Secure Communications over Insecure Channels Based on Short Authenticated Strings

Serge Vaudenay

EPFL
CH-1015 Lausanne, Switzerland
http://lasecwww.epfl.ch

Abstract. We propose a way to establish peer-to-peer authenticated communications over an insecure channel by using an extra channel which can authenticate very short strings, e.g. 15 bits. We call this SAS-based authentication as for authentication based on Short Authenticated Strings. The extra channel uses a weak notion of authentication in which strings cannot be forged nor modified, but whose delivery can be maliciously stalled, canceled, or replayed. Our protocol is optimal and relies on an extractable or equivocable commitment scheme.
This approach offers an alternative (or complement) to public-key infrastructures, since we no longer need any central authority, and to password-based authenticated key exchange, since we no longer need to establish a confidential password. It can be used to establish secure associations in ad-hoc networks. Applications could be the authentication of a public key (e.g. for SSH or PGP) by users over the telephone, the user-aided pairing of wireless (e.g. Bluetooth) devices, or the restore of secure associations in a disaster case, namely when one remote peer had his long-term keys corrupted.

1 On Building Secure Communications

One of the key issue of modern cryptography is the problem of establishing a secure peer-to-peer communication over an insecure channel. Assuming that we can establish a private and authenticated key, standard tunneling techniques can achieve it. In the seminal work of Merkle [32] and Diffie and Hellman [18], the private and authenticated key establishment problem was reduced to establishing a communication in which messages are authenticated. Public key cryptosystems such as RSA [39] further reduce to the establishment of an authenticated public key. Note that the seed authentication is also a limiting factor for quantum cryptography [10].

Another major step was the notion of password-based authenticated key agreement which was first proposed by Bellovin and Merritt [8,9] and whose security was proven by Bellare, Pointcheval, and Rogaway [5] in the random oracle model. Another protocol, provably secure in the standard model, was proposed by Katz, Ostrovsky, and Yung [29]. Here, we assume that a private and authenticated short password was set up prior to the protocol. The key agreement protocol is such that no *offline* dictionary attack is feasible against the password so that the threat model restricts to *online* password-guessing attacks which are easily detectable.[1] When compared to the above approach, we thus reduce the size of the initial key, but we require its confidentiality again.

[1] See Chapter 7 of [12] for a survey on password-based authenticated key agreement.

3-party models offer other solutions. The Needham-Schroeder model [34] assumes that everyone has a private authenticated key with a Trusted Third Party (TTP). Kerberos [30] is a popular application. The authenticated (only) key model is achieved with the notion of certificate by a Certificate Authority (CA). TLS [19] typically uses X.509 [27] certificates. Note that TLS authenticates the server to the client (which is enough to open a secure tunnel), but that the client authentication is typically based on a (short) password through the tunnel. Finally, fully password-based 3-party authenticated key agreement was studied by Abdalla, Fouque, and Pointcheval [3].

Ah-hoc networks cannot assume the availability of a central third party and setting up a secure network is a real challenge. Networks which are not attended by a human operator (e.g. sensor networks) can use a pragmatic solution such as the "resurrecting duckling" paradigm of Stajano and Anderson [40]. Smaller networks which are attended by a human operator such as networks of personal mobile devices (laptops, cell phones, PDAs, headsets, ...) can use the human operator as a third party, but must minimize his job. A familiar example is the Bluetooth [2] pairing: the operator picks a random PIN code and types it on devices to be associated, and a pairing protocol is run through a wireless link to establish a 128-bit private authenticated key. Operator-to-device transmissions is assumed to be secure (i.e. confidential and authenticated). However, as shown by Jakobsson and Wetzel [28], the standard Bluetooth pairing protocol is insecure unless we assume that either the radio communications in the pairing protocol are confidential as well, or the PIN code is long enough.

	long key	short key
A + C channel	symmetric-key cryptography	password-based authenticated key agreement
A channel	public-key cryptography	*SAS-based authentication*

Fig. 1. Two-Party Private and Authenticated Key Establishment Paradigms.

Solutions to the secure communications over insecure channels therefore seem to go to two opposite directions (which further translate in a 3-party model): remove the confidential channel (and use public keys) or use short passwords rather than long secret keys. A natural additional step consists of combining the two approaches: using an extra channel which only provides authentication and which is limited to the transmission of short bitstrings. A straightforward solution consists of authenticating every message of a regular key agreement protocol such as the Diffie-Hellman protocol [18] as suggested by Balfanz et al. [4]. The size of messages is typically pretty high, but can be reduced by authenticating only the hashed values of the messages. By using a collision resistant hash function, the number of bits to authenticate typically reduces to 160 bits, but a 160-bit string is still pretty long: by using the encoding rules of the RFC 1760 [23] standard we can represent 160 bits in a human friendly way by using 16 small English words. A second solution by Hoepman [25,26] can significantly reduce this number. It is based on special purpose hash functions. However, the security proof is incomplete and no hash functions with the required properties happen to exist. Another approach by Gehrmann, Mitchell, and Nyberg [21] (dedicated to the Bluetooth pairing problem) called MANA I (as for Manual Authentication), based on a universal hash function family, can perform a message authentication. They however require a stronger notion of authentication channel. Interestingly, those protocols were proposed to replace the current insecure Bluetooth pairing protocol and are now suggesting new solutions in more traditional secure communication standards, e.g. IKEv2 (see Nyberg [35]).

In this paper, we study solutions which can achieve message authentication by using the (weak) authentication of a short bitstring. We call them SAS-based schemes as for "Short Authenticated Strings". A typical application is the pairing problem in wireless networks such as Bluetooth. Another application is secure peer-to-peer communication: if two persons who know each other want to set up a secure communication they can exchange SAS on a postcard, by fax, over a phone call, a voice message, or when they physically encounter.

The other MANA protocols [21,22], as well as the extension of the Hoepman protocols by Peyrin and Vaudenay [38], can be seen as a 3-party translation called the "User-Aided Key Exchange (UAKE)". The user becomes a real participant in the protocol who does simple computations like comparing strings or picking a random one. The security proof in the present paper could equally apply to these cases.

2 Preliminaries

2.1 Communication and Adversarial Models

We consider a communication network with (insecure but cheap) broadband communication channels and narrowband channels which can be used to authenticate short messages. Authenticated channels are related to a node identity ID. An illustration for this model is the location-limited channel of Balfanz et al. [4]. For instance, a user A working on his laptop in an airport lounge would like to print a confidential document on a laser printer B through a wireless link. The user reading a message on the LCD screen of B and typing it on the laptop keyboard is an authenticated channel from the identified printer to the laptop. A SAS-based authentication protocol can be used to transmit and authenticate the public key of the printer by keeping small the transmission over the authenticated channel. Another example is when Bob would like to authenticate the PGP public key of Alice in his key ring. If he can recognize her voice, she can spell a SAS on his voice mail. If he can recognize her signature, she can send a signed SAS to him by fax or even on a postcard.

