Authenticated Key Agreement with Key Re-use in the Short Authenticated Strings Model

Stanisław Jarecki and Nitesh Saxena

[1] University of California, Irvine
[2] Polytechnic Institute of New York University

Abstract. Serge Vaudenay [20] introduced a notion of Message Authentication (MA) protocols in the Short Authenticated String (SAS) model. A SAS-MA protocol authenticates arbitrarily long messages sent over insecure channels as long as the sender and the receiver can additionally send a very short, e.g. 20 bit, *authenticated* message to each other. The main practical application of a SAS-MA protocol is Authenticated Key Agreement (AKA) in this communication model, i.e. SAS-AKA, which can be used for so-called "pairing" of wireless devices. Subsequent work [9,12,10] showed three-round SAS-AKA protocols. However, the Diffie-Hellman (DH) based SAS-AKA protocol of [10] requires choosing fresh DH exponents in each protocol instance, while the generic SAS-AKA construction given by [12] applies only to AKA protocols which have no *shared state* between protocol sessions. Therefore, both prior works exclude the most efficient, although not perfect-forward-secret, AKA protocols that re-use private keys (for encryption-based AKAs) or DH exponents (for DH-based AKAs) across multiple protocol sessions.

In this paper, we propose a novel three-round *encryption-based* SAS-AKA protocol, using non-malleable commitments and CCA-secure encryption as tools, which we show secure (but without perfect-forward secrecy) if each player re-uses its private/public key across protocol sessions. The cost of this protocol is dominated by a single public key encryption for one party and a decryption for the other, assuming the Random Oracle Model (ROM). When implemented with RSA encryption the new SAS-AKA protocol is especially attractive if the two devices being paired have asymmetric computational power (e.g., a desktop and a keyboard).

1 Introduction

Serge Vaudenay [20] introduced a notion of a message authentication protocol (MA) based on so-called short authenticated strings (SAS). Such a protocol allows authenticating messages of arbitrary sizes (sent over insecure channel) making use of an auxiliary channel which can authenticate short, e.g. 20-bit, messages. It is assumed that an adversary has complete control over the insecure channel, i.e., it can eavesdrop, delay, drop, replay, inject and/or modify messages, while the only restriction on the auxiliary channel is that the adversary cannot modify or inject messages on it, but it can eavesdrop, delay, drop, or replay them. Crucially, no other infrastructure assumptions are made, i.e. the players do not share any keys or passwords, and there is no Public Key Infrastructure they can use. The only leverage for establishing security is this bandwidth-restricted, public but authenticated “SAS channel” connecting every pair of players.

The primary application of SAS-MA protocols is to enable SAS-based authenticated key agreement (SAS-AKA) between devices with no reliance on key pre-distribution or a public-key infrastructure. A perfectly fitting and urgently needed application of SAS-AKA protocols is establishing secure communication channels between two devices communicating over a publicly-accessible medium (such as Bluetooth, WiFi), which in addition can also send short authenticated messages to each other (and are hence equipped with a SAS channel), given some amount of manual supervision or involvement from the users. (This problem is referred to as “device pairing” in the systems literature.) Implementations of such SAS channels have been proposed for a variety of device types, assuming various user interfaces and different type of manual supervision. In the simplest example of two cell-phones, phone owners can be asked to type a 20 bit string (6 digits) displayed by one phone into the keypad of the other. The systems proposed in [19,1,14,7,16,18,13] show that the same effect can be accomplished with more primitive devices (e.g., with no keypads) or with less user involvement (e.g. relying on sound, blinking LED lights, cameras on the phones, etc). In all of these schemes, it is desirable to have SAS-AKA protocols which are inexpensive both in computation and communication, since the underlying devices might have limited computation and battery power, and which provably achieve an optimal $2^{-k} + \epsilon$ bound on the probability of adversary’s attack given a k-bit SAS channel, where ϵ is a negligible factor in the security parameter independent of k. The SAS-AKA protocol we propose in this paper significantly improves upon the first goal compared to the previous work, at the expense of achieving a slightly weaker bound on adversary’s attack, namely $2^{-k+1} + \epsilon$.

1.1 Prior Work on SAS-MA Protocols

Following [20,12], we refer to a bi-directional message authentication protocol in the SAS model as SAS-MCA, which stands for “message *cross*-authentication”. Note that two instances of a SAS-MA protocol run in each direction always yield such SAS-MCA scheme, but at twice the cost of the underlying SAS-MA scheme. A straightforward solution for a SAS-MCA was suggested by Balfanz, et al. [1]: Devices A and B exchange the messages m_A, m_B over the insecure channel, and the corresponding hashes $H(m_A)$ and $H(m_B)$ over the SAS channel. Although non-interactive, the protocol requires H to be a collision-resistant hash function and therefore it needs at least 160 bits of the SAS bandwidth in each direction. Pasini and Vaudenay [11] showed a non-interactive protocol which weakens the requirement on the hash function to weak (i.e. second-preimage) collision resistance, and reduces the SAS bandwidth to 80-bits. The ’MANA’ protocols in Gehrmann et al. [6] reduce the SAS bandwidth to any k bits while assuring the 2^{-k} bound on attack probability,[1] but these protocols require a stronger assumption on the SAS channel, namely the adversary is assumed to be incapable of delaying or replaying

[1] Formally, by “2^{-k} bound on attack probability” we mean that the probability that any adversary that runs in time polynomial in a security parameter n, which is independent of the SAS-bandwidth k, succeeds against a single instance of the protocol is upper-bounded by $2^{-k} + \epsilon(n)$, where $\epsilon(n)$ is negligible in n.

$$\begin{array}{ccc}
\underline{\mathbf{P}_i(m)} & & \underline{\mathbf{P}_j} \\
\text{Pick } R_i \leftarrow \{0,1\}^k & & \\
(c,d) \leftarrow \mathsf{com}([m|R_i]) & \xrightarrow{\quad m,c \quad} & \text{Pick } R_j \leftarrow \{0,1\}^k \\
 & \xleftarrow{\quad R_j \quad} & \\
 & \xrightarrow{\quad d \quad} & \\
SAS = R_i \oplus R_j & \xRightarrow{\quad SAS \quad} & [m|R_i] \leftarrow \mathsf{open}(c,d) \\
 & & \text{Output } (P_i, m) \text{ if } SAS = R_i \oplus R_j
\end{array}$$

Fig. 1. V-MA : unidirectional SAS-MA authentication (P_i to P_j) of Vaudenay [20]

the SAS messages, which in practice requires synchronization between the two devices, e.g. one device never abandons one session and restarts another session without the other device also doing the same.

In [20], Vaudenay presented the first SAS-MA scheme, called V-MA and depicted in Figure 1, with the analysis that bounds the attack probability by 2^{-k} for a k-bit SAS channel. In [20] this protocol is shown secure under the assumption that the commitment scheme satisfies what Vaudenay refers to as "extractable commitment", and subsequently [9] pointed out that this proof goes through under the more standard and possibly weaker assumption of a non-malleable commitment. The bi-directional SAS-MCA protocol presented in [20] results from running two instances of the V-MA protocol, one for each direction, but with each player $P_{i/j}$ using the same challenge $R_{i/j}$ in both protocol instances. This SAS-MCA scheme requires 4 communication rounds over the insecure channel and was shown to give a 2^{-k} security bound.

In subsequent work, Laur, Asokan, and Nyberg [9,10] and Pasini and Vaudenay [12] independently gave three-round SAS-MCA protocols. Both schemes are modifications of the V-MA protocol of Figure 1, and both employ (although differently) a universal hash function in computation of the SAS message. Both of these protocols make just a few symmetric key operations if the commitment scheme is implemented using a cryptographic hash function modeled as a Random Oracle. Both protocols claim the 2^{-k} security bound at least in the ROM model, although the scheme of [9,10] was analyzed only in a "synchronized" setting where the same pair of players never execute multiple parallel protocol instances with each other[2] (see Theorem 3, Note 5 of [10]).

