Unilaterally-Authenticated Key Exchange

Yevgeniy Dodis[1] and Dario Fiore[2]

[1] Department of Computer Science, New York University, USA
[2] IMDEA Software Institute, Spain

Abstract. Key Exchange (KE), which enables two parties (e.g., a client and a server) to securely establish a common private key while communicating over an insecure channel, is one of the most fundamental cryptographic primitives. In this work, we address the setting of *unilaterally-authenticated key exchange* (UAKE), where an unauthenticated (unkeyed) client establishes a key with an authenticated (keyed) server. This setting is highly motivated by many practical uses of KE on the Internet, but received relatively little attention so far.

Unlike the prior work, defining UAKE by downgrading a relatively complex definition of *mutually authenticated* key exchange (MAKE), our definition follows the opposite approach of upgrading existing definitions of public key encryption (PKE) and signatures towards UAKE. As a result, our new definition is short and easy to understand. Nevertheless, we show that it is *equivalent* to the UAKE definition of Bellare-Rogaway (when downgraded from MAKE), and thus captures a very strong and widely adopted security notion, while looking very similar to the simple "one-oracle" definition of traditional PKE/signature schemes. As a benefit of our intuitive framework, we show two *exactly-as-you-expect* (i.e., having no caveats so abundant in the KE literature!) UAKE protocols from (possibly interactive) signature and encryption. By plugging various one- or two-round signature and encryption schemes, we derive provably-secure variants of various well-known UAKE protocols (such as a unilateral variant of SKEME with and without perfect forward secrecy, and Shoup's A-DHKE-1), as well as new protocols, such as the first 2-round UAKE protocol which is both (passively) forward deniable and forward-secure.

To further clarify the intuitive connections between PKE/Signatures and UAKE, we define and construct stronger forms of (necessarily interactive) PKE/Signature schemes, called *confirmed encryption* and *confidential authentication*, which, respectively, allow the sender to obtain confirmation that the (keyed) receiver output the correct message, or to hide the content of the message being authenticated from anybody but the participating (unkeyed) receiver. Using confirmed PKE/confidential authentication, we obtain two concise UAKE protocols of the form: "send confirmed encryption/confidential authentication of a random key K."

1 Introduction

Key exchange (KE) is one of the most fundamental cryptographic primitives. Using a KE protocol, two parties can securely establish a common, private, cryptographic key while communicating over an insecure channel. Although the basic idea of KE dates back to the seminal work of Diffie and Hellman [9], a proper formalization of this notion was proposed only much later by Bellare and Rogaway [2]. In particular, Bellare and Rogaway considered the problem of *mutually authenticated* key exchange where two parties (e.g., a client and a server), each holding a valid long-term key pair, want to agree on a fresh common cryptographic key, while being assured about the identity of their protocol's partner. In [2], Bellare and Rogaway proposed a model for mutually-authenticated KE which allows to formally define security in this context, and in particular formalizes the adversary's capabilities in a proper way.

Building on this remarkable work, many other papers addressed KE in multiple directions, such as efficient and provably-secure realizations [20], or alternative security models [1,5,6]. Notably, the vast majority of papers in this area considered only the mutually authenticated setting where *both* the server and the client have long-term keys. However, it is striking to observe that many practical uses of KE protocols on the Internet work in a restricted setting where only the server has a long-term (certified) public key. A notable example of this setting is perhaps the simple access to web servers using the well known SSL/TLS protocol. This notion of KE has been often called *unilaterally-authenticated* (or, sometimes, anonymous, one-way or server-only) KE. To emphasize the distinction, in our work we will denote unilaterally-authenticated KE as *UAKE*, and mutually-authenticated KE as *MAKE*.

In spite of the practical relevance of unilaterally-authenticated key-exchange, we notice that most prior KE definitions targeted MAKE, and those works that focused on UAKE (e.g., [25,15,14,21]) used definitions that were obtained by slightly "downgrading" definitions of MAKE to the unilateral setting. The problem here is that existing definitions of MAKE are rigorous, but also pretty complex and hard to digest. Therefore, when analyzing the simple notion of UAKE by downgrading existing definitions of MAKE, one ends up with other complex definitions.

One goal of this work is thus to address this state of affairs by taking a different approach. Instead of considering UAKE as a downgraded version of MAKE, we propose a new definition of UAKE obtained by slightly "upgrading" the short and simple definitions of public key encryption and digital signatures. Precisely, we build on the recent work of Dodis and Fiore [10] that proposes a definitional framework for interactive message transmission protocols, and gives new notions of *interactive* public key encryption (PKE) and interactive public key message authentication (PKMA). These two notions naturally extend the classical notions of $\mathsf{IND-CCA}$ encryption (resp. strongly unforgeable signatures) to the interactive setting. By building on this framework, we obtain a UAKE definition which is (in our opinion) more intuitive and easier to digest.[3] Nevertheless, we show that our differently-looking UAKE definition is *equivalent* to the one of Bellare-Rogaway (BR) restricted to the single authenticated setting. This shows that we are not providing a new KE notion, but simply suggesting a different, simpler, way to explain the same notion when restricted to the unilateral setting. In fact, the BR UAKE definition "downgraded-from-MAKE" is actually noticeably simpler than the MAKE definition, but still (in our opinion) not as intuitive as our new definition. Hence, by establishing our equivalence, we offer a new path of teaching/understanding MAKE: (1) present our definition of UAKE, and use it to design and prove simple UAKE protocols (see below); (2) point out new subtleties of MAKE, making it hard (impossible?) to have a simple "one-oracle" definition of MAKE; (3) introduce the "downgraded" BR-framework (which has more finer-grain oracles available to the attacker) which is equivalent to our UAKE framework; (4) extend the "downgraded" BR framework to the full setting of MAKE. **We view this philosophy as a major educational contribution of this work.**

In the following, we describe our definitional framework and the remaining results (including simple and intuitive UAKE protocols) in more detail.

1.1 Our Results

Definitional Framework. The definitional framework proposed by Dodis and Fiore [10] consists of two parts. The first part is *independent* of the particular primitive, and simply introduces the bare minimum of notions/notation to deal with interaction. For example, they define (a) what it means to have *concurrent* oracle access to an *interactive party* under attack; and (b) what it means to 'act as a wire' between two honest parties (this trivial, but unavoidable, attack is called a 'ping-pong' attack). Once the notation is developed, the actual definitions become *as short and simple as in the non-interactive setting* (e.g., see Definitions 5 and 6). So, by building on this framework, we propose a simple notion of UAKE (cf. Definition 8) which we briefly discuss now. The attacker $\mathcal{A}$ has *concurrent* oracle access to the honest secret key owner (the "server"), and simultaneously tries to establish a (wlog single) session key with an honest unauthenticated client (the "challenger"). If the challenger rejects, $\mathcal{A}$ 'lost'.[4] If it accepts and the session is *not* a ping-pong of one of its conversations with the server, then $\mathcal{A}$ 'won', since it 'fooled' the challenger without trivially forwarding messages from the honest server. Otherwise, if $\mathcal{A}$ established a valid key with the challenger by a ping-pong attack, $\mathcal{A}$ 'wins' if it can distinguish a (well-defined) 'real' session key from a completely random key.[5]

[3] We stress, we are not suggesting that we can similarly simplify the more complicated definitions of MAKE. In fact, we believe that UAKE is *inherently easier* than MAKE, which is precisely why we managed to obtain our simpler definition only for UAKE.

[4] Notice, since anybody can establish a key with the server, to succeed $\mathcal{A}$ must establish the key with an honest client.

[5] Notice, for elegance sake our basic definition does not demand advanced properties, such as forward security or deniability, but (as we show) can be easily extended to do so. Indeed, our goal was not to get the most 'advanced' KE definition, but rather to get a strong and useful definition which is short, intuitive, and easy to digest.

Key Exchange Protocols. As we mentioned, our unilaterally-authenticated key-exchange (UAKE) definition can be seen as a natural extension of the interactive PKE/PKMA definitions in [10]. As a result, we show two simple and *very natural* constructions of UAKE protocols: from any possibly interactive PKE scheme and a PRF, depicted in Figure 2, and from any possibly interactive PKMA scheme and CPA-secure key encapsulation mechanism (KEM), depicted in Figure 3. By plugging various non-interactive or 2-round PKE/PKMA schemes (and KEMs, such as the classical Diffie-Hellman KE), we get a variety of simple and natural UAKE protocols. For example, we re-derive the A-DHKE-1 protocol from [25], the unilateral version of the SKEME protocol [19], and we get (to the best of our knowledge) the first 2-round UAKE, depicted in Figure **??**, which is both forward-deniable and forward-secure.

Hence, the main contribution of our work is not to design new UAKE protocols (which we still do due to the generality of our results!), but rather to have a simple and intuitive UAKE framework where *everything works as expected, without any caveats* (so abundant in the traditional KE literature). Namely, the fact that immediate corollaries of our work easily establish well known and widely used UAKE protocols is a big feature of our approach. Unlike prior work, however, our protocols: (1) work with *interactive* PKE/PKMA; (2) are directly analyzed in the unilateral setting using our simple definition, instead of being "downgraded" from more complex MAKE protocols.

Confirmed PKE and Confidential PKMA. To provide a further smoother transition from basic notions of PKE/PKMA towards KE, another contribution of our work is to define two strengthenings of PKE/PKMA which inherently require interaction. We call these notions *confirmed encryption* and *confidential authentication*, and study them in Section 6. In brief, confirmed encryption is an extension of the interactive encryption notion of Dodis and Fiore [10] in which the (unkeyed) sender gets a confirmation that the (keyed) receiver obtained the correct encrypted message, and thus accepts/rejects accordingly. Confidential authentication, instead, adds a privacy property to PKMA protocols [10] in such a way that no information about the message is leaked to adversaries controlling the communication channel (and, yet, the unkeyed honest receiver gets the message). Clearly, both notions require interaction, and we show both can be realized quite naturally with (optimal) two rounds of interaction. Moreover, these two notions provide two modular and "dual" ways to build secure UAKE protocols.Namely, we further abstract our UAKE constructions in Figures 2 and 3 by using the notions of confirmed PKE and confidential PKMA, by showing that "confirmed encryption of random K" and "confidential authentication of random K" both yield secure UAKE protocols.

Summary. Although we do not claim a special novelty in showing a connection between PKE/signatures and KE, we believe that the novelty of our contribution is to formally state such connection in a general and intuitive way. In particular, our work shows a path from traditional non-interactive PKE/PKMA schemes, to interactive PKE/PKMA, to (interactive) confirmed PKE/confidential PKMA, to UAKE, to MAKE (where the latter two steps use the equivalence of our simple "one-oracle" definition with the downgraded Bellare-Rogaway definition). Given that unilaterally-authenticated key-exchange, aside from independent interest, already introduces many of the subtleties of mutually-authenticated key-exchange (MAKE), we hope our work can therefore simplify the introduction of MAKE to students. Indeed, we believe all our results can be easily taught in an undergraduate cryptography course.

1.2 Related Work

Following the work of Bellare and Rogaway [2], several works proposed different security definitions for (mutually-authenticated) KE, e.g., [3,4,1,5,22]. Notably, some of these works focused on achieving secure composition properties [25,6]. Unilaterally-Authenticated Key-Exchange has been previously considered by Shoup [25] (who used the term "anonymous key-exchange"), Goldberg *et al.* [15] (in the context of Tor), Fiore *et al.* [14] (in the identity-based setting), and by Jager *et al.* [16] and Krawczyk *et al.* [21] (in the context of TLS). All these works arrived at unilaterally-authenticated key-exchange by following essentially the same approach: they started from (some standard definitions of) mutually-authenticated KE, and then they relaxed this notion by introducing one "dummy" user which can run the protocol without any secret (so, the unauthenticated party will run the protocol on behalf of such user), and by slightly changing the party-corruption condition.

Our authentication- (but not encryption-) based UAKE protocols also have conceptual similarities with the authenticator-based design of KE protocols by Bellare et al. [1]. Namely, although [1] concentrate on the mutually-authenticated setting, our UAKE of Figure 3 is similar to what can be obtained by applying a (unilateral) authenticator to an unauthenticated protocol, such as a one-time KEM. As explained in Section 4, however, the derived protocols are not exactly the same. This is because there are noticeable differences between authenticators and interactive PKMA schemes. For example, authenticators already require security against replay attack (and, thus, standard signature schemes *by themselves* are not good authenticators), and also use a very different real/ideal definition than our simple game-based definition of PKMA. In summary, while the concrete protocols obtained are similar (but not identical), the two works use very different definitions and construction paths to arrive at these similar protocols.

Finally, in a concurrent and independent work, Maurer, Tackmann and Coretti [24] considers the problem of providing new definitions of unilateral KE, and they do so by building on the constructive cryptography paradigm of Maurer and Renner [23]. Using this approach, they proposed a protocol which is based only on a CPA-secure KEM and an unforgeable digital signature, and is very similar to one of our UAKE protocols.

