Witness-Authenticated Key Exchange Revisited

Improved Models, Simpler Constructions, Extensions to Groups

Matteo Campanelli[1], Rosario Gennaro[1,2], Kelsey Melissaris[2], and Luca Nizzardo[1]

[1] Protocol Labs
{matteo,rosario.gennaro,luca.nizzardo}@protocol.ai
[2] City University of New York
kelseymelissaris@gmail.com

Abstract. We revisit the notion of Witness Authenticated Key Exchange (WAKE) where a party can be authenticated through a generic witness to an NP statement. We point out shortcomings of previous definitions, protocols and security proofs in Ngo et al. (Financial Cryptography 2021) for the (unilaretally-authenticated) two-party case. In order to overcome these limitations we introduce new models and protocols, including *the first definition* in literature of group witness-authenticated key exchange. We provide simple constructions based on (succinct) signatures of knowledge. Finally, we discuss their concrete performance for several practical applications in highly decentralized networks.

1 Introduction

Public-Key Cryptography as introduced in the seminal paper by Diffie and Hellman [DH76] allows two parties who have never met to exchange confidential information by either interactively establishing a secret key or by means of a single encrypted message from the sender to the receiver [RSA78]. The identity of the communicating parties is traditionally established through the use of *certificates:* digital signatures from trusted authorities binding the parties' identities to their public keys used to encrypt the messages.

More flexible ways to determine the recipient of an encrypted message were devised with *identity-based* [Sha84] (where public keys can be arbitrary strings) and *attribute-based* [SW05] encryption (where messages are encrypted under security policies and only parties holding attributes satisfying the policy can decrypt) removing the need to obtain certificates in advance of sending the message.[3]

Witness encryption [GGSW13] is arguably the most general way to determine the intended recipient of an encrypted message. In WE a message is encrypted under a specific instance ϕ of an NP language L, and can be decrypted if and only if $\phi \in L$, using an efficient decryption procedure that takes a witness w for ϕ as input[4]. Remarkably, no trusted party is required for WE: the secret decryption key is the witness itself.

In a theoretical sense all of these notions only make sense in the non-interactive setting of a sender encrypting a message for a receiver. Indeed, in the interactive key agreement setting, certificates can be sent as part of the communication flow in the case of IBE/ABE and generic secure two-party computation techniques can be used to establish a secure channel between a sender and a receiver who knows a specific witness. However interactive versions of these models have been introduced mostly for efficiency reasons (e.g. [FG10, BBC$^+$13, CCGS10, NMKW21]).

Another, perhaps more important, advantage of looking at interactive key agreement in these models is that they additionally allow for the establishment of *group* keys which enable confidential communication among arbitrary numbers of authorized participants.

In this paper we revisit the notion of witness authenticated key exchange (WAKE) introduced in [NMKW21], by (i) pointing out shortcomings in their definition and proofs, and (ii) proposing a new and much simpler protocol for the two-party case. We then (iii) extend our results to the group setting and construct the first witness-based group key agreement protocol.

[3] In these models there is a trusted party which issues secret keys to the users matching respectively their identity and their attributes (and as a consequence can decrypt all messages).

[4] The original notion requires that a witness is sufficient to decrypt but does not guarantee that knowledge of w is necessary for a successful decryption.

1.1 Motivation

As mentioned above, the main application of witness based key exchange is the establishment of secure communication channels based on arbitrary conditions that are satisfied by what the parties know or hold. The lack of a centralized trusted authority that issues secret keys or certificates makes this tool particularly interesting for decentralized applications, where parties can dynamically and flexibly confidentially connect with other parties based on common policies. Decoupling authentication and the notion of identity also allows for more flexible deniable and anonymous authentication. In this section we present some concrete examples, many of them exploiting our novel construction for groups.

Dark Pools Transactions. This was the original motivation in [NMKW21]. In this scenario Alice wants to confidentially negotiate with a party who has enough funds. Given a public commitment to his balance Bob can establish a secret key with Alice if his funds satisfy her condition. Here the witness w is the balance held by Bob and the NP relation that must be satisfied is that w is the correct committed value and satisfies Alice's conditions.

Chat with the same wallet. Several services are offered that allow parties to create chatrooms and schedule meetings amongst parties that hold similar tokens in a blockchain (e.g. [cwe, mee]). A group witness-authenticated key exchange can be used to establish secure communication channels for such tasks. Thanks to the inherent flexibility of a witness key agreement these schemes can also be extended to more general conditions (e.g. confidential chatrooms for owners of NFTs by a particular artist).

Retrieval Markets. In decentralized storage systems such as Filecoin [Proa] and IPFS [Prob] files are stored by providers and addressed by a *content identifier (CID)* which is basically a cryptographic hash of the file. While the CID does not say where the file is stored, it does provide a unique handle for the file to be retrieved later. The CID also serves as a commitment to the file and therefore providers can use the file itself as a witness to establish confidential negotiation channels with clients interested in its retrieval. Similarly, providers storing the same file (or files satisfying certain properties) can establish a confidential group channel to communicate via a group witness key agreement.

Decentralized Anonymous Routing. Several proposals have been put forward for decentralized naming and routing protocols over the internet (see e.g. [han] and [ens]). We believe that WAKE can play an important role in securing such protocols since it provides a method with which parties can "authenticate" themselves without the need for centralized trusted authorities.

1.2 Our Contributions

- **A critique of the previous model and construction in [NMKW21]:** we observe that the definition of *witness key agreement* in [NMKW21] is not sufficient for the proposed applications. We also point out limitations of their construction.
- **A new definition for WAKE:** we rectify the above limitations and propose a new definition for (group) WAKE (Witness Authenticated Key Exchange).
- **A construction of U-WAKE**: we provide a definition for *Unilaterally* Witness Authenticated Key Exchange (two-party key exchange where only one party is authenticated) as a special case of WAKE, along with a simple construction for U-WAKE. This construction generalizes UAKE [DF17] in the standard setting.
- **A construction of (group) WAKE**: we show a compiler turning any key exchange with "passive security" into one with full security in the witness-based setting. This compiler revisits that in [KY03] for standard group key exchange. We also show a three-round WAKE protocol, obtained by applying our compiler to the passively secure key exchange from [BD95].
- **Evaluation in practice and applications:** we show the feasibility of our construction by experimentally evaluating its efficiency.

1.3 Technical Overview

Definition of WAKE. Our WAKE definition extends the definition of unilaterally-authenticated key agreement from [DF17] in the PKI model to work in the case of witness authentication. This is not a trivial

task: one main difference is how to model the adversary's knowledge with respect to the NP instance ϕ, acting as the "public key" for the authenticating party. In the PKI case the adversary does not know the secret key of the honest parties, but in the WAKE case it is conceivable that for a specific instance ϕ the adversary may know the corresponding witness w. Under these circumstances obviously we cannot prevent the adversary from successfully authenticating, but we *can* still require that the adversary is not able to decrypt conversations involving any honest parties that have authenticated with respect to the same instance ϕ. This is somewhat the equivalent of perfect forward secrecy in the PKI case, where compromise of the long-term secret key of a party does not compromise the secrecy of the session keys established by that party. Our definition requires the extraction of a witness from any adversary that successfully authenticates and completes a WAKE.

By leveraging interaction we obtain stronger security properties than (non-interactive) witness encryption, both in terms of confidentiality (parties can confidentially exchange a secret even if the adversary knows a witness) and authentication (we guarantee knowledge of a witness upon successful authentication). We finally remark that our confidentiality definition also guarantees perfect forward secrecy.

Simulatability. In general it is not necessary to enforce that no information about the witness is leaked during the protocol as long as we can prove that knowledge of the witness is required to successfully authenticate. This brings up an interesting question: assume we have a protocol where the adversary can learn a bit of the witness. After several executions, the adversary will learn the witness and be able to authenticate and perform the WAKE on its own. This would obviously be a problem in PKI-based KA protocols, as the adversary would be able to impersonate a different party. But in WAKE such a protocol does not violate the definition of security since once the adversary is able to complete the protocol it does *know* the witness. This leads us to introduce an additional *simulatability* condition enforcing that no information about the witness is leaked.

Our Protocol. Our two-party protocol uses Signatures of Knowledge [CL06] to perform witness authentication. The initiator (who does not have the witness) sends the "public key" for a Key Encapsulation Module (KEM) and the responder sends the KEM message "signed" with a Signature of Knowledge of the witness. The initiator accepts if the signature verifies and then decrypts the session key from the KEM message. We point out that [NMKW21] had ruled out the use of SOK, due to problems with their proposed solution but our solution is different from that proposed in [NMKW21].

Group WAKE Construction. The group protocol adapts the Katz-Yung [KY03] general compiler (which maps unauthenticated group key exchange protocols to authenticated ones) to be able to use Signatures of Knowledge. Then, following [KY03], we obtain a 3-round group WAKE by applying our transformation to the Burmester-Desmedt [BD95] group key exchange. We point out that in our group protocols, each party may refer to a different NP statement when authenticating, if required by the authentication policy in place.

1.4 Related Work

1.4.1 Comparison to and Limitations of [NMKW21] Model and Definitions: A basic requirement of key agreement schemes is that one should not to able to learn anything about the session-key. This is usually modeled by requiring that the adversary should not be able to distinguish the real session key from a random key (in a session not tampered by the adversary obviously). Indistinguishability is required to claim that exchanging messages in a session protected by the key is equivalent to sending those messages over a secure channel. We note that an adversary can either be passive (eavesdropping exchanges by honest players), or active (man in the middle).

The definition in [NMKW21] does not follow the typical modeling of authenticated key agreement protocols, and in doing so weakens the security of the session key. In fact, indistinguishability is only required for passive adversaries. A separate definition for active adversaries only requires *unpredictability* of the session key.

Constructions. Both our construction and that in [NMKW21] rely on non-interactive arguments of knowledge; [NMKW21] employs designated-verifier SNARKs while we use succint, publicly-verifiable signatures of knowledge.

A primary limitation of [NMKW21] is that it requires a trusted setup *every time* a new party wants to initiate a key exchange[5] (see Fig. 4 and end of Section 4 in [NMKW21]). This is due to the designated-verifier argument systems. Although the verifier often has access to "all the secrets" used during parameter generation it is still necessary for a trusted party to ensure that the parameters are computed correctly. This is true of the specific instantiation considered, for example ([Gro16]). We provide general and modular constructions, along with concrete instantiations, without this limitation.

Finally, we find it unclear whether the security of the construction in [NMKW21] actually holds. Their construction and proof use a variant of the techniques in [BCI+13] where secret points in the setup are encrypted with a limited-malleability encryption scheme. The construction in [NMKW21] diverges, however, in that they add additional encryptions of the randomness used to encrypt the other ciphertexts. This introduces additional leakage and plausibly requires a stronger encryption scheme: one with randomness-dependent message security [BCPT13]. Nonetheless, neither the theorem statement nor the proof in [NMKW21] explicitly acknowledges this fact.

1.4.2 LAKE Language Authenticated Key Exchange LAKE [BBC+13] (and its predecessor CAKE [CCGS10]) enables two parties to establish a shared key over an insecure network. Authentication is done on the basis of words in languages; participants terminate with a common session key if and only if each participant has knowledge of a word that lies in the language defined by their partner. This implies that not only does the witness remain secret, but that the language and the statement are also secret. In contrast, our definition of WAKE does not guarantee secrecy of the statement. Additionally, LAKE is defined in the UC setting with a common reference string while our WAKE definitions are game-based.

The construction provided handles algebraic languages (languages which admit a smooth projective hash function [CS98]) and therefore is not as general as WAKE. There is no mention of group key agreement in [BBC+13, CCGS10], whereas WAKE is defined for groups. The construction provided in [BBC+13] gives a three round protocol with an additional preliminary round. Our two-party U-WAKE protocol, when run bilaterally, can achieve WAKE for two parties in two rounds.

1.4.3 Conditional Disclosure of Secrets Conditional disclosure of secrets (CDS) [GIKM98] is an interactive primitive that allows a Sender to disclose a secret message m to a Receiver holding some secret input w under some condition C on w. We now show a CDS protocol based on Fully Homomorphic Encryption (FHE)[6] for any NP Language. On input a language L and an instance ϕ, let $\mathcal{R}$ be the corresponding relation for L (i.e. $\mathcal{R}(\phi, w) = 1$ if $\phi \in L$ and w is the witness). The Initiator (who owns the witness) sends an FHE public key E and the value $c = E[w]$ to the Responder, who, using the FHE property, sends back $c' = E[r(R(\phi, w) - 1) + m]$ where r is a uniform random value. The Initiator can now decrypt c' to m if $(\phi, w) \in \mathcal{R}$ or a random value otherwise.

It would be tempting to use the above protocol to establish a session key (by setting the session key K as m in the above protocol). However that would lead to an easy "malleability" attack, where instead of computing $r(1-\mathcal{R}(\phi, w))+K$, the responder computes $r(1-\mathcal{R}(\phi, w))+K'$ where K' is a session key related to K. This will break the indistinguishability condition on K.

The above attack can probably be thwarted by adding a zero knowledge proof that the Responder really computed $r(1 - \mathcal{R}(\phi, w)) + K$ and knows r, K. But this brings us back to our solution requiring the computation of a ZK proof over the relation $\mathcal{R}$, this time with the FHE overhead on top. On the other hand it has the advantage of putting the cost of the ZK proof on the party who does *not* know the witness which could be an advantage in some cases (e.g. when the party with the witness has to establish many sessions).

