Secure Association for the Internet of Things

Almog Benin
almogbenin@gmail.com

Sivan Toledo
stoledo@tau.ac.il

Eran Tromer
tromer@cs.tau.ac.il

Blavatnik School of Computer Science, Tel-Aviv University

Abstract. Existing standards (ZigBee and Bluetooth Low Energy) for networked low-power wireless devices do not support *secure association* (or *pairing*) of new devices into a network: their association process is vulnerable to man-in-the-middle attacks. This paper addresses three essential aspects in attaining secure association for such devices.
First, we define a user-interface primitive, *oblivious comparison,* that allows users to approve authentic associations and abort compromised ones. This distills and generalizes several existing approve/abort mechanisms, and moreover we experimentally show that OC can be implemented using very little hardware: one LED and one switch.
Second, we provide a new *Message Recognition Protocol* (MRP) that allows devices associated using oblivious comparison to exchange authenticated messages without the use of public-key cryptography (which exceeds the capabilities of many IoT devices). This protocol improves upon previously proposed MRPs in several respects.
Third, we propose a robust definition of security for MRPs that is based on *universal composability,* and show that our MRP satisfies this definition.

1 Introduction

Devices that compute and communicate need to ensure that they communicate with the intended party rather than an attacker. Achieving this goal requires two main components: a mechanism for the user to specify the intended parties, and a protocol that allows secure (authenticated and perhaps private) communication between the intended parties. These should be proven secure under a well-defined model and definition, that capture the capabilities of pertinent adversaries. Many recent work address this challenges for various types for devices; see [32, 6, 20, 34] for discussions and surveys.

Achieving the same goal with low-power computationally-weak devices with a very minimal user interface is extremely hard. In fact, the task is considered so hard that standardization bodies gave up. In both ZigBee and Bluetooth Low Energy (BLE, also called Bluetooth Smart) the association phase, in which a new device is added to a mesh network or paired with another device, is completely vulnerable to simple man-in-the-middle (MITM) attacks[1].

The goal of this paper is to show that secure association is possible for the low-power computationally-weak devices that make up the Internet of Things (IoT),

[1] In some cases, this vulnerability is limited to a time window of a few minutes.

even if they have a very minimal user interface. Our solutions are applicable to interoperable devices intended to be deployed by the end consumer. If adopted, our solutions can rectify severe vulnerabilities in ZigBee and BLE and can secure emerging IoT standards.

To address this problem, this paper addresses the three components that are necessary for secure association. First, we propose (in Section 2) an abstract user-interface primitive for specifying the intended parties in an association (pairing) process. The specification is *passive*; the devices are told to associate (without telling them with whom to associate), for example by pressing a button on each device. They then attempt to find a willing party. If successful, they present to the user enough information to allow the user to decide whether the two parties are the intended ones or not. If yes, she approves the association (say by pressing the button again). If not, she aborts the association attempt (e.g. by not pressing the button until a timeout elapses). We show that wireless devices with only a single LED as a display device can present sufficient information to the user to allow her to reliably approve or abort associations. This primitive, which we call *oblivious comparison (OC)*, generalizes several concrete user-interface mechanisms that have been proposed for devices with richer user interfaces.

Once the intended parties are established using OC, they need a protocol for secure exchange of (logical) messages over the insecure (physical) connection. Establishes an authenticated and private channel, in this setting, seems to require public-key cryptography, which is too expensive for many IoT devices (as is indeed the assumption in both ZigBee and BLE). Instead, we pursue (in Sections 4 and 5) the notion of *message recognition protocol (MRP):* informally, a protocol to exchange authenticated messages but without ensuring privacy. This allows lightweight implementations that rely only on symmetric-key cryptography. MRPs have been discussed in the literature, but without adequate formal definitions, and indeed we show that ad-hoc security definitions for MRPs can completely fail to authenticate messages if the attacker can somehow affect the choice of transmitted messages (a chosen-message attack). We propose stronger definitions, in both the chosen-message stand-alone setting and in the framework of *universal composability* (UC) [4]; we show them equivalent to each other, and strictly stronger than the weaker definition.

Finally, we propose (in Section 6) a new MRP: the *PEBIUS* protocol. PEBIUS improves on prior MRPs in being (1) bidirectional, thereby more efficient, (2) can run forever, whereas some of the prior MRPs in the literature are limited to an a priori fixed number of rounds, and (3) is proved secure under the UC sense, and in particular is secure against chosen-message attacks.

2 User Interaction for Secure Association

Protocols for secure association of two devices (and transitively of a device into a network) require a mechanism to install information about one of the devices into the other. Some protocols require both devices to have information about the other. The type of information required varies among protocols: it may constitute a public key, a secret symmetric key, a commitment, etc. Many

IoT devices have only one rich interface to the outside world, namely a data radio. The radio cannot be used to transfer identifying information for secure pairing because doing so leaves devices vulnerable to MITM attacks. (ZigBee does use this mechanism in its so-called *standard-security mode*, which transfers the network's *master key* over the air and in the clear; this mode is fundamentally insecure.) What other mechanisms can be used to transfer this information into IoT devices?

2.1 Background on Unsuitable or Ineffective Mechanisms

One mechanism relies on *preinstallation* of this information. One example is ZigBee's *high-security* mode that assume that the network's master key is stored at manufacture time or at firmware-update time. This is infeasible for interoperable IoT devices intended to be deployed by consumer, and as a consequence, no ZigBee device that we are aware of uses it. Another common example of preinstallation is the inclusion of prestored cryptographic certificates in mainstream operating systems (e.g., CA root certificates). This mechanism appears to be inappropriate for IoT devices because (1) there is no usable and reliable mechanism to invalidate/replace compromised certificates on IoT devices, (2) the mechanism requires public-key computations that are considered too expensive for ZigBee and Bluetooth Low Energy (BLE) devices, and (3) the stored certificates allow authentication of a *named* party in a protocol (e.g., domain name), but many IoT devices are nameless (some devices may be large enough to carry a printed identifying name, similar to a MAC address, but this is not a common practice). Even if the difficulties with certificates are overcome, they only allow association if at least one of the devices has a rich enough interface to allow the user to specify or approve the name of the other device; this is not the case when two simple IoT devices need to be associated (e.g., a light bulb and a controlling switch). We that this mechanism is unsuitable for IoT.

Another class of mechanisms relies on *out-of-band data transfer*. In these mechanisms, one device provides the user with the information to transfer (e.g., encodes it and displays it on a screen), which the user types into the other device (e.g., using a keyboard). This mechanism is used in Bluetooth's [12] *passkey-entry* association mode (unfortunately, the implementation of this mode in Bluetooth 4 is insecure [18], and so is the similar *PIN mode* in Bluetooth 2.0 [11][31]). The data can be transferred not only by a human user, but also using high-bandwidth input devices such as a high-resolution camera [24]. There are numerous suggested methods for out-of-band data transfer (see [6, 14] and the references therein).

However, analysis of the mechanisms described so far suggests that they are still too expensive for very small IoT devices, due to severe constraints in hardware interfaces, power, storage, chip sizes and deployment scenarios. This leaves us with one class of effective mechanisms, some new and some known, which we describe in the following.

2.2 Out-of-Band Oblivious Comparisons

We now show that a number of user-interaction mechanisms, in existing standards and literature, all implement an abstract primitive that we call *oblivious comparison (OC)*. This common functionality has not been recognized and defined in the literature. In the following, we provide a definition of oblivious comparison, and review prior implementations. In Section 2.3 we describe a new implementation using very little hardware, and in Section 5 we discuss a universally-composable definition of OC

Definition 1. *An out-of-band oblivious comparison between devices A and B consists of the following steps. Initially, A and B hold values, v_A and v_B respectively (derived from their previous interaction, which in our setting is over a non-confidential and unauthenticated bidirectional channel). They would like check whether $v_A = v_B$.*

1. *Each of A and B sends the value they derived from the first step to a* verifier *(e.g., a human user). The (unidirectional) channel to the verifier must allow the verifier to ascertain that the values came from A and B, and to determine whether the sent values are identical.*
2. *The verifier informs A and B whether the values were identical, via a (unidirectional, non-confidential) channel that cannot be spoofed (e.g., a button).*

In Bluetooth's *numeric comparison* mode [17], the two devices attempt to agree on a string (six decimal digits). The negotiation is done in-band on the not-yet-authenticated wireless channel. Both then display the string to the user (this is the first channel; the verifier is the human user and she ascertains the identity of the devices by visual inspection and hand-eye coordination). The user presses a switch on each device to tell it whether the two strings are identical, in which case the association succeeds (this is the second channel). Note that the user only has to compare the values, and not remember them. In the Bluetooth standard, the string is essentially a hash of the concatenation of the public keys of the two devices, which are also transmitted in-band.

Bluetooth's numeric comparison requires a rich out-of-band user interface (a numeric display), but the literature contains other mechanisms that essentially implement oblivious comparison, using less resources.

Saxena et al. proposed a mechanism that uses an LED on one device and a video camera on the other [30]. The device with the LED blinks it in a pattern that represents the value. The device with the camera observes the blinking and compares it to the value it stores. This device also serves as a verifier. If the values are identical, the device with the camera tells the user (e.g., using an LED or a display) to tell the other devices that the comparison passed. The human is used to implement the third channel; physical proximity and the direction of the camera are used to implement the second.

Roth et al. [29] proposed *Press-to-advance*, an implementation of out-of-band oblivious comparison using bi-color LEDs or an LED and a graphical display. Whenever the user presses a button on one of the devices, both encode the next bit of the value as a color and display it (on the green/red LED or as a dot on

the display). The sequence advances manually by one bit every time the user presses a key on the device that she holds. If the colors differ, the user aborts the association. If both devices display the same value, they associate when the entire bit string has been presented.

Prasad and Saxena [28] describe an alternative, where the user copies a bit pattern from one device to the other by pressing a button every time an LED blinks. They also propose *Blink-Blink* [28], based on a human comparing the blinking and glowing patterns of LEDs (two LEDs on each device), as well as Blink-Beep and Beep-Beep variants. We discuss this further below. Note that press-to-advance, Blink-Blink and 1-LED OC (described below) all contradict an assertion by Kuo et al. [15] who claimed that devices with only a button and an LED cannot be securely paired.

Goodrich et al. [10], and [28] in their *Beep-Beep* scheme, propose a mechanism based on the human ability to tell whether two simultaneous series of audio tones are identical. The authenticity of the display channel relies on identification of audio sources; this mechanism is appropriate for headsets, perhaps not to loudspeakers. Combinations of audio and visual signals are also possible.

Transitive OC. Association protocols based on OC can be easily extended from pairwise association to network-wide association. Suppose that in a communication network, nodes A and T have established an authenticated channel and that nodes B and T have established such a channel. Nodes A and B can now associate so that they can communicate directly (they can obviously exchange authenticated messages through the trusted party T, but direct communication is usually more efficient). They attempt to agree on a value v using an unauthenticated channel (first channel in the definition) and then both send the value they believed they agreed on to the verifier T using the authenticated channels. T checks whether the two values are identical and informs both parties of the result, also using the authenticated channels. For example, in a ZigBee network a user might associate a new node A with a nearby router T that is already part of a mesh network with trust center B. Now A and B establish an authenticated communication channel through T.

2.3 1-LED Oblivious Comparison

We propose a new oblivious comparison mechanism, "1-LED OC", that requires merely a single LED and a single button in each device, and moreover require minimal input from the user (a single button press). The devices simultaneously blink their LED using an on-off pattern derived the value that the devices agreed upon, and the user decides whether the patterns are identical or very different, pressing the buttons accordingly, as explained next.

In 1-LED OC, the parties A and B proceed as follows. First, each party obtains its local value, v_A or v_B, by executing a protocol that uses the OC primitive. They synchronize their real-time-clocks using a radio channel. In the first step, A sends to B a random number R_A, and likewise B sends to A a random R_B. A then blinks its LED in a pattern derived from $h(v_A, R_A, \widehat{R_B})$, and concurrently B blinks its LED in a pattern derived from $h(v_B, \widehat{R_A}, R_B)$.

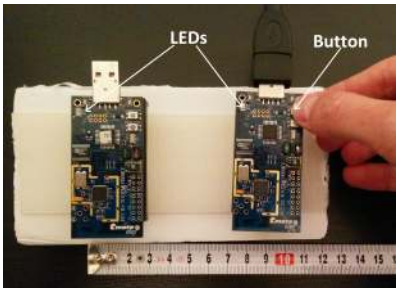

Fig. 2.1. The LED comparison experiment.

Here, h is a hash function, that we formally model as a random oracle, and concretely instantiate using SHA-256. The hat values, $\widehat{R_A}$ and $\widehat{R_B}$, represents the corresponding received value, which might be corrupted. The encoding of a string s into a blinking pattern is as follows: we derive a pseudorandom sequence S from s, by using s as the seed into a pseudorandom generator (e.g., any stream cipher); then for each bit S_i of S, in sequence, the LED is turned off for a fixed period τ if $S_i = 0$, and turned on for a period τ if $S_i = 1$. The two devices perform the blinking synchronously (up to the accuracy of their shared clock), until their respective buttons are pressed by the user.Figure 2.1 shows a physical realization

The blink period τ can be pretty short: our experiments below suggest that $\tau \approx 66$ms works well. Practically, delays and clock drifts are insignificant at the time scale of human perception. Low-power low-cost RF devices can easily synchronize their clocks to within better than a millisecond, and their real-time clocks are accurate to 100ppm or better. These translate to a drift of about 4.4ms over 256 periods of 66ms each[2]. For comparison, Steinmetz researched the human perception of media synchronization [33], and found that humans fail to detect synchronization error of ±80ms. Moreover, if drift becomes a problem, the two devices can wirelessly re-synchronize while blinking.

Comparison to Blink-Blink. The aforementioned Blink-Blink scheme [28] is also based on comparing LED, but takes a different approach. Blink-Blink uses a Short Authentication String protocol and requires bit-accurate comparison of short strings (15 bits). It thus uses slow blinking rates (300ms to 800ms), and uses two LEDs per device to ensure synchronization of the strings. 1-LED OC, instead, compares long pseudorandom, relying not on bit-level accuracy but on the ability to distinguish identical sequences from nearly-independent sequences; we can thus use a blinking rate that is an order of magnitude faster (40ms to 66ms), and do not need an extra LED for synchronization.

Experimental methodology. To demonstrate the feasibility of the 1-LED OC, we tested this mechanism on human subjects. The devices were bare Tmote

[2] In the extreme case, each device deviate from the desired time by $256 \cdot 66\mu s \cdot 100ppm = 1.7$ms, so in total 3.4ms, plus at most 1ms due to message latency.

Sky boards running Contiki that we custom programmed for this task. The devices were attached to flat surface to keep the LEDs exactly 7cm from each other (see Figure 2.1). The devices coordinated the values and the timing using radio messages. They also recorded the interaction for analysis.

