SAS-Based Group Authentication and Key Agreement Protocols

Sven Laur[1,⋆] and Sylvain Pasini[2,⋆⋆]

[1] Helsinki University of Technology, Finland
slaur@tcs.hut.fi
[2] EPFL, Lausanne, Switzerland
sylvain.pasini@epfl.ch

Abstract. New trends in consumer electronics have created a strong demand for fast, reliable and user-friendly key agreement protocols. However, many key agreement protocols are secure only against passive attacks. Therefore, message authentication is often unavoidable in order to achieve security against active adversaries. Pasini and Vaudenay were the first to propose a new compelling methodology for message authentication. Namely, their two-party protocol uses short authenticated strings (SAS) instead of pre-shared secrets or public-key infrastructure that are classical tools to achieve authenticity. In this article, we generalise this methodology for multi-party settings. We give a new group message authentication protocol that utilises only limited authenticated communication and show how to combine this protocol with classical key agreement procedures. More precisely, we describe how to transform any group key agreement protocol that is secure against passive attacks into a new protocol that is secure against active attacks.

Keywords: Groups, multi-party, message authentication, key agreement.

1 Introduction

Recently, Pasini and Vaudenay [18] analysed a peer-to-peer Voice over IP (VoIP) protocol and deduced that two users starting an (insecure) call through the Internet can build an authenticated channel thanks to their ability to recognise the voice and behaviour of the other speaker. This channel can thus be used to exchange authenticated data. In particular, exchanging Diffie-Hellman [10] public values leads to a shared secret key. As such messages are very long, they proposed to use a message cross-authentication (MCA) protocol instead of authenticating them directly. Indeed, an MCA protocol sends messages through an insecure channel and then authenticates them by using *short authenticated strings* (SAS), e.g. 20 bits. Similar protocols are used in Bluetooth and WUSB

⋆ Partially supported by Finnish Academy of Sciences and by Estonian Doctoral School in Information and Communication Technologies.
⋆⋆ Supported by the Swiss National Science Foundation, 200021-113329.

R. Cramer (Ed.): PKC 2008, LNCS 4939, pp. 197–213, 2008.

standards for authentication [14]. Different from other approaches such as certificate chains and password-based authentication, the security can be introduced as an afterthought—there is no need for a supporting infrastructure, the mere presence of limited authentic communication is sufficient.

The main aim of this article is to extend the SAS-based methodology previously outlined in [21] from a two-party setting to a group setting. Namely, manual authentication can be used to secure group key agreement protocols, i.e., group members can establish a shared secret over an insecure network. Afterwards, the group can use standard cryptographic methods to establish secure communication. The corresponding group formation protocol significantly simplifies common key establishment and works even if the participants of the group are not known ahead. Although the group structure is often predetermined, e.g. participants of the conference calls know to whom they want to talk, ad hoc group formation is quite common, too. The most obvious example is automatic device detection in wireless networks. In particular, a user may form a secure piconet from all accessible Bluetooth devices. Ad hoc formation of secure WLAN groups is another natural example both in the military and civil context.

In principle, two party protocols are sufficient to establish message authentication for groups. On the other hand, such an approach requires a lot of user-interaction that diminishes usability of the corresponding solutions in practical applications. It is clearly more convenient to join 10 guest computers into a WLAN network together, than repeat the same procedure over and over again. Motivated by this concern, we propose a new SAS-based group authentication protocol that significantly minimises the required user interaction, see Section 3. Essentially, the amount of user interaction for the pairwise and group authentication coincides—user has to remember only single test value. The latter is significantly more convenient than operating with 10 different test values that are needed when we iterate pairwise authentication protocol.

The security of our SAS-based protocol is based on the non-malleability of a commitment scheme. Each user chooses a secret key, then commits to it while revealing the input message to be authenticated. When all participants have committed, then the secrets are opened. Next, each party uses an almost universal hash function to compute a test value from the received messages and secrets and then compares it with the others using authenticated communication. Thus, an adversary that wants to modify input messages has to find a "collision" on the hash function or break the commitment scheme. The corresponding security proof itself is straightforward but technical due to the complicated nature of non-malleability. All definitions that are needed for the formal proof are given in Sections 2 and 3 and the proof itself is presented in Section 4.

Section 6 provides a solution to the group key agreement problem. Shortly put, we can achieve immunity against active attacks if we first run a standard group key agreement protocol over the insecure channel and then authenticate the corresponding protocol transcript. Moreover, if we additionally authenticate some long term public keys, then we can form separate subgroups without relying on authenticated communication. In other words, there is no need for

additional user interaction when we decide to expel some group members. Such an "authenticate once" philosophy is particularly useful in the context of wireless home networks, as it provides a simple and provably secure method for hosting guest computers in the network for limited time.

2 Cryptographic Preliminaries

All of our results are stated in the framework of exact security, i.e., our main goal is to construct protocols that are secure against all t-time adversaries. In particular, all security properties are formally specified by a game or a game pair between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. For a single game $\mathcal{G}$, the advantage is defined by $\mathrm{Adv}(\mathcal{A}) = \Pr[\mathcal{G}^{\mathcal{A}} = 1]$. For a game pair $\mathcal{G}_0, \mathcal{G}_1$, the advantage is defined $\mathrm{Adv}(\mathcal{A}) = |\Pr[\mathcal{G}_0^{\mathcal{A}} = 1] - \Pr[\mathcal{G}_1^{\mathcal{A}} = 1]|$. Typically, one requires that for all t-time adversaries $\mathcal{A}$ the advantage $\mathrm{Adv}(\mathcal{A})$ is upper bounded by ε. Of course, all results can be translated back to the non-uniform polynomial security model by considering asymptotics.

Keyed Hash Functions. A keyed hash function $h : \mathcal{M} \times \mathcal{R} \to \mathcal{T}$ takes two arguments: a message $m \in \mathcal{M}$ and a key $r \in \mathcal{R}$, and outputs a digest $t \in \mathcal{T}$. A hash function h is ε_{u}-*almost universal*, if for any two inputs $x_0 \neq x_1$,

$$\Pr[r \in_{\mathrm{u}} \mathcal{R} : h(x_0, r) = h(x_1, r)] \leq \varepsilon_{\mathrm{u}} \ .$$

The notion can be extended to handle n sub-keys of the same domain, i.e., $h : \mathcal{M} \times \mathcal{R}^n \to \mathcal{T}$. A hash function h is ε_{u}-*almost universal w.r.t. the sub-key pairs*, if for any two inputs $x_0 \neq x_1$, indices i, j and $r_1, \ldots, r_n, \hat{r}_1, \ldots, \hat{r}_n \in \mathcal{R}$:

$$\Pr[r_* \in_{\mathrm{u}} \mathcal{R} : h(x_0, \boldsymbol{r}) = h(x_1, \hat{\boldsymbol{r}})] \leq \varepsilon_{\mathrm{u}} \ ,$$

where $\boldsymbol{r} = (r_1, \ldots, r_{i-1}, r_*, r_{i+1}, \ldots, r_n)$, $\hat{\boldsymbol{r}} = (\hat{r}_1, \ldots, \hat{r}_{j-1}, r_*, \hat{r}_{j+1}, \ldots, \hat{r}_n)$ and $i = j$ is allowed. That is, output values are likely to be different if the corresponding hash functions share at least one correctly formed sub-key $r_* \in_{\mathrm{u}} \mathcal{R}$. A function h is ε_{r}-*almost regular w.r.t. to the sub-key* r_i, if for any $x, \hat{r}_1, \ldots, \hat{r}_n, y$:

$$\Pr[r_i \in_{\mathrm{u}} \mathcal{R}_i : h(x_1, \hat{r}_1, \ldots, \hat{r}_{i-1}, r_i, \hat{r}_{i+1}, \ldots, \hat{r}_n) = y] \leq \varepsilon_{\mathrm{r}} \ .$$

We need a hash function that is ε_{u}-almost universal and ε_{r}-almost regular and could handle variable number of sub-keys at the same time. A priori it is not clear that such hash functions exist. Therefore, we give one possible explicit construction. Let all sub-keys be from $\{0,1\}^{2s}$ and messages from $\{0,1\}^s$ for a certain integer s which bound the message space. To hash a message x, we first compute an intermediate key $a \leftarrow r_1 \oplus \cdots \oplus r_n$; split a into two halves a_1, a_2; interpret x, a_1, a_2 as elements of the Galois field $\mathsf{GF}(2^s)$ and define $h(x, r_1, \ldots, r_n) = a_1 x + a_2$ over $\mathsf{GF}(2^s)$. If $x_0 \neq x_1$ then it is straightforward to verify that a pair $h(x_0, \boldsymbol{r}), h(x_1, \hat{\boldsymbol{r}})$ is uniformly distributed over $\{0,1\}^{2s}$ in the universality experiment. To get shorter hash values, we can output ℓ lowest bits. Then the hash function has optimal bounds $\varepsilon_{\mathrm{r}} = \varepsilon_{\mathrm{u}} = 2^{-\ell}$.