1.5 Outline

We define our mode for (group) WAKE in section 3.1. We then specialize it to the unilaterally-authenticated case in section 4; we provide a simple construction in that same section. We show a compiler from passively

[5] Although, if the same party wants to run multiple key agreements on the same relation, it could potentially reuse the setup.

[6] It is possible to build CDS also from additively homomorphic encryption [AIR01] where however communication grows linearly in the description of the condition C

secure to actively secure (group) WAKE in section 5; we elaborate on how to instantiate it in section 6. Finally in Sections 7 and 8 we respectively show how to optimize our constructions with an offline preprocessing and discuss their concrete practical costs .

2 Preliminaries

2.1 Notation

The concatenation of two strings, $a, b \in \{0,1\}^*$, is denoted $a||b$, the concatenation of two vectors $u = (u_1, \ldots, u_n)$ and $v = (v_1, \ldots, v_m)$ is denoted $u||v = (u_1, \ldots, u_n, v_1, \ldots, v_m)$. We write $x \leftarrow X$ to denote sampling the element x uniformly at random from the set X, $x \leftarrow \mathcal{D}$ to denote sampling x according to distribution $\mathcal{D}$, and $x \leftarrow A(y)$ to denote running algorithm A on input y to get output x.

The security parameter is denoted λ and is considered public. We say a function f is negligible if $|f(\lambda)| = \lambda^{-\omega(1)}$. PPT is used to denote probabilistic polynomial time. A participant U is initialized with inputs $(\mathsf{input}_1, \ldots, \mathsf{input}_n)$ using square brackets: $U[\mathsf{input}_1, \ldots, \mathsf{input}_n]$. A protocol run between participants $U_1, \ldots, U_\ell$ is written as $\langle U_1, \ldots, U_\ell \rangle$. For an oracle $\mathcal{O}$ we use $A^{\mathcal{O}}$ to say that algorithm A has access to oracle $\mathcal{O}$. The keyspace in a key exchange is denoted $\mathcal{K}$.

2.2 Key Encapsulation Mechanism

Definition 1 (Key Encapsulation Mechanism (KEM)). *A key encapsulation mechanism is defined by a triple of algorithms* $(\mathsf{KG}, \mathsf{Encap}, \mathsf{Decap})$ *with the following syntax:*

$\mathsf{KG}(1^\lambda) \to (\mathsf{ek}, \mathsf{dk})$ *: the key generation algorithms is randomized and outputs an encapsulation and a decapsulation key.*

$\mathsf{Encap}(\mathsf{ek}) \to (C, \mathsf{k})$ *: the encapsulation algorithm is randomized and outputs a ciphertext C and a session key K.*

$\mathsf{Decap}(\mathsf{dk}, C) \to \mathsf{k}$ *: the decapsulation algorithm is deterministic and retrieves the session key from the ciphertext and the decapsulation key.*

For correctness we require that for all $(\mathsf{ek}, \mathsf{dk})$ output by the key generation algorithm, $\mathsf{k}_e = \mathsf{k}_d$ for $\mathsf{Encap}(\mathsf{ek}) \to (C, \mathsf{k}_e)$ and $\mathsf{Decap}(\mathsf{dk}, C) \to \mathsf{k}_d$. For CPA security we require that an adversary cannot distinguish the real session key from a random one.

Definition 2 (KEM-CPA). *We say that a* KEM *is* CPA*-secure if for any PPT adversary $\mathcal{A}$ and for any $\lambda \in \mathbb{N}$ the advantage of $\mathcal{A}$, as defined, is negligible:*

$$\mathbf{Adv}_{\mathsf{KEM\text{-}CPA},\mathcal{A}}(\lambda) := \mathbf{2} \cdot \Pr[\mathbf{Exp}_{\mathsf{KEM\text{-}CPA},\mathcal{A}}(\lambda) = \mathbf{1}] - \mathbf{1} \leq \mathsf{negl}(\lambda)$$

where $\mathbf{Exp}_{\mathsf{KEM\text{-}CPA},\mathcal{A}}$ *is defined in fig. 1.*

2.3 Signature of Knowledge

A signature of knowledge allows signers to produce signatures that verify under the condition that they were generated with knowledge of a witness to associated statement ϕ.

Definition 3 (Signature of Knowledge (SOK)). *A Signature of Knowledge is a tuple of four efficient algorithms* $(\mathsf{SSetup}, \mathsf{SSign}, \mathsf{SVfy}, \mathsf{SSimSetup}, \mathsf{SSimSign})$*, where $\mathcal{R}$ is a relation generator and $\{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ a sequence of message spaces, with the following syntax:*

$\mathsf{SSetup}(1^\lambda, R) \to pp$ *: the setup algorithm is randomized and takes as input a relation $R \in \mathcal{R}_\lambda$, and the security parameter λ, and returns public parameters pp.*

$\mathsf{SSign}(pp, \phi, w, m) \to \sigma$ *: the signing algorithm is randomized and takes as input the public parameters pp, an instance-witness pair in the relation $(\phi, w) \in R$ and a message $m \in \mathcal{M}_\lambda$ and returns a signature σ.*

$\mathbf{Exp}^{\mathsf{CPA}}_{\mathsf{KEM},\mathcal{A}}(\lambda)$

$b \leftarrow\!\!\$\ \{0,1\}$
$(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KEM.KG}(1^\lambda)$
$(C, \mathsf{k}_1) \leftarrow \mathsf{KEM.Encap}(\mathsf{ek})$
$\mathsf{k}_0 \xleftarrow{\$} \mathcal{K}$
$b' \leftarrow \mathcal{A}(\mathsf{ek}, C, \mathsf{k}_b)$
if $b' = b$: output 1
else : output 0

Fig. 1: $\mathbf{Exp}^{\mathsf{CPA}}_{\mathsf{KEM},\mathcal{A}}$: Experiment for KEM-CPA security.

$\mathbf{Exp}^{\mathsf{simul}}_{\mathsf{SOK},\mathcal{A}}(\lambda)$

$R \leftarrow \mathcal{R}_\lambda;\ b \xleftarrow{\$} \{0,1\}$
$pp_0 \leftarrow \mathsf{SSetup}(R)$
$(pp_1, \tau) \leftarrow \mathsf{SSimSetup}(R)$
$b' \leftarrow \mathcal{A}^{\mathcal{S}^b_{pp_b,\tau}}(pp_b)$
if $b = b'$: return 1
else : return 0

$\mathcal{S}^0_{pp_0,\tau}(\phi_i, w_i, m_i)$

assert $(\phi_i, w_i) \in R \wedge m_i \in \mathcal{M}_\lambda$
$\sigma_i \leftarrow \mathsf{SSign}(pp_0, \phi, w, m)$
return σ_i

$\mathcal{S}^1_{pp_1,\tau}(\phi_i, w_i, m_i)$

assert $(\phi_i, w_i) \in R \wedge m_i \in \mathcal{M}_\lambda$
$\sigma_i \leftarrow \mathsf{SSimSign}(pp_1, \tau, \phi, m)$
return σ_i

Fig. 2: $\mathbf{Exp}^{\mathsf{simul}}_{\mathsf{SOK},\mathcal{A}}$: Experiment for SOK perfect simulatability.

$\mathsf{SVfy}(pp, \phi, m, \sigma) \to \{0,1\}$ *: the verification algorithm is deterministic and takes as input the public parameters pp, an instance ϕ, a message $m \in \mathcal{M}_\lambda$ and a signature σ, and outputs either 0 for reject or 1 for accept.*

$\mathsf{SSimSetup}(R) \to (pp, \tau)$ *: the simulated setup algorithm is randomized and takes as input a relation $R \in \mathcal{R}_\lambda$ and returns the public parameters pp and a trapdoor τ.*

$\mathsf{SSimSign}(pp, \tau, \phi, m) \to \sigma$ *: the simulated signing algorithm is randomized and takes as input some public parameters pp, a simulation trapdoor τ and an instance ϕ and returns a signature σ.*

Perfect correctness requires that the verifier will always be convinced by a signature of knowledge produced with an instance-witness pair in the relation.

Definition 4 (Perfect Correctness). *A signature of knowledge is perfectly correct if for all security parameters $\lambda \in \mathbb{N}$ and for all relations $R \in \mathcal{R}_\lambda$, for all valid instance-witness pairs satisfying the relation $(\phi, w) \in R$ and for all messages $m \in \mathcal{M}_\lambda$:*

$$\Pr[\mathsf{SVfy}(pp, \phi, m, \sigma) = 1 \mid pp \leftarrow \mathsf{SSetup}(R); \sigma \leftarrow \mathsf{SSign}(pp, \phi, w, m)] = 1$$

Perfect simulatability requires that a verifier cannot learn anything from a signature about the witness used to generate that signature. This can be captured by requiring that the signature can be simulated without a witness.

Definition 5 (Perfect Simulatability). *We say that a signature of knowledge is perfectly simulatable if for any PPT adversary $\mathcal{A}$, the advantage of the adversary $\mathcal{A}^{\mathsf{simul}}_{\mathsf{SOK},\mathcal{A}}(\lambda) = 2 \cdot \Pr[\mathbf{Exp}^{\mathsf{simul}}_{\mathsf{SOK},\mathcal{A}}(\lambda) = 1] - 1 = 0$, where $\mathbf{Exp}^{\mathsf{simul}}_{\mathsf{SOK},\mathcal{A}}$ is defined in fig. 2.*

For simulation extractability we require that an adversary cannot generate a new signature with respect to a statement ϕ without knowledge of a witness w for ϕ, and from any adversary that outputs a verifying signature we can extract a witness.

$\mathbf{Exp}^{\mathsf{sig-ext}}_{\mathsf{SOK},\mathcal{A},\mathcal{E}_\mathcal{A}}(\lambda)$	$\mathsf{SSimSign}_{pp,\tau}(\phi_i, m_i)$
$R \leftarrow \mathcal{R}_\lambda; \mathcal{Q} = \emptyset$	$\sigma_i \leftarrow \mathsf{SSimSign}(pp, \tau, \phi_i, m_i)$
$(pp, \tau) \leftarrow \mathsf{SSimSetup}(R)$	$\mathcal{Q} = \mathcal{Q} \cup \{(\phi_i, m_i, \sigma_i)\}$
$(\phi, m, \sigma) \leftarrow \mathcal{A}^{\mathsf{SSimSign}_{pp,\tau}}(pp)$	return σ_i
$w \leftarrow \mathcal{E}_\mathcal{A}(\text{trans}_\mathcal{A})$	
assert $(\phi, w) \notin R$	
assert $(\phi, m, \sigma) \notin \mathcal{Q}$	
return $\mathsf{SVfy}(pp, \phi, m, \sigma)$	

Fig. 3: $\mathbf{Exp}^{\mathsf{sig-ext}}_{\mathsf{SOK},\mathcal{A},\mathcal{E}_\mathcal{A}}$: Experiment for SOK simulation extractability.

Definition 6 (Simulation Extractability). *We say that a signature of knowledge is simulation-extractable if for any PPT adversary $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}_\mathcal{A}$ such that:* $\mathbf{Adv}^{\mathsf{sig-ext}}_{\mathsf{SOK},\mathcal{A},\mathcal{E}_\mathcal{A}}(\lambda) = \Pr[\mathbf{Exp}^{\mathsf{sig-ext}}_{\mathsf{SOK},\mathcal{A},\mathcal{E}_\mathcal{A}}(\lambda) = \mathbf{1}] \approx \mathbf{0}$*, where* $\mathbf{Exp}^{\mathsf{sig-ext}}_{\mathsf{SOK},\mathcal{A},\mathcal{E}_\mathcal{A}}$ *is defined in fig. 3.*

2.4 Transcripts and Views

The transcript of a protocol execution is defined to be the concatenation of all messages sent by any participant in the execution.

Definition 7 (Protocol Transcript). *The transcript of a protocol session between k participants is the sequence of messages exchanged by the participants during a run of the protocol Π. If Π is n-round then a transcript T is of the form $T = M_1^1||M_2^1||\cdots||M_{k_1}^1||\cdots||M_1^n||M_2^n||M_{k_n}^n$ where messages $M_1^i, \ldots, M_{k_i}^i$ are the k_i messages sent in round i.*

All messages are recorded by each party in the order in which they were received so a transcript is sorted by round, but different participants may have messages appear in different order in the transcript they recorded. As the transcript for a session will be the session identifier (see Section 3.1) there must be a notion of equivalence for transcripts containing the same messages but messages within each round appear in arbitrary order. Two transcripts match if they have the same set of messages in each round. This is formalized in Definition 8.

Definition 8 (Matching Transcripts). *Let T and $\hat{T}$ be two protocol transcripts and define $R_{i,T}$ (resp $R_{i,\hat{T}}$) to be the set of messages appearing in round i of T (resp $\hat{T}$). We say that T matches $\hat{T}$ (or $T \equiv \hat{T}^*$) if $R_{i,T} = R_{i,\hat{T}}$ for all i.*

Finally, we define the view of a participant to be the following:

Definition 9 (Participant View). *The view of participant P_i, written as view_{P_i}, is defined to be all inputs and outputs of that participant including the transcript, all oracle queries, oracle responses, and all random coins given to that participant.*

3 Defining WAKE

The goal of WAKE is that for any set of participants engaging in the key exchange protocol, authenticating with respect to (not necessarily distinct) statements, if each participant has knowledge of a witness to their associated statement then the participants terminate with a shared key, otherwise the participants terminate without a session key.