2 Background and Definitions

In our paper we use relatively standard notation recalled in Appendix A. Before giving the definitions of message transmission protocols and unilateral key exchange, we discuss two aspects of our definitions.

Session IDs . Throughout this paper, we consider various protocols (e.g., message transmission or key exchange) that may be run concurrently many times between the same two parties. In order to distinguish one execution of a protocol from another, one typically uses session identifiers, denoted sid, of which we can find two main uses in the literature. The first one is to consider purely "administrative" session identifiers, that are used by a user running multiple session to differentiate between them, i.e., to associate what session a message is going to or coming from. This means that the honest parties need some concrete mechanism to ensure the uniqueness of sid's, when honestly running multiple concurrent sessions. E.g., administrative sid can be a simple counter or any other nonce (perhaps together with any information necessary for communication, such as IP addresses or some mutually agreed upon timing information), or could be jointly selected by the parties, by each party providing some part of the sid. However, rather than force some particular choice which will complicate the notation, while simultaneously getting the strongest possible security definition, in our definitions we let the adversary *completely control* all the administrative sid's (as the adversary anyway controls all the protocol scheduling). In order not to clutter the notation with this trivial lower level detail, in our work *we will ignore such administrative* sid*'s from our notation*, but instead implicitly model them as stated above.

The second use of session identifiers in the literature is more technical as sid's are used in security definitions in order to define "benign" adversaries that simply act as a wire in the network. With respect to the use of sid's in security definitions we see three main approaches in the literature. The modern KE approach lets parties define sid's as part of the protocol. While this is more relaxed and allows for more protocols to be proven secure, it also somewhat clutters the notation as the choice of the sid is now part of the protocol specification. The second approach is to let sid be the transcript of a protocol execution, which simplifies the notation and implies the previous approach. In both the first and second approach, benign adversaries are those that cause two sessions have *equal* sid's. The third approach instead does not use explicit sid's, and considers benign adversaries those that cause two sessions have same transcript (seen as a "timed object"). All the approaches have pros and cons. For example, both the second and the third approach rule out some good protocols, but save on syntax and notation. Moreover, the third approach is the strongest one for security: it leaves to protocol implementers the freedom of picking the most convenient "administrative" sid selection mechanism, without worrying about security, since in this model adversaries can arbitrarily control the administrative sid's. For these reasons, in this work we follow the third approach, which also gives us the possibility of making our definitions more in line with those of PKE/signatures, where there are no explicit session identifiers.

Party Identities. Unlike the traditional setting of encryption and authentication, in the KE literature parties usually have external (party) identities in addition to their public/secret keys. This allows the same party to (claim to) have multiple keys, or, conversely, the same key for multiple identities. While generality is quite useful in the mutually authenticated setting, and could be easily added to all our definitions and results in the unilateral setting, we decided to avoid this extra layer of notation. Instead, we implicitly set the identity of the party to be its public key (in case of the server), or null (in case of the client). Aside from simpler notation, this allowed us to make our definitions look very similar to traditional PKE/signatures, which was one of our goals. We remark that this is a trivial and inessential choice which largely follows a historic tradition for PKE/PKMA. Indeed, having party identities is equally meaningful for traditional PKE/PKMA schemes, but is omitted from the syntax, because it can always be trivially achieved by appending the identities of the sender and/or recipient to the message. We stress, we do not assume any key registration authority who checks knowledge of secret keys. In fact, in our definition the attacker pretends to be the owner of the victim's secret key (while having oracle access to the victim), much like in PKE/PKMA the attacker tries to "impersonate" the honest party (signer/decryptor) with only oracle access to this party.

2.1 Message Transmission Protocols

In this section, we recall the definitional framework of *message transmission protocols* as defined in [10], along with suitable security definitions for confidentiality (called iCCA security) and authenticity (called iCMA security).

A message transmission protocol involves two parties, a sender S and a receiver R, such that the goal of S is to send a message m to R while preserving certain security properties on m. Formally, a message transmission protocol Π consists of algorithms $(\mathsf{Setup}, \mathsf{S}, \mathsf{R})$ defined as follows:

$\mathsf{Setup}(1^\lambda)$**:** on input the security parameter λ, the setup algorithm generates a pair of keys $(\mathsf{sendk}, \mathsf{recvk})$. In particular, these keys contain an implicit description of the message space $\mathcal{M}$.

$\mathsf{S}(\mathsf{sendk}, m)$**:** is a possibly interactive Turing machine that is run with the sender key sendk and a message $m \in \mathcal{M}$ as private inputs.

$\mathsf{R}(\mathsf{recvk})$**:** is a possibly interactive Turing machine that takes as private input the receiver key recvk, and whose output is a message $m \in \mathcal{M}$ or an error symbol $\perp$.

We say that Π is an n-round protocol if the number of messages exchanged between S and R during a run of the protocol is n. If Π is 1-round, then we say that Π is *non-interactive*. Since the sender has no output, it is assumed without loss of generality that the S *always speaks last*. This means that in an n-round protocol, R (resp. S) speaks first if n is even (resp. odd). For compact notation, $\langle \mathsf{S}(\mathsf{sendk}, m), \mathsf{R}(\mathsf{recvk}) \rangle = m'$ denotes the process of running S and R on inputs (sendk, m) and recvk respectively, and assigning R's output to m'. In our notation, we will use $m \in \mathcal{M}$ for messages (aka plaintexts), and capital M for protocol messages.

Definition 1 (Correctness). *A message transmission protocol* $\Pi = (\mathsf{Setup}, \mathsf{S}, \mathsf{R})$ *is* correct *if for all honestly generated keys* $(\mathsf{sendk}, \mathsf{recvk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$*, and all messages* $m \in \mathcal{M}$*, we have that* $\langle \mathsf{S}(\mathsf{sendk}, m), \mathsf{R}(\mathsf{recvk}) \rangle = m$ *holds with all but negligible probability.*

Defining Security: Man-in-the-Middle Adversaries. Here we recall the formalism needed to define the security of message transmission protocols. The basic idea is that an adversary with full control of the communication channel has to violate a given security property (say confidentiality or authenticity) in a run of the protocol that is called the *challenge session*. Formally, this session is a protocol execution $\langle \mathsf{S}(\mathsf{sendk}, m), \mathcal{A}^{\mathsf{R}(\mathsf{recvk})}\rangle$ or $\langle \mathcal{A}^{\mathsf{S}(\mathsf{sendk},\cdot)}, \mathsf{R}(\mathsf{recvk})\rangle$ where the adversary runs with an honest party (S or R). $\mathcal{A}^{\mathsf{P}}$ denotes that the adversary has oracle access to *multiple* honest copies of party P (where $\mathsf{P} = \mathsf{R}$ or $\mathsf{P} = \mathsf{S}$), i.e., $\mathcal{A}$ can start as many copies of P as it wishes, and it can run the message transmission protocol with each of these copies. In order to differentiate between several copies of P, formally $\mathcal{A}$ calls the oracle providing a session identifier sid. However, as mentioned earlier, to keep notation simple we do not write sid explicitly. The model assumes that whenever $\mathcal{A}$ sends a message to the oracle P, then $\mathcal{A}$ always obtains P's output. In particular, in the case of the receiver oracle, when $\mathcal{A}$ sends the last protocol message to R, $\mathcal{A}$ obtains the (private) output of the receiver, i.e., a message m or $\perp$.

Due to its power, the adversary might entirely replay the challenge session by using its oracle. Since this can constitute a trivial attack to the protocol, in what follows we recall the formalism of [10] to capture replay attacks. The approach is similar to the one introduced by Bellare and Rogaway [2] in the context of key exchange, based on the idea of "matching conversations".

Let t be a global counter which is progressively incremented every time a party (including the adversary) sends a message. Every message sent by a party (S, R or $\mathcal{A}$) is timestamped with the current time t. Using this notion of time,[6] the transcript of a protocol session is defined as follows:

Definition 2 (Protocol Transcript). *The* transcript of a protocol session *between two parties is the timestamped sequence of messages (including both sent and received messages) viewed by a party during a run of the message transmission protocol Π. If Π is n-round, then a transcript T is of the form $T = \langle (M_1, t_1), \ldots, (M_n, t_n)\rangle$, where $M_1, \ldots, M_n$ are the exchanged messages, and $t_1, \ldots, t_n$ are the respective timestamps.*

In a protocol run $\langle \mathsf{S}(\mathsf{sendk}, m), \mathcal{A}^{\mathsf{R}(\mathsf{recvk})}\rangle$ (resp. $\langle \mathcal{A}^{\mathsf{S}(\mathsf{sendk},\cdot)}, \mathsf{R}(\mathsf{recvk})\rangle$) we denote by T^* the transcript of the challenge session between S and $\mathcal{A}$ (resp. $\mathcal{A}$ and R), whereas $T_1, \ldots, T_Q$ are the Q transcripts of the sessions established by $\mathcal{A}$ with R (resp. S) via the oracle.

Definition 3 (Matching Transcripts). *Let $T = \langle (M_1, t_1), \ldots, (M_n, t_n)\rangle$ and $T^* = \langle (M_1^*, t_1^*), \ldots, (M_n^*, t_n^*)\rangle$ be two protocol transcripts. We say that T* matches *T^* ($T \subseteq T^*$, for short) if $\forall i = 1, \ldots, n$, $M_i = M_i^*$ and the two timestamp sequences are "alternating", i.e., $t_1 < t_1^* < t_2^* < t_2 < t_3 < \cdots < t_{n-1} < t_n < t_n^*$ if R speaks first, or $t_1^* < t_1 < t_2 < t_2^* < t_3^* < \cdots < t_{n-1} < t_n < t_n^*$ if S speaks first. Note that the notion of match is not commutative.*

Using the above definitions, we recall the notion of ping-pong adversary:

Definition 4 (Ping-pong Adversary). *Consider a run of the protocol Π involving $\mathcal{A}$ and an honest party (it can be either $\langle \mathsf{S}(\mathsf{sendk}, m), \mathcal{A}^{\mathsf{R}(\mathsf{recvk})}\rangle$ or $\langle \mathcal{A}^{\mathsf{S}(\mathsf{sendk},\cdot)}, \mathsf{R}(\mathsf{recvk})\rangle$), and let T^* be the transcript of the challenge session, and $T_1, \ldots, T_Q$ be the transcripts of all the oracle sessions established by $\mathcal{A}$. Then we say that $\mathcal{A}$ is a* ping-pong adversary *if there is a transcript $T \in \{T_1, \ldots, T_Q\}$ such that T matches T^*, i.e., $T \subseteq T^*$.*

Now that we have introduced all the necessary definitions, we recall the two notions of interactive chosen-ciphertext PKE (iCCA) and interactive chosen-message secure PKMA (iCMA) that capture, respectively, confidentiality and authenticity of the messages sent by S to R. Let $\Pi = (\mathsf{Setup}, \mathsf{S}, \mathsf{R})$ be a message transmission protocol, and $\mathcal{A}$ be an adversary. The two notions are defined as follows by considering the experiments in Figure 1.

[6] We stress that timestamps are only used in the security definition; in particular they are not used by real-world parties.

Experiment $\mathbf{Exp}^{\mathsf{iCCA}}_{\Pi,\mathcal{A}}(\lambda)$	Experiment $\mathbf{Exp}^{\mathsf{iCMA}}_{\Pi,\mathcal{A}}(\lambda)$
$b \xleftarrow{\$} \{0,1\}$	$(\mathsf{sendk}, \mathsf{recvk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
$(\mathsf{sendk}, \mathsf{recvk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$	$m^* \leftarrow \langle \mathcal{A}^{\mathsf{S}(\mathsf{sendk},\cdot)}(\mathsf{recvk}), \mathsf{R}(\mathsf{recvk})\rangle$
$(m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{R}(\mathsf{recvk})}(\mathsf{sendk})$	If $m^* \neq \perp$ and $\mathcal{A}$ is not "ping-pong", then output 1
$b' \leftarrow \langle \mathsf{S}(\mathsf{sendk}, m_b), \mathcal{A}^{\mathsf{R}(\mathsf{recvk})}(\mathsf{sendk})\rangle$	Else output 0.
If $\mathcal{A}$ is "ping-pong", then output $\tilde{b} \xleftarrow{\$} \{0,1\}$	
Else if $b' = b$ and $\mathcal{A}$ is not "ping-pong", then output 1	
Else output 0.	

Fig. 1. Security experiments of $\mathsf{iCCA}$and iCMAsecurity.