Adversarial model. Except for the authentication channels, we assume that the adversary has full control on the communication channels. In particular, she can prevent a message from being delivered, she can delay it, replay it, modify it, change the recipient address, and of course, read it. We adopt here the stronger security model of Bellare-Rogaway [6,7] which even assumes that the adversary has full control on which node launches a new instance of a protocol, and on which protocol instance runs a new step of the protocol. Bellare-Rogaway [6,7] considered protocols for access control or key agreement which basically have no input. Here, protocols do have inputs and we assume that the adversary can choose it. Namely, we assume that the adversary has access to a

$\mathsf{launch}(n,r,x)$ oracle in which n is a node of the network, r is a character (i.e. a role to play in the protocol), and x is the input of the protocol for this character. This oracle returns a unique instance tag π_n^i. Since a node can a priori run concurrent protocols, there may be several instances related to the same node n. To simplify we restrict ourselves to 2-party protocols so that there are only two characters Alice and Bob in the protocol. Any node can play any of these characters. A q-shot adversary is an adversary limited to q $\mathsf{launch}(\cdot,\text{Alice},\cdot)$ queries and q $\mathsf{launch}(\cdot,\text{Bob},\cdot)$ queries. The adversary also has access to the oracle $\mathsf{send}(\pi_n^i,m)$ which sends a message m to a given instance and returns an m' message which is meant to be sent to the other participant. For example, a protocol with input x and y respectively can be run on node A and B by

1. $\pi_a \leftarrow \mathsf{launch}(A,\text{Alice},x)$
2. $\pi_b \leftarrow \mathsf{launch}(B,\text{Bob},y)$
3. $m_1 \leftarrow \mathsf{send}(\pi_a,\emptyset)$
4. $m_2 \leftarrow \mathsf{send}(\pi_b,m_1)$
5. $m_3 \leftarrow \mathsf{send}(\pi_a,m_2)$
6. ...

until a message is a termination message. Note that the Bellare-Rogaway model [6,7] considers additional oracles $\mathsf{reveal}(\pi_n^i)$ (which reveals the output from a protocol instance), $\mathsf{corrupt}(n,x)$ (which corrupts the collection of instances related to the node n and forces their private states to become x), and test (which is specific to the semantic security of key agreement protocols). These oracles are not relevant here since we never use long-term secrets and the output of the protocols is not secret.

Authentication channel. The authentication channels provide to the recipient of a message the insurance on whom sent it as is. In particular the adversary cannot modify it (i.e. integrity is implicitly protected). On the other hand she can stall it, remove it, or replay it. We stress that those channels are not assumed to provide confidentiality. Formally, an authentication channel from a node n refers to the identifier ID_n. The send oracle maintains an unordered set of authenticated messages in all authenticated channels from the node n. Only send oracles with a π_n^i instance can insert a new message x in this set. Later, when a send oracle is queried with any instance tag and a message of form $\mathsf{authenticate}_{\mathsf{ID}_n}(x)$, it is accepted by the oracle only if x is in the set. Note that concurrent or successive instances related to the same node write in the same set: messages from the node are authenticated, but the connection to the right instance is not. Authenticated channels can typically be implemented, e.g. by human telephone conversations, voice mail messages, handwritten postcards, etc.

Stronger authentication channel. We can also consider *strong authentication channels*, namely authenticated channels which provide an additional security property. We list here a few possible properties in an informal way.

Stall-free transmission: from the time an authenticated message is released by a send oracle to the time it is given as input to a send oracle query, no other oracle query can be made. Hence, either the message is treated by the immediately following oracle query, or it is never used.

Transmission with acknowledgment: messages are released together with a destination node identifier, and the sending instance is given a way to check whether at least one instance related to this node has received the message or not.

Listener-ready transmission: similarly, the sending instance can check if an instance related to the destination node is currently listening to the authenticated channel.

Face-to-face conversations achieve all properties. Telephone conversations achieve the last two properties: Alice starts talking to Bob when she is aware that Bob is listening, and subtle human senses assure her that Bob has heard her message. Less interactive communications such as voice mail messages do not provide these properties: Bob may not even be aware that Alice wants to send him a message, and Alice has neither way to know when Bob has received it, nor insurance that her message was recorded.

Message authentication protocol. Our message authentication protocols have input m on the side of the claimant Alice and output $I||\hat{m}$ on the side of the verifier Bob. Intuitively, they should be such that $I = \mathsf{ID}_A$ and $\hat{m} = m$, meaning that $\hat{m}$ coming to node B was authenticated as sent by ID_A, the identifier of Alice.

On a global perspective, several $\mathsf{launch}(A_k, \text{Alice}, m_k)$ and $\mathsf{launch}(B_\ell, \text{Bob}, \emptyset)$ are queried, which creates several $\pi_{A_k}^{i_k}$ instances of Alice (authentication claims) and several $\pi_{B_\ell}^{j_\ell}$ instances of Bob (authentication verifications). If no attack occurs then we have a perfect matching between the k's and ℓ's, related instances have matching conversations which fully follow the protocol specifications, and the $\pi_{B_\ell}^{j_\ell}$ ends with output $\mathsf{ID}_{A_k}||m_k$ for the matching k. In any other case we say that an attack occurred. We say that an attack is successful if there exists at least an instance $\pi_{B_\ell}^{j_\ell}$ which terminated and output $I||\hat{m}$ such that there is no k for which $I = \mathsf{ID}_{A_k}$ and $\hat{m} = m_k$. Note that many protocol instances can endlessly stay in an unterminated state or turn in an abort state. In particular, we do not consider denial-of-services attacks.

2.2 Commitment Schemes

Our protocols are based on commitment schemes. They are used to commit on an arbitrary non-hidden message m together with a hidden k-bit string r. We formalize them by three algorithms.

setup which generates a random parameter K_P (which is used by all other algorithms and omitted from notations for simplicity reasons) and a secret key K_S.

$\mathsf{commit}(m, r)$ which takes a message $x = m||r$ and produces two strings: a *commit* value c and a decommit value d. Here, we consider that x includes a part m which is not meant to be hidden and a part r which is a hidden k-bit string. We can call m a tag for the commitment so that we have a *tag-based commitment* to r. Note that this algorithm is typically non deterministic.

$\mathsf{open}(m, c, d)$ which takes m, c, and d and yields a message r or an error signal. We require this algorithm to be *deterministic* and to be such that whenever there exists r such that (c, d) is a possible output for $\mathsf{commit}(m, r)$, $\mathsf{open}(m, c, d)$ yields r.

Note that the setup plays no real role so far. It is used in extensions of commitment schemes. We keep it anyway to have definitions well suited to all kinds of commitment schemes that will be used. Commitment schemes have two security properties.

– (T,ε)-*hiding*: no algorithm $\mathcal{A}$ bounded by a time complexity T can win the following game by interacting with a challenger $\mathcal{C}$ with a probability higher than $2^{-k}+\varepsilon$.
  1. $\mathcal{C}$ runs setup and sends K_P to $\mathcal{A}$.
  2. $\mathcal{A}$ selects a tag m and sends it to $\mathcal{C}$.
  3. $\mathcal{C}$ picks a random r, runs commit on (m,r), gets (c,d), and sends c to $\mathcal{A}$.
  4. $\mathcal{A}$ yields r' and wins if $r=r'$.

  When $T=+\infty$ and $\varepsilon=0$, we say that the scheme is *perfectly hiding*.
– (T,ε)-*binding*: no algorithm $\mathcal{A}$ bounded by a time complexity T can win the following game by interacting with a challenger $\mathcal{C}$ with a probability higher than $2^{-k}+\varepsilon$.
  1. $\mathcal{C}$ runs setup and sends K_P to $\mathcal{A}$.
  2. $\mathcal{A}$ selects a tag m and sends it to $\mathcal{C}$.
  3. $\mathcal{A}$ selects a c and sends it to $\mathcal{C}$.
  4. $\mathcal{C}$ picks a random r and sends it to $\mathcal{A}$.
  5. $\mathcal{A}$ computes a d and wins if (m,c,d) opens to r.