Human subjects were presented with LED blinking sequences on these devices. They were instructed to press the button on the rightmost device once they decided whether the sequences are identical: a long press (3 seconds or longer) if the user believed the sequences are identical, and a short press (less than 3 seconds) if different. The next sequence was presented 3 seconds after the button press. To help subjects understand the task, the first 6 sequences were training sequences. In these sequences, a green LED would light after the user's input if the user was right and a red LED would light if she was wrong. The training runs were excluded from the analysis. After the training runs, which were displayed at a rate of 15bps, the user classified 8 sequences at the same rate. They were followed by 2 training sequences at 25bps and then 8 test sequences at 25bps.

The experiment was conducted on 10 volunteers, age 14–66, of both genders.[3]

Experimental results. Most of the subjects gave the correct answer to all of the sessions. In a few cases, the subject classified the sequence correctly but accidentally pressed the button incorrectly; if the subject immediately indicated verbally that he knows that he was wrong, we regarded such answers as correct, under the assumption that in a real scenario, the user would immediately dissociate the devices after mis-pressing the validation button.

All except one subject classified all sequences correctly. That single subject had 2 false-positive answers (out of 16). We conclude that the oblivious comparison using blinking LEDs is an easy task for humans.

3 Message Authenticity With Privacy

In the previous section, we discuss how to transfer a short bit string authentically. In this section and the rest of the paper, we discuss how to bootstrap from a shared string into exchanging authenticated messages: a *Message Authentication Protocol (MAP)*.

In this section we discuss association models which can provide both authentic and private communication channel. All the methods in this section aim to establish a shared secret key between two devices. Once the key is established, the devices can communicate securely using standard symmetric primitives (symmetric encryption and MAC, or signcryption such as CCM-AES).

Preinstallation. If both devices are manufactured by the same origin, and the devices are intended to communicate with each other in advance, then the manufacturer can install the secret keys for the communication in both devices in advance.

Out of band (OOB). By out of band we mean that the secrets does not relay over the RF channel. If one device has an input interface and the other one has

[3] The experiment is ongoing. The final version of the paper will include a larger subject population and more detailed statistics.

an output interface such that a (short) bit string can be transferred securely (authentically and confidentially), then both devices can generate their secret key and output them to such interface. A human user then can take these keys and enter them into the input interface of the respective other device. Examples of such mechanisms are the Bluetooth's passkey-entry[12] and barcode-camera pairing mechanism [24].

Public key infrastructure (PKI). PKI is very common way to bootstrap a secure channel in the Internet. Loosely speaking, in PKI there is a trusted third party (Certificate Authority, CA), which holds the bit strings to be authenticated (public-keys) of servers. Whenever a client in the Internet wants to initiate a secured connection with a server, it asks the server for its public key and a certification for that key, signed by the CA. By verifying this certificate the client can be sure that the public key belongs to the server.

Anonymous channel. This protocol [5] assumes the existence of an anonymous channel between the devices. Anonymous channel is a broadcast channel that hides the origin of the messages. In this protocol, A can send the secret bit 1 to B by broadcasting an (empty) packet with the source field set to A. Similarly, A can send the secret bit 0 to B by broadcasting an (empty) packet with the source field set to B. Only B can identify the real source of the packet (since it did not send it, the source is A), and can recover the secret bit (1 if the source is set to A or 0 otherwise). An eavesdropper cannot retrieve the secret bit since it cannot figure out whether the packet was actually sent by A or B. By randomly generating n such packets A and B can agree on an n-bit secret key. While the theoretical idea is very elegant, it is unclear whether such anonymous channels can be implemented at reasonable cost, given the numerous methods available for fingerprinting and geometrically locating devices.

Accelerometer. This method [23] assumes that both devices have an accelerometer peripheral. The human user holds these two devices, and shakes them together. Using the accelerometer, the devices tracks the movements of the device, and generates a bit string based on this information. Since the devices are shaken in the similar way, they will generate similar bit strings. By assuming that humans can create random bit strings by shaking the devices, this bit string is the shared secret.

Bluetooth's numeric comparison. Bluetooth [12] offers some association models for initiating secure connection. One of them is the numeric comparison association model. In this model, each device outputs to a screen a string, and a human user is requested to compare them, and inform the devices with the answer. Loosely speaking, Bluetooth uses Diffie-Hellman to exchange the public-keys between the parties, and asks the human user whether the parties agrees on the same public-keys. After that, the parties can agree on a new common private-key confidentially, using the authenticated public-key. This protocol was proved to be secured by Lindell [17].

In 2.2, we generalized this association method into a primitive we call oblivious comparison, and in Section 4 we will show that it can be used to provide authenticity (but not privacy) even without using PKC.

Figure 3.1 compares these methods and explains why they infeasible for low-power devices. As can be figured out from the table, these association models are not suitable for low-power devices. In the next section, we show that if authenticity (without secrecy) suffices, then there exist alternatives that surmount all of these hurdles.

Method	Assumptions	Hurdle
Preinstallation	Devices are created by the same manufacturer	Interoperability requirement
OOB	Both devices have input/output interface for bit strings	Very simple interfaces
PKI	A trusted third party exists; both devices can use PKC	Generally no trusted third party; PKC requires too much computational power
Anonymous channel [5]	An anonymous channel exists	Questionable security
Accelerometer [23]	Accelerometer exists on both devices; humans can create random bit strings by shaking	Accelerometer does not exists on all devices
Numeric comparison [12]	The user can compare short strings	Currently implemented using PKC. We show how to implement without PKC.

Fig. 3.1. Comparison of bootstrapping methods for initial shared secret between two devices.

4 Message Authenticity Without Privacy

Many existing mechanisms in the literature and standards provide message authentication protocols using public-key cryptography (PKC). For example: (1) public-key infrastructure uses trusted third party to authenticate servers in the internet and (2) classic Bluetooth's simple pairing mechanism [12], whose numeric comparison mode comply with our OC abstraction. Although they achieve all desired functionality (authenticity and privacy), PKC operations are usually not suitable for cheap-low power devices, because of the relatively high computational requirements of PKC and the perception that it would be too slow or too expensive for such devices (the increased cost is due to the increased silicon area required for memory and computation, as well as power consumption on battery-powered devices). PKC can be avoided if we give up on *confidentiality*, and focus just on *authenticity* of messages. This trade-off makes sense in many IoT applications. For example, letting the adversary to know when a door is opened is less significant than letting the adversary to open a door. One previously suggestion, that uses just symmetric-key cryptographic and achieves only authenticity is the TESLA broadcast authentication protocol [27]. Although providing a stronger

primitive (broadcasting), this approach requires the devices to be synchronized[4] — which in turn is achievable using heavy tools such as PKC [27].

Another approach, that does not require synchronization (but does not provide broadcasting), is several protocols called Message Recognition Protocol [1, 35, 13, 19, 21, 8, 22]. In this section, we focus on this notion, and see how it can be combined with OC to achieve message authentication protocol.

Recognition vs. Authentication. MRP provides a weaker security guarantee than the standard message authentication protocol (MAP). While the former provides guarantee that the messages originated by a party he negotiated in the past, the latter provides guarantee that the messages originated by a party with a particular identity (expressed by knowledge of some identity-specific secret). For example, consider a standard Diffie-Hellman key agreement protocol, in which there is no exchange of certificate or other involvement of third party. The protocol ends with agreement on a symmetric key which is used later for authenticating messages (by means of MAC) of each other. This protocol provides message recognition, since the messages can be recognized to be originated by the same party which was involved in the Diffie-Hellman key exchange. However, the identity of that party is not assured, since it is vulnerable to MITM attack; thus this protocol does not provide message authentication. In this section will show a generic method which utilizes OC and MRP to achieve authentication.

4.1 Defining Message Recognition Protocols

Weimerskirch and Westhof [35] provided a candidate message recognition protocol, without clearly stating its security property. Subsequently, Hammel et al. [13] implemented that protocol and introduced the term "message recognition" for its claimed security property, but their definition and treatment were still informal (and indeed that protocol was later found out to be completely insecure [19]). Mashatan and Vaudenay [22] they gave a different protocol, with an ad-hoc definition of Message Recognition Protocol tailored to their construction. Informal tailored definitions were also used in other papers [19, 21, 8], and as we discuss in 5, this leave crucial aspects undefined. One such aspect is who chooses the logical messages to be transmitted, with what computational power and with what knowledge. Although message choice arises within Gehrmann's proof style [7], used by some of those papers, it is impossible to infer that from the informal security definitions. We address this gap by providing a formal, general definition of *Message Recognition Protocol (MRP)*.

Our definition focuses on the *upward* interface of the protocol in the protocol stack, as it will be used by higher layers, separately from the trivial *downward* interfaces of the protocol (plain, insecure "transmit" and "receive"). For simplifying the notation, we chose to use the formalism of Interactive Turing Machine (ITM), as used previously in the literature (such as in [4]), which are randomized and encapsulate the reception and transmission of the underlying messages.

[4] Here the synchronization lasts during all of the session. This should not be confused with the synchronization discussed in the context of OC (Section 2.2), which lasts for short time in the initialization, and thus is a simpler task.

Definition 2 (Message Recognition Protocol). *Let $k \in \mathbb{N} \cup \{\infty\}$. A k-Message Recognition Protocol (k-MRP) is a tuple of interactive algorithms $\Pi = (A_I, B_I, A_D, B_D)$. The initialization-phase algorithms A_I and B_I exchange messages with each other.*[5] *A_I outputs the initial state s_0^A for the data phase. Similarly, B_I outputs s_0^B. The data phase algorithms A_D and B_D can exchange messages with each other over the underlying insecure channel; we call these* physical messages. *At iteration i of the data phase, A_D receives as input: (1) a state s_{i-1}^A, and (2) a string to transmit m_i^A called the* logical message. *A_D outputs: (1) a new state s_i^A; (2) a binary value of acceptance/rejection of the session b_i^A, and (3) a logical message that was allegedly sent by the other B_D machine $\widehat{m}_i^B$; (4) . Then, in that iteration, B_D receives as input s_{i-1}^B and m_i^B, and outputs s_i^B, b_i^B and $\widehat{m}_i^A$. Without adversary interference, $m_i^A = \widehat{m}_i^A$ and $m_i^B = \widehat{m}_i^B$ for all $k \geq i \in \mathbb{N}$.*

Adversarial model. The adversary is modeled as stateful interactive algorithm M which can passively eavesdrop on A_I and B_I, and actively corrupt communication between A_D and B_D. The adversary cannot observe or corrupt the local state maintained by A (i.e., A_I and A_D) or B (i.e., B_I and B_D). To model the eavesdropping on the initialization step, we let the initial state s_0^M of M be the whole transcript of A_I and B_I. Thereafter, the state of M, denoted s_i^M, is updated during the execution of A_D and B_D.

M receives as input a state s_{i-1}^M and returns as output a new state s_i^M. For simplicity, we define that running a protocol Π in the presence of an adversary M is done by passing the whole message exchanged between A_D and B_D through M, so the adversary can perform any MITM operation on their communication (change, delay, reorder or instantiate new messages).

Protocol execution. Each party receives the logical message to be sent from the higher layer in the network protocol stack. We model it here by the functions GenMsg^P ($P \in A, B$), which receives as input all of the messages received till now from the other party.[6] The execution defined in Definition 3 using Algorithm 1, and demonstrated in Figure 4.1. We use the hat notation $\widehat{(\cdot)}$ to represent the corresponding received value, which might be corrupted by the adversary.

Definition 3. *Let $\Pi = (A_I, B_I, A_D, B_D)$ be an MRP. Let M be an adversary. The k-*execution *of these instances of the protocol alongside the adversary M is defined by running Algorithm 1, and denoted by $\mathsf{EXEC}_{\Pi,M}^k$.*[7]

[5] Note that some protocols limit the number of logical messages that can be sent (because of implementation issues). This number is represented here by k. To handle this case, we require the protocol instance to output $b_i^P = 0$ when it reaches this limit.

[6] Alternative ways to model the message choice are discussed in Section 5.

[7] In the first iteration ($i = 1$), since the parties have not sent yet any message, A_D and B_D have no message for output (m_0^A and m_0^B are not defined). We define m_0^A and m_0^B to be the empty message, and expect A_D and B_D to output it in this iteration.

Algorithm 1 Adversarial game

1. Run A_I and B_I, and let $\left(s_0^A\right) \leftarrow A_I()$ and $\left(s_0^B\right) \leftarrow B_I()$.
2. Denote by t the whole transcript of the execution of A_I and B_I.
3. Start running A_D, B_D and M.
4. $s_{0,B}^M \leftarrow t$.
5. For i from 1 to k do:
   (a) Run M: $\left(s_{i-1,A}^M\right) \leftarrow M(s_{i-1,B}^M)$
   (b) $m_i^A \leftarrow \mathsf{GenMsg}^A\left(m_0^B, \ldots, m_{i-1}^B\right)$
   (c) $\left(s_i^A, b_i^A, \hat{m}_i^B\right) \leftarrow A_D\left(s_{i-1}^A, m_i^A\right)$
   (d) If $b_i^A = 1$ and $m_{i-1}^B \neq \hat{m}_{i-1}^B$, return win. Else if $b_i^A = 0$, return fail.
   (e) Run M: $\left(s_{i,B}^M\right) \leftarrow M(s_{i-1,A}^M)$
   (f) $m_i^B \leftarrow \mathsf{GenMsg}^B\left(m_0^A, \ldots, m_i^A\right)$
   (g) $\left(s_i^B, b_i^B, \hat{m}_i^A\right) \leftarrow B_D\left(s_{i-1}^B, m_i^B\right)$
   (h) If $b_i^B = 1$ and $m_{i-1}^A \neq \hat{m}_{i-1}^A$, return win. Else if $b_i^A = 0$, return fail.
6. Return fail.

Definition 4 (Non-chosen-message secure k-MRP). *Let $\Pi = (A_I, B_I, A_D, B_D)$ be an k-MRP. Let $k \in \mathbb{N} \cup \{\infty\}$. We say that Π is a* non-chosen-message secure k-MRP *if for any $n \in \mathbb{N}$, for any k' s.t. $k \geq k' = K(n)$, where K is any polynomial function, any functions GenMsg^A and GenMsg^B, and any efficient (PPT) adversary M, $\Pr\left[\mathsf{EXEC}_{\Pi,M}^{k'} = \mathsf{win}\right] = \mathsf{negl}(n)$.*

Delay. Some approaches to optimize the overhead of the protocol (in terms of communication, memory and computation) use information across iterations. For that, these data phase ITMs (A_D and B_D) need to buffer the messages received and verify the correctness of the messages only in future iterations. This delay can be captured by our non-chosen-message security definition by modifying the data phase ITMs to return the buffered message (instead of the current), and the execution algorithm (Algorithm 1) to verify the correctness of these buffered messages. Definition 5 formalizes these modifications, by introducing the delay parameter $d \in \mathbb{N}^0$. Note that protocols with delay $d > 0$ can be easily modified to hold the original non-chosen-message security definition (Definition 4) with $d = 0$, by sending a dummy empty logical messages d times after each logical message sent. (Of course, this modification influence the communication, memory and computation overhead). In Section 6 we show our delayed secure MRP with $d = 1$, and in Section 7 we proof that this protocol satisfies Definition 5.

