Witness-Authenticated Key Exchange, Revisited

Extensions to Groups, Improved Models, Simpler Constructions*

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

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

Abstract. We study *witness-authenticated key exchange* (WAKE), in which parties authenticate through knowledge of a witness to any NP statement. WAKE achieves generic authenticated key exchange in the absence of trusted parties; WAKE is most suitable when a certificate authority is either unavailable or undesirable, as in highly decentralized networks. In practice WAKE approximates witness encryption, its elusive non-interactive analogue, at the cost of minimal interaction.
This work is the first to propose, model and build witness-authenticated key exchange amongst *groups* of more than two parties, as well as the first to provide *practical* and *provably secure* constructions in the two-party case for *general* NP statements. Specifically our contributions are:

1. both game-based and universally composable (Canetti, FOCS '01) definitions for WAKE along with equivalence conditions between the two definitions,
2. a highly general compiler that introduces witness-authentication to any key exchange protocol along with, as a direct consequence, a three-round group WAKE protocol from DDH and signatures of knowledge (SOK), and
3. an optimized two-round group WAKE construction from DDH and SOK along with experimental benchmarks to demonstrate concrete practicality.

Additionally, we study the specialized two-party case and provide a critique of prior work on this topic (Ngo et al., Financial Crypto '21) by pinpointing nontrivial weaknesses in the model, constructions and security proofs seen therein. We rectify those limitations with this work, significantly diverging in our techniques, design and approach.

1 Introduction

Public-Key cryptography as introduced in the seminal paper by Diffie and Hellman [23] allows two parties who have never met to confidentially exchange information. This can be achieved via *non-interactive* encryption with which a sender

* Up to date full version at https://eprint.iacr.org/2022/382.

encrypts a message under a receiver's public key [36], or via an *interactive* key exchange [23].

Traditionally parties' identities are endorsed by a trusted certifying authority (CA) through published *certificates* i.e. digital signatures binding identities to public keys. Beyond such a public key infrastructure, in both the encryption and key-exchange cases, there are more flexible methods of designating for which entities secure communications are intended: in *identity-based* cryptography [38] public keys are replaced by arbitrary party identity strings and in *attribute-based* cryptography [37] policies define sets of attributes that parties must satisfy. Critically, both of these primitives employ a trusted party to issue certificates or keys corresponding to a party's claimed identity or attributes and therefore can be considered to assume a centralized CA.

Witness encryption (WE) [27] is arguably the most general way to specify the intended recipient of an encrypted message. With WE a message is encrypted to an instance ϕ of some NP language L such that if $\phi \in L$ then the message can be recovered via an efficient decryption algorithm which requires as input a witness w for ϕ. As an example one can encrypt a message to a Sudoku puzzle resulting in a WE ciphertext that is decryptable with any valid solution to that puzzle. Remarkably this primitive does not require a trusted party "to certify a receiver's key"; in WE the secret decryption key is the (uncertified) witness itself. Despite recent progress [11] practical WE for all of NP from standard assumptions remains elusive.

In this work we turn our attention to another "witness-flavored" primitive that *(i)* similarly to WE, can be used in several applications where an *arbitrary*[1] secret enables secure communication, yet *(ii)* differently from WE, can be conretely and efficiently instantiated under computational assumptions *held today*. We achieve the latter through an extremely low amount of interaction – in some cases only two messages. We refer to this primitive, the interactive process in which parties mutually authenticate with respect to knowledge of a witness to an NP statement, as *witness-authenticated key exchange* (WAKE). We argue that the interactive abstraction deserves to be studied in its own right, due to both its utility in several applications and its generality. Just as WE is the most general form of encryption, the interactive witness-authenticated key exchange subsumes all efficiently verifiable means of authentication as special cases.

Our Contributions. This work is the first to propose, model and build witness-authentication amongst *groups* of authenticated parties. We are also the first to provide *practical* and *provably secure* constructions for two-parties authenticating under *general* statements. Our contributions are as follows:

(1) We revisit witness-authentication: we identify the shortcomings of [33]—the model's applicability, the construction's practicality, and the security proof. Our discussion demonstrates a need for new models and constructions in the two-party case, a specialization of our general model for groups.

[1] That is, not certified by an authority and not the output of a specific key generation algorithm e.g. the Sudoku solution in the example above.

(2) We define WAKE via both a modular and intuitive game-based formalization and a universally composable (UC) ideal functionality. UC is the most desirable security guarantee for authenticated key exchange; satisfaction of the stronger variant of our game-based definition is proven equivalent to UC-realizing our ideal functionality.

(3) We construct new protocols for group WAKE. We first show a *general compiler* transforming any *passively* secure key exchange protocol (without any guarantees on witnesses held by each user) into one with witness-authentication and thus full security. The main features of the general construction is that it is conceptually simple, modular and efficient. Abstractly the compiler relies on strongly simulation-extractable signatures of knowledge [30], the properties of which can be leveraged to provide different security and efficiency guarantees. Concretely we instantiate our compiler on the Diffie-Hellman style group key exchange of Burmester and Desmedt to provide a practical 3-round WAKE protocol.

(4) We optimize the above 3-round solution to a 2-round *concretely efficient* group WAKE protocol. This improved solution achieves UC security, though this does not trivially follow from the security of the above compiler and demands particular care in the proof. We show that this protocol remains practical, as evidenced by our estimated benchmarks in Section 4.

Our general protocol in a nutshell. We begin with the Katz-Yung [31] approach to authenticated group key exchange but replace the standard digital signature authentication mechanism with simulation-extractable signatures of knowledge (SOK) [30], enabling a party to verifiably claim knowledge of a particular secret. While the idea is fundamentally simple, modeling the WAKE primitive and proving security of the protocols requires much care and should be considered our main technical contribution.

Applications. The main application of WAKE is, of course, the establishment of secure communication channels between groups of parties contingent upon the information that parties provably know. The lack of a trusted CA renders WAKE especially compelling for decentralized applications requiring parties to confidentially connect based on arbitrary, dynamic policies. Decoupling authentication and identity permits flexible, deniable and anonymous authentication. We present several concrete examples of applications employing WAKE. Estimated runtime benchmarks for parties authenticating in dark pools and retrieval markets can be found in Section 4.

Dark Pool Transactions. Appearing as the primary motivation for a predecessor to WAKE [33], dark pool transactions allow for confidential and anonymous negotiation. In this scenario Alice is selling an item and wants to confidentially negotiate with any party holding enough funds to purchase that item. Alice determines a minimum balance B and any party Bob can establish a key with Alice if, for a public commitment c to his private balance b, the following relation is satisfied $\mathcal{R}\big((B,c),(b,r)\big) = \big(c = \mathsf{comm}(b;r) \wedge b \geq B\big)$. Alice can remain unauthenticated, as in *unilateral* WAKE, or Bob may wish that Alice authenticates her ownership of the item for sale. Given our group WAKE the dark pools

Primitive	Model						Protocol	
	GB	UC	All NP	No CA	ML	Groups	$O(1)$-rnd	All NP
PAKE [12]	✓	✓		✓	✓		✓	
CAKE [17]		✓	✓		✓			
ABKE [32]		✓	✓				✓	✓
LAKE [13]		✓	✓	✓	✓		✓	
WKA* [33]	✓		✓	✓				
WAKE **[This Work]**	✓	✓	✓	✓	✓	✓	✓	✓

Table 1. Comparison to related work. `Model/GB`: "has a game-based model?" `Model/UC`: "has a UC-based model?" `Model/All NP`: "does the model support any efficiently computable relation over the inputs of each party?" `Model/No CA`: "can do without an underlying CA?" `Model/ML`: "supports multi-lateral authentication?" `Model/grp`: "supports more than two parties?" `Protocol/`$O(1)$`-rnd`: "has a constant number of rounds?" `Protocol/All NP`: "does the construction support any efficiently computable relation?"; `WKA*` is discussed further in Section 1.

scenario can be extended to a group chat between many parties, each satisfying the condition of holding enough funds to participate.

Retrieval Markets. In decentralized storage systems such as Filecoin [34] and IPFS [35] files are stored by providers and addressed with a *content identifier (CID)* that is typically a cryptographic hash of the file additionally serving as a commitment to that file. A provider can authenticate to any client interested in retrieving the file if that provider holds the private file associated to the public CID. Similarly providers storing the same file can establish a confidential group channel to communicate via group WAKE.

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. [3, 7]). A group WAKE can be used to establish secure communication channels for such tasks. Thanks to the inherent flexibility of WAKE these existing schemes can be extended to more general conditions, e.g. confidential group chatrooms between owners of NFTs by a certain artist.

Decentralized Anonymous Routing. Several proposals have been put forward for decentralized naming and routing protocols over the internet (see e.g. [6] and [4]). WAKE can play an important role in securing such protocols by providing a method by which parties can authenticate without a CA.

Related Work. The relevant generalizations of authenticated key exchange are summarized in Table 1. Credential-authenticated and attribute-based key exchange (CAKE [17] and ABKE [32]) respectively model bilateral and unilateral authentication on the basis of efficiently computable relations over a certified set of credentials. Both CAKE and ABKE stipulate an authority to issue these certificates (CA) and therefore depend upon the strong assumption of unani-

mous trust in an honest, incorruptible CA. Beyond this impracticality and other certificate governance concerns, no CA could possibly certify any and every arbitrary property potentially desirable for authentication; any CA is a limitation against a truly general key exchange.