Common Reference String Model. In the *common reference string* (CRS) model, a trusted third party generates system wide initial parameters pk and automatically transfers them to all participants. Most of the communication and computation efficient commitment schemes are specified for the CRS model.

Although such a model seems quite restrictive at first glance, all communication standards provide system-wide public parameters such as specifications of hash functions or a bit length of public keys. In other words, the CRS model is not problem in practise. Nevertheless, one should make a trade-off between computational efficiency and reusability and the size of system-wide public parameters pk. Also, there are theoretic constructions that allow generation of a common reference string in the standard model.

Commitment Schemes. A *commitment scheme* $\mathcal{C}om$ is specified by a triple $(\mathsf{setup}, \mathsf{commit}, \mathsf{open})$. The setup algorithm setup generates public parameters pk for the commitment scheme. The randomised commitment algorithm $\mathsf{commit}_{\mathsf{pk}} : \mathcal{M} \to \mathcal{C} \times \mathcal{D}$ maps messages $m \in \mathcal{M}$ into a commitment string $c \in \mathcal{C}$ of fixed length and a decommitment value $d \in \mathcal{D}$. Usually the decommitment value is a pair $d = (m, r)$, where r is the randomness used to compute c. A commitment scheme is functional if for all $(c, d) \leftarrow \mathsf{commit}_{\mathsf{pk}}(m)$ the equality $\mathsf{open}_{\mathsf{pk}}(c, d) = m$ holds. Incorrect decommitment values should yield a special abort value $\perp$.

Proofs usually rely on three cryptographic properties of commitment schemes: hiding, binding and non-malleability. Non-malleability is the strongest property, as binding and hiding properties directly follow from non-malleability and not vice versa. Many notions of non-malleable commitments have been proposed in cryptographic literature [11,9,12,7,14]. All these definitions try to capture requirements that are necessary to defeat man-in-the-middle attacks. We adopt the modernised version of non-malleability w.r.t. opening. The corresponding definition [14] mimics the framework of non-malleable encryption [5] and leads to more natural security proofs compared to the simulation based definitions [9,7].

Non-malleability and security against chosen ciphertext attacks (CCA) are known to be tightly coupled. In fact, these notions coincide if the adversary is allowed to make decryption queries throughout the entire attack [1] and thus usage of decryption oracles can simplify many proofs without significantly increasing the security requirements. Unfortunately, a similar technique is not applicable to commitment schemes as there can be several different valid decommitment values d_i for a single commitment c. Thus, we must use explicit definitions of binding and non-malleability properties in our proofs. A commitment scheme $\mathcal{C}om$ is $(t, \varepsilon_{\mathrm{b}})$-*binding* if for any t-time adversary $\mathcal{A}$:

$$\mathrm{Adv}^{\mathsf{bind}}_{\mathcal{C}om}(\mathcal{A}) = \Pr\left[\begin{array}{l}\mathsf{pk} \leftarrow \mathsf{setup}, (c, d_0, d_1) \leftarrow \mathcal{A}(\mathsf{pk}) : \\ \perp \neq \mathsf{open}_{\mathsf{pk}}(c, d_0) \neq \mathsf{open}_{\mathsf{pk}}(c, d_1) \neq \perp\end{array}\right] \leq \varepsilon_{\mathrm{b}} \ ,$$

The non-malleability property is defined by complicated games, and thus we use an illustrative pictorial style to specify security games, see Fig. 1. Intuitively, the goal is: given a valid commitment c, it is infeasible to generate related commitments $\hat{c}_1, \ldots, \hat{c}_n$ that can be successfully opened after seeing a decommitment

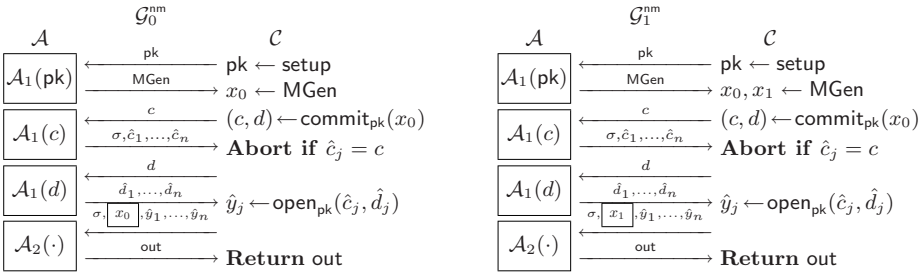

Fig. 1. Non-malleability games $\mathcal{G}_0^{\mathsf{nm}}$ and $\mathcal{G}_1^{\mathsf{nm}}$

value d. More formally, the adversary $\mathcal{A}$ consists of two parts: $\mathcal{A}_1$ corresponds to the *active* part of the adversary that tries to create and afterwards open commitments related to c while $\mathcal{A}_2$ captures a desired *target relation*. Note that $\mathcal{A}_1$ is a stateful algorithm and can pass information from one stage to the other but no information can be passed from $\mathcal{A}_1$ to $\mathcal{A}_2$ except σ. By convention, a game is ended with the output $\perp$ if any operation leads to $\perp$.

Fig. 1 should be read as follows. In $\mathcal{G}_0^{\mathsf{nm}}$, a challenger $\mathcal{C}$ first generates the public parameters pk. Given pk, the adversary outputs a message generator MGen. Next, the challenger $\mathcal{C}$ selects $x_0 \leftarrow \mathsf{MGen}$ and computes (c, d). Given c, the adversary outputs some commitment values $\hat{c}_i$ and an advice σ for $\mathcal{A}_2$ and then, given d he generates some decommitment values $\hat{d}_i$. Finally, $\mathcal{C}$ opens all commitments $\hat{y}_i \leftarrow \mathsf{open}_{\mathsf{pk}}(\hat{c}_i, \hat{d}_i)$ and tests whether $\mathcal{A}_1$ won or not by computing $\mathcal{A}_2(\sigma, x_0, \hat{y}_1, \ldots, \hat{y}_n)$. The condition $\hat{c}_j \neq c$ eliminates trivial attacks. The game $\mathcal{G}_1^{\mathsf{nm}}$ is almost the same, except the challenger tests a relation $\mathcal{A}_2(\sigma, x_1, \hat{y}_1, \ldots, \hat{y}_n)$ instead, where $x_1 \leftarrow \mathsf{MGen}$ is chosen independently from the rest of the game. A commitment scheme is $(t, \varepsilon_{\mathrm{nm}})$-*non-malleable* w.r.t. to opening if for any adversary $\mathcal{A}$ such that the working times of $\mathcal{G}_0^{\mathsf{nm}}$ and $\mathcal{G}_1^{\mathsf{nm}}$ are less than t, the advantage

$$\mathrm{Adv}_{\mathcal{C}om}^{\mathsf{nm}}(\mathcal{A}) = \left|\Pr\left[\mathcal{G}_0^{\mathsf{nm}} = 1\right] - \Pr\left[\mathcal{G}_1^{\mathsf{nm}} = 1\right]\right| \leq \varepsilon_{\mathrm{nm}} \ .$$

Note that $\mathcal{A}_2$ can be any computable relation that is completely fixed after seeing c. For instance, we can define $\mathcal{A}_2(\sigma, x, y) = [x = y]$. Hence, it must be infeasible to construct a commitment $\hat{c}$ that can be opened later to the same value as c.