The model proposed is fundamentally similar to that of the group key exchange provided in [KY03] with minor modifications related to the unique setting of witness-authentication. The most impactful change is to the concept of identity. In authenticated group key exchange a participant's identity is associated to their

public key and is interpreted as *participant i is the holder of the secret key corresponding to public key* PK_i. Below the public key vector is replaced by a public statement vector $\Phi = \langle \phi_1, \ldots, \phi_\ell \rangle$ and each participant V_i claims to have knowledge of a witness w_i for the statement of the same index. This, in conjunction with the simulatability (zero knowledge) requirement, implies that all participants authenticating with respect to the same statement are indistinguishable. Crucially, any meaningful notion of personal identity is absent; witness-authentication remains agnostic to the true identity of the sender and instead asks: *were these messages generated with knowledge of a witness?*

3.1 Model

We fix a relation $\mathcal{R}$ and assume a polynomial-size set of potential participants $\mathcal{P} = \{V_1, \ldots, V_\ell\}$ for some polynomial $\ell = \ell(\lambda)$. Each participant U is associated with some public statement ϕ_U and has knowledge of a witness w_U. We assume that each witness w_i is sampled according to an arbitrary distribution $\mathcal{D}_{\phi_i}$ such that $(\phi_i, \mathsf{w}_i) \in \mathcal{R}$. The statement vector is $\Phi = \langle \phi_1, \ldots, \phi_\ell \rangle$ and the distribution over witnesses $\mathcal{D}_\Phi$ is such that a sample $\mathsf{w} \leftarrow \mathcal{D}_\Phi$ is a vector of witnesses $\mathsf{w} = \langle w_1, \ldots, w_\ell \rangle$ corresponding to the statements in Φ. The subscript notation is overloaded for ease; it is convenient to associate participants V_i, statements ϕ_i and witnesses w_i with the same index i when listing or assigning these values, but it is also convenient to index statements ϕ_U and witnesses w_U by their associated participant U when discussing a single instance.

Each participant U can participate in polynomially many protocol executions with an arbitrary subset of potential participants. This is modelled with single use instances denoted Π^i_U, meaning the ith instance of participant U. Each instance Π^i_U has the following associated variables, in addition to their statement and witness ϕ_U, w_U:

- state^i_U: the current internal state of the instance
- acc^i_U: a boolean denoting if the instance has accepted
- term^i_U: a boolean denoting if the instance has terminated
- sid^i_U: the concatenation of messages sent and received by the instance thus far
- sk^i_U: the session key

The adversary has control over all communication between the participants in every execution via the following oracles:

- $\mathsf{Send}(U, i, M)$ sends message M to instance Π^i_U and returns their reply
- $\mathsf{Execute}(U_{i_1}, j_1, U_{i_2}, j_2, \ldots, U_{i_n}, j_n)$: outputs a transcript of an execution between instances $\Pi^{j_1}_{U_{i_1}}, \Pi^{j_2}_{U_{i_2}}, \ldots, \Pi^{j_n}_{U_{i_n}}$
- $\mathsf{Reveal}(U, i)$: outputs the session key sk^i_U generated by Π^i_U

Prior to the first execution of the key exchange these public parameters are generated with a setup algorithm, $\mathsf{pp} \leftarrow \mathsf{SetUp}(1^\lambda, \mathcal{R})$. The SetUp algorithm takes as input the security parameter λ and the relation $\mathcal{R}$.[7] WAKE is also equipped with an additional $\mathsf{SimSetUp}$ algorithm, towards simulatability, which is discussed further below. The common inputs to all participants is (pp, Φ): the set of public parameters for the key exchange including the relation $\mathcal{R}$, and the public statement vector.

The first message in an execution initiated by some participant U involving participants $\{U, U_{i_2}, \ldots, U_{i_n}\}$ is realized by the adversary querying Send with input $(U, i, U||U_{i_2}|| \cdots ||U_{i_n})$. For brevity multiple sequential messages $M_1, \ldots, M_n$ can be sent to the instance Π^i_U with $\mathsf{Send}(U, i, M_1|| \cdots ||M_n)$. Instances Π^i_U and Π^j_V have a record of the same messages being sent throughout the interaction and thus can be said to have participated in the same interaction if $\mathsf{sid}^i_U \equiv \mathsf{sid}^j_V$ according to Definition 8.

Correctness requires that any instances participating in the same execution of Π with valid witnesses to their associated statements will terminate and accept with equal session keys.

Definition 10 (Correctness). *A* WAKE *protocol Π is correct if for all relations $\mathcal{R}$, for all sets of potential participants $\mathcal{P}$ of size $\ell = \ell(\lambda)$, for all participants $U, V \in \mathcal{P}$, for all instances $i, j \in \mathbb{N}$ such that $(\phi_U, \mathsf{w}_U), (\phi_V, \mathsf{w}_V) \in \mathcal{R}$, $\mathsf{sid}^i_U \equiv \mathsf{sid}^j_V$ and $\mathsf{acc}^i_U = \mathsf{acc}^j_V = \mathsf{TRUE}$ then $\mathsf{sk}^i_U = \mathsf{sk}^j_V$.*

[7] We notice this syntax can easily be extended to the case where the setup is universal [GKM+18].

3.2 Security

This subsection discusses the security requirements for WAKE. First we discuss which adversaries should be considered admissible in the confidentiality and authenticity games (see Definition 13). Then we introduce the adaptations of the notions of authenticity and confidentiality to the witness-authenticated setting (see Definitions 14 and 19). Then we discuss the distinction between passively and actively secure protocols (see Remark 2). Finally we introduce the definition of Simulatability (see Definition 16).

Admissible Adversaries: The goal of an adversary in the authenticity experiment (Figure 5) is to force a challenge instance to accept without knowledge of a witness: $\mathsf{acc}_U^i = \mathsf{TRUE}$. But, any adversary can convince any instance Π_U^i to accept by playing as a wire between that instance and the authenticated participants, forwarding messages and responses according to Π without injecting any of her own messages. In summary, a forwarding $\mathcal{A}$ does not *adversarially* convince Π_U^i to accept. The instance would accept because he is interacting with authenticated participants. A forwarding adversary (Definition 11) forwards all messages between the challenge instance and instances of participants authenticating with respect to the expected statements.[8] This behavior, referred to as ping-pong by [DF17] in the two party case, is generalized to groups and modified to witness-authentication in Definition 11.

Definition 11 (Forwarding Adversary). *Let $\mathcal{P} = \{V_1, \ldots, V_\ell\}$ be a set of potential parties, authenticating with respect to (not necessarily distinct) statements $\Phi = \langle \phi_1, \ldots, \phi_\ell \rangle$ in* WAKE *protocol Π. Let the challenge instance be (U, i), with associated session identifier $\mathsf{sid}_U^i = M_1 || \cdots || M_n$ containing first message M_1 the set of participants for the session. Then, $\mathcal{A}$ is forwarding for (U, i) if either there exists a query to the* Execute *oracle with input including the instance (U, i) or if for all $V' \in M_1$ with $V' \neq U$ there exists an instance $(V, j) \in \mathcal{P} \times \mathbb{N}$ such that all of the following conditions hold:*

1. *$\phi_V = \phi_{V'}$, and*
2. *for each query to the send oracle of the form $\mathsf{Send}(U, i, M) \to R$ outputting response $R \neq \mathsf{NULL}$ from instance Π_U^i there exists a corresponding query to the send oracle $\mathsf{Send}(V, j, R)$ forwarding the response to instance Π_V^j, unless the response was empty, and*
3. *for each query to the send oracle of the form $\mathsf{Send}(V, j, M') \to R'$ outputting a response $R' \neq \mathsf{NULL}$ from instance Π_V^j there exists a corresponding query to the send oracle $\mathsf{Send}(U, i, R')$ forwarding the response to instance Π_U^i, unless the response was empty.*

The set of participants $V' \in M_1$ such that the above outlined conditions do not hold are called the impersonated set, denoted $\mathcal{IS}(U, i)$.

As usual, the goal of an adversary in the confidentiality experiment (Definition 4) is to distinguish the challenge session key from a random key given access to (1) transcripts of valid executions via the Execute oracle, (2) the ability to reveal keys for eavesdropped transcripts via the Reveal oracle, and (3) access to long term secrets (witnesses) of the participants. Notably, any adversary with the long term secret of a participant can participate in the key exchange on behalf of that participant and then will then be able to trivially distinguish the computed session key. Additionally, an adversary that has revealed the session key for the challenge session[9] can also trivially distinguish the session key from random. This motivates an additional requirement, namely that the challenge participant must be fresh according to Definition 12.

The freshness requirement specifies that for challenge Π_U^i the adversary has neither revealed the session key for any instance participating in the execution with (U, i) nor has she injected any messages after learning a participant's witness.

Definition 12 (Freshness). *An instance Π_U^i is considered to be fresh if for all $V \in \mathcal{P}$, $\mathcal{A}$ has not queried $\mathsf{Reveal}(V, j)$ for any Π_V^j such that $\mathsf{sid}_V^j \equiv \mathsf{sid}_U^i$.*

An admissible adversary is one that does not trivially violate the properties of authenticity or confidentiality; the only adversaries considered are those that output a fresh challenge (U, i) with respect to which they are not forwarding.

[8] The expected statements are the statements associated to the parties appearing in message 1 of the session identifier. For $\mathsf{sid}_U^i = M_1 || \cdots || M_n$ with $M_1 = U_{i_1}, \ldots, U_{i_k}$ the challenge instance would expect to interact with participants authenticating with respect to $\{\phi_{i_1}, \ldots, \phi_{i_k}\}$.

[9] The challenge session is the session executed by the challenge instance.

Definition 13 (Admissible Adversary). *Consider participants $\mathcal{P} = \{V_1, \ldots, V_\ell\}$ executing a* WAKE *protocol Π. An adversary $\mathcal{A}$ is considered admissible if $\mathcal{A}$ outputs a fresh challenge (U, i) on which $\mathcal{A}$ is not forwarding.*

CONFIDENTIALITY: Confidentiality is the requirement that an eavesdropping adversary cannot distinguish the real session key from a random one. In the confidentiality experiment, seen in Figure 4, the adversary is an eavesdropper that cannot inject any messages. The adversary must then output a challenge instance (U, i). If the challenge instance has accepted and generated a key then the adversary receives as input either the real session key or a random key, Confidentiality stipulates that no admissible adversary can determine which key she has received, even given the secret vector of witnesses.

Definition 14 (WAKE Confidentiality). *The advantage of an adversary $\mathcal{A}$ with respect to the* WAKE *protocol Π in the confidentiality game seen in Figure 4 is defined as the following quantity:* $\mathbf{Adv}^{\mathsf{WAKE\text{-}confid}}_{\Pi,\mathcal{A}}(\lambda, \mathcal{R}, \Phi, \mathcal{D}_\Phi) = |2 \cdot \mathsf{Pr}[\mathbf{Exp}^{\mathsf{WAKE\text{-}confid}}_{\Pi,\mathcal{A}}(\lambda, \mathcal{R}, \Phi, \mathcal{D}_\Phi) = 1] - 1|$. *The* WAKE *protocol Π is confidential if, for all $\lambda \in \mathbb{N}$, for all relations $\mathcal{R}$, for all statement vectors Φ, for all distributions over witness sets $\mathcal{D}_\Phi$ and for all admissible non-uniform PPT $\mathcal{A}$, the advantage of $\mathcal{A}$ is negligible.*

$\mathbf{Exp}^{\mathsf{WAKE\text{-}confid}}_{\Pi,\mathcal{A}}(\lambda, \mathcal{R}, \Phi, \mathcal{D}_\Phi)$

$b \xleftarrow{\$} \{0,1\}$

$pp \leftarrow \mathsf{SetUp}(1^\lambda, \mathcal{R})$

$\mathsf{w} \leftarrow \mathcal{D}_\Phi$

$\mathcal{P} \leftarrow \{V_i[\phi_i, \mathsf{w}_i]\}_{i=1}^{|\mathsf{w}|}$

$(U, i) \leftarrow \mathcal{A}^{\mathsf{Execute}(),\mathsf{Reveal}()}(\Phi, \mathsf{w})$

if $\mathsf{acc}^i_U = \mathsf{FALSE}$: output b

$\mathsf{k}_1 \leftarrow \mathsf{sk}^i_U, \mathsf{k}_0 \xleftarrow{\$} \mathcal{K}$

$b' \leftarrow \mathcal{A}^{\mathsf{Execute}(),\mathsf{Reveal}()}(\mathsf{k}_b)$

if $b = b'$: output 1

else : output 0

Fig. 4: $\mathbf{Exp}^{\mathsf{WAKE\text{-}confid}}_{\Pi,\mathcal{A}}$: Experiment for WAKE confidentiality.

In the confidentiality experiment the adversary is granted access to Execute, but not Send. The adversary is also provided the complete vector of witnesses at the start of the experiment. This models the fact that an eavesdropping adversary with access to valid witnesses should not be able to distinguish the key from random. The Send oracle can be simulated by the adversary with knowledge of the witnesses.

Remark 1 (Forward Secrecy). We observe that our model of confidentiality guarantees forward secrecy, i.e. that a session key remains indistinguishable from random even if the long term secrets of the participants are compromised. This is modelled by providing $\mathcal{A}$ with the entire vector of witnesses w at the beginning of the confidentiality experiment. We also observe that this is a somewhat stronger notion of forward secrecy than the one modelled through a corruption oracle in [KY03].