In the absence of a CA password-authenticated key exchange (PAKE) permits parties to boost shared low entropy passwords into a shared high entropy key [12], and has been generalized to noisy, approximately equal passwords [25]. Even closer to the goal of witness-authentication is language-authenticated key exchange (LAKE [13]), which enables two parties to establish a shared key over an insecure network if each participant has knowledge of a word that lies in the language defined by their partner. Notably, the language and the statement remain secret with LAKE. In contrast, our definition of WAKE does not *straightforwardly* guarantee secrecy of the statement and language as modeled. As the model closest to our goal, it is important to note that the LAKE protocol exclusively supports self-randomizable algebraic languages, the a subset of languages which admit a smooth projective hash function [22]. Therefore the LAKE protocol cannot support all of NP.[2]

We note that in comparison to generic multiparty computation (MPC) or fully homomorphic encryption, state of the art SNARKs are both mature and efficient. One primary observation that we wish to communicate is the simplicity of our solutions. A custom protocol for WAKE, as in Section 4, is lighter and simpler than employing the heavy hammer that is MPC.

Authenticated key exchange (AKE) can be modeled in a game-based way or in the universal composability framework. A game-based security definition is a natural, modular formalization of our intuitions about exactly which properties are desirable for security. A composable definition models the key exchange as an ideal functionality, or a trusted party that is given all of the relevant secrets to carry out that task, and ultimately UC security guarantees that an AKE protocol is (1) just as secure as the ideal functionality, and (2) remains secure independently of how the keys are employed afterwards. We provide both formalizations and give equivalence conditions between the two.

Analysis of Witness Key Agreement (WKA) [33]. We briefly discuss witness-key-agreement (WKA) [33], which inspired our work and proposed interesting applications. We identify three primary shortcomings of WKA, discussed further in Appendix A. First, a note on terminology: WAKE and WKA aim at modeling similar settings but the models in our work are more general. In light of the observations in this section we chose to reflect these major differences in approach by further differentiating WAKE from WKA in name.

Limitations of the WKA model. A standard security requirement for key exchange is that an adversary should not be able to learn anything about the session key from the interaction—a random key should be *indistinguishable* from those output by participants in the protocol. Against active adversaries, however, the [33] definition employs *unpredictability* in lieu of indistinguishability.

[2] If every language in NP admits a smooth projective hash function then the polynomial hierarchy collapses.

As a consequence this model may consider secure a construction where an active adversary *is able* to distinguish keys. Additionally, active adversaries are unrealistically weakened: they can generate *only a single session* to leverage against the challenge. Furthermore, confidentiality in WKA neglects the case that an adversary knows some relevant witness as auxiliary input.[3]

Is the WKA protocol provably secure? We observe that security for the scheme in [33] requires a more elaborate proof under stricter assumptions than those acknowledged in the paper. We do not know if the assumed primitive is realizable. The construction in [33] can be seen as a designated-verifier proof system, adapted and extended from the ideas in [15]. The [15] construction argues soundness under the IND-CPA security of a linear-only encryption (LOE) scheme and includes these LOE ciphertexts in the CRS. WKA [33] augments that CRS with encryptions of the randomness used to generate the CRS ciphertexts. As such WKA requires a stronger variant of IND-CPA security that accounts for randomness-dependent message (RDM) security [14], yet the proof sketch fails to discuss RDM security. A concrete instance would require an IND-CPA and RDM secure LOE scheme. We are not aware of any such schemes in literature.

Impracticality of the WKA protocol. A primary limitation of WKA is that a trusted setup is required *every time* a new party wants to initiate a key exchange.[4] Every time a party aims to initiate an exchange an authority must be invoked to distribute a secret (tantamount to a verification key in a designated-verifier SNARK) for the key exchange; the trusted authority is invoked at least once per party in the system. Such an exchange is highly impractical. Beyond this impracticality it is undesirable to have a trusted centralized authority producing—and potentially leaking—trapdoors that allow impersonation. If relied on frequently this may compromise the entire system. On the other hand our instantiations can rely on no trusted setup [21], one trusted setup generated once and for all per given computation (reusable in all relevant key exchanges) [30] or a single trusted setup generated once and for all [26].

2 Our Model: Witness-Authenticated Key Exchange

We provide both a game-based definition and a universally composable (UC) ideal functionality [19] for WAKE, ultimately proving the two equivalent under certain conditions. The two modeling approaches offer a tradeoff: game-based definitions are modular and make explicit the exact properties we expect from a secure protocol whereas UC guarantees security under composition with arbitrary protocols, as is desirable for key exchange.

The objective of WAKE, and the primary modeling challenge, is that any set of participants terminate with a shared key if each participant *has knowledge*

[3] If Alice and Bob authenticate using witness w the adversary should not learn their key even with knowledge of w; our definition models this case, while in WKA the definition is silent.

[4] Although if the same party wants to run multiple key agreements for the same relation this setup could potentially be reused.

of a witness to their own efficiently verifiable statement. Clearly witness encryption is not a useful starting point in this interactive setting; the unintuitive WE "security-correctness gap"—explicitly requiring semantic security for statements *not in the language* and correctness for statements in the language—is at odds with our goal. Instead our approach takes inspiration from the group key exchange of [31] modified to the unique case of witness-authentication.

In a model with a public key infrastructure (PKI) a participant's identity is typically synonymous with their certified public key. Yet again a direct analogue to the witness-based setting is not straightforward, erroneously conflating knowledge of a witness with identity. The first dissimilarity is redundancy; as opposed to party identity, statements are not necessarily unique. Secondly, a WAKE protocol is required to maintain secrecy of the witnesses. This zero-knowledge flavored security requirement implies that all participants authenticating with respect to the same statement are indistinguishable. Crucially any meaningful notion of personal identity is absent from WAKE; witness-authentication must remain agnostic to the true identity of the sender and instead exclusively asks: *were these protocol messages generated with knowledge of a witness?*

NOTATION. For an oracle $\mathcal{O}$ we use $A^{\mathcal{O}}$ to say that algorithm A has access to oracle $\mathcal{O}$. The transcript of a protocol execution is defined to be the concatenation of all messages sent and received by any participant in the execution. A participant U is initialized with input input_U using square brackets and a protocol executed between a set of participants generating transcript T is written as $(T, \mathsf{out}_1, \ldots, \mathsf{out}_\ell) \leftarrow \langle P_1[\mathsf{input}_1], \ldots, P_\ell[\mathsf{input}_\ell]\rangle$. The view of party P_i is written view_{P_i} and is defined to be the entire state of the party.

SIGNATURES OF KNOWLEDGE. A signature of knowledge (SOK) [21] is a witness-based generalization of a traditional digital signature. For security parameter λ, NP relation $\mathcal{R}$, statement ϕ, witness w and message m, a SOK has three main algorithms: $\mathsf{SSetup}(1^\lambda, \mathcal{R}) \to pp$ is a randomized relation-specific setup outputting public parameters pp, $\mathsf{SSign}(pp, \phi, w, m) \to \sigma$ is a randomized signing algorithm outputting signature σ and $\mathsf{SVfy}(pp, \phi, m, \sigma) \to \{0, 1\}$ is a deterministic verification algorithm outputting 1 for acceptance. The ideal functionality for SOK can be found in Appendix B.

Correctness requires that verification will accept if the SOK was produced with a valid witness. Towards security we require that a signature of knowledge be simulation-extractable. Simulatability, at a high level, requires that there exists a simulator which can output public parameters that are indistinguishable from those output by SSetup along with a trapdoor τ that can then be used to simulate signatures without witnesses. Simulation-extractability, at a high level, requires that an efficient adversary with access to simulated signatures cannot output a verifying signature without knowledge of a witness. Knowledge of a witness is modeled via the existence of an efficient extractor that can output a witness from the view of the adversary.

THE MODEL. We fix a relation $\mathcal{R}$ and consider a set of participants $\mathcal{P}$ of size $\ell = \ell(\lambda)$. Each $P \in \mathcal{P}$ is associated with the public statement ϕ_P and has input a private string w_P. The public statement vector is $\Phi = \langle \phi_1, \ldots, \phi_\ell \rangle$. We assume

a distribution over witnesses $\mathcal{D}_{\Phi}$ such that each $\mathsf{w} \leftarrow \mathcal{D}_{\Phi}$ is a vector of witnesses $\mathsf{w} = \langle w_1, \ldots, w_\ell \rangle$ corresponding to the statements in Φ.[5]

Each $P \in \mathcal{P}$ can participate in a polynomial number of protocol sessions with arbitrary subsets of $\mathcal{P}$. This is modelled via single-execution instances: Π_P^i is the ith instance of participant P. In addition to (ϕ_P, w_P) each instance stores a boolean acc_P^i indicating acceptance, a session identifier sid_P^i i.e. the transcript, and the session key sk_P^i. Relation-specific public parameters are generated via a setup algorithm: $\mathsf{pp} \leftarrow \mathsf{SetUp}(1^\lambda, \mathcal{R})$.[6] **Correctness** requires that for all NP relations $\mathcal{R}$ and set of participants $\mathcal{P}$, instances terminate with a shared key if all parties know witnesses to their associated statements and the instances have matching session identifiers.[7]

Admissible Adversaries. The adversary controls all communication between participants in the network via three oracles: $\mathsf{Send}(P, i, M)$ sends message M to instance Π_P^i and returns the response, $\mathsf{Execute}(P_{i_1}, j_1, \ldots, P_{i_k}, j_k)$ runs an honest execution of the protocol between the queried instances and $\mathsf{Reveal}(P, i)$ outputs session key sk_P^i. A *passive* adversary, playing as a wire between parties, attempts to learn about the key from the session transcript. An *active* adversary, participating in an exchange by injecting her own messages into a session, attempts to successfully authenticate to any other participant in that exchange. Notably the passive adversary trivially convinces every party in the exchange to accept by merely forwarding protocol messages between parties but does not do so adversarially. We therefore must clearly define what adversarial behavior is considered admissible.

