A NEW MESSAGE RECOGNITION PROTOCOL FOR AD HOC PERVASIVE NETWORKS

ATEFEH MASHATAN[1] AND DOUGLAS R. STINSON[2]

Abstract. We propose a message recognition protocol which is suitable for ad hoc pervasive networks without the use of hash chains. Hence, we no longer require the sensor motes to save values of a hash chain in their memories. This relaxes the memory requirements. Moreover, we do not need to fix the total number of times the protocol can be executed which implies a desired flexibility in this regard. Furthermore, our protocol is secure without having to consider families of assumptions that depend on the number of sessions the protocol is executed. Hence, the security does not weaken as the protocol is executed over time. Last but not least, we provide a practical procedure for resynchronization in case of any adversarial disruption or communication failure.

1. Introduction

In this paper, we examine the notion of *message recognition* which is a weaker notion compared to message authentication. It refers to the process where two parties initially meet in an authenticated setting, and later, one party is assured that the received messages over the insecure channel is sent from the same second party. In other words, message recognition provides data integrity with respect to the source of the message. Standard approaches of public-key and secret-key cryptography provide many proposals for message authentication or recognition, digital signature schemes for instance. However, these techniques either require enough computational power to handle public-key operations, or they assume the existence of a shared secret. These assumptions may not be reasonable for some constraint scenarios such as ad hoc pervasive networks. Moreover, assuming a trusted third party in ad hoc networks settings is often undesirable. Indeed, constraint devices such as sensor network motes or RFID tags spread over a hostile environment have limited computational power and it maybe unrealistic to assume the availability of confidential channel to pre-authenticate some data, such as shared secret-key.

Let Alice and Bob be small devices, such as sensor motes, to be spread in a hostile environment. They once meet in an authenticated setting. Later, Alice sends messages to Bob and she wants Bob to recognize the messages that are sent from her. The adversary, Eve, is present all the time and would like to make Bob recognize a message sent from Eve as a message from Alice. Eve has the

Date: July 2, 2008.

[1] Department of Combinatorics and Optimization
`amashatan@uwaterloo.ca`
[2] David R. Cheriton School of Computer Science
`dstinson@uwaterloo.ca`
University of Waterloo
Waterloo, Ontario CANADA N2L 3G1
.

ability to choose messages and make Alice send them to Bob. Note that Eve wins if Bob recognizes a message that Alice has never sent to be a message from Alice.

The common approach in the literature is to assume the availability of two channels: an authenticated narrow-band channel for the initialization step and the usual insecure broadband channel. The authenticated channel is denoted by $\Rightarrow$ and the insecure channel by $\rightarrow$. The adversary is assumed to have full control over the messages sent over the insecure channel. She can listen to these messages, modify them or stops any flow over this channel. She can also insert a new message of her choice over this channel. On the other hand, she can only read the messages sent over the authenticated channel, but cannot modify them or stall them from delivery.

Recently, there has been a lot of activities in this area of research to design message recognition protocols. Assuming availability of a time-stamping service or the use of signatures, [1] proposes a message recognition protocol. Assuming a rather heavy amount of computation and communication, [6] proposes a recognition protocol called 'Remote User Authentication Protocol'. Using a hash chaining technique, [5] and [7] proposed an interactive recognition protocol where the number n is fixed to be the maximum number of messages to be authenticated. Each pair of users willing to communicate, must have a separate pair of hash chain which puts a relatively heavy memory requirement on the sensor motes. Furthermore, the security assumptions for this protocol depend on the number of sessions the protocol has been executed which gives birth to the notion of "depth-i security". We briefly summarize the existing results and discuss why they need to be improved.

We further propose a new design for message recognition in ad hoc pervasive networks and explain the advantages of using this new design compared to previous alternatives. Our proposed recognition protocol does not make use of hash chains. As a result, we no longer require the sensor motes to save values of a hash chain in their memories. This relaxes the memory requirements. Moreover, the passwords are set to be chosen at random in each session. Hence, they are independent of one another and are being refreshed in each session. This can be done for any arbitrary number of times, so we do not need to fix the total number of times the protocol can be executed.

As the passwords corresponding to each session are chosen at random and are independent of one another, we do not need to consider assumptions that depend on the number of sessions the protocol is executed. Consequently, the security does not weaken as the protocol is executed over time. We *commit* to a password by sending its hash value, so that Eve cannot change it. Further, we need to *bind* two consecutive passwords, in order to detect adversarial intrusions and to be able to resynchronize in such a case. Last but not least, we provide a practical procedure for resynchronization in case of any possible adversarial disruption or communication failure.

Section 2 summarizes the existing results on message recognition protocols. In Section 3, we propose a new message recognition protocol and its resynchronization technique followed by a discussion about the advantages of using this protocol. Section 5 is devoted to proof the security of this protocol based in the assumptions listed in Section 4.

2. Literature review on Message Recognition Protocol

Here, we briefly examine the existing recognition protocols and discuss their performance and practicality in ad hoc pervasive settings where communication bandwidth, computational power, and memory capacity are rather low.

Anderson et. al. proposed the Guy Fawkes protocol in [1]. The first variant of this protocol assumes that a time-stamping service is available for every session. The second variant avoids this assumption by using a signature in the initialization step. Moreover, it requires that users commit to messages one session a head of time. That is, users are supposed to perform two sessions to authenticate a single message.

The 'Remote User Authentication Protocol' was proposed in [6]. It uses a message authentication code (MAC) and requires that users compute a lot of MAC values. The MAC values are sent over the authenticated channel. This is a concern in our setting since the authenticated channels usually have low bandwidth. Moreover, the amount of computations and communication assumed in this protocol may not be desirable in a pervasive network with devices with low computational power.

The 'Zero Common-Knowledge (ZCK)' protocol was proposed in [7]. They use the values of a hash chain as keys for a MAC. This protocol was implemented in [4] as a proof-of-concept. The observations from this implementation ensured that this protocol is suitable for devices with low computational power, low code space, low communication bandwidth and low energy resources. It also raised a couple of areas of concern, mainly denial-of-service and memory complexity.

Note that [4] explored the practicality of the ZCK protocol and not its security proof. Later, [5] found a problem in the security proof of this protocol and proposed a variant to fix the problem. Similar to the original ZCK protocol, they form a hash chain by fixing the number n to be the maximum number of messages to be authenticated. Alice and Bob randomly choose a_0 and b_0, respectively. Then, they respectively form hash chains $a_i = h(a_{i-1})$ and $b_i = h(b_{i-1})$, $i = 1, \ldots, n$. Note that for each pair of users willing to communicate, there must be a separate pair of hash chains. This means that if a sensor mote wants to communicate with m users, it has to deal with m different hash chains of length n. This is of concern when dealing with sensor motes in a pervasive network with memory constraints, also noted by [4].

Furthermore, [5] has to consider security assumptions that depend on the number of sessions the protocol has been executed in order to prove the security of their protocol in the standard model. In particular, they have to treat the first session separately and then deal with the security of session i inductively. In other words, they prove "depth-i security" of their protocol based on assumptions such as "depth-i non-invertability", "depth-i Second Preimage Resistance", "depth-i unforgability", and "depth-i combined security". In other words, they are with families of assumptions that should hold for each i, $1 \leq i \leq n$. As they note in their paper, one could argue that, as the number of times the protocol is executed, its security weakens.

It is of interest to design a message recognition protocol that avoids assumptions that depend on the number of sessions the protocol has been executed and do not require the devices to have a high enough memory capacity to save several hash chains. Next, we describe a new message recognition protocol with such properties.