Definition 5 (iCCA security). *For any $\lambda \in \mathbb{N}$, we define the advantage of an adversary $\mathcal{A}$ in breaking* iCCA *security of a message transmission protocol Π as $\mathbf{Adv}^{\mathsf{iCCA}}_{\Pi,\mathcal{A}}(\lambda) = \Pr[\mathbf{Exp}^{\mathsf{iCCA}}_{\Pi,\mathcal{A}}(\lambda) = 1] - \frac{1}{2}$, and we say that Π is* iCCA*-secure if for any PPT $\mathcal{A}$, $\mathbf{Adv}^{\mathsf{iCCA}}_{\Pi,\mathcal{A}}(\lambda)$ is negligible.*

Note that for 1-round protocols, the above notion is the same as the classical $\mathsf{IND}{-}\mathsf{CCA}$ security.

Definition 6 (iCMA security). *For any $\lambda \in \mathbb{N}$, the advantage of $\mathcal{A}$ in breaking the* iCMA *security of a message transmission protocol Π is $\mathbf{Adv}^{\mathsf{iCMA}}_{\Pi,\mathcal{A}}(\lambda) = \Pr[\mathbf{Exp}^{\mathsf{iCMA}}_{\Pi,\mathcal{A}}(\lambda) = 1]$, and we say that Π is* iCMA*-secure if for any PPT $\mathcal{A}$, $\mathbf{Adv}^{\mathsf{iCMA}}_{\Pi,\mathcal{A}}(\lambda)$ is negligible.*

Note that for 1-round protocols, the above notion is the same as the notion of strong unforgeability for digital signatures.

3 Unilaterally-Authenticated Key-Exchange

In this section we build on the notions of $\mathsf{iCCA}/\mathsf{iCMA}$ secure message transmission protocols recalled in the previous section in order to obtain a smoother and clean transition from encryption/authentication towards key exchange. In particular, in this work we focus on *unilaterally-authenticated key-exchange* (UAKE, for short). UAKE is a weaker form of mutually-authenticated key-exchange in which only one of the two protocol parties is authenticated.

Following the definitional framework of message transmission protocols [10], we define UAKE as a protocol between two parties—in this case, an un-keyed user U and a keyed (aka authenticated) user T—so that, at the end of a successful protocol run, both parties (privately) output a common session key.

Formally, a UAKE protocol Π consists of algorithms $(\mathsf{KESetup}, \mathsf{U}, \mathsf{T})$ working as follows:

$\mathsf{KESetup}(1^\lambda)$**:** on input the security parameter λ, the setup algorithm generates a pair of keys $(\mathsf{uk}, \mathsf{tk})$. Implicitly, it also defines a session key space $\mathcal{K}$.

$\mathsf{U}(\mathsf{uk})$**:** is a possibly interactive algorithm that takes as input the public key uk of the authenticated user, and outputs a session key or a symbol $\perp$.

$\mathsf{T}(\mathsf{tk})$**:** is a possibly interactive algorithm that takes as input the private key tk, and outputs a session key K or an error symbol $\perp$.

In our security definitions we explicitly include the property that U terminates correctly (i.e., no $\perp$ output) only if U gets confirmation that T can terminate correctly. For this reason, we assume without loss of generality that T *always speaks last*. For compact notation, we denote with $\langle \mathsf{U}(\mathsf{uk}), \mathsf{T}(\mathsf{tk})\rangle = (\mathsf{K}_\mathsf{U}, \mathsf{K}_\mathsf{T})$ a run of the protocol in which U and T output session keys K_U and K_T respectively.

Definition 7 (Correctness). *An unilaterally-authenticated key-exchange protocol $\Pi = (\mathsf{KESetup}, \mathsf{U}, \mathsf{T})$ is* correct *if for all honestly generated key pairs $(\mathsf{uk}, \mathsf{tk}) \xleftarrow{\$} \mathsf{KESetup}(1^\lambda)$, and all session keys $\langle \mathsf{U}(\mathsf{uk}), \mathsf{T}(\mathsf{tk})\rangle = (\mathsf{K}_\mathsf{U}, \mathsf{K}_\mathsf{T})$, we have that, when $\mathsf{K}_\mathsf{U}, \mathsf{K}_\mathsf{T} \neq \perp$, $\mathsf{K}_\mathsf{U} = \mathsf{K}_\mathsf{T}$ holds with all but negligible probability.*

Security. For UAKE protocols we aim at formalizing two main security properties: *authenticity* and *confidentiality*. Intuitively, authenticity says that the only way for an adversary to make the un-keyed party terminate correctly (no $\perp$ output) is to be ping-pong. Confidentiality aims to capture that, once the un-keyed party U accepted, then the adversary cannot learn any information about the session key (unless it is ping-pong up to learning the key). We formalize these two properties in a single experiment in which $\mathcal{A}$ runs a challenge session with the un-keyed party U while having access to the keyed party T. As for the case for message transmission protocols, the adversary formally refers to the keyed party T oracle by specifying a session id sid. For simplicity of notation, however we do not write explicitly these session identifiers.

Since in UAKE T speaks last, we allow the adversary to make one additional query to T after T generated the last message: in this case T reveals its private output K_T. If $\mathcal{A}$ makes such an additional query in a ping-pong session then we say that $\mathcal{A}$ is "*full-ping-pong*".

Although the resulting experiment looks a bit more complex compared to the ones of iCCA and iCMA security, we stress that it can be seen as a natural combination of these two security notions. At a high level, the experiment consists in first running $(K_0, \cdot) \leftarrow \langle \mathsf{U}(\mathsf{uk}), \mathcal{A}^{\mathsf{T}(\mathsf{tk})}(\mathsf{uk}) \rangle$ and then analyzing U's output K_0 ($\cdot$ means that we do not care about $\mathcal{A}$'s output at this stage). If $K_0 \neq \perp$ and $\mathcal{A}$ is *not* ping-pong, then $\mathcal{A}$ wins (it broke authenticity). Otherwise, we give to $\mathcal{A}$ a real-or-random key K_b and $\mathcal{A}$ wins if it can tell these two cases apart *without*, of course, pushing the ping-pong attack up to getting K_0 revealed from the oracle T. Notice that when $K_0 = \perp$ (i.e., the honest sender did not accept in the challenge session), we also set $K_1 = \perp$. This is meant to capture that if U does not accept, then there is no common session key established by the two parties (essentially, no secure channel will be established). In this case the adversary will have no better chances of winning the game than guessing b.

Experiment $\mathbf{Exp}^{\mathsf{UAKE-Sec}}_{\Pi,\mathcal{A}}(\lambda)$

$(\mathsf{uk}, \mathsf{tk}) \xleftarrow{\$} \mathsf{KESetup}(1^\lambda)$;
$b \xleftarrow{\$} \{0,1\}$
$(K_0, \cdot) \leftarrow \langle \mathsf{U}(\mathsf{uk}), \mathcal{A}^{\mathsf{T}(\mathsf{tk})}(\mathsf{uk}) \rangle$
If $K_0 = \perp$, then $K_1 = \perp$
Else if $K_0 \neq \perp$ and $\mathcal{A}$ is not "ping-pong", then output 1
Else $K_1 \xleftarrow{\$} \mathcal{K}$
$b' \leftarrow \mathcal{A}^{\mathsf{T}(\mathsf{tk})}(K_b)$
If $\mathcal{A}$ is "full-ping-pong", then output $\tilde{b} \xleftarrow{\$} \{0,1\}$
Else if $b' = b$ and $\mathcal{A}$ is not "full-ping-pong", then output 1
Else output 0.

Definition 8 (Security of UAKE). *We define the advantage of an adversary $\mathcal{A}$ in breaking the security of Π as $\mathbf{Adv}^{\mathsf{UAKE-Sec}}_{\Pi,\mathcal{A}}(\lambda) = \left|\Pr[\mathbf{Exp}^{\mathsf{UAKE-Sec}}_{\Pi,\mathcal{A}}(\lambda) = 1] - \frac{1}{2}\right|$, and we say that a UAKE protocol Π is secure if for any PPT $\mathcal{A}$, $\mathbf{Adv}^{\mathsf{UAKE-Sec}}_{\Pi,\mathcal{A}}(\lambda)$ is negligible.*

Multi-User Extension of Our Notion. While we defined unilaterally-authenticated key-exchange in the single-user setting, we stress that the definition easily extends to the multi-user setting. The reason is that in our notion there is only one keyed user, T. So, when considering the multi-user setting with keyed users $\mathsf{T}_1, \ldots, \mathsf{T}_n$, we can assume that an adversary attacking a given T_j could simulate the keys of all remaining users $\mathsf{T}_i \neq \mathsf{T}_j$. In contrast, such an extension is not equally straightforward in MAKE, where, for example, the adversary could choose arbitrary keys for one of the two parties in the challenge session. We also refer the interested reader to [21] for a discussion on the multi-user extension of UAKE.

Single-Challenge vs. Multiple Challenges. Similarly to CCA-secure encryption and other privacy primitives, our attacker has only a single challenge session. Using a standard hybrid argument, this is asymptotically equivalent to the multi-challenge extension of our notion (with all challenge sessions sharing the same challenge bit b). We stress, however, that *single-challenge does not mean single oracle access to* T. Indeed, the attacker $\mathcal{A}^{\mathsf{T}}$ can start *arbitrarily many interleaved sessions with the keyed user* T, both before and after receiving the (single) challenge K_b. In particular, any UAKE protocol where one can recover the secret key tk given (multiple) oracle access to T will never be secure according to our definition, as then the attacker will trivially win the (single) challenge session by simulating honest T.

Relation with Existing Definitions. As we mentioned earlier in this section, the notion of UAKE has been considered in prior work with different definitions. Notably, two recent works [16,18] and [21] use a definition (Server only Authenticated and Confidential Channel Establishment – SACCE) which formally captures whether a party accepts or not in a protocol session, and requires that the adversary $\mathcal{A}$ should not let the party accept if $\mathcal{A}$ does not correctly relay messages. If we compare our security definition of UAKE given above and the SACCE notion, we then observe the following main facts. (i) Our notion of ping-pong is stronger than the notion of matching conversations used in SACCE in that ping-pong takes into account the timing of the messages included in the transcripts. (ii) While UAKE and SACCE are very similar w.r.t. capturing the authenticity property, they instead *differ w.r.t. confidentiality.* In particular, our notion aims to capture indistinguishability of the keys, whereas SACCE aims to capture the security of the channel built by using the established session key. As observed in [16], the latter security notion is weaker than mere session key indistinguishability, and might thus be realized from weaker assumptions.

Finally, we formally consider the relation between our security notion of UAKE and the security notion obtained by downgrading the Bellare-Rogaway [2] definition for mutually-authenticated key exchange to the case of a single authenticated party. Although the two definitions use a slightly different formalism, below we show that the notions are essentially the same. For completeness, we recall the Bellare-Rogaway security definition in Appendix B.

The motivation of proving the equivalence to the BR model is to show that our notion does not weaken existing, well studied notions, and can in fact be used in place of them. Indeed, we believe our notion is shorter and more intuitive to work with, as we illustrate in this work. It is worth noting that this is not surprising. Overall, the one-way authenticated setting is simpler than the mutually-authenticated one as there are fewer attacks to be modeled. For example, in UAKE the security definition can involve only one long-term key, and some advanced security properties such as *key-compromise impersonation* no longer apply to the unilateral setting. In other words, this equivalence gives the opportunity of modeling UAKE using our definition, and perhaps using the equivalence to BR as a transition towards the more complex MAKE definition.

Theorem 1. *Π is a secure UAKE protocol if and only if Π is secure in the (unilateral version of) Bellare-Rogaway model.*

Proof. To begin with, observe that the notion of matching conversation in the BR model is basically the same as our notion of matching transcripts (when considered for UAKE protocols). In the BR experiment recalled in Appendix B, there are two main events: $NoMatch$ is the event that oracle $\Pi_{\mathsf{U},\mathsf{T}}$ accepted but there is no other oracle $\Pi^s_{\mathsf{T},\mathsf{U}}$ that has a matching conversation with $\Pi_{\mathsf{U},\mathsf{T}}$ (i.e., the adversary broke the authenticity property); $GoodGuess$ is the event that the adversary correctly guessed which session key it received from the test query.

UAKE $\Rightarrow$ BR. First, we show that if Π is a correct and secure UAKE, then it is also BR-secure. Assume by contradiction that Π is not BR-secure. Then either one of the following cases occurs: (1) the adversary is benign and there are two oracles which either accept two different keys, or the accepted key is not honestly distributed; (2) $\Pr[NoMatch]$ is non-negligible, or (3) $\Pr[NoMatch]$ is negligible but $|\Pr[GoodGuess] - 1/2|$ is non-negligible.

If (1) occurs, then it means that Π is not a correct $UAKE$.