1.2 Prior Work on SAS-AKA Protocols

Pasini and Vaudenay [12] argue that one can construct a 3-round SAS-based key agreement protocol (SAS-AKA), from any 3-round SAS-based message cross-authentication protocol (SAS-MCA) like the SAS-MCA protocol presented in [12], and any 2-round

[2] While in practice it might be the case that a pair of players is not *supposed* to execute several protocol instances concurrently, a man-in-the-middle can cause that several instances of the protocol between the same pair of players are effectively alive, if he manages to force one device to time-out and start a new session while the other device is still waiting for an answer.

key agreement scheme (KA) which is secure over authenticated links, e.g. a Diffie-Hellman or an encryption-based KA scheme. The idea is to run the 2-round KA protocol over an insecure channel, and authenticate the two messages m_1, m_2 produced by the KA protocol using the SAS-MCA protocol. (To achieve a 3-round SAS-AKA protocol, the KA messages m_1, m_2 are piggybacked with the SAS-MCA protocol messages.) This compilation is significantly different from the standard compilation from a protocol secure over authenticated links to a protocol secure over insecure channels, which works by running a separate unidirectional message-authentication sub-protocol (MA) for each message of the underlying protocol, e.g. as in Canetti and Krawczyk's MA + KA $\rightarrow$ AKA compilation in [4]. If the SAS-MA authentication protocol has k rounds then this compilation would result in a $2k$-round SAS-AKA scheme, because the responder cannot, in general, send the second KA message m_2 before successful completion of the SAS-MA sub-protocol that authenticates the first KA message m_1. In contrast, to achieve a $(k + 1)$-round SAS-AKA protocol, the compilation given in [12] prescribes that the second message of the KA protocol, m_2, is sent by the responder straight away, i.e. on the basis of the first KA message m_1, which at this moment has not been authenticated yet.

The compilation of Pasini and Vaudenay does result in secure 3-round SAS-AKA schemes, but only when it utilizes a KA scheme which does not keep shared state between different instances of the KA protocol run by the same player. (This was indeed the implicit assumption taken by the proof of security for this compilation given in [12].) Moreover, such SAS-MCA + KA $\rightarrow$ SAS-AKA compilation cannot be applied to KA schemes which *do* share state between instances. For a simple counter-example, consider a 2-round KA protocol secure in the authenticated links model, which is amended so that (1) the computed session key is sent in the last message encrypted using responder P_j's long term public key pk_{ij} chosen for a particular initiator P_i, and (2) the responder P_j reveals the corresponding private key sk_{ij} if the initiator P_i's first message is a special symbol which is never used by an honest sender. Such protocol remains secure in the authenticated links model (in the static corruption case), because only a dishonest sender P_i can trigger P_j to reveal sk_{ij}. However, this protocol is insecure when compiled using the method above, because when P_j computes its response it does not know if the message sent by P_i is authentic, and thus a man-in-the-middle adversary can trigger P_j to reveal sk_{ij} by replacing P_i's initial message in the KA protocol with that special symbol. This way the adversary's interference in a single protocol session leads to revealing the keys on *all* sessions shared between the same pair of players, and thus the compiled protocol is not a secure SAS-AKA. (We elaborate on this counter-example in more detail in Appendix A.)

Independently, Laur and Nyberg also proposed a SAS-AKA protocol [10], based on their own SAS-MCA protocol [9]. In this (Diffie-Hellman based) SAS-AKA protocol, the Diffie-Hellman exponents are picked afresh in each protocol instance, and so this protocol also does not support key re-use across multiple sessions.

1.3 Limitations of SAS-AKA Protocols Without Key Re-use

The key agreement protocols that do not share state between sessions, and thus in particular do not allow for re-use of private keys, are by definition Perfect-Forward Secret

(PFS) but they are also significantly more expensive than non-PFS key agreement protocols. Specifically, the standard Diffie-Hellman PFS KA requires two exponentiations per player, while the encryption-based PFS KA requires generation of a (public,private) key pair and a decryption operation by one player, and a public key encryption by the other player. These are also the dominant costs of the corresponding SAS-AKA schemes implied by the above results of [9,12]. In contrast, the non-PFS Diffie-Hellman with fixed exponents costs only one exponentiation per player, and the encryption-based KA costs one decryption for one player and one encryption for the other. Note that in practice the efficiency of the non-PFS KA schemes often takes precedence over the stronger security property offered by perfect forward secret KA schemes. For example, even though SSL supports PFS version of Diffie-Hellman KA, almost all commercial SSL sessions run the non-PFS encryption-based KA using RSA encryption, since this mode offers dramatically faster client's time (and twice faster server's time). Also, just as the asymmetric division of work in the RSA-encryption based key agreement was attractive for the SSL applications, the same asymmetric costs in the RSA-encryption based SAS-AKA could be attractive for "pairing" of devices with unequal computational power, e.g. a PC and a keyboard, a PC and a cell-phone, or a cell-phone and an earset speaker.

Other applications could also benefit from SAS-AKA protocols which allow for re-use of public keys across multiple protocol sessions. One compelling application is in secure initialization of a sensor network [17]. Sensor initialization can be achieved by the base station simultaneously executing an instance of the SAS-AKA protocol with each sensor. However, since the number of sensors can be large, generating fresh (RSA or DH) encryption keys per protocol instance would impose a large overhead on the base station. An encryption-based SAS-AKA protocol with re-usable public key would be especially handy because it would minimize sensors' computation to a single RSA encryption, and the base station would pick one RSA key pair and then perform one RSA decryption per each sensor. Another application where key re-use in SAS-AKA offers immediate benefits is protection against so-called "Evil Twin" attacks in a cyber-cafe, where multiple users run SAS-AKA protocols to associate their devices with one central access point [15].

1.4 Our Contributions

In this work, we present a provably secure and minimal cost SAS-AKA scheme which re-uses public key pairs across protocol sessions and thus presents a lower-cost but non-PFS alternative to the perfect-forward secret SAS-AKA protocols of [10,12]. Our SAS-AKA relies on a non-malleable commitments just like the SAS-AKA schemes of [20,9,12], but unlike the previous schemes it is built directly on CCA-secure encryption, and it relies on encryption not just for key-establishment but also for authentication security. As a consequence, the new SAS-AKA is somewhat simpler than the previous SAS-AKA's which were built on top of the three-round SAS-MCA's of [9,12], and in particular it does not need to use universal hash functions. However, the most important contribution of the new SAS-AKA scheme is that it remains secure if each player uses a permanent public key, and hence shares a state across all protocol sessions it executes. This leads to two minimal-cost 3-round non-PFS SAS-AKA protocols where the same public/private key pair or the same Diffie-Hellman random contribution is re-used across protocol instances. Specifically, when instantiated with the hash-based commitment and the CCA-secure OAEP-RSA, this implies a 3-round SAS-AKA protocol secure under the RSA assumption in ROM, with the cost of a single RSA encryption for the responder and a single RSA decryption for the initiator. When instantiated with the randomness-reusing CCA-secure version of ElGamal [3] this implies a 3-round SAS-AKA protocol secure under the DH assumption in ROM, with the cost of one exponentiation per player. In other words, the costs of the SAS-AKA protocols implied by our result are (for the first time) essentially the same as the costs of the corresponding basic unauthenticated key agreement protocols. By contrast, previously known *PFS* SAS-AKA protocols require two exponentiations per player if they are based on DH [12,10] or a generation of fresh public/private RSA key pair for each protocol instance if the general result of [12] is instantiated with an RSA-based key agreement.

We note that the SAS-MCA/AKA protocol we show secure is very similar to the SAS-AKA protocols of [20,9,12], and it is indeed only a new variant of the same three-round commitment-based SAS-MA protocol analyzed in [20], which also forms a starting point for protocols of [9,12]. However, prior to our work there was no argument that such SAS-AKA scheme remains secure when players re-use their public/private key pairs across multiple sessions. Moreover, as we explain above, it is unlikely that such result can be proven using a modular argument similar to the one used by [12] for KA protocols that do *not* keep state between protocol instances, which is also why our analysis of the proposed protocol proceeds "from scratch" rather than proceeding in a modular fashion based on already known properties of Vaudenay's SAS-MA scheme. Secondly, our analysis shows that the SAS-AKA protocol can be simpler than even a standard encryption-based KA protocol executed over the 3-round SAS-MCA protocol of [9] or [12]. In fact, our protocol consists of a single instance of the basic *unidirectional* SAS-MA scheme of [20], shown in Figure 1, which authenticates only the initiator's message, but this message includes the initiator's (long-term) public key, which the responder uses to encrypt its message. It turns out that this encryption not only transforms this protocol to a SAS-AKA scheme but also authenticates responder's message, thus yielding not just a cheaper but also a simpler three-round SAS-AKA protocol.

Paper Organization. Section 2 contains our cryptographic tools. Section 3 contains the communication and adversarial models for SAS-MCA and SAS-AKA protocols. We propose our SAS-MCA / SAS-AKA protocol in Section 4. In the same section we argue that this protocol is a secure SAS-MCA scheme, but for lack of space we relegate the (very similar) argument that this protocol is also a secure SAS-AKA scheme protocol) to the full version of this paper [8].