AUTHENTICITY: Authenticity is the requirement that an unauthenticated participant cannot convince another participant to accept and consequently generate a session key; if an instance accepts in the authenticity experiment then either the adversary was forwarding on that instance or the adversary *knows some witness*. Knowledge of a witness is modeled by the existence of an extractor that can output a witness from the view of the adversary. In the authenticity experiment, seen in Figure 5, each participant U receives a witness and the

adversary facilitates communication between the authenticated participants via the Send and Reveal oracles. The adversary ultimately outputs a challenge: (U, i, V). For any admissible adversary outputting a challenge consisting of an accepting instance Π_U^i there exists an extractor which can output a witness to the statement ϕ_V. It is important that V must appear in the impersonation set of the challenge instance, meaning that the adversary was not merely forwarding messages between Π_U^i and any participant authenticating with respect to ϕ_V. This is formalized in Definition 15.

Definition 15 (WAKE Authenticity). *The advantage of an adversary $\mathcal{A}$ with respect to* WAKE *protocol Π in the authentication game seen in Figure 5 is defined as*

$$\mathbf{Adv}^{\mathsf{WAKE\text{-}auth}}_{\Pi,\mathcal{A},\mathcal{E}_\mathcal{A}}(\lambda, \mathcal{R}, \Phi, \mathcal{D}_\Phi) = \mathsf{Pr}[\mathbf{Exp}^{\mathsf{WAKE\text{-}auth}}_{\Pi,\mathcal{A},\mathcal{E}_\mathcal{A}}(\lambda, \mathcal{R}, \Phi, \mathcal{D}_\Phi) = 1]$$

A WAKE *protocol Π is witness-authenticated if for all admissible non-uniform PPT $\mathcal{A}$ there exists a PPT extractor $\mathcal{E}_\mathcal{A}$, for all $\lambda \in \mathbb{N}$, for all relations $\mathcal{R}$, for all statement vectors Φ, and for all witness distributions $\mathcal{D}_\Phi$, the advantage of $\mathcal{A}$ is negligible.*

$\mathbf{Exp}^{\mathsf{WAKE\text{-}auth}}_{\Pi,\mathcal{A}}(\lambda, \mathcal{R}, \Phi, \mathcal{D}_\Phi)$

$pp \leftarrow \mathsf{SetUp}(1^\lambda, \mathcal{R})$
$\mathsf{w} \leftarrow \mathcal{D}_\Phi$
$\mathcal{P} \leftarrow \{V_i[\phi_i, \mathsf{w}_i]\}_{i=1}^{|\mathsf{w}|}$
$(U, i, V) \leftarrow \mathcal{A}^{\mathsf{Send}(),\mathsf{Reveal}()}(\Phi)$
assert $(V \in \mathcal{IS}(U, i))$
$b_{\mathsf{acc}} \leftarrow \mathsf{acc}_U^i$
$\mathsf{w}' \leftarrow \mathcal{E}_\mathcal{A}(\mathsf{view}_\mathcal{A})$
$b_{\mathsf{ext}} \leftarrow (\phi_V, \mathsf{w}') \in \mathcal{R}$
output $(b_{\mathsf{acc}} \wedge \bar{b}_{\mathsf{ext}})$

Fig. 5: $\mathbf{Exp}^{\mathsf{WAKE\text{-}auth}}_{\Pi,\mathcal{A},\mathcal{E}_\mathcal{A}}$: Experiment for WAKE authenticity.

In the authentication experiment $\mathcal{A}$ is not equipped with an Execute oracle; the Execute oracle is redundant as it can be simulated with the Send oracle.

Active & Passive Security: The distinction between passively and actively secure group key agreement, as seen in [KY03], is centered around the Send oracle; a protocol is considered actively secure if it is secure against an adversary that has access to Send, otherwise the protocol is considered passively secure. This distinction is not applicable to WAKE.

Consider a variant of the WAKE confidentiality experiment in which the adversary is also granted access to the Send oracle. The inclusion of Send does not change the experiment; $\mathcal{A}$ has access to w, the vector of witnesses, and can therefore simulate Send even if she is not granted access to it explicitly. Additionally, any queries to the Send oracle will not affect the challenge session as the adversary is necessarily eavesdropping on the challenge. Alternatively, consider a variant of the WAKE authenticity experiment in which the adversary is not allowed any Send queries but instead can query Execute. The removal of Send renders every adversary inadmissible; an adversary exclusively using the Execute oracle has not injected any of her own messages and is forwarding. Therefore, in the absence of a distinction between passive and active variants of the confidentiality and authenticity experiments, we define passive and active security for WAKE as in Remark 2. Passively secure protocols are those that satisfy confidentiality while actively secure protocols additionally satisfy authenticity and simulatability.

Remark 2 (Active and Passive Security for WAKE*).* We say a WAKE protocol achieves passive security if it satisfies confidentiality (Definition 14); it achieves active security if it satisfies confidentiality, authenticity (Definition 15).

Passive security for WAKE (as defined here) is equivalent to passive security for group key exchange (as defined in [KY03]). This can be seen by comparing the relevant experiments; a passively secure group key exchange generates session keys that are indistinguishable from random by an adversary that controls all communication in the network and has access to Execute, Reveal, Corrupt and outputs a fresh challenge. This is equivalent to the WAKE-confid experiment, except instead of providing a Corrupt oracle the adversary is given all of the secrets at the start of the experiment.

In fact, a passively secure WAKE does not have to require that the participants make use of their witnesses at all. Therefore, the following simulatability requirement is only meaningful for actively-secure WAKE.

SIMULATABILITY: Given a correct and confidential group key exchange protocol Π between participants $\mathcal{P} = \{P_1, \ldots, P_\ell\}$, one can obtain witness-authentication for associated statements Φ and witnesses $\mathcal{W} = \{\mathsf{w}_{V_1}, \ldots, \mathsf{w}_{V_\ell}\}$ by following Π with the following modifications: (1) when participant U should send a message M according to Π he concatenates his witness to the message, instead sending $M||\mathsf{w}_U$, (2) when participant U receives a message $M = M'||\mathsf{w}_{U'}$ from participant U' he verifies that $(\phi_{U'}, \mathsf{w}_{U'}) \in \mathcal{R}$ and aborts if this verification fails. Such a solution does not align with our intuition. This, along with the malleability attack seen in Section 1, motivates a third requirement: simulatability. Simulatability is the requirement that the adversary cannot learn anything about the witness from the messages sent by that participant. This is implied by the existence of a simulator which, without access to the witness, can generate messages indistinguishable from those of a real participant.

Simulatability requires the existence of a second setup algorithm, SimSetUp, which outputs public parameters indistinguishable from those output by the usual setup algorithm along with a trapdoor τ. With the trapdoor any participant can be simulated without access to their associated witness. In the simulatability experiment the adversary is given access to a SetKey oracle which takes as input (ϕ, w) and generates a new participant associated with the statement-witness pair. The adversary then interacts with either the real participants or the simulated versions using Send_b^*. The real participants behave honestly and interact using the witness provided to them by some query to SetKeys, whereas the simulated version only has access to the public statements and the trapdoor. Any oracle query must have a corresponding SetKeys query for each participant appearing in the input. The adversary outputs her guess b', indicating if she believes she is interacting with real participants or simulators.

Definition 16 (WAKE Simulatability). *A* WAKE *protocol Π is simulatable if there exist efficient algorithms* (SimSetUp, Sim) *(the latter stateful) such that for all $\lambda \in \mathbb{N}$, relations $\mathcal{R}$ and for all non-uniform PPT $\mathcal{A}$, the advantage of $\mathcal{A}$ in the simulatability game seen in Figure 6 is negligible, where the latter is defined as* $\mathbf{Adv}^{\mathsf{WAKE\text{-}sim}}_{\Pi,\mathcal{A}}(\lambda, \mathcal{R}) = 2 \cdot \mathsf{Pr}[\mathbf{Exp}^{\mathsf{WAKE\text{-}sim}}_{\Pi,\mathcal{A}}(\lambda, \mathcal{R}) = 1] - 1$.

4 Unilateral WAKE

Unilateral Witness-Authenticated Key Exchange (U-WAKE) is a specialization of the above group WAKE to the two party, unilaterally authenticated, case. U-WAKE can also be seen as an adaptation of Unilaterally Authenticated Key Exchange (UAKE) [DF17], in which an unauthenticated participant establishes a key with an authenticated participant, to the witness-authenticated case. In U-WAKE, authentication is done with respect to a witness for some statement associated to the authenticated participant, called the Responder (Res). All adaptations of the WAKE model (appearing in Section 3) to the unilaterally authenticated two participant setting are listed:

- The set of potential participants is denoted $\mathcal{P} = \{\mathsf{Init}, \mathsf{Res}_1, \ldots, \mathsf{Res}_{\ell(\lambda)-1}\}$[10]
- ϕ_{Init} is a dummy statement (one that is in the language and is easy to decide)

[10] Notice this is just a change of notation to make explicit which parties are authenticated or not, namely the responders and the initiatior.

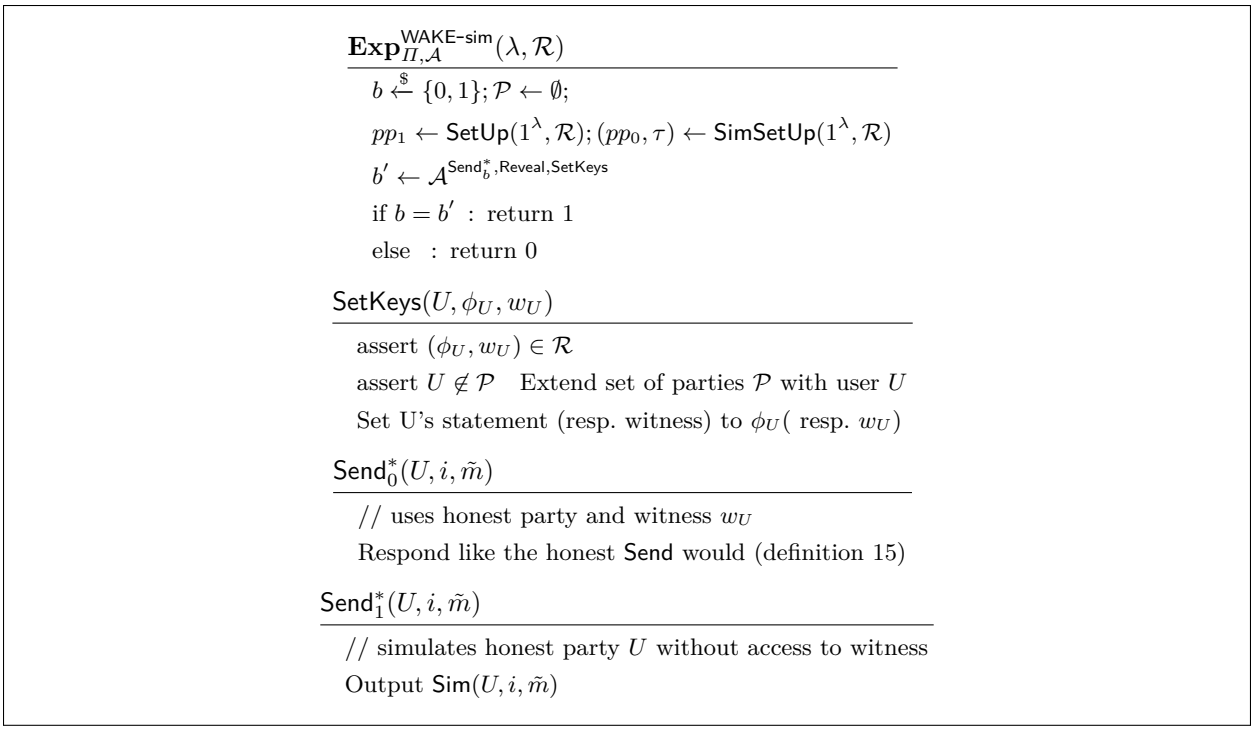

Fig. 6: $\mathbf{Exp}^{\mathsf{WAKE\text{-}sim}}_{\Pi,\mathcal{A}}$: Experiment for WAKE simulatability. Simulator Sim is stateful, has access to statements ϕ_U and other parameters (e.g., trapdoor τ), but not to the witnesses w_U.

– Challenge instances: the authenticity challenge must be of the form $(\mathsf{Init}, i, \mathsf{Res}_j)$ and the confidentiality challenge must be of the form (Init, i)

Importantly, as Init is unauthenticated the associated statement ϕ_{Init} is such that a witness W can be computed in polynomial time.[11] Additionally, in a two party interaction the challenge must be an instance of the Init. Therefore, any admissible adversary for U-WAKE is an admissible adversary for WAKE that outputs such a challenge. The curious reader is referred to Appendix A for explicit formulations of the U-WAKE Experiments.

4.1 Construction from KEM and SOK

Given a signature of knowledge and a key encapsulation mechanism one can construct U-WAKE. As a concrete example to keep in mind throughout this section, we recommend considering Diffie-Hellman (DHKE): Init first samples randomness x and sends as their first message the associated public key $h_I = g^x$ and Res does the same, computing $h_R = g^y$, along with a signature of knowledge σ on $m = h_I||h_R$. The response is then h_R, σ. Contingent upon signature validation, the Initiator then computes $\mathsf{sk} = (h_R)^x$ and the Responder computes $\mathsf{sk} = (h_I)^y$ as the session key.