If there is an adversary $\mathcal{B}$ running in the BR security experiment such that (2) occurs, then we can build another adversary $\mathcal{A}$ playing in $\mathbf{Exp}^{\mathsf{UAKE-Sec}}_{\Pi,\mathcal{A}}(\lambda)$ such that the advantage $\mathbf{Adv}^{\mathsf{UAKE-Sec}}_{\Pi,\mathcal{A}}(\lambda)$ is non-negligible. $\mathcal{A}$ simply runs $\mathcal{B}$ by simulating the $\Pi_{\mathsf{U},\mathsf{T}}$ oracle with the messages received by U in the challenge session, and simulating every oracle $\Pi^s_{\mathsf{T},\mathsf{U}}$ by forwarding messages to a different copy of its T oracle. Clearly, $\mathcal{A}$ can provide a perfect simulation to $\mathcal{B}$. Finally, if the event $NoMatch$ occurs in the simulation, then $\mathcal{A}$ sends all messages and stops. Otherwise, if $NoMatch$ does not occur, then $\mathcal{A}$ will continue running in the second part of $\mathbf{Exp}^{\mathsf{UAKE-Sec}}_{\Pi,\mathcal{A}}(\lambda)$ and output a random bit b'. To conclude, we have:

$$\begin{aligned}\Pr[\mathbf{Exp}^{\mathsf{UAKE-Sec}}_{\Pi,\mathcal{A}}(\lambda) = 1] - \frac{1}{2} &= \Pr[\mathbf{Exp}^{\mathsf{UAKE-Sec}}_{\Pi,\mathcal{A}}(\lambda) = 1 \wedge NoMatch] + \\ &\quad \Pr[\mathbf{Exp}^{\mathsf{UAKE-Sec}}_{\Pi,\mathcal{A}}(\lambda) = 1 \wedge \overline{NoMatch}] - \frac{1}{2} \\ &= \Pr[NoMatch] + \frac{1}{2}\Pr[\overline{NoMatch}] - 1/2 \\ &= \frac{\Pr[NoMatch]}{2}\end{aligned}$$

If there is an adversary $\mathcal{B}$ running in the BR security experiment such that (3) occurs, then we can easily build another adversary $\mathcal{A}$ such that $\Pr[\mathbf{Exp}^{\mathsf{UAKE-Sec}}_{\Pi,\mathcal{A}}(\lambda) = 1] - 1/2$ is non-negligible, and thus Π is not a secure UAKE according to our definition. To see this, the observations are that: any adversary who is *benign* in the BR model, is ping-pong in our security experiment; if the adversary makes the test query on a fresh oracle in the BR model, then $\mathcal{A}$ is not full-ping-pong in our model.

BR $\Rightarrow$ UAKE. The converse proof is similar. If we assume by contradiction that Π is not a secure UAKE, then we can show that it is not BR-secure either.

Considering a run of the experiment $\mathbf{Exp}^{\mathsf{UAKE-Sec}}_{\Pi,\mathcal{A}}(\lambda)$, and let $\mathcal{E}$ be the event that $K_0 \neq \perp$ and $\mathcal{A}$ is not ping-pong. We will do our proof partitioning on successful adversaries $\mathcal{A}_1$ for which $\mathcal{E}$ occurs with non-negligible probability and successful adversaries $\mathcal{A}_2$ for which $\mathcal{E}$ occurs with at most negligible probability.

First, if there is an adversary $\mathcal{A}_1$ running in $\mathbf{Exp}^{\mathsf{UAKE-Sec}}_{\Pi,\mathcal{A}_1}(\lambda)$ such that $\Pr[\mathcal{E}]$ is non-negligible, we can immediately build an adversary $\mathcal{B}$ running in the BR security experiment such that $\Pr[NoMatch]$ is also non-negligible. The construction of $\mathcal{B}$ is straightforward: $\mathcal{B}$ simply uses its oracle $\Pi_{\mathsf{U},\mathsf{T}}$ to simulate the challenger to $\mathcal{A}_1$ and its oracles $\Pi^s_{\mathsf{T},\mathsf{U}}$ to simulate the oracles T. Then we observe that whenever the event $\mathcal{E}$ occurs in the simulation, the event $NoMatch$ occurs in the game played by $\mathcal{B}$.

Second, if there is an adversary $\mathcal{A}_2$ such that $\Pr[\mathbf{Exp}^{\mathsf{UAKE-Sec}}_{\Pi,\mathcal{A}_2}(\lambda) = 1] - 1/2$ is non-negligible (but event $\mathcal{E}$ does not occur), we can build an adversary $\mathcal{B}$ running in the BR security experiment such that $\Pr[GoodGuess] - 1/2$ is also non-negligible. Again the construction of $\mathcal{B}$ is straightforward: it simply uses its oracles to simulate the $UAKE$ experiment to $\mathcal{A}_2$. Then the main observation is that whenever $\mathcal{A}_2$ would cause experiment $\mathbf{Exp}^{\mathsf{UAKE-Sec}}_{\Pi,\mathcal{A}_2}(\lambda)$ output 1 (without having $\mathcal{E}$ occur), then $GoodGuess$ occurs in the game played by $\mathcal{B}$. $\square$

Uniqueness of Matching Transcript. It is interesting to note that our security definition implies that for any secure protocol there can be *at most one* matching transcript. This for instance means that it is hard for an adversary to force two distinct protocol sessions (in which one of the two parties is honest) to have the same session key.[7] Bellare and Rogaway prove in [2] that such property is achieved by any protocol secure according to their (mutually-authenticated) definition. By the equivalence of

[7] We stress that here we mean to force two distinct *oracle* sessions to have the same session key.

our UAKE notion to BR security one might be tempted to conclude that this uniqueness property holds for UAKE-secure protocols as well. This is only partially true as the proof in [2] is done for the mutually-authenticated case, and in particular one case of the proof uses the fact mutually-authenticated (BR-secure) protocols require at least 3 rounds. Below we give a separate proof of this statement for UAKE protocols .

Proposition 1. *Let* $\mathsf{MultipleMatch}$ *be the event that in a run of* $\mathbf{Exp}^{\mathsf{UAKE-Sec}}_{\Pi,\mathcal{A}}(\lambda)$ $\mathcal{A}$ *is ping-pong and there are at least two sessions* i *and* j*, with transcripts* T_i *and* T_j*, such that both* $T_i \subseteq T^*$ *and* $T_j \subseteq T^*$*. Then if* Π *is a secure UAKE protocol,* $\Pr[\mathsf{MultipleMatch}]$ *is negligible.*

Proof. Assume by contradiction there exists an adversary $\mathcal{A}$ such that $\Pr[\mathsf{MultipleMatch}]$ is non-negligible. We build another adversary $\mathcal{A}'$ such that $\mathbf{Adv}^{\mathsf{UAKE-Sec}}_{\Pi,\mathcal{A}'}(\lambda)$ is non-negligible. Very intuitively, the basic idea of the proof si that $\mathcal{A}'$ can re-arrange the order in which the messages of session j are delivered so that session j will no longer be matching and thus $\mathcal{A}'$ can legally get the session key revealed. Since session j has still the same messages as the challenge session they also have the same session key, which allows $\mathcal{A}'$ to distinguish and win the game. A more detailed proof follows.

If Q is an upper bounded of the T oracle sessions executed by $\mathcal{A}$ in its run, $\mathcal{A}'$ first chooses random $i', j' \xleftarrow{\$} \{1, \ldots, Q\}$ (with $i' \neq j'$) as a guess on the two sessions i and j for which $\mathsf{MultipleMatch}$ will occur. Without loss of generality, assume that session i ends before session j (i.e., $t_i < t_j$ where t_i and t_j are the timestamps of the last message in session i and session j respectively). Next, $\mathcal{A}'$ simulates the experiment to $\mathcal{A}$ by forwarding all messages to the oracles except for the following change. When $\mathcal{A}$ asks to deliver the last message to the T oracle in session j', $\mathcal{A}'$ does not forward the message to its corresponding oracle of session j', but replies to $\mathcal{A}$ with the response received by its T oracle on the last message in session i'. Next, $\mathcal{A}'$ continues the simulation, and after $\mathcal{A}$ terminates the challenge session, $\mathcal{A}'$ concludes the challenge session too, receives the challenge session key K_b and then: it delivers the last message of session j' to its oracle, and makes a last query on session j' to obtain the session key $K_{j'}$. Finally, $\mathcal{A}'$ outputs 1 if $K_b = K'_j$, and 0 otherwise.

First, we observe that if $\mathsf{MultipleMatch}$ occurs and $i' = i \wedge j' = j$ is true, then the simulation provided by $\mathcal{A}'$ to $\mathcal{A}$ is perfect. Indeed the only difference is the response to the last message of session j'. However, $\mathcal{A}'$ response is the last message of session i' which is the same as the last message of session j' (this follows from $\mathsf{MultipleMatch}$). Second, observe that in the game played by $\mathcal{A}'$ it holds $T_{i'} \subseteq T^*$ but $T_{j'} \not\subseteq T^*$. The latter follows from the fact that in the run of $\mathcal{A}'$, session j' ended *after* the challenge session. Therefore the session key reveal asked by $\mathcal{A}'$ is legal (i.e., $\mathcal{A}'$ is not full ping-pong). So, since the messages of sessions j' and the challenge session are identical, the two sessions have the same session key – $K_0 = K_{j'}$ – that is $\mathcal{A}'$ wins with probability 1. To conclude, observe that $\mathcal{A}'$ advantage is $\Pr[\mathsf{MultipleMatch}]/Q^2$ where the factor $1/Q^2$ is the probability of correctly guessing i, j. □

4 Constructions of UAKE Protocols based on iCCA and iCMA Security

In this section we show two realizations of unilaterally-authenticated key-exchange based on message transmission protocols. The constructions are simple and they essentially show how to obtain a clean and smooth transition from encryption/authentication towards key exchange. The first construction (described in Figure 2) uses an iCCA-secure protocol Π' and a pseudorandom function (see Appendix A.1 for the PRF definition). Our second construction of UAKE (described in Figure 3) uses an IND−CPA-secure key encapsulation mechanism (see Appendix A for the definition of KEM) and an iCMA-secure protocol Π'.

The security of these protocols is proven via the following theorems :

Theorem 2. *If* Π' *is* iCCA*-secure, and* F *is a pseudo-random function, then the protocol* Π *in Figure 2 is a secure UAKE.*

Setting: a key pair $(\mathsf{sendk}', \mathsf{recvk}')$ for an iCCA-secure protocol Π' is generated. $F : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ is a PRF.

$\mathsf{U}(\mathsf{sendk}')$		$\mathsf{T}(\mathsf{recvk}')$
$r \xleftarrow{\$} \{0,1\}^\lambda$	$\boxed{\mathsf{U} \text{ sends } r \text{ to } \mathsf{T} \text{ using } \Pi'}$ $\longrightarrow$	
		Get r'
		If $r' \neq \bot : s \xleftarrow{\$} \{0,1\}^\lambda$
$\mathsf{K_U} \vert c \leftarrow F_r(s)$	$\xleftarrow{\quad c', s \quad}$	$\mathsf{K_T} \vert c' \leftarrow F_{r'}(s)$
If $c' = c$ return $\mathsf{K_U}$		
Else return $\bot$		return $\mathsf{K_T}$

Fig. 2. UAKE from iCCA-secure encryption.

Setting: a key pair $(\mathsf{sendk}', \mathsf{recvk}')$ for an iCMA-secure protocol Π' is generated. $\mathcal{E} = (\mathsf{KG}, \mathsf{Encap}, \mathsf{Decap})$ is a public-key KEM.

$\mathsf{U}(\mathsf{recvk}')$		$\mathsf{T}(\mathsf{sendk}')$
$(\mathsf{ek}', \mathsf{dk}') \xleftarrow{\$} \mathsf{KG}(1^\lambda)$	$\xrightarrow{\quad \mathsf{ek}' \quad}$	$(c, K) \xleftarrow{\$} \mathsf{Encap}(\mathsf{ek}')$
Get $(\mathsf{ek}'' \vert c')$	$\boxed{\mathsf{T} \text{ sends } (\mathsf{ek}' \vert c) \text{ to } \mathsf{U} \text{ using } \Pi'}$	
If $((\mathsf{ek}'' \vert c') \neq \bot$ and $\mathsf{ek}'' = \mathsf{ek}')$	$\longleftarrow$	return K
return $\mathsf{Decap}(\mathsf{dk}', c')$		

Fig. 3. UAKE from iCMA-secure PKMA and IND−CPA-secure KEM.

Proof. To prove the security of Π we define the following hybrid games:

Game 0: this is the real $\mathbf{Exp}^{\mathsf{UAKE-Sec}}_{\Pi,\mathcal{A}}(\lambda)$ experiment.