Definition 5. [Delayed non-chosen-message secure k-MRP] Delayed non-chosen-message secure k-MRP *with delay $d \in \mathbb{N}^0$ is defined similarly to Definition 4, with the following modifications: (1) The data phase ITMs, A_D and B_D, are modified to return the $\widehat{m}_{i-d}^B$ and $\widehat{m}_{i-d}^A$, respectively*[8]*. (2) In Algorithm 1, the correctness check of the messages, $m_i^A \neq \hat{m}_i^A$ and $m_i^B \neq \hat{m}_i^B$, are replaced by $m_{i-d}^A \neq \hat{m}_{i-d}^A$*

[8] For brevity, we assign $m_j^P = \lambda$ for all $1 > j \in \mathbb{N}$, $P \in \{A, B\}$.

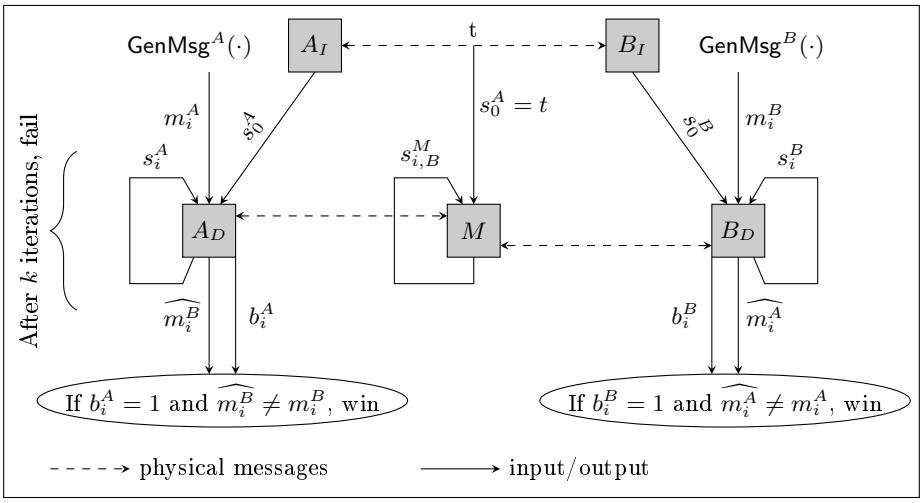

Fig. 4.1. Demonstration of MRP execution algorithm (Algorithm 1).

and $m_{i-d}^B \neq \hat{m}_{i-d}^B$, respectively. (3) In Algorithm 1, the logical messages generation of A_D is modified to be generated by $\mathsf{GenMsg}^A\left(m_0^B, \ldots, m_{i-1-d}^B\right)$*, and the logical messages generation of B_D is modified to be generated by* $\mathsf{GenMsg}^B\left(m_0^A, \ldots, m_{i-d}^A\right)$.

4.2 General MRP framework

In the following, we describe a unified framework that is common to the various MRP protocols in the literature.

Stateful protocols. All of the proposals of an MRP protocol are stateful. That means that each party maintains and updates information about the session. In these protocols, the state of each party consists of (1) a transient *key* that is secret to that party; (2) authenticated non-secret commitment to some secret information held by the other party; and (3) possibly other information, based on the specific MRP. The protocols proceed by iterations. During each iteration, each party can transfer one logical message authentically to the other party, while opening the commitment and revealing the key in the old state, and updating the state to contain a new commitment and a new key. The protocols vary in the way they update the state. There are two basic approaches.

The first approach is based on hash chains [35, 19, 8, 22]. Let $h : \{0,1\}^* \to \{0,1\}^n$ be a CRH. Hash-chains of length k are series of secrets of the form $a_i = h(a_{i-1})$, $\forall 0 < i < k$, and $a_0 \leftarrow \{0,1\}^n$. In this approach, during the initialization phase, each party generates an hash-chain, a_i and b_i respectively. The i-th state of party A contains as a key the index of the current chain's item a_{k-i} and as the commitment on the current key of the other party the value of b_{k-i+1}. The state of party B is symmetrically similar. From the preimage and collision resistance of CRH-s, knowing the commitment to a key does not provide the adversary to ability to forge a key in this chain. The main disadvantage of

this approach is the need to hold the whole\part of the chain of length k in the memory. Choosing small k results in significant limitation on the messages to transmit, and large k results in memory waste. However, it seems to be easier to prove the correctness of protocol based on such approach.

The second approach generates a new state "on-the-fly" [1, 21]. The keys are not generated in advance during the initialization phase. Instead, during each iteration, each party x generates a new key, computes a commitment on that key, and sends the commitment to the other party y. The key is revealed only after verifying that y received (but not yet accepted) *any* message of that iteration. For that, upon receiving *any* message, y should reveal a committed secret of his own, and x should verify that the secret matches this commitment. The commitments for the next iteration is added to the current iteration's messages. The authenticity of the commitment of the next iteration is implied by the authenticity of the message of the current iteration. As oppose to the former approach, in this one the parties are not limited on the number of messages to send/receive, and should not store many keys in advance.

Phases. Each protocol consists of two phases. The *initialization phase* is performed once at the beginning of each session and usually lasts for short time. The main mission of this phase is to established initial state for both parties. The phase basically consists of the following steps for each party: (1) generation of secret(s); (2) generation of commitment(s) on the secret(s); (3) authenticated exchange of the commitment on a secret; (4) completion of initial state establishment. It is possible to replace the transmission of this authenticated data by our suggestion of oblivious comparison of Section 2.2, as we explain later in this section. If this phase finishes correctly (that means, the adversary does not changed the authenticated data), the parties move to the next phase.

The *data phase* consists of the iterations mentioned above, and lasts till the end of the session. The main mission of this phase is to perform the transmission of logical messages between the parties. The general structure of each iteration consists of the following general steps for each party: (1) transmission of a new message with some authentication data about it; (2) transmission of the current secret information (and thus making it public); (3) receiving of the current secret information of the other party and authenticating it using the authenticated commitment; (4) receiving a new message from the other party and authenticating it by similar way[9]; (5) updating the state, as discussed above.

Table 4.2 compares existing MRPs .

4.3 Discussion

Constructing MAP from MRP. Recall that MRP guarantees merely that incoming messages in the MRP data phase have been sent by the same party as in the MRP initialization phase; but does not guarantee the identity of that party (so MRP is vulnerable to MITM attack during initialization). However,

[9] In case of "on-the-fly" approach of state management, the party should receive a new commitment, and it should be authenticated in the same way of the message itself.

	Limited #messages	cryptographic primitives	Proved under	Notes
Guy Fawkes [1]	no	hash	no proof	
Remote User Authentication [25]	no	MAC	no proof	high commnic.
ZCK [35, 13]	yes	OWF	insecure [19]	
Jane Doe [19]	yes	MAC, hash	nonstandard assumptions	
Goldberg et al [8]	yes	MAC, hash	nonstandard assumptions	
Mashatan and Stinson [21]	no	hash	nonstandard assumptions	Resync flawed [26]
Mashatan and Vaudenay [22]	yes	MAC, PRF	standard assumptions	
PEBIUS (this paper)	no	MAC, hash	nonstandard assumptions / random oracle	bidirectional

Fig. 4.2. Comparison of several MRPs. The "hash" primitive refers to hash functions with either well-defined but nonstandard properties, undefined properties, or modeled as random oracles (see "Proved under").

by combining MRPs with Oblivious-Comparison (Section 2.2), we can achieve MAP-like protocol, as follows. Both parties executes the initialization phase of the MRP over the insecure channel. Thereafter, OC is used to authenticate the conversation in the MRP initialization phase, to ensure the correct parties are involved. Subsequently, in the data phase, the exchanged messages are assured to be generated by same (hence, correct) parties as in the initialization phrase.

MRP-based MAPs vs PKC-based MAPs. Table 4.3 quantitatively compares an MRP-based MAP (using the recent MRP of [22]) to PKC-based MAPs and some other alternatives.

Synchronization. MRPs are inherently vulnerable to denial-of-service attacks because their states must remain synchronized. Some suggestion for resynchronization process were made [35, 21], but later proved to be flawed [19, 26]. Resynchronization is impossible without public-key cryptography [9].

Bidirectional protocol. We define MRP as bidirectional protocol, since most targeted devices of interest here require authenticated bidirectional communication channel (for example, a remote controller asks a sensor for a command, and the sensor sends back the response). Most of the current MRPs are designed as unidirectional (A sends logical messages, and B receives them). One can convert an existing (unidirectional) MRP into bidirectional one, by instantiating two instances of the protocol at each party: one as A (the sender) and the other as B (the receiver). The drawback of this method is a doubling of the communication and memory overhead. But in some MRPs, state and messages can be shared between the instances, getting bidirectional communication nearly for free. More specifically, one can use the secret information in B (the receiver party), which is used to re-establish a new commitment, to also transfer also a logical message of B; but this must be done and analyzed carefully, since for hash-chain-based protocols it was proven to be insecure [19].

Basis of the protocol	Cryptographic primitives	Cryptographic schemes	Initial communication overhead	Message communication overhead	State (memory) overhead
Digital sig. with CRH	Sig, CRH	Hashed-RSA-2048 SHA-256	$\|\mathsf{pk}\|$ (2048)	$\|\mathsf{d}\|$ (256)	$\|\mathsf{pk}\|$ (2048)
Public-key enc. with MAC	PKE, MAC	RSA-2048 AES-CBC-MAC-128	$\|\mathsf{pk}\| + \|\mathsf{key}\|$ (2176)	$\|\mathsf{tag}\|$ (128)	$\|\mathsf{key}\|$ (128)
One-time sig.	OTS	Lamport [16]	$4\,\|\mathsf{owf}\|\,l$ ($512l$)	$\|\mathsf{key}\|\,l + 2\,\|\mathsf{owf}\|\,l$ ($384l$)	Various
One-time sig. with CRH	OTS, CRH	Lamport [16] SHA-256	$4\,\|\mathsf{owf}\|\,\|\mathsf{d}\|$ (131,072)	$(\|\mathsf{key}\| + 2\,\|\mathsf{owf}\|)\,\|\mathsf{d}\|$ (98,304)	Various
One-time sig. with UOWHF	OTS, UOWHF	Lamport [16] SHA-256	$4\,\|\mathsf{owf}\|\,\|\mathsf{d}\|$ (131,072)	$(\|\mathsf{key}\| + 2\,\|\mathsf{owf}\|)\,\|\mathsf{d}\|$ (98,304)	Various
Recent MRP [22]	PRF, MAC	AES AES-CBC-MAC-128	$\|\mathsf{d}\|$ (128)	$2\,\|\mathsf{idx}\| + 3\,\|\mathsf{d}\| + 2\,\|\mathsf{sk}\|$ (704)	$\|\mathsf{idx}\| + 2\,\|\mathsf{d}\| + \|\mathsf{sk}\|$ (448)

Fig. 4.3. Comparison of our suggestion to the alternatives. OWF is one way function, PKE is public-key encryption, Sig is digital signature scheme, CRH is collision resistant hash function, OTS is one time signature and UOWHF is universal one way hash function. For overhead calculations, $|\mathsf{pk}|$ denotes the length of the public-key, $|\mathsf{owf}|$ denotes the length of the output of the OWF, $|\mathsf{d}|$ denotes the length of the digest of the hashes/PRF, $|\mathsf{key}|$ denotes the length of the secret key of MACs, symmetric-key encryption or secret preimage of OWF, $|\mathsf{tag}|$ denotes the length of the tag of the MAC, $|\mathsf{idx}|$ is an incremental index of the current message to send and l denotes the length of the message. The overhead is the extra communication requires (on top of the logical messages), in bits, when the protocol is instantiated using the specified scheme. Where the Lamport [16] scheme is used, for concreteness we instantiate using AES-128.

5 Chosen-Message Attacks and Composability

In the formalization of the security of MRP, there arises the question of modeling the generation of logical messages. Obviously, these sent messages can depend on the previously-received messages, but they may also depend on additional inputs, perhaps adversarially-controlled, which are not modeled as network communication. For example, consider a sensor which transmits some physical measurement to a remote server. An adversary which has some control over the sensor's environment can thereby affect its logical messages — exceeding the above model and definition.

5.1 Chosen-Message Resistance

The parties A and B are supposed to generate logical messages (provided as inputs to A_D and B_D). In deployment within a larger system, these logical messages are chosen by some application at the higher layer of the network stack. These messages may of course depend on the logical messages reported as received by the MRP, but ideally the sent messages would not depend on the internal execution (i.e., transcript and randomness) of the MRP. However, an adversary may influence the choice of logical messages produced by the high-level application logic, e.g., when that logic contains "quote and forward" functionality or when it depends on external sensors. The behavior of the high-level application, and the extent to which it is affected by adversarial inputs, is potentially complex, and — essentially — unknown to the lower level MRP. Thus, in mod-

eling security of the MRP, we should make the worst case assumption that the *adversary* chooses the logical messages.

Hence, we extend Definition 4 to handle this issue. We model the choice of logical messages by instructing the parties to receive them from the adversary (which can choose them as any efficiently-computable function of the transcript thus far). For that, we modify Definition 4 and Algorithm 1, so the adversary outputs (in addition to the other values) also the logical message *to be sent*, and the adversarial game just forwards it to the respective party as an input.

Definition 6 (Chosen-message secure k-MRP). Chosen-message secure k-MRP *is similar to* non-chosen-message secure k-MRP *(Definition 4), with the following modifications: (1) The adversary have an additional output value, which represents the logical message to be generated by the respective party. In Algorithm 1, this value is named m_i^A for the former call to the adversary, and m_i^B to the latter. (2) In Algorithm 1, the (two) calls of* GenMsg^P *function are removed.*

We can gain the delayed version of the chosen-message version by applying to Definition 6 the same modifications done by Definition 5 to Definition 4. Definition 7 captures this notion.

Definition 7 (Delayed chosen-message secure k-MRP). *The delayed chosen-message secure k-MRP definition is obtained from Definition 4 by applying the modifications of both Definition 5 and Definition 6.*

5.2 Importance of Chosen-Message Security

We now show that chosen-message attacks can indeed break schemes that are secure in the non-chosen-message sense. Intuitively, the adversary can observe the physical messages exchanged between π_A and π_B to learn information about their randomized internal state, and exploit this to maliciously choose logical messages that will interact badly this internal state. Theorem 1 shows that such an attack could exist, even against a protocol that is secure against non-chosen-message adversaries.