  When $T=+\infty$ and $\varepsilon=0$, we say that the scheme is *perfectly binding*.

Commitment schemes can be relative to an oracle, in which case all algorithms and adversaries have access to the oracle. However, they have no access to the complete history of oracle calls. Extensions of commitment schemes have extra algorithms which do have access to this history.

Extractable commitment. In this extension of commitment schemes, there is an additional *deterministic* algorithm $\mathsf{extract}_{K_S}(m,c)$ which yields r when there exists d such that (m,c,d) opens to r. When using oracles, this algorithm is given the history of oracle queries. Clearly, extractable commitments are perfectly binding. Adversaries playing the hiding game can make oracle calls to $\mathsf{extract}$, except on the committed m tag.

Equivocable commitment. In this extension of commitment schemes, there are two algorithms $\mathsf{simcommit}_{K_S}(m)$ and $\mathsf{equivocate}_{K_S}(m,c,r,\xi)$. $\mathsf{simcommit}$ returns a fake commit value c and an information ξ, and $\mathsf{equivocate}$ returns a decommit value d such that (m,c,d) opens to an arbitrary r for (c,ξ) obtained from $\mathsf{simcommit}$. For any $K_P||K_S$ and any m, the distribution of fake commit values is assumed to be identical to the distribution of real commit values to any r with tag m. From this we deduce that the commitment is perfectly hiding. Adversaries playing the binding game can make oracle calls to $\mathsf{simcommit}$ and $\mathsf{equivocate}$, except on the committed m tag, and are assumed not to see ξ. Namely, the $\mathsf{equivocate}$ oracle works only if there was a matching oracle call to $\mathsf{simcommit}$ before, and gets ξ directly from the history. In our paper, we further assume that adversaries are limited to a single query to $\mathsf{simcommit}$ and $\mathsf{equivocate}$. This is a quite restrictive assumption, but it will be enough for our purpose.

Example 1 (Ideal commitment model). A first commitment scheme model which can be used is the *ideal commitment model*. Here, we assume that the network includes a trusted third party (TTP) with whom anyone can communicate in a perfectly secure way. The commit algorithm consists of securely sending a message x to the TTP. The TTP attaches it to a nonce value c which is returned to the sender and inserts (c,x) in an archive with a protection flag set. There is no decommit value, but the same sender can replace the disclosure of it by a decommit call to the TTP. Then, the TTP clears the protection flag of the (c,x) entry which becomes readable by anyone. Obviously we obtain a perfectly binding and hiding scheme. Note that it is extractable and equivocable.

Example 2 (Extractable commitment based on a random oracle). We take an easy commitment scheme which was already mentioned in Pass [37]. Here, we use a random oracle H which upon a query d returns a random value $c \leftarrow H(d)$ in $\{0,1\}^{\ell_c}$. The $\mathsf{commit}(m,r)$ algorithm simply picks a random value e in $\{0,1\}^{\ell_e}$, takes $d = r||e$, and calls $c \leftarrow H(m||d)$. The $\mathsf{open}(m,c,d)$ algorithm simply extracts r from d and checks that $c = H(m||d)$. When all oracle queries to H produce no collision on $m||c$, commitments can trivially be extracted form the history. When the number of queries is q, this is the case, except with probability less than $q^2.2^{-\ell_c-1}$. We prove that the best strategy for an adversary to play the hiding game is to look for $r||e$ exhaustively by trying it with H. Actually, if for each r the adversary has queried q_r values $H(m||r||e)$ on the committed tag m, the probability of success when answering r is one if the right $r||e$ was found, which happens with probability $2^{-\ell_e-k}\sum_s q_s$, and $(2^{\ell_e} - q_r)/(2^{\ell_e+k} - \sum_s q_s)$ in the other case. Hence, the overall probability of success is $2^{-k} + 2^{-\ell_e-k}(\sum_s q_s - q_r)$ which is at most $2^{-k} + q.2^{-\ell_e-k}$. Hence, when H is limited to q accesses, the commitment scheme is $(+\infty, q.2^{-\ell_e-k})$-extractable with probability at least $1 - q^2.2^{-\ell_c-1}$. So with $\ell_c = 2\ell_e$ and $\ell_e = 80$, the scheme is pretty safe until the complexity reaches a number of oracle calls within the order of magnitude of 2^{80-k}.

Example 3 (Equivocable commitment based on a signature scheme). One can easily prove (see Appendix B) that the notion $\mathsf{SSTC}(2)$ of *simulation-sound trapdoor commitment* as defined in MacKenzie-Yang [31], where the adversary is given two $\mathsf{equivocate}$ oracle calls, provides equivocable commitments following our definition. Hence, from [31] we get a nice equivocable commitment scheme based on DSA [1] and another one based on the Cramer-Shoup signature scheme [14]. We can also use the stronger notion of non-malleable commitments [15,16,20], and in particular the Damgård-Groth commitment scheme [17] based on the strong RSA assumption in the common reference string (CRS) model.

Introducing a public key may look paradoxical since the purpose of our work is to get rid of any a priori authenticated public key. This is the puzzling aspect of the CRS model: we assume usage of a public key for which a secret key exists, but that nobody can use it. We can even rely on the uniform random string (URS) model (see [17]) in which the public key is a uniformly distributed reference bitstring.

2.3 Previous Work

The Gehrmann-Mitchell-Nyberg MANA I [21] protocol is depicted in Fig. 2.[2] By convention we put a hat on received messages which are not authenticated since they can differ from sent messages in the case of an active attack. Other MANA protocols are designed for two devices attended by a user who can do simple operations. MANA I is the only protocol in which the user is a passive messenger who only forwards messages.

[2] Note that the original MANA I protocol is followed by an authenticated acknowledgment.

MANA I uses a universal hash function family H. Proposed constructions lead to 16–20 bit long SAS values. Although MANA I is essentially non-interactive, the security requires a stronger authentication channel. Otherwise, one can run the following attack.

1. $\pi_a \leftarrow \mathsf{launch}(A, \text{Alice}, m)$
2. $\pi_b \leftarrow \mathsf{launch}(B, \text{Bob}, \emptyset)$
3. $m \leftarrow \mathsf{send}(\pi_a, \emptyset)$
4. $\mathsf{authenticate}_{\text{Alice}}(K||\mu) \leftarrow \mathsf{send}(\pi_a, \emptyset)$
5. find $\hat{m} \neq m$ such that $H_K(\hat{m}) = \mu$ by random search
6. $\mathsf{send}(\pi_b, \hat{m})$
7. $\mathsf{send}(\pi_b, \mathsf{authenticate}_{\text{Alice}}(K||\mu))$

MANA I is nevertheless secure when using an authentication channel which provides stall-free transmission or listener-ready transmission as defined in Section 2.1.

$$
\begin{array}{rcl}
\text{Alice} & & \text{Bob} \\
\textbf{input}: m & & \\
\text{pick } K \in_U \{0,1\}^k & \xrightarrow{\quad\quad m \quad\quad} & \\
\mu \leftarrow H_K(m) & \xrightarrow{\mathsf{authenticate}_{\text{Alice}}(K||\mu)} & \text{check } \mu = H_K(\hat{m}) \\
& & \textbf{output}: \text{Alice}, \hat{m}
\end{array}
$$

Fig. 2. The MANA I Protocol.