Game 1: this is the same as Game 0 except for the following modifications:

- In the challenge session, U picks a random string $r^* \xleftarrow{\$} \{0,1\}^\lambda$, but runs the encryption protocol Π' sending message 0. Yet, given the adversary's message c', s^* in the challenge session, U still computes $\mathsf{K_T}^* \vert c^* \leftarrow F_{r^*}(s^*)$, and uses these values as in Game 0 (i.e., to check if $c^* = c'$ and, if so, to define $K_0 = \mathsf{K_T}^*$).
- Whenever an oracle T is queried for the last message on a session in which the Π' portion (i.e., all but the last message) is a ping-pong of the encryption protocol execution of the challenge session, then the oracle T samples a fresh $s \xleftarrow{\$} \{0,1\}^\lambda$, and returns c', s, where $K' \vert c' \leftarrow F_{r^*}(s)$.

Via a simple reduction to the iCCA-security of Π', it is possible to show that there exists $\mathcal{B}$ such that: $|\Pr[G_0] - \Pr[G_1]| \leq 2 \cdot \mathbf{Adv}^{\mathsf{iCCA}}_{\Pi',\mathcal{B}}(\lambda)$.

Game 2: this is the same as Game 1 except that the PRF computations $F_{r^*}(\cdot)$ are replaced by a random function $\mathcal{R}(\cdot)$ (simulated via lazy sampling).

It is not hard to see that under the assumption that F is a PRF Game 2 is computationally indistinguishable from Game 1.

Game 3: Let Forge be the event that in Game 2 the challenge session completes with $K_0 \neq \bot$ while $\mathcal{A}$ is *not* ping-pong. Then Game 3 proceeds exactly as Game 2, except that if Forge occurs, then the game outputs 0 (instead of 1, as is done in $\mathbf{Exp}^{\mathsf{UAKE-Sec}}_{\Pi,\mathcal{A}}(\lambda)$ and in Game 2). Hence, Game 3 and Game 2 are identically distributed unless Forge occurs, i.e., $|\Pr[G_2] - \Pr[G_3]| \leq \Pr[\mathsf{Forge}]$.

We observe that the event Forge occurs if $\mathcal{A}$ sends the correct value $c' = c^*$ to the challenger. However, in Game 2 c^* is generated as $\mathsf{K_T}^* \vert c^* \leftarrow \mathcal{R}(s^*)$ (i.e., it is uniformly at random in $\{0,1\}^\lambda$). Hence, unless $\mathcal{A}$ obtained (c', s^*) from a copy of the T oracle that computed $K' \vert c' \leftarrow \mathcal{R}(s^*)$, we have that $\Pr[\mathsf{Forge}] = \Pr[c' = c^*] = 1/2^\lambda$. Note that if $\mathcal{A}$ is not ping-pong, then we had never computed $\mathcal{R}(s^*)$ in the simulation of T. Indeed, we use $\mathcal{R}$ in the simulation of T only for sessions in which the first portion is a ping-pong of the first portion of the challenge session (the part related to protocol Π'). So, if $\mathcal{A}$ is not ping-pong it must be either that the last message obtained by T in such sessions is different from the one sent by $\mathcal{A}$ in the challenge session; or that $\mathcal{A}$ concluded the challenge session before getting the last message from the oracle.

To conclude the proof, if we analyze the probability that Game 3 outputs 1 (hence, Forge does not occur), then $\mathcal{A}$'s view of Game 3 in the second part (i.e., when $\mathcal{A}$ is given K_b) is exactly the same no matter which is the bit b (both session keys K_0 and K_1 are indeed randomly chosen or they are both $\perp$). Hence, $\mathcal{A}$ has probability $1/2$ of guessing the right $b' = b$. □

Theorem 3. *If Π' is iCMA-secure, and $\mathcal{E}$ is an $\mathsf{IND{-}CPA}$-secure KEM, then the protocol Π in Figure 3 is a secure UAKE.*

Proof. To prove the security of Π we define the following simple hybrid games:

Game 0: This is the same as experiment $\mathbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{UAKE-Sec}}(\lambda)$.

Game 1: Consider experiment Game 0, and let $T^* = \langle(\mathsf{ek}', t_1^*), T^{*'}\rangle$ be the transcript of the challenge session, where $T^{*'}$ is the portion of transcript which corresponds to the run of the protocol Π' (in the challenge session). Similarly, consider the transcripts T_i of all the oracle sessions and write $T_i = \langle(\mathsf{ek}_i', t_1^i), T_i'\rangle$. Let Forge be the event that in the challenge session the protocol Π' completes correctly but $\mathcal{A}$ is not ping-pong.
Game 1 proceeds as Game 0 except that, if Forge occurs, then Game 1 outputs 0 (instead of 1).
Clearly, Game 1 is identical to Game 0 unless Forge occurs, i.e., $|\Pr[G_0] - \Pr[G_1]| \leq \Pr[\mathsf{Forge}]$. Under the assumption that Π' is iCMA-secure, one can easily prove that $\Pr[\mathsf{Forge}]$ is negligible.

So, we are left with bounding $|\Pr[G_1] - 1/2|$. Let us split the event G_1 around the event $K_0 = \perp$. If $K_0 = \perp$, it is easy to see that $\Pr[G_1]$ is at most $1/2$. On the other hand, if $K_0 \neq \perp$ then recall that Game 1 can output 1 only when $\mathcal{A}$ is ping-pong (i.e., Forge does not occur). Then we argue that under the assumption that the scheme $\mathcal{E}$ is $\mathsf{IND{-}CPA}$-secure we have that $p_1 = |\Pr[G_1 \wedge K_0 \neq \perp] - 1/2|$ is negligible. The reduction is straightforward. We provide it below for completeness.

Assume there exists $\mathcal{A}$ such that $p_1 \geq \epsilon$ is non-negligible, then we construct an adversary $\mathcal{B}$ which has non-negligible advantage ϵ/Q against the $\mathsf{IND{-}CPA}$ security of $\mathcal{E}$ (where Q is an upper bound on the number of oracle sessions opened by $\mathcal{A}$). $\mathcal{B}$ receives the public key ek^* and works as follows. It picks two random strings $K_0^*, K_1^* \xleftarrow{\$} \{0,1\}^\lambda$, submits them to its challenger and obtains a ciphertext c^*. Moreover, it initializes a counter $j = 0$ and picks a random integer $\mu \xleftarrow{\$} \{1, \ldots, Q\}$, which represents a guess on which of the Q oracle sessions will be a ping-pong of the challenge session. For simplicity, we restrict such a choice only to oracle sessions such that the first message is ek^* (as this is necessary for $\mathcal{A}$ to be ping-pong). Next, $\mathcal{B}$ generates a pair of keys $(\mathsf{sendk}', \mathsf{recvk}') \xleftarrow{\$} \mathsf{Setup}'(1^\lambda)$ for Π' and runs $\mathcal{A}(\mathsf{recvk}')$. $\mathcal{B}$ sends ek^* as the first message in the challenge session and uses the private key sendk' to simulate the authentication in the answers to all oracle queries to T. To generate the ciphertext $\mathcal{B}$ proceeds as follows. If the adversary sends a public key $\mathsf{ek}_i \neq \mathsf{ek}^*$, $\mathcal{B}$ simply chooses a random $K_i \xleftarrow{\$} \{0,1\}^\lambda$, encrypts $c_i \xleftarrow{\$} \mathsf{Enc}(\mathsf{ek}_i, K_i)$, and runs Π' on (ek_i, c_i). If $\mathsf{ek}_i = \mathsf{ek}^*$, $\mathcal{B}$ first increments j. If $j = \mu$, then $\mathcal{B}$ proceeds by using the challenge ciphertext c^*. Otherwise, it chooses a random K_i and proceeds as before. Now, assume that $\mathcal{A}$ completes the challenge session, and recall that since Forge does not occur $\mathcal{A}$ is ping-pong. If $\mathcal{A}$ completes by sending c^* (i.e., μ was the right guess), then $\mathcal{B}$ returns K_0. Otherwise, if $\mathcal{A}$ does not send c^* or $\mathcal{A}$ asks to reveal the session key on the μ-th oracle session, then $\mathcal{B}$ aborts and outputs a random bit (this is essentially the case that μ was not the right guess). Finally, if there is no abort $\mathcal{B}$ returns the same bit of $\mathcal{A}$.

Notice that as long as $\mathcal{B}$ does not abort, its simulation is perfectly distributed. Thus the choice of μ is perfectly hidden, i.e., $\mathcal{B}$ does not abort with probability $1/Q$. To conclude the proof, observe that if c^* encrypts K_0, then $\mathcal{B}$ is simulating $\mathbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{UAKE-Sec}}(\lambda)$ with bit $b = 0$, whereas if c^* encrypts K_1, then $\mathcal{B}$ is perfectly simulating the distribution of $\mathbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{UAKE-Sec}}(\lambda)$ with $b = 1$. Hence, we have that $\mathcal{B}$ has advantage at least ϵ/Q against the $\mathsf{IND{-}CPA}$ security of $\mathcal{E}$. □

On the connection to authenticators [1]. We note that, due to the similarity between iCMA-secure message transmission and the notion of authenticators from [1], our design approach of Figure 3 is similar to what can be obtained by applying a (unilateral) authenticator to an unauthenticated protocol, such as a one-time KEM. However, the derived protocols are not exactly the same. For example, to obtain our same protocols when using the signature-based authenticator one should slightly deviate from the approach of [1] and consider ek' as the nonce of the authenticator.

More conceptually, while the concrete protocols obtained are similar (but not identical), the two works use very different definitions and construction paths to arrive at these similar protocols. Our interactive PKMA notion is game-based and essentially extends the simple notion of signature schemes, whereas authenticators follow the real/ideal paradigm and also require built-in protection against replay attacks. For instance, a regular signature scheme is a 1-round iCMA secure message transmission, whereas it can be considered an authenticator only with certain restrictions, (as per Remark 1 in [1]).

Instantiations of our protocols. In Section 5.1, we discuss four efficient UAKE protocols resulting from instantiating the generic protocols in Figures 2 and 3 with specific 1- or 2-round iCCA- and iCMA-secure schemes.

About freshness of session keys. It is worth noting that both above protocols have the property that the keyed party T generates the session key in a "fresh" way (by sampling a fresh random s in the protocol of Fig. 2, or by running Encap with fresh coins in the protocol of Fig. 3), even if the first part of the protocol is replayed. Such a freshness property is necessary for the security of the protocols in our model. For instance, one might consider a simpler version of the protocol of Fig. 2 in which T generates $\mathsf{K}_\mathsf{T}|c' \leftarrow G(r)$ using a PRG G. Such a protocol however would not be secure because of the following attack. Consider an instantiation of Π' with a non-interactive CCA encryption scheme. First the adversary plays a ping-pong attack between the challenge session and an oracle session with T: it obtains a real-or-random key K_b. In the second part of the experiment, the adversary starts a new oracle session with T by sending to it the first message of the challenge session. Finally, the adversary makes a last query to T in this second session in order to obtain the corresponding session key. Now, observe that the session key will be the same key as the real key K_0 of the challenge session, and thus the adversary can trivially use it to test whether $K_b = K_0$. To see the legitimacy of the attack note that the second oracle session began *after* the challenge session ended, and thus it does not constitute a full ping-pong. In contrast this attack does not apply to our protocol of Fig. 2: there, even if one replays the first messages, every new session will sample a fresh session key with overwhelming probability.

5 Advanced Security Properties and Concrete Protocols

In this section, we discuss advanced properties of *forward security* and *deniability* for unilaterally-authenticated key-exchange, and then we discuss four possible concrete instantiations of our protocols given in Section 4.

5.1 Concrete Protocol Instantiations

Here we discuss four efficient UAKE protocols resulting from instantiating the generic protocols in Figures 2 and 3 with specific 1- or 2-round iCCA- and iCMA-secure schemes. Before proceeding to the analysis, let us briefly recall the instantiations of the iCCA- and iCMA-secure schemes that we consider. First, note that any IND−CCA encryption scheme is a 1-round iCCA protocol, and similarly any strongly unforgeable signature scheme is a 1-round iCMA protocol. Second, Dodis and Fiore [10] show a 2-round iCCA-secure protocol based solely on IND−CPA security and a 2-round iCMA-secure protocol based on IND−CCA encryption and a MAC. Briefly, the iCCA protocol works as follows: the receiver chooses a "fresh" public key ek (of a 1-bounded IND−CCA encryption) and sends this key, signed, to the sender; the sender encrypts the message using ek. The iCMA protocol instead consists in the receiver sending a random MAC key r to the sender using the $\mathsf{IND{-}CCA}$ encryption, while the sender sends the message authenticated using r.