Theorem 1. *Assume there exists an MRP π which is chosen-message secure (fulfills Definition 6). Then there exists an MRP π' which is not chosen-message secure, but is non-chosen-message secure (fulfills Definition 4).*

Proof. Consider any MRPs, π that is chosen-message secure by Definition 6. i.e., resistant to MITM even when the logical message are chosen by the adversary. We shall construct a new protocol, π' protocol, which is not secure in this sense, but is still secure by Definition 4 (i.e., without chosen-messages). protocol π' proceeds like π, with the following modification. Whenever a π' instance π'_A is initialized for party A, a random number t, called the *trapdoor,* is drawn and sent verbatim in a physical message to B. Party B simply ignore this message (but the adversary observes it). From this point on, π'_A keeps track of whether it has ever been asked to send a logical message equal to the trapdoor t. Once that happens, π'_A enters "failure mode" and simply output the fixed logical message 0 whenever it receives a physical message.

In the chosen-message case, the adversary (who eavesdropped the trapdoor can then cause A to send the trapdoor as a logical message. Thereafter π'_A will always claim to receive the message 0, which clearly violates security.

However, π' is still secure under Definition 4, since π'_A has a negligible probability of entering failure mode. Indeed, if the adversary cannot choose the logical messages (but can perform any MITM operation), then the logical messages are generated by functions (GenMsg^P) which depend only on the previous received logical messages, and not the randomly chosen trapdoor value. Hence there is negligible probability that π'_A will be asked to send the trapdoor value. □

5.3 Universal Composability Overview

The previous discussion, of the separation between the two definition, was born as a result of careful analysis of recent MRP's security proofs, which revealed the absence of consideration in chosen-message attack. After revealing that, one may wonder if there is more considerations about the protocol's environment that should be made, and how to handle them? Since such an issue applies for many other protocols (even beyond the context of authentication), we decided to look for a more standardized methodology to handle composition of protocols of our concern.

The Universal Composability (UC) methodology can be used for handling this issue. Loosely speaking, UC adopts an indistinguishability approach for proving properties of security protocols in such a way that make it easy to compose these protocols with other protocols, preserving the security properties guaranteed for each protocol alone. Here we give a short introduction to the notion of Universal Composability framework; here we use the UC variant defined in [4, Section 4.4.1], and the terminology and notation of [4] for *interactive Turing machine (ITM)*, and "public delayed output" (which means an output that the ideal functionality generates to a party, and the adversary can read it and delay its delivery).

The adversary in UC is called the *environment* (recall that we regard to the dummy-adversary alternative definition of UC as defined in [4, Section 4.4.1]), which represents whatever is external to the current protocol execution (including other protocols and their adversaries, the human users, physical message transmission, etc.). The environment outputs a binary value (0 or 1). The security is defined by indistinguishability of the output of the environment between two execution models.

In the *real model*, the environment communicated with parties, who executes an instance of the protocol to be proved secure. The parties receive input and produce output to the environment, and exchanges physical messages between themselves via the environment (so these messages may be modified by the environment).

In the *ideal model*, a new principal is introduced, called the *ideal functionality*, $\mathcal{F}_{\mathsf{IDEAL}}$, which can be thought as a trusted third party. $\mathcal{F}_{\mathsf{IDEAL}}$ is essentially an ITM, which communicated securely (providing both secrecy and authenticity) with the parties. $\mathcal{F}_{\mathsf{IDEAL}}$ performs the required task locally, and sends to

each party the result of the protocol's calculations. The actual parties are represented by *dummy parties*, who just provides the input to the ideal functionality $\mathcal{F}_{\mathsf{IDEAL}}$, and receives from it the output. In addition, there is a simulator $\mathcal{S}$, which can communicate with the ideal functionality, and suppose to convince the environment that she works in the real model, rather than the ideal model.

UC realization. A protocol π is said to *realize* an ideal functionality $\mathcal{F}_{\mathsf{IDEAL}}$, if for each environment $\mathcal{E}$, there exists a simulator $\mathcal{S}$ such that the environment can distinguish (by outputting 0 or 1) between execution in the real model and execution in the ideal model, with at most negligible probability. That is, π mimics the operation of the ideal functionality correctly. By defining an $\mathcal{F}_{\mathsf{IDEAL}}$, one is basically defines the security in the UC methodology.

Universal composition. Let π be some arbitrary protocol where the parties make ideal calls to some ideal functionality $\mathcal{F}_{\mathsf{IDEAL}}$. That is, in addition to the standard set of instructions, π may include instructions to provide instances of $\mathcal{F}_{\mathsf{IDEAL}}$ with some input values, and to obtain output values from these instances of $\mathcal{F}_{\mathsf{IDEAL}}$. We call such protocols $\mathcal{F}_{\mathsf{IDEAL}}$-*hybrid protocols*. Now, let ρ be a protocol that UC-realizes $\mathcal{F}_{\mathsf{IDEAL}}$, according to the above definition. Construct the composed protocol π^{ρ} by starting with protocol π, and replacing each invocation of a new instance of $\mathcal{F}_{\mathsf{IDEAL}}$ with an invocation of a new instance of ρ. Similarly, inputs given to an instance of $\mathcal{F}_{\mathsf{IDEAL}}$ are now given to the corresponding instance of ρ, and any output of an instance of ρ is treated as an output obtained from the corresponding instance of $\mathcal{F}_{\mathsf{IDEAL}}$. The universal composition theorem states that running protocol π^{ρ}, with no access to $\mathcal{F}_{\mathsf{IDEAL}}$, has essentially the same effect as running the original $\mathcal{F}_{\mathsf{IDEAL}}$-hybrid protocol π.

5.4 Defining UC-secure MAP

Another methodology to handle this issue is Universal Composability (UC) [4]. Security definitions in the UC style are guaranteed to "compose", i.e., security can be analyzed for a single instance of the protocol, but is guaranteed to hold even when the protocol runs concurrently with other instances of itself and of other protocols (a property which is nontrivial, and often false for other definition styles). This includes, in particular, the case where inputs to the protocol are affected by the adversary, i.e., chosen-message attacks.

In the UC framework, security is defined by specifying an *ideal functionality*, that describes the correct functionality of the protocol. Proving that a protocol satisfies such security definition is done by proving that the protocol and the ideal functionality are *indistinguishable* to the surrounding environment (including the adversary), and thus the latter cannot gain advantage, beyond what's permitted by the ideal functionality, from corrupting the protocol's execution. Here we give the security definition for multiple message authentication in UC, by defining the ideal functionality, and show how MRP can be combined with OC to provide UC-secure message authentication.

Ideal functionality for multiple message authentication. An ideal functionality for single-message authentication protocol, called $\mathcal{F}_{\mathsf{AUTH}}$, was given in [4]. That ideal functionality transmits a single message from one party to the

other. Multiple messages can be transferred with authentication, and bidirectionally, by composing that ideal functionality multiple times (which is indeed secure, by the universal composition theorem of [4, Section 2.3]). However, in our context this yields an impractical protocol, since every session of the protocol requires an oblivious comparison (involving the human user) in the initialization phase. To allow the results of a single OC to be stored and reused for multiple messages, we extend the ideal functionality into a new one, $\mathcal{F}_{\mathsf{AUTH}^k}$, capturing message authentication protocol that are bidirectional and transmit up to k back-and-forth messages (where $k \in \mathbb{N} \cup \{\infty\}$).[10] Basically, $\mathcal{F}_{\mathsf{AUTH}^k}$ operates much like $\mathcal{F}_{\mathsf{AUTH}}$ of [4], but is extended to multiple messages, as follows.

Definition 8 ($\mathcal{F}_{\mathsf{AUTH}^k}$). *Let $k \in \mathbb{N} \cup \{\infty\}$. The ideal functionality $\mathcal{F}_{\mathsf{AUTH}^k}$ is defined as follows.*

1. *For rounds $i = 1, 2, 3, \ldots, k$ do:*
   (a) *Await an input $(\mathsf{Send}, sid, B, m_i^A)$ from party A, and generate a public delayed output $(\mathsf{Sent}, sid, A, m_i^A)$ to B.*
   (b) *Await an input$(\mathsf{Send}, sid, B, m_i^B)$ from party B, and generate a public delayed output $(\mathsf{Sent}, sid, B, m_i^B)$ to A.*
2. *Upon receiving a message $(\mathsf{Abort}, sid, P, m)$ from the adversary, where $P \in \{A, B\}$, send $(\mathsf{Abort}, sid, P, m)$ to the other party, and halt.*
3. *Upon receiving $(\mathsf{Corrupt\text{-}party}, A, sid, m')$ from the adversary, if the $(\mathsf{Sent}, A, sid, m)$ output is not yet delivered to B, then output $(\mathsf{Sent}, A, sid, m')$ to B.*
4. *Upon receiving $(\mathsf{Corrupt\text{-}party}, B, sid, m')$ from the adversary, if the $(\mathsf{Sent}, B, sid, m)$ output is not yet delivered to A, then output $(\mathsf{Sent}, B, sid, m')$ to A.*

The two last lines in Definition 8 capture the notion of party corruption [4]. Note that party corruption is not a significant issue for us, since we do not deal with secrecy, but we included that notion inside the definition for completeness.

Ideal functionality for oblivious comparison . In order to realize the message authentication protocols we discussed in this paper to the ideal functionality $\mathcal{F}_{\mathsf{AUTH}^k}$, we need to use the oblivious-comparison primitive, that compares bit strings between two parties (see Section 2.2). To express this in the UC framework, we define its ideal functionality, $\mathcal{F}_{\mathsf{OC}}$, as follows[11].

Definition 9 (Oblivious comparison ideal functionality: $\mathcal{F}_{\mathsf{OC}}$).

1. *Upon receiving $(\mathsf{Send}, sid, A, x^A)$ from A and $(\mathsf{Send}, sid, B, x^B)$ from B:*
   (a) *If $x^A = x^B$ then generate public delayed output $(\mathsf{Sent}, sid, \mathsf{OC}, \mathsf{Equal})$ to B and $(\mathsf{Sent}, sid, \mathsf{OC}, \mathsf{Equal})$ to A.*
   (b) *else generate public delayed output $(\mathsf{Sent}, sid, \mathsf{OC}, \mathsf{Unequal})$ to B and $(\mathsf{Sent}, sid, \mathsf{OC}, \mathsf{Unequal})$ to A.*

[10] We support a limit k on the number of logical messages transferred by ideal functionality, since many existing MRPs support only an a priori bounded number of messages. Our "PEBIUS" MRP, in Section 6, supports $k = \infty$.

[11] We omitted the explicit corrupting handling from $\mathcal{F}_{\mathsf{OC}}$, because it is instantiated out-of-band (e.g. by human, who see the LEDs).

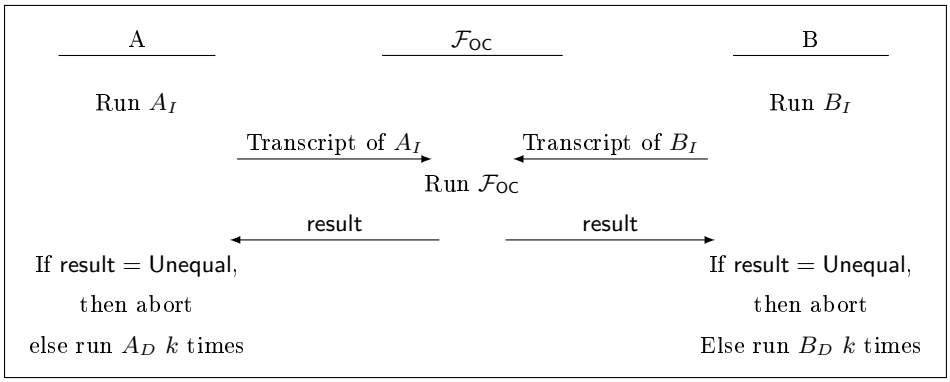

Fig. 5.1. The hybrid protocol π_{AUTH^k}.

5.5 Attaining UC-secure MAP

Analogously to Section 4, we show here how to create an MAP from an MRP, given access to $\mathcal{F}_{\mathsf{OC}}$, the ideal functionality of oblivious comparison. That is, how to create a $\mathcal{F}_{\mathsf{OC}}$-hybrid protocol that realizes the $\mathcal{F}_{\mathsf{AUTH}^k}$ ideal functionality.[12] Given an k-MRP, $\pi_{\mathsf{MRP}^k} = (A_I, B_I, A_D, B_D)$, the corresponding $\mathcal{F}_{\mathsf{OC}}$-hybrid protocol π_{AUTH^k} is specified in Figure 5.1.

Theorem 2. *Let $\pi_{\mathsf{MRP}^k} = (A_I, B_I, A_D, B_D)$ be a chosen-message secure k-MRP (Definition 6). Let π_{AUTH^k} be its hybrid protocol, as specified in Figure 5.1. Then π_{AUTH^k} UC-realizes $\mathcal{F}_{\mathsf{AUTH}^k}$ in the $\mathcal{F}_{\mathsf{OC}}$-hybrid model.*

Proof. We need to prove that if for each chosen-message adversary M, M cannot break the adversarial game of Definition 6, then there exists a simulator $\mathcal{S}$ such that for each environment $\mathcal{E}$, $\mathsf{EXEC}_{\mathcal{F}_{\mathsf{AUTH}^k},\mathcal{S},\mathcal{E}} \approx \mathsf{EXEC}_{\pi,\mathcal{E}}$.

We shall construct the simulator $\mathcal{S}$ as an ITM that interacts with an environment $\mathcal{E}$ and the ideal functionality $\mathcal{F}_{\mathsf{AUTH}^k}$ by emulating the instances of the protocol, as follows (see Figure 5.2). On startup, $\mathcal{S}$ initializes two instances of the protocol: π_A, π_B. The exchange of physical messages between π_A and π_B is done by communicating with $\mathcal{E}$ (so $\mathcal{E}$ can modify these messages). Whenever $\mathcal{E}$

[12] Formally, there is a difficulty here: the oblivious-comparison protocols Section 2.2, involving physical devices and human beings, are not built of "interactive Turing machines" as necessitated by the definitions of [4]. To model this rigorously, one may appeal to the extended Church-Turing thesis about the polynomial-time equivalence of the aforementions, i.e., to the existence of polynomial-time Turing-machines that simulate the physical setup. Alternatively, one may note that the relevant properties of the oblivious-comparison protocol are its external interface (and indeed interface via ITM tapes is in principle equivalent to the digital interface of the electronic devices) and the fact that its computational power is inadequate for breaking specific cryptographic assumptions (which we assume holds for the physical devices and their human operator).