The Hoepman authenticated key agreement protocol [25] is depicted in Fig. 3. It consists of a commitment exchange and an authentication exchange, followed by a regular Diffie-Hellman protocol [18].[3] The protocol is based on the decisional Diffie-Hellman problem in a group G. It works with the hypothesis that H_1 and H_2 are two hash functions such that H_2 is balanced from G to $\{0,1\}^k$, and given a uniformly distributed X in G, the two random variables $H_1(X)$ and $H_2(X)$ are independent. Although no example which meet these criteria is provided in [25][4], Hoepman provided a sketch of security proof for the complete protocol[5] in the Bellare-Rogaway model.

3 Non-Interactive Message Authentication

We first present a solution based on a collision resistant hash function inspired by Balfanz et al. [4]. Since the result is quite straightforward, the proof is omitted.

[3] Note that first committing to the Diffie-Hellman values was already suggested by Mitchell-Ward-Wilson [33].

[4] One can note that the criterion on H_2 seemingly suggests that the order of G should be a multiple of 2^k which is not the case in classical Diffie-Hellman groups so (H_1, H_2) instances may not exist at all.

[5] Fig. 3 presents a simplified version of the protocol. The complete protocol is followed by a key confirmation and a key derivation based on the leftover hash lemma [24] (see Boneh [11]).

$$
\begin{array}{rcl}
\text{Alice} & & \text{Bob} \\
\text{pick } x_A \in_U \{0,\ldots,|G|-1\} & & \text{pick } x_B \in_U \{0,\ldots,|G|-1\} \\
y_A \leftarrow g^{x_A} & & y_B \leftarrow g^{x_B} \\
h_A \leftarrow H_1(y_A) & \xrightarrow{\quad h_A \quad} & \\
& \xleftarrow{\quad h_B \quad} & h_B \leftarrow H_1(y_B) \\
\mathsf{SAS}_A \leftarrow H_2(y_A) & \xrightarrow{\mathsf{authenticate}_{\mathrm{Alice}}(\mathsf{SAS}_A)} & \\
& \xleftarrow{\mathsf{authenticate}_{\mathrm{Bob}}(\mathsf{SAS}_B)} & \mathsf{SAS}_B \leftarrow H_2(y_B) \\
& \xrightarrow{\quad y_A \quad} & \text{check } \hat{h}_A, \mathsf{SAS}_A \\
\text{check } \hat{h}_B, \mathsf{SAS}_B & \xleftarrow{\quad y_B \quad} & \\
k_A \leftarrow (\hat{y}_B)^{x_A} & & k_B \leftarrow (\hat{y}_A)^{x_B} \\
\textbf{output}\text{: Bob}, k_A & & \textbf{output}\text{: Alice}, k_B
\end{array}
$$

Fig. 3. The Hoepman Authenticated Key Agreement Protocol.

Theorem 4. *Let μ be the overall time complexity of the message authentication protocol in Fig. 4. Given an adversary of time complexity T, number of* launch *oracle queries Q, and probability of success p, we can find collisions on H within a complexity $T + \mu Q$ and same probability.*

One advantage of this protocol is that it is non-interactive. Collision resistance requires the number of authenticated bits to be at least 160 which is still quite large. We can actually half this number by using the Pasini-Vaudenay protocol [36] based on a hash function resisting to second preimage attacks (a.k.a. weakly collision resistant hash function).[6] Note that the MANA I protocol requires less bits, but a stronger authentication channel which renders the protocol "less non-interactive".

$$
\begin{array}{rcl}
\text{Alice} & & \text{Bob} \\
\textbf{input}\text{: } m & & \\
& \xrightarrow{\qquad m \qquad} & \\
h \leftarrow H(m) & \xrightarrow{\mathsf{authenticate}_{\mathrm{Alice}}(h)} & \text{check } h = H(\hat{m}) \\
& & \textbf{output}\text{: Alice}, \hat{m}
\end{array}
$$

Fig. 4. Message Authentication using a Collision Resistant Hash Function.

4 SAS-based Message Authentication

In Fig. 5 is depicted a SAS-based message authentication protocol. Basically, Alice first commits on her (non-hidden) input message m together with a hidden random string R_A.

[6] This protocol consists of sending $m||c||d||\mathsf{authenticate}(H(c))$ where (c,d) is obtained from m by using a trapdoor commitment.

After reception of m and the commit value c, Bob picks a random string R_B and gives it to Alice. Alice then opens her commitment by sending a d and sends $\mathsf{SAS} = R_A \oplus R_B$ through her authenticated channel. Bob can finally check the consistency of this string with the commitment. This protocol can be used to authenticate in both directions and henceforth for authenticated key agreement as detailed in Appendix A.

With weak authentication channels, the adversary can run as follows:

– impersonate Bob and start the protocol with Alice with $m, c, \hat{R}_B, d$,
– stall the SAS message,
– launch several Bob's and impersonate Alice with $\hat{m}, \hat{c}, \hat{d}$ until Alice's SAS matches,
– deliver the SAS and complete the protocol.

This attack works within a number of trials around 2^k. Note that the attack is not discrete on Bob's side since many protocols abort. Similarly, the adversary can launch many instances of Alice and make a catalog of Alice's SAS messages. After Alice has performed quite a lot of protocols, the catalog can be close to the complete catalog of 2^k messages. With this collection, the adversary can impersonate Alice. Note that the attack is pretty discrete here. We can further trade the number of instances of Bob against the number of instances of Alice and have a birthday paradox effect. Namely, with $2^{\frac{k}{2}}$ concurrent runs of Alice and Bob we have fair chances of success.

$$
\begin{array}{rcl}
\text{Alice} & & \text{Bob} \\
\textbf{input}: m & & \\
\text{pick } R_A \in_U \{0,1\}^k & & \text{pick } R_B \in_U \{0,1\}^k \\
c||d \leftarrow \mathsf{commit}(m||R_A) & \xrightarrow{\quad m||c \quad} & \\
& \xleftarrow{\quad R_B \quad} & \\
& \xrightarrow{\quad d \quad} & \hat{R}_A \leftarrow \mathsf{open}(\hat{m}, \hat{c}, \hat{d}) \\
\mathsf{SAS} \leftarrow R_A \oplus \hat{R}_B & \xrightarrow{\mathsf{authenticate}_{\mathrm{Alice}}(\mathsf{SAS})} & \text{check } \mathsf{SAS} = \hat{R}_A \oplus R_B \\
& & \textbf{output}: \text{Alice}, \hat{m}
\end{array}
$$

Fig. 5. SAS-based Message Authentication.

Theorem 5. *We consider one-shot adversaries against the message authentication protocol in Fig. 5. We denote by T and p their time complexity and probability of success, respectively. We assume that the commitment scheme is either (T_C, ε)-extractable or (T_C, ε)-equivocable. There exists a (small) constant μ such that for any adversary, we have either $p \leq 2^{-k} + \varepsilon$ or $T \geq T_C - \mu$.*

Our results seemingly suggest that for any secure commitment scheme the success probability of practical one-shot attacks is bounded by $2^{-k} + \varepsilon$ where ε is negligible. Our results are pretty tight since adversaries with a probability of success 2^{-k} clearly exist.

Proof. Due to the protocol specifications, a successful adversary must perform the following sequences of steps to interact with Alice and Bob. The way the adversary interleaves the two sequences is free.