2 Preliminaries

Public Key Encryption. A public key encryption scheme is a tuple of algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, where KeyGen on input of a security parameter produces a pair of public and secret keys (pk, sk), $\mathsf{Enc}_{pk}(m)$ outputs ciphertext c for message m, and $\mathsf{Dec}_{sk}(c)$ decrypts m from $c = \mathsf{Enc}_{pk}(m)$. In the SAS-MCA/AKA protocol construction, the encrypted messages come from a special space $\mathcal{M}_{\overline{m}} = \{[\overline{m}|R]$ s.t. $R \in$

$\{0,1\}^k\}$ where $\overline{m}$ is some (adversarially chosen) string. Since this message space contains 2^k elements, a chosen-ciphertext secure encryption ensures that an adversary who is given an encryption of a random message in this space can predict this message with probability at most negligibly higher than 2^{-k}. Namely, the following is a simple fact about CCA-secure encryption.

Fact 1. *If an encryption scheme is* (T,ϵ)-$\mathsf{SS\text{-}CCA}$ *then for every* T*-bounded algorithm* $\mathcal{A}$ *and every* $\overline{m}$*,*

$$\Pr[\mathcal{A}^{\mathsf{Dec}^{C}_{sk}(\cdot)}(pk,C)=\hat{m}\mid(pk,sk)\leftarrow\mathsf{KeyGen},\ \ m\leftarrow\mathcal{M}_{\overline{m}},$$

$$C\leftarrow\mathsf{Enc}_{pk}(m)]\leq\ 2^{-k}+\epsilon$$

where $\mathsf{Dec}^{C}_{sk}(\cdot)$ *is a decryption oracle except it outputs* $\perp$ *on* C*.*

Commitment Schemes. Similarly to the SAS-channel message authentication protocols given before by [20,9,12], the protocols here are also based on commitment schemes with some form of non-malleability. In fact, the assumption on commitment schemes we make is essentially the same as in the SAS-MCA protocols of [20,12], but we slightly relax (and re-name) this property of commitment schemes here, so that, in particular, it is satisfied by a very efficient hash-based commitment scheme in the ROM model for a hash function.

The commitment scheme consists of following three functions: gen generates a public parameter K_p on input a security parameter, $\mathsf{com}_{K_p}(m)$, on input of message m, outputs a pair of a "commitment" c and "decommitment" d, and $\mathsf{open}_{K_p}(c,d)$, on input (c,d), either outputs some value m' or rejects. This triple of algorithms must meet a completeness property, namely for any K_p generated by gen and for any m, if (c,d) is output by $\mathsf{com}_{K_p}(m)$ then $\mathsf{open}_{K_p}(c,d)$ outputs m. We assume a *common reference string* (CRS) model, where a trusted third party generates the commitment key K_p and this key is then embedded in every instance of the protocol. Therefore, we will use a simplified notation, and write $\mathsf{com}(m)$ and $\mathsf{open}(c,d)$ without mentioning the public parameter K_p explicitly. For simplicity of notation in the SAS-MCA/AKA protocols, we sometimes use $m_2\leftarrow\mathsf{open}(m_1,c,d)$ do denote a procedure which first does $m\leftarrow\mathsf{open}(c,d)$ and then compares if m is of the form $m=[m_1|m_2]$ for the given m_1. If it is, the modified open procedure outputs m_2, and otherwise it rejects.

Non-Malleable Commitment Scheme. In our protocols, we use the same notion of non-malleable commitments as in [9], adopted from [5]. An adversary is a quadruple $\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3,\mathcal{A}_4)$ of efficient algorithms interacting with Challenger. $(\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3)$ represents an active part of the adversary that creates and afterwards tries to open related commitments and $\mathcal{A}_4$ represents a distinguisher. Challenger is initialized to be in either of two environments, called "World_0" and "World_1". $\mathcal{A}$ succeeds if $\mathcal{A}_4$ can distinguish between these two environments World_0 and World_1.

Challenger first runs gen to produce K_p and sends it to $\mathcal{A}_1$. $\mathcal{A}_1$ outputs a message space $\mathcal{M}$ along with state σ and sends it back to Challenger. Challenger picks two messages m_0 and m_1 at random from $\mathcal{M}$ and computes a challenge commitment $(c,d)=\mathsf{com}_{K_p}(m_1)$ and sends c it to $\mathcal{A}_2$. $\mathcal{A}_2$ in turn responds with a commitment c^*. Challenger aborts if any $c^*=c$, and otherwise sends d to $\mathcal{A}_3$. Now, $\mathcal{A}_3$

must output a valid decommitment d^*. Challenger computes $y^* = \mathsf{open}_{K_p}(c^*, d^*)$. If $y^* = \perp$, then $\mathcal{A}$ is halted. Finally, in the environment World_0, Challenger invokes $\mathcal{A}_4$ with inputs (m_0, y^*), whereas in World_1, it invokes $\mathcal{A}_4$ with inputs (m_1, y^*). A commitment scheme is (T, ϵ)-NM (non-malleable) iff for any t time adversary $\mathcal{A}$ it holds that $Adv_{com}^{\mathsf{NM}}(\mathcal{A}) = |\Pr[\mathcal{A}_4 = 1|\mathsf{World}_1] - \Pr[\mathcal{A}_4 = 1|\mathsf{World}_0]| \leq \epsilon$.

For notational convenience, we use a specialization of this non-malleability notion to message space $\mathcal{M}_{\overline{m}} = \{[\overline{m}|R]$ s.t. $R \in \{0,1\}^k\}$, which our SAS-MCA/AKA protocol deals with, and to a particular simple type of tests which our reductions use to distinguish between the two distributions above. Namely, we say that the commitment scheme is (T, ϵ)-NM if for every T-limited adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, it holds that $Pr[m^* \oplus m = \sigma \mid K_P \leftarrow \mathsf{gen}; (\overline{m}, s) \leftarrow \mathcal{A}_1(K_P); m \leftarrow \mathcal{M}_{\overline{m}}; (c, d) \leftarrow \mathsf{com}_{K_P}(m); (c^*, \sigma) \leftarrow \mathcal{A}_2(c, s); d^* \leftarrow \mathcal{A}_3(c, d, s); m^* = \mathsf{open}_{K_P}(c^*, d^*)] \leq 2^{-k} + \epsilon$

Note that a (T, ϵ)-NM commitment scheme can be created from any (T, ϵ)-SS-CCA encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ [5]. The (K_s, K_p) is a private/public key pair (sk, pk) of the encryption scheme. $\mathsf{com}_{pk}(m)$ picks a random string r and outputs $c = \mathsf{Enc}_{pk}(m; r)$ and $d = (m, r)$, where $\mathsf{Enc}_{pk}(\cdot; r)$ denotes the (randomized) encryption procedure with randomness r. Procedure $\mathsf{open}_{pk}(c, (m, r))$ outputs m if $c = \mathsf{Enc}_{pk}(m; r)$ and $\perp$ otherwise.

Non-Malleable Commitment in the Random Oracle Model (ROM). One can make a fast and simple commitment scheme using a hash function $H : \{0,1\}^* \rightarrow \{0,1\}^{l'}$ modeled as a random oracle, where the adversary's advantage in the NM-Security game can be set arbitrarily low at very little cost. Generator gen in this scheme is a null procedure, $\mathsf{com}(m)$ picks $r \in \{0,1\}^l$ and returns $c = H(m, r)$ and $d = (m, r)$, $\mathsf{open}(c, (m, r))$ returns m if $c = H(m, r)$ and $\perp$ otherwise. This scheme is (T, ϵ)-NM for $\epsilon = q_H 2^{-l} + q_H^2 2^{-l'}$, where q_H is the number of H-function queries that can be made by a T-bounded adversary $\mathcal{A}$. This is because the probability that $\mathcal{A}_2$ learns anything about the value committed by the challenger is $q_H 2^{-l}$ because the only information $\mathcal{A}_2$ can get on m chosen by the challenger is by querying hash function H for some $m \in \mathcal{M}$ and r used by the challenger, but the probability that $\mathcal{A}$ hits the same r as the challenger is bounded by $q_H 2^{-l}$. Moreover, the probability that $\mathcal{A}_3$ is able to decommit to more than one value is bounded by $q_H^2 2^{-l'}$, because this is the probability that within q_H queries to H, the adversary gets a pair of values which collide.