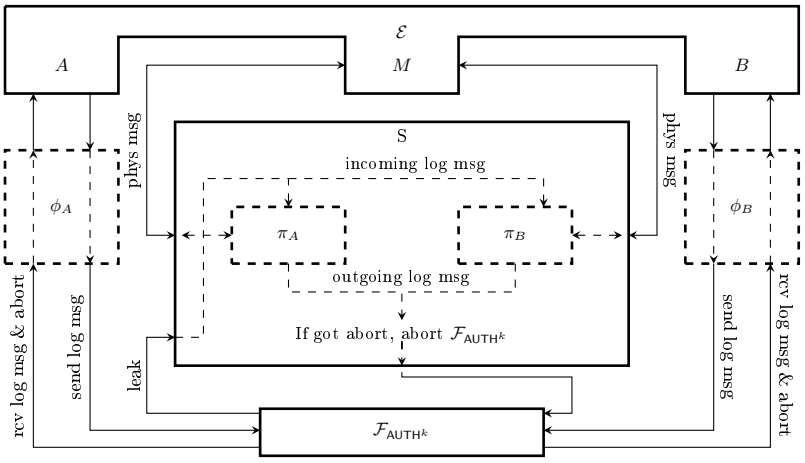

Fig. 5.2. The simulator $\mathcal{S}$.

sends a logical message to $\mathcal{F}_{\mathsf{AUTH}^k}$ (via a dummy party ϕ_A, ϕ_B), $\mathcal{F}_{\mathsf{AUTH}^k}$ leaks the message to $\mathcal{S}$, and upon receiving such a leaked logical message sent by $P \in \{A, B\}$, S forwards that message to π_P (as a logical message to be sent by P). Consequentially, π_P generates physical message(s) and transmits them; these are routed by S into $\mathcal{E}$. When π_P outputs a logical message (supposedly received from the other party), $\mathcal{S}$ tells $\mathcal{F}_{\mathsf{AUTH}^k}$ to release the delayed logical message (even in the case of wrong logical message). When π_P outputs abort, $\mathcal{S}$ sends $(\mathsf{abort}, sid, P, m)$ to $\mathcal{F}_{\mathsf{AUTH}^k}$.

By proving that the simulator $\mathcal{S}$ indeed succeeds in fooling the environment (i.e., simulating the real model), we infer that $\mathsf{EXEC}_{\mathcal{F}_{\mathsf{AUTH}^k},\mathcal{S}',\mathcal{E}} \equiv \mathsf{EXEC}_{\pi,\mathcal{E}}$, as desired. □

Let us now argue that the simulator $\mathcal{S}$ indeed succeeds in fooling the environment (i.e., simulating the real model). To do so, let us define another simulator $\mathcal{S}'$, similarly to $\mathcal{S}$ but for the following modification. When $\mathcal{S}'$ invokes π_A and the latter produces a logical message $\hat{m}_i^B$ allegedly sent by B, instead of ignoring the content of $\hat{m}_i^B$ that message, $\mathcal{S}'$ checks whether this message is correct (i.e., matches m_i^B that was earlier leaked by $\mathcal{F}_{\mathsf{AUTH}^k}$). If $\hat{m}_i^B \neq m_i^B$ then $\mathcal{S}'$ will not ignore this (as S does) but instead forward the incorrect message $\hat{m}_i^B$ to $\mathcal{E}$ (bypass $\mathcal{F}_{\mathsf{AUTH}^k}$) and moreover block (i.e. delay the sending of the message forever) the sending of the correct message m_i^B from $\mathcal{F}_{MRP}$ to $\mathcal{E}$. The analogous change is also made in the other direction (when π_B outputs $\hat{m}_i^A$ allegedly sent from A). Note that $\mathcal{S}'$ is not a valid simulator as required by UC, because the UC ideal model does not allow the simulator to talk directly to the environment; but as we will show in Lemma 2, $\mathcal{S}'$ behaves essentially the same as $\mathcal{S}$.

Let us first show that the ideal model with simulator $\mathcal{S}'$ is essentially the same as the chosen-message secure MRP model (Lemma 1), and thus essentially

the same as the UC real model (Corollary 1). For that, we define the notion of *view,* as follows.

Definition 10 (View). *Let π be an MRP, and $\mathcal{S}'$ as defined above.*

- *Define the view $V^{\mathcal{F}_{\mathsf{AUTH}^k}}_{\mathcal{E},\mathcal{S}'}$ of an execution of $\mathcal{F}_{\mathsf{AUTH}^k}$ in the UC ideal model with environment $\mathcal{E}$ and simulator $\mathcal{S}'$ as the concatenation of: (1) the physical (but not logical) messages that M receives; (2) the logical messages the each dummy party receives as input and (3) the randomness of the environment $\mathcal{E}$.*
- *Define the view V^{π}_{M} of an adversary M in the chosen-message definition as the concatenation of: (1) the physical (but not logical) messages that M receives; (2) the logical messages the each party receives as input and (3) the randomness of the adversary M.*

Note that the view definition should not include the physical/logical messages that outputted to the environment/adversary, because it is given by the randomness and the other parts of the view.

Lemma 1. *Let π be an MRP, and $\mathcal{S}'$ as defined above. For each environment $\mathcal{E}$, there exists a chosen-message adversary $M^{\mathcal{E}}$, such that $V^{\mathcal{F}_{\mathsf{AUTH}^k}}_{\mathcal{E},\mathcal{S}'} \equiv V^{\pi}_{M^{\mathcal{E}}}$.*

Proof. Let $\mathcal{E}$ be a UC environment. We define the chosen-message adversary $M^{\mathcal{E}}$ to behave as $\mathcal{E}$: $M^{\mathcal{E}}$ operates in the same way as $\mathcal{E}$ does with regard to exchanging the physical messages, operates in the same way $\mathcal{E}$ does with regard to generation of logical messages of A, B respectively.

Consider the execution of π in the chosen-message model in the presence of the adversary $M^{\mathcal{E}}$, versus the execution of π in the UC ideal model with $\mathcal{E}$ and $\mathcal{S}'$, when fixing identical randomness for the analogous principals (e.g., the π_A ITI in the chosen-message execution vs. the π_A ITI inside S' in the UC ideal execution). We shall show that the two executions essentially identical, i.e., the same messages are exchanges between the analogous principals. This invariant follows from the essentially identical handling of all messages, as follows.

Indeed, in both models, the logical messages are generated in the same way (by $\mathcal{E}$ in the UC ideal model, and similarly by definition, by $M^{\mathcal{E}}$ in the chosen-message model), and are received as input by π_A and π_B. The intervention in the physical message communication between π_A and π_B is also done in the same way (by $\mathcal{E}$ in the UC ideal model, and similarly by definition, by $M^{\mathcal{E}}$ in the chosen-message model). Therefore, π_A and π_B produces the same physical messages and outputs in both executions.

Let us verify that the outputs of π_A are handled identically in both model (the analysis for π_B is analogous). There are three cases. (1) π_A outputs abort in the chosen-message model, and A aborts. In the ideal model, $\mathcal{S}'$ sends abort to $\mathcal{F}_{\mathsf{AUTH}^k}$, which causes ϕ_A to abort as well. (2) π_A outputs the correct logical message in the chosen-message model. In the ideal model, $\mathcal{S}'$ tells $\mathcal{F}_{\mathsf{AUTH}^k}$ to send the correct logical message the A, so A also receives the correct logical message. (3) π_A outputs a wrong logical message in the chosen-message model. In the

ideal model, $\mathcal{S}'$ understands that the message is wrong, and bypasses $\mathcal{F}_{\mathsf{AUTH}^k}$ so A receives the wrong message as well. □

Let π be an MRP, and $\mathcal{E}$ be an environment. Let $M^{\mathcal{E}}$ be the chosen-message adversary given by Lemma 1. In the real model of UC, the parties run the protocol instances π_A and π_B. Note that the environment is involved in their physical messages exchange, and controls the generation of logical messages, exactly as in the chosen-message execution. Thus, the chosen-message execution of π with $M^{\mathcal{E}}$ and the execution in the real model of UC of π are essentially equivalent, because their respective views are equivalent, by Lemma 1 (the environment $\mathcal{E}$ and the adversary $M^{\mathcal{E}}$ have the same view distribution, so they have the same output distribution).

It follows, in particular, that the ideal-model execution with $\mathcal{S}'$ is equivalent to the real-model UC execution:

Corollary 1. $\mathsf{EXEC}_{\mathcal{F}_{\mathsf{AUTH}^k},\mathcal{S}',\mathcal{E}} \equiv \mathsf{EXEC}_{\pi,\mathcal{E}}$

There remains to show that $\mathcal{S}'$ is essentially the same as $\mathcal{S}$:

Lemma 2. $\mathsf{EXEC}_{\mathcal{F}_{\mathsf{AUTH}^k},\mathcal{S},\mathcal{E}} \approx \mathsf{EXEC}_{\mathcal{F}_{\mathsf{AUTH}^k},\mathcal{S}',\mathcal{E}}$

Proof. Let $\mathsf{BAD}_{\pi}^{\mathcal{E}}$ be the event that during the execution of $\mathcal{E}$ against S and $\mathcal{F}_{\mathsf{AUTH}^k}$, π_A or π_B (which are emulated inside $\mathcal{S}$) ever output wrong logical message. The probability that $\mathcal{E}$ will output 1 when running with $\mathcal{S}$ and $\mathcal{F}_{\mathsf{AUTH}^k}$ is, by Bayes's theorem:

$$\begin{aligned}
&\Pr\left[\mathsf{EXEC}_{\mathcal{F}_{\mathsf{AUTH}^k},\mathcal{S},\mathcal{E}} = 1\right] = \\
&\quad \Pr\left[\mathsf{EXEC}_{\mathcal{F}_{\mathsf{AUTH}^k},\mathcal{S},\mathcal{E}} = 1 | \mathsf{BAD}_{\pi}^{\mathcal{E}}\right] \Pr\left[\mathsf{BAD}_{\pi}^{\mathcal{E}}\right] \\
&\quad + \Pr\left[\mathsf{EXEC}_{\mathcal{F}_{\mathsf{AUTH}^k},\mathcal{S},\mathcal{E}} = 1 | \mathsf{BAD}_{\pi}^{\mathcal{E}}\right] \Pr\left[\mathsf{BAD}_{\pi}^{\mathcal{E}}\right]
\end{aligned}$$

Similarly for S':

$$\begin{aligned}
&\Pr\left[\mathsf{EXEC}_{\mathcal{F}_{\mathsf{AUTH}^k},\mathcal{S}',\mathcal{E}} = 1\right] = \\
&\quad \Pr\left[\mathsf{EXEC}_{\mathcal{F}_{\mathsf{AUTH}^k},\mathcal{S}',\mathcal{E}} = 1 | \mathsf{BAD}_{\pi}^{\mathcal{E}}\right] \Pr\left[\mathsf{BAD}_{\pi}^{\mathcal{E}}\right] \\
&\quad + \Pr\left[\mathsf{EXEC}_{\mathcal{F}_{\mathsf{AUTH}^k},\mathcal{S}',\mathcal{E}} = 1 | \mathsf{BAD}_{\pi}^{\mathcal{E}}\right] \Pr\left[\mathsf{BAD}_{\pi}^{\mathcal{E}}\right]
\end{aligned}$$

Observe that $\mathcal{S}$ and $\mathcal{S}'$ behaves identically outside the event $\mathsf{BAD}_{\pi}^{\mathcal{E}}$, so above, the two last terms are equal. Thus the difference in the probability of the environment's output is at most:

$$\begin{aligned}
&|\Pr\left[\mathsf{EXEC}_{\mathcal{F}_{\mathsf{AUTH}^k},\mathcal{S},\mathcal{E}} = 1\right] - \Pr\left[\mathsf{EXEC}_{\mathcal{F}_{\mathsf{AUTH}^k},\mathcal{S}',\mathcal{E}} = 1\right]| = \\
&|\Pr\left[\mathsf{EXEC}_{\mathcal{F}_{\mathsf{AUTH}^k},\mathcal{S},\mathcal{E}} = 1 | \mathsf{BAD}_{\pi}^{\mathcal{E}}\right] - \Pr\left[\mathsf{EXEC}_{\mathcal{F}_{\mathsf{AUTH}^k},\mathcal{S}',\mathcal{E}} = 1 | \mathsf{BAD}_{\pi}^{\mathcal{E}}\right]| \\
&\cdot \Pr\left[\mathsf{BAD}_{\pi}^{\mathcal{E}}\right] \quad \leq \quad 1 \cdot \Pr\left[\mathsf{BAD}_{\pi}^{\mathcal{E}}\right]
\end{aligned}$$

There remains to show that $\Pr\left[\mathsf{BAD}_{\pi}^{\mathcal{E}}\right] = \mathsf{negl}(n)$.

Let $M^{\mathcal{E}}$ be the adversary given by Lemma 1. By the premise that π satisfies Definition 4, $M^{\mathcal{E}}$ cannot convince a party to receive a wrong logical message with non-negligible probability, so $\Pr\left[\mathsf{BAD}_{\pi}^{\mathcal{E}}\right] = \mathsf{negl}(n)$. Now, because the views are equivalent ($V_{\mathcal{E},\mathcal{S}'}^{\mathcal{F}_{\mathsf{AUTH}^k}} \equiv V_{M^{\mathcal{E}}}^{\pi}$), it follows that $\mathsf{EXEC}^{\mathcal{F}_{\mathsf{AUTH}^k},\mathcal{S},\mathcal{E}} \approx \mathsf{EXEC}^{\mathcal{F}_{\mathsf{AUTH}^k},\mathcal{S}',\mathcal{E}}$. □

Corollary 2. $\mathsf{EXEC}^{\mathcal{F}_{\mathsf{AUTH}^k},\mathcal{S},\mathcal{E}} \approx \mathsf{EXEC}^{\pi,\mathcal{E}}$

6 PEBIUS MRP

We present the *PEBIUS* MRP, a *Pe*rpetual *Bi*directional *UC*-*S*ecure message recognition protocol. PEBIUS is secure under the strongest definitions given above: it is UC-secure and, equivalently, chosen-message secure). Our protocol differs from prior MRPs (see Section 4.2), as follows.

Most prior MRPs [8, 19, 22, 35] used hash chains, which entailed two drawbacks: an a-priori limit on the number of logical messages that can be sent, and expensive storage (or recomputation) of the hash chain. Some also use a trusted third party, or a high network overhead.

To avoid such issues, PEBIUS generate fresh key in each session, as done in [21]. However, PEBIUS is also *inherently* bidirectional and thus more efficient than using a unidirectional MRP in both directions (see Section4.3).

6.1 Cryptographic Primitives for PEBIUS

PEBIUS use the following cryptographic primitives.