3. A New Message Recognition Protocol Without the Use of Hash Chains

In this section, we describe the details of our proposed protocol. The initialization phase, execution of the protocol, and the resynchronization process are separately described. The section is concluded by examining the advantages of using this protocol in comparison to previous designs.

We begin by describing the internal states of Alice and Bob. Internal state of Alice includes:

- x_0 and x_1: the *passwords* for this session and the next session, respectively.
- $X_0 = H(x_0)$ and $X_1 = H(x_1)$: the *committing hash values* of the passwords.
- $\mathcal{X}_0 = H(x_0, X_1) = H(x_0, H(x_1))$: the *binding hash value* of the passwords.
- y^*_{-1}, Y^*_0, $\mathcal{Y}^*_0$: Bob's most recent password, committing hash value, and binding hash value accepted by Alice.

Similarly, internal state of Bob includes:

- y_0 and y_1: the *passwords* for this session and the next session, respectively.
- $Y_0 = H(y_0)$ and $Y_1 = H(y_1)$: the *committing hash values* of the passwords.
- $\mathcal{Y}_0 = H(y_0, Y_1) = H(y_0, H(y_1))$: the *binding hash value* of the passwords.
- x^*_{-1}, X^*_0, $\mathcal{X}^*_0$: Alice's most recent password, committing hash value, and binding hash value accepted by Bob.

In this protocol, x_0 and y_0 are considered to be passwords of the current session. Similarly, x_1 and y_1 are the passwords of the next session. We *commit* to a password by sending its hash value, so that Eve cannot change it. Further, we *bind* two consecutive passwords, in order to detect adversarial intrusions and to be able to resynchronize in such a case.

Alice performs the initialization phase as follows:

- Choose random x_0 and x_1.
- Compute $X_0 := H(x_0), X_1 := H(x_1)$, and $\mathcal{X}_0 := H(x_0, X_1)$.
- Send $X_0, \mathcal{X}_0$ to Bob over the authenticated channel.
- Receive $Y_0, \mathcal{Y}_0$ from Bob over the authenticated channel.
- Let $y^*_{-1} := \perp, Y^*_0 = Y_0$, and $\mathcal{Y}^*_0 = \mathcal{Y}_0$.

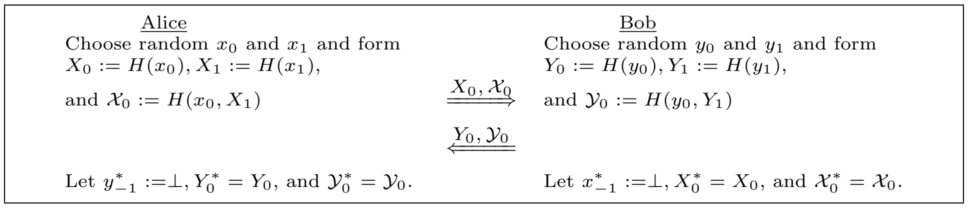
Similarly, Bob executes the initialization phase according to the following steps:

- Choose random y_0 and y_1.
- Compute $Y_0 := H(y_0), Y_1 := H(y_1)$, and $\mathcal{Y}_0 := H(y_0, Y_1)$.
- Receive $X_0, \mathcal{X}_0$ from Alice over the authenticated channel.
- Send $Y_0, \mathcal{Y}_0$ to Alice over the authenticated channel.
- Let $x^*_{-1} := \perp, X^*_0 = X_0$, and $\mathcal{X}^*_0 = \mathcal{X}_0$.

The initialization phase of the protocol is depicted in Figure 1. Next, we move on to the description of the proposed message recognition protocol illustrated in Figure 2.

On input (m, Bob), Alice's execution can be described as follows:

- Choose a random x_2.
- Compute $X_2 := H(x_2), \mathcal{X}_1 := H(x_1, X_2)$, and $h = H[m, x_0]$.
- Send m, h to Bob and wait to receive $y'_0, Y'_1, \mathcal{Y}'_1$ from Bob. Resend m, h if Bob did not respond.


Alice

Choose random x_0 and x_1 and form
$X_0 := H(x_0), X_1 := H(x_1)$,
and $\mathcal{X}_0 := H(x_0, X_1)$

$\xrightarrow{X_0, \mathcal{X}_0}$

$\xleftarrow{Y_0, \mathcal{Y}_0}$

Let $y^*_{-1} :=\perp, Y^*_0 = Y_0$, and $\mathcal{Y}^*_0 = \mathcal{Y}_0$.

Bob

Choose random y_0 and y_1 and form
$Y_0 := H(y_0), Y_1 := H(y_1)$,
and $\mathcal{Y}_0 := H(y_0, Y_1)$

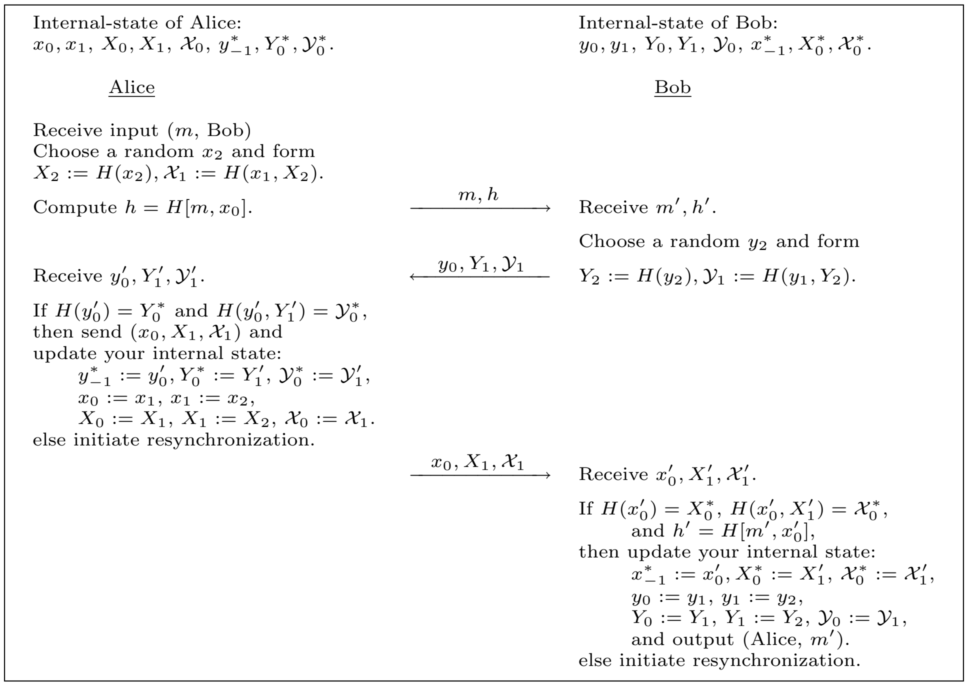
Let $x^*_{-1} :=\perp, X^*_0 = X_0$, and $\mathcal{X}^*_0 = \mathcal{X}_0$.

FIGURE 1. Initialization Phase of the New Message Recognition Protocol

- If $H(y'_0) = Y^*_0$ and $H(y'_0, Y'_1) = \mathcal{Y}^*_0$, then send $(x_0, X_1, \mathcal{X}_1)$ to Bob and update your internal state: $y^*_{-1} := y'_0$, $Y^*_0 := Y'_1$, $\mathcal{Y}^*_0 := \mathcal{Y}'_1$, $x_0 := x_1$, $x_1 := x_2$, $X_0 := X_1$, $X_1 := X_2$, and $\mathcal{X}_0 := \mathcal{X}_1$. Otherwise, initiate resynchronization with Bob.