1. select m, $\pi_a \leftarrow \mathsf{launch}(\cdot, \text{Alice}, m)$
2. $c \leftarrow \mathsf{send}(\pi_a, \emptyset)$
3. select $\hat{R}_B$, $d \leftarrow \mathsf{send}(\pi_a, \hat{R}_B)$

1. $\pi_b \leftarrow \mathsf{launch}(\cdot, \text{Bob}, \emptyset)$
2. select $\hat{m}||\hat{c}$, $R_B \leftarrow \mathsf{send}(\pi_b, \hat{m}||\hat{c})$
3. select $\hat{d}$, $\mathsf{send}(\pi_b, \hat{d})$

The attack runs like a game for the adversary $\mathcal{A}$ who wins if (m,c,d) (resp. $(\hat{m},\hat{c},\hat{d})$) opens to R_A (resp. $\hat{R}_A$) such that $R_A \oplus \hat{R}_B = \hat{R}_A \oplus R_B$. The game starts by receiving the selected public parameter K_P (if any) for the commitment scheme. We can make $\mathcal{A}$ play with a simulator for Alice and Bob. Note that an attack implies that $m \neq \hat{m}$.

$$
\begin{array}{lcl}
\mathcal{A} & \mathcal{B} & \mathcal{C} \\
\xleftarrow{K_P} & & \xleftarrow{K_P} \\
\xrightarrow{m} & & \xrightarrow{m} \\
\xleftarrow{c} & & \xleftarrow{c} \\
\xrightarrow{\hat{R}_B} & \text{pick } R & \xrightarrow{R} \\
\xleftarrow{d} & R_A \leftarrow \mathsf{open}(m,c,d) & \xleftarrow{d} \\
\xrightarrow{\hat{m}||\hat{c}} & \hat{R}_A \leftarrow \mathsf{extract}(\hat{m},\hat{c}) & \\
\xleftarrow{R_B} & \text{pick } R_B & \\
\xrightarrow{\hat{d}} & &
\end{array}
$$

Case 1

$$
\begin{array}{lcl}
\mathcal{A} & \mathcal{B} & \mathcal{C} \\
\xleftarrow{K_P} & & \xleftarrow{K_P} \\
\ldots & & \\
\xrightarrow{m} & & \xrightarrow{m} \\
\xleftarrow{c} & & \xleftarrow{c} \\
\ldots & & \\
\xrightarrow{\hat{m}||\hat{c}} & \hat{R}_A \leftarrow \mathsf{extract}(\hat{m},\hat{c}) & \\
\xleftarrow{R_B} & \text{pick } R_B & \\
\ldots & & \\
\xrightarrow{\hat{R}_B} & R_A \leftarrow \hat{R}_A \oplus R_B \oplus \hat{R}_B & \xrightarrow{R_A} \\
\xleftarrow{d} & & \xleftarrow{d} \\
\ldots & & \\
\xrightarrow{\hat{d}} & &
\end{array}
$$

Case 2

Fig. 6. Reduction with Extractable Commitments.

Extractable commitments. We construct from $\mathcal{A}$ an adversary $\mathcal{B}$ who plays an augmented hiding game with the help of one query to the extract oracle. The augmented hiding game consists of the regular one followed by $\mathcal{C}$ sending the right decommit value d. Obviously, an adversary playing this augmented game can be transformed into an adversary playing the regular game with the same winning probability. As depicted in Fig. 6, $\mathcal{B}$ first receives K_P and sends it to $\mathcal{A}$. Then, he simulates Alice and Bob to $\mathcal{A}$. Whenever $\mathcal{A}$ sends m to Alice, $\mathcal{B}$ submits it in the augmented hiding game. Then, $\mathcal{B}$ receives a challenge c which is sent back to $\mathcal{A}$ (i.e., Alice does not compute any commitment but rather uses the challenge). Whenever $\mathcal{A}$ sends $\hat{m}||\hat{c}$ to Bob, $\mathcal{B}$ can extract $\hat{R}_A$ by calling $\mathsf{extract}(\hat{m},\hat{c})$. When $\mathcal{A}$ sends $\hat{R}_B$ to Alice, we distinguish two cases.

Case 1. If $\mathcal{A}$ did not send $\hat{m}||\hat{c}$ to Bob yet, there is essentially one way to interleave the two sequences which consists in first playing with Alice, then playing with Bob. Here, $\mathcal{B}$ answers a random R to the challenge and wins with probability 2^{-k}. $\mathcal{B}$ continues the simulation and plays with $\mathcal{A}$ by receiving d and sending it from Alice to $\mathcal{A}$. Then, $\mathcal{A}$ sends $\hat{m}||\hat{c}$ to Bob from which $\mathcal{B}$ extracts $\hat{R}_A$. Bob's simulation picks a random R_B, but R_A, $\hat{R}_A$, and $\hat{R}_B$ are fixed. So, $\mathcal{A}$ wins with probability 2^{-k}. Hence, $\mathcal{A}$ and $\mathcal{B}$ win their respective game with the same probability in this case.

Case 2. If now $\mathcal{A}$ has sent $\hat{m}||\hat{c}$ to Bob, $\mathcal{B}$ can compute $R_A = \hat{R}_A \oplus R_B \oplus \hat{R}_B$ and answer R_A. $\mathcal{B}$ receives d and sends it to $\mathcal{A}$. A typical example is depicted on Fig. 6. Here, $\mathcal{A}$ and $\mathcal{B}$ win or loose at the same time, thus win with the same probability.

We observe that the simulation by $\mathcal{B}$ is perfect and that the extraction is legitimate since $m \neq \hat{m}$. We deduce that we can win the hiding game with the same probability as the attack. Let μ be the complexity of the protocol plus one oracle call. The complexity of $\mathcal{B}$ is essentially the complexity of $\mathcal{A}$ plus μ. The success probability of the attack is thus at most $2^{-k} + \varepsilon$.

$$
\begin{array}{lcr}
\mathcal{A} & \mathcal{B} & \mathcal{C} \\
\xleftarrow{K_P} & & \xleftarrow{K_P} \\
\xrightarrow{m} & & \\
\xleftarrow{c} & c \leftarrow \mathsf{simcommit}(m) & \\
\xrightarrow{\hat{R}_B} & \text{pick } R_A & \\
\xleftarrow{d} & d \leftarrow \mathsf{equivocate}(m,c,R_A) & \\
\xrightarrow{\hat{m}||\hat{c}} & & \xrightarrow{\hat{m}||\hat{c}} \\
\xleftarrow{R_B} & R_B \leftarrow R_A \oplus \hat{R}_A \oplus \hat{R}_B & \xleftarrow{\hat{R}_A} \\
\xrightarrow{\hat{d}} & & \xrightarrow{\hat{d}}
\end{array}
$$

Case 1

$$
\begin{array}{lcr}
\mathcal{A} & \mathcal{B} & \mathcal{C} \\
\xleftarrow{K_P} & & \xleftarrow{K_P} \\
\cdots & & \\
\xrightarrow{m} & & \\
\xleftarrow{c} & c \leftarrow \mathsf{simcommit}(m) & \\
\cdots & & \\
\xrightarrow{\hat{m}||\hat{c}} & & \xrightarrow{\hat{m}||\hat{c}} \\
\xleftarrow{R_B} & \text{pick } R_B & \xleftarrow{\hat{R}_A} \\
\cdots & & \\
\xrightarrow{\hat{R}_B} & R_A \leftarrow \hat{R}_A \oplus R_B \oplus \hat{R}_B & \\
\xleftarrow{d} & d \leftarrow \mathsf{equivocate}(m,c,R_A) & \\
\cdots & & \\
\xrightarrow{\hat{d}} & & \xrightarrow{\hat{d}}
\end{array}
$$

Case 2

Fig. 7. Reduction with Equivocable Commitments.