Message Authentication Code (MAC). We denote the usage of this function by $\mathsf{MAC}(k, x)$, where k is the private key and x is the (variable length) message to be authenticated. The key generation of the MAC is denoted by $\mathsf{GenKey}()$. One such MAC that we can use is AES-CBC-MAC. We note that this method should deal properly with variable-length messages. Otherwise, it will be vulnerable to message concatenation, as described in [2] .

Hash function. For simplicity here, we model this as a *random oracle* $f(x)$, where x is the input to the functions. In Section 7.3, we discuss the concrete and plausible (though nonstandard) properties requires of the hash function, which are indeed all fulfilled by random oracles. Practically, $f(x)$ would be implemented by a standard cryptographic hash, such as SHA-256.

6.2 The PEBIUS Protocol

We next describe the PEBIUS MRP $\Pi = (A_I, B_I, A_D, B_D)$, which (as shown Section 7) is a delayed chosen-message secure ∞-MRP (Definition 7). Following Theorem 2, Π is also universally composable. Let f be a random oracle and MAC be a message authentication code, as defined above.

Intuitive explanation. The protocol consists of theoretical infinite iterations. During each iteration, each party sends one logical message to the other party using fresh keys (i.e. keys that were not used by earlier logical messages), and a MAC of the logical message. At the beginning of each iteration, each of the

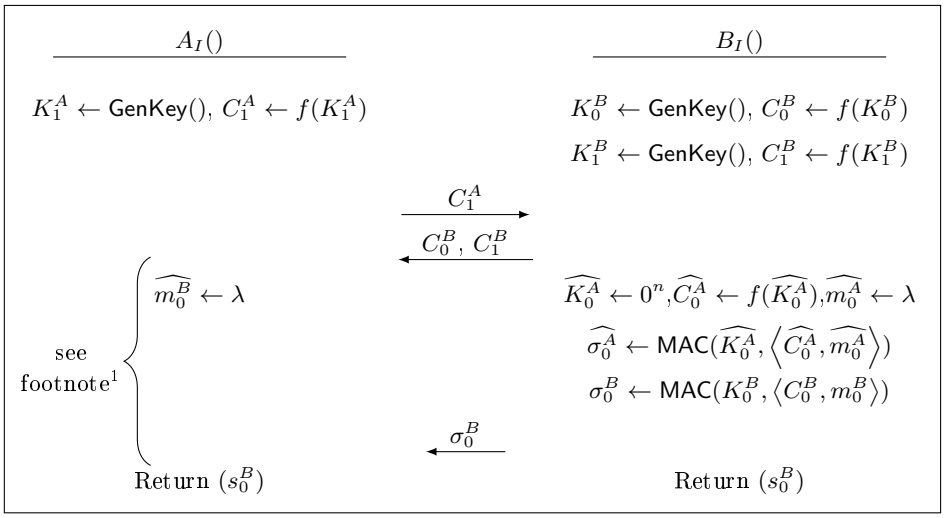

Fig. 6.1. The PEBIUS initialization phase, Π. The state variable s_0^A contains all of the local variables used in A_I; and analogously for B.

parties has a commitment on the key of the other party, and the keys are revealed at the next iteration. As a result, when each of the parties receives the key, he can verify that the key is correct (using the commitment on the key) and that the logical message is correct (using the correct key and the MAC on the logical message). In order to be able to send other logical messages at the following iterations, we need new keys. We achieve that by transferring a commitment on the fresh keys inside the logical message of each iteration.

Initialization phase. The goal of the initialization phase (implemented by A_I and B_I, see Figure 6.1) is to transfer commitments on the first keys to each party. Recall that in MRP we assume that this phase is executed with authenticity (but without confidentiality), so each party can trust these commitments. Each party generates keys: K_1^A of A and K_0^B, K_1^B of B, and commitments on these keys: C_1^A, C_0^B, C_1^B, respectively. We note that during the data phase, A sends the first logical message, and that is the reason of this asymmetric (that B generates two keys and A generates just one). In the following lines, in order to simplify the description of the data phase, we generate some dummy variables to compensate on this asymmetric.

[1] These lines address a technicality in definition of the data phase. The first iteration of the data-phase refers to "previous logical messages" ($\widehat{m_0^A}$, $\widehat{m_0^B}$), "previous key" ($\widehat{K_0^A}$), "previous commitment" ($\widehat{C_0^A}$) and "previous signatures" ($\widehat{\sigma_0^A}$, $\widehat{\sigma_0^B}$) variables, so we define dummy values for these variables. Alternatively (but less elegantly), the first iteration of the data-phase can be modified to be special, in that it does not verify or refer to such past variables.

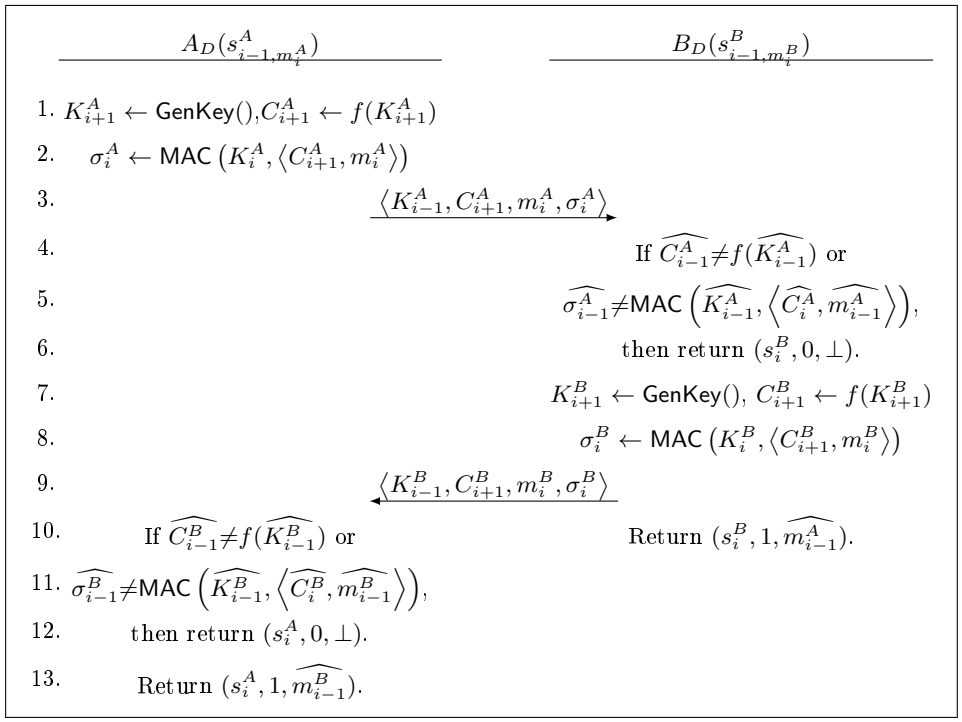

Fig. 6.2. Iteration i ($i >= 1$) of the PEBIUS data phase, Π. The state variable s^A_j ($j \in \mathbb{N}$) contains all of the local variables of party A at the end of the j-th iteration of A_D; and analogously for B.

Data phase. During each iteration of the data phase (i.e., a single execution of A_D and B_D, see Figure 6.2), each party sends one logical message to the other party. Both parties start by generating a fresh new key for the next iteration and a commitment on this key. Then, they use the key of the current iteration to produce a signature on the logical message and the new commitment. Thereafter, A sends the logical message, the new commitment and the signature on them. Upon reception, B responds with symmetrical values, plus the key of the previous iteration. Upon receiving this key, A can recognize that the key, the logical message and the commitment are generated by the previously-negotiated entity (of the initialization phase) and accept the message, or reject, if the verification fails. Then, A sends a key to B, which performs similar verifications.

7 Security Proof of PEBIUS

7.1 Conversations

We define here the notion of matching conversations, in the spirt of the definition in [3]. Basically, a conversation is the transcript of the executed protocol, as seemed by an involved party. Two conversations are matching conversations if

they represents the transcript of the protocol executed by them (one conversation for A, and the other one of B), without interference of any adversary.

Definition 11 (Conversation). *Let* $\mathsf{EXEC}^k_{\Pi,M}$ *be an execution of the protocol. For any party* $P \in A, B$ *we can capture its* conversation *(for this execution) by a sequence:*

$$\mathcal{K} = (\tau_1, \alpha_1, \beta_1), (\tau_2, \alpha_2, \beta_2), \ldots, (\tau_m, \alpha_m, \beta_m).$$

This sequence encodes that at time[13] τ_1 party P was asked α_1 and responded with β_1; and then, at some time $\tau_2 > \tau_1$, party P was asked α_2 and answered β_2; and so forth, until, finally, at time τ_m party P was asked α_m and answered β_m.

Definition 12 (Matching conversations). *Let* $\mathsf{EXEC}^k_{\Pi,M}$ *be an execution of the protocol with* $R = 2\rho - 1$ *moves. Let* $\mathcal{K}$ *and* $\mathcal{K}'$ *be the conversations of* A *and* B*, respectively.*

1. *We say that* $\mathcal{K}'$ *is a* matching conversation *to* $\mathcal{K}$ *if there exist* $\tau_0 < \tau_1 < \cdots < \tau_R$ *and* $\alpha_1, \beta_1, \ldots, \alpha_\rho, \beta_\rho$ *such that* $\mathcal{K}$ *is prefixed by*

   $$(\tau_0, \lambda, \alpha_1), (\tau_2, \beta_1, \alpha_2), (\tau_4, \beta_2, \alpha_3), \ldots, (\tau_{2\rho-4}, \beta_{\rho-2}, \alpha_{\rho-1}), (\tau_{2\rho-2}, \beta_{\rho-1}, \alpha_\rho)$$

   and $\mathcal{K}'$ *is prefixed by*

   $$(\tau_1, \alpha_1, \beta_1), (\tau_3, \alpha_2, \beta_2), (\tau_5, \alpha_3, \beta_3), \ldots, (\tau_{2\rho-3}, \alpha_{\rho-1}, \beta_{\rho-1}).$$

2. *We say that* $\mathcal{K}$ *is a* matching conversation *to* $\mathcal{K}'$ *if there exist* $\tau_0 < \tau_1 < \cdots < \tau_R$ *and* $\alpha_1, \beta_1, \ldots, \alpha_\rho, \beta_\rho$ *such that* $\mathcal{K}'$ *is prefixed by*

   $$(\tau_1, \alpha_1, \beta_1), (\tau_3, \alpha_2, \beta_2), (\tau_5, \alpha_3, \beta_3), \ldots, (\tau_{2\rho-3}, \alpha_{\rho-1}, \beta_{\rho-1}), (\tau_{2\rho-1}, \alpha_\rho, *)$$

   and $\mathcal{K}$ *is prefixed by*

   $$(\tau_0, \lambda, \alpha_1), (\tau_2, \beta_1, \alpha_2), (\tau_4, \beta_2, \alpha_3), \ldots, (\tau_{2\rho-4}, \beta_{\rho-2}, \alpha_{\rho-1}), (\tau_{2\rho-2}, \beta_{\rho-1}, \alpha_\rho).$$

Figure 7.1 illustrates the matching conversation notion.

Definition 13. *Let* $a \leq b \in \mathbb{N}$ *and let* $\mathcal{K}$ *be a conversation. A conversation that consist from* $(b - a + 1)$*-tuples of* $\mathcal{K}$ *starting at the* a*-tuple till the* b*-tuple is a* sub-conversation *of* $\mathcal{K}$*. We denote such conversations by* $\mathcal{K}_{a,b}$*, and use the shorthand* $\mathcal{K}_b = \mathcal{K}_{1,b}$*.*

By convention, for each conversation $\mathcal{K}$ we denote by $\mathcal{K}_0 = \lambda$ the empty conversation. Matching-conversations are defined for conversations. We define a similar notion for sub-conversations:

[13] Here, *time* is any monotonically increasing function of the protocol's progress, e.g., the number of messages sent so far.

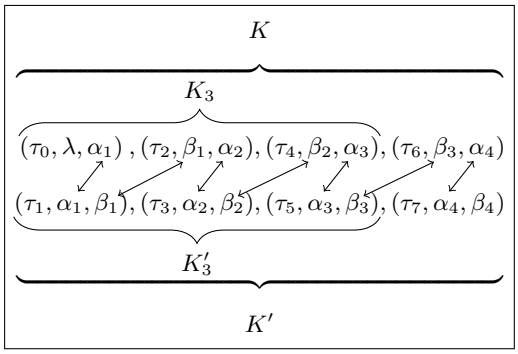

Fig. 7.1. The relations between conversations and sub-conversations. In this figure:

Definition 14. *Let* $\mathsf{EXEC}^k_{\Pi,M}$ *be an execution of the protocol with* $R = 2\rho - 1$ *moves. Let* $\mathcal{K}$ *and* $\mathcal{K}'$ *be the conversations of* A *and* B*, respectively. Let* $b \in \mathbb{N}$*. We say that* $\mathcal{K}$ *is a* matching-sub-conversation *to* $\mathcal{K}'$ *of length* b*, and denote* $\mathcal{K} \overset{b}{\approx} \mathcal{K}'$*, if* $\mathcal{K}$ *is a matching-conversation to* $\mathcal{K}'$ *when setting* $\rho = b+1$ *(and thus,* $R = 2b+1$*).*

In other words, in case A *is the initiator,* $\mathcal{K} \overset{b}{\approx} \mathcal{K}'$ *if* $\mathcal{K}_{b+1}$ *is a matching conversation to* $\mathcal{K}'_b$*. Note that if* A *is the initiator, for* $b = 0$ *the sub-conversations are* $\mathcal{K}_{b+1} = (\tau_0, \lambda, \alpha_1)$ *and* $\mathcal{K}'_b = \lambda$*, so* $\mathcal{K} \overset{0}{\approx} \mathcal{K}'$ *trivially. Figure 7.1 shows the relations between conversations and sub-conversations.*

7.2 The Proof

The proof consists of some groundwork which is done by proving some claims, and it is concluded in Theorem 3. The proof involves an argument by induction on the protocol's iteration counter, i. For clarity, we divide the proof into several claims, to be used in Theorem 3. In the lemma and all of the claims below, we let f be a random oracle and $\mathsf{MAC}(.)$ be a MAC function.

We first note a property of acceptance (outputting a value to s_i^P variables, $P \in A, B$) during the protocol execution. Intuitively, if each of the parties accepted the k-th logical message, then he accepted also all of the earlier logical messages. (The trivial proof is omitted.)