Non-malleable commitments schemes can be easily constructed based on simulation-sound trapdoor commitments from Mac-Kenzie and Yang [16] as detailed by Vaudenay [21]. They can also be built using a CCA2 secure encryption scheme, or by using a hash function as detailed by Laur and Nyberg [14].

3 Manual Group Message Authentication

Although our final goal is to establish a secure group key agreement protocol, we start from the group message authentication. Since active attacks can be detected by group authentication, any cryptographic key agreement protocol

secure against passive attacks can be made fully-secure, see Theorem 1 in [18] and Section 6.

Communication Model. As usual, the communication is asynchronous. Participants can send two types of messages. Insecure in-band communication is routed via an active adversary $\mathcal{A}$ who can drop, delay, modify and insert messages. But participants can also send *short authenticated strings* (SAS) aka *out-of-band messages*. Out-of-band communication is authentic: the adversary can only read and possibly delay SAS messages.

Note that there are no *true* broadcast channels in our model. Although several networks such as WLAN in ad hoc mode offer physical broadcast channels, there are no guarantees that the signal actually reaches all nodes. If we can guarantee this by physical means, then the authentication task becomes trivial. Otherwise, different recipients can receive different broadcast messages and there is no difference between broadcasting and standard messaging except for efficiency. Similarly, broadcasting authenticated messages does not change the security analysis, although in practise, broadcasting can significantly reduce the necessary human interaction and make the protocol more user-friendly. For instance, considering the Bluetooth pairing, a human entering the same PIN on each mobile device is considered a broadcast primitive. Considering a VoIP-based conference, when participants are talking together, they use an (insecure) authenticated channel that broadcasts messages. The authentication comes from the ability of other users to recognise the speaker, e.g. by its voice and behaviour.

It is hard to formalise desired security properties for group authentication, as there are many different attack scenarios and security goals. Hence, we first consider a simple stand-alone security model and then gradually extend our definitions to cover more complex settings including key agreement protocols.

Idealised Functionality. Consider a network $\mathcal{P}_1, \ldots, \mathcal{P}_N$ of N nodes. A node name is a label $\mathsf{id} \in \{1, \ldots, N\}$ that uniquely determines the corresponding node $\mathcal{P}_{\mathsf{id}}$. In principle, node names can be non-consecutive such as hardware addresses, i.e., $\{1, \ldots, N\}$ is only a set of potential group members. A *group message authentication* (GMA) protocol for an n-element subgroup $\mathcal{G} = \{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$ works as follows: each participant $\mathcal{P}_{\mathsf{id}}$, $\mathsf{id} \in \mathcal{G}$ starts with inputs m_{id} and ends with outputs $\mathcal{G}$ and $\boldsymbol{m}$, where $\boldsymbol{m} = (m_{\mathsf{id}_1}, \ldots, m_{\mathsf{id}_n})$ is ordered w.r.t. the sender identities $\mathsf{id}_1 < \mathsf{id}_2 < \cdots < \mathsf{id}_n$. In other words, given $\mathcal{G}$ and $\boldsymbol{m}$ it is trivial to restore who participated in the protocol and what was its input.

Stand-Alone Security. There are several important aspects to note. First, a group may be dynamically formed based on the participation in a GMA protocol, for example fast setup of ad hoc military networks. But then an adversary can always split the group into several subgroups and block the traffic between the subgroups. As a result, each subgroup agrees on a different output. Such attacks cannot be defeated unless parties know the description of $\mathcal{G}$ in advance, i.e., there is some authenticated way to broadcast $\mathcal{G}$. Second, an adversary may set up several dummy network nodes in order to corrupt communication or secretly shuffle different groups. Thus, we consider a scenario where a subset $\mathcal{G}$ of all

network nodes wants to establish a common message $\boldsymbol{m}$. At the end of the protocol, either all participants halt or each $\mathcal{P}_{\mathsf{id}}$, $\mathsf{id} \in \mathcal{G}$ obtains values $\hat{\mathcal{G}}_{\mathsf{id}}$ and $\hat{\boldsymbol{m}}_{\mathsf{id}}$. We allow adaptive malicious corruption[1] of group participants, i.e., at any time during the protocol execution $\mathcal{A}$ can take total control over any node $\mathcal{P}_{\mathsf{id}}$.

Let $\mathcal{H} \subseteq \mathcal{G}$ be the set of uncorrupted participants at the end of the protocol. Then the adversary $\mathcal{A}$ *succeeds in deception* if at least two uncorrupted group members $\alpha, \beta \in \mathcal{H}$ have different outputs $(\hat{\mathcal{G}}_\alpha, \hat{\boldsymbol{m}}_\alpha) \neq (\hat{\mathcal{G}}_\beta, \hat{\boldsymbol{m}}_\beta)$ and the group was not trivially split, i.e., $\mathcal{G} \subseteq \hat{\mathcal{G}}_\gamma$ for some $\gamma \in \mathcal{H}$. In other words, at least one honest participant gets messages from all members of $\mathcal{G}$. Formally, it is impossible to assure $\mathcal{G} = \hat{\mathcal{G}}_\gamma$, as an honest party cannot distinguish whether a node freely joined or was forced to join by $\mathcal{A}$. If the question of free will is irrelevant, then we can postulate that after the successful execution honest participants obtain $\mathcal{G}$. This is the maximum achievable security level, as honest members cannot detect corruption and missing messages caused by the splitting of the network. An alternative is to state correctness for each subgroup separately, but then protocol instances are run in parallel and this is covered by Section 5.

Since commitment schemes are often defined only for common reference string model, we give the security definition in the CRS model. To assure reusability of public parameters, we must consider chosen input attacks. More precisely, an adversary $\mathcal{A}$ can choose the group members $\mathcal{G}$ and their contributed messages m_{id} depending on the shared authentic common reference string $\mathsf{pk} \leftarrow \mathsf{setup}$. The advantage of $\mathcal{A}$ against a protocol instance π is defined as

$$\mathrm{Adv}_\pi^{\mathsf{forge}}(\mathcal{A}) = \Pr\left[\mathsf{pk} \leftarrow \mathsf{setup}, (\boldsymbol{m}, \mathcal{G}) \leftarrow \mathcal{A}(\mathsf{pk}) : \mathcal{A} \text{ succeeds in deception}\right] \ .$$

A protocol instance π is (t, ε)-secure in the stand-alone model if for any t-time adversary $\mathcal{A}$, the corresponding advantage is bounded $\mathrm{Adv}_\pi^{\mathsf{forge}}(\mathcal{A}) \leq \varepsilon$.

Note that stand-alone security model covers only the case where no other protocols are executed together with π. In particular, it is not clear whether parallel execution of several different instances of π remains secure. We will return to this issue in Section 5 and show that parallel composition remains secure if some natural assumptions are satisfied. Still, for many cases where GMA is used once, the stand-alone security is sufficient. For example, many ad hoc groups use GMA to share a common secret to establish secure channels.

4 A SAS-Based Group Message Authentication Protocol

Our new group message authentication protocol SAS-GMA (See Fig. 2) borrows ideas from Vaudenay's cross-authentication protocol SAS-MCA [21, App. A] and Mana IV [14,19]. Both aforementioned protocols use commitments to temporarily hide certain keys. Similarly to SAS-MCA, all sub-keys are released after the adversary has delivered all messages. And similarly to Mana IV, messages m_i are sent in the clear and authenticated test values are ℓ-bit hash codes.