Equivocable commitments. From $\mathcal{A}$ we construct an adversary $\mathcal{B}$ who plays the binding game with the help of one query to the $\mathsf{simcommit}$ and $\mathsf{equivocate}$ oracles. (See Fig. 7.) Namely, $\mathcal{B}$ runs $\mathcal{A}$ and first forwards K_P. Whenever $\mathcal{B}$ receives $\hat{m}$ and $\hat{c}$ from $\mathcal{A}$, he submits it in the binding game. Whenever $\mathcal{B}$ must send c to $\mathcal{A}$, he launches the $\mathsf{simcommit}$ oracle to get c. When $\mathcal{A}$ sends $\hat{R}_B$ to Bob, we distinguish two cases.

Case 1. If $\mathcal{A}$ did not send $\hat{m}||\hat{c}$ to Bob yet, there is essentially one way to interleave the two sequences. Here, $\mathcal{B}$ picks a random R_A and equivocate his commitment by calling $\mathsf{equivocate}(m,c,R_A)$ so that he can send d to $\mathcal{A}$. When receiving the challenge $\hat{R}_A$, $\mathcal{B}$ chooses $R_B = R_A \oplus \hat{R}_A \oplus \hat{R}_B$ and send it to $\mathcal{A}$.

Case 2. If now $\mathcal{A}$ has sent $\hat{m}||\hat{c}$ to Bob, $\mathcal{B}$ has already answered some random R_B to $\mathcal{A}$ and received some challenge $\hat{R}_A$. He can thus compute $R_A = \hat{R}_A \oplus R_B \oplus \hat{R}_B$ and equivocate his commitment by calling $\mathsf{equivocate}(m,c,R_A)$. A typical example is depicted on Fig. 6.

Here, $\mathcal{A}$ and $\mathcal{B}$ win or loose at the same time, thus win with the same probability so we conclude as for the extractable commitments. □

5 On the Selection of the SAS Length

We now study the security in a multiparty and concurrent setting.

Lemma 6. *We consider a message authentication protocol with claimant Alice and verifier Bob in which a single SAS is sent. We denote by μ_A (resp. μ_B) the complexity of Alice's (resp. Bob's) part. We consider adversaries such that the number of instances of Alice (resp. Bob) is at most Q_A (resp. Q_B). We further denote by T_0 and p_0 their time complexity and probability of success, respectively. There is a generic transformation which, for any Q_A, Q_B, and any adversary, transforms it into a one-shot adversary with complexity $T \leq T_0 + \mu_A Q_A + \mu_B Q_B$ and probability of success $p \geq p_0/Q_A Q_B$.*

The lemma tells us that once we proved that a protocol resists one-shot adversaries up to a probability of success of p, then it resists to adversaries up to a probability of success which is close to $Q_A Q_B p$. With the protocol in Fig. 5, this probability is basically $Q_A Q_B.2^{-k}$. This bound is tight as shown by the attacks in Section 4.

Proof. Let us consider an adversary $\mathcal{A}$. We number every instance of Alice and every instance of Bob by using two separate counters. We say that an instance π_a of Alice is compatible with an instance π_b of Bob if π_b succeeded and received an authenticated message which was sent by π_a. The number of possible compatible pairs of instances is upper bounded by $K = Q_A Q_B$. When an attack is successful, it yields a random pair (I, J) of compatible instances of Alice and Bob.

We transform $\mathcal{A}$ into a one-shot adversary $\mathcal{B}$ as follows: we run $\mathcal{A}$ and simulate launch and send oracle calls. We pick a random pair (I^*, J^*) with uniformly distributed $I^* \in \{1, \ldots, Q_A\}$ and $J^* \in \{1, \ldots, Q_B\}$. When $\mathcal{A}$ queries $\mathsf{launch}(\cdot, \text{Alice}, \cdot)$ for the I^*th time, $\mathcal{B}$ forwards the query to the real launch oracle. send queries to the related instance are also forwarded to the real send oracle. The same holds for the J^*th query $\mathsf{launch}(\cdot, \text{Bob}, \cdot)$. Clearly, the attack succeeds with probability $p_0/Q_A Q_B$ on the only non-simulated instances. It runs with complexity $T_0 + \mu_A Q_A + \mu_B Q_B$. □

For applications, we assume that the number of network nodes is $N \approx 2^{20}$, and that the number of protocol runs per node is limited to $R \approx 2^{10}$. Actually, the protocols are not meant to be run so many times: only for seed authentication. Indeed, they can be used to authenticate a public key, and authentication can later be done using the public key itself so the protocol is no longer useful. We target a probability of success limited to $p \approx 2^{-10}$. Using Th. 5 and the previous lemma tells us that we can take $k \geq \log_2 \frac{Q_A Q_B}{p}$.

When considering the probability of success *at large* over the network, i.e. the probability that an attack occurs somewhere in the network, we have the constraint $Q_A + Q_B \leq NR$. Thus we have $Q_A Q_B \leq N^2 R^2/4$ for our message authentication protocol. Thus we can take $k = \log_2 \frac{N^2 R^2}{4p} = 68$ which is already shorter than the solution based on hash functions.

When considering the probability of success *against a target verifier* node, i.e. the probability that a given user will accept a forged message, we take $Q_B \leq R$ as an additional constraint. Thus we have $Q_A Q_B \leq NR^2$ which leads us to $k = \log_2 \frac{NR^2}{p} = 50$. By using the encoding rules of the RFC 1760 [23] standard, this represents five 4-character human-friendly (or at least English) words.

Credit cards ATM use 4-digit PIN codes which are confidential and quite strongly authenticated. Protocols are also limited to three trials. In our settings, this translates into a 3-shot 2-party model: $N = 2$, $R = 3$, and $p = 3 \cdot 10^{-4}$. To reach the same security level with weak authentication and no confidentiality, we need a SAS of size $k = \log_2 \frac{NR^2}{4p} \approx 15$ bits, i.e. a 5-digit PIN code.

6 Conclusion

We have shown how to achieve authentication over an insecure channel by using a narrowband authentication channel. The later channel is used to authenticate a short string: the SAS. Using weak authentication, we can obtain high security level in a multiparty setting by using 50-bit SAS. Note that in a 3-shot 2-party adversarial model, a 15-bit SAS (i.e. a 5-digit PIN code) is enough. This is similar to the MANA I protocol, except that we no longer require a strong notion of authentication. SAS channels are widely available for human beings: they can transmit SAS by fax, voice mail, type them on mobile devices, etc.

Our protocol is well suited to ad-hoc message authentication. It can be used for PKI-less public key transmission or to run a key agreement protocol. It can also be used to restore a secure association in disaster cases when two remote peers have compromised their secret keys or a PKI is badly broken. Another application could be a Bluetooth-like pairing between physically identified wireless devices with higher security: we no longer rely on the secrecy of a PIN code but on the authentication through a human user of a short string.

Our protocol relies on a commitment scheme and is provably secure in the strongest security model so far, namely the Bellare-Rogaway model, by using extractable or equivocable commitment schemes. They can be constructed in the ideal commitment model, in the random oracle commitment model, and in the CRS model.

7 Acknowledgments

I wish to thank Anna Lysyanskaya, Ivan Damgård, Kaisa Nyberg, Moti Yung, Phil MacKenzie, and the anonymous reviewers for insightful references and comments. This paper was highly inspired by [32] which was originally submitted (and rejected!) in 1975, so I would like to thank Ralph Merkle for 30 years of research fun.

