|\Pr[\, \mathsf{G}_8 \Rightarrow \mathsf{T}\,] - \Pr[\, \mathsf{G}_7 \Rightarrow \mathsf{T}\,]|$ is negligible.

Game G_9. In this game, the output of the pseudorandom generator prng is replaced with a random string whenever its seed is chosen uniformly at random.

Since it is straightforward to convert any adversary that detects this change into one that breaks the security of the pseudorandom generator prng, it follows that $|\Pr[\, \mathsf{G}_9 \Rightarrow \mathsf{T}\,] - \Pr[\, \mathsf{G}_8 \Rightarrow \mathsf{T}\,]|$ is negligible.

Finally, the claim in Theorem 4.1 follows by noticing that, for every instance in Game G_9, at least one of the values used for the SAS comparison is chosen uniformly at random. □

5 Conclusion

As the analysis in Section 4 formally shows, the trust establishment protocol used by Olvid is a secure message cross-authentication protocol. The analysis assumes users know each other and have an authentic channel between them (e.g, phone calls, face-to-face meetings, voice chats). The security proof is in the random-oracle model and relies on the security of the underlying pseudorandom generator, which has been instantiated according to NIST SP 800-90A [BK15].

References

ABR01. M. Abdalla, M. Bellare, and P. Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In *CT-RSA 2001*, *LNCS* 2020, pages 143–158. Springer, Heidelberg, April 2001.

BK15.   E. Barker and J. Kelsey. Recommendation for random number generation using deterministic random bit generators. National Institute of Standards and Technology, NIST Special Publication 800-90A Revision 1, U.S. Department of Commerce, June 2015. http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf.

Pas09. S. Pasini. *Secure Communication using Authenticated Channels*. PhD Thesis, École polytechnique fédérale de Lausanne, 2009.

PV06.   S. Pasini and S. Vaudenay. SAS-based authenticated key agreement. In *PKC 2006*, *LNCS* 3958, pages 395–409. Springer, Heidelberg, April 2006.

SHA04. Secure hash standard. National Institute of Standards and Technology, NIST FIPS PUB 180-2, U.S. Department of Commerce, February 2004.

Vau05. S. Vaudenay. Secure communications over insecure channels based on short authenticated strings. In *CRYPTO 2005*, *LNCS* 3621, pages 309–326. Springer, Heidelberg, August 2005.