[1] In many cases, adaptive corruption is impossible, but with our new protocol being secure against adaptive corruption, it makes no sense to consider weaker models.

As the SAS-GMA protocol is symmetric, Fig. 2 only specifies the behaviour of a single party $\mathcal{P}_i$ who wants to participate in the protocol. Here $\hat{\mathcal{G}}_i$ denotes the group of participants who joined $\mathcal{P}_i$ during the first round before the timeout. Of course, if the group $\hat{\mathcal{G}}_i$ is known beforehand then $\mathcal{P}_i$ can wait until all other group members have sent their first messages. For clarity, variables $\hat{m}_{ji}, \hat{c}_{ji}, \hat{d}_{ji}$ denote the values from $\mathcal{P}_j$ that are received by $\mathcal{P}_i$. The hats indicate a possible modification by an adversary. The output vector $\hat{\boldsymbol{m}} = (\hat{m}_{ji})$ and the sub-key vector $\hat{\boldsymbol{r}}_i = (\hat{r}_{ji})$ are ordered w.r.t. sender identities, see Section 3. To be exact, $\hat{m}_{ii} = m_i$, $\hat{r}_{ii} = r_i$ and j ranges over $\hat{\mathcal{G}}_i$. Also note that (i, r_i) and $(\hat{\mathcal{G}}_i, \hat{\boldsymbol{m}}_i)$ are shorthands for binary strings that uniquely encode the corresponding elements.

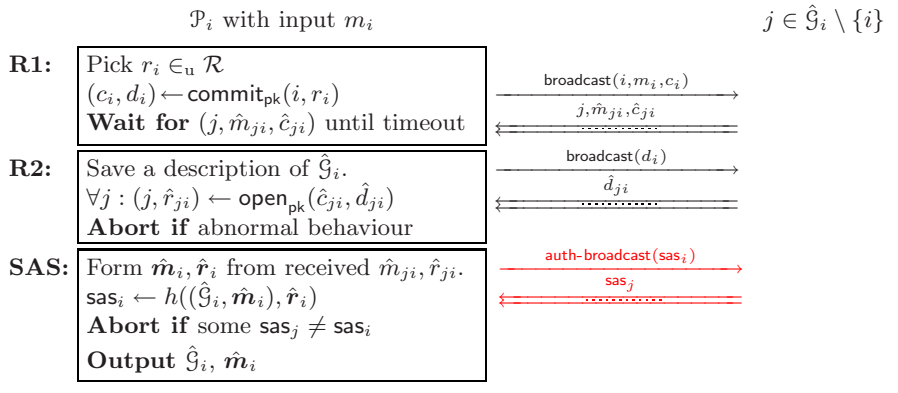

Fig. 2. The proposed SAS-GMA Protocol

Implementation Details. The cryptographic requirements for the hash function h and the commitment scheme $\mathcal{C}om$ are formally specified by Theorem 1, but there are many other minor details that are not covered by Fig. 2.

Assume that the final output $(\hat{\mathcal{G}}_i, \hat{\boldsymbol{m}}_i)$ can be always encoded as s-bit string. Then the hash function $h : \{0,1\}^s \times \mathcal{R}^* \to \mathcal{T}$ must support variable number of sub-keys r_j, since the size of the group can vary. For example, we can use a single keyed hash function h_1 and some sort of secure combiner to derive a new master key from sub-keys, as described in Section 2. The restriction $(\hat{\mathcal{G}}_i, \hat{\boldsymbol{m}}_i) \in \{0,1\}^s$ is not limiting in practise, as we can use collision resistant hash functions like SHA-256 to compress an encoding of any length to 256-bit string.

Secondly, we assume that the description of h and the public parameters of $\mathcal{C}om$ are fixed and distributed by a trusted authority. Thirdly, we assume that a participant $\mathcal{P}_i$ halts if there is any hint of an attack: (a) some group member halts; (b) there are duplicates $(j, \hat{m}_{ji}, \hat{c}_{ji}) \neq (j, \hat{m}'_{ji}, \hat{c}'_{ji})$; (c) a sub-key is in invalid form $(j, \star) \neq \mathsf{open}_{\mathsf{pk}}(\hat{c}_{ji}, \hat{d}_{ji})$; (d) some SAS messages do not match.

Another important aspect is secure comparison of SAS messages. In principle, it is sufficient to deliver minimal amount of messages so that participants can detect $\mathsf{sas}_\alpha \neq \mathsf{sas}_\beta$ for $\alpha, \beta \in \mathcal{G}$, where $\mathcal{G}$ is the set of all active participants of the protocol. If it is possible to detect all these active nodes, then a single

node can broadcast the SAS message so that the remaining nodes can compare to their SAS messages. For many applications such as securing conference calls over VoIP, forming Bluetooth piconets and other wireless device networks, the group is known in advance and thus broadcast of a single SAS message is a viable option. Also note that the group formation can be combined with node detection in the Bluetooth networks and thus the timeout effect is marginal.

Stand-Alone Security. The security proof for SAS-GMA is straightforward but quite technical. Hence, we present the proof of Theorem 1 in smaller chunks to make it more comprehensible. Note that the security level depends linearly on $|\mathcal{G}|$ but the constant term $\max\{\varepsilon_{\mathrm{u}}, \varepsilon_{\mathrm{r}}\} \approx 1/|\mathcal{T}| \approx 2^{-\ell}$ dominates over the term $n \cdot \varepsilon_{\mathrm{nm}} + \varepsilon_{\mathrm{b}}$. Therefore, the deception probability asymptotically approaches the theoretical lower bound $2^{-\ell}$.

Theorem 1. *Let n be the maximal size of the group $\mathcal{G}$ and h be ε_{u}-almost universal w.r.t. each sub-key pair and ε_{r}-almost regular w.r.t. each sub-key. Then for any t there exists $\tau = t + \mathcal{O}(1)$ such that if the commitment scheme is $(\tau, \varepsilon_{\mathrm{b}})$-binding and $(\tau, \varepsilon_{\mathrm{nm}})$-non-malleable, then the SAS-GMA protocol is $(t, n \cdot \varepsilon_{\mathrm{nm}} + \varepsilon_{\mathrm{b}} + \max\{\varepsilon_{\mathrm{u}}, \varepsilon_{\mathrm{r}}\})$-secure in the stand-alone model.*

Proof. For a sake of contradiction, assume that t-time adversary $\mathcal{B}$ violates the bound on the deception probability. Then we transform $\mathcal{B}$ to an adversary against the non-malleability games $\mathcal{G}_0^{\mathsf{nm}}$, $\mathcal{G}_1^{\mathsf{nm}}$ depicted in Fig. 1. The exact reduction is depicted on Fig. 3 and explained further in Lemma 1 and 2. Here, we just note that $\mathcal{A}_1$ simulates an instance π of the SAS-GMA protocol for $\mathcal{B}$ so that $\mathcal{A}_2$ can compute the predicate '$\mathcal{B}$ succeeds in deception' in the non-malleability game.