Bob, on the other hand executes the protocol in the following manner:

- After receiving m', h', choose a random y_2.
- Compute $Y_2 := H(y_2)$ and $\mathcal{Y}_1 := H(y_1, Y_2)$.
- Send $y_0, Y_1, \mathcal{Y}_1$ to Alice and wait to receive $x'_0, X'_1, \mathcal{X}'_1$. Resend $y_0, Y_1, \mathcal{Y}_1$ to Alice if Alice did not respond.
- If $H(x'_0) = X^*_0$ and $H(x'_0, X'_1) = \mathcal{X}^*_0$, and $h' = H[m', x'_0]$, then update your internal state: $x^*_{-1} := x'_0$, $X^*_0 := X'_1$, $\mathcal{X}^*_0 := \mathcal{X}'_1$, $y_0 := y_1$, $y_1 := y_2$, $Y_0 := Y_1$, $Y_1 := Y_2$, and $\mathcal{Y}_0 := \mathcal{Y}_1$, and output (Alice, m'). Otherwise, initiate resynchronization with Alice.


Internal-state of Alice:
$x_0, x_1, X_0, X_1, \mathcal{X}_0, y^*_{-1}, Y^*_0, \mathcal{Y}^*_0$.

Internal-state of Bob:
$y_0, y_1, Y_0, Y_1, \mathcal{Y}_0, x^*_{-1}, X^*_0, \mathcal{X}^*_0$.

Alice

Receive input (m, Bob)
Choose a random x_2 and form
$X_2 := H(x_2), \mathcal{X}_1 := H(x_1, X_2)$.
Compute $h = H[m, x_0]$.

$\xrightarrow{m, h}$

Bob

Receive m', h'.
Choose a random y_2 and form
$Y_2 := H(y_2), \mathcal{Y}_1 := H(y_1, Y_2)$.

$\xleftarrow{y_0, Y_1, \mathcal{Y}_1}$

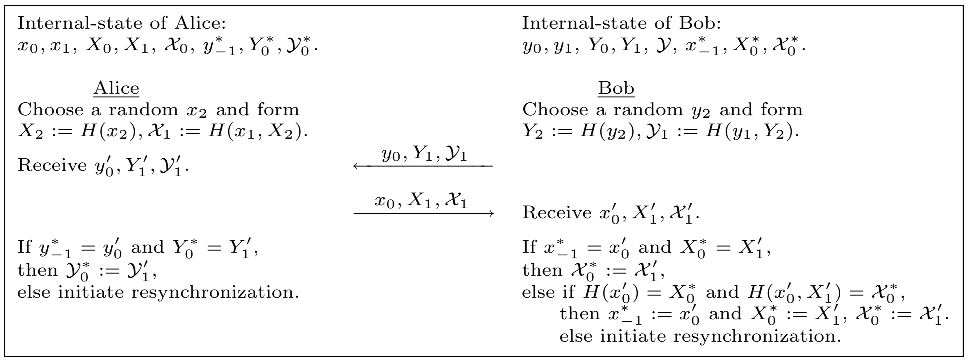
Receive $y'_0, Y'_1, \mathcal{Y}'_1$.
If $H(y'_0) = Y^*_0$ and $H(y'_0, Y'_1) = \mathcal{Y}^*_0$,
then send $(x_0, X_1, \mathcal{X}_1)$ and
update your internal state:
    $y^*_{-1} := y'_0, Y^*_0 := Y'_1, \mathcal{Y}^*_0 := \mathcal{Y}'_1$,
    $x_0 := x_1$, $x_1 := x_2$,
    $X_0 := X_1$, $X_1 := X_2$, $\mathcal{X}_0 := \mathcal{X}_1$.
else initiate resynchronization.

$\xrightarrow{x_0, X_1, \mathcal{X}_1}$

Receive $x'_0, X'_1, \mathcal{X}'_1$.
If $H(x'_0) = X^*_0$, $H(x'_0, X'_1) = \mathcal{X}^*_0$,
    and $h' = H[m', x'_0]$,
then update your internal state:
    $x^*_{-1} := x'_0, X^*_0 := X'_1, \mathcal{X}^*_0 := \mathcal{X}'_1$,
    $y_0 := y_1$, $y_1 := y_2$,
    $Y_0 := Y_1$, $Y_1 := Y_2$, $\mathcal{Y}_0 := \mathcal{Y}_1$,
    and output (Alice, m').
else initiate resynchronization.

FIGURE 2. New Message Recognition Protocol

In case of no adversarial intrusion or communication failure, all the conditions verify and Alice and Bob will not initiate a resynchronization process. When they realize that one of the conditions does not hold, they suspect a communication failure or a possible adversarial intrusion. Hence, they need to resynchronize in order to make sure they have the correct commitment and binding hash values. The synchronization process is illustrated in Figure 3. Bob sends $y_0, Y_1, \mathcal{Y}_1$ to Alice and Alice sends $x_0, X_1, \mathcal{X}_1$ to Bob. Note, Alice should already have y_0, Y_1 and is verifying if they match with what she has. Similarly, Bob is verifying if x_0, X_1 match with what he has. However, the values of $\mathcal{X}_1$ and $\mathcal{Y}_1$ are new. It is possible for the adversary to make either Alice or Bob to compute a binding hash value in a bogus session. In that case, the binding hash value is being refreshed. Note that the resynchronization process is not symmetrical. This is due to the fact that Bob may detect an intrusion after Alice has updated her state. In this case, the values $x_0, X_1, \mathcal{X}_1$ that Alice sends over the resynchronization process need to be verified differently.



Internal-state of Alice: $x_0, x_1,\ X_0, X_1,\ \mathcal{X}_0,\ y^*_{-1}, Y^*_0, \mathcal{Y}^*_0$.		Internal-state of Bob: $y_0, y_1,\ Y_0, Y_1,\ \mathcal{Y},\ x^*_{-1}, X^*_0, \mathcal{X}^*_0$.
Alice		Bob
Choose a random x_2 and form $X_2 := H(x_2), \mathcal{X}_1 := H(x_1, X_2)$.		Choose a random y_2 and form $Y_2 := H(y_2), \mathcal{Y}_1 := H(y_1, Y_2)$.
Receive $y'_0, Y'_1, \mathcal{Y}'_1$.	$\xleftarrow{y_0, Y_1, \mathcal{Y}_1}$	
	$\xrightarrow{x_0, X_1, \mathcal{X}_1}$	Receive $x'_0, X'_1, \mathcal{X}'_1$.
If $y^*_{-1} = y'_0$ and $Y^*_0 = Y'_1$, then $\mathcal{Y}^*_0 := \mathcal{Y}'_1$, else initiate resynchronization.		If $x^*_{-1} = x'_0$ and $X^*_0 = X'_1$, then $\mathcal{X}^*_0 := \mathcal{X}'_1$, else if $H(x'_0) = X^*_0$ and $H(x'_0, X'_1) = \mathcal{X}^*_0$, then $x^*_{-1} := x'_0$ and $X^*_0 := X'_1$, $\mathcal{X}^*_0 := \mathcal{X}'_1$. else initiate resynchronization.