Claim. $\mathsf{EXEC}^k_{\Pi,M}$ be an execution of the protocol. Let $P \in A, B$. Let $j \in \mathbb{N}$. If P_D outputs $s_j^P = 1$, then for each $1 \leq i \leq j$, P_D sets $s_i^P = 1$. □

The essential insight about the protocol is that the following "iteration-invariant" holds in the beginning of each iteration of the parties during the protocol. Intuitively, it says that in the beginning of the i-th iteration, both parties have already generated new secret keys, which have not been sent yet (so the adversary do not know them), but the image of these keys under f is known to the other party. Formally:

Definition 15. *Let* $\mathsf{EXEC}^k_{\Pi,M}$ *be an execution of the protocol. We say that the* iteration-invariant *holds for* i *if at the beginning of the* i *iteration of* A_D*:*

1. A_D *generated* $K_i^A \leftarrow \mathsf{GenKey}()$ *, and* B_D *generated* $K_{i-1}^B \leftarrow \mathsf{GenKey}(), K_i^B \leftarrow \mathsf{GenKey}()$.
2. $\langle C_i^A, C_{i-1}^B, C_i^B \rangle = \left\langle \widehat{C_i^A}, \widehat{C_{i-1}^B}, \widehat{C_i^B} \right\rangle$*, except for negligible probability.*
3. *For each* $K \in \left\{K_i^A, K_{i-1}^B, K_i^B\right\}$*, the only value depending on* K *that* M *received from the parties, up to this time, is* $f(K)$*.*

We will use the following claim in the proof. Intuitively, knowing $f(K)$ does not help the adversary to forge signatures on messages of his choice,

Claim. Let M be a PPT, K be a randomly-chosen key, f be a random oracle and MAC be a message authentication code. Assume that M receives as an input $C = f(K)$. The probability (over the choice of f and MAC) that M will succeed to output a message m and a signature σ such that $\sigma = \mathsf{MAC}(K, m)$ is negligible.

Proof. Assume by contradiction that M outputs such m and σ. Then we can construct M', based on M, except that M' first samples $\tilde{C} \leftarrow \{0,1\}^{|f(K)|}$, discards the original value of C and assigns $C \leftarrow \tilde{C}$. The resulting PPT M' operates essentially like M, so M' also outputs m and σ such that $\sigma = \mathsf{MAC}(K, m)$, with non-negligible probability. This contradicts the definition of MACs. □

We wish to prove that the iteration-invariant holds for all $i \in \mathbb{N}$. We prove that by induction in Claim 7.2, which uses claims 7.2 for proving the induction base, and Claim 3 for proving the induction step. The following proves that the iteration-invariant holds for $i = 1$, i.e., immediately after the initialization phase.

Claim. Let $\mathsf{EXEC}^k_{\Pi,M}$ be an execution of the protocol. The iteration-invariant holds for $i = 1$ (i.e., at the end of the initialization phase).

Proof. Condition 1 holds trivially from the protocol. Because the A_I and B_I interacts authentically, we infer that $\left\langle C_1^A, \widehat{C_0^B}, \widehat{C_1^B} \right\rangle = \left\langle \widehat{C_1^A}, C_0^B, C_1^B \right\rangle$, and equivalently $\langle C_1^A, C_0^B, C_1^B \rangle = \left\langle \widehat{C_1^A}, \widehat{C_0^B}, \widehat{C_1^B} \right\rangle$ so condition 2 holds too. For each $K \in \left\{K_1^A, K_0^B, K_1^B\right\}$, K is chosen uniformly and independently from previous executions of the protocol, and the only information that the adversary receives about it is $C = f(K)$, as desired by condition 3. □

Now, we prove that the iteration-invariant holds for $i > 1$. First, note the first two iterations are a little bit different from the others, because they rely on the authenticity of the initialization phase rather than the previous iterations. We wish to prove them in the same manner as the others. Recall that we defined some pseudo-variables in the initialization phase for simplifying this issue in the protocol definition, so we can use them in the proof too.

The following claim proves an important property about the data phase. Intuitively, the claim says that if the parties have matching-sub-conversations

of length $i-1$ and iteration-invariant holds for the i-th iteration, then the parties have matching sub-conversations of length i. Note that $\forall j \in \mathbb{N}$, $\alpha_j = \langle K^A_{j-1}, C^A_{j+1}, m^A_j, \sigma^A_j \rangle$, $\beta_j = \langle K^B_{j-1}, C^B_{j+1}, m^B_j, \sigma^B_j \rangle$.

Lemma 3. *Let* $\mathsf{EXEC}^k_{\Pi,M}$ *be an execution of the protocol. Let* $i \in \mathbb{N}$. *Assume that* $\mathcal{K}^A \overset{i-1}{\approx} \mathcal{K}^B$ *and that the iteration-invariant holds for* i. *If* A_D *outputted* $b^A_i = 1$ *and* B_D outputted $b^B_i = 1$, *then* $\mathcal{K}^A \overset{i}{\approx} \mathcal{K}^B$, except for negligible probability.

Proof. Because $\mathcal{K}^A \overset{i-1}{\approx} \mathcal{K}^B$, *we can denote the last matching tuple of each conversation as the following:* $\mathcal{K}^A_{i,i} = (\tau_{2i-2}, \beta_{i-1}, \alpha_i)$, $\mathcal{K}^B_{i-1,i-1} = (\tau_{2i-3}, \alpha_{i-1}, \beta_{i-1})$. *So* $\beta_{i-1} = \widehat{\beta_{i-1}}$, *namely:*

$$\langle K^B_{i-2}, C^B_i, m^B_{i-1}, \sigma^B_{i-1} \rangle = \left\langle \widehat{K^B_{i-2}}, \widehat{C^B_i}, \widehat{m^B_{i-1}}, \widehat{\sigma^B_{i-1}} \right\rangle \tag{7.1}$$

In order to prove that $\mathcal{K}^A \overset{i}{\approx} \mathcal{K}^B$ (except for negligible probability), we need to prove that the next tuple of each of these conversations, $\mathcal{K}^A_{i+1,i+1} = \left(\tau_{2i}, \widehat{\beta_i}, \alpha_{i+1}\right)$, $\mathcal{K}^B_{i,i} = (\tau_{2i-1}, \widehat{\alpha_i}, \beta_i)$, *are correct. More explicitly:*

1. $\tau_{2i-1} > \tau_{2i-2}$
2. $\tau_{2i} > \tau_{2i-1}$
3. $\widehat{\alpha_i} = \alpha_i$
4. $\widehat{\beta_i} = \beta_i$

The intuition of this proof relies on the observation that each key (except K^B_0*) fulfills two goals. Let* $P \in \{A, B\}$ *be the sender party, and* $Q \in \{A, B\}$ *be the receiver party. Consider the key* K^P_i ($i \in \mathbb{N}$). *The first goal of this key is guaranteeing that the receiver of the logical message* m^Q_{i+1} *received* any *logical message* before *the sender reveals the key* K^Q_{i+1} *(which may also be incorrect logical message, but the receiver will not consider any new bits to be part of this message afterwards). The second goal of the key is verifying the correctness of the logical message* m^P_i *to be generated by* P. *We define 2 claims in this proof. Claim 3 deals with the first goal, and Claim 3 deals with the second goal.*

The following claim will help us to prove that the timestamps are correct and the keys sent inside the messages are correct, except for negligible probability.

Claim. Let $P, Q \in \{A, B\}$ *be parties* ($x \neq y$) *and* $j \in \mathbb{N}$ *s.t* $j \leq i$. *Let* τ' *be the time when* Q_D *sends* K^Q_j *and* τ'' *be the time when* P_D *receives* $\widehat{K^Q_j}$. *If* $\widehat{C^Q_j} = f(\widehat{K^Q_j})$, then $\widehat{K^Q_j} = K^Q_j$ *and* $\tau'' > \tau'$, *except for negligible probability.*

Proof. $\widehat{C^Q_j} = C^Q_j$ *because* $\mathcal{K}^A \overset{i-1}{\approx} \mathcal{K}^B$ *(for* $j = 1$, *this is true because the iteration-invariant holds for* $i = 1$*), and* $j \leq i$. *In order to satisfy* $\widehat{C^Q_j} = f(\widehat{K^Q_j})$, *the adversary should send at time* τ'' *a valid preimage of* C^Q_j. *Since* f *is a random oracle, and since* K^Q_j *was chosen uniformly by* Q_D, *then the probability that*

she succeed to guess it correctly before τ' is negligible, so we can consider only the case that she sends the preimage K_j^Q she learned after time τ' . Namely $\widehat{K_j^Q} = K_j^Q$ and $\tau'' > \tau'$. □

We now use Claim 3 four times:

1. We assumed that $b_i^B = 1$, which means that during iteration $i+1$ of B_D, the verification at line 4 has passed, namely $\widehat{C_i^A} = f(\widehat{K_i^A})$. *So, by setting: $P = B$, $Q = A$, $j = i$, $\tau' = \tau_{2i}$, $\tau'' = \tau_{2i+1}$, Claim 3 implies that $\widehat{K_i^A} = K_i^A$ and $\tau_{2i+1} > \tau_{2i}$, except for negligible probability.*
2. We assumed that $b_i^A = 1$, which means that during iteration $i+1$ of A_D, the verification at line 10 has passed, namely $\widehat{C_i^B} = f(\widehat{K_i^B})$. *So, by setting: $P = A$, $Q = B$, $j = i$, $\tau' = \tau_{2i+1}$, $\tau'' = \tau_{2i+2}$, Claim 3 implies that $\widehat{K_i^B} = K_i^B$ and $\tau_{2i+2} > \tau_{2i+1}$, except for negligible probability.*
3. Using Claim 7.2 we can do the same as the two above again, but for $i-1$ instead of i and infer that $\widehat{K_{i-1}^A} = K_{i-1}^A$, $\widehat{K_{i-1}^B} = K_{i-1}^B$ *and $\tau_{2i} > \tau_{2i-1} > \tau_{2i-2}$, except for negligible probability.*

So far we proved items 1 and 2 (that the times are correct: $\tau_{2i+2} > \tau_{2i+1} > \tau_{2i} > \tau_{2i-1} > \tau_{2i-2}$, except for negligible probability) and just part of 3 and 4 (that keys are correct: $\widehat{K_{i-1}^A} = K_{i-1}^A$, $\widehat{K_{i-1}^B} = K_{i-1}^B$,$\widehat{K_i^A} = K_i^A$, $\widehat{K_i^B} = K_i^B$, except for negligible probability). The following claim will help us to prove that the other part of items 3 and 4 are also correct, except for negligible probability.

Claim. Let $P, Q \in \{A, B\}$ be parties ($x \neq y$). If $\widehat{\sigma_i^Q} = \mathsf{MAC}\left(\widehat{K_i^Q}, \left\langle \widehat{C_{i+1}^Q}, \widehat{m_i^Q} \right\rangle\right)$, then $\left\langle \widehat{C_{i+1}^Q}, \widehat{m_i^Q}, \widehat{\sigma_i^Q} \right\rangle = \left\langle C_{i+1}^Q, m_i^Q, \sigma_i^Q \right\rangle$ *except for negligible probability.*

Proof. Let τ' be the time Q_D sends K_i^Q and τ'' be the time P_D receives $\left\langle \widehat{C_{i+1}^Q}, \widehat{m_i^Q}, \widehat{\sigma_i^Q} \right\rangle$. We have already proved that $\tau'' < \tau'$ (using Claim 3). Before time τ'', the only information depends on K_i^Q that the adversary received from the parties is $C_i^Q = f(K_i^Q)$. *By using* Claim 7.2, we infer that the probability that she *will succeed to guess a triplet $\langle C, m, \sigma \rangle$* that satisfy $\sigma = \mathsf{MAC}\left(K_i^Q, \langle C, m \rangle\right)$ *is negligible, so she must choose $\left\langle \widehat{C_{i+1}^Q}, \widehat{m_i^Q}, \widehat{\sigma_i^Q} \right\rangle = \left\langle C_{i+1}^Q, m_i^Q, \sigma_i^Q \right\rangle$*, except for negligible probability. □

We now use Claim 3 twice, once for each message direction:

1. From the assumption that $b_i^B = 1$, using Claim 7.2 we infer that $b_{i-1}^B = 1$, which means that during iteration i of B_D, the verification at line 5 has passed, namely $\widehat{\sigma_i^A} = \mathsf{MAC}\left(\widehat{K_i^A}, \left\langle \widehat{C_{i+1}^A}, \widehat{m_i^A} \right\rangle\right)$. *So, by setting: $P = B$, $Q = A$, we infer that $\left\langle \widehat{C_{i+1}^A}, \widehat{m_i^A}, \widehat{\sigma_i^A} \right\rangle = \langle C_{i+1}^A, m_i^A, \sigma_i^A \rangle$, except for negligible probability.*

2. From the assumption that $b_i^A = 1$, using Claim 7.2 we infer that b_{i-1}^A, which means that during iteration i of A_D, the verification at line 11 has passed, namely $\widehat{\sigma_i^B} = \mathsf{MAC}\left(\widehat{K_i^B}, \left\langle \widehat{C_{i+1}^B}, \widehat{m_i^B} \right\rangle\right)$. *So, by setting:* $P = A$, $Q = B$, *we infer that* $\left\langle \widehat{C_{i+1}^B}, \widehat{m_i^B}, \widehat{\sigma_i^B} \right\rangle = \langle C_{i+1}^B, m_i^B, \sigma_i^B \rangle$, *except for negligible probability.*

Concluding the proof of Lemma 3. *We found that, except for negligible probability:*

$$\tau_{2i} > \tau_{2i-1} > \tau_{2i-2}$$

$$\widehat{\alpha_i} = \left\langle \widehat{K_{i-1}^A}, \widehat{C_{i+1}^A}, \widehat{m_i^A}, \widehat{\sigma_i^A} \right\rangle = \langle K_{i-1}^A, C_{i+1}^A, m_i^A, \sigma_i^A \rangle = \alpha_i$$

$$\widehat{\beta_i} = \left\langle \widehat{K_{i-1}^B}, \widehat{C_{i+1}^B}, \widehat{m_i^B}, \widehat{\sigma_i^B} \right\rangle = \langle K_{i-1}^B, C_{i+1}^B, m_i^B, \sigma_i^B \rangle = \beta_i$$

as desired. Thus, $\mathcal{K}^A \overset{i}{\approx} \mathcal{K}^B$, except for negligible probability.

□

Using the above lemma and claims, we can prove that the iteration-invariant holds for all $i \in \mathbb{N}$:

Claim. Let $\mathsf{EXEC}_{\Pi,M}^k$ be an execution of the protocol. For each $i \in \mathbb{N}$, if the execution completed the i-th iteration with $b_i^A = 1$ and $b_i^B = 1$ at end, then the iteration-invariant holds for i, and moreover $\mathcal{K}^A \overset{i}{\approx} \mathcal{K}^B$, except for negligible probability.

Proof. By induction.