For U-WAKE, the public parameters output by SetUp are $pp_{\mathsf{U\text{-}WAKE}} = \{\lambda, \mathcal{R}, pp_{\mathsf{SOK}}\}$, the security parameter, the relation and the public parameters for the signature of knowledge. The public parameters output by SimSetUp also include the trapdoor for the signature τ_{SOK}. In Figure 7 we present the construction of U-WAKE from KEM and SOK.

Theorem 1 (Π is secure.). *Let* KEM *be a correct and* CPA*-secure key encapsulation mechanism. Let* SOK *be a perfectly correct and simulatable, simulation-extractable signature of knowledge. Then, the protocol Π, as seen in Figure 7, is an actively secure and simulatable* U-WAKE*.*

[11] This modification can allow the group-WAKE to generalize to a setting where an arbitrary subset of the potential parties are unauthenticated in such a way.

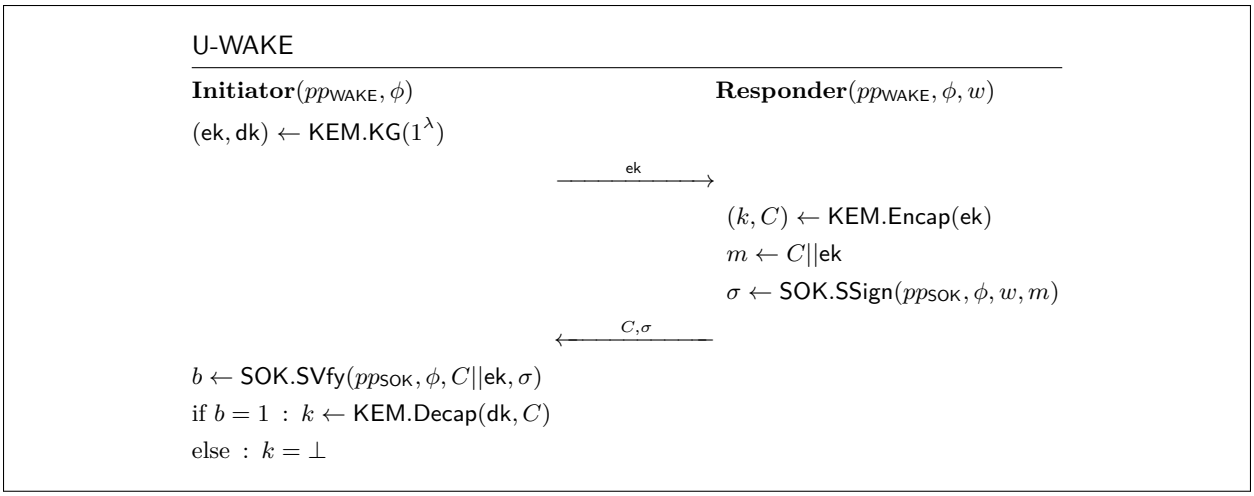

Fig. 7: U-WAKE from KEM and SOK

A full proof of this theorem appears in Appendix B.

5 A Compiler From Passive to Active Security

In this section we describe a compiler that transforms any passively secure key-exchange into a witness-authenticated, actively secure protocol. Our compiler revisits the one presented in [KY03], adapting it to the witness-authenticated setting.

5.1 The Compiler Construction

The compiler takes as input a passively secure protocol Π and outputs an actively secure, simulatable, protocol Π^*. This transformation is at the expense of an additional round in which each participant samples and distributes a random nonce. Following this round, each participant proceeds as they would in Π with a few additional steps. To each message m they should send according to Π, they add a signature of knowledge on that message concatenated with the nonces from Round 0 and upon receipt of any message the participant must first verify the signature. The explicit construction is provided in fig. 8.

5.2 Security

Theorem 2. *If Σ is a passively-secure protocol (remark 2) then Σ^* (fig. 8) is a fully secure and simulatable witness-authenticated key exchange protocol (section 3.2 and remark 2) .*

Lemma 1. *The protocol in fig. 8 satisfies confidentiality (definition 14).*

Proof. We reduce to the confidentiality of the underlying passive scheme as follows. For a confidentiality adversary $\mathcal{A}_{\text{act}}$ against compiled protocol Σ^*, we construct a confidentiality adversary $\mathcal{A}_{\text{pas}}$ against the original passively secure protocol. See fig. 9.

We claim that the advantage of $\mathcal{A}_{\text{pas}}$ in the confidentiality game for protocol Σ is negligibly close to that of $\mathcal{A}_{\text{act}}$ in the confidentiality game for protocol Σ^*. We now define three hybrids:

- $\mathcal{H}_{\text{act}}$: this is the advantage of $\mathcal{A}_{\text{act}}$ in the confidentiality game for protocol Σ^*
- $\mathcal{H}_{\text{act-zk}}$. this is the advantage of $\mathcal{A}_{\text{act}}$ in a modified confidentiality game for protocol Σ^*, where the challenger acts as in $\mathcal{H}_{\text{act}}$, *except* that it uses a simulator for signatures of knowledge. We claim that $\mathcal{H}_{\text{act}} \approx \mathcal{H}_{\text{act-zk}}$: this follows from simulatability of signatures of knowledge. If the two advantages were not negligibly close than we can easily build a distinguisher for real/simulated signatures of knowledge.

– **Setup:** We run the setup for Σ and the setup for the signature of knowledge scheme.

– **Round 0:** each user U_i samples a random nonce $r_i \leftarrow\!\!\$\ \{0,1\}^\lambda$ and sends message $(U_i||0||r_i)$ to all the other parties. After receiving the related message from all the other parties, each instance Π_U^j sets and stores $\mathsf{nonces}_U^j := U_1||r_1||\ldots||U_m||r_m$ and $\mathsf{S}_U^j := \{U_1,\ldots,U_m\}$ (the set of participants) as local information.

– **Following rounds:**

- Whenever instance Π_U^i should **send** $(U||j||m)$ by Σ to all other parties:
  1. it produces $m^* = \left(U||j||m||\mathsf{nonces}_U^j\right)$
  2. it signs it as $\sigma \leftarrow \mathsf{SOK.SSign}(\mathsf{pp}, \phi_U, w_U, m^*)$
  3. it sends $(m^*||\sigma)$ to all other parties
- Whenever instance Π_U^i **receives** $(V||j||m||\mathsf{nonces}||\sigma)$:
  1. it checks $V \in \mathsf{S}_U^i \setminus \{U\}$; it aborts otherwise
  2. it checks signature σ on $V||j||m||\mathsf{nonces}$ against the statement of party V and that $\mathsf{nonces} = \mathsf{nonces}_U^i$[12]; it aborts otherwise
  3. it checks the sequence number j is the expected one; it aborts otherwise
  4. it continues as in protocol Σ

Above $\mathcal{U} = \{U_1,\ldots,U_m\}$ denotes the set of parties willing to establish a common key.

Fig. 8: Compiler from protocol Σ with passive security to one with active security (Σ^*).

$\mathcal{A}_{\text{pas}}(\mathsf{pp}_{\text{pas}})$

$(\mathsf{pp}_{\text{sok}}, \tau) \leftarrow \mathsf{SSimSetup}(1^\lambda, R)$

Run $\mathcal{A}_{\text{act}}(\mathsf{pp}_{\text{sok}}||\mathsf{pp}_{\text{pas}})$ and emulate each oracle query as follows

$\mathsf{Execute}(\cdot)$:

- Invoke $\mathsf{Execute}$ for Σ obtaining transcript(s) $\mathcal{T}$
- Extend $\mathcal{T}$ emulating rest of the protocol Σ^* as in fig. 8 :
  - sample nonces as appropriately
  - Compute signatures invoking SoK simulator
- Return $\mathcal{T}$ "compiled" with nonces and simulated signatures

$\mathsf{Reveal}(\cdot)$:

- Respond using its own Reveal oracle

Let (U, i) be the output of $\mathcal{A}_{\text{act}}$ at the end of interaction

Send (U, i) to the challenger receiving back a challenge key k

Run $\mathcal{A}_{\text{act}}(\mathsf{k})$(emulating oracles as before) till it outputs bit b

Output b

Fig. 9: Reduction for confidentiality in compiled construction.

– $\mathcal{H}_{\text{pas}}$: this is the advantage of $\mathcal{A}_{\text{pas}}$ in the confidentiality game for protocol Σ. We claim that $\mathcal{H}_{\text{act-zk}} \equiv \mathcal{H}_{\text{pas}}$: in fact they are exactly the same distribution except with different syntaxes (the former produces simulated signatures in the challenge while the latter in the adversary $\mathcal{A}_{\text{pas}}$).

This concludes the proof.

Lemma 2. *The protocol in fig. 8 satisfies authentication (definition 15)*

Proof. In order to argue authentication security, given an adversary $\mathcal{A}^*$, we construct an extractor $\mathcal{E}^*$[13] such that: either $\mathcal{E}^*$ can extract a witness with reasonable probability (this probability should in particular be a noticeable fraction of the advantage of $\mathcal{A}^*$), or $\mathcal{A}^*$'s advantage was too low to be of interest to begin with. To build this extractor $\mathcal{E}^*$ we rely on the simulation-soundness property of signatures of knowledge. The latter states that, given an adversary forging a valid signature (with access to a simulator oracle), there is an extractor $\mathcal{E}_{\text{SoK}}$ for it that is able to produce a valid witness. We then first define adversary $\mathcal{A}_{\text{SoK}}$ for the simulation-security experiment: it internally runs $\mathcal{A}^*$ and responds to each of its Send queries by emulating the behavior of the honest parties. Adversary $\mathcal{A}_{\text{SoK}}$ may not know the witnesses of the honest parties, but can still emulate each of their signatures by using its simulation oracle. At the end of this interaction $\mathcal{A}^*$ will declare an instance challenge (U, i) where it "impersonated" some of the honest parties and that resulted in an acceptance. Adversary $\mathcal{A}_{\text{SoK}}$ will then retrieve a forged message $(m^*||\sigma)$ (any message with sequence number greater or equal than 1 will have this structure) in the transcript in Π_U^i. It will then return the challenge triple (x_V, σ, m^*) where party V is the party having message m^* claims as a sender. We can claim that either the returned triple is a valid challenge for the simulation-extractability game (with non-negligible probability), or $\mathcal{A}^*$ is not admissible.

Recall that, for $\mathcal{A}^*$ to be admissible, it must be "forging" one of the messages in the instance Π_U^i. If it forges a message[14] at round 0, then it's using a nonce that is not the output of any of the honest parties. Say it's nonce r_j. The adversary would not be able to query that party U_j using that nonce since with overwhelming probability it would be rejected if appended to any message (since U_j did not produce it). Therefore, $\mathcal{A}^*$ would be forced to forge the following messages too to stay admissible and thus we can reduce to the following case directly.

If adversary $\mathcal{A}^*$ is forging a message at a round $j \geq 1$ on behalf of some party V, then it must also be producing a signature for it (recall that all messages at this stage are of the form $m^*||\sigma$ where $m^* = (V||j||m||\mathsf{nonces}_U^j)$). By construction this will be a valid challenge for the simulation-extraction game of SoK (in the event that the interaction produced by $\mathcal{A}^*$ is admissible). Reducing to the security of signatures of knowledge concludes the proof.

We describe the adversary $\mathcal{A}_{\text{SoK}}$ and extractor $\mathcal{E}^*$ in fig. 10. In the description of $\mathcal{E}^*$, we denote by $\mathcal{E}_{\text{SoK}}$ the extractor for $\mathcal{A}_{\text{SoK}}$ from the simulation-extractability of signatures of knowledge.

Lemma 3. *The protocol in fig. 8 satisfies simulatability (definition 16)*

Proof. We simulate relying on the simulation property of signatures of knowledge. We define algorithm $\mathsf{SimSetUp}$ as the algorithm that runs $\mathsf{SSimSetUp}$ and returns its output (see definition 5). We define the simulator in fig. 11. Given $\mathcal{A}^*$ for the simulatability game of the key agreement, we build an adversary $\mathcal{A}$ (fig. 12) for the zero-knowledge property of signatures of knowledge. This adversary internally uses $\mathcal{A}^*$ and has access to an oracle $\mathcal{O}$ that can be either a simulator or the honest signing algorithm. The advantage of $\mathcal{A}$ is the same as that of $\mathcal{A}^*$. Observing that the former must be negligible by simulatability of the SoK concludes the proof.

6 Achieving Three-Round WAKE for Groups

Here we show how we can instantiate a passive secure scheme (see remark 2) in our compiler in section 5 to obtain a three-round WAKE for groups (we discuss instantiations for signatures of knowledge in section 8) Presented in Figure 13 is Π_{GKE}, a passively secure protocol for group key exchange. This protocol was constructed by Burmester and Desmedt [BD95], and then was adapted and proven secure under the Decisional Diffie-Hellman assumption by Katz and Yung [KY03]. Running the compiler seen in Figure 8 on Π_{gke} yields a three round, actively secure and simulatable WAKE protocol. This is formally stated in Theorem 1.

First we review the Decisional Diffie-Hellman Assumption. For $\mathbb{G}$ a cyclic group of order $q \in \mathbb{P}$ with generator g, the Decisional Diffie-Hellman (DDH) Problem is to distinguish between Diffie-Hellman tuples (g^x, g^y, g^{xy}) and random tuples of the form (g^x, g^y, g^z) for $x, y \in \mathbb{Z}_q^*$, $z \in \mathbb{Z}_q^* \setminus \{xy\}$. Consider an infinite

[13] We can show that we can build such an extractor for each of the parties in the impersonation set.