If we plug these concrete schemes in our UAKE protocols of Figures 2 and 3, we obtain the following four UAKE instantiations that we analyze with a special focus on the properties of forward security vs. deniability:

1. Protocol of Figure 2 where the iCCA protocol Π' is a non-interactive $\mathsf{IND{-}CCA}$ scheme: we obtain a *2-round* UAKE based on $\mathsf{IND{-}CCA}$ that *is (forward) passive deniable* (a perfectly indistinguishable transcript for an honest U is easily simulatable), but it is *not forwardœsecure* (recovering the long-term key recvk' trivially allows to recover r). This protocol recover the unilateral version of SKEME [19] (without PFS).
2. Protocol of Figure 2 where the iCCA protocol Π' is the 2-round protocol in [10] based on $\mathsf{IND{-}CPA}$ security: we obtain a *3-round* UAKE based on $\mathsf{IND{-}CPA}$ security that is *not deniable* (as T signs the first message with a digital signature) but it *is passive forward secure* (since so is the 2-round iCCA protocol, as shown in [10]).
3. Protocol of Figure 3 where the iCMA protocol Π' is a digital signature: we obtain a *2-round* UAKE based on $\mathsf{IND{-}CPA}$ security that is clearly *not deniable* (as T signs c) but it can be shown *passive forward-secure* (as dk' is a short-term key which is deleted once the session is over). It is worth noting that when implementing the KEM with standard DH key-exchange ($\mathsf{ek}' = g^x, c = g^y, K = g^{xy}$) we essentially recover protocol A-DHKE-1 in [25]. A very similar protocol based on $\mathsf{IND{-}CPA}$ KEM is also recovered in the recent, independent, work of Maurer et al. [24].
4. Protocol of Figure 3 where the iCMA protocol Π' is the 2-round PKMA proposed in [10] (called Π_{mac}) which is based on $\mathsf{IND{-}CCA}$ encryption and MACs: we obtain a *2-round* UAKE (as we can piggy-back the first round of Π_{mac} on the first round of the UAKE). Somewhat interestingly, this instantiation achieves *the best possible properties for a 2-round protocol*: it enjoys *both* passive forward deniability (as Π_{mac} is passive forward-deniable) and passive forward security (since dk' is short-term, as in the previous case). The resulting protocol is depicted in Figure **??**, and we note that it essentially recovers the unilateral version of SKEME [19]. Moreover, by using the MAC of [11] and by applying some optimizations[8], we obtain a UAKE protocol based only on CCA security. While for practical efficiency one may use faster MACs, we show this protocol based only on CCA security mostly for elegance. The resulting protocol is depicted in Figure **??**, where we use a "labeled" CCA-secure PKE: $\mathsf{Enc}^L(\mathsf{ek}, m)$ denotes a run of the encryption algorithm to encrypt a message m w.r.t. label L; analogously $\mathsf{Dec}^L(\mathsf{dk}, c)$ denotes decryption w.r.t. label L. We recall that decryption of a ciphertext c w.r.t. L succeeds only if c was created with the same label L.

6 Confirmed Encryption and Confidential Authentication

In this section we introduce two advanced notions of (interactive) PKE and PKMA that we call *confirmed encryption* and *confidential authentication* (ConfPKE and ConfPKMA, for short). The basic idea is to extend encryption in such a way that the sender receives confirmation that the receiver obtained the transmitted message, and to extend authentication so that the transmitted messages remain private. At a high level, these two notions have similarities as they both aim to capture at the same time confidentiality and some notion of integrity. The main difference is which of the two parties obtains such integrity guarantee. This is essentially due to the fact that in the two notions the role of the keyed parties (i.e., who has a public/private key) is swapped.

Confirmed Encryption. To define ConfPKE, consider a message transmission protocol Π defined as in Section 2.1, with the only change that the sender also returns a local output – a plaintext $m \in \mathcal{M}$

[8] By directly observing the MAC of [11], we notice that the ephemeral secret key dk' (which is part of the MAC key with r) is only used for verification, and there is no need to encrypt it inside c; instead, we can use labels to bind ek' with c.

Experiment $\mathbf{Exp}^{\mathsf{ConfEnc}}_{\Pi,\mathcal{A}}(\lambda)$	Experiment $\mathbf{Exp}^{\mathsf{ConfAuth}}_{\Pi,\mathcal{A}}(\lambda)$
1. $b \xleftarrow{\$} \{0,1\}$; $(\mathsf{sendk}, \mathsf{recvk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$	1. $b \xleftarrow{\$} \{0,1\}$; $(\mathsf{sendk}, \mathsf{recvk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
2. $(m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{R}(\mathsf{recvk})}(\mathsf{sendk})$	2. $(m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{S}(\mathsf{sendk},\cdot)}(\mathsf{recvk})$
3. $(m', b') \leftarrow \langle \mathsf{S}(\mathsf{sendk}, m_b), \mathcal{A}^{\mathsf{R}(\mathsf{recvk})}(\mathsf{sendk})\rangle$	3. $(b', m') \leftarrow \langle \mathcal{A}^{\mathsf{S}_1(\mathsf{sendk}, m_b), \mathsf{S}(\mathsf{sendk},\cdot)}(\mathsf{recvk}), \mathsf{R}(\mathsf{recvk})\rangle$
4. If $m' \neq \bot$ and $\mathcal{A}$ is not "ping-pong", then output 1	4. If $m' \neq \bot$ and $\mathcal{A}$ is not "ping-pong", then output 1
5. Else if $\mathcal{A}$ is "full-ping-pong", then output $\tilde{b} \xleftarrow{\$} \{0,1\}$	5. Else if $\mathcal{A}$ is "ping-pong" w.r.t. $\mathsf{S}(\mathsf{sendk}, m_b)$, then output $\tilde{b} \xleftarrow{\$} \{0,1\}$
6. Else if $b' = b$ and $\mathcal{A}$ is not "full-ping-pong", then output 1	6. Else if $b' = b$ and $\mathcal{A}$ is not "ping-pong" w.r.t. $\mathsf{S}(\mathsf{sendk}, m_b)$, then output 1
7. Else output 0.	7. Else output 0.

Fig. 4. Security experiments of ConfPKE and ConfPKMA.

or an error $\bot$ – according to whether it receives evidence that the receiver obtained the transmitted plaintext m. As in UAKE, observe that such a change implies that wlog the receiver always speaks last. Correctness of ConfPKE is thus obtained by extending the one of message transmission protocols so that both sender and receiver output the same message, i.e., $\langle \mathsf{S}(\mathsf{sendk}, m), \mathsf{R}(\mathsf{recvk})\rangle = (m, m)$ holds for all honestly generated keys and all plaintexts $m \in \mathcal{M}$.

For security, we want essentially two properties: confidentiality (no information about the transmitted plaintexts is leaked) and confirmation (the sender is correctly assured that the receiver obtained the transmitted plaintext). To formalize this notion we define experiment $\mathbf{Exp}^{\mathsf{ConfEnc}}_{\Pi,\mathcal{A}}(\lambda)$ in Figure 4. Briefly, it works as follows: given oracle access to the keyed party $\mathsf{R}(\mathsf{recvk})$, $\mathcal{A}$ first chooses two plaintexts m_0, m_1, and then runs a challenge session with $\mathsf{S}(\mathsf{sendk}, m_b)$. Since here R speaks last, as in UAKE we extend the ping-pong definition, and we say that $\mathcal{A}$ is "full-ping-pong" if $\mathcal{A}$ is ping-pong and, in the ping-pong session, $\mathcal{A}$ makes a last query to R that returns m'. $\mathcal{A}$ wins the game in two cases: (line 4) it breaks confirmation by letting S accept for some plaintext and without trivially forwarding messages, or (line 6) it breaks confidentiality by correctly guessing b and without being full-ping-pong. Therefore, we say that Π is a secure ConfPKE scheme if for every PPT $\mathcal{A}$, its advantage $\mathbf{Adv}^{\mathsf{ConfEnc}}_{\Pi,\mathcal{A}}(\lambda) = \left|\Pr[\mathbf{Exp}^{\mathsf{ConfEnc}}_{\Pi,\mathcal{A}}(\lambda) = 1] - \frac{1}{2}\right|$ is negligible.

Confidential Authentication. For confidential authentication, we consider a standard message transmission protocol Π (without any syntactic change), and we say that Π is a secure ConfPKMA if for any PPT $\mathcal{A}$ its advantage $\mathbf{Adv}^{\mathsf{ConfAuth}}_{\Pi,\mathcal{A}}(\lambda) = \left|\Pr[\mathbf{Exp}^{\mathsf{ConfAuth}}_{\Pi,\mathcal{A}}(\lambda) = 1] - \frac{1}{2}\right|$ is negligible. The experiment $\mathbf{Exp}^{\mathsf{ConfAuth}}_{\Pi,\mathcal{A}}(\lambda)$ is described in Figure 4 and is similar in the spirit to the one of ConfPKE, except that the keyed parties are swapped. So, given oracle access to $\mathsf{S}(\mathsf{sendk}, \cdot)$, $\mathcal{A}$ first chooses two plaintexts m_0, m_1 and then runs a challenge session with the receiver. In this session, however, $\mathcal{A}$ is also given oracle access to a single specific sender's copy $\mathsf{S}_1(\mathsf{sendk}, m_b)$ transmitting m_b. $\mathcal{A}$ wins the game in two cases: (line 4) it breaks confirmation by letting S accept for some plaintext and without trivially forwarding messages, or (line 6) it breaks confidentiality by correctly guessing b and without being ping-pong (but notice that in this case we only care about ping-pong w.r.t. to the session transmitting m_b). We remark that although our $\mathsf{ConfAuth}$ security definition considers a single challenge (aka "left-or-right") oracle $\mathsf{S}_1(\mathsf{sendk}, m_b)$, it can be extended to the multi-challenge setting via a standard hybrid argument since here the adversary has also access to (multiple instances of) the sender oracle $\mathsf{S}(\mathsf{sendk}, \cdot)$.

Application to Unilaterally-Authenticated Key-Exchange. We use the notions of ConfPKE and ConfPKMA to obtain a further smooth and clean transition from $\mathsf{iCCA}/\mathsf{iCMA}$ security to unilaterally-authenticated key-exchange. In the following lemmas, we show that by doing either "confirmed encryption of random K" or "confidential authentication of random K" we obtain secure UAKE protocols, that are essentially a re-interpretation of the two UAKE protocols in Figures 2 and 3.

Theorem 4. *Let $\mathsf{\Pi}$ be a message transmission protocol, and let $\mathsf{\Pi}_1$ be the UAKE protocol in which U chooses a random K, sends K to T by running $\mathsf{S}(\mathsf{sendk}, K)$ (and, of course, T runs $\mathsf{R}(\mathsf{recvk})$), and K is used as the session key for both parties (if $\mathsf{\Pi}$ succeeds). If $\mathsf{\Pi}$ is a secure ConfPKE, then $\mathsf{\Pi}_1$ is a secure UAKE.*

Theorem 5. *Let $\mathsf{\Pi}$ be a message transmission protocol, and let $\mathsf{\Pi}_2$ be the UAKE protocol in which: T chooses random K_1, K_2 and sends (K_1, K_2) to U by running $\mathsf{S}(\mathsf{sendk}, K_1|K_2)$; U (running $\mathsf{R}(\mathsf{recvk})$) gets K_1, K_2, sends K_2 back to T, and sets K_1 as its session key; T accepts (and let K_1 be the session key) iff the received K_2 is the same as the one it sent. If $\mathsf{\Pi}$ is a secure ConfPKMA, then $\mathsf{\Pi}_2$ is a secure UAKE.*

We provide a proof sketch for Theorem 4. The proof of Theorem 5 is very similar.

Proof (Sketch). Assume that $\mathcal{A}$ can break the security of the UAKE protocol $\mathsf{\Pi}_1$, we build an adversary $\mathcal{B}$ which breaks the security of the $ConfPKE$ scheme. Essentially, $\mathcal{B}$ runs $\mathcal{A}$ by forwarding all $\mathcal{A}$'s queries to its oracles. In particular, $\mathcal{B}$ simulates the challenge session by choosing two random keys K_0, K_1 so that $\mathcal{B}$'s challenger will run the challenge session with one of these two keys. Once $\mathcal{A}$ sends the last protocol message in the challenge session, $\mathcal{B}$ checks if $\mathcal{A}$ (and thus also $\mathcal{B}$ itself) was ping-pong and proceeds as follows: (i) if $\mathcal{A}$ was not ping-pong, then $\mathcal{B}$ runs $b' \leftarrow \mathcal{A}(\perp)$ and outputs the same b'. (ii) Otherwise, if $\mathcal{A}$ was ping-pong (in this case the sender must accept by correctness), then $\mathcal{B}$ runs $b' \leftarrow \mathcal{A}(K_0)$ and returns b'. To see why $\mathcal{B}$'s simulation is correct, observe that $\mathcal{A}$ has to obey essentially the same rules in the security experiments of $\mathsf{ConfEnc}$ and the one of $\mathsf{UAKE{-}Sec}$. The definition of ping-pong is the same and the only difference between the security experiments is that in $\mathsf{UAKE{-}Sec}$, if U rejects, then $\mathcal{A}$ can win only with probability $1/2$ (recall that in this case it has to distinguish between two keys $K_0 = K_1 = \perp$). In contrast, in $\mathsf{ConfEnc}$, even if the sender rejects, the adversary may have the chance to win the game with probability non-negligibly higher than $1/2$. However, it is not hard to see that this asymmetry between the security definitions is not relevant while proving that $ConfPKE$ implies UAKE. Intuitively, if a non-ping-pong $\mathcal{A}$ makes U accept, the same holds for $\mathcal{B}$ w.r.t. the sender. Otherwise, if a ping-pong $\mathcal{A}$ has non-negligible advantage in distinguishing a real-or-random session key, then $\mathcal{B}$ will also have non-negligible advantage in distinguishing which of the two messages were sent in the challenge session.