3 Communication and Adversarial Model

3.1 Network and Communication Setting

We consider the same model as in [20,9,12], but we explicitly cast it in the multi-player/multi-session world. In other words, we consider a network consisting of n players $P_1, \cdots, P_n$. Each ordered pair of players (P_i, P_j) is connected by two unidirectional point-to-point communication channels: (1) an insecure channel, e.g. internet or a Bluetooth or a WiFi channel, over which an adversary has complete control by eavesdropping, delaying, dropping, replaying, and/or modifying messages, and (2) a low-bandwidth out-of-band authenticated (but not secret) channel, referred to as a *SAS channel* from here on, which preserves the integrity of messages and also provides source and target authentication. In other words, on the insecure channel, an adversary can behave arbitrarily, but it is *not* allowed to modify (or inject) messages sent on the SAS channel (which we'll call *SAS messages* for short), although it can still read them, as well as delay, drop, or re-order them.

3.2 SAS-MCA and Its Security

Our security model follows the Canetti-Krawczyk model for authenticated key exchange protocols [4], and the earlier work of [2], which allows modeling concurrent executions of multiple protocol instances. While in practice it will very often be the case (e.g. in the device pairing application) that a single player is not *supposed* to execute several protocol instances concurrently, a man-in-the-middle can cause that several instances of the protocol between the same pair of players are effectively alive, if he manages to force device A to time-out and start a new SAS-AKA protocol session, while device B is still waiting for an answer. In this case the adversary can choose which messages to forward to device B among the messages sent on the different sessions started by device A.

A SAS-MCA protocol is a "cross-party" message authentication protocol, executed between two players P_i and P_j, whose goal is for P_i and P_j to send authenticated messages to one another. We denote the τ-th protocol instance run by a player P_i as Π_i^τ, where τ is a locally unique index. The inputs of Π_i^τ are a tuple $(\mathsf{role}_i^\tau, P_j, m_i^\tau)$ where role_i^τ designates P_i as either the initiator ("$init$") or a responder ("$resp$") in this instance of the SAS-MCA protocol, P_j identifies the communication partner for this protocol instance, i.e. it identifies a pair of SAS channels $(P_i \rightarrow P_j)$ and $(P_i \leftarrow P_j)$ with an entity (P_j) with whom P_i's application wants to communicate, and m_i^τ is the message to be sent to P_j in this session. With each session Π_i^τ there is associated a unique string sid_i^τ, which is a concatenation of all messages sent and received on this session, including the messages on the SAS channel. We denote input P_j on session Π_i^τ as $\mathsf{Peer}(\Pi_i^\tau)$. We say that sessions Π_i^τ and Π_j^η executed by two different players are **matching** if $\mathsf{Peer}(\Pi_i^\tau) = P_j$, $\mathsf{Peer}(\Pi_j^\eta) = P_i$, and $\mathsf{role}_j^\eta \neq \mathsf{role}_i^\tau$. We say that the sessions are **partnered** if they are matching and their messages are properly exchanged between them, i.e. $sid_i^\tau = sid_j^\eta$. By the last requirement, and by inclusion of random nonces in the protocol, we ensure that except of negligible probability each session can be partnered with at most one other session. The output of Π_i^τ can be either a tuple $(\mathsf{Peer}(\Pi_i^\tau), m_i^\tau, \hat{m}_i^\tau, sid_i^\tau)$, for some $\hat{m}_i^\tau$, or a rejection. Similarly, Π_j^η can either output $(\mathsf{Peer}(\Pi_j^\eta), m_j^\eta, \hat{m}_j^\eta, sid_j^\eta)$ or reject. The SAS-MCA protocol should satisfy the following **correctness** condition: If sessions Π_i^τ and Π_j^η are partnered then both sessions accept and output the messages sent by the other player, i.e. $\hat{m}_i^\tau = m_j^\eta$ and $\hat{m}_j^\eta = m_i^\tau$.

We model the **security** of a SAS-MCA protocol via a following game between the challenger performing the part of the honest players $P_1, ..., P_n$, and the adversary $\mathcal{A}$. We consider only the *static* corruption model, where the adversary does not adaptively corrupt initially honest players. The challenger and the adversary communicate by exchanging messages as follows: At the beginning of the interaction, the challenger initializes the long-term private state of every player P_i, e.g. by generating a public/private key pair for each player. In the rest of the interaction, the challenger keeps the state of every initialized protocol instance and follows the SAS-MCA protocol on its behalf. $\mathcal{A}$ can trigger a new protocol instance Π_i^τ on inputs (role, P_j, m) by issuing a query $\mathsf{launch}(\Pi_i^\tau, \mathsf{role}, P_j, m)$. The challenger responds by initializing the state of session Π_i^τ and sending back to $\mathcal{A}$ the message this session generates. If $\mathcal{A}$ issues a query $\mathsf{send}(\Pi_i^\tau, M)$ for any previously initialized Π_i^τ and any M, the challenger delivers message M to session Π_i^τ and responds by following the SAS-MCA protocol on its behalf, handing the response of Π_i^τ on M to $\mathcal{A}$. However, if Π_i^τ outputs a SAS message, the challenger hands this message to $\mathcal{A}$ and adds it to a multiset $\mathsf{SAS}(i,j)$, for $P_j = \mathsf{Peer}(\Pi_i^\tau)$, which models the unidirectional SAS channel from P_i to P_j, denoted $\mathsf{SAS}(P_i \rightarrow P_j)$. $\mathcal{A}$ can issue a $\mathsf{SAS\text{-}send}(\Pi_j^\tau, M)$ query for any message M in set $\mathsf{SAS}(i,j)$, where $P_i = \mathsf{Peer}(\Pi_j^\tau)$. The challenger then removes element M from $\mathsf{SAS}(i,j)$ and delivers M on the $\mathsf{SAS}(P_j \rightarrow P_i)$ channel to Π_i^τ. This models the fact that the adversary can see, stall, delete, and re-order messages on each $\mathsf{SAS}(P_i \rightarrow P_j)$ channel, but $\mathcal{A}$ cannot modify, duplicate, or add to any of the messages on such channel.

We say that $\mathcal{A}$ *wins* in attack against SAS-MCA if there exists session Π_i^τ which outputs (P_j, m_i, m_j, sid) but there is no session Π_j^η which ran on inputs $(*, P_i, m_j)$. In other words, if Π_i^τ outputs a message m_j as sent by P_j but P_j did not send m_j to P_i on any session. We call a SAS-MCA protocol (T, ϵ)-**secure** if for every adversary $\mathcal{A}$ running in time T, $\mathcal{A}$ wins with probability at most ϵ. Note that in the SAS-MCA game the adversary can launch multiple concurrent sessions among every pair of players. To make our security results concrete in the multi-player setting, we will consider an (n, τ_t, τ_c)-attacker $\mathcal{A}$ against the SAS-MCA protocol, where the above game is restricted to n players P_i, at most τ_t total number of sessions per player, and at most τ_c sessions that can be concurrently held by any *pair* of players, i.e. $\mathsf{SAS}(i,j) \leq \tau_c$ for all i, j. We note that the τ_c bound is determined by how long the adversary can lag the SAS messages, how many sessions he can cause to re-start at one side, and how long he can keep alive a session waiting for its SAS message on the other side. In many applications it will be rather small, but it is important to realize that in many applications it is greater than 1.

3.3 SAS-AKA and Its Security

SAS-AKA is an Authenticated Key Agreement (AKA) protocol in the SAS model. The inputs to the protocol are as in the SAS-MCA but with no messages. Each instance Π_i^τ outputs either a rejection or a tuple $(\mathsf{Peer}(\Pi_i^\tau), K, sid)$, where K is a fresh, authenticated, and secret key which P_i hopes to have shared with $P_j = \mathsf{Peer}(\Pi_i^s)$, and sid is a locally unique session id. An SAS-AKA scheme protects the secrecy of keys output by honest players on sessions involving other uncorrupted player. The correctness property for a SAS-AKA protocol is that if two sessions Π_i^τ and Π_j^η are partnered then both sessions accept and output the same key $K_i^\tau = K_j^\eta$.