More precisely, $\mathcal{A}_1$ replaces the commitment c_k of $\mathcal{P}_k$ by the challenge commitment $c \leftarrow \mathsf{commit}_{\mathsf{pk}}(k, r)$ for $r \in_{\mathrm{u}} \mathcal{R}$. As $\mathcal{A}_1$ can pass information to $\mathcal{A}_2$ only via the commitment vector $\hat{\boldsymbol{c}}$ and the advice σ, then the predicate '$\mathcal{B}$ succeeds in deception' must be computable from σ, $\hat{\boldsymbol{c}}$ and corresponding decommitment vector $\hat{\boldsymbol{d}}$. The latter is possible only if $\mathcal{P}_k$ is the last honest party to release his decommitment value d_k, see Lemma 2. Thus, $\mathcal{A}_1$ must choose k randomly from the group $\mathcal{G}$ provided by $\mathcal{B}$ after seeing pk. Lemma 1–3 establish

$$\begin{aligned}\mathrm{Adv}^{\mathsf{nm}}(\mathcal{A}) &= \Pr[\mathcal{A}_1 \neq \perp] \cdot |\Pr[\mathcal{G}_0^{\mathsf{nm}} = 1 | \mathcal{A}_1 \neq \perp] - \Pr[\mathcal{G}_1^{\mathsf{nm}} = 1 | \mathcal{A}_1 \neq \perp]| \\ &\geq \frac{1}{n}(\mathrm{Adv}^{\mathsf{forge}}(\mathcal{B}) - \varepsilon_{\mathrm{b}}) - \frac{1}{n} \cdot \max\{\varepsilon_{\mathrm{u}}, \varepsilon_{\mathrm{r}}\} > \varepsilon_{\mathrm{nm}} \ .\end{aligned}$$

As the working time of $(\mathcal{A}_1, \mathcal{A}_2)$ is $\tau = t + 2t_\pi + \mathcal{O}(n) = t + \mathcal{O}(1)$ where t_π is the working time of the honest parties, we have reached a desired contradiction. □

Lemma 1. *The sub-adversary $\mathcal{A}_1$ described below satisfies $\Pr[\mathcal{A}_1 \neq \perp] \geq \frac{1}{n} \cdot (\mathrm{Adv}^{\mathsf{forge}}(\mathcal{A}) - \varepsilon_{\mathrm{b}})$ and the challenger $\mathcal{C}$ never halts unless $\mathcal{A}_1 = \perp$.*

Proof. The sub-adversary $\mathcal{A}_1$ sketched by Fig. 3 first forwards pk to $\mathcal{B}$ that replies $\mathcal{G}$ and $\boldsymbol{m}$. Hence, $\mathcal{A}_1(\mathsf{pk})$ can choose $k \in_{\mathrm{u}} \mathcal{G}$ and return a description of the uniform distribution over $\{k\} \times \mathcal{R}$ as MGen. Given $c \leftarrow \mathsf{commit}_{\mathsf{pk}}(k, r_k)$,

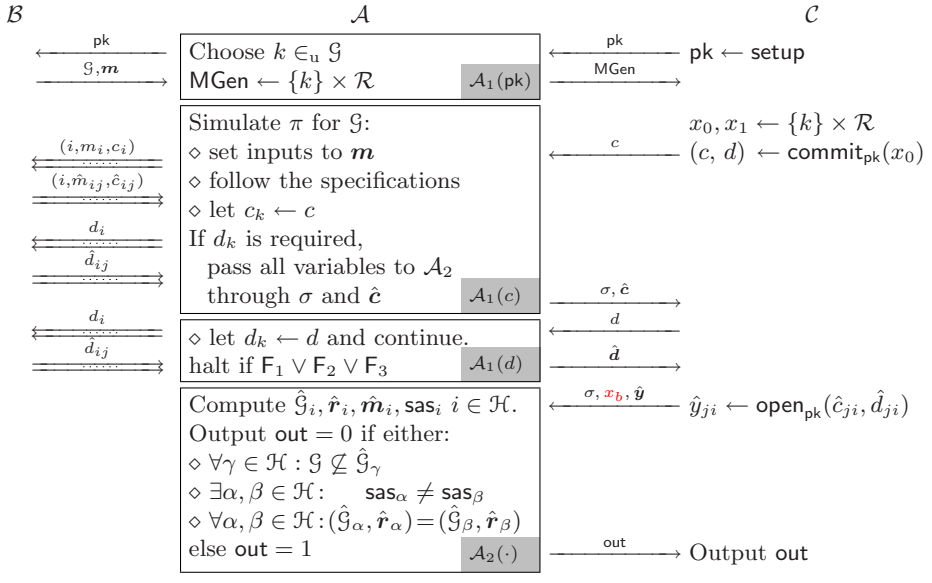

Fig. 3. Reduction to the NM game $\mathcal{G}_b^{\mathsf{nm}}$ for $b \in \{0, 1\}$

the sub-adversary $\mathcal{A}_1$ can continue simulation of π so that $c_k \leftarrow c$ and collect all messages received by all nodes in $\mathcal{G}$. To be precise, the simulation follows the specification of SAS-GMA except for computing c_k, d_k. In particular, if $\mathcal{B}$ corrupts $\mathcal{P}_i$, then $\mathcal{A}_1$ gives the control over $\mathcal{P}_i$ to $\mathcal{B}$ as in the real execution of π (If $\mathcal{P}_k$ is corrupted then d_k must be released). The simulation continues until $\mathcal{P}_k$ must release d_k. To proceed, $\mathcal{A}_1$ passes all variables that are needed to compute the predicate '$\mathcal{B}$ succeeds in deception' to $\mathcal{C}$:

1. Compute sets $\mathcal{I} = \{(j,i) : \hat{c}_{ji} \neq c\}$ and $\mathcal{J} = \{(j,i) : \hat{c}_{ji} = c\}$.
2. Send sets $\mathcal{I}, \mathcal{J}, \mathcal{G}$, all observed $\hat{m}_{ji}$, and current value of $\mathcal{H}$ as σ to $\mathcal{C}$.
3. Send all plausible commitments $\hat{\boldsymbol{c}} = (\hat{c}_{ji})$ for $(j,i) \in \mathcal{I}$ to $\mathcal{C}$.

Then the challenger $\mathcal{C}$ releases d, and $\mathcal{A}_1$ continues the simulation of π with $d_k \leftarrow d$ until the end and halts if one of the following conditions is satisfied:

F_1: The adversary $\mathcal{B}$ fails in deception.
F_2: A double opening is revealed: $\mathsf{open}_{\mathsf{pk}}(c, d) \neq \mathsf{open}_{\mathsf{pk}}(c, \hat{d}_{ji}) \neq \perp$.
F_3: The node $\mathcal{P}_k$ is not the last honest node to reveal the decommitment.

By this construction, $\Pr[\neg\mathsf{F}_1] = \mathrm{Adv}_\pi^{\mathsf{forge}}(\mathcal{B})$ and $\Pr[\mathsf{F}_2] \leq \varepsilon_{\mathrm{b}}$ or otherwise $\mathcal{A}_1$ can be used to defeat the binding property of the commitment scheme. Note that the simulation is perfect and thus $\mathcal{P}_k$ is the last honest node that releases d_k with probability[2] $\frac{1}{|\mathcal{G}|}$. The latter is true even if $\neg\mathsf{F}_1$ and $\neg\mathsf{F}_2$ and we obtain

$$\Pr[\mathcal{A}_1 \neq \perp] = \Pr[\neg\mathsf{F}_3 | \neg\mathsf{F}_1 \wedge \neg\mathsf{F}_2] \cdot \Pr[\neg\mathsf{F}_1 \wedge \neg\mathsf{F}_2] \geq \frac{1}{n} \cdot (\mathrm{Adv}_\pi^{\mathsf{forge}}(\mathcal{B}) - \varepsilon_{\mathrm{b}}) \ .$$

[2] Note that $\mathcal{B}$ cannot succeed if it corrupts all nodes and thus w.l.o.g. that $\mathcal{H} \neq \emptyset$.

Finally, note that $\mathcal{C}$ halts only if some $\hat{d}_{ji}$ is an invalid decommitment value but then $\mathcal{B}$ fails also in the simulation of π and $\mathcal{A}_1 = \perp$. □

Lemma 2. *If $\mathcal{A}_1 \neq \perp$ in the game $\mathcal{G}_0^{\mathsf{nm}}$, then $\mathcal{A}_2$ described below correctly recovers the end state of the simulation and thus* $\Pr[\mathcal{G}_0^{\mathsf{nm}} = 1 | \mathcal{A}_1 \neq \perp] = 1$.