A **forwarding adversary** engages in passive behavior; a forwarding adversary is such that for every Send query with input message M and instance Π_Q^j (except the first) there exists a preceding call to Send which output the message M as a response. Moreover, the query which output message M must have taken as input an instance Π_P^i with session identifier $\mathsf{sid}_P^i \equiv \mathsf{sid}_Q^j$. For each instance Π_P^i we define the **impersonation set** of an instance as $\mathcal{I}(P, i)$ as the set of instances to which the adversary *impersonated* Π_P^i by injecting her own message "from" Π_P^i that was not output by a corresponding query $\mathsf{Send}(P, i, \cdot)$. Notably, by definition, a passive adversary without access to the Send oracle cannot be forwarding. Likewise, queries to Reveal can trivially compromise session keys. This motivates the following freshness requirement: an instance is considered **fresh** if the adversary has neither revealed the session key stored by that instance nor the key stored by any instance participating in the same session.[8]

[5] 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_P and witnesses w_P by their associated participant P when discussing a single instance.

[6] The syntax requires one setup per relation but it can be easily extended to the case where a setup is universal [29].

[7] As the session identifiers are set to be the transcript of the session having matching session identifiers indicates that the instances have recorded the same transcript and therefore were participating in the same session.

[8] Instances are participating in the same protocol session if the stored session identifiers agree on the first round messages.

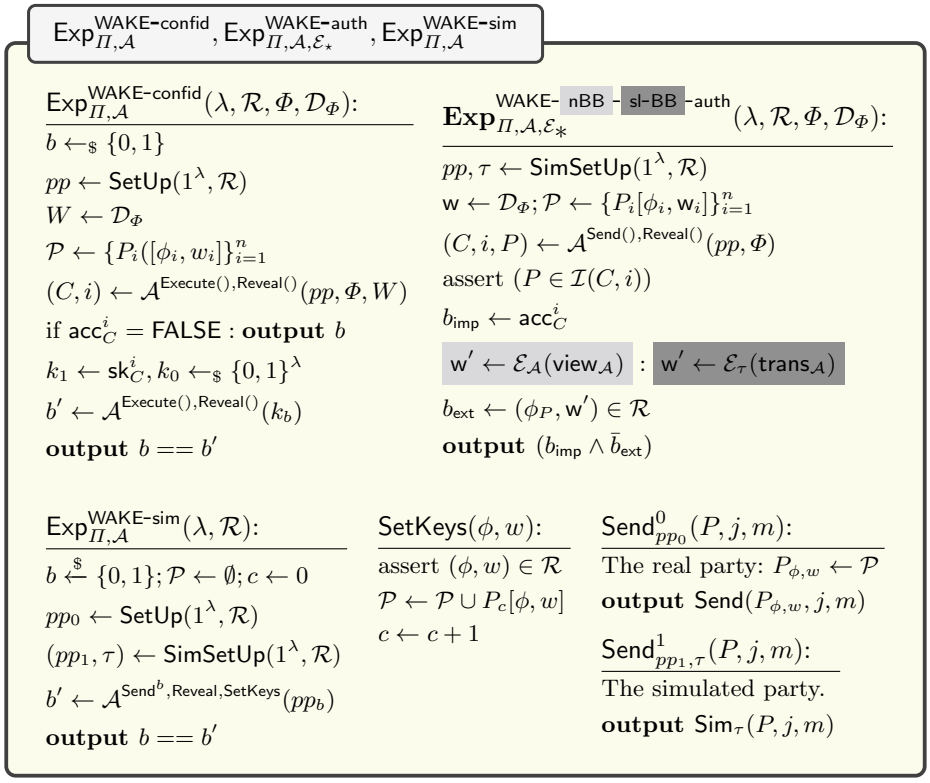

Fig. 1. The WAKE confidentiality, authenticity and simulatability experiments.

An **admissible adversary** is an adversary that does not trivially compromise the session key; an adversary $\mathcal{A}$ is considered admissible for an experiment if $\mathcal{A}$ outputs a fresh challenge (P,i) on which $\mathcal{A}$ is not forwarding.

Security. Minimally a WAKE protocol should be secure in the standard unauthenticated key exchange sense. Confidentiality is the standard security notion and guarantees that an eavesdropping adversary with access to an arbitrary number of honest transcripts and session keys cannot distinguish a random string from the real session key associated to some adversarially-chosen challenge transcript. In the confidentiality experiment, $\mathsf{Exp}^{\mathsf{WAKE\text{-}confid}}_{\Pi,\mathcal{A}}$ in Figure 1, the adversary has access to Execute and Reveal and is also given the entire vector of witnesses. This formulation strengthens confidentiality to additionally provide forward secrecy. We observe that this is a slightly stronger variant of forward secrecy than that modelled via a corruption oracle as in [31].

Definition 1 (Confidentiality). *Consider experiment* $\mathsf{Exp}^{\mathsf{WAKE\text{-}confid}}_{\Pi,\mathcal{A}}$ *in Figure 1. A* WAKE *protocol* Π *is confidential if 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* $\mathcal{A}$*:* $|2 \cdot \Pr[\mathbf{Exp}^{\mathsf{WAKE\text{-}confid}}_{\Pi,\mathcal{A}}(\lambda,\mathcal{R},\Phi,\mathcal{D}_\Phi)] - 1| \leq \mathsf{negl}(\lambda)$.

Simulatability is the requirement that the messages exchanged in a protocol hide the witnesses used to generate them. This means that an adversary cannot learn anything about a participant's witness from the messages sent by that participant. The simulatability experiment is $\mathsf{Exp}^{\mathsf{WAKE\text{-}sim}}_{\Pi,\mathcal{A}}$ in Figure 1. Simulatability stipulates the existence of a two part simulator, $\mathcal{S} = \{\mathsf{SimSetUp}, \mathsf{Sim}\}$. The simulated setup algorithm $\mathsf{SimSetUp}$ takes as input the security parameter and the relation $\mathcal{R}$ and outputs a trapdoor τ along with public parameters that are indistinguishable from those output by SetUp. The simulated party algorithm Sim_τ is a stateful party simulator which uses the trapdoor τ to output messages that are indistinguishable from those generated by real parties with knowledge of witnesses. Simulatability then requires that no efficient adversary can distinguish between access to the simulated parameters and parties with $\mathsf{Send}^1_{pp_1,\tau}$ from access to real parties with $\mathsf{Send}^0_{pp_0}$. The adversary is permitted to use the $\mathsf{SetKeys}$ oracle to decide the statements and witnesses used for authentication.

Definition 2 (Simulatability). *Consider the experiment* $\mathsf{Exp}^{\mathsf{WAKE\text{-}sim}}_{\Pi,\mathcal{A}}$ *in Figure 1,* $\mathsf{Exp}^{\mathsf{WAKE\text{-}sim}}_{\Pi,\mathcal{A}}$*. A* WAKE *protocol* Π *is simulatable if there exist efficient algorithms* $(\mathsf{SimSetUp}, \mathsf{Sim})$ *(the latter stateful) such that for all relations* $\mathcal{R}$ *and for all non-uniform PPT* $\mathcal{A}$*:* $|2 \cdot \mathsf{Pr}[\mathbf{Exp}^{\mathsf{WAKE\text{-}sim}}_{\Pi,\mathcal{A}}(\lambda, \mathcal{R})] - 1| \leq \mathsf{negl}(\lambda)$.

Authenticity is the requirement that an unauthenticated participant cannot convince another participant to accept. In the related experiment seen in Figure 1, the adversary is given access to Send and Reveal with the goal of authenticating to at least one instance of one participant. Authenticity guarantees that if the adversary can authenticate then either she was forwarding (and therefore inadmissible) or she had knowledge of a witness. The adversary can be said to know a witness if there exists an efficient extractor that can output a witness from that adversary. The adversary then wins the experiment if the extractor fails.

We define two variants of authenticity that differ exclusively in the style of extraction: straightline-black-box versus non-black-box extraction. A straightline-black-box extractor ($\mathsf{sl\text{-}BB}$) has access to the adversary's transcript along with the simulation trapdoor τ and is *black-box* in the adversary. The $\mathsf{sl\text{-}BB}$ extractor is additionally restricted from employing a common rewinding technique in which, while interacting with the adversary, the extractor reverts the adversary to a previous state after receiving a response to a challenge message and can then extract if the adversary outputs a distinct response to the same challenge. Thus the $\mathsf{sl\text{-}BB}$ extractor is limited to *straightline* extraction techniques. The $\mathsf{sl\text{-}BB}$ extractor does not depend on the adversary and therefore a single extractor can extract from *any* successfully authenticating adversary. A protocol admitting such an extractor satisfies our *stronger* notion of authenticity which is ultimately proven, in conjunction with our other WAKE security properties, to be equivalent to UC security in Theorem 1.

The strength of straightline black-box extraction comes at a price. In order for a protocol to admit such an extractor, producing witnesses from transcripts alone, the protocol messages must somehow encode witnesses for extraction. This

seemingly puts a lower bound on message length where message length must depend on the encoded witness length. This motivates a non-black-box (nBB) variant of extraction: an nBB extractor outputs a witness from the *entire view* of the adversary. This correspondingly weakens our authenticity requirement, in part because the nBB extractor is adversarially dependent. Fortunately such an extractor ultimately permits efficiency as discussed further in Section 4.

In summary our authenticity definition provides the following tradeoff: the stronger sl-BB authenticity provides equivalence to UC security while the weaker nBB authenticity admits practical protocols. The authenticity experiments are $\mathsf{Exp}^{\mathsf{WAKE\text{-}\star\text{-}auth}}_{\Pi,\mathcal{A},\mathcal{E}_\star}$ in Figure 1, and are color-coded by variant.