We model **security** of the SAS-AKA protocol similarly as in the SAS-MCA case, by an interaction between the (n, τ_t, τ_c)-attacker $\mathcal{A}$ and the challenger that operates the network of n players $P_1, ..., P_n$. In this game, however, the challenger has a *private input* of bit b. The rules of communication model between the challenger and $\mathcal{A}$ and the set-up of all honest players are the same as in the SAS-MCA game above, and the challenger services $\mathcal{A}$'s requests launch, send, and $\mathsf{SAS\text{-}send}$ in the same way as in the SAS-MCA game, except that there's no message in inputs to the launch request. In addition, $\mathcal{A}$ can issue a query of the form reveal(Π_i^τ) for any Π_i^τ, which gives him the key K_i^τ output by Π_i^τ if this session computed a key, and a null value otherwise. Finally, on one of the sessions Π_i^τ subject to the constraints specified below, the adversary can issue a Test(Π_i^τ) query. If Π_i^τ has not completed, the adversary gets a null value. Otherwise, if $b = 1$ then $\mathcal{A}$ gets the key K_i^τ, and if $b = 0$ then $\mathcal{A}$ gets a random bitstring of the same length. The constraint on the tested session Π_i^τ is that the adversary issues no reveal(Π_i^τ) query *and* no reveal(Π_j^η) query for any Π_j^η which is partnered with Π_i^τ. After testing a session, the adversary can then keep issuing the launch, send, SASsend and reveal commands, except it cannot reveal the tested session or a session that is partnered with it. Eventually $\mathcal{A}$ outputs a bit $\hat{b}$. We say that an adversary has *advantage* ϵ in the SAS-AKA attack if the probability that $\hat{b} = b$ is at most $1/2 + \epsilon$. We say that the SAS-AKA protocol is (T, ϵ)-**secure** if for all $\mathcal{A}$'s bounded by time T this advantage is at most ϵ.

We note that the above model includes only *static* corruption patterns. Indeed, the protocols we present here do *not* have perfect forward secrecy, since we are interested in provable security of minimal-cost AKA protocols in which players re-use their private key material across all protocol sessions.

4 Encryption-Based SAS Message Authentication Protocol

In this section, we present a novel 3-round encryption-based bidirectional SAS-MCA protocol denoted Enc-MCA. The protocol is depicted in Figure 2. It runs between the initiator P_i, who intends to authenticate a message m_i, and the responder P_j, who intends to authenticate a message m_j. (SK_i, PK_i) denotes P_i's private/public key pair of an IND-CCA encryption scheme, which w.l.o.g. we assume to be permanent. The protocol assumes the CRS model where the instance K_P of the CCA-Secure commitment scheme is globally chosen. The protocol is based on the *unidirectional* message-authentication V-MA protocol of Vaudenay [20], Figure 1. The only difference is that P_i adds to its message m_i its public key PK_i and a random nonce $s_i \in \{0,1\}^l$, and the responder P_j sends its randomness R_j *encrypted* under PK_i, together with its message m_j and a random nonce $s_j \in \{0,1\}^l$. In other words, P_i sends (m_i, s_i, PK_i) along with a commitment c_i to (m_i, s_i, PK_i, R_i) where R_i is a random k-bit bitstring. P_j replies with an encryption of m_j, s_j, and a random value $R_j \in \{0,1\}^k$. Finally P_i sends to P_j its decommitment d_i to c_i, and P_i and P_j exchange over the SAS channel values $SAS_i = R_i \oplus R_j$, where P_i obtains R_j by decrypting e_j, and $SAS_j = R_i \oplus R_j$, where P_j obtains R_i by opening the commitment c_i. The players accept if the SAS values match, and reject otherwise. P_i and P_j also output session identifiers sid_i and sid_j, respectively, which are outputs of a collision-resistant hash function H on the concatenation of all messages sent (received resp.) and received (sent resp.) on this session, including the messages on the SAS channel. (This is done only for simplicity of security analysis: In fact the same security argument goes through if $sid_i = sid_j = [s_i|s_j]$.) The following theorem states the security of this protocol against an (n, τ_t, τ_c)-adversary:

Enc-MCA Protocol

(We denote as $\hat{v}$ the value received by P_i/P_j if the value sent by P_j/P_i is denoted as v.)

$$\begin{array}{ccc}
\underline{\mathbf{P}_i(P_j,(SK_i,PK_i),m_i,init)} & & \underline{\mathbf{P}_j(P_i,m_j,resp)} \\
\text{Pick } R_i \in \{0,1\}^k,\ s_i \in \{0,1\}^l & & \text{Pick } R_j \in \{0,1\}^k,\ s_j \in \{0,1\}^l \\
(c_i,d_i) \leftarrow \mathsf{com}([m_i|s_i|PK_i|R_i]) & \xrightarrow{m_i,s_i,PK_i,c_i} & \\
 & \xleftarrow{e_j} & e_j = \mathsf{Enc}_{\hat{PK_i}}([m_j|s_j|R_j]) \\
[\hat{m}_j|\hat{s}_j|\hat{R}_j] \leftarrow \mathsf{Dec}_{SK_i}(\hat{e}_j) & \xrightarrow{d_i} & \hat{R}_i \leftarrow \mathsf{open}([\hat{m}_i|\hat{s}_i|\hat{PK}_i],\hat{c}_i,\hat{d}_i) \\
SAS_i = R_i \oplus \hat{R}_j & \xRightarrow{SAS_i} & SAS_j = \hat{R}_i \oplus R_j \\
sid_i = H(m_i,s_i,PK_i,c_i,\hat{e}_j, & \xLeftarrow{SAS_j} & sid_j = H(\hat{m}_i,\hat{s}_i,\hat{PK}_i,\hat{c}_i,e_j, \\
d_i,SAS_i,\hat{SAS}_j) & & \hat{d}_i,\hat{SAS}_i,SAS_j) \\
\text{Output } (P_i,m_i,\hat{m}_j,sid_i) \text{ if} & & \text{Output } (P_j,m_j,\hat{m}_i,sid_j) \text{ if} \\
SAS_j = R_i \oplus \hat{R}_j & & SAS_i = \hat{R}_i \oplus R_j
\end{array}$$

Enc-AKA Protocol

The protocol follows Enc-MCA with m_i set to null and $m_j = K$, for random $K \in \{0,1\}^l$ chosen by P_j. If its SAS test passes player P_j, resp. P_i, outputs m_j $[= K]$, resp. $\hat{m}_j$.

Fig. 2. Encryption-based SAS-MCA protocol (Enc-MCA) and SAS-AKA protocol (Enc-AKA)

Theorem 1 (Security of Enc-MCA). *If commitment scheme is (T_C,ϵ_C)-NM and encryption scheme is (T_E,ϵ_E)-SS-CCA, then the Enc-MCA protocol is (T,p)-secure against (n,τ_t,τ_c)-attacker for $p \geq 2n\tau_t\tau_c(2^{-k}+\max(\epsilon_C,\epsilon_E))$ and $T \leq \min(T_C,T_E)$ $-\mu$, for a small constant μ.*

Note on the Security Claim and the Proof Strategy. The $n\tau_t\tau_c 2^{-k}$ security bound would be optimally achievable in the context of (n,τ_t,τ_c)-adversary because this is the probability, for $n\tau_t\tau_c \ll 2^{-k}$, that the k-bit SAS messages are equal on some two matching sessions, even though the adversary substitutes sender's messages on every session, since there are $n\tau_t$ sessions, each of which can succeed if the SAS message it requires to complete is present among τ_c SAS messages produced by the sessions concurrently executed by its peer player. We note that if adversary's goal is to attack any *particular* player and session, the same theorem applies with values $n = \tau_t = 1$.

However, the security bound $n\tau_t\tau_c 2^{-k+1}$ we show is factor of 2 away from the optimal. This factor is due to the fact that the reduction has to guess whether the adversary essentially attacks the encryption or the commitment tool used in our protocol. This also accounts for the essential difference between our proof and those of [9,12]. Even assuming the simplest $n = \tau_t = \tau_c = 1$ case, there are several patterns of attack, corresponding to three possibilities for interleaving messages and other decisions the adversary can make (in our case the crucial switch is whether or not the adversary modifies the initiator's payload m, s, PK). For each pattern of attack, we provide a reduction, which given an attack that breaks the SAS-MCA/AKA scheme with probability $2^{-k}+\epsilon$, *conditioned on this attack type being chosen*, attacks either the commitment or the encryption scheme with probability ϵ. While some of these component reductions are identical to those shown for the same underlying SAS-MA protocol by Vaudenay in [20], others are different e.g. because they attack the encryption scheme. However, it is not clear how to use such reductions to show any better security bound than $q * 2^{-k}$ where q is the number of such attack cases. Fortunately, we manage to group these attack patterns into just two groups, with two reductions, the first translating *any* attack in the first group into an encryption attack, the second translating *any* attack in the second group into a commitment attack. Crucially, both reductions are non-rewinding, and hence they are security-preserving. However, faced with an adversary which adaptively decides which group his attack will fall in we still need to guess which reduction to follow, hence the bound on attacker's probability we show for our SAS-MCA/AKE scheme is a factor of 2 away from the optimal.