Proof. Assuming that $\mathcal{A}_1 \neq \perp$, then the simulation conducted by $\mathcal{A}_1$ ended with a successful deception. As $\mathcal{P}_k$ was indeed the last honest node to release d_k, then $\hat{m}_{ji}, \hat{r}_{ji}$ for indices $(j,i) \in \mathcal{I} \cup \mathcal{J}$ are sufficient to recover all SAS messages computed by $\mathcal{H}$. By the construction, $(j, \hat{r}_j) = \mathsf{open}_{\mathsf{pk}}(\hat{c}_{ji}, \hat{d}_{ji}) = \hat{y}_{ji}$ for $(j,i) \in \mathcal{I}$ and $\mathsf{open}_{\mathsf{pk}}(\hat{c}_{ji}, \hat{d}_{ji}) = \mathsf{open}_{\mathsf{pk}}(c,d) = x_0$ for $(j,i) \in \mathcal{J}$ since F_2 cannot happen. As a result, $\mathcal{A}_2$ can compute all $\hat{\boldsymbol{m}}_i$ and $\hat{\boldsymbol{r}}_i$ for $i \in \mathcal{H}$ by setting $(k, r_{kk}) \leftarrow x_b$ and replacing $\mathsf{open}_{\mathsf{pk}}(\hat{c}_{ji}, \hat{d}_{ji})$ calls with appropriate values specified above. Then it remains to restore $\mathsf{sas}_i \leftarrow h((\hat{\mathcal{G}}_i, \hat{\boldsymbol{m}}_i), \hat{\boldsymbol{r}}_i)$ for $i \in \mathcal{H}$ and test $\mathsf{sas}_\alpha = \mathsf{sas}_\beta$ for $\alpha, \beta \in \mathcal{H}$ and output 1 in case of deception. Recall that deception happens only if the test values sas_α match but some $(\hat{\mathcal{G}}_\alpha, \hat{\boldsymbol{m}}_\alpha) \neq (\hat{\mathcal{G}}_\beta, \hat{\boldsymbol{m}}_\beta)$ and $\mathcal{G} \subseteq \hat{\mathcal{G}}_\alpha$.

As $\mathcal{A}_2$ computes the predicate '$\mathcal{B}$ succeeds in deception' and since $\mathcal{A}_1 \neq \perp$ implies $\neg\mathsf{F}_1$, we have $\Pr[\mathcal{G}_0^{\mathsf{nm}} = 1 | \mathcal{A}_1 \neq \perp] = 1$. □

Lemma 3. *Let $\mathcal{A}_2$ be as described in Lemma 2. Then we can bound the conditional probability* $\Pr[\mathcal{G}_1^{\mathsf{nm}} = 1 | \mathcal{A}_1 \neq \perp] \leq \max\{\varepsilon_{\mathrm{u}}, \varepsilon_{\mathrm{r}}\}$.

Proof. Assuming that $\mathcal{A}_1 \neq \perp$, then the simulation conducted by $\mathcal{A}_1$ ended with a successful deception. Consequently, $c = \mathsf{commit}_{\mathsf{pk}}(k, r_k)$ could have been broadcast only as $\hat{c}_{ki}$, otherwise $\mathcal{B}$ would have failed in deception. Therefore, $\mathcal{I} \subseteq \{k\} \times \mathcal{H}$ and $\hat{\mathcal{G}}_i$, $\hat{\boldsymbol{m}}_i$ and all components of $\hat{\boldsymbol{r}}_i$ except $\hat{r}_{ki}$ for $i \in \mathcal{H}$ are fixed when $\mathcal{A}_2$ starts. Next, we bound the probability $\mathsf{sas}_\alpha = \mathsf{sas}_\beta$ for $\alpha, \beta \in \mathcal{H}$.

Consider the authentic broadcast of c_k first, i.e., the case $\mathcal{I} = \{k\} \times \mathcal{H}$. The condition $\neg\mathsf{F}_1$ implies $(\hat{\mathcal{G}}_\alpha, \hat{\boldsymbol{m}}_\alpha) \neq (\hat{\mathcal{G}}_\beta, \hat{\boldsymbol{m}}_\beta)$ for some $\alpha, \beta \in \mathcal{H}$. As $x_1 \in_{\mathrm{u}} \{k\} \times \mathcal{R}$ the universality of h w.r.t. to all sub-key pairs[3] yields

$$\Pr[\hat{r}_k \in_{\mathrm{u}} \mathcal{R} : h((\hat{\mathcal{G}}_\alpha, \hat{\boldsymbol{m}}_\alpha), \ldots, \hat{r}_k, \ldots) = h((\hat{\mathcal{G}}_\beta, \hat{\boldsymbol{m}}_\beta), \ldots, \hat{r}_k, \ldots)] \leq \varepsilon_{\mathrm{u}}$$

where $\ldots$ denote the fixed components of $\hat{\boldsymbol{r}}_\alpha$ and $\hat{\boldsymbol{r}}_\beta$. So, we have obtained $\Pr[\mathcal{A}_2 = 1 | \mathcal{I} = \{k\} \times \mathcal{H}] \leq \Pr[\mathsf{sas}_\alpha = \mathsf{sas}_\beta | \mathcal{I} = \{k\} \times \mathcal{H}] \leq \varepsilon_{\mathrm{u}}$.

In the remaining case, let $\mathcal{H}_0$ be the set of honest nodes that receive c_k, i.e., $\mathcal{I} = \{k\} \times \mathcal{H}_0$. Since there is a compulsory node γ such $\mathcal{H} \subseteq \mathcal{G} \subseteq \hat{\mathcal{G}}_\gamma$ there are nodes $\alpha \in \mathcal{H}_0$ and $\beta \in \mathcal{H} \backslash \mathcal{H}_0$ such that $\mathcal{A}_2$ compares sas_α and sas_β. Moreover, α, β and sas_β are fixed before x_1 and almost regularity w.r.t. all sub-keys provides

$$\Pr[\hat{r}_k \in_{\mathrm{u}} \mathcal{R} : h((\hat{\mathcal{G}}_\alpha, \hat{\boldsymbol{m}}_\alpha), \ldots, \hat{r}_k, \ldots) = \mathsf{sas}_\beta] \leq \varepsilon_{\mathrm{r}}$$

where $\ldots$ denote the fixed components of $\hat{\boldsymbol{r}}_\alpha$. Therefore, we have proved the desired claim, i.e. $\Pr[\mathcal{G}_1^{\mathsf{nm}} = 1 | \mathcal{A}_1 \neq \perp] \leq \max\{\varepsilon_{\mathrm{u}}, \varepsilon_{\mathrm{r}}\}$. □

[3] Note that the varying components $\hat{r}_{k\alpha} = \hat{r}_{k\beta}$ can be in different locations of $\hat{\boldsymbol{r}}_\alpha$, $\hat{\boldsymbol{r}}_\beta$.

5 Security of Parallel Compositions

The parallel composition of message authentication protocols is often insecure although a single instance of the protocol is secure in a stand-alone setting. The phenomenon is caused by shared long term secrets. Bellare and Rogaway formalised a corresponding security model [3,4] where an adversary can execute several protocol instances concurrently and succeeds in deception if at least one protocol reaches an accepting state with incorrect outputs. The model was later extended to capture security of key agreement protocols [2] and then used in the context of manual authentication [21,20,17,18].

The possible security drop emerges only if two protocol instances are not statistically independent, i.e., share long-term keys. Clearly, an independent protocol instance cannot help the adversary, as the adversary can generate the protocol transcript himself. Therefore, the SAS-GMA protocol can be securely composed with any other protocol, provided that the following restrictions hold:

$\mathcal{R}_1$: Randomness used in the SAS-GMA instance is freshly generated.
$\mathcal{R}_2$: The output $(\mathcal{G}, \boldsymbol{m})$ is never used before all parties reach accepting state.
$\mathcal{R}_3$: The SAS messages determine unique instance of SAS-GMA.
$\mathcal{R}_4$: All group members have different identities, i.e., $\mathcal{G}$ is indeed a set.