This work was supported in part by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

References

1. Digital Signature Standard (DSS). *Federal Information Processing Standards* publication #186-2. U.S. Department of Commerce, National Institute of Standards and Technology, 2000.
2. Specification of the Bluetooth System. Vol. 2: Core System Package. Bluetooth Specification version 1.2, 2003.

3. M. Abdalla, P.-A. Fouque, D. Pointcheval. Password-Based Authenticated Key Exchange in the Three-Party Setting. In *Public Key Cryptography'05*, Les Diablerets, Switzerland, Lecture Notes in Computer Science 3386, pp. 65–84, Springer-Verlag, 2005.
4. D. Balfanz, D. K. Smeeters, P. Stewart, H. Chi Wong. Talking to Strangers: Authentication in Ad-Hoc Wireless Networks. In *Network and Distributed System Security Symposium Conference (NDSS 02)*, San Diego, California, U.S.A., The Internet Society, 2002.
5. M. Bellare, D. Pointcheval, P. Rogaway. Authenticated Key Exchange Secure against Dictionary Attacks. In *Advances in Cryptology EUROCRYPT'00*, Brugge, Belgium, Lecture Notes in Computer Science 1807, pp. 139–155, Springer-Verlag, 2000.
6. M. Bellare, P. Rogaway. Entity Authentication and Key Distribution. In *Advances in Cryptology CRYPTO'93*, Santa Barbara, California, U.S.A., Lecture Notes in Computer Science 773, pp. 232–249, Springer-Verlag, 1994.
7. M. Bellare, P. Rogaway. Provably Secure Session Key Distribution: the Three Party Case. In *Proceedings of the 27th ACM Symposium on Theory of Computing*, Las Vegas, Nevada, U.S.A., pp. 57–66, ACM Press, 1995.
8. S. M. Bellovin, M. Merritt. Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks. In *IEEE symposium on Research in Security and Privacy*, Oakland, California, USA, pp. IEEE Computer Society Press, 72–84, 1992.
9. S. M. Bellovin, M. Merritt. Augmented Encrypted Key Exchange. In *1st ACM Conference on Computer and Communications Security*, Fairfax, Virginia, U.S.A., pp. 244–250, ACM Press, 1993.
10. C. H. Bennett, G. Brassard. Quantum Cryptography: Public Key Distribution and Coin Tossing. In *Proc. IEEE International Conference on Computers, Systems, and Signal Processing*, Bangalore, India, pp. 175–179, IEEE Press, 1984.
11. D. Boneh. The Decision Diffie-Hellman Problem. In *Proceedings of the 3rd Algorithmic Number Theory Symposium*, Portland, Oregon, U.S.A., Lecture Notes in Computer Science 1423, pp. 48–63, Springer-Verlag, 1998.
12. C. Boyd, A. Mathuria. *Protocols for Authentication and Key Establishment*, Information Security and Cryptography, Springer Verlag, 2003.
13. M. Čagalj, S. Čapkun, J.-P. Hubaux. Key Agreement in Peer-to-Peer Wireless Networks. To appear in the Proceedings of the IEEE, fall 2005.
14. R. Cramer, V. Shoup. Signature Schemes based on the Strong RSA Assumption. *ACM Transactions on Information and System Security*, vol. 3, pp. 161–185, 2000.
15. G. Di Crescenzo, Y. Ishai, R. Ostrovsky. Non-Interactive and Non-Malleable Commitment. In *Proceedings of the 30th ACM Symposium on Theory of Computing*, Dallas, Texas, U.S.A., pp. 141–150, ACM Press, 1998.
16. G. Di Crescenzo, J. Katz, R. Ostrovsky, A. Smith. Efficient and Non-Interactive Non-Malleable Commitments. In *Advances in Cryptology EUROCRYPT'01*, Innsbruck, Austria, Lecture Notes in Computer Science 2045, pp. 40–59, Springer-Verlag, 2001.
17. I. Damgård, J. Groth. Non-interactive and Reusable Non-malleable Commitment Schemes. In *Proceedings of the 35th ACM Symposium on Theory of Computing*, San Diego, California, U.S.A., pp. 426–437, ACM Press, 2003.
18. W. Diffie, M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, vol. IT-22, pp. 644–654, 1976.
19. T. Dierks, C. Allen. The TLS Protocol Version 1.0. RFC 2246, standard tracks, the Internet Society, 1999.
20. D. Dolev, C. Dwork, M. Naor. Non-Malleable Cryptography. *SIAM Journal of Computing*, vol. 30, pp. 391–437, 2000.
21. C. Gehrmann, C. Mitchell, K. Nyberg. Manual Authentication for Wireless Devices. *RSA Cryptobytes*, vol. 7, pp. 29–37, 2004.

22. C. Gehrmann, K. Nyberg. Security in Personal Area Networks. In *Security for Mobility*, C. Mitchell (Ed.), pp. 191–230, IEE, 2004.
23. N. Haller. The S/KEY One-Time Password System. RFC 1760, 1995.
24. J. Håstad, R. Impagliazzo, L. Levin, M. Luby. A Pseudorandom Generator from any One-way Function. *SIAM Journal on Computing*, vol. 4, pp. 1364–1396, 1999.
25. J.-H. Hoepman. The Ephemeral Pairing Problem. In *Financial Cryptography*, Key West, Florida, USA, Lecture Notes in Computer Science 3110, pp. 212–226, Springer-Verlag, 2004.
26. J.-H. Hoepman. Ephemeral Pairing on Anonymous Networks. In *Proceedings of the Second International Conference on Security in Pervasive Computing (SPC'05)*, Boppard, Germany, Lecture Notes in Computer Science 3450, pp. 101–116, Springer-Verlag, 2005.
27. R. Housley, W. Ford, W. Polk, D. Solo. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. Internet Standard. RFC 2459, The Internet Society, 1999.
28. M. Jakobsson, S. Wetzel. Security Weaknesses in Bluetooth. In *Topics in Cryptology (CT–RSA'01)*, San Francisco, California, USA, Lecture Notes in Computer Science 2020, pp. 176–191, Springer-Verlag, 2001.
29. J. Katz, R. Ostrovsky, M. Yung. Efficient Password-Authenticated Key Exchange using Human-Memorable Passwords. In *Advances in Cryptology EUROCRYPT'01*, Innsbruck, Austria, Lecture Notes in Computer Science 2045, pp. 475–494, Springer-Verlag, 2001.
30. J. Kohl, C. Neuman. The Kerberos Network Authentication Service (V5). Internet standard. RFC 1510, 1993.
31. P. MacKenzie, K. Yang. On Simulation-Sound Trapdoor Commitments. In *Advances in Cryptology EUROCRYPT'04*, Interlaken, Switzerland, Lecture Notes in Computer Science 3027, pp. 382–400, Springer-Verlag, 2004.
32. R. C. Merkle. Secure Communications over Insecure Channels. *Communications of the ACM*, vol. 21, pp. 294–299, 1978.
33. C. Mitchell, M. Ward, P. Wilson. On Key Control in Key Agreement Protocols. *Electronics Letters*, vol. 34, pp. 980–981, 1998.
34. R. M. Needham, M. D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM*, vol. 21, pp. 993–999, 1978.
35. K. Nyberg. IKE in Ad-hoc IP Networking. In *Security in Ad-hoc and Sensor Networks, 1st European Workshop (ESAS'04)*, Heidelberg, Germany, Lecture Notes in Computer Science 3313, pp. 139–151, Springer-Verlag, 2005.
36. S. Pasini, S. Vaudenay. Optimized Message Authentication Protocols. Unpublished.
37. R. Pass. On Deniability in the Common Reference String and Random Oracle Model. In *Advances in Cryptology CRYPTO'03*, Santa Barbara, California, U.S.A., Lecture Notes in Computer Science 2729, pp. 316–337, Springer-Verlag, 2003.
38. T. Peyrin, S. Vaudenay. The Pairing Problem with User Interaction. In *Security and Privacy in the Age of Ubiquitous Computing IFIP TC11 20th International Information Security Conference (SEC'05)*, Chiba, Japan, pp. 251–265, Springer-Verlag, 2005.
39. R. L. Rivest, A. Shamir and L. M. Adleman. A Method for Obtaining Digital Signatures and Public-key Cryptosystem. *Communications of the ACM*, vol. 21, pp. 120–126, 1978.
40. F. Stajano, R. Anderson. The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In *Proceedings of the 7th International Workshop on Security Protocols*, Cambridge, United Kingdom, Lecture Notes in Computer Science 1796, pp. 172–194, Springer-Verlag, 1999.