Proof: We prove the above by showing that if there exists (n, τ_t, τ_c)-adversary $\mathcal{A}$ which can attack the proposed protocol in time $T < \min(T_C, T_E) - \mu$ and probability $p > 2n\tau_t\tau_c(2^{-k} + \max(\epsilon_C, \epsilon_E))$, then there exists *either* a $T+\mu < T_C$ adversary $\mathcal{B}_\mathcal{C}$ which breaks NM security of the commitment scheme with probability better than $2^{-k}+\epsilon_C$, *or* there exists a $T+\mu < T_E$ adversary $\mathcal{B}_\mathcal{E}$ which wins the SS-CCA game for the encryption scheme with probability better than $2^{-k} + \epsilon_E$. $\mathcal{A}$ succeeds if it can find a player P_i and a session Π_i^s with a peer party P_j, such that Π_i^s accepts message $\hat{m_j}^{(s)}$ but the adversary never launches an instance of P_j on message $\hat{m_j}^{(s)}$. To achieve this $\mathcal{A}$ in particular has to route to Π_i^s a SAS message $SAS_j{}^{(s')}$ originated by *some* session $\Pi_j^{s'}$ s.t. $\mathsf{Peer}(\Pi_j^{s'}) = P_i$. By inspection of the protocol, Π_i^s accepts only if $R_i{}^{(s)} \oplus \hat{R}_j^{(s)} = \hat{R}_i^{(s')} \oplus R_j{}^{(s')}$, or equivalently, $SAS_i{}^{(s)} = SAS_j{}^{(s')}$. Note that this condition must hold regardless whether the attacked session Π_i^s is an initiator or a responder. This allows us to simplify the notation and in the remainder of the proof we assume Π_i^s is the initiator, $\Pi_j^{s'}$ is the responder, and we assume that *either* $\hat{m}_i^{(s)} \neq m_i{}^{(s)}$ *or* $\hat{m}_j^{(s')} \neq m_j{}^{(s')}$.

In Figure 3 we show adversary's interactions as a man in the middle between Π_i^s and $\Pi_j^{s'}$. Note that $\mathcal{A}$ can control the *sequence* in which the messages received by these two players are interleaved, and $\mathcal{A}$ has a choice of the following three possible sequences:

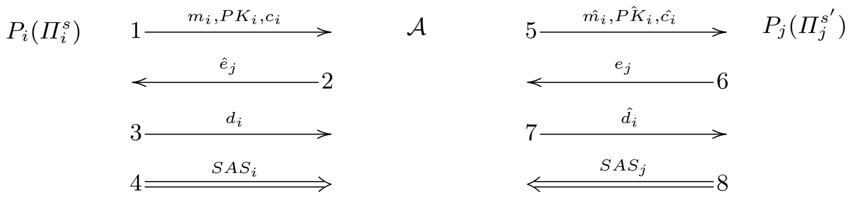

Fig. 3. Adversarial Behavior in the $\mathsf{Enc\text{-}MCA}$ protocol

$$\text{Interleaving pattern I}: \ (1 \prec 5 \prec 6 \prec 2 \prec 3 \prec 4 \prec 7 \prec 8)$$
$$\text{Interleaving pattern II}: \ (1 \prec 5 \prec 6 \prec 7 \prec 8 \prec 2 \prec 3 \prec 4)$$
$$\text{Interleaving pattern III}: \ (1 \prec 2 \prec 3 \prec 4 \prec 5 \prec 6 \prec 7 \prec 8)$$

In each of these three message interleaving patterns we consider two subcases, depending on whether the pair $(\hat{m}_i, \hat{PK}_i)$ that the adversary delivers to $\Pi_j^{s'}$ in message $\#5$ (see Figure 3) is equal to $(m_i; PK_i)$ that Π_i^s sends in message $\#1$.

We denote the event that adversary succeeds in an attack as AdvSc, the event that $(\hat{m}_i, \hat{PK}_i) = (m_i, PK_i)$ *and* that the attack succeeds as SM, the event that $(\hat{m}_i, \hat{PK}_i) \neq (m_i, PK_i)$ *and* that the attack succeeds as NSM, and we use Int[1], Int[2], Int[3] to denote events when the adversary follows, respectively, the 1st, 2nd, or 3rd message interleaving pattern. We divide the six possible patterns which the successful attack must follow into the following two cases:

$\mathsf{Case1} = \mathsf{NSM} \vee (\mathsf{AdvSc} \wedge \mathsf{Int}[2])$ & $\mathsf{Case2} = \mathsf{SM} \wedge (\mathsf{Int}[1] \vee \mathsf{Int}[3])$

We construct two reduction algorithms, $\mathcal{B}_\mathcal{C}$ and $\mathcal{B}_\mathcal{E}$, attacking respectively the NM property of the commitment, and the SS-CCA property of the encryption scheme used in the Enc-MCA protocol. Both $\mathcal{B}_\mathcal{C}$ and $\mathcal{B}_\mathcal{E}$ use the Enc-MCA attacker $\mathcal{A}$ as a black box, and both reductions have only constant computational overhead which we denote as μ, hence both $\mathcal{B}_\mathcal{C}$ and $\mathcal{B}_\mathcal{E}$ run in time at most $T - \mu < \min(T_C, T_E)$. We show that if $\Pr[\mathsf{Case1}] \geq p/2$ then $\mathcal{B}_\mathcal{C}$ wins the NM game with probability greater than $2^{-k} + \epsilon_C$, and if $\Pr[\mathsf{Case2}] \geq p/2$ then $\mathcal{B}_\mathcal{E}$ wins the SS-CCA game with probability greater than $2^{-k} + \epsilon_E$. This will complete the proof because $\mathsf{AdvSc} = \mathsf{Case1} \cup \mathsf{Case2}$, and therefore if $\Pr[\mathsf{AdvSc}] = p$ then either $\Pr[\mathsf{Case1}] \geq p/2$ or $n\tau_t\tau_c)$ or $\Pr[\mathsf{Case2}] \geq p/2$.

Both $\mathcal{B}_\mathcal{C}$ and $\mathcal{B}_\mathcal{E}$ proceed by first guessing the sessions Π_i^s and $\Pi_j^{s'}$ involved in $\mathcal{A}$'s attack. The probability that the guess is correct is at least $1/n\tau_t\tau_c$ because $\mathcal{A}$ runs at most $n\tau_t$ sessions and each session can have at most τ_c concurrently running peer sessions. Since the probability of a correct guess is independent of adversary's view, for either $i = 1$ or $i = 2$, the probability that the guess is correct *and* Case$i$ happens is at least $p/2 * 1/n\tau_t\tau_c > 2^{-k} + \max(\epsilon_C, \epsilon_E)$. We show that if $i = 1$ then $\mathcal{B}_\mathcal{C}$ wins in the NM game, and hence its probability of winning is greater than $2^{-k} + \epsilon_C$, and if $i = 2$ then $\mathcal{B}_\mathcal{E}$ wins the SS-CCA game, and hence its probability of winning is greater than $2^{-k} + \epsilon_E$.

It remains for us to construct algorithms $\mathcal{B}_\mathcal{C}$ and $\mathcal{B}_\mathcal{E}$ with the properties claimed above. Algorithm $\mathcal{B}_\mathcal{C}$, depending on the behavior of $\mathcal{A}$, executes one of the three sub-algorithms, $\mathcal{B}_\mathcal{C}$[i] for $i = 1, 2, 3$. These three algorithms correspond to three cases of message interleaving by the adversary. For lack of space we relegate these reductions to the full version [8], but each of these reductions are attacks the non-malleability of the commitment scheme, so each of them is essentially the same as the reduction given by Vaudenay [20] for the corresponding message interleaving pattern for the underlying MCA protocol. More specifically:

If $(\hat{m}_i, \hat{s}_i, \hat{PK}_i) \neq (m_i, s_i, PK_i)$ and $\mathcal{A}$ chooses interleaving pattern I or III, then $\mathcal{B}_\mathcal{C}$ executes sub-algorithms, respectively, $\mathcal{B}_\mathcal{C}[1]$ and $\mathcal{B}_\mathcal{C}[3]$.
If $\mathcal{A}$ chooses interleaving pattern II, $\mathcal{B}_\mathcal{C}$ executes $\mathcal{B}_\mathcal{C}[2]$.
Otherwise, i.e. if $\mathcal{A}$ sends $(\hat{m}_i, \hat{s}_i, \hat{PK}_i) = (m_i, s_i, PK_i)$ and $\mathcal{A}$ follows patterns I or III, $\mathcal{B}_\mathcal{C}$ fails.