Figure 3. The Resynchronization Process

Since we are not using a hash chain, the memory requirement on the devices is relaxed. The octuple $(x_0, x_1, X_0, X_1, \mathcal{X}_0, y^*_{-1}, Y^*_0, \mathcal{Y}^*_0)$ is all Alice needs to communicate with Bob and she will need another octuple for each different user. In the previous protocols, the devices had to deal with a hash chain for every single device they wanted to communicate with. Storing all the values of a hash chain, for example $a_0, a_1, \ldots,\ a_n$ is too demanding for low-end devices. On the other hand, storing only the root value of the hash chain, for instance a_0, requires too many computations at each session. The alternative is to employ a time-storage trade-off and store some of the hash values, see for example [2]. Still, there are some storage and computational requirements associated with this implementation. Our proposal for not having to deal with a hash chaining technique avoids any memory or computational requirement of this nature for every session.

Moreover, the passwords are set to be chosen at random in each session. Hence, they are independent of one another and are being refreshed in each session. As a result, we do not need to consider assumptions that depend on the number of sessions the protocol is executed. Consequently, the security does not weaken as the protocol is executed over time.

Furthermore, the devices can run this protocol as many times as they want and the total number of sessions is not fixed. This provides extra flexibility compared to the protocols based on the hash chain technique. Next, we look at the security assumptions relevant for this new protocol.

4. Security Assumptions

In this section, we define new notions of hash function security, mainly **Paired Preimage Resistance, (PPR)**, **Paired Second-Preimage Resistance, (PSPR)**, **Paired Collision Resistance, (PCR)**, **Binding Preimage Resistance, (BPR)** . Each notion is presented as a game between a player Oscar and a Challenger. Note that these assumptions are independent of the number of sessions the protocol has been executed. In other words, in contrast to the approach taken in [5], where they have to assume "depth-i non-invertability", "depth-i Second Preimage Resistance", "depth-i unforgability", and "depth-i combined security" for every i, $1 \leq i \leq n$, we are only assuming four assumptions.

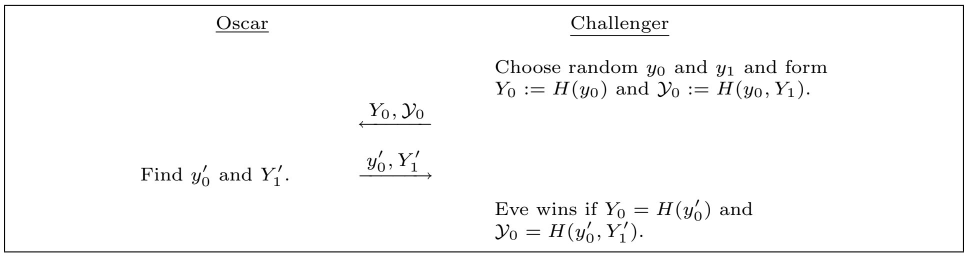

Oscar
Challenger
Choose random y_0 and y_1 and form $Y_0 := H(y_0)$ and $\mathcal{Y}_0 := H(y_0, Y_1)$.
$\xleftarrow{Y_0, \mathcal{Y}_0}$
Find y_0' and Y_1'.
$\xrightarrow{y_0', Y_1'}$
Eve wins if $Y_0 = H(y_0')$ and $\mathcal{Y}_0 = H(y_0', Y_1')$.

Figure 4. Paired Preimage Resistance

Depicted in Figures 4 is the PPR notion. We note that the PPR property is analogous to the notion of "depth-2 non-invertability" defined in [5]. Furthermore, this one assumption is replacing a whole family of assumptions "depth-i non-invertability", for $1 \leq i \leq n$.

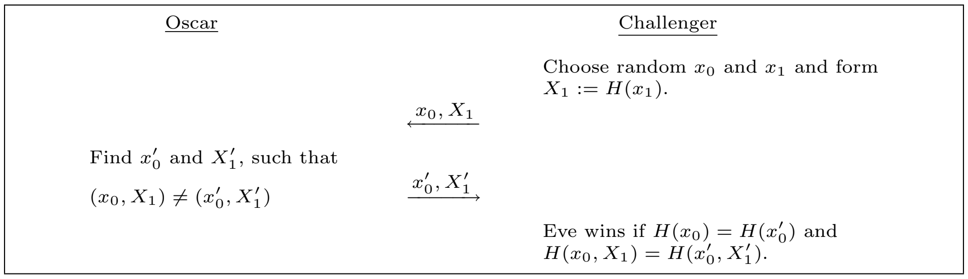

Oscar
Challenger
Choose random x_0 and x_1 and form $X_1 := H(x_1)$.
$\xleftarrow{x_0, X_1}$
Find x_0' and X_1', such that $(x_0, X_1) \neq (x_0', X_1')$
$\xrightarrow{x_0', X_1'}$
Eve wins if $H(x_0) = H(x_0')$ and $H(x_0, X_1) = H(x_0', X_1')$.

Figure 5. Paired Second-Preimage Resistance

Figure 5 illustrates the PSPR notion. This notion is analogous to "depth-2 Second Preimage Resistance" defined in [5]. It is replacing the family of assumptions "depth-i Second Preimage Resistance", for i, $1 \leq i \leq n$.

The notion of PCR is depicted in Figure 6. Analogous to this notion, [5] defines "depth-2 unforgability". Note that the PCR notion is replacing a family of assumptions "depth-i unforgability", for i, $1 \leq i \leq n$.

In Section 5, we will see that the PCR, PPR, and PSPR notions correspond to attacks that start and finish over one session. Moreover, attack scenarios spanning over two sessions are also analyzed and the BPR notion illustrated in Figure 7 is associated to these attacks.

Next, we prove the security of our protocol based on these assumption that PPR, PSPR, PCR, and BPR games are hard to win.

Oscar		Challenger
		Choose random x_0, x_1, x_2 and and form $X_0 := H(x_0)$, $X_1 := H(x_1)$, $X_2 := H(x_2)$, $\mathcal{X}_0 := H(x_0, X_1)$ and $\mathcal{X}_1 := H(x_1, X_2)$.
	$\xleftarrow{X_0, \mathcal{X}_0}$	
	$\xrightarrow{m}$	Compute $h := H(m, x_0)$.
	$\xleftarrow{h}$	
Choose m such that $m \neq m'$.	$\xrightarrow{m', h'}$	Eve wins if $h' = H(m', x_0)$.

Figure 6. Paired Collision Resistance

Oscar		Challenger
		Choose random y_0, y_1, y_2 and and form $Y_0^* := H(y_0)$, $Y_1 := H(y_1)$,$Y_2 := H(y_2)$,
	$\xleftarrow{Y_0^*, \mathcal{Y}_0^*}$	$\mathcal{Y}_0^* := H(y_0, Y_1)$, and $\mathcal{Y}_1 := H(y_1, Y_2)$.
$\mathcal{Y}_0^* \neq (\mathcal{Y}_0^*)'$	$\xrightarrow{(\mathcal{Y}_0^*)'}$	
	$\xleftarrow{y_0, Y_1, \mathcal{Y}_1}$	
	$\xrightarrow{Y_1'}$	Eve wins if $H(y_0, Y_1') = (\mathcal{Y}_0^*)'$.

Figure 7. Binding Preimage Resistance

5. Security of the Proposed Recognition Protocol

Recall that the goal of the adversary is to make Bob accept a message m' that was never sent from Alice. A successful attack is one that Bob is deceived and he outputs (Alice, m').