Definition 3 (Authenticity). *Consider the experiments* $\mathsf{Exp}^{\mathsf{WAKE\text{-}\star\text{-}auth}}_{\Pi,\mathcal{A},\mathcal{E}_\star}$ *in Figure 1. A* WAKE *protocol* Π *is* nBB*-witness-authenticated if for all admissible non-uniform PPT* $\mathcal{A}$ *there exists a PPT extractor* $\mathcal{E}_\mathcal{A}$*, such that for all relations* $\mathcal{R}$*, for all statement vectors* Φ *and for all witness distributions* $\mathcal{D}_\Phi$*:*

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

A WAKE *protocol* Π *is* sl-BB*-witness-authenticated if there exists a straightline PPT extractor* $\mathcal{E}$ *such that for all admissible non-uniform PPT* $\mathcal{A}$*, for all relations* $\mathcal{R}$*, for all statement vectors* Φ *and for all witness distributions* $\mathcal{D}_\Phi$*:*

$$\mathsf{Pr}[\mathsf{Exp}^{\mathsf{WAKE\text{-}sl\text{-}BB\text{-}auth}}_{\Pi,\mathcal{A},\mathcal{E}}(\lambda,\mathcal{R},\Phi,\mathcal{D}_\Phi)] \leq \mathsf{negl}(\lambda)$$

A WAKE is called **passively secure** if it satisfies confidentiality. We observe that the confidentiality-only security requirement corresponds to passive security in classical group key exchange, though with different syntax. A WAKE protocol achieves **full security** if it is passively secure and additionally satisfies authenticity and simulatability. A WAKE protocol satisfying nBB-extraction is also specified as nBB-WAKE-secure, whereas a protocol satisfying sl-BB-extraction is sl-bb-WAKE-secure. As a sl-BB extractor is also a nBB extractor we note that any sl-bb-WAKE-secure protocol Π is also nBB-WAKE-secure.

We remark that unilateral authentication in the two party case, or a WAKE generalization where an arbitrary subset of the participants are *unauthenticated* in the group case, is achievable if that subset authenticates with respect to trivial or empty statements.[9] This is further explored in Appendix C.

Universally Composable WAKE. We provide a UC definition in addition to our game-based one. In the UC framework a cryptographic task is defined with respect to an ideal functionality $\mathcal{F}$, a trusted party that behaves ideally with access to every secret. Proving that a protocol Π is indistinguishable from the ideal functionality then guarantees that the protocol is *just as good* as the ideal functionality and that the protocol is resilient against arbitrary adversarial attacks in arbitrary environments. Such a proof of UC-security then ensures that our WAKE will remain secure when composed with any protocol making

[9] This can also be seen as an adaptation of Unilaterally Authenticated Key Exchange [24] to the witness-based setting.

use of the generated keys, for example symmetric encryption, as is ideal. In the case of WAKE parties provide the ideal functionality $\mathcal{F}_{\mathsf{WAKE}}$ with their witnesses and the functionality verifies the witnesses and issues keys accordingly. The ideal functionality also models ideal leakage and adversarial influence. The UC framework is discussed in detail in Appendix B.1.

In fact, as mentioned above, UC-security is equivalent to our game-based definition of sl-bb-WAKE-security. Theorem 1 formalizes this equivalence by stating that any sl-bb-WAKE protocol Ω can be used to UC-realize the WAKE ideal functionality $\mathcal{F}_{\mathsf{WAKE}}$ directly. The simplicity of realizing UC security is the point; the code for parties executing the UC secure Π_Ω is the same as that for parties executing Ω, except inputs are received from and outputs are forwarded to "the environment". The proof, along with a presentation of the ideal functionality and a complete discussion of WAKE in the UC framework, can be found in Appendix D.

Theorem 1. *Let Ω be a fully secure group* sl-bb-WAKE *protocol. Then there exists a simple protocol Π_Ω which UC-realizes $\mathcal{F}_{\mathsf{WAKE}}$ in the presence of static, malicious adversaries without the assumption of authenticated channels.*

3 A General Compiler to Witness-Authentication

We describe a compiler that transforms any passively-secure key exchange, i.e. a WAKE satisfying confidentiality-only, into a fully-secure witness-authenticated protocol. Our compiler adapts the compiler presented in [31] to and can be applied to arbitrary passively-secure key exchange protocols. Given a passively secure key exchange protocol Π between set of parties $\mathcal{P}$ and a strongly simulation-extractable signature of knowledge Σ the compiled protocol Π^* satisfying full WAKE security is presented in Figure 2.

To transform Π to a witness-authenticated protocol with respect to relation $\mathcal{R}$ with statement set Φ, each party P_i on input (ϕ_i, w_i) uses the passively secure protocol Π as a black box. The compilation is at the expense of an additional round in which participants sample and exchange random nonces. After the preliminary round parties use these nonces as the session identifier and proceed according to Π with two additional steps: (1) each message to be sent according to Π by party P is first concatenated to the session identifier and signed with the signature of knowledge under ϕ_P, and (2) all message-signature pairs are verified according to $(\Phi, \mathcal{R})$ upon receipt.

Theorem 2. *If Π is a passively secure key exchange, satisfying confidentiality only, and Σ is a strongly simulation-extractable signature of knowledge then the resulting protocol Π^* obtained from applying the compiler on Π is a fully secure* nBB-WAKE *protocol.*

The confidentiality of Π^* is inherited from the underlying protocol Π, simulatability is reducible to the simulatability of Σ, and authenticity is reducible

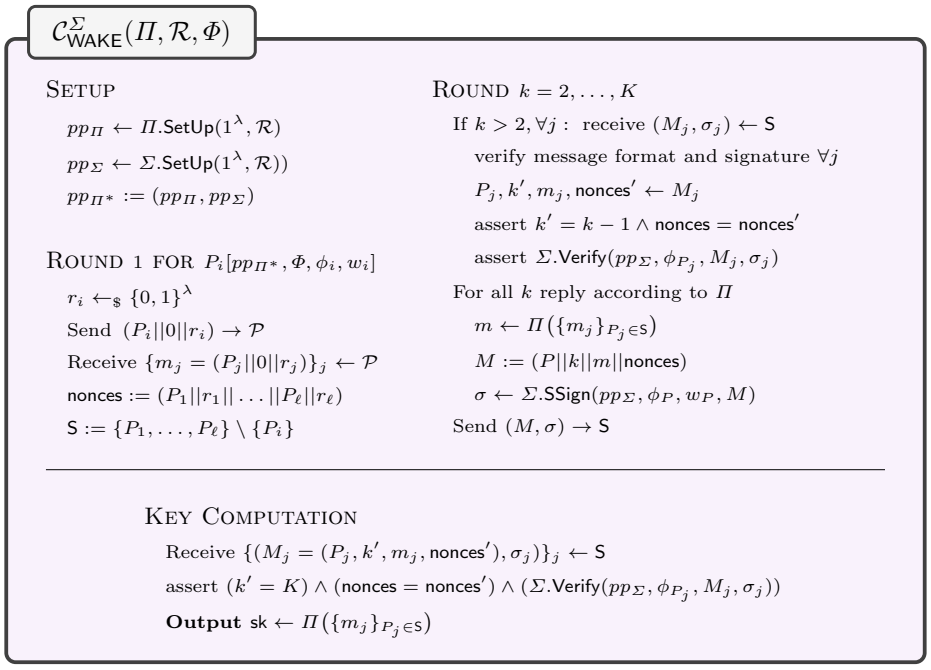

Fig. 2. A compiler from passively secure key exchange to full WAKE security.

to the simulation-extractability of Σ. A full proof of the theorem appears in Appendix E.

Theorem 8 directly implies a three round group WAKE protocol as the image of our compiler applied to Π_{BDKE}, the two round passively secure Burmester-Desmedt group key exchange protocol [16]. This is discussed further in Appendix F.

4 Two-Round Group WAKE

The two round protocol $\Pi_{\text{2-WAKE}}$ in Figure 3 UC-realizes $\mathcal{F}_{\mathsf{WAKE}}$. This two round protocol $\Pi_{\text{2-WAKE}}$ directly achieves **session authenticity**, the guarantee that senders are consistent throughout the exchange. Security is stated in Theorem 3 and proven in Appendix G.

The protocol below is optimized when compared to straightforwardly applying either the split functionality transformation [10] to achieve UC security or our compiler from Section 3 to achieve game-based security. The split functionality transformation, discussed at length in Appendix D, is the standard method of achieving UC-secure authenticated key exchange and adds *two rounds* to any passively secure protocol. This transformation introduces session authenticity and removes the assumption of authenticated channels in the UC framework. Our compiler achieves nBB-WAKE by adding a single round to any passively secure protocol.

Most notably the second round of protocol $\Pi_{\mathsf{2\text{-}WAKE}}$ requires each participant to *simultaneously authenticate with respect to their statement and their first round message*; party P_i signs their second message m_i with a signature of knowledge under statement ϕ_i and the ephemeral Diffie-Hellman value z_i sent in the first round: $\phi_i' = (\phi_i, z_i)$ with witness (w_i, x_i). The signature of knowledge is initialized with relation $\mathcal{R}' = \mathcal{R} \times \mathcal{R}_{\mathsf{DLP}}$ at setup, where $\mathcal{R}_{\mathsf{DLP}}^{(\mathbb{G},g)} := \{(h = g^x, x) | h \in \mathbb{G}, x \in \mathsf{ord}(h)\}$ is the discrete logarithm relation. Note that indices are taken modulo the number of parties n.