Similarly, based on the behavior of $\mathcal{A}$, algorithm $\mathcal{B}_{\mathcal{E}}$ executes one of two sub-algorithms $\mathcal{B}_{\mathcal{E}}$[i] for $i = 1, 2$. In contrast to the original MCA protocol of Vaudenay, these two reductions attack CCA security of encryption. We show reduction $\mathcal{B}_{\mathcal{E}}[1]$ in Figure 4. For lack of space we relegate reduction $\mathcal{B}_{\mathcal{E}}[2]$ to the full version [8], but it is easy to reconstruct given the message interleaving pattern it involves, and it is similar to $\mathcal{B}_{\mathcal{E}}[1]$. The $\mathcal{B}_{\mathcal{E}}$ algorithms proceeds in one of the following ways:

If $(\hat{m}_i, \hat{s}_i, \hat{PK}_i) = (m_i, s_i, PK_i)$ and $\mathcal{A}$ chooses interleaving pattern I, $\mathcal{B}_{\mathcal{E}}$ executes $\mathcal{B}_{\mathcal{E}}[1]$.
If $(\hat{m}_i, \hat{s}_i, \hat{PK}_i) = (m_i, s_i, PK_i)$ and $\mathcal{A}$ chooses interleaving pattern III, $\mathcal{B}_{\mathcal{E}}$ executes $\mathcal{B}_{\mathcal{E}}[2]$.
Otherwise, i.e. if $\mathcal{A}$ sends $(\hat{m}_i, \hat{s}_i, \hat{PK}_i) \neq (m_i, s_i, PK_i)$ or $\mathcal{A}$ follows interleaving pattern II, $\mathcal{B}_{\mathcal{E}}$ fails.

Note that if $(\hat{m}_i, \hat{s}_i, \hat{PK}_i) \neq (m_i, s_i, PK_i)$ then $\mathcal{A}$ essentially attacks the V-MA protocol of Vaudenay, because pair (m_i, PK_i) in the Enc-MCA protocol plays a role of the message in the V-MA protocol, so this event in the Enc-MCA protocol is equivalent to P_j accepting the wrong message in the V-MA protocol. Therefore, the three reduction (sub)algorithms $\mathcal{B}_{\mathcal{C}}[1]$, $\mathcal{B}_{\mathcal{C}}[2]$, and $\mathcal{B}_{\mathcal{C}}[3]$, essentially perform the same attacks on the NM game of the commitment scheme as the corresponding three reductions given by Vaudenay for the V-MA protocol. The only difference is that our reductions put a layer of encryption on the messages sent by P_j, as is done in our protocol Enc-MCA. As in Vaudenay's reductions, the NM game needs to be extended so that the challenger, at the end of the game sends to the attacker the decommitment d corresponding to the challenge commitment c. Since this happens after the attacker sends its R, the difficulty of the NM game remains the same. However, if the $\mathcal{B}_{\mathcal{C}}$ reduction gets the decommitment d from the NM challenger, the reduction can complete the view of the protocol to $\mathcal{A}$, which makes it easier to compare the probability of $\mathcal{A}$'s success with the probability of success of $\mathcal{B}_{\mathcal{C}}$. We refer the reader to the full version [8] for the specification of these three subcases of the reduction to an NM attack. An important feature of these algorithms is that each of these sub-cases of the $\mathcal{B}_{\mathcal{C}}$ reduction at first follows the same protocol with the NM challenger, and that $\mathcal{B}_{\mathcal{C}}$ can decide which path to follow, namely whether to switch to sub-algorithm $\mathcal{B}_{\mathcal{C}}$[1,2] or $\mathcal{B}_{\mathcal{C}}[3]$, based on the first message it receives from $\mathcal{A}$. Specifically, $\mathcal{B}_{\mathcal{C}}$ switches to $\mathcal{B}_{\mathcal{C}}[3]$ if $\mathcal{A}$ first sends message $\hat{e}_j$, and otherwise $\mathcal{B}_{\mathcal{C}}$ follows $\mathcal{B}_{\mathcal{C}}$[1,2]. Similarly, in the latter case, $\mathcal{B}_{\mathcal{C}}$ switches to either $\mathcal{B}_{\mathcal{C}}[1]$ or $\mathcal{B}_{\mathcal{C}}[2]$ based on $\mathcal{A}$'s next response. Therefore these three algorithms are really just three subcases of a single reduction algorithm $\mathcal{B}_{\mathcal{C}}$. By inspection of these three subcases one can conclude that $\mathcal{B}_{\mathcal{C}}$ wins in its non-malleability attack game with probability at least $\Pr[\mathsf{Case1}]$.

Similarly algorithm $\mathcal{B}_{\mathcal{E}}$ at first follows the same algorithm and then can dispatch into $\mathcal{B}_{\mathcal{E}}[1]$ or $\mathcal{B}_{\mathcal{E}}[2]$ depending on adversary's messages. By inspection of these two subcases one concludes that $\mathcal{B}_{\mathcal{E}}$ wins in its CCA attack game with probability at least $\Pr[\mathsf{Case2}]$, which ends the proof.

Encryption-based SAS Authenticated Key Agreement Protocol. The SAS-AKA protocol Enc-AKA based on the Enc-MCA protocol is just an instance of Enc-MCA where P_i's

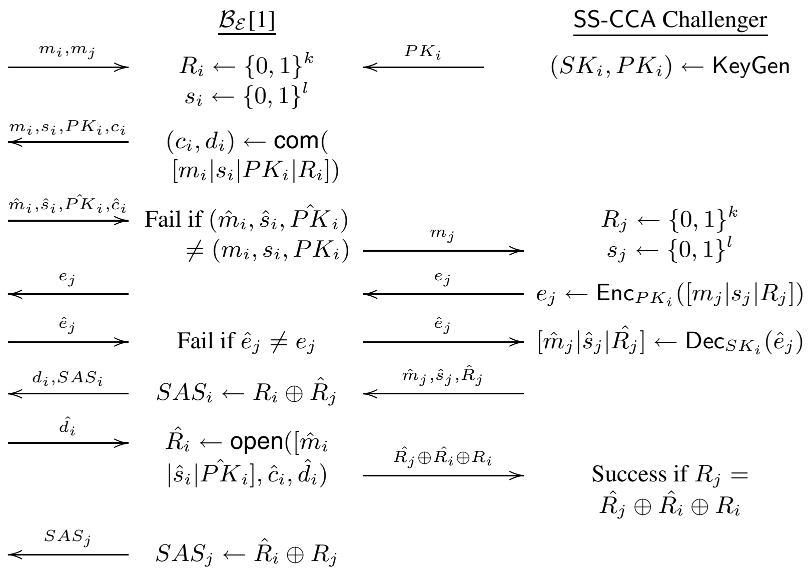
$\underline{\mathcal{B}_\mathcal{E}[1]}$
SS-CCA Challenger
m_i, m_j
$R_i \leftarrow \{0,1\}^k$
$s_i \leftarrow \{0,1\}^l$
PK_i
$(SK_i, PK_i) \leftarrow$ KeyGen
m_i, s_i, PK_i, c_i
$(c_i, d_i) \leftarrow$ com($[m_i|s_i|PK_i|R_i]$)
$\hat{m}_i, \hat{s}_i, \hat{PK}_i, \hat{c}_i$
Fail if $(\hat{m}_i, \hat{s}_i, \hat{PK}_i) \neq (m_i, s_i, PK_i)$
m_j
$R_j \leftarrow \{0,1\}^k$
$s_j \leftarrow \{0,1\}^l$
e_j
e_j
$e_j \leftarrow \mathsf{Enc}_{PK_i}([m_j|s_j|R_j])$
$\hat{e}_j$
Fail if $\hat{e}_j \neq e_j$
$\hat{e}_j$
$[\hat{m}_j|\hat{s}_j|\hat{R}_j] \leftarrow \mathsf{Dec}_{SK_i}(\hat{e}_j)$
d_i, SAS_i
$SAS_i \leftarrow R_i \oplus \hat{R}_j$
$\hat{m}_j, \hat{s}_j, \hat{R}_j$
$\hat{d}_i$
$\hat{R}_i \leftarrow$ open($[\hat{m}_i|\hat{s}_i|\hat{PK}_i], \hat{c}_i, \hat{d}_i$)
$\hat{R}_j \oplus \hat{R}_i \oplus R_i$
Success if $R_j = \hat{R}_j \oplus \hat{R}_i \oplus R_i$
SAS_j
$SAS_j \leftarrow \hat{R}_i \oplus R_j$

Fig. 4. Construction of $\mathcal{B}_\mathcal{E}[1]$ ($(m_i, s_i, PK_i) = (\hat{m}_i, \hat{s}_i, \hat{PK}_i)$, interleaving case I)

message m_i is set to null and P_j's message m_j is a fresh random key which P_j picks for each session, as shown in Figure 2. For lack of space we relegate the proof of the following theorem to the full version [8], but it is very similar to the proof of security of the Enc-MCA protocol given above.