The claim itself is valid for any protocol but we prove only that the SAS-GMA protocol is self-composable. The proof for the general case is analogous but requires a very fine-grained formalism similar to [15, p. 394–396] and provides no additional insight. Due to the space limitations, we omit such dubious details.

Bellare-Rogaway Model. Similarly to the stand-alone setting, an adversary $\mathcal{A}$ has complete control over the protocol participants $\mathcal{G}$ and their inputs $\boldsymbol{m}$ and in addition adaptive corruption is allowed. However, as opposed to the stand-alone model, $\mathcal{A}$ can adaptively launch[4] new instances $\pi^{(i)}$ of the protocol for $\mathcal{G}^{(i)}$ and $\boldsymbol{m}^{(i)}$. The adversary $\mathcal{A}$ succeeds in deception if the end state of at least one protocol instance $\pi^{(i)}$ is invalid, i.e., honest parties accept different outputs. Since a single instance of SAS-GMA has non-negligible deception probability we must bound the number of protocol instances that can be launched. A protocol π is (t, q, ε)-*self-composable* if any t-time adversary $\mathcal{A}$ that can launch up to q instances of π succeeds in deception with probability less than ε.

The SAS-GMA protocol in the original form is not suitable for parallel execution, as a party $\mathcal{P}_i$ who receives two first round messages from $\mathcal{P}_j$ cannot decide whether $\mathcal{P}_i$ invites him to participate in two separate group authentication protocols or an adversary tries to attack a single protocol instance. There must be a legitimate way to divide message between several protocols. As a solution, we assume that each protocol has an initiator $\mathcal{P}_i$ who first broadcasts or sends directly to group members a unique tag tag for the GMA protocol and tag is appended as an identifier to each protocol message. We emphasise that an adversary can alter tag. To assure condition $\mathcal{R}_3$, no participant $\mathcal{P}_i$ can have two parallel runs of SAS-GMA with the same set of participants $\hat{\mathcal{G}}_i$.

[4] See [2] for the thorough formalisation of the Bellare-Rogaway model.

Theorem 2. *Let the parameters of SAS-GMA protocol be such that a SAS-GMA instance is (t, ε)-secure in the stand-alone model. Then the protocol instances are also $(\tau, q, q\varepsilon)$-self-composable for $\tau = t - \mathcal{O}(1)$ if restrictions $\mathcal{R}_1$–$\mathcal{R}_4$ are satisfied.*

Proof. Let $\mathcal{B}$ be such a τ-time adversary that contradicts the claim. W.l.o.g. we can assume that an adversary launches the protocol instances in the following way. First, it chooses the initiator $\mathcal{P}_i$ and then the set of participants that get the introduction message tag from $\mathcal{P}_i$ and decide to reply. Second it provides the corresponding inputs to the participants. For simplicity, assume that $\mathsf{tag} \in \{1, \ldots, q\}$ and let $\varepsilon_{\mathsf{tag}}$ denote the probability that $\mathcal{B}$ succeeds in deception w.r.t. the instance $\pi^{(\mathsf{tag})}$. By the assumption $\varepsilon_1 + \cdots + \varepsilon_q > q\varepsilon$. Hence, we have the following simple reduction strategy $\mathcal{A}$. Given pk from $\mathcal{C}$:

1. Choose a protocol instance $k \in_{\mathrm{u}} \{1, \ldots, q\}$.
2. Simulate the Bellare-Rogaway model until $\mathcal{B}$ specifies $\mathcal{G}_k$ and $\hat{\boldsymbol{m}}_k$.
3. Send $\mathcal{G}_k$ and $\hat{\boldsymbol{m}}_k$ to the challenger $\mathcal{C}$ in the stand-alone model.
4. Continue the simulation by generating all messages tagged by $\mathsf{tag} \neq k$.
5. Obtain other messages with $\mathsf{tag} = k$ from the stand-alone environment.
6. If required by $\mathcal{B}$, corrupt the true nodes in the stand-alone environment.

Clearly, $\mathcal{A}$ provides a perfect simulation of the Bellare-Rogaway model, thus

$$\Pr\left[\mathcal{A} \text{ succeeds in deception}\right] = \frac{\varepsilon_1 + \cdots + \varepsilon_q}{q} > \varepsilon$$

and we have a desired contradiction. □

Note 1. Recall that we had a problem in the stand-alone model if an adversary decided to split the group. The latter cannot happen anymore as the initiator is always in $\hat{\mathcal{G}}_i$ and thus all nodes in the group must have same SAS test values.

6 Manually Authenticated Group Key Agreements

The main application of manual group message authentication (MGMA) is to establish a commonly shared secret key among the group members. We show how to combine MGMA with any group key agreement (GKA) protocol so that the resulting group key agreement protocol is secure against active attacks.

There is a trade-off between the security and the amount of authenticated communication. For many practical applications, the SAS message consists of 6 digits and thus has only 20 bits of entropy. So, an adversary can always succeed in deception with probability 2^{-20}. On the other hand, 2^{-20} is also the probability of not noticing an active attack. The latter is small enough to demotivate most of the possible attackers. Consequently, the subjective security level can be much higher, for example 2^{-40} if the probability of an active attack is below 10^{-6}.

Of course, the cryptographic security levels can be achieved only with sufficiently long SAS messages. Therefore, it is important to minimise the amount of manually authenticated communication in scenarios where nodes can form many subgroups. In particular, it should be easy to exclude corrupted nodes from the group without transferring any additional SAS messages.

Idealised Functionality. A *group key agreement* protocol π between n participants $\mathcal{G} = \{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$ starts with no input, is independent from the current state, and outputs $\mathcal{G}$ and a shared common secret key $\mathsf{key} \in_{\mathrm{u}} \mathcal{K}$.

Immunity Against Active Attacks. A group key agreement protocol π is (t, ε)-*immune against active attacks* if for any t-time adversary $\mathcal{A}$ that can choose a group $\mathcal{G} = \{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$ then the probability that uncorrupted parties $\mathcal{H}$ do not detect active attack is less than ε. Obviously, any GKA protocol that is (t, ε_1)-immune against active attacks and (t, ε_2)-secure against passive attacks is also $(t, \varepsilon_1 + \varepsilon_2)$-secure, as long as both definitions are given in the same attack model. For many practical cases, stand-alone security is sufficient.

Burmester-Desmedt Key Agreement Protocol. The Burmester-Desmedt (BD) key agreement protocol [8] is provably secure against passive attacks [6] and thus is a perfect starting point for a manually authenticated GKA. Though the Burmester-Desmedt GKA protocol is a generalisation of the Diffie-Hellman key agreement protocol, it can also be generalised for other two-party key agreement protocols, see the compiler of Just and Vaudenay [13]. For simplicity, consider a group of n participants[5] $\mathcal{P}_0, \ldots, \mathcal{P}_{n-1}$ arranged in a ring, see Fig. 4. The protocol has two rounds over an authenticated channel, while most of the schemes requires $\mathcal{O}(n)$ rounds. Here, let g be a generator of a q-element secure Diffie-Hellman Decision Group $\mathbb{G}$. At the end of the protocol, each participant $\mathcal{P}_i$ obtains $\widehat{\mathsf{key}}_i = g^{k_1 k_2 + k_2 k_3 + \ldots + k_n k_1}$, see Appendix A.