[14] In this discussion we consider only messages that look valid to the receiver, since otherwise the latter would abort.

$\mathcal{A}^{\mathcal{S}(\cdot)}_{\mathrm{SoK}}(\mathsf{pp}_{\mathrm{SoK}})$

Run $\mathcal{A}^*$ and emulate each oracle query as follows

$\mathsf{Send}(\cdot)$:

– honestly run Send for all steps in fig. 8 *except* signatures

– Compute signatures invoking simulator oracle $\mathcal{S}(\cdot)$

$\mathsf{Reveal}(\cdot)$:

– Respond honestly using the internal state from the Send queries

Let (U, i, V) be the output of $\mathcal{A}^*$ at the end of interaction

For transcript of instance Π^i_U:

– Find message $(m^*||\sigma)$ where m^* is of the form $(V||\dots)$ in $\mathcal{IS}(U, i)$

(this corresponds to a message claimed by party V but forged by $\mathcal{A}^*$)

– If no such message exists, abort

return (ϕ_V, m^*, σ)

$\mathcal{E}^*(\mathsf{view}^*)$

Compute $\mathsf{view}_{\mathrm{SoK}}$ from view^*.This includes:

– the randomness used by $\mathcal{A}_{\mathrm{SoK}}$

– the query responses from Sim

$w \leftarrow \mathcal{E}_{\mathrm{SoK}}(\mathsf{view}_{\mathrm{SoK}})$

return w

Fig. 10: Adversary for the simulation-extractability of signature of knowledge (fig. 3) and extractor for the authenticy game (definition 15). The impersonating set $\mathcal{IS}$ is defined in definition 11.

$\mathsf{Sim}(U, i, \tilde{m})$

Respond like the honest Send (definition 15) would for construction in fig. 8

except that signatures are produced as follows:

$\sigma \leftarrow \mathsf{SSimSign}(\tau_{\mathrm{SoK}}, \phi_U, m^*)$

where U is the party claiming to send message m^*

(see also bullet 2, case **send**, in fig. 8)

Fig. 11: Simulator for construction in fig. 8

sequence of groups $\mathcal{G} = \{\mathbb{G}_\lambda\}_{\lambda\geq 1}$ indexed by the security parameter λ and define the advantage of an adversary $\mathcal{A}$ against DDH in $\mathbb{G}_\lambda$ as follows:

$$\mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_\lambda,\mathcal{A}}(\lambda) := |\mathsf{Pr}[\mathcal{A}(g^x, g^y, g^{xy}) = 1|x, y \leftarrow \mathbb{Z}^*_q] - \mathsf{Pr}[\mathcal{A}(g^x, g^y, g^z) = 1|x, y, z \leftarrow \mathbb{Z}^*_q \setminus \{xy\}]|$$

The DDH assumption states that for all PPT $\mathcal{A}$, the advantage $\mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_\lambda,\mathcal{A}}(\lambda)$ is negligible. The variant of DDH described above excludes the possibility that $z = xy$ for simplicity.

The participant set is denoted $\mathcal{P} = \{U_i\}^n_{i=1}$ with participants indexed mod n, such that $U_n = U_0$ and $U_{n+1} = U_1$. The inputs $\mathbb{G}, g$ are generated beforehand but can also be generated by a single player at the expense of an additional round. The communication style is referred to as broadcasting but it is important to note that a broadcast channel is not assumed in the construction; participants send all messages via point-to-point links which is referred to as broadcasting.

The session key generated by the protocol in Fig. 13 is $\mathsf{sk} = g^{x_1x_2+x_2x_3+\cdots+x_nx_1}$ and is common to all participants. Π_{GKE} achieves passive security for group key exchanges; the protocol is secure against an eavesdropping adversary in a RoR experiment. Passive security, along with forward security, is proven in [KY03].

$\mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pp}_{\mathrm{SoK}})$

Run $\mathcal{A}^*$ emulating its oracles (and keeping appropriate state):
- all oracles but Send are run as for the honest case
- a query to Send is run as the honest Send for construction in fig. 8 except that signatures are produced invoking oracle $\mathcal{O}$

Return the same bit as $\mathcal{A}^*$ at the end of interaction

Fig. 12: Adversary for reduction to zero-knowledge of SoK

Π_{GKE}: group key exchange between participant set $\mathcal{P} = \{U_1, \ldots, U_n\}$

Input: $\mathbb{G}, g$

Round 1: Each U_i samples $x_i \xleftarrow{\$} \mathbb{Z}_q$ and broadcasts $z_i = g^{x_i}$

Round 2: Each U_i broadcasts $X_i = (z_{i+1}/z_{i-1})^{x_i}$

Key Computation: Each U_i computes session key

$$sk_i = (z_{i-1})^{nx_i} \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdots X_{i+n-2}$$

Fig. 13: A passively secure group key exchange protocol [KY03]

Theorem 3 (Passive Security of Π_{GKE}). *The group key exchange protocol Π_{GKE} seen in Figure 13 is passively secure, as defined in Remark 2, under the* DDH *assumption.*

For concreteness, let Σ_{SOK} be the signature of knowledge detailed in [GM17]. We apply the compiler detailed in Section 5 to Π_{GKE}, using Σ_{SOK} as the signature of knowledge, to get a three-round actively secure WAKE protocol.

Corollary 1 (Three Round WAKE). *Π_{WAKE}, the protocol resulting from applying the compiler (Figure 8) on Π_{GKE} (Figure 13) and Σ_{SOK}, yields a three round actively secure* WAKE.

7 Offline/Online Computation

The most expensive part of our protocol is the computation of the Signature of Knowledge, which is implemented using a Non-Interactive Proof of Knowledge of the witness. If implemented with a SNARK, this proof can be constructed with small bandwidth and verification time. It is well known that the bottleneck cost is the time it takes for the Prover to compute such proof, something which is confirmed by our evaluation experiments described in Section 8. We note that we can modify the protocol so that the cost of computing the SNARK can be moved to an *offline* phase, before the participant is contacted for a WAKE.

The intuition is as follows: during the offline phase the witness holder (Responder) generates $(\mathsf{sk}, \mathsf{vk})$ a key pair for a signature scheme where sk is the secret signing key and vk is the public verification key. Then the Prover uses an SOK SNARK to sign vk. Let σ_1 the resulting signature. The Prover stores $(\mathsf{sk}, \mathsf{vk}), \sigma_1$. During the online phase when the Responder receives ek it will sign $m = C||\mathsf{ek}$ it using sk, let σ_2 the resulting signature. The Responder then sends back $\mathsf{vk}, \sigma_1, \sigma_2$. The Initiator checks that σ_1 is a correct SOK for vk, and that σ_2 is a correct signature of ek under vk.

The modified protocol can be seen in Figure 14, where DS is an EUF-CMA digital signature scheme defined by the three algorithms: key generation $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$, signature $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$ and verification $\{0, 1\} \leftarrow \mathsf{Vfy}(\mathsf{vk}, m, \sigma)$.

Intuitively, the proof of security follows from the security of both the SOK and the regular signature scheme, which incidentally can be a one-time signature since each verification key is used to sign only one message. The main technical issue is in the proof of extractability for the witness, since the Prover is guaranteed to know the witness when it computed σ_1 and not when it successfully completes the protocol.

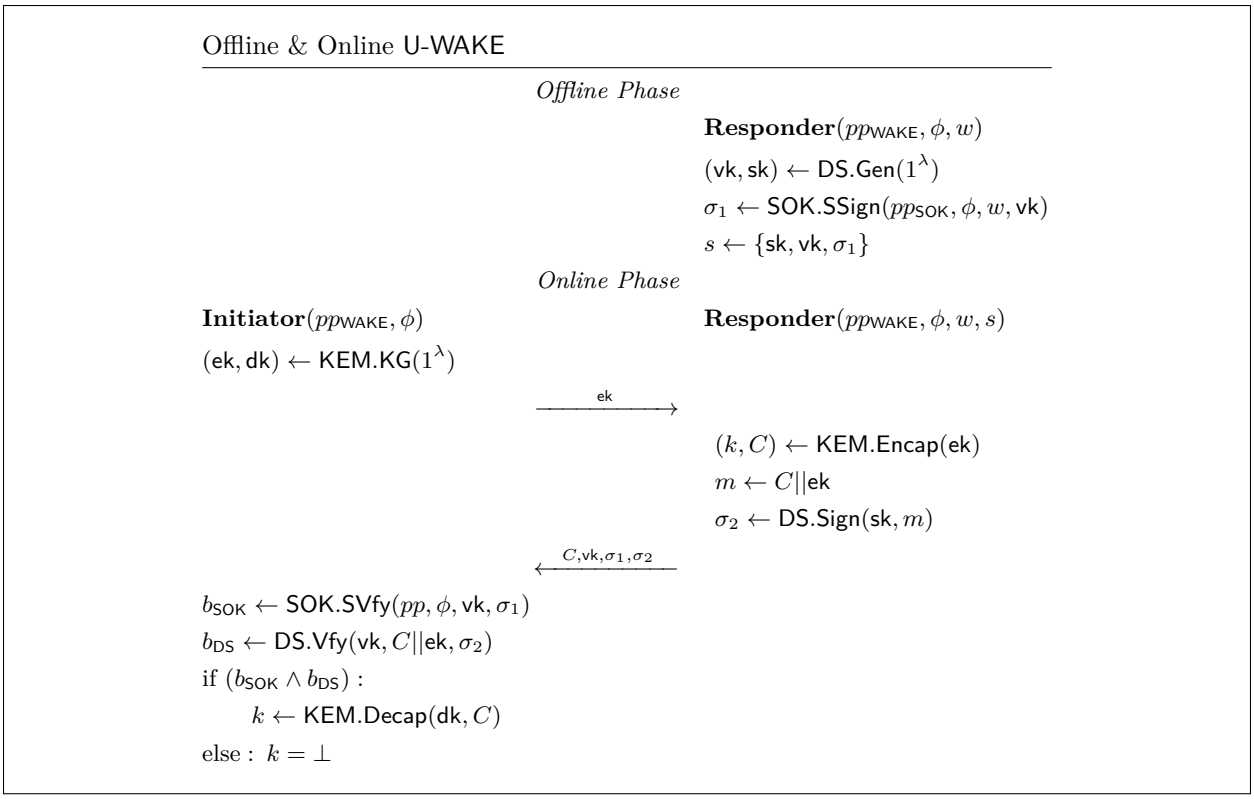

Fig. 14: Offline & Online phases for U-WAKE

In some applications this may be an issue (e.g. when the WAKE should guarantee that the Responder still owns a particular file). One way to address this issue in practice is to add some form of timestamp to the message signed in the offline case, which guarantees at least that the Responder *knew* the witness relatively recently. A full proof will appear in the final version.

While the above discussion, and Figure 14, refer only to the U-WAKE setting this modification can be generalized and applied to the group-WAKE setting by having all authenticated participants execute the offline phase.

8 Experimental Evaluation

In this section we discuss the concrete efficiency of our constructions in some of the practical settings mentioned in the Introduction. While we explicitly focus on the two-party case where only one of the two parties is authenticated, the concrete complexity roughly extends to the group-authenticated case. The dominating costs in our constructions is that of signatures of knowledge. As discussed in Section 7, this cost can be pushed to an offline stage performed by the authenticated parties. We consider the following settings (see table 1):

– *Dark pools*: we consider the case of a committed value (e.g., a coin) and authentication through a proof that the value is above a certain range. We use ranges of 32 bits in our benchmarks and SHA256 as a commitment method.
– *Zero-Knowledge Contingent Payment (ZKCP)*: this corresponds to the scenario where a seller wants to start a channel from a party claiming to have a digital good satisfying a certain property. A witness-authenticated private channel can be used for example to negotiate a price before engaging in a ZKCP protocol. We consider two settings for ZKCP: the "Sudoku benchmark" used in previous works on ZKCP and the bug bounty setting (we benchmark a sudoku of standard size $N = 10$, but in general the circuit

Setting	Relation	Authenticated party (offline running time)
Bidding/Dark Pools [NMKW21]	$\underline{c} = \mathsf{Comm}(\boldsymbol{s}, \boldsymbol{\rho}) \wedge \boldsymbol{s} \geq \underline{B}$	3–4s
ZKCP [Max, CGGN17]	$\mathsf{solvesSudoku}(\boldsymbol{s}, \underline{\mathrm{pzl}})$	0.5–1s
Bug bounty	$C_{\mathrm{buggy}}(\boldsymbol{w}) = 1 \wedge \underline{C_{\mathrm{expect}}}(\boldsymbol{w}) = 0$	55–58s
IPFS [Prob]	$\underline{h} = \mathsf{blake3hash}(\boldsymbol{F})$	65–68s

Table 1: Running times for different settings/relations for construction in fig. 7 using Diffie-Hellman Key Exchanges as underlying KEM. Underlined identifiers denote public inputs; boldface denotes a witness held by the authenticated party.

complexity of checking a Sudoku solution roughly grows with N^3). In the latter, a software producer of program C_{buggy} is incentivizing users to find potential bugs in the program. Here we model this by introducing an additional input, a (small) program C_{expect} checking some necessary expected condition that is violated by the bug. As an example consider a prime-testing program C_{buggy}. here the bug could consist of an even number greater than 2 that the program erroneously recognizes as a prime. In this case we could have for example $C_{\mathrm{expect}}(z) :=$ "z is odd $\vee\ z = 2$". In our benchmarks we use $|C_{\mathrm{expect}}| \approx 500K$ wires and $|C_{\mathrm{buggy}}| \approx 10K$ wires .