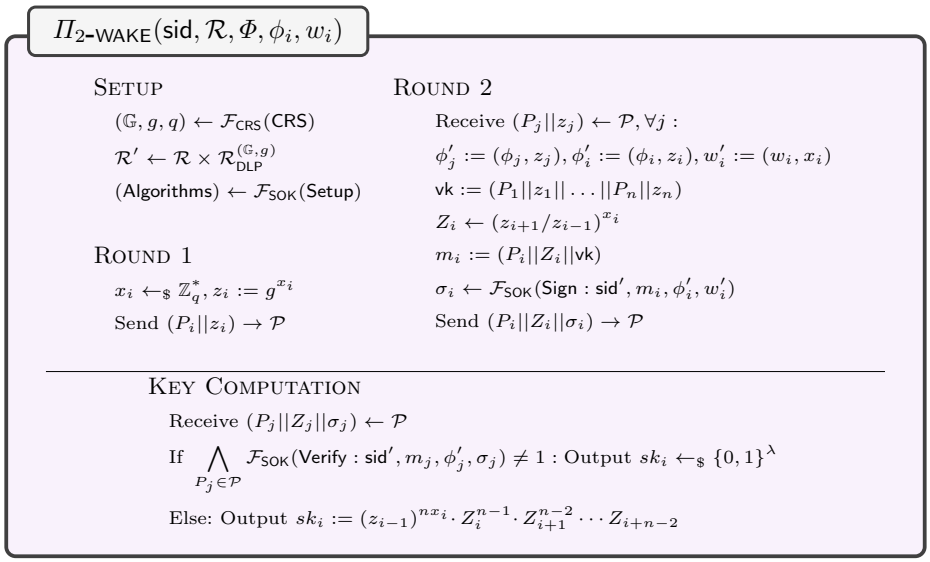

Fig. 3. A two-round WAKE protocol.

Theorem 3. *For any NP relation $\mathcal{R}$ protocol $\Pi_{\mathsf{2\text{-}WAKE}}$ UC-realizes $\mathcal{F}_{\mathsf{WAKE}}$ in the $(\mathcal{F}_{\mathsf{CRS}}, \mathcal{F}_{\mathsf{SOK}})$-hybrid model against malicious adaptive adversaries in the unauthenticated, asynchronous setting.*

The protocol can be instantiated without any reference to $\mathcal{F}_{\mathsf{CRS}}$ at the expense of one round in which some party generates and broadcasts the group parameters as $(\mathbb{G}, g, q)$ prior to the exchange. When our protocol is instantiated with a *succinct* simulation-extractable signature of knowledge it is possible to achieve efficiency at the expense of weakening security to nBB-WAKE from sl-bb-WAKE, and thereby sacrificing provable security in the UC setting.

Estimated benchmarks for the above protocol are presented in Table 2. Our *communication complexity* is constant and estimated to be below 0.5 KB in total for the unilateral two-party case and of approximately N KB for the group authenticated case with N parties. When using BLS12-381 [1] as the concrete

Setting	Rel $\mathcal{R} \in \mathsf{NP}$	Stmnt ϕ	Wtns w	T(s)
Dark Pools	$\bar{c} = \mathsf{Comm}(\mathbf{s}; \rho) \wedge \mathbf{s} \geq \bar{\mathbf{B}}$	$\bar{c}, \bar{\mathbf{B}}$	$(\rho, \mathbf{s})$	4
IPFS	$\bar{h} = \mathsf{blake3hash}(\mathbf{F})$	$\bar{h}$	$\mathbf{F}$	68
ZKCP	$\mathsf{solvesSudoku}(\mathbf{sol}, \overline{pzl})$	$\overline{pzl}$	**sol**	1
Bug bounty	$C_{\text{buggy}}(\mathbf{bug}) \wedge \neg C_{\text{expect}}(\mathbf{bug})$	$C_{\{\text{buggy,exp}\}}$	**bug**	58

Table 2. Run time estimation (in seconds, rounded through ceiling). The *authenticated party running time* can be found in the last column. The timings refer to our protocol instantiated through *Snarky Signature* [30], a succinct SOK variant of [30]. See also Appendix I for more on the experimental setting.

curve a signature is 224 bytes.[10] We remark that an additional *offline-online* optimization of our protocol can be applied to migrate a majority of the online running time, i.e. the signature, to an offline phase. This example is further discussed in the unilaterally-authenticated two-party case in Appendix H.

We briefly describe detail one more application scenario we benchmarks in our table, that of *Zero-Knowledge Contingent Payment (ZKCP)*, where a seller wants to initiate a channel with any party claiming to have a digital good that satisfies a certain property to negotiate a price for that good prior to a ZKCP protocol [5, 18]. We benchmark the ZKCP case of payments for Sudoku Solutions (also used in prior work [5, 18] and for the case of bug bounties. In the latter, the software producer of a (potentially buggy) program C_{buggy} can incentivize users to find bugs **bug** in it. These can be checked through additional program C_{expect}, guaranteeing an expected condition for an input the program accepts (which will be violated by the bug, a false positive). [11]

References

1. Bls12-381 curve. `https://electriccoin.co/blog/new-snark-curve/`.
2. Bug in primes testing in swiss post voting system. `https://gitlab.com/swisspost-evoting/crypto-primitives/crypto-primitives/-/issues/13`.
3. Cweb3. `https://www.npmjs.com/package/cweb3`.
4. Ethereum Name Service. `https://ens.domains/`.
5. Greg maxwell's zero knowledge contingent payment (bitcoin wiki). `https://en.bitcoin.it/wiki/Zero_Knowledge_Contingent_Payment`.
6. Handshake: Decentralized naming and certificate authority. `https://handshake.org`.
7. MeetWallet: The Meet JS SDK Library for MEET.ONE Client. `https://meet-common.gitlab.io/fe/meet-js-sdk/classes/meetwallet.html`.

[10] More specifically, 192 bytes for the group elements and 32 bytes for the SHA256.

[11] 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 $C_{\text{expect}}(z) :=$ "z is odd $\vee$ $z = 2$". Anything not satisfying the latter would be a false positive. This is not just a toy setting but it is relevant in real world systems [2].

8. D. F. Aranha, E. Pagnin, and F. Rodríguez-Henríquez. Love a pairing. In *International Conference on Cryptology and Information Security in Latin America*, pages 320–340. Springer, 2021.
9. J.-P. Aumasson, J. O'Connor, S. Neves, and Zooko. Blake3 hash. `https://github.com/BLAKE3-team/BLAKE3`.
10. B. Barak, R. Canetti, Y. Lindell, R. Pass, and T. Rabin. Secure computation without authentication. *Journal of Cryptology*, 24(4):720–760, Oct. 2011.
11. O. Barta, Y. Ishai, R. Ostrovsky, and D. J. Wu. On succinct arguments and witness encryption from groups. In D. Micciancio and T. Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 776–806. Springer, Heidelberg, Aug. 2020.
12. S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *1992 IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, May 1992.
13. F. Ben Hamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. Efficient UC-secure authenticated key-exchange for algebraic languages. In K. Kurosawa and G. Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 272–291. Springer, Heidelberg, Feb. / Mar. 2013.
14. E. Birrell, K.-M. Chung, R. Pass, and S. Telang. Randomness-dependent message security. In A. Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 700–720. Springer, Heidelberg, Mar. 2013.
15. N. Bitansky, A. Chiesa, Y. Ishai, R. Ostrovsky, and O. Paneth. Succinct non-interactive arguments via linear interactive proofs. In A. Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 315–333. Springer, Heidelberg, Mar. 2013.
16. M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system (extended abstract). In A. D. Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 275–286. Springer, Heidelberg, May 1995.
17. J. Camenisch, N. Casati, T. Groß, and V. Shoup. Credential authenticated identification and key exchange. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 255–276. Springer, Heidelberg, Aug. 2010.
18. M. Campanelli, R. Gennaro, S. Goldfeder, and L. Nizzardo. Zero-knowledge contingent payments revisited: Attacks and payments for services. In B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, editors, *ACM CCS 2017*, pages 229–243. ACM Press, Oct. / Nov. 2017.
19. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. `https://eprint.iacr.org/2000/067`.
20. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, Oct. 2001.
21. M. Chase and A. Lysyanskaya. On signatures of knowledge. In C. Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96. Springer, Heidelberg, Aug. 2006.
22. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 13–25. Springer, Heidelberg, Aug. 1998.
23. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
24. Y. Dodis and D. Fiore. Unilaterally-authenticated key exchange. In A. Kiayias, editor, *FC 2017*, volume 10322 of *LNCS*, pages 542–560. Springer, Heidelberg, Apr. 2017.

25. P.-A. Dupont, J. Hesse, D. Pointcheval, L. Reyzin, and S. Yakoubov. Fuzzy password-authenticated key exchange. In J. B. Nielsen and V. Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 393–424. Springer, Heidelberg, Apr. / May 2018.
26. C. Ganesh, H. Khoshakhlagh, M. Kohlweiss, A. Nitulescu, and M. Zajac. What makes fiat–shamir zksnarks (updatable srs) simulation extractable? Cryptology ePrint Archive, Report 2021/511, 2021. `https://ia.cr/2021/511`.
27. S. Garg, C. Gentry, A. Sahai, and B. Waters. Witness encryption and its applications. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *45th ACM STOC*, pages 467–476. ACM Press, June 2013.
28. J. Groth. On the size of pairing-based non-interactive arguments. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.
29. J. Groth, M. Kohlweiss, M. Maller, S. Meiklejohn, and I. Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Heidelberg, Aug. 2018.
30. J. Groth and M. Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 581–612. Springer, Heidelberg, Aug. 2017.
31. J. Katz and M. Yung. Scalable protocols for authenticated group key exchange. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 110–125. Springer, Heidelberg, Aug. 2003.
32. V. Kolesnikov, H. Krawczyk, Y. Lindell, A. J. Malozemoff, and T. Rabin. Attribute-based key exchange with general policies. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *ACM CCS 2016*, pages 1451–1463. ACM Press, Oct. 2016.
33. C. N. Ngo, F. Massacci, F. Kerschbaum, and J. 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. `https://fc21.ifca.ai/papers/113.pdf`.
34. Protocol Labs. Filecoin. `https://filecoin.io/`.
35. Protocol Labs. IPFS: Interplanetary file system. `https://ipfs.io`.
36. R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, 1978.
37. A. Sahai and B. R. Waters. Fuzzy identity-based encryption. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Heidelberg, May 2005.
38. A. Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and D. Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53. Springer, Heidelberg, Aug. 1984.