A Message Cross-Authentication

We consider protocols which perform message authentication in both directions at the same time. These protocols have inputs m_A on the side of Alice and m_B on the side of Bob, and outputs $I_B||\hat{m}_B$ on the side of Alice and $I_A||\hat{m}_A$ on the side of Bob. They should be such that they achieve message authentication in both directions. In Fig. 8 is a message cross-authentication protocol. It requires k authenticated bits in both ways. Obviously, an adversary against this protocol transforms into an adversary against the message authentication protocol in Fig. 5. So Theorem 5 holds for this new protocol as well. This protocol can be used e.g. to run the Diffie-Hellman authenticated key agreement protocol [18] with $m_A = g^{x_A}$ and $m_B = g^{x_B}$. This is essentially the protocol called DH-SC in [13].

$$
\begin{array}{rcl}
\text{Alice} & & \text{Bob} \\
\textbf{input}\text{: } m_A & & \textbf{input}\text{: } m_B \\
\text{pick } R_A \in_U \{0,1\}^k & & \text{pick } R_B \in_U \{0,1\}^k \\
c_A||d_A \leftarrow \mathsf{commit}(0||m_A||R_A) & \xrightarrow{m_A||c_A} & \\
 & \xleftarrow{m_B||c_B} & c_B||d_B \leftarrow \mathsf{commit}(1||m_B||R_B) \\
 & \xrightarrow{d_A} & \hat{R}_A \leftarrow \mathsf{open}(0||\hat{m}_A, \hat{c}_A, \hat{d}_A) \\
\hat{R}_B \leftarrow \mathsf{open}(1||\hat{m}_B, \hat{c}_B, \hat{d}_B) & \xleftarrow{d_B} & \\
\mathsf{SAS}_A \leftarrow R_A \oplus \hat{R}_B & \xrightarrow{\mathsf{authenticate}_{\mathrm{Alice}}(\mathsf{SAS}_A)} & \mathsf{SAS}_B \leftarrow \hat{R}_A \oplus R_B \\
\text{check } \mathsf{SAS}_A = \mathsf{SAS}_B & \xleftarrow{\mathsf{authenticate}_{\mathrm{Bob}}(\mathsf{SAS}_B)} & \text{check } \mathsf{SAS}_A = \mathsf{SAS}_B \\
\textbf{output}\text{: Bob, } \hat{m}_B & & \textbf{output}\text{: Alice, } \hat{m}_A
\end{array}
$$

Fig. 8. SAS-based Cross Authentication.

B Using Simulation-Sound Trapdoor Commitments

MacKenzie-Yang [31] defines SSTC commitments by five algorithms setup', commit', verify', $\mathsf{simcommit}'$, and $\mathsf{equivocate}'$. The only syntaxic difference with our definition is in the verify' algorithm which replaces open, but without message recovery. Namely, $\mathsf{commit}'(m,r)$ yields a pair (c,e) and $\mathsf{verify}'(m,c,r,e) = \mathsf{true}$ whenever (c,e) is a possible output of $\mathsf{commit}'(m,r)$. Obviously, by letting $d = r||e$, we define a commitment scheme in our sense.

In [31], the hiding game is restricted to a 2-fold game. Namely, the adversary yields r_0 and r_1 together with c, and the challenger picks r equal to one of these two values. The adversary should have a probability of success less than $\frac{1}{2}+\varepsilon$. The following lemma proves that such a commitment scheme is 2ε-hiding in our sense.

Lemma 7. *There exists a (small) constant ν such that for any T and ε, a $(T+\nu,\varepsilon)$-2-fold-hiding commitment scheme is a $(T,2\varepsilon)$-hiding commitment scheme.*

Proof. Let $\mathcal{A}$ be an adversary of complexity at most T which plays our 2^k-fold hiding game. We construct an adversary $\mathcal{B}$ for the 2-fold hiding game as follows.

1. $\mathcal{B}$ receives K_P and forwards it to $\mathcal{A}$.

2. $\mathcal{A}$ sends m to $\mathcal{B}$.
3. $\mathcal{B}$ picks two random different r_0 and r_1 and plays m, r_0, r_1.
4. $\mathcal{B}$ receives a challenge c which commits to either r_0 or r_1 and forwards it to $\mathcal{A}$.
5. $\mathcal{A}$ answers to the challenge by a string r.
6. If $r = r_b$, $\mathcal{B}$ answers b. Otherwise, $\mathcal{B}$ picks a random bit b and answers b.

We let ν be the complexity of $\mathcal{B}$ without $\mathcal{A}$. Obviously, $\mathcal{B}$ perfectly simulates a challenger for the 2^k-fold game to $\mathcal{A}$. Let $p' = 2^{-k} + \varepsilon'$ be the probability of success of $\mathcal{A}$. When $\mathcal{A}$ is successful, so is $\mathcal{B}$. When $\mathcal{A}$ is not successful and $r \notin \{r_0, r_1\}$, $\mathcal{B}$ succeeds with probability $\frac{1}{2}$. When $\mathcal{A}$ is not successful and $r \in \{r_0, r_1\}$, $\mathcal{B}$ fails. Hence, the probability that $\mathcal{B}$ answers correctly to the 2-fold game is

$$\begin{aligned} p &= p' + \frac{1-p'}{2}\left(1 - \frac{1}{2^k - 1}\right) \\ &= \frac{1}{2} + p' \frac{2^k}{2(2^k-1)} - \frac{1}{2(2^k-1)} \\ &= \frac{1}{2} + \varepsilon' \frac{2^k}{2(2^k-1)} \end{aligned}$$

Since we must have $p - \frac{1}{2} \leq \varepsilon$, we deduce $\varepsilon' \leq 2\varepsilon(1 - 2^{-k}) \leq 2\varepsilon$. □

In [31], our binding game is replaced by the ability to produce a collision, namely a (m, c, d, d') quadruplet such that (m, c, d) and (m, c, d') successfully open to two different values. Following the simulation-sound binding property definition, the adversary has access to $\mathsf{simcommit}'$ and $\mathsf{equivocate}'$ like in our definition. Namely, they do not see ξ and cannot decommit values which are not issued by $\mathsf{simcommit}'$. By usual rewinding techniques, we show that from an adversary who wins our binding property with probability $2^{-k} + \varepsilon$ in time T and one $\mathsf{simcommit}$ query we can make an adversary who finds collisions with probability $2^{-2k}\varepsilon$ in time $2T$ and two $\mathsf{simcommit}$ queries.