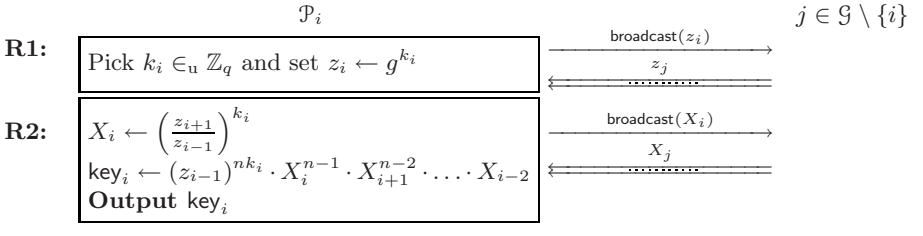

Fig. 4. The BD Group Key Agreement Protocol

New Manually Authenticated Group Key Agreement. Ideally, group members should run manually authenticated GKA only once to obtain a common group key key and long-term pairwise authentication keys, which provides possibility to re-run ordinary GKA protocols without additional SAS messages. The long-term pairwise authentication keys are formed based on Diffie-Hellman key exchange and the group key key generated by the BD GKA, see Fig. 5.

As the transcript of the BD GKA is authenticated with the SAS-GMA, the protocol is immune against active attacks with the same guarantees as Theorem 1 and Theorem 2 specify. Moreover, any two parties $\alpha, \beta \in \mathcal{H}$ can establish a pairwise secret key $\mathsf{key}_{\alpha,\beta} = f(g^{x_\alpha x_\beta})$, as they both know the corresponding

[5] The protocol can be trivially generalised to any n-element group $\mathcal{G}$.

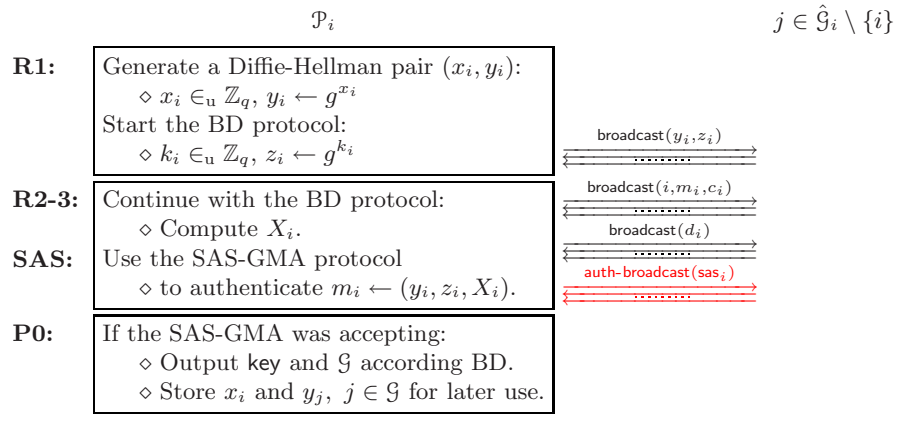

Fig. 5. The final SAS-based AKA Protocol with simplified notations

long-term public keys $y_i = g^{x_i}$ for all group members $i \in \mathcal{G}$. Hence, they can use any classical authentication protocol to protect new instances of GKA against active attacks. In particular, we can merge small groups $\mathcal{G}_1, \mathcal{G}_2$, if there is an honest party $\mathcal{P}_i \in \mathcal{G}_1 \cap \mathcal{G}_2$, by sending all intergroup communication through $\mathcal{P}_i$.

Of course, if the formed group is known to have a static nature, then one can skip the setup of long-term Diffie-Hellman keys $\mathsf{key}_{\alpha,\beta}$.

7 Applications and Conclusion

As shown in this article, our new SAS-based group message authentication protocol is provably secure in any computational context, provided that simple and natural restrictions $\mathcal{R}_1$–$\mathcal{R}_4$ are fulfilled. We also provided proofs under the natural non-malleability requirement that must be satisfied for all protocols that use commitments to temporarily hide sub-keys of hash function.

It allows building of secure SAS-based group key agreements, as presented in the last section. Such a key agreement protocol has the advantage that it does not require any trusted third party, any public-key infrastructure, nor any pre-shared key. Security is ensured peer-to-peer by using an authentication primitive, e.g. voice recognition for VoIP or string copy for devices. Therefore, consumers can establish and reconfigure security associations for electronic devices with minimal effort. In a certain sense, security can be provided as an add-on feature.

References

1. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations Among Notions of Security for Public-Key Encryption Schemes. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 26–45. Springer, Heidelberg (1998)
2. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated Key Exchange Secure against Dictionary Attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)

3. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
4. Bellare, M., Rogaway, P.: Provably secure session key distribution: the three party case. In: STOC 1995: Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, Las Vegas, Nevada, U.S.A, pp. 57–66. ACM press, New York (1995)
5. Bellare, M., Sahai, A.: Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 519–536. Springer, Heidelberg (1999)
6. Burmester, M., Desmedt, Y.: A secure and scalable Group Key Exchange system. Information Processiong Letter 94(3), 137–143 (2005)
7. Damgård, I., Groth, J.: Non-interactive and reusable non-malleable commitment schemes. In: STOC 2003: Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing, San Diego, California, U.S.A., pp. 426–437. ACM Press, New York (2003)
8. Desmedt, Y., Burmester, M.: A secure and efficient conference key distribution system (extended abstract). In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 275–286. Springer, Heidelberg (1995)
9. Di Crescenzo, G., Ishai, Y., Ostrovsky, R.: Non-interactive and non-malleable commitment. In: STOC 1998: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, Dallas, Texas, USA, pp. 141–150. ACM Press, New York (1998)
10. Diffie, W., Hellman, M.E.: New Directions in Cryptography. IEEE Transactions on Information Theory IT–22(6), 644–654 (1976)
11. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography (extended abstract). In: STOC 1991: Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing, New Orleans, Louisiana, U.S.A., pp. 542–552. ACM Press, New York (1991)
12. Fischlin, M., Fischlin, R.: Efficient non-malleable commitment schemes. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 413–431. Springer, Heidelberg (2000)
13. Just, M., Vaudenay, S.: Authenticated Multi-Party Key Agreement. In: Kim, K.-c., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 36–49. Springer, Heidelberg (1996)
14. 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)
15. Lindell, Y.: General composition and universal composability in secure multi-party computation. In: FOCS 2003, pp. 394–403. IEEE Computer Society, Los Alamitos (2003)
16. MacKenzie, P., Yang, K.: On Simulation-Sound Trapdoor Commitments. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 382–400. Springer, Heidelberg (2004)
17. 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)
18. 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)

19. Valkonen, J., Asokan, N., Nyberg, K.: Ad hoc security association for groups. In: Buttyán, L., Gligor, V.D., Westhoff, D. (eds.) ESAS 2006. LNCS, vol. 4357, pp. 150–164. Springer, Heidelberg (2006)
20. Vaudenay, S.: On Bluetooth repairing: Key agreement based on symmetric-key cryptography. In: Feng, D., Lin, D., Yung, M. (eds.) CISC 2005. LNCS, vol. 3822, pp. 1–9. Springer, Heidelberg (2005)
21. 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 Burmester Desmedt Key Derivation Proof

$$\begin{aligned}
\mathsf{key}_i &= (z_{i-1})^{nk_i} \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdot \ldots \cdot X_{i-2} \\
&= \left[z_{i-1}^{k_i}\right] \cdot \left[z_{i-1}^{k_i} \cdot X_i\right] \cdot \left[z_{i-1}^{k_i} \cdot X_i \cdot X_{i+1}\right] \cdot \ldots \cdot \left[z_{i-1}^{k_i} \cdot X_i \cdot X_{i+1} \cdot \cdots \cdot X_{i-2}\right] \\
&= \left[g^{k_{i-1}k_i}\right] \cdot \left[g^{k_ik_{i+1}}\right] \cdot \left[g^{k_{i+1}k_{i+2}}\right] \cdot \ldots \cdot \left[g^{k_{i-2}k_{i-1}}\right] \\
&= g^{k_1k_2+k_2k_3+\ldots+k_nk_1}
\end{aligned}$$