Supplementary Material

A More on Prior Work: Witness-Key-Agreement (WKA)

This appendix discusses the work of Ngo, Massacci, Kerschbaum and Williams [33], which proposed the first model, definition and construction of a witness-authenticated key exchange as witness key agreement (WKA). Three main limiations are identified in the work of NMKW, namely: (1) the formalized model is not sufficient for the desired application, (2) the construction is impractical in that it requires a trusted authority (roughly) for each key agreement, and (3) the security proof of the construction is flawed.

Model and Definitions: A general requirement for key agreement is that the transcript should not leak any information about the session key. This property is usually modeled as *indistinguishability* requiring that an eavesdropping (passive) adversary cannot efficiently distinguish the real session key from a random key. Indistinguishability, while not always required, is used to argue that exchanging messages encrypted under the shared key is equivalent to sending those messages over a secure channel.

The definition of NMKW (see Page 7 of [33]) weakens this security requirement on the session key. The set of properties, as defined, only requires indistinguishability against passive adversaries. The weaker property of unpredictability is required against active adversaries. Property (4) only requires that an adversary cannot compute the key rather than that an adversary cannot distinguish the key from a random one. Notice that Property (4) itself does not ameliorate this because it concerns a leakage of the witness, not of the key. As a consequence this model may consider a construction where an active adversary is able to learn 80% of the bits in the session key as secure. This is clearly unacceptable.

Another important security consideration is *forward security*, the case when long term secrets are compromised after the exchange is initiated. In the case of witness-authentication this consideration is fundamental, in that there is the inherent possibility of an offline attack in which the adversary recovers a witness that can be verified independently. The definition of NMKW fails to address the case where the adversary has a valid witness for one of the relevant statements, in which security should be maintained.

Finally, a key exchange should maintain security against an active adversary. Specifically the adversary can initiate additional sessions with any party in the exchange and can inject arbitrary messages into the transcript. NMKW considers an adversary that is restricted from these behaviors. Property (5) provides the adversary with a single session that can be used against the challenge session and, along with Property (4), still does not permit adversarially chosen messages and challenges.

Constructions: It is unclear if the NMKW construction is secure. While we have not identified a concrete attack against the construction, it is unclear if

the latter is provably secure, given one major flaw in the security proof in the original paper.

On a high level, the construction and proof use a variation on the techniques of [15] where secret points in the setup are encrypted with a limited-malleability encryption scheme. The original proof in [15] relies on IND-CPA for security. The NMKW construction diverges from [15] through the addition of encryptions of the randomness used to encrypt the other ciphertexts, but still uses IND-CPA which is no longer sufficient due to the extension. The modification introduces additional leakage and plausibly requires a stronger encryption scheme: one with randomness-dependent message security [14]. Neither the theorem statement nor the proof explicitly acknowledge this fact.

Here the issue is explained in more detail, but is still simplified. The NMKW construction is roughly an adapted designated-verifier proof system [15]. The scheme uses a Linear-Only Encryption scheme (i.e. a scheme with limited malleability) to encrypt messages $\mathsf{ct} = \mathsf{Enc}(pk, m; r)$ that are then included in the CRS. To ensure soundness, the prover must not learn the corresponding plaintexts. This is common both in [15] and [33]. Where the two constructions diverge is that NMKW also extends the CRS with public encryptions of the randomness r used to generate ct, that is it includes $\mathsf{ct}' = \mathsf{Enc}(pk, r; \rho)$ for some randomness ρ.

The proof of security in [15] invokes the IND-CPA of these ciphertexts to argue soundness of the system. This IND-CPA is no longer sufficient for security in the case that encryption of the randomness is also provided. A stronger variant of IND-CPA which accounts for randomness-dependent message (RDM) security is required [14]. Yet, the NMKW proof sketch (page 25, "adaptive knowledge soundness") only informally invokes the standard notion of IND-CPA and neglects to mention any requirements for RDM security anywhere.

It is unclear whether the proof can be mended. While it is possible that invoking and appropriately using RDM encryption could patch the proof, this requirement may still not be sufficient to obtain a concrete construction. It is unknown (to the writer) if there exist encryption schemes which are both Linear-Only and RDM secure, or if such schemes are even possible.

It is also unclear if the NMKW construction satisfies the stronger game-based security requirements provided in Section 2. It is unclear if the construction from [33] is secure in their own model. Due to the use of frequent setups, the complexity of the construction, the additional requirement of RDM security and the lack of detail in the existing proof, we do not attempt to prove the NMKW construction secure with respect to our game-based model for WAKE.

Frequent Trusted Setup: All of the constructions in this work employ publicly-verifiable signatures of knowledge. The construction provided by NMKW also relies on NIZK through the use of designated-verifier SNARKs.

One primary limitation of NMKW is the requirement of a trusted setup *each time* a party initiates a key exchange.[12] The construction implicitly requires

[12] Although, if the same parties want to run multiple key agreements on the same relation, it could potentially reuse the setup.

a designated-verifier SNARK (e.g., a designated verifier version of [28]). Every time a party initiates a key exchange a trusted authority must be invoked that provides a secret (tantamount to a verification key in a designated-verifier SNARK) which is then used in the key exchange. Thus, at least one trusted party must be invoked per party in the system, and even more if the parties intend to exchange keys with respect to different relations.On the other hand, the constructions provided herein use the same parameters generated once and for all per relation.[13]

A key exchange requiring frequent trusted setups is highly impractical. First, parties interested in exchanging keys would need to engage a trusted third party. This party would need to be available, adding complexity and overhead to the system[14]. Additionally, it is undesirable to have a centralized entity acting as such trusted authority. Whoever produces the trusted setup knows (and may inadvertently leak) trapdoors allowing the system to be completely compromised. In order to mitigate this problem one can distribute the setup through a large scale MPC ceremony. It is not plausible to perform such distributed setups each time a new key generation is executed, as in [33].

B Expanded Preliminaries

B.1 Universal Composition

A minimal and simplified description of the UC model is contained within this section. The curious reader is referred to the detailed and regularly updated manuscript by Canetti [19]. A majority of this review section is adapted from that work by Canetti.

Game-based definitions of a cryptographic primitive can explicitly capture most security requirements. However there is always the possibility that some necessary condition is absent from even the best cryptographer's intuition. This motivates the question: *how can we be certain that a witness-authenticated key exchange satisfying our game-based definition is actually secure? Did we think of everything?* Universal composability (UC) can eliminate these concerns [19].

Systems of Interactive Turing Machines In the universal composability (UC) model all protocols, adversaries and parties are modeled as interactive turning machines (ITM). An ITM is a turing machine augmented with special tapes and instructions used to facilitate interactivity between ITMs in a system. ITMs can interact through these shared tapes via the special instructions. An ITM M has the following special tapes:

[13] Once if our constructions employ signatures of knowledge with universal setup obtainable from [26].

[14] A trusted setup can alternatively generated through a large scale multiparty computation. They have been performed before, but they can practically be carried out only occasionally since they are expensive, require hours of computations, days of logistical coordination and plenty of resources/money. Such distributed ceremonies have been before, e.g., for Zcash and Filecoin. See `https://z.cash/technology/paramgen` and `https://filecoin.io/blog/posts/update-trusted-setup`.

- identity tape: a read-only tape containing the *extended identity* of the ITM as $(\mathsf{code}_M, \mathsf{id})$ consisting of a canonical representation of the code describing the behavior of the ITM (specifically the state transition function and initial tape contents) and an identity string used to uniquely identify the ITM within the system
- outgoing-message tape: this tape is written to by M with any outgoing messages along with addressing information for delivery
- input tape: this tape is externally writable and is written to by the *caller* of M
- subroutine-output tape: this tape is externally writable and is written to by any *subroutines* of M ie any ITMs called by M
- backdoor tape: this tape is externally writable, is written to by the adversary ITM and allows for adversarial influence on the system
- activation tape: this tape contains a single bit indicating if M is currently executing

And two special instructions:

- external-write instruction: indicates that M intends to write the message m on its outgoing-message tape to some other ITM in the system
- read-next-message instruction: this instruction moves the reading head to the beginning of the next message on the tape specified

A configuration of an ITM M is the contents on all the tapes, its current state and the location of the head. An instance of an ITM M, called an ITI, is a run-time object consisting of the immutable contents of the identity tape. An activation of ITI $M = (\mathsf{code}_M, \mathsf{id})$ is a sequence of configurations corresponding to code_M, starting from when the bit on the activation tape is 1 and ending when that bit is 0.

A system of ITMs is a pair (I, C) with I the initial ITM and $C : \{0,1\}^* \to \{0,1\}$ the control function. The system can be executed on input z and random tape r via a sequence of activations, starting with the activation of the initial ITI I with identity $\mathsf{id} = 0$ as $(I, 0)$ and contents of the input tape set to z. The execution terminates when I halts and the output is the first message on I's outgoing message tape. After activation ITI I can invoke new ITIs with which to interact via the shared tapes and instructions.

The control function is responsible for determining if messages are allowed, transferring allowed messages and activating new ITIs according to external-write instructions. Outgoing messages m from ITI M are written by M to it's outgoing-message tape as the external-write instruction:

$$(f, M', t, r, M, m)$$

This instruction indicates the ITI with extended identity M intends to send the message m to the ITI with extended identity M' on tape t. The component f is the forced-write flag indicating if a new ITI should be activated in the case that the intended recipient does not exist within the system. The component r is the

reveal-sender-id flag, indicating if M is written to the receiver. In an extended system the control function is also permitted to modify the tuple in any way, in order to formalize UC emulation and security.