Let $(x_0, x_1, X_0, X_1, \mathcal{X}_0, y_{-1}^*.Y_0^*, \mathcal{Y}_0^*)$ and $(y_0, y_1, Y_0, Y_1, \mathcal{Y}_0, x_{-1}^*, X_0^*, \mathcal{X}_0^*)$ be the internal states of Alice and Bob, respectively. Now, assume that Eve, having been passive all along, mounts a successful attack for the first time and Bob actually outputs (Alice, m'), where $m \neq m'$. Since, Eve had been passive before this attack, we can assume that $y_{-1}^* = y_0, Y_0^* = H(y_0) = Y_0, \mathcal{Y}_0^* = \mathcal{Y}_0 = H(y_0, H(y_1))$, $x_{-1}^* = x_0, X_0^* = X_0 = H(x_0)$, and $\mathcal{X}_0^* = \mathcal{X}_0 = H(x_0, H(x_1))$. Eve may complete her attack in one session, or she may mount an attack that spans over more than one session. Next, we examine one-round attacks.

5.1. **One-session Attacks.**

In order to exhaustively list all possible one-round attacks against our protocol, we adapt the notation of [3] in labelling different orderings of the flows. This notation labels each flow sent by the adversary by either **A**, if the recipient is Alice, or by **B**, when the recipient is Bob. For example, an ordering of ABAB corresponds to the following attack scenario:

- **A**: Eve sends m to Alice and she responds with m, h.
- **B**: Eve sends m', h' to Bob and he replies with $y_0, Y_1, \mathcal{Y}_1$.
- **A**: Eve sends $y_0', Y_1', \mathcal{Y}_1'$ to Alice and receives $x_0, X_1, \mathcal{X}_1$ from her.
- **B**: Eve sends $x_0', X_1', \mathcal{X}'_1$.

It is proved in [3] that the number of different possible attacks against a three round protocol is $\binom{4}{2} = 6$. These attacks are described, using the above mentioned labelling, to be AABB, ABBA, BABA, ABAB, BBAA, and BAAB. Next, we will analyze each of these attack scenarios.

It can be proved that the BABA attack scenario can be reduced to the ABBA attack. In other words, if the adversary can mount a successful attack of type BABA, then she also succeeds in the ABBA attack scenario. Similarly, one can show that the BAAB and ABBA attack scenarios will be reduced to the ABAB case. These three reductions are described in the Appendix. Hence, it remains to investigate the AABB, BBAA, and ABAB attack scenarios. We prove that the AABB, BBAA, and ABAB attacks are not possible by reducing them to the PPR, PSPR, or PCR games.

5.1.1. *Attack of Type AABB.*

Illustrated in Figure 8 is the attack of type AABB. In this attack scenario, Eve finishes her interactions with with Alice before she starts her interactions with Bob. In other words, Eve has to first deceive Alice in order to get her to reveal the information she needs to then deceive Bob.

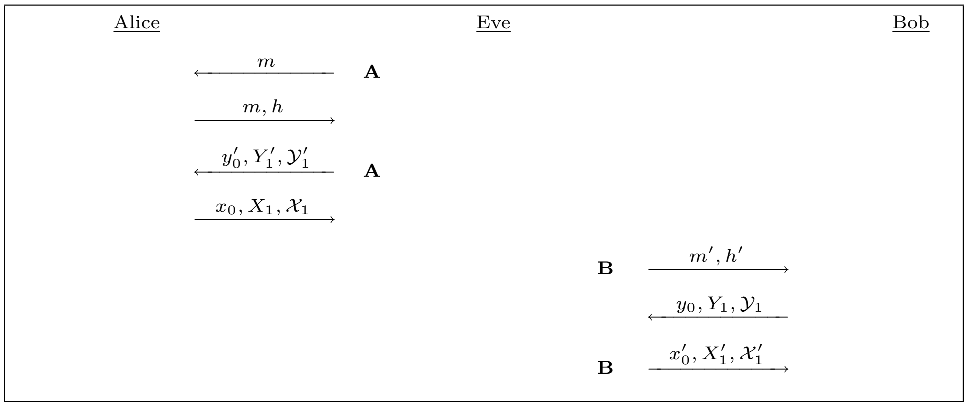

Alice
Eve
Bob
m
A
m, h
$y'_0, Y'_1, \mathcal{Y}'_1$
A
$x_0, X_1, \mathcal{X}_1$
B
m', h'
$y_0, Y_1, \mathcal{Y}_1$
B
$x'_0, X'_1, \mathcal{X}'_1$

FIGURE 8. Attack of Type AABB

If Eve successfully deceives Alice, then she receives $(x_0, X_1, \mathcal{X}_1)$. Now, Eve computes $h' = H(m', x_0)$, for m' of her choice. She then sends m', h' to Bob. Finally, she completes her attack with setting $(x'_0, X'_1, \mathcal{X}'_1) = (x_0, X_1, \mathcal{X}_1)$ and sending it to Bob.

In order to deceive Alice, Eve has to find y'_0 and Y'_1 such that $Y_0 = H(y'_0)$ and $\mathcal{Y}_0 = H(y'_0, Y'_1)$, where Y_0 and $\mathcal{Y}_0$ were transmitted in the session immediately before the attack. Note that Eve, not having seen (y_0, Y_1), has sent (y'_0, Y'_1), which has been accepted by Alice. This is exactly the problem of PPR, depicted in Figure 4.

5.1.2. *Attack of Type BBAA.*

The attack of type BBAA is illustrated in Figure 9. In this scenario, Eve interacts with Alice after she has finished interacting with Bob. That is, she receives $(y_0, Y_1, \mathcal{Y}_1)$ from Bob before she has to choose $(y'_0, Y'_1, \mathcal{Y}'_1)$. If she chooses $(y'_0, Y'_1, \mathcal{Y}'_1)$ such that $(y_0, Y_1) \neq (y'_0, Y'_1)$ and remains undetected by Alice, then, Eve can be reduced to a successful player against the PSPR game of Figure 5. We deal with the case where $(y_0, Y_1) = (y'_0, Y'_1)$ and $\mathcal{Y}_1 \neq \mathcal{Y}'_1$ in Section 5.2. The only remaining case is that, having received $(y_0, Y_1, \mathcal{Y}_1)$ from Bob, Eve lets $(y'_0, Y'_1, \mathcal{Y}'_1) = (y_0, Y_1, \mathcal{Y}_1)$ to avoid being detected by Alice.

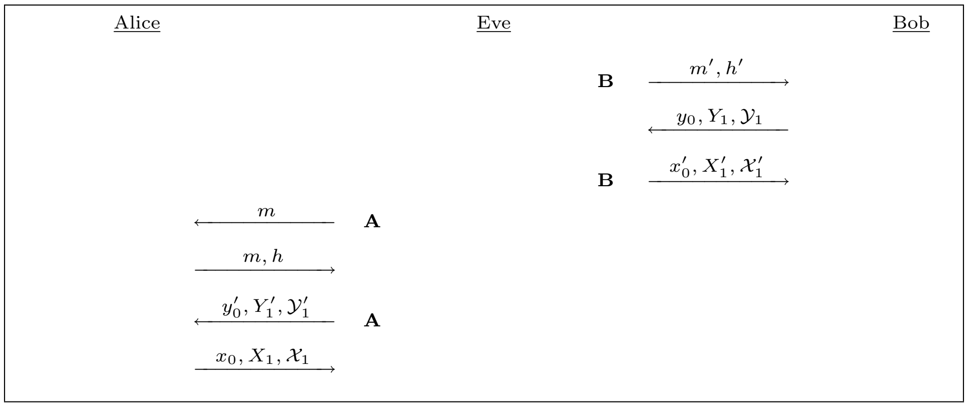

Alice
Eve
Bob
B
m', h'
$y_0, Y_1, \mathcal{Y}_1$
B
$x'_0, X'_1, \mathcal{X}'_1$
m
A
m, h
$y'_0, Y'_1, \mathcal{Y}'_1$
A
$x_0, X_1, \mathcal{X}_1$

FIGURE 9. Attack of Type BBAA

A successful attack of this type implies that Bob has accepted m'. That is, not having seen (x_0, X_1), Eve has found x'_0, X'_1. Once Eve finds the appropriate x'_0 and X'_1, she can compute $h' = H(m', x'_0)$, for an m' of her choice. Note that Eve has received X_0 and $\mathcal{X}_0$ from the previous session. Now, she has to find x'_0, X'_1 such that $X_0 = H(x'_0)$ and $\mathcal{X}_0 = H(x'_0, X'_1)$. This translates to the notion of PPR if we replaces each x value by its corresponding y value.