Instantiations. Finally, we focus on realizing confirmed encryption and confidential authentication. In particular, we show how to build a ConfPKE scheme based on an iCCA-secure protocol (see Figure 5), and a ConfPKMA scheme based on an iCMA-secure protocol and a (non-interactive) $\mathsf{IND{-}CPA}$-secure PKE scheme (see Figure 6).

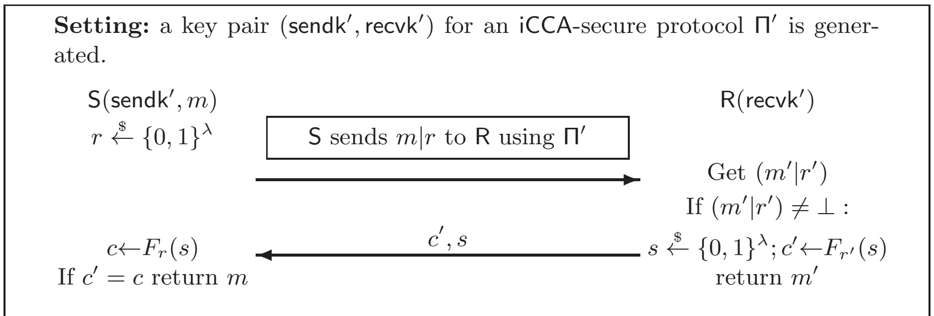

Fig. 5. ConfPKE from iCCA-secure encryption.

Theorem 6. *If $\mathsf{\Pi}'$ is iCCA-secure, then the protocol $\mathsf{\Pi}$ in Figure 5 is a secure ConfPKE scheme.*

Theorem 7. *If $\mathsf{\Pi}'$ is iCMA-secure, and $\mathcal{E}$ is $\mathsf{IND{-}CPA}$-secure, then the protocol $\mathsf{\Pi}$ in Figure 6 is a secure ConfPKMA scheme.*

Setting: a key pair $(\mathsf{sendk}', \mathsf{recvk}')$ for an iCMA-secure protocol Π' is generated. $\mathcal{E} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ is an $\mathsf{IND-CPA}$-secure PKE.

$\mathsf{S}(\mathsf{sendk}', m)$		$\mathsf{R}(\mathsf{recvk}')$
$c \xleftarrow{\$} \mathsf{Enc}(\mathsf{ek}', m)$	$\xleftarrow{\quad \mathsf{ek}' \quad}$	$(\mathsf{ek}', \mathsf{dk}') \xleftarrow{\$} \mathsf{KG}(1^\lambda)$
	$\boxed{\mathsf{S} \text{ sends } (\mathsf{ek}'\|c) \text{ to } \mathsf{R} \text{ using } \Pi'}$	Get $(\mathsf{ek}''\|c')$
	$\longrightarrow$	If $(\mathsf{ek}''\|c') \neq \perp \wedge \mathsf{ek}'' = \mathsf{ek}'$ return $\mathsf{Dec}(\mathsf{dk}', c')$

Fig. 6. ConfPKMA from iCMA-secure PKMA and $\mathsf{IND-CPA}$-secure PKE.

The proofs of security of these two protocols are very similar to the ones of Theorems 2 and 3, and are omitted.

Finally, to see how the intermediate notions of ConfPKE and ConfPKMA offer a smooth transition from $\mathsf{iCCA}/\mathsf{iCMA}$ security towards UAKE, it is interesting to observe that our two constructions of UAKE in Figures 2 and 3 can be seen as the result of applying (with some optimizations) Theorems 4 and 5 to the protocol of Figures 5 and 6 respectively. Precisely, we consider the following optimizations: in the protocol of Figure 2 only a single random value r is sent and we use a PRF with longer outputs to obtain the two strings $K|c$, while the protocol of Figure 3 uses a KEM instead of a PKE.

Acknowledgements. The authors would like to thank the anonymous reviewers for helpful comments on prior submissions of this paper. The research of Yevgeniy Dodis is partially supported by gifts from VMware Labs and Google, and NSF grants 1319051, 1314568, 1065288, 1017471. The research of Dario Fiore is partially supported by the European Commission Seventh Framework Programme Marie Curie Cofund Action AMAROUT-II (grant no. 291803), and the Madrid Regional Government under project PROMETIDOS-CM (ref. S2009/TIC1465).

References

1. M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract). In *30th ACM STOC*, pages 419–428. ACM Press, May 1998.
2. M. Bellare and P. Rogaway. Entity authentication and key distribution. In D. R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 232–249. Springer, Aug. 1993.
3. S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. In M. Darnell, editor, *6th IMA International Conference on Cryptography and Coding*, volume 1355 of *LNCS*, pages 30–45. Springer, Dec. 1997.
4. S. Blake-Wilson and A. Menezes. Authenticated Diffie-Hellman key agreement protocols (invited talk). In S. E. Tavares and H. Meijer, editors, *SAC 1998*, volume 1556 of *LNCS*, pages 339–361. Springer, Aug. 1998.
5. R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In B. Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 453–474. Springer, May 2001.
6. R. Canetti and H. Krawczyk. Universally composable notions of key exchange and secure channels. In L. R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 337–351. Springer, Apr. / May 2002.
7. R. Cramer, G. Hanaoka, D. Hofheinz, H. Imai, E. Kiltz, R. Pass, A. Shelat, and V. Vaikuntanathan. Bounded CCA2-secure encryption. In K. Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 502–518. Springer, Dec. 2007.
8. M. Di Raimondo and R. Gennaro. New approaches for deniable authentication. In V. Atluri, C. Meadows, and A. Juels, editors, *ACM CCS 05*, pages 112–121. ACM Press, Nov. 2005.
9. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
10. Y. Dodis and D. Fiore. Interactive encryption and message authentication. SCN 2014, 2014.
11. Y. Dodis, E. Kiltz, K. Pietrzak, and D. Wichs. Message authentication, revisited. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 355–374. Springer, Apr. 2012.
12. C. Dwork and M. Naor. Zaps and their applications. In *41st FOCS*, pages 283–293. IEEE Computer Society Press, Nov. 2000.
13. C. Dwork, M. Naor, and A. Sahai. Concurrent zero-knowledge. In *30th ACM STOC*, pages 409–418. ACM Press, May 1998.

14. D. Fiore, R. Gennaro, and N. P. Smart. Constructing certificateless encryption and ID-based encryption from ID-based key agreement. In M. Joye, A. Miyaji, and A. Otsuka, editors, *PAIRING 2010*, volume 6487 of *LNCS*, pages 167–186. Springer, Dec. 2010.
15. I. Goldberg, D. Stebila, and B. Ustaoglu. Anonymity and one-way authentication in key exchange protocols. *Designs, Codes and Cryptography*, 67(2):245–269, 2013.
16. T. Jager, F. Kohlar, S. Schäge, and J. Schwenk. On the security of TLS-DHE in the standard model. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 273–293. Springer, Aug. 2012.
17. J. Katz. Efficient and non-malleable proofs of plaintext knowledge and applications. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 211–228. Springer, May 2003.
18. F. Kohlar, S. Schge, and J. Schwenk. On the security of tls-dh and tls-rsa in the standard model. Cryptology ePrint Archive, Report 2013/367, 2013.
19. H. Krawczyk. Skeme: a versatile secure key exchange mechanism for internet. In *Network and Distributed System Security, 1996., Proceedings of the Symposium on*, pages 114 –127, feb 1996.
20. H. Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 546–566. Springer, Aug. 2005.
21. H. Krawczyk, K. G. Paterson, and H. Wee. On the security of the TLS protocol: A systematic analysis. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 429–448. Springer, Aug. 2013.
22. B. A. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. In W. Susilo, J. K. Liu, and Y. Mu, editors, *ProvSec 2007*, volume 4784 of *LNCS*, pages 1–16. Springer, Nov. 2007.
23. U. Maurer and R. Renner. Abstract cryptography. In B. Chazelle, editor, *ICS 2011*, pages 1–21. Tsinghua University Press, Jan. 2011.
24. U. Maurer, B. Tackmann, and S. Coretti. Key exchange with unilateral authentication: Composable security definition and modular protocol design. Cryptology ePrint Archive, Report 2013/555, 2013. `http://eprint.iacr.org/`.
25. V. Shoup. On formal models for secure key exchange. Cryptology ePrint Archive, Report 1999/012, 1999. `http://eprint.iacr.org/`.

A Standard Cryptographic Primitives

We describe notation and recall some basic definitions that will be useful in our work. We denote with $\lambda \in \mathbb{N}$ a security parameter, and we say that a function $\epsilon(\lambda)$ is *negligible* if it is a positive function that vanishes faster than the inverse of any polynomial in λ. If X is a set, we denote with $x \stackrel{\$}{\leftarrow} X$ the process of selecting x uniformly at random in S. An algorithm $\mathcal{A}$ is called PPT if it is a probabilistic Turing machine whose running time is bounded by some polynomial in λ. If $\mathcal{A}$ is a PPT algorithm, then $y \stackrel{\$}{\leftarrow} \mathcal{A}(x)$ indicates the process of running $\mathcal{A}$ on input x and assigning its output to y.

A.1 Pseudorandom Functions

Let λ be the security parameter, and ℓ, L be polynomials in λ. A function $F : \mathcal{K} \times \{0,1\}^\ell \to \{0,1\}^L$, where $\mathcal{K}$ is the key space, is a pseudorandom function (PRF) if for any PPT adversary $\mathcal{A}$ its advantage

$$\mathbf{Adv}_{\mathcal{A},F}(\lambda) = \left| \Pr[\mathcal{A}^{F_K(\cdot)}(1^\lambda) = 1 : K \stackrel{\$}{\leftarrow} \mathcal{K}] - \Pr[\mathcal{A}^{\mathcal{R}(\cdot)}(y) = 1] \right|$$

is at most negligible. Above $\mathcal{R} : \{0,1\}^\ell \to \{0,1\}^L$ denotes a random function.

A.2 (Non-Interactive) CCA-secure Public-Key Encryption

A public key encryption scheme $\mathcal{E}$ is a tuple of algorithms $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ defined as follows:

$\mathsf{KG}(1^\lambda)$ on input the security parameter, the key generation returns a public key ek and a secret key dk.
$\mathsf{Enc}(\mathsf{ek}, m)$ on input the public key ek and a message m, it outputs a ciphertext c.
$\mathsf{Dec}(\mathsf{dk}, c)$ given the secret key dk and a ciphertext c, it outputs a message m or an error symbol $\perp$.

Consider the following experiment involving the scheme $\mathcal{E}$ and an adversary $\mathcal{A}$:

Experiment $\mathbf{Exp}^{\mathsf{IND-CCA}}_{\mathcal{E},\mathcal{A}}(\lambda)$

$b \xleftarrow{\$} \{0,1\}$
$(\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} \mathsf{KG}(1^\lambda)$
$(m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{Dec}(\mathsf{dk},\cdot)}(\mathsf{ek})$
$c^* \xleftarrow{\$} \mathsf{Enc}(\mathsf{ek}, m_b)$
$b' \leftarrow \mathcal{A}^{\mathsf{Dec}(\mathsf{dk},\cdot)}(c^*)$
If $\mathcal{A}$ is not "legal", then output $\tilde{b} \xleftarrow{\$} \{0,1\}$
Else if $b' = b$ and $\mathcal{A}$ is "legal" output 1
Else output 0.

In the above experiment, $\mathcal{A}$ is called "legal" if it does not query the decryption oracle $\mathsf{Dec}(\mathsf{dk}, \cdot)$ on the challenge ciphertext c^* (after $\mathcal{A}$ receives c^*).

The advantage of an adversary $\mathcal{A}$ in breaking the $\mathsf{IND-CCA}$ security of an encryption scheme $\mathcal{E}$ is

$$\mathbf{Adv}^{\mathsf{IND-CCA}}_{\mathcal{E},\mathcal{A}}(\lambda) = \left|\Pr[\mathbf{Exp}^{\mathsf{IND-CCA}}_{\mathcal{E},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}\right|$$

Definition 9 ($\mathsf{IND-CCA}$ security). *An encryption scheme $\mathcal{E}$ is $\mathsf{IND-CCA}$-secure if for any PPT $\mathcal{A}$, $\mathbf{Adv}^{\mathsf{IND-CCA}}_{\mathcal{E},\mathcal{A}}(\lambda)$ is negligible.*

A weaker notion of $\mathsf{IND-CCA}$ security that we consider in our work is *q-bounded* $\mathsf{IND-CCA}$ security [7]. This notion is defined as $\mathsf{IND-CCA}$ security except that the adversary is restricted to query the decryption oracle at most q times (where q is a pre-fixed bound).