– *Retrieval Market*: we model a settings similar to that of IPFS [Prob], in which files are identified through a content ID (CID) which roughly corresponds to a hash of the file. The typical block size files are broken up into is 256KB, which is what we use in our benchmarks[15].

Concrete costs All the costs discussed here refer to the instantiations and experimental setting described at the end of this section. The offline running time of the authenticated party is summarized in table 1.

Communication complexity is constant. We estimate it to be below 0.5 KB in total for the unilateral two-party case and of approximately N KB for the group authentication of N parties.

The online running time is also always constant and is of the order of tens of milliseconds. In the unauthenticated setting it is even lower for the unautheunticated party. Naturally, if we do not use an offline/online approach the total running time of each party is the sum of the offline and online running times.

Details on Instantiations and Experimental Setting For signatures of knowledge we consider the construction from [GM17] based on a simulation-extractable variant of [Gro16]. A signature of knowledge consists of three group elements (two elements in $\mathbb{G}_1$ and one in $\mathbb{G}_2$), plus a hash. Using BLS12-381 [bls] as a concrete curve a signature of knowledge consists of 224 bytes (192 bytes for the group elements, plus 32 bytes for SHA256). For the online stage (see section 7), using BLS signatures [BLS04] as DS in section 7 would give us public keys of 48 bytes and signatures of 96 bytes (again using curve BLS12-381).

We run all our experiments on Amazon EC2 c5ad.16xlarge with 128 GiB of RAM running 3.3GHz AMD EPYC 7002 series CPUs. We ran our experiments using a single thread. For our estimates, we rely on the implementation of [GM17] in libsnark[16] using the curve implemented in the libsnark library, BN254, (this

[15] Our benchmarks differ from the current implementation of IPFS in the hash function: we use Blake3 [AONZ] instead of SHA256. Producing a (very succinct) signature of knowledge for a SHA computation of that size is significantly more expensive: while hashing with Blake3 requires approximately 2^{19} constraints, SHA256 would require approximately 2^{27}. It is plausible IPFS will support proof-friendly hash functions such as Blake3 in the future.

[16] `https://github.com/scipr-lab/libsnark`

has comparable running times[17] to BLS12-381 which provides 128 bit of security, which provides 110 bits of security of BN254).

References

AIR01. William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 119–135. Springer, Heidelberg, May 2001.

AONZ. Jean-Philippe Aumasson, Jack O'Connor, Samuel Neves, and Zooko. Blake3 hash. `https://github.com/BLAKE3-team/BLAKE3`.

APRH21. Diego F Aranha, Elena Pagnin, and Francisco Rodríguez-Henríquez. Love a pairing. In *International Conference on Cryptology and Information Security in Latin America*, pages 320–340. Springer, 2021.

BBC+13. Fabrice Ben Hamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. Efficient UC-secure authenticated key-exchange for algebraic languages. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 272–291. Springer, Heidelberg, February / March 2013.

BCI+13. Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 315–333. Springer, Heidelberg, March 2013.

BCPT13. Eleanor Birrell, Kai-Min Chung, Rafael Pass, and Sidharth Telang. Randomness-dependent message security. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 700–720. Springer, Heidelberg, March 2013.

BD95. Mike Burmester and Yvo Desmedt. A secure and efficient conference key distribution system (extended abstract). In Alfredo De Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 275–286. Springer, Heidelberg, May 1995.

bls. Bls12-381 curve. `https://electriccoin.co/blog/new-snark-curve/`.

BLS04. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, September 2004.

CCGS10. Jan Camenisch, Nathalie Casati, Thomas Groß, and Victor Shoup. Credential authenticated identification and key exchange. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 255–276. Springer, Heidelberg, August 2010.

CGGN17. Matteo Campanelli, Rosario Gennaro, Steven Goldfeder, and Luca Nizzardo. Zero-knowledge contingent payments revisited: Attacks and payments for services. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 229–243. ACM Press, October / November 2017.

CL06. Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96. Springer, Heidelberg, August 2006.

CS98. Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 13–25. Springer, Heidelberg, August 1998.

cwe. Cweb3. `https://www.npmjs.com/package/cweb3`.

DF17. Yevgeniy Dodis and Dario Fiore. Unilaterally-authenticated key exchange. In Aggelos Kiayias, editor, *FC 2017*, volume 10322 of *LNCS*, pages 542–560. Springer, Heidelberg, April 2017.

DH76. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

ens. Ethereum Name Service. `https://ens.domains/`.

FG10. Dario Fiore and Rosario Gennaro. Identity-based key exchange protocols without pairings. In *Transactions on computational science X*, pages 42–77. Springer, 2010.

GGSW13. Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 467–476. ACM Press, June 2013.

GIKM98. Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. In *30th ACM STOC*, pages 151–160. ACM Press, May 1998.

GKM+18. Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Heidelberg, August 2018.

[17] Specifically, the work in [APRH21] reports a slowdown factor of roughly 2× for $\mathbb{G}_1$ operations and 3× for $\mathbb{G}_2$ operations in BLS12-381 compared to BN254.

GM17. Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 581–612. Springer, Heidelberg, August 2017.

Gro16. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.

han. Handshake: Decentralized naming and certificate authority. `https://handshake.org`.

KY03. Jonathan Katz and Moti Yung. Scalable protocols for authenticated group key exchange. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 110–125. Springer, Heidelberg, August 2003.

Max. Greg maxwell's zero knowledge contingent payment (bitcoin wiki). `https://en.bitcoin.it/wiki/Zero_Knowledge_Contingent_Payment`.

mee. MeetWallet: The Meet JS SDK Library for MEET.ONE Client. `https://meet-common.gitlab.io/fe/meet-js-sdk/classes/meetwallet.html`.

NMKW21. Chan Nam Ngo, Fabio Massacci, Florian Kerschbaum, and Julian Williams. Practical witness-key-agreement for blockchain-based dark pools financial trading. In *International Conference on Financial Cryptography and Data Security*, pages 579–598. Springer, 2021.

Proa. Protocol Labs. Filecoin. `https://filecoin.io/`.

Prob. Protocol Labs. IPFS: Interplanetary file system. `https://ipfs.io`.

RSA78. Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, 1978.

Sha84. Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53. Springer, Heidelberg, August 1984.

SW05. Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Heidelberg, May 2005.

A Unilaterally Witness-Authenticated Key Exchange Model

Unilateral authentication is defined in the two participant setting. One participant, the initiator (Init) is unauthenticated. The authenticated party is called the Responder (Res). The notation for the set of potential participants is changed to be $\mathcal{P} = \{\mathsf{Init}, \mathsf{Res}_1, \ldots, \mathsf{Res}_{\ell-1}\}$, authenticating with respect to the vector $\Phi = \langle \phi_{R_1}, \ldots, \phi_{R_{\ell-1}}, \phi_{\mathsf{Init}} \rangle$ where ϕ_{Init} is a "dummy statement" for which a witness can be computed in polynomial time. All participant-associated variables and oracles remain the same as in the group setting of WAKE.

Correctness for $\mathsf{U\text{-}WAKE}$ then requires that Init, when executing the protocol with any Res_j (authenticated with respect to the instance ϕ_{R_j}), will accept and the two participants will terminate with a common session key. Correctness is adapted to explicitly apply to the unilateral authentication case but it should be noted that Definition 10 still applies under the condition that the statement ϕ_{Init} is easy.

Definition 17 ($\mathsf{U\text{-}WAKE}$ Correctness). *A $\mathsf{U\text{-}WAKE}$ protocol Π is correct if for all relations $\mathcal{R}$, for all sets of potential participants $\mathcal{P} = \{\mathsf{Init}, \mathsf{Res}_1, \ldots, \mathsf{Res}_{\ell-1}\}$ of size $\ell = \ell(\lambda)$, for all $i, j, k \in \mathbb{N}$ such that $\phi_{\mathsf{Res}_k}, \mathsf{w}_{\mathsf{Res}_k} \in \mathcal{R}$, $\mathsf{sid}^i_{\mathsf{Init}} \equiv \mathsf{sid}^j_{\mathsf{Res}_k}$ and $\mathsf{acc}^i_{\mathsf{Init}} = \mathsf{acc}^j_{\mathsf{Res}_k} = \mathsf{TRUE}$ then $\mathsf{sk}^i_{\mathsf{Init}} = \mathsf{sk}^j_{\mathsf{Res}_k}$.*

Confidentiality: The goal of the adversary in the confidentiality game is to be able to distinguish a random key from the real session key generated by an eavesdropped protocol execution. The main modification to this experiment is that the challenge should be an instance of Init.

Definition 18 ($\mathsf{U\text{-}WAKE}$ Confidentiality). *The advantage of an adversary $\mathcal{A}$ with respect to the $\mathsf{U\text{-}WAKE}$ protocol Π in the confidentiality game seen in Figure 15 is defined as*

$$\mathbf{Adv}^{\mathsf{U\text{-}WAKE\text{-}confid}}_{\Pi,\mathcal{A}}(\lambda, \mathcal{R}, \Phi, \mathcal{D}_\Phi) = |2 \cdot \mathsf{Pr}[\mathbf{Exp}^{\mathsf{U\text{-}WAKE\text{-}confid}}_{\Pi,\mathcal{A}}(\lambda, \mathcal{R}, \Phi, \mathcal{D}_\Phi) = 1] - 1|$$

The $\mathsf{U\text{-}WAKE}$ protocol Π is confidential if, for all $\lambda \in \mathbb{N}$, for all relations $\mathcal{R}$, for all statement vectors Φ, for all distributions over witness sets $\mathcal{D}_\Phi$ and for all non-uniform admissible PPT $\mathcal{A}$, the advantage of $\mathcal{A}$ is negligible.

$\mathbf{Exp}^{\mathsf{U\text{-}WAKE\text{-}confid}}_{\Pi,\mathcal{A}}(\lambda, \mathcal{R}, \Phi, \mathcal{D}_\Phi)$

$b \xleftarrow{\$} \{0,1\}$

$pp \leftarrow \mathsf{SetUp}(1^\lambda, \mathcal{R})$

$\mathsf{w} \leftarrow \mathcal{D}_\Phi$

$\mathcal{P} \leftarrow \{\mathsf{Res}_i[\phi_i, \mathsf{w}_i]\}_{i=1}^{|\mathsf{w}|} \cup \{\mathsf{Init}[\phi_{\mathsf{Init}}, \mathsf{w}_{\mathsf{Init}}]\}$

$i \leftarrow \mathcal{A}^{\mathsf{Execute}(),\mathsf{Reveal}()}(\Phi, \mathsf{w})$

if $\mathsf{acc}^i_{\mathsf{Init}} = \mathsf{FALSE}$: output b

else : $\mathsf{k}_1 \leftarrow \mathsf{sk}^i_{\mathsf{Init}}, \mathsf{k}_0 \xleftarrow{\$} \mathcal{K}$

$b' \leftarrow \mathcal{A}^{\mathsf{Execute}(),\mathsf{Reveal}()}(\mathsf{k}_b)$

if $b = b'$: output 1

else : output 0

Fig. 15: $\mathbf{Exp}^{\mathsf{U\text{-}WAKE\text{-}confid}}_{\Pi,\mathcal{A}}$: Experiment for U-WAKE confidentiality.

Authenticity: The goal of the adversary in the authenticity experiment should be to convince the Initiator to accept and consequently generate a session key. Therefore, the first and most obvious change to Figure 5 should be that the challenge instance output by $\mathcal{A}$ should always be an instance of Init. Then, $\mathcal{A}$ must convince Π^i_{Init} to accept without knowledge of a witness. The adversary $\mathcal{A}$ must accomplish this goal without merely playing as a wire between Init and some Res and without otherwise cheating. Note that the extractor outputs a witness for the participant output along with the challenge instance of Init, which is a witness corresponding to ϕ_j for $\Pi^k_{R_j}$.

Definition 19 (U-WAKE Authenticity). *The advantage of an adversary $\mathcal{A}$ with respect to the* U-WAKE *protocol Π in the authentication game seen in Figure 16 is defined as*

$$\mathbf{Adv}^{\mathsf{U\text{-}WAKE\text{-}auth}}_{\Pi,\mathcal{A}}(\lambda, \mathcal{R}, \Phi, \mathcal{D}) = \mathsf{Pr}[\mathbf{Exp}^{\mathsf{U\text{-}WAKE\text{-}auth}}_{\Pi,\mathcal{A}}(\lambda, \mathcal{R}, \Phi, \mathcal{D}) = 1]$$

A U-WAKE *protocol Π is witness-authenticated if for all admissible non-uniform PPT $\mathcal{A}$ there exists a PPT extractor $\mathcal{E}_\mathcal{A}$ which for all $\lambda \in \mathbb{N}$, for all relations $\mathcal{R}$, for all statement vectors Φ, for all witness distributions $\mathcal{D}_\Phi$, the advantage of $\mathcal{A}$ is negligible.*

Simulatability: The simulatability experiment remains unchanged.