A protocol is defined as a single ITM which describes the program(s) to be run by each participant in the computation as π. The extended identity $(\mathsf{code}_P, \mathsf{id}_P)$ of an ITI P that is participating in a protocol session can be parsed as a session identifier and party identifier $\mathsf{id}_P = (\mathsf{sid}, \mathsf{pid}_P)$ with code_P set to π. Then the protocol session is defined as the set of all ITIs participating in the session sid and running the same code, namely that of the protocol π.

Resource Bounded Computation Security is argued via the presentation of an efficient transformation from one computation to another (called the simulator), and indistinguishability requires a bound on feasible computations. In order to formulate security the model must include some notion of efficiency and resource bounded computation. A probabilistic polynomial time ITM includes, in both incoming and outgoing messages, an import field. The sum of the incoming imports minus the sum of the outgoing imports n is the runtime-budget of the ITI. Each ITI M is required to be T-bounded in n, meaning that for some polynomial T the number of computational steps executed by M are bounded by $T(n)$. The imports can be thought of as runtime tokens which are distributed along with messages within the system. This resource bounded computation is consistent with the standard notions; for the system (I, C) if both I and C are PPT then a single non-interactive Turning machine μ which is PPT in the import to the system can simulate the entire execution.

An ITM is parameterized by security parameter λ if it does not start running unless the overall import is at least λ. A system of ITMs is parameterized with security parameter λ if all of the ITIs in the system are parameterized with λ and the import of the input to the initial ITI is at most polynomial in λ. The system is called M-balanced if, at any point during the execution the imports given to ITI M is at least the sum of the imports of all other inputs in the system except the initial ITI.

Execution and Emulation Protocol execution is modeled via a system of ITMs (I, C), where I corresponds to the environment machine $\mathcal{Z}$ and C encodes the adversary $\mathcal{A}$, the protocol π and any communication rules. Below is a simplified explanation of the model for protocol execution.

Given ITMs $\mathcal{A}$, $\mathcal{Z}$, Π, the model for protocol execution of a single instance of Π is the system of ITMs $(\mathcal{Z}, \mathcal{C}_{\mathsf{Exec}}^{\Pi,\mathcal{A}})$ with input z encoding an initial state of the environment and parameterized by security parameter λ. The control function $\mathcal{C}_{\mathsf{Exec}}^{\Pi,\mathcal{A}}$ enforces the following rules. The program of the first ITI invoked by $\mathcal{Z}$ is $\mathcal{A}$, and the subsequent programs are set to be Π. All of the ITIs invoked by $\mathcal{Z}$ in the system belong to the same protocol session so the session identifiers of all ITIs must be the same. The environment $\mathcal{Z}$ can only write to the input tapes of the main ITIs in the system. The adversary $\mathcal{A}$ can write to any backdoor tapes of the ITIs and cannot invoke new ITIs. All other ITIs P in the system can write

to the backdoor tape of $\mathcal{A}$, the input and output tapes of subroutines and calling ITIs and, if P is in the protocol session, can write to the subroutine output tape of $\mathcal{Z}$.

As an example consider a system $(\mathcal{Z}, C_{\mathsf{Exec}}^{\Pi,\mathcal{A}})$ executing a protocol session Π with session identifier sid between set of parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ on inputs $\mathbf{z} = (z_1, \ldots, z_n)$ with adversary $\mathcal{A}$. The initial ITI $\mathcal{Z}$ with input $\mathbf{z}$, random tape r and $\mathsf{id} = 0$ first activates the adversary $\mathcal{A} = (A, (\mathsf{sid}, \star))$ and then activates the main parties $P_i = (\Pi, (\mathsf{sid}, i))$ via $\mathsf{external\text{-}write}$ instructions: $(1, P_i, \mathsf{input}, 0, \mathcal{Z}, z_i)$. Parties invoke subroutines, send messages and return outputs via $\mathsf{external\text{-}write}$ instructions according to Π on their $\mathsf{outgoing\text{-}message}$ tapes. Parties can not communicate to each other directly; messages permitted by $C_{\mathsf{Exec}}^{\Pi,\mathcal{A}}$ are sent *through* and scheduled by $\mathcal{A}$. Certain subroutine ITIs, for example the ideal functionality for a CRS, are shared and can write to the $\mathsf{subroutine\text{-}output}$ tape of any parties executing Π. The parties terminate and output to the $\mathsf{subroutine\text{-}output}$ tape of $\mathcal{Z}$. Eventually the initial ITI $\mathcal{Z}$ terminates with output written on the $\mathsf{outgoing\text{-}message}$ tape.

Define $\mathsf{Exec}_{\Pi,\mathcal{A},\mathcal{Z}}(\lambda, z)$ to be the binary output of environment $\mathcal{Z}$ when interacting with adversary $\mathcal{A}$ and executing protocol Π with security parameter λ and input z. Consider the binary distribution ensemble $\mathsf{Exec}_{\Pi,\mathcal{A},\mathcal{Z}} := \{\mathsf{Exec}_{\Pi,\mathcal{A},\mathcal{Z}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*}$ restricted to the case that the import of z is polynomial in λ.

Definition 4 (UC-Emulation). *Let Π and Γ be PPT protocols. The protocol Π UC-emulates Γ if, for any PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{S}$ such that for any $\mathcal{A}$-balanced PPT environment $\mathcal{Z}$:*

$$\mathsf{Exec}_{\Gamma,\mathcal{S},\mathcal{Z}} \approx \mathsf{Exec}_{\Pi,\mathcal{A},\mathcal{Z}}$$

Informally, the protocol Π emulates Γ if for every adversary $\mathcal{A}$ there is a simulator $\mathcal{S}$ such that for all PPT environments $\mathcal{Z}$, the environment cannot distinguish an execution involving $(\Pi, \mathcal{A})$ from an execution involving $(\Gamma, \mathcal{S})$. This property is transitive, meaning that if Π_1 emulates Π_2 and Π_2 emulates Π_3 then Π_1 emulates Π_3.

Security and Composition In order to capture the desired functionality of a protocol meant to achieve some task, one must specify an ideal process for carrying out that task. This is done by defining an $\mathsf{ideal\ functionality}$ as is a single *trusted party* that works to accomplish the task in an idealized way. The ITM ideal functionality $\mathcal{F}$ then participates in a session of the $\mathsf{ideal\ protocol}$ for the task: $\mathsf{Ideal}_{\mathcal{F}}$. The code for dummy party P_i executing the ideal protocol $\mathsf{Ideal}_{\mathcal{F}}$ for the ideal functionality $\mathcal{F}$ with session identifier sid is (simplified) as follows:

- Upon activation with input $(z, \mathsf{eid}_{\mathsf{caller}}, (\mathsf{sid}, i))$ forward $(z, \mathsf{eid}_{\mathsf{caller}})$ to $\mathcal{F}$ with the $\mathsf{forced\text{-}write}$ and $\mathsf{reveal\text{-}sender\text{-}id}$ flags set.
- Upon activation with subroutine output $(z, (\mathsf{sid}, i), \mathsf{eid}_{\mathsf{caller}})$ pass output z to $\mathsf{eid}_{\mathsf{caller}}$ with the $\mathsf{forced\text{-}write}$ and $\mathsf{reveal\text{-}sender\text{-}id}$ flags set.

– Ignore all messages written on the backdoor tape as the adversary gives corruption instructions directly to the ideal functionality.

An execution of the ideal protocol $\mathsf{Ideal}_{\mathcal{F}}$ by system of ITMs $(\mathcal{Z}, \mathcal{C})$ consist of main parties $P_i = (\mathsf{Ideal}_{\mathcal{F}}, (\mathsf{sid}, i))$ all sharing $\mathcal{F}$ as a subroutine. The main ITIs, upon activation by $\mathcal{Z}$ with input z, merely format and forward this input to the ideal functionality $\mathcal{F}$ and wait for subroutine output from $\mathcal{F}$. This output is then written to the subroutine-output tape of $\mathcal{Z}$.

A protocol Π is called subroutine respecting if for each session the ITIs in that session do not accept or pass input or subroutine output to ITIs that are not in the session.

Definition 5 (UC-Realization). *Let $\mathcal{F}$ be an ideal functionality and Π be a protocol. The protocol Π is said to UC-realize $\mathcal{F}$ if Π is subroutine respecting and Π UC-emulates $\mathsf{Ideal}_{\mathcal{F}}$, the ideal protocol for $\mathcal{F}$.*

UC-emulation is essentially possible through the control function. The control function (1) enforces the communication structure within the system which is required for $\mathcal{Z}$ to remain unaware of the presence of $\mathcal{F}$ and (2) modifies the external-write instructions to account for differences in the extended identities of the ITIs executing Π versus $\mathsf{Ideal}_{\mathcal{F}}$. Then the environment $\mathcal{Z}$ remains unaware of the actual protocol under execution.

If protocol Π UC-emulates $\mathsf{Ideal}_{\mathcal{F}}$ then Π UC-realizes $\mathcal{F}$ and it can be said that Π is *at least as secure* as the ideal functionality. This is relatively intuitive, if the environment colluding with adversary $\mathcal{A}$ cannot distinguish if it is in the system $(\mathcal{Z}, \mathcal{C}_{\mathsf{Exec}}^{\Pi,\mathcal{A}})$ with ITMs $(\Pi, \mathcal{A})$ or in the system $(\mathcal{Z}, \mathcal{C}_{\mathsf{Exec}}^{\mathsf{Ideal}_{\mathcal{F}},\mathcal{S}})$ with ITMs $(\mathsf{Ideal}_{\mathcal{F}}, \mathcal{S})$ then Π must not leak more information to the environment than $\mathsf{Ideal}_{\mathcal{F}}$ and must be "just as good" as $\mathsf{Ideal}_{\mathcal{F}}$ in every way detectable by $\mathcal{Z}$.