A further weaker notion of security for public key encryption is semantic security, or indistinguishability against chosen-plaintext attacks ($\mathsf{IND-CPA}$). Its definition is the same as $\mathsf{IND-CCA}$ security except that the adversary does not get access to any decryption oracle.

Finally, we recall the notion of *key encapsulation mechanism* (KEM) which is closely related to public key encryption. A KEM is defined by three algorithms $(\mathsf{KG}, \mathsf{Encap}, \mathsf{Decap})$: the key generation KG is the same as in PKE; the probabilistic encapsulation algorithm Encap uses the public key ek to generate a ciphertext C and a key K; the decapsulation algorithm Decap takes as input the secret key dk and a ciphertext C and outputs a key K. For correctness, it is required that for all honestly generated pairs of keys $(\mathsf{ek}, \mathsf{dk})$, and $(C, K) \xleftarrow{\$} \mathsf{Encap}(\mathsf{ek})$ it holds $K = \mathsf{Decap}(\mathsf{dk}, C)$. The security definition of KEM is basically the same as that of PKE except that the goal of the adversary is to distinguish an honestly generated session key from a random one. It is worth noting that the standard Diffie-Hellman protocol (aka a simplified version of ElGamal) is a KEM: Let $\mathbb{G}$ be a cyclic group where DDH holds and $g \in \mathbb{G}$ be a generator: KG outputs $\mathsf{ek} = g^x$ and $\mathsf{dk} = x$ for a random x, Encap outputs $C = g^y$ $K = g^{xy}$ for a random y, and $\mathsf{Decap}(\mathsf{dk}, C)$ recovers the same $K = C^x = g^{xy}$.

A.3 Digital Signatures

A digital signature scheme consists of a triple of algorithms $\Sigma = (\mathsf{\Sigma.kg}, \mathsf{Sign}, \mathsf{Ver})$ working as follows:

$\mathsf{\Sigma.kg}(1^\lambda)$ the key generation takes as input a security parameter λ and returns a pair of keys $(\mathsf{sk}, \mathsf{vk})$.
$\mathsf{Sign}(\mathsf{sk}, m)$ on input a signing key sk and a message m, the signing algorithm produces a signature σ.
$\mathsf{Ver}(\mathsf{vk}, m, \sigma)$ given a triple vk, m, σ the verification algorithm tests if σ is a valid signature on m with respect to verification key vk.

For security we define the following experiment:

Experiment $\mathbf{Exp}^{\mathsf{uf\text{-}cma}}_{\mathcal{A},\Sigma}(\lambda)$

$(\mathsf{sk}, \mathsf{vk}) \xleftarrow{\$} \mathsf{\Sigma.kg}(1^\lambda)$

$(m^*, \sigma^*) \xleftarrow{\$} \mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{vk})$
If $\mathsf{Ver}(\mathsf{vk}, m^*, \sigma^*) = 1$ and (m^*, σ^*) is "new" then output 1
Else Output 0

We say that the forgery (m^*, σ^*) is "new" if it is different from all the pairs (m_i, σ_i) obtained from the signing oracle $\mathsf{Sign}(\mathsf{sk}, \cdot)$. We define the advantage of an adversary $\mathcal{A}$ in breaking the strong unforgeability against chosen-message attacks ($\mathsf{suf\text{-}cma}$) of Σ as $\mathbf{Adv}^{\mathsf{suf\text{-}cma}}_{\mathcal{A},\Sigma}(\lambda) = \Pr[\mathbf{Exp}^{\mathsf{suf\text{-}cma}}_{\mathcal{A},\Sigma}(\lambda) = 1]$.

Definition 10 ($\mathsf{suf\text{-}cma}$ security). *A digital signature scheme Σ is $\mathsf{suf\text{-}cma}$-secure if for any PPT $\mathcal{A}$, $\mathbf{Adv}^{\mathsf{suf\text{-}cma}}_{\mathcal{A},\Sigma}(\lambda)$ is negligible.*

A weaker notion of security is (simple) unforgeability against chosen-message attacks ($\mathsf{uf\text{-}cma}$), which is defined as the strong version above, except that (m^*, σ^*) is considered "new" if only the message m^* (instead of the pair) is different from all messages m_i queried to the signing oracle.

A.4 Message Authentication Codes

A message authentication code consists of a triple of algorithms $\mathsf{MAC} = (\mathsf{Gen}, \mathsf{Tag}, \mathsf{Ver})$ working as follows:

$\mathsf{Gen}(1^\lambda)$: the key generation algorithm takes as input the security parameter λ and returns a key $k \in \mathcal{K}$.
$\mathsf{Tag}(k, m)$: on input a secret key $k \in \mathcal{K}$ and a message $m \in \mathcal{M}$, the authentication algorithm produces an authentication tag σ.
$\mathsf{Ver}(k, m, \sigma)$: given the secret key k, a message m and an authentication tag σ, the verification algorithm tests if σ correctly authenticates m.

A scheme MAC is correct if for all $\lambda \in \mathbb{N}$ and $m \in \mathcal{M}$, the probability $\Pr[\mathsf{Ver}(k, m, \sigma) = 1 : k \xleftarrow{\$} \mathsf{Gen}(1^\lambda), \sigma \xleftarrow{\$} \mathsf{Tag}(k, m)]$ is overwhelming. The security is defined via the following experiment:

Experiment $\mathbf{Exp}^{\mathsf{suf\text{-}cmva}}_{\mathcal{A},\mathsf{MAC}}(\lambda)$
$k \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$
Run $\mathcal{A}^{\mathsf{Tag}(k,\cdot),\mathsf{Ver}(k,\cdot,\cdot)}(\lambda)$
If $\mathcal{A}$ makes a verification query (m^*, σ^*) such that
$\mathsf{Ver}(k, m^*, \sigma^*) = 1$ and (m^*, σ^*) is "new",
then output 1.
Else output 0

where for (m^*, σ^*) being "new" we mean that it must be different from all the pairs (m_i, σ_i) obtained from the tag oracle $\mathsf{Tag}(k, \cdot)$. The advantage of an adversary $\mathcal{A}$ in breaking the strong unforgeability against chosen-message and chosen verification queries attacks ($\mathsf{suf\text{-}cmva}$) of MAC is $\mathbf{Adv}^{\mathsf{suf\text{-}cmva}}_{\mathcal{A},\mathsf{MAC}}(\lambda) = \Pr[\mathbf{Exp}^{\mathsf{suf\text{-}cmva}}_{\mathcal{A},\mathsf{MAC}}(\lambda) = 1]$.

Definition 11 ($\mathsf{suf\text{-}cmva}$ security). *A message authentication code MAC is $\mathsf{suf\text{-}cmva}$-secure if for any PPT $\mathcal{A}$, $\mathbf{Adv}^{\mathsf{suf\text{-}cmva}}_{\mathcal{A},\mathsf{MAC}}(\lambda)$ is negligible.*

B The Bellare-Rogaway security model in the unilateral setting

Here we briefly recall the Bellare-Rogaway security model for authenticated key-exchange [2], in a version which directly considers the case where only one of the two parties is authenticated. For syntax, we identify the parties as in our UAKE definition. Namely, we have a single keyed (aka authenticated) party T and an unkeyed party U. The security is defined by a game between an adversary $\mathcal{A}$ and a challenger. At the beginning of the game $\mathcal{A}$ is given the public key uk of the keyed party T. During the game the adversary is given access to several oracles $\Pi^s_{i,j}$ where $i, j \in \{\mathsf{U}, \mathsf{T}\}$ and $1 \le s \le Q$. Oracle $\Pi^s_{i,j}$ models the party i which attempts to establish a key with party j in the s-th session. Precisely, since $\mathcal{A}$ can simulate oracles $\Pi_{\mathsf{U},\mathsf{T}}$ on its own (they do not require any secret), we assume that $\mathcal{A}$ has access to a single oracle $\Pi_{\mathsf{U},\mathsf{T}}$ (representing an honest client) while it can query as many oracles $\Pi^s_{\mathsf{T},\mathsf{U}}$ as it wishes.

The adversary $\mathcal{A}$ interacts with the oracles by sending messages of the form (i, j, s, M), where such a tuple is intended to mean that $\mathcal{A}$ sends a message M to party i, claiming it is from party j in the session s. Each oracle $\mathcal{O} = \Pi^s_{i,j}$ maintains some meta-information:

- $\delta_{\mathcal{O}} \in \{\perp, accepted, error\}$ which determines whether the session is in a finished state or not;
- $\gamma_{\mathcal{O}} \in \{\perp, revealed\}$, which signals whether the oracle has been "opened" (by revealing the session key) or not;
- $K_{\mathcal{O}}$ which denotes the session key of the protocol run, if the protocol has completed.

Basically, the oracle $\mathcal{O} = \Pi^s_{i,j}$ models a copy of party i when running a protocol session with party j. So, on a message (i, j, s, M) from the adversary, the oracle $\Pi^s_{i,j}$ answers with the message M' generated by the corresponding copy of i and by revealing the state $\delta_{\mathcal{O}}$.

In addition to sending messages to an oracle, the adversary can also make a query $Reveal(\mathcal{O})$ which is answered as follows: If $\delta_{\mathcal{O}} \neq accepted$ then return $\perp$, otherwise it returns $K_{\mathcal{O}}$ and $\gamma_{\mathcal{O}}$ is then changed to *revealed*.

Finally, the adversary can make a single query $Test(\mathcal{O}^*)$ on the oracle $\mathcal{O}^* = \Pi_{\mathsf{U},\mathsf{T}}$ which must be *fresh* (see below for the definition of freshness). In this case, the challenger selects a bit $b \in \{0, 1\}$. If $b = 0$ then the challenger responds with the value of $K_{\mathcal{O}^*}$, otherwise it responds with a random key chosen from the space of session keys. We call the oracle $\mathcal{O}^*$ on which $Test(\mathcal{O}^*)$ is called the "Test-oracle". At the end of the game, the adversary has to output a bit b'.

The oracle $\mathcal{O}^* = \Pi_{\mathsf{U},\mathsf{T}}$ is said to be *fresh* if: (1) $\delta_{\mathcal{O}^*} = accepted$, (2) $\gamma^*_{\mathcal{O}} \neq revealed$, (3) there is no oracle $\mathcal{O}'$ with $\gamma_{\mathcal{O}'} = revealed$ with which $\mathcal{O}^*$ has had a matching conversation. After the $Test(\mathcal{O}^*)$ query has been made, the adversary can continue making queries as before, except that it cannot call $Reveal(\mathcal{O}')$ on an oracle $\mathcal{O}'$ that is partner of $\mathcal{O}^*$, if $\mathcal{O}'$ exists, and it cannot call $Reveal(\mathcal{O}^*)$.[9]

For the notion of matching conversations between two oracles, we recall this is basically the same as our notion of matching transcripts. Namely, it combines equality of transcripts and interleaving of timestamps.

Now, the following notation and events are defined in the above experiment. An adversary $\mathcal{A}$ is called *benign* if it faithfully conveys messages between two oracles $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$. Next, $NoMatch$ is the event that $\Pi_{\mathsf{U},\mathsf{T}}$ accepted but there is no oracle $\Pi^s_{\mathsf{T},\mathsf{U}}$ which has a matching conversation with $\Pi_{\mathsf{U},\mathsf{T}}$. Finally, $GoodGuess$ is the event that at the end of the experiment $b' = b$.

Definition 12. *Then a protocol Π is a secure unilateral authenticated key exchange protocol in the BR model described above if for any PPT adversary $\mathcal{A}$ running in the game described above the following conditions hold:*

1. *the probability of $NoMatch$ is negligible (i.e., the protocol is a secure mutual authentication protocol);*
2. *in the presence of a benign adversary, which faithfully conveys messages on $\Pi_{\mathsf{U},\mathsf{T}}$ and $\Pi^s_{\mathsf{T},\mathsf{U}}$, both oracles always accept holding the same session key, and this key is distributed as an honest session key;*
3. *For any PPT $\mathcal{A}$ running in the experiment described above, we have that $|\Pr[GoodGuess] - 1/2|$ is negligible.*

[9] Notice, allowing test queries on oracles $\Pi^s_{\mathsf{T},\mathsf{U}}$ would not make sense in the unilateral setting as the partner of such oracle can be the adversary itself. So, we want to guarantee security for session keys established by an honest client U.