B Proof of Theorem 1

Restatement of Theorem 1: Let KEM be a correct and CPA-secure key encapsulation mechanism. Let SOK be a perfectly correct and simulatable, simulation-extractable signature of knowledge. Then, the protocol Π, as seen in Figure 7, is an actively secure and simulatable U-WAKE.

Proof. **Correctness**: Correctness follows directly from the correctness of the KEM and SOK. As these are both perfectly correct the U-WAKE is also perfectly correct.

Confidentiality: An adversary $\mathcal{A}_C$ with nonnegligible advantage in the confidentiality experiment implies the existence of an adversary $\mathcal{A}_K$ with nonnegligible advantage in the KEM-CPA experiment. Assume that $\mathcal{R}$, Φ, $\mathcal{D}_\Phi$ are such that $\mathcal{A}_C$ has a nonneligible advantage in the confidentiality experiment.

The adversary $\mathcal{A}_K$ is given challenge tuple $(\mathsf{ek}^*, \mathsf{C}^*, \mathsf{k}^*_b)$ and must guess the KEM-CPA real or random bit $b_{\mathsf{KEM\text{-}CPA}}$. $\mathcal{A}_K$ sets up the U-WAKE as follows: generates the public parameters for the signature of knowledge $pp_{\mathsf{SOK}} \leftarrow \mathsf{SOK.SSetup}(\mathcal{R})$, and sets the public parameters as $pp_{\mathsf{WAKE}} = \{\lambda, \mathcal{R}, pp_{\mathsf{SOK}}\}$. Then, $\mathcal{A}_K$ runs $\mathcal{A}_C$ with input $pp_{\mathsf{WAKE}}, \Phi, \mathsf{w}$.

$\mathbf{Exp}^{\mathsf{U\text{-}WAKE\text{-}auth}}_{\Pi,\mathcal{A}}(\lambda, \mathcal{R}, \Phi, \mathcal{D}_\Phi)$

$pp \leftarrow \mathsf{SetUp}(1^\lambda, \mathcal{R})$

$\mathsf{w} \leftarrow \mathcal{D}_\Phi$

$\mathcal{P} \leftarrow \{\mathsf{Res}_i[\phi_i, \mathsf{w}_i)]\}_{i=1}^{|\mathsf{w}|} \cup \{\mathsf{Init}[\phi_{\mathsf{Init}}, \mathsf{w}_{\mathsf{Init}}]\}$

$(i, j) \leftarrow \mathcal{A}^{\mathsf{Send}(),\mathsf{Execute}()}(\Phi)$

assert $(\mathsf{Res}_j \in \mathcal{IS}(\mathsf{Init}, i))$

$b_{\mathsf{acc}} \leftarrow \mathsf{acc}^i_{\mathsf{Init}}$

$\mathsf{w}' \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{view}_{\mathcal{A}})$

$b_{\mathsf{ext}} \leftarrow (\phi_{R_j}, \mathsf{w}') \in \mathcal{R}$

output $(b_{\mathsf{acc}} \wedge \bar{b}_{\mathsf{ext}})$

Fig. 16: $\mathbf{Exp}^{\mathsf{U\text{-}WAKE\text{-}auth}}_{\Pi,\mathcal{A}}$: Experiment for U-WAKE authenticity.

In response to queries to Execute of the form $q_k = (\mathsf{Init}, k, \mathsf{Res}_j, i)$, $\mathcal{A}_K$ generates an honest transcript between Π^k_{Init} and $\Pi^i_{\mathsf{Res}_j}$ of the form $T_k = \langle \mathsf{ek}_k, (C_k, \sigma_k)\rangle$ with signature generated as $\sigma_k \leftarrow \mathsf{SOK.SSign}(pp_{\mathsf{SOK}}, \phi_j, \mathsf{w}_j, \mathsf{ek}_k||C_k)$. $\mathcal{A}_K$ stores the generated session key as $\mathsf{sk}^k_{\mathsf{Init}}$ and $\mathsf{sk}^i_{\mathsf{Res}_j}$, sets $\mathsf{sid}^k_{\mathsf{Init}} = \mathsf{sid}^i_{\mathsf{Res}_j} = T_k$ and sets $\mathsf{acc}^k_{\mathsf{Init}} = \mathsf{acc}^i_{\mathsf{Res}_j} = \mathsf{TRUE}$. Let Q_E be the total number of queries made to Execute by $\mathcal{A}_C$. With probability $\frac{1}{Q_E}$, $\mathcal{A}_C$ responds to the query with $T^* = \langle \mathsf{ek}^*, (C^*, \sigma^*)\rangle$ for $\sigma^* \leftarrow \mathsf{SOK.SSign}(pp_{\mathsf{SOK}}, \phi_j, \mathsf{w}_j, \mathsf{ek}^*||C^*)$. In response to queries to Reveal of the form (U, i), $\mathcal{A}_K$ replies with sk^i_U.

If $\mathcal{A}_C$ outputs challenge i such that $\mathsf{sid}^i_{\mathsf{Init}} \equiv T^*$ then $\mathcal{A}_K$ provides k^*_b as the challenge session key. In this case, when $\mathcal{A}_C$ outputs her guess bit b'_{confid}, $\mathcal{A}_K$ forwards this bit as her guess for $b_{\mathsf{KEM\text{-}CPA}}$. If $\mathcal{A}_C$ outputs some other challenge instance $\mathcal{A}_K$ flips a bit and provides either the real key or a random key. Given that T^* appears only once in the list of Q_E queries, the probability that $\mathcal{A}_C$ selects T^* as her challenge is $\frac{1}{Q_E}$.

$\mathcal{A}_K$ runs $2Q_E$ independent copies of $\mathcal{A}_C$. Call E the event that T^* appears only once in the set of responses to all queries. Consequently:

$$\begin{aligned}
\mathsf{Pr}[\mathsf{Exp}^{\mathsf{KEM\text{-}CPA}}_{\mathcal{A}_K}(\lambda)] &\geq 2Q_E \cdot \mathsf{Pr}[\mathsf{Exp}^{\mathsf{WAKE\text{-}confid}}_{\Pi,\mathcal{A}_C}(\lambda, \mathcal{R}, \Phi, \mathcal{D}_\Phi)] \cdot \mathsf{Pr}[T^* \leftarrow \mathcal{A}_C(\Phi, \mathsf{w})] + \mathrm{negl}(\lambda) \\
&= 2Q_E \cdot \mathsf{Pr}[\mathsf{Exp}^{\mathsf{WAKE\text{-}confid}}_{\Pi,\mathcal{A}_C}(\lambda, \mathcal{R}, \Phi, \mathcal{D}_\Phi)] \cdot \mathsf{Pr}[T^* \leftarrow \mathcal{A}_C(\Phi, \mathsf{w})|\mathsf{E}] \cdot \mathsf{Pr}[\mathsf{E}] + \mathrm{negl}(\lambda) \\
&= 2 \cdot \mathsf{Pr}[\mathsf{Exp}^{\mathsf{WAKE\text{-}confid}}_{\Pi,\mathcal{A}_C}(\lambda, \mathcal{R}, \Phi, \mathcal{D}_\Phi)] \cdot \mathsf{Pr}[\mathsf{E}] + \mathrm{negl}(\lambda) \\
&\geq 2 \cdot (1 - \frac{1}{e}) \cdot \mathsf{Pr}[\mathsf{Exp}^{\mathsf{WAKE\text{-}confid}}_{\Pi,\mathcal{A}_C}(\lambda, \mathcal{R}, \Phi, \mathcal{D}_\Phi)] + \mathrm{negl}(\lambda) \\
&\geq \mathsf{Pr}[\mathsf{Exp}^{\mathsf{WAKE\text{-}confid}}_{\Pi,\mathcal{A}_C}(\lambda, \mathcal{R}, \Phi, \mathcal{D}_\Phi)] + \mathrm{negl}(\lambda)
\end{aligned}$$

Authenticity: An adversary $\mathcal{A}_A$ with nonnegligible advantage against the authenticity game implies an adversary $\mathcal{A}_S$ against the sig-ext of the SOK. Assume that $\mathcal{R}$, Φ, $\mathcal{D}_\Phi$ are such that $\mathcal{A}_A$ has a nonneligible advantage in the authenticity experiment.

$\mathcal{A}_S$ assigns as the public parameters $pp_{\mathsf{WAKE}} \leftarrow \{pp_{\mathsf{SOK}}, \mathcal{R}, \lambda\}$ and runs $\mathcal{A}_A$ with input Φ. Queries to Send and Reveal are answered honestly by $\mathcal{A}_S$, all signatures are generated with queries to the SSimSign oracle.

$\mathcal{A}_A$ outputs her challenge $(\mathsf{Init}, i, \mathsf{Res}_j)$ with $\mathsf{Res}_j \in \mathcal{IS}(\mathsf{Init}, i)$. $\mathcal{A}_A$ is admissible, thus $\mathcal{A}_A$ was not forwarding for Π^i_{Init} and Res_j. By Definition 11 this implies that for all $\mathsf{Res}_{j'} \in \mathcal{P}$ such that $\phi_{\mathsf{Res}_{j'}} = \phi_{\mathsf{Res}_j}$ we have for all k': $\mathsf{sid}^{k'}_{\mathsf{Res}_{j'}} \not\equiv \mathsf{sid}^i_{\mathsf{Init}}$. We assume that the only participant authenticating with respect to ϕ_j is Res_j, as there is only a single message sent from the Res to the Init. $\mathcal{A}_A$ either (1) sent to the instance $\Pi^k_{\mathsf{Res}_j}$, for some k, a new encapsulation key not generated by the initiator $\mathsf{ek}' \neq \mathsf{ek} \leftarrow \mathsf{Send}(\mathsf{Init}, i, \mathsf{Res}_j)$, or (2) sent to the initiator a signed ciphertext not output by a corresponding send query $(C', \sigma') \neq (C, \sigma) \leftarrow \mathsf{Send}(\mathsf{Res}_j, k, \hat{\mathsf{ek}})$, or (3) both. Consider case (1) where $\mathcal{A}_A$ did not forward the first message, but forwarded the second. The signature σ must be a signature of the message $m = C||\mathsf{ek}$, so if $\mathcal{A}_A$ only supplied an $\mathsf{ek}' \neq \mathsf{ek}$, σ will not verify and Init

will not accept. $\mathcal{A}_A$ must not have replaced only the first message, she must have replaced either the second or both messages. In the remaining cases (2) and (3), (C', σ') was not generated as a response to a query $\mathsf{Send}(\mathsf{Res}_j, k, \hat{\mathsf{ek}})$ for any $k, \hat{\mathsf{ek}}$. Therefore σ' was not an output of some query to the $\mathsf{SSimSign}$ oracle made by $\mathcal{A}_C$, but Π^i_{Init} accepted and therefore we know that σ' verifies: $\mathsf{SVfy}(pp_{\mathsf{SOK}}, \phi_{\mathsf{Res}_j}, m = (C'||\mathsf{ek}), \sigma') = 1$ for ek output by Π^i_{Init}. So, $\mathcal{A}_S$ outputs $(\phi_j, m = (C'||\mathsf{ek}), \sigma')$ as her forgery.

If there exists an extractor for $\mathcal{A}_S$ then there necessarily exists an extractor for $\mathcal{A}_A$. The view of $\mathcal{A}_S$ contains no information that cannot be calculated in polynomial time from $\mathsf{view}_{\mathcal{A}_A}$. Let this transformation be $\mathsf{view}_{\mathcal{A}_S} = T(\mathsf{view}_{\mathcal{A}_A})$. Construct $\mathcal{E}_{\mathcal{A}_A}$, running on $\mathsf{view}_{\mathcal{A}_A}$, assuming the existence of $\mathcal{E}_{\mathcal{A}_S}$ as follows: $\mathcal{E}_{\mathcal{A}_A}$ runs T on the view of the $\mathcal{A}_A$ to get the $\mathsf{view}_{\mathcal{A}_S}$ then runs $\mathcal{E}_{\mathcal{A}_S}$ to get a witness w' and outputs this witness. Therefore, because there is no extractor for $\mathcal{A}_A$ there is no extractor for $\mathcal{A}_S$.

The advantage $\mathcal{A}_S$ is then non-negligible, as it is greater that that of $\mathcal{A}_A$.

Simulatability: $\mathcal{A}_W$ with nonnegligible advantage against $\mathsf{U\text{-}WAKE}$ simulatability implies the existance of an adversary $\mathcal{A}_S$ against SOK simulation. Assume that $\mathcal{R}$ is such that $\mathcal{A}_W$ has a nonnegligle advantage in the simulation experiment.

On input pp_b, $\mathcal{A}_S$ constructs the public parameters for the $\mathsf{U\text{-}WAKE}$ as $pp_b = \{pp^b_{\mathsf{SOK}}, \mathcal{R}, \lambda\}$ and forwards this to $\mathcal{A}_W$.

For the ith call to $\mathsf{SetKeys}^b$ by $\mathcal{A}_W$, $\mathcal{A}_S$ saves the statement witness pair as (ϕ_i, w_i) to use when responding to queries for Responder Res_i. $\mathcal{A}_S$ honestly responds to all queries to Send and Reveal, except instead of using the signing algorithm she queries her oracle $\mathcal{S}^b_{pp_b,\tau}$ to generate any signatures. Upon receipt of the guess bit b' from $\mathcal{A}_W$, $\mathcal{A}_S$ uses this as her own guess for the real or random bit.

The advantage of $\mathcal{A}_S$ is then non-negligible, as it is greater than that of $\mathcal{A}_W$.