Let protocol Γ emulate protocol Π. If protocol Γ is "just as good" as protocol Π then every call to Π can be replaced by a call to Γ without problem. Consider protocol ρ making subroutine calls to protocol Π. Consider protocol $\rho^{\Gamma \to \Pi}$ which is exactly ρ except that every call to Π is replaced with a call to Γ. Formally, it is the case that $\rho^{\Gamma \to \Pi}$ emulates ρ.[15] This is the universal composition theorem, appearing in Theorem 4.

Theorem 4 (Universal Composition [20]). *Let Π, Γ be subroutine respecting protocols such that Π emulates Γ. Let ρ be a protocol, and $\rho^{\Gamma \to \Pi}$ be the protocol ρ with all calls to Π replaced by corresponding calls to Γ. Then protocol $\rho^{\Gamma \to \Pi}$ emulates ρ with UC security. If ρ realizes some ideal functionality $\mathcal{F}$, then so does $\rho^{\Gamma \to \Pi}$.*

Consider the ideal functionality $\mathcal{F}$ for cryptographic task x and the ideal protocol $\mathsf{Ideal}_{\mathcal{F}}$ in which main parties $\mathcal{P}$ engage to accomplish x. Ideal functionality $\mathcal{F}$ is a trusted third party ITM with code written to specify each desired requirement for task x, therefore protocol $\mathsf{Ideal}_{\mathcal{F}}$ is essentially ideal for achieving

[15] This applies only to subroutine respecting protocols for this particular formulation of UC security.

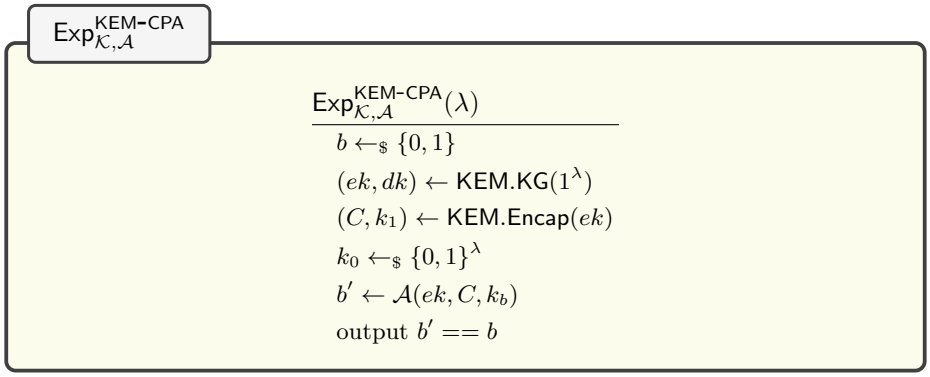

Fig. 4. The KEM chosen plaintext attack experiment.

task x. Consider protocol Π which UC-emulates $\mathsf{Ideal}_{\mathcal{F}}$, meaning that Π is "just as good" as the ideal protocol. Consider protocol ρ which invokes and executes multiple sessions of ideal protocol $\mathsf{Ideal}_{\mathcal{F}}$ as a subroutine along with sequential, parallel and concurrent executions of arbitrary other protocols. Universal composition guarantees that the protocol $\rho^{\Pi \to \mathsf{Ideal}_{\mathcal{F}}}$ is *at least as secure* as ρ.

Universal composition guarantees that protocols remain secure in any environment and context. A more detailed discussion and taxonomy of composition appears in [19].

B.2 Key Encapsulation Mechanism

A key encapsulation mechanism (KEM) can be thought of as a key exchange protocol between two parties in which each party sends a single message. Definition 6 provides the details.

Definition 6 (Key Encapsulation Mechanism (KEM)). *A key encapsulation mechanism is a triple of algorithms* $(\mathsf{KG}, \mathsf{Encap}, \mathsf{Decap})$ *with the following syntax:*

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

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

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

Correctness requires that for all (ek, dk) output by the key generation algorithm, $k_e = k_d$ for $\mathsf{Encap}(ek) \to (C, k_e)$ and $\mathsf{Decap}(dk, C) \to k_d$. Security against a chosen plaintext attack (KEM-CPA) requires that an efficient adversary cannot distinguish the real session key from a random one, as described in Definition 7.

Definition 7 (KEM-CPA). *A* KEM *protocol* $\mathcal{K}$ *is* KEM-CPA*-secure if for any PPT adversary* $\mathcal{A}$ *the advantage of* $\mathcal{A}$ *as defined is negligible in* λ*, where* $\mathsf{Exp}^{\mathsf{KEM\text{-}CPA}}_{\mathcal{K},\mathcal{A}}$ *is as in Figure 4:*

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

B.3 Signatures of Knowledge

Game-based security for signatures of knowledge. Towards security, a signature should not leak any information about the witness used during signing. This is captured by the existence of a polynomial time simulator composed of an additional two algorithms $\{\mathsf{SSimSetup}, \mathsf{SSimSign}\}$:

$\mathsf{SSimSetup}(1^\lambda, \mathcal{R}) \to (pp, \tau)$: randomized simulated setup takes as input relation $\mathcal{R}$ and security parameter λ and returns public parameters pp along with trapdoor τ.

$\mathsf{SSimSign}(pp, \tau, \phi, m) \to \sigma$: randomized simulated signing takes as input pp, trapdoor τ and instance ϕ and returns signature σ.

Simulatability then requires for all efficient adversaries $\mathcal{A}$ querying the (simulated) signing oracle with $(\phi, w, m) \in \mathcal{R} \times \mathcal{M}_\lambda$:

$$\left| \begin{array}{c} \Pr[\mathcal{A}^{\mathsf{SSimSign}_{pp,\tau}(\cdot,\cdot,\cdot)}(pp) : (pp, \tau) \leftarrow \mathsf{SSimSetup}(1^\lambda, \mathcal{R})] - \\ \Pr[\mathcal{A}^{\mathsf{SSign}_{pp}(\cdot,\cdot,\cdot)}(pp) : pp \leftarrow \mathsf{SSetup}(1^\lambda, \mathcal{R})] \end{array} \right| \leq \mathsf{negl}(\lambda)$$

Simulation extractability requires that an adversary cannot generate a new signature with respect to any statement ϕ without knowledge of a witness, which is modeled via the existence of an efficient extractor that can output a witness to ϕ from the view of any adversary outputting a verifying signature under ϕ. The adversary is granted access to a simulated signing oracle for statements in the language. The set $\mathcal{Q}$ is the set of queries to the signing oracle. This *strong* definition of simulation extraction also does not permit an adversary to create new signatures on a queried statement-message pair. Simulation extractability requires that the following value is bounded from below by $1 - \mathsf{negl}(\lambda)$.

$$\Pr\left[\begin{array}{c} (\phi, w) \in \mathcal{R} \vee (\phi, w, m) \in \mathcal{Q} \vee \mathsf{SVfy}(pp, \phi, m, \sigma) = 0 : w \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{view}_{\mathcal{A}}); \\ (\phi, m, \sigma) \leftarrow \mathcal{A}^{\mathsf{SSimSign}_{pp,\tau}(\cdot,\cdot,\cdot)}(pp); (pp, \tau) \leftarrow \mathsf{SSimSetup}(1^\lambda, \mathcal{R}) \end{array} \right]$$

Universally composable signatures of knowledge. Consider a language $\mathcal{L} \in \mathsf{NP}$ and polynomial time Turing machine $M_{\mathcal{L}}$ that computes the associated relation $\mathcal{R}_{\mathcal{L}}$ such that $\phi \in \mathcal{L}$ iff there exists a witness w such that $(|w| = p(|x|)) \wedge (M_{\mathcal{L}}(\phi, w) = 1)$.

The ideal functionality for a signature of knowledge $\mathcal{F}_{\mathsf{SOK}}$ is in Figure 5. The functionality as written in [21] was parameterized by $M_{\mathcal{L}}$ but the functionality provided in Figure 5 is instead parameterized by the relation $\mathcal{R}_{\mathcal{L}} = \mathcal{R}$. It is

required that the parameter $\mathcal{R}$ (alternatively the TM $M_{\mathcal{L}}$) be included in the session identifier, as it is mandatory that all participants agree on this value.

Upon receipt of the first SetUp query the functionality prompts the adversary $\mathcal{S}$ to supply algorithms $\{\mathsf{Verify}, \mathsf{Sign}, \mathsf{Simsign}, \mathsf{Extract}\}$ for the signature. The algorithms $\{\mathsf{Sign}, \mathsf{Verify}\}$ are then output to the querying party party, and each party in $\mathcal{P}$ upon receipt of a query from that party. Parties can compute signatures on messages m either through the signing interface of $\mathcal{F}_{\mathsf{SOK}}$ or by using the Sign algorithm. It is required that the signing query (m, ϕ, w) contains a valid witness w to statement ϕ, otherwise the query is ignored. The functionality uses the simulated signing algorithm to produce all signatures output by the signing interface, thereby guaranteeing that all signatures are independent of, and thus cannot leak information about, the witnesses used to generate them. All signatures are verified prior to being output by $\mathcal{F}_{\mathsf{SOK}}$, providing completeness.

Verification can be accomplished either through the Verify interface or the Verify algorithm. The verification algorithm is required to be perfectly complete; in the case that verification cannot be completed on a correctly generated signature the functionality halts and outputs a completeness error. On verification query (m, ϕ, σ), verification checks that the signature was generated with knowledge of a witness by using the Extract algorithm if there is no record of a signature for (ϕ, m). An unforgeability error is triggered when a signature verifies but a valid witness cannot be extracted.

As discussed in [21], when parameterized by a polynomial time TM $M_{