5.1.3. *Attack of Type ABAB.*

Figure 10 illustrates the attack of type ABAB. In this attack, Eve receives the correct $(y_0, Y_1, \mathcal{Y}_1)$ from Bob before she has to send $(y'_0, Y'_1, \mathcal{Y}'_1)$ to Alice. As it was discussed in the case of the BBAA attack, Eve will be detected by Alice unless she sets $(y'_0, Y'_1, \mathcal{Y}'_1) = (y_0, Y_1, \mathcal{Y}_1)$. This way Alice will not detect Eve and she will reveal $(x_0, X_1, \mathcal{X}_1)$. The adversary has two choices now. She either sets $(x'_0, X'_1) = (x_0, X_1)$ and send it to Bob, or she sends (x'_0, X'_1) to Bob where $(x'_0, X'_1) \neq (x_0, X_1)$. We will analyze each of these two cases separately.

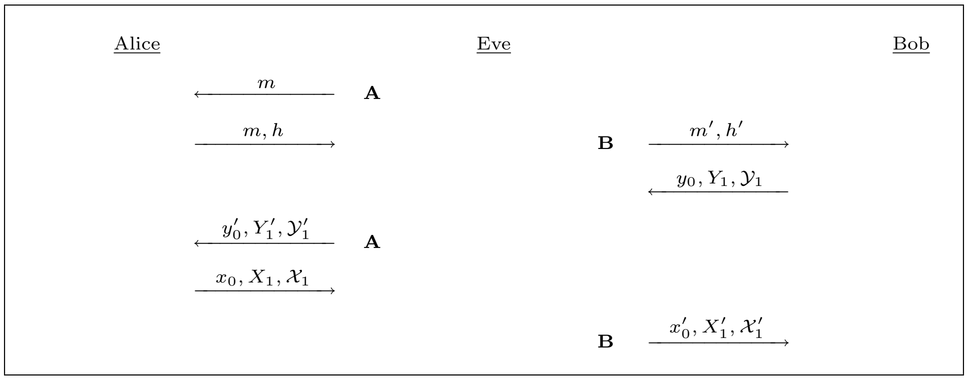

Alice
Eve
Bob
m
A
m, h
B
m', h'
$y_0, Y_1, \mathcal{Y}_1$
$y'_0, Y'_1, \mathcal{Y}'_1$
A
$x_0, X_1, \mathcal{X}_1$
B
$x'_0, X'_1, \mathcal{X}'_1$

FIGURE 10. Attack of Type ABAB

Let us first consider the case where $(x'_0, X'_1) = (x_0, X_1)$. In this case, the adversary has collected $(X_0, \mathcal{X}_0)$ from previous session. She then sends m to Alice and Alice replies with (m, h). She will then send (m', h') to Bob. At this point the rest of the flows are determined to be the following: She receives $(y_0, Y_1, \mathcal{Y}_1)$ from Bob, sets $(y'_0, Y'_1, \mathcal{Y}'_1) = (y_0, Y_1, \mathcal{Y}_1)$, and sends it to Alice. Further, she receives $(x_0, X_1, \mathcal{X}_1)$ from Alice, lets $(x_0', X_1', \mathcal{X}_1') = (x_0, X_1, \mathcal{X}_1)$, and sends it Bob. Hence, this case is exactly the notion of PCR depicted in Figure 6.

The second case is when $(x_0', X_1') \neq (x_0, X_1)$. Assume that Eve can mount a successful attack of type ABAB with $(x_0', X_1') \neq (x_0, X_1)$. That is, she has collected $X_0, \mathcal{X}_0$ from previous session. She chooses m and receives h such that $h = H(m, x_0)$. Then, she submits m', h'. Finally, she receives x_0, X_1 and she is supposed to send x_0', X_1' such that $(x_0', X_1') \neq (x_0, X_1)$, $H(x_0) = H(x_0')$, $H(x_0, X_1) = H(x_0', X_1')$, and $h' = H(m', X_0')$. We reduce Eve to a successful player against the Challenger of PSPR game, depicted in Figure 5. The reduction is illustrated in Figure 11.

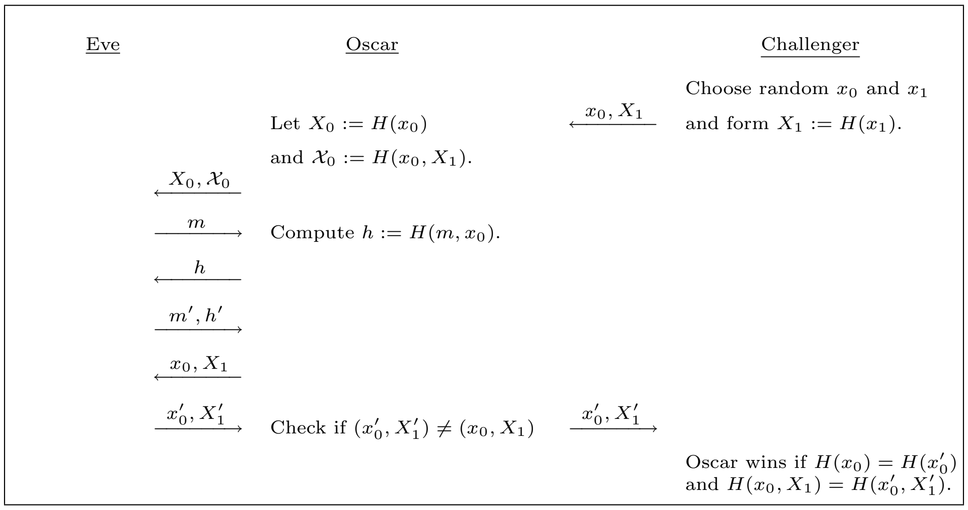

Eve
Oscar
Challenger
Choose random x_0 and x_1
and form $X_1 := H(x_1)$.
x_0, X_1
Let $X_0 := H(x_0)$
and $\mathcal{X}_0 := H(x_0, X_1)$.
$X_0, \mathcal{X}_0$
m
Compute $h := H(m, x_0)$.
h
m', h'
x_0, X_1
x_0', X_1'
Check if $(x_0', X_1') \neq (x_0, X_1)$
x_0', X_1'
Oscar wins if $H(x_0) = H(x_0')$
and $H(x_0, X_1) = H(x_0', X_1')$.

FIGURE 11. Reducing Eve to a Player Against the Challenger of PSPR

Note that, Oscar is playing against the Challenger of PSPR and at the same time he is playing the role of both Alice and Bob against Eve.