Induction base. Claim 7.2 proves that the iteration-invariant holds for $i = 1$. Also, as discussed in Section 7.1, $\mathcal{K}_1^A = (\tau_0, \lambda, \alpha_1)$ and $\mathcal{K}_0^B = \lambda$, and thus $\mathcal{K}^A \overset{0}{\approx} \mathcal{K}^B$. Now, using Claim 3 with $i = 1$ we infer that $\mathcal{K}^A \overset{1}{\approx} \mathcal{K}^B$, except for negligible probability.

Induction step. Let $1 < i \in \mathbb{N}$. Assume that the following holds: If $b_{i-1}^A = 1$ and $b_{i-1}^B = 1$, then the iteration-invariant holds for $i-1$, and that $\mathcal{K}^A \overset{i-1}{\approx} \mathcal{K}^B$. We wish to prove that if $b_i^A = 1$ and $b_i^B = 1$, then the iteration-invariant holds for i, and that $\mathcal{K}^A \overset{i}{\approx} \mathcal{K}^B$.

Assume $b_i^A = 1$ and $b_i^B = 1$. From Claim 7.2 we can infer that b_{i-1}^A and b_{i-1}^B, so from the induction assumption, the iteration-invariant holds for $i-1$, and that $\mathcal{K}^A \overset{i-1}{\approx} \mathcal{K}^B$. Thus, except for negligible probability:

$$\tau_{2i-2} > \tau_{2i-3} > \tau_{2i-4}$$

$$\widehat{\alpha_{i-1}} = \left\langle \widehat{K_{i-2}^A}, \widehat{C_i^A}, \widehat{m_{i-1}^A}, \widehat{\sigma_{i-1}^A} \right\rangle = \langle K_{i-2}^A, C_i^A, m_{i-1}^A, \sigma_{i-1}^A \rangle = \alpha_{i-1}$$

$$\widehat{\beta_{i-1}} = \left\langle \widehat{K_{i-2}^B}, \widehat{C_i^B}, \widehat{m_{i-1}^B}, \widehat{\sigma_{i-1}^B} \right\rangle = \langle K_{i-2}^B, C_i^B, m_{i-1}^B, \sigma_{i-1}^B \rangle = \beta_{i-1}$$

So $\widehat{C_i^A} = C_i^A$ and $\widehat{C_i^B} = C_i^B$, except for negligible probability. Also, since the iteration-invariant holds for $i-1$ (except for negligible probability), $\widehat{C_{i-1}^B} = C_{i-1}^B$, except for negligible probability. So condition 2 of the iteration-invariant holds for i. Condition 1 holds trivially from the description of the protocol. Recall that K_{i-1}^B, K_i^A, K_i^B are sent in times $\tau_{2i-1} < \tau_{2i} < \tau_{2i+1}$, respectively, and that the beginning of the i iteration of the initiator is at time τ_{2i-2}, for which $\tau_{2i-2} < \tau_{2i-1}$. So for each $K \in \left\{K_i^A, K_{i-1}^B, K_i^B\right\}$, K is chosen uniformly and independently from the past, and the only information depending on K that the adversary receives from the parties is $C = f(K)$, so condition 3 holds as well. So the iteration-invariant holds for i. Now we can use Claim 3 with i and infer that $\mathcal{K}^A \overset{i}{\approx} \mathcal{K}^B$.

□

In Claim 7.2 we proved that the probability that the parties will fail to reveal the adversary's interference in an *individual* iteration of the execution is negligible. Now we want to prove that the probability that the probability that they will fail to reveal it in *any* iteration is negligible. We do so in Claim 7.2, by applying union bound on the individual iterations,

Claim. Let $\mathsf{EXEC}_{\Pi,M}^k$ be an execution of the protocol. The probability that there exists $i \in \mathbb{N}$ such that the execution completed the i-th iteration with $b_i^A = 1$ and $b_i^B = 1$ at end, but $\mathcal{K}^A \overset{i}{\approx} \mathcal{K}^B$ does not hold, is negligible.

Proof. The adversary is efficient, so she can do only polynomial-time iterations. Let $K(n)$ be a polynomial function and Let $k = K(n)$ be the number of iterations that the adversary perform in the execution. Let BAD_i be the event that the execution completed the i-th iteration with $b_i^A = 1$ and $b_i^B = 1$ at end, but $\mathcal{K}^A \overset{i}{\approx} \mathcal{K}^B$ does not hold. By using Claim 7.2, we infer that for all $k \geq i \in \mathbb{N}$, $\Pr\left[\mathsf{BAD}_i\right] = \mathsf{negl}(n)$. By applying union bound, we get

$$\Pr\left[\bigcup_{k \geq i \in \mathbb{N}} \mathsf{BAD}_i\right] \leq \sum_{k \geq i \in \mathbb{N}} \mathsf{BAD}_i = k \cdot \mathsf{BAD}_i = k \cdot \mathsf{negl}(n) = \mathsf{negl}(n)$$

The first move is the union bound. The last move is true because k is polynomial. □

The security of the protocol follows:

Theorem 3. *Let f be a random oracle and let $\mathsf{MAC}(.)$ be a MAC function. PEBIUS is a delayed chosen-message secure ∞-MRP (Definition 7 with delay $d = 1$.*

Proof. Let $n \in \mathbb{N}$, $k = K(n)$ where K is any polynomial function. Let GenMsg^A and GenMsg^B be any functions. Let M be any efficient (PPT) adversary. We need to prove that $\Pr\left[\mathsf{EXEC}_{\Pi,M}^k = \mathsf{win}\right] = \mathsf{negl}(n)$.

In order to return win, there should be $P, Q \in \{A, B\}$, $P \neq Q$ and $i \in \mathbb{N}$, such that $b_i^P = 1$ and $m_{i-1}^Q \neq \widehat{m_{i-1}^Q}$. Assume by contradiction that such P, Q, i exists with non-negligible probability. From Claim 7.2, we infer that $\mathcal{K}^A \overset{i}{\approx} \mathcal{K}^B$, except for negligible probability. In particular, $m_{i-1}^A = \widehat{m_{i-1}^A}$ and $m_{i-1}^B = \widehat{m_{i-1}^B}$, in contradiction. □

7.3 Tightening the Cryptographic Primitives

Our security prove of the protocol in the random oracle model, because there is no provable truely secure random function construction. Alternatively, we can avoid using random oracles by replacing the cryptographic primitives by two other primitives f, MAC, which satisfy the following criteria:

Definition 16. *Two functions f, MAC are sufficient for our protocol, if f is a collision-resistant hash function (CRH), MAC is a message authentication code (MAC), and no efficient adversary A can win the following game win non-negligible probability:*

1. $k \leftarrow \mathsf{GenKey}()$
2. $\mu, \sigma \leftarrow A(f(k))$
3. *Win iff* $\sigma = \mathsf{MAC}(k, \mu)$

To see why it is true, pay attention that in the proof of Section 7, we could swap the random oracle f by a CRH and leave the proof as is, except for Claim 7.2. However, the requirement in Definition 16 essentially matches this claim.

8 Conclusion

We are already surrounded by large numbers of small devices that sense, compute, communicate, and actuate. Today these devices use the Bluetooth, ZigBee, and proprietary wireless protocols. In the future, it is expected that many of these devices will use Internet protocols. It is crucial to be able to form secure network links between such devices. Yet, current standards for low-power devices (ZigBee and Bluetooth Smart) are vulnerable to simple MITM attacks, due to the perception that security would be too expensive in computation, power and user-interface components.

This paper disproves that widely-held belief. Current devices already include simple user-interface components (a lamp and a switch) to initiate and report the state of *insecure* association protocols. We show that these components can be repurposed in software to implement an abstract use-interface primitive that can enable *secure* association. This user-interface primitive, *oblivious comparison*, can enable both the public-key based "secure simple pairing" protocol, which provides both authentication and privacy, and a range of *message-recognition protocols* that provide only message authentication but require no public-key computations. We also argue that the latter should and can be secure in a strong sense, using the Universal Composability framework, which guarantees security

when the protocol is used by arbitrary high-level applications and in the presence of arbitrary other protocols. Finally, we presented our MRP, named PEBIUS, and proved that it holds the strongest security definition for MRP that we defined in this paper.

The Internet of Things needs to be secure. It can be secure. This paper shows how to do it.

References

1. Ross Anderson, Francesco Bergadano, Bruno Crispo, Jong-Hyeon Lee, Charalampos Manifavas, and Roger Needham. A new family of authentication protocols. *SIGOPS Oper. Syst. Rev.*, 32(4):9–20, 1998.
2. Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of the cipher block chaining message authentication code. *J. Comput. Syst. Sci.*, 61(3):362–399.
3. Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *CRYPTO 1993*, CRYPTO '93, pages 232–249. Springer, 1994.
4. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000.
5. Claude Castelluccia and Pars Mutaf. Shake them up!: a movement-based pairing protocol for cpu-constrained devices. In *International conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 51–64. ACM, 2005.
6. Ming Ki Chong, Rene Mayrhofer, and Hans Gellersen. A survey of user interaction for spontaneous device association. *ACM Computing Surveys (CSUR)*, 47(1):8, 2014.
7. Christian Gehrmann. Multiround unconditionally secure authentication. *Designs, Codes and Cryptography*, 15(1):67–86, 1998.
8. Ian Goldberg, Atefeh Mashatan, and Douglas R. Stinson. A new message recognition protocol with self-recoverability for ad hoc pervasive networks. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *ACNS*, volume 5536 of *LNCS*, pages 219–237.
9. Madeline Gonzalez Muniz and Peeter Laud. On the (im) possibility of perennial message recognition protocols without public-key cryptography. In *ACM Symposium on Applied Computing 2011*, pages 1510–1515. ACM, 2011.
10. Michael T. Goodrich, Michael Sirivianos, John Solis, Claudio Soriente, Gene Tsudik, and Ersin Uzun. Using audio in secure device pairing. *IJSN*, 4(1/2):57–68, 2009.
11. Bluetooth Special Interest Group. Specification of the Bluetooth system, 2007.
12. Bluetooth Special Interest Group. Specification of the Bluetooth system, 2010.
13. Jonathan Hammell, Andre Weimerskirch, Joao Girao, and Dirk Westhoff. Recognition in a low-power environment. In *Workshop on Wireless Ad Hoc Networking, held in conjunction with IEEE International Conference on Distributed Computing Systems*. WWAN2005, ICDCS2005.
14. Arun Kumar, Nitesh Saxena, Gene Tsudik, and Ersin Uzun. Caveat eptor: A comparative study of secure device pairing methods. In *Pervasive Computing and Communications (PerCom) 2009*, pages 1–10. IEEE, 2009.
15. Cynthia Kuo, Jesse Walker, and Adrian Perrig. Low-cost manufacturing, usability, and security: An analysis of bluetooth simple pairing and wi-fi protected setup. In Sven Dietrich and Rachna Dhamija, editors, *Financial Cryptography and Data Security*, volume 4886 of *LNCS*, pages 325–340. Springer, 2007.

16. L. Lamport. Constructing digital signatures from a one-way function. Technical Report CSL-98, SRI International, Oct 1979.
17. Andrew Y. Lindell. Comparison-based key exchange and the security of the numeric comparison mode in Bluetooth v2.1. In Marc Fischlin, editor, *CT-RSA*, volume 5473 of *LNCS*, pages 66–83. Springer.
18. Andrew Y. Lindell. Attacks on the pairing protocol of Bluetooth v2.1, 2008.
19. Stefan Lucks, Erik Zenner, Andre Weimerskirch, and Dirk Westhoff. Concrete security for entity recognition: The jane doe protocol. In Dipanwita Roy Chowdhury, Vincent Rijmen, and Abhijit Das, editors, *INDOCRYPT*, volume 5365 of *LNCS*, pages 158–171. Springer.
20. Yasir Arfat Malkani and L Das Dhomeja. Secure device association for ad hoc and ubiquitous computing environments. In *International Conference on Emerging Technologies (ICET) 2009*, pages 437–442. IEEE, 2009.
21. Atefeh Mashatan and Douglas R. Stinson. A new message recognition protocol for ad hoc pervasive networks. *IACR Cryptology ePrint Archive*, 2008:294.
22. Atefeh Mashatan and Serge Vaudenay. A message recognition protocol based on standard assumptions. In Jianying Zhou and Moti Yung, editors, *ACNS*, volume 6123 of *LNCS*, pages 384–401.
23. Rene Mayrhofer and Hans Gellersen. Shake well before use: Authentication based on accelerometer data. In Anthony LaMarca, Marc Langheinrich, and KhaiN. Truong, editors, *Pervasive Computing*, volume 4480 of *LNCS*, pages 144–161. Springer, 2007.
24. J.M. McCune, A Perrig, and M.K. Reiter. Seeing-is-believing: using camera phones for human-verifiable authentication. In *IEEE Symposium on Security and Privacy 2005*, pages 110–124, 2005.
25. Chris J. Mitchell. Remote user authentication using public information. In Kenneth G. Paterson, editor, *IMA International Conference on Cryptography and Coding*, volume 2898 of *LNCS*, pages 360–369. Springer, 2003.
26. Madeline Gonzalez Muniz and Rainer Steinwandt. Cryptanalysis of a message recognition protocol by mashatan and stinson. In *Information, Security and Cryptology–ICISC 2009*, pages 362–373. Springer, 2010.
27. Adrian Perrig, Ran Canetti, J Doug Tygar, and Dawn Song. The TESLA broadcast authentication protocol. 2005.
28. Ramnath Prasad and Nitesh Saxena. Efficient device pairing using "human-comparable" synchronized audiovisual patterns. In *Applied Cryptography and Network Security*, pages 328–345. Springer, 2008.
29. Volker Roth, Wolfgang Polak, Eleanor Rieffel, and Thea Turner. Simple and effective defense against evil twin access points. In *ACM Conference on Wireless Network Security*, WiSec '08, pages 220–235. ACM, 2008.
30. N. Saxena, J.-E. Ekberg, K. Kostiainen, and N. Asokan. Secure device pairing based on a visual channel. In *IEEE Symposium on Security and Privacy 2006*, pages 6 pp.–313, 2006.
31. Yaniv Shaked and Avishai Wool. Cracking the Bluetooth PIN. In *in USENIX/ACM Conf. Mobile Systems, Applications, and Services (MobiSys)*, pages 39–50, 2005.
32. Frank Stajano. The resurrecting duckling. In *Security Protocols*, pages 183–194. Springer, 2000.
33. R. Steinmetz. Human perception of jitter and media synchronization. *IEEE Journal on Selected Areas in Communications*, 14(1):61–72, 1996.
34. Jani Suomalainen, Jukka Valkonen, and N Asokan. Standards for security associations in personal networks: a comparative analysis. *International Journal of Security and Networks*, 4(1):87–100, 2009.

35. Andre Weimerskirch and Dirk Westhoff. Zero common-knowledge authentication for pervasive networks. In *Workshop Selected Areas in Cryptography (SAC '03)*, pages 73–87, 2003.