Theorem 2 (Security of Enc-AKA). *If commitment scheme is (T_C, ϵ_C)-NM and encryption scheme is (T_E, ϵ_E)-SS-CCA, then the Enc-AKA protocol is (T, p)-secure against (n, τ_t, τ_c)-attacker for $p \geq 2n\tau_t\tau_c(2^{-k} + \max(\epsilon_C, \epsilon_E)$ and $T \leq \min(T_C, T_E)$ $-\mu$, for a small constant μ.*

References

1. Balfanz, D., Smetters, D., Stewart, P., Wong, H.C.: Talking to strangers: Authentication in ad-hoc wireless networks. In: Network and Distributed System Security Symposium (2002)
2. Bellare, M., Canetti, R., Krawczyk, H.: A modular approach to the design and analysis of authentication and key-exchange protocols. In: Symposium on Theory of Computing (2001)
3. Bellare, M., Kohno, T., Shoup, V.: Stateful public-key cryptosystems: How to encrypt with one 160-bit exponentiation. In: ACM Conference on Computer and Communications Security (2006)
4. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
5. Crescenzo, G.D., Katz, J., Ostrovsky, R., Smith, A.: Efficient and non-interactive non-malleable commitment. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 40–59. Springer, Heidelberg (2001)

6. Gehrmann, C., Mitchell, C.J., Nyberg, K.: Manual authentication for wireless devices. RSA CryptoBytes 7(1), 29–37 (Spring 2004)
7. Goodrich, M.T., Sirivianos, M., Solis, J., Tsudik, G., Uzun, E.: Loud and Clear: Human-Verifiable Authentication Based on Audio. In: International Conference on Distributed Computing Systems, ICDCS (July 2006), http://www.ics.uci.edu/ccsp/lac
8. Jarecki, S., Saxena, N.: Authenticated key agreement with key re-use in the short authenticated strings model. Available from the authors (2010)
9. Laur, S., Asokan, N., Nyberg, K.: Efficient mutual data authentication based on short authenticated strings. IACR Cryptology ePrint Archive: Report 2005/424 (November 2005), http://eprint.iacr.org/2005/424
10. Laur, S., Nyberg, K.: Efficient mutual data authentication using manually authenticated strings. In: Pointcheval, D., Mu, Y., Chen, K. (eds.) CANS 2006. LNCS, vol. 4301, pp. 90–107. Springer, Heidelberg (2006)
11. Pasini, S., Vaudenay, S.: An optimal non-interactive message authentication protocol. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 280–294. Springer, Heidelberg (2006)
12. Pasini, S., Vaudenay, S.: SAS-Based Authenticated Key Agreement. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 395–409. Springer, Heidelberg (2006)
13. Prasad, R., Saxena, N.: Efficient device pairing using human-comparable synchronized audiovisual patterns. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 328–345. Springer, Heidelberg (2008)
14. Rohs, M., Gfeller, B.: Using camera-equipped mobile phones for interacting with real-world objects. In: Ferscha, A., Hoertner, H., Kotsis, G. (eds.) Advances in Pervasive Computing, Vienna, Austria, pp. 265–271. Austrian Computer Society, OCG (April 2004)
15. Roth, V., Polak, W., Rieffel, E., Turner, T.: Simple and effective defenses against evil twin access points. In: ACM Conference on Wireless Network Security (WiSec), short paper (2008)
16. Saxena, N., Ekberg, J.-E., Kostiainen, K., Asokan, N.: Secure device pairing based on a visual channel (short paper). In: IEEE Symposium on Security and Privacy (S&P 2006) (May 2006)
17. Saxena, N., Uddin, B.: Blink 'em all: Scalable, user-friendly and secure initialization of wireless sensor nodes. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 154–173. Springer, Heidelberg (2009)
18. Soriente, C., Tsudik, G., Uzun, E.: BEDA: Button-Enabled Device Association. In: International Workshop on Security for Spontaneous Interaction, IWSSI (2007)
19. Stajano, F., Anderson, R.J.: The resurrecting duckling: Security issues for ad-hoc wireless networks. In: Malcolm, J.A., Christianson, B., Crispo, B., Roe, M. (eds.) Security Protocols 1999. LNCS, vol. 1796, pp. 172–194. Springer, Heidelberg (2000)
20. Vaudenay, S.: Secure communications over insecure channels based on short authenticated strings. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 309–326. Springer, Heidelberg (2005)

A On the General Compilation Theorem of Pasini-Vaudenay

We claim that the general composition theorem given by Pasini and Vaudenay [12] for transforming KA protocols to SAS-AKA protocols given any SAS-MCA scheme, cannot be applied in general to KA schemes which share state between sessions. The theorem of [12] constructs a SAS-AKA protocol by running any 2-round (non-authenticated) KA protocol and then inputting the two messages generated by this KA, m_i of the initiator P_i and m_j of the responder P_j, into a SAS-MCA protocol, where P_i goes first, and m_j is possibly based on m_i. Known 3-round SAS-MCA protocols allow the responder's message m_j to be picked in the second round, and thus this compilation creates a 4-round SAS-AKA from 2-round KA and 3-round SAS-MCA. Note that at the time P_j computes his response m_j, following the algorithm of the KA protocol on the received message m_i, the message m_i is not yet authenticated by P_j. If the KA protocol does not share state between sessions, having P_j compute m_j on adversarially-chosen $\hat{m}_i$ can endanger only the current session, and since the SAS-MCA subprotocol will let P_j know that $\hat{m}_i$ was not sent by P_i, P_j will reject this session.

However, if P_j keeps a shared state between sessions then the information P_j reveals in m_j, computed on *unauthenticated* message $\hat{m}_i$, could potentially reveal some secret information that endangers all other sessions of player P_j, or at least all other sessions between P_j and P_i. It's easy to create a contrived example of a Key Agreement protocol which is secure in the static adversarial model when implemented over authenticated channels but yields an insecure SAS-AKA protocol when implemented with a SAS-MCA scheme in this fashion. For example, take any Key Agreement protocol, KA, secure over authenticated links, let each player P_j keep an additional long-term secret s_j and compute a per-partner secret $k_{ij} = F_{s_j}(< P_i >)$ where F is a PRF. If the initiator's message m_i contains a special symbol $\perp$, P_j sends $m_j = k_{ij}$ to P_i in the open. Otherwise, P_j follows the KA protocol to compute its response m_j, except that it attaches to it the resulting session key encrypted with a symmetric encryption scheme under k_{ij}. In the authenticated link model, and considering a static adversary, an honest player never sends the $\perp$ symbol. If the encryption is secure, encrypting the session key does not endanger its security. Also, if F is a PRF then learning values of the F function under indices corresponding to the corrupt players does not reveal any information about the values of F on indices corresponding to the honest players. On the other hand, this protocol is an insecure SAS-AKE protocol, because an adversary can inject message $\hat{m}_i =\perp$ on the insecure channel on behalf of any player P_i, and since P_j will reply with k_i, this allows the attacker to compute the keys for *all* sessions, past and future, between P_j and P_i.

This counter-example relies on an admittedly artificial KA protocol with shared session state where interference with a single session between a pair of players trivially reveals the keys on all sessions between the same players. Still, this does show that the compilation technique of [12] can apply only to KA protocols with no shared state. Of course while this general compilation does not apply, a combination of any *particular* SAS-MCA protocol and a KA scheme with shared state can still be shown secure, and that, with some simplifications to the SAS-MCA protocol of Vaudenay [20] made in the process, is exactly what we show in this paper.