5.2. **Two-session Attacks.**

Now consider attack scenarios which span over two or more sessions. That is the adversary is active but remains undetected in all sessions of the attack. She then submits her message in the last session of the attack. If she tampers with y_0, Y_1, x_0, or X_1 and remains undetected, then we go back to the cases described above. Hence, it remains to examine the cases when she changes the binding hash values. We look at the case where Eve changes the value of $\mathcal{Y}_1$ to $\mathcal{Y}_1'$. The case where Eve alters $\mathcal{X}_1$ to $\mathcal{X}_1'$ is analogous due to the symmetry of the protocol structure.

Assume that Eve changes $\mathcal{Y}_1$ to $\mathcal{Y}_1'$ and does not touch y_0 or Y_1. She goes undetected in this session because Alice verifies y_0 and Y_1, but only records $\mathcal{Y}_1'$ without verification. She then updates her state as follows $y_{-1}^* := y_0$, $Y_0^* := Y_1$, $(\mathcal{Y}_0^*)' := \mathcal{Y}_1'$.

Now, in the next session, Alice sends $(y_0, Y_1, \mathcal{Y}_1)$ and Eve has to change it to an appropriate $(y_0', Y_1', \mathcal{Y}_1')$ to remain undetected. Otherwise, Alice will call for resynchronization. Alice checks to see if $H(y_0') = Y_0^*$ and $H(y_0', Y_1') = (\mathcal{Y}_0^*)'$. We treat the two cases $y_0 = y_0'$ and $y_0 \neq y_0'$ separately.

If $H(y_0') = Y_0^*$ and $y_0 \neq y_0'$, then Eve having seen y_0 has found y_0' such that $H(y_0') = H(y_0)$. This means that Eve has found a second pre-image of y_0.

On the other hand, when $y_0 = y_0'$, the condition $H(y_0') = Y_0^*$ holds. Then, Alice verifies to see if $H(y_0, Y_1') = (\mathcal{Y}_0^*)'$. If it holds, then Eve is a successful player in the BPR game of Figure 7.

If the adversary were to mount an attack that spans over more than two rounds, she would have to successfully pass the second round. However, the above discussion shows that the adversary can only pass the second session without being detected if she can win the BPR game or SPR game.

5.3. **The Security Theorem.**

We investigated all possible attacks against the message recognition protocol of Figure 2 by considering two different cases, mainly if the attack is taking place over one session, or if it spans over more than one session. We examined these two cases separately.

In the first case, there are six possible attack scenarios: BABA, BAAB, ABBA, AABB, BBAA, and ABAB. Attacks of type BABA, BAAB, and ABBA can be reduced to the ABAB case. Further, we showed that a successful adversary (Eve) in attacks of type AABB, BBAA, and ABAB attacks can be reduced to a successful player (Oscar) in the PPR, PSPR, or PCR games.

In the case of attacks that occur over more than one session, we showed that the successful adversary can be reduced to a successful player against the BPR or SPR games.

This concludes the analysis of different attack scenarios and implies the following theorem

Theorem 1. *A successful adversary who can efficiently deceive Bob in outputting (Alice, m'), where Alice never sent m', implies an efficient algorithm in winning PPR, PSPR, PCR, or BPR hash function games.*

This theorem precisely identifies the required properties for a hash function to be used in the message recognition protocol of Figure 2. There is no concrete construction of such a hash function. However, no one knows how to prove that a concrete construction of a hash function has any non-trivial property. It is a standard approach taken in the literature to assume some properties for an idealized hash function and to prove security of a given protocol assuming these assumptions. Note that the same approach was taken in [5].

6. Conclusion and Final Remarks

We proposed a new design for message recognition protocols suitable for ad hoc pervasive networks. This proposal does not make use of hash chains. Hash chaining techniques have been used in recent designs of message recognition protocols. In this approach, the sensor motes are required to save values of a hash chain in their memories for every single user they want to communicate with. Since we do not use this technique, we no longer require the sensor motes to save values of a hash chain in their memories. This relaxes the memory requirements.

Moreover, the passwords are chosen at random in each session. Hence, they are independent of one another and are being refreshed in each session. This can be done for any arbitrary number of times, so we do not need to fix the total number of times the protocol can be executed which implies a desired flexibility in this regard.

As the passwords are independent of one another, we do not need to consider assumptions that depend on the number of sessions the protocol is executed. Whereas recent designs based on the hash chaining technique had to assume families of assumptions based on the number of sessions the protocol is executed. This implies that their security weakens as the number of sessions increases. Since we are not using hash chains, the security of our protocol is independent of the number of times the protocol is executed.

Finally, a practical procedure for resynchronization is provided. This implies that in case of any possible adversarial disruption or communication failure, the protocol can be recovered.

References

[1] Ross Anderson, Francesco Bergadano, Bruno Crispo, Jong-Hyeon Lee, Charalampos Manifavas, and Roger Needham. A new family of authentication protocols. In *ACMOSR: ACM Operating Systems Review*, volume 32, pages 9–20, 1998.

[2] Don Coppersmith and Markus Jakobsson. Almost optimal hash sequence traversal. In *Financial Cryptography*, pages 102–119, 2002.

[3] Christian Gehrmann. Multiround unconditionally secure authentication. *Designs, Codes, and Cryptography*, 15(1):67–86, 1998.

[4] Jonathan Hammell, André Weimerskirch, Joao Girao, and Dirk Westhoff. Recognition in a low-power environment. In *ICDCSW '05: Proceedings of the Second International Workshop on Wireless Ad Hoc Networking (WWAN) ICDCSW'05)*, pages 933–938, Washington, DC, USA, 2005. IEEE Computer Society.

[5] Stefan Lucks, Erik Zenner, André Weimerskirch, and Dirk Westhoff. Entity recognition for sensor network motes. In *GI Jahrestagung (2)*, pages 145–149, 2005.

[6] Chris J. Mitchell. Remote user authentication using public information. In Kenneth G. Paterson, editor, *IMA Int. Conf.*, volume 2898 of *Lecture Notes in Computer Science*, pages 360–369. Springer, 2003.

[7] André Weimerskirch and Dirk Westhoff. Zero common-knowledge authentication for pervasive networks. In *Selected Areas in Cryptography*, volume 3006 of *Lecture Notes in Computer Science*, pages 73–87. Springer, 2003.

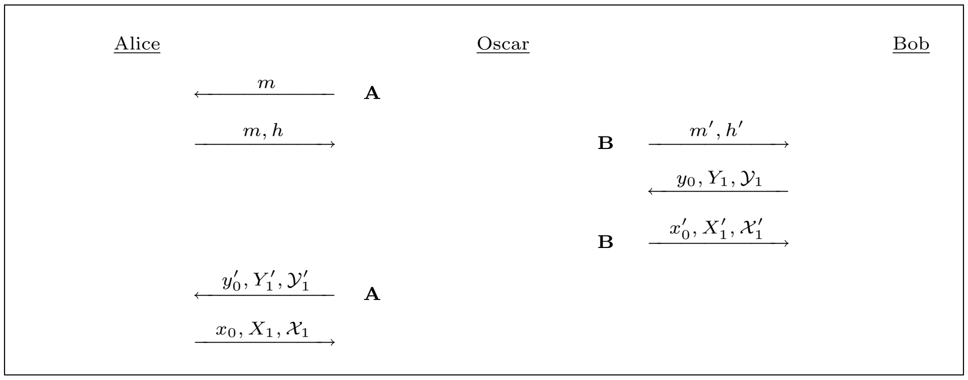
7. Appendix

As described in in Section 5, there are six possible one-session attacks against the protocol of Figure 2. Three of these attacks, mainly AABB, BBAA, and ABAB, were analyzed previously. Here we want to examine the remaining three attacks, mainly BABA, BAAB and ABBA.

We begin by reducing the BABA attack to the ABBA attack. Further, we reduce the ABBA attack to the ABAB attack that was analyzed in Section 5. Finally, the only remaining attack scenario, BAAB, is also reduced to the ABAB attack. This concludes the analysis of the six different attack scenarios.

7.1. **Reducing BABA attack to ABBA attack.**

Attack of type ABBA is depicted in Figure 12 and Figure 13 illustrates the attack of Type BABA.



Alice
Oscar
Bob
m
A
m, h
B
m', h'
$y_0, Y_1, \mathcal{Y}_1$
B
$x'_0, X'_1, \mathcal{X}'_1$
$y'_0, Y'_1, \mathcal{Y}'_1$
A
$x_0, X_1, \mathcal{X}_1$

Figure 12. Attack of Type ABBA

These two attacks differ only in the order of the first two steps. The ABBA attack is as follows:

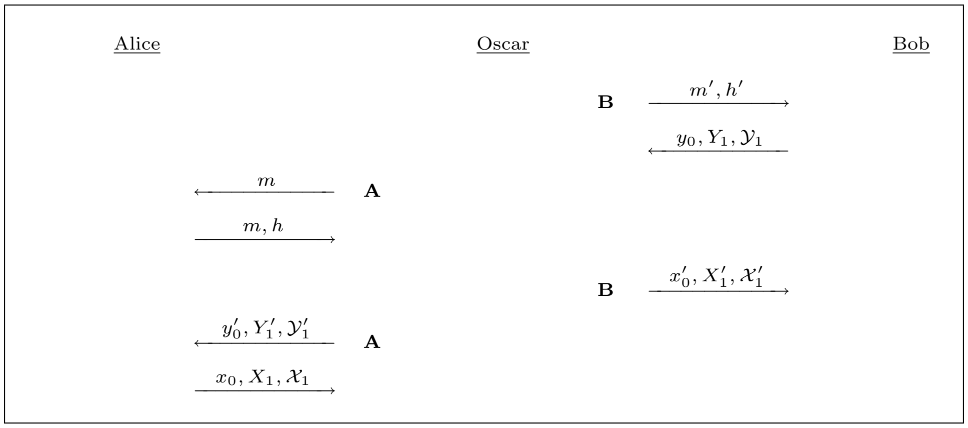

Alice
Oscar
Bob
B
m', h'
$y_0, Y_1, \mathcal{Y}_1$
m
A
m, h
B
$x'_0, X'_1, \mathcal{X}'_1$
$y'_0, Y'_1, \mathcal{Y}'_1$
A
$x_0, X_1, \mathcal{X}_1$

FIGURE 13. Attack of Type BABA

- **A**: Oscar sends m to Alice and she responds with m, h.
- **B**: Oscar sends m', h' to Bob and he replies with $y_0, Y_1, \mathcal{Y}_1$.
- **B**: Oscar sends $x'_0, X'_1, \mathcal{X}'_1$.
- **A**: Oscar sends $y'_0, Y'_1, \mathcal{Y}'_1$ to Alice and receives $x_0, X_1, \mathcal{X}_1$ from her.

The BABA attack has the following order:

- **B**: Oscar sends m', h' to Bob and he replies with $y_0, Y_1, \mathcal{Y}_1$.
- **A**: Oscar sends m to Alice and she responds with m, h.
- **B**: Oscar sends $x'_0, X'_1, \mathcal{X}'_1$.
- **A**: Oscar sends $y'_0, Y'_1, \mathcal{Y}'_1$ to Alice and receives $x_0, X_1, \mathcal{X}_1$ from her.

Note that in the BABA attack scenario, the choice of m is independent of what the values of y_0, Y_1 and $\mathcal{Y}_1$ are. That is, knowing $y_0, Y_1, \mathcal{Y}_1$ before choosing m is not going to help Oscar. On the other hand, he is committing himself to m', h' before receiving any values, such as h, that could possibly help him. If Oscar wins by first choosing m', h' and then receiving h in the BABA attack scenario, then he can also win the ABBA attack by using the same values m, m', and h'.

7.2. Reducing ABBA attack to ABAB.

Recall the ABAB attack described in Section 5:

- **A**: Eve sends m to Alice and she responds with m, h.
- **B**: Eve sends m', h' to Bob and he replies with $y_0, Y_1, \mathcal{Y}_1$.
- **A**: Eve sends $y'_0, Y'_1, \mathcal{Y}'_1$ to Alice and receives $x_0, X_1, \mathcal{X}_1$ from her.
- **B**: Eve sends $x'_0, X'_1, \mathcal{X}'_1$.

This attack differers from the ABBA attack in the order of the last two steps. In the ABAB attack, Eve first receives $x_0, X_1, \mathcal{X}_1$ from Alice, then she has to send $x'_0, X'_1, \mathcal{X}'_1$ to Bob. Whereas in the case of the ABBA attack, Oscar has to send $x'_0, X'_1, \mathcal{X}'_1$ to Bob before he receives $x_0, X_1, \mathcal{X}_1$ from Alice. If Oscar has a winning strategy in the ABBA attack, the Eve can use him in her ABAB attack by sending the same values of $x'_0, X'_1, \mathcal{X}'_1$ that Oscar sends to Bob.

7.3. Reducing BAAB attack to ABAB.

Depicted in Figure 14 is the attack Type of BAAB.

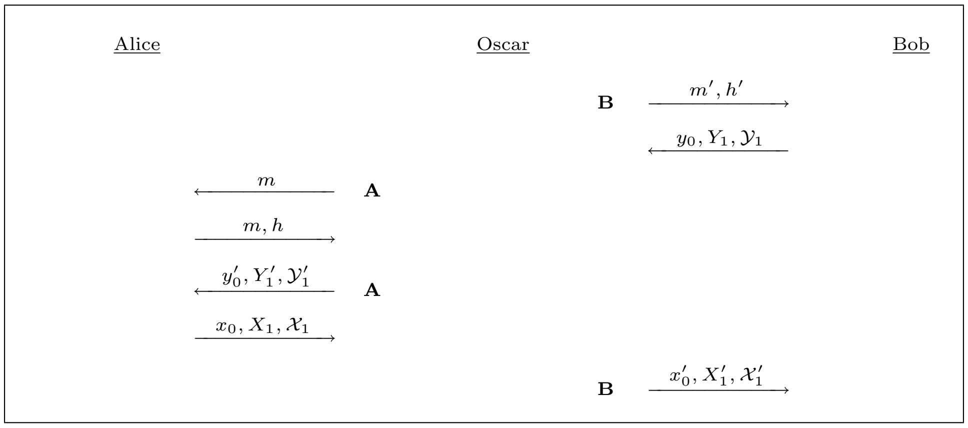

Alice
Oscar
Bob
B
m', h'
$y_0, Y_1, \mathcal{Y}_1$
m
A
m, h
$y_0', Y_1', \mathcal{Y}_1'$
A
$x_0, X_1, \mathcal{X}_1$
B
$x_0', X_1', \mathcal{X}_1'$

FIGURE 14. Attack of Type BAAB

Recall that knowing $y_0, Y_1, \mathcal{Y}_1$ before choosing m is not going to help Oscar. Moreover, in the BAAB attack, Oscar is first committing himself to m', h'. If Oscar wins the BAAB attack by first choosing m', h' and then receiving h, then so will Eve in the ABAB attack, by just using the same values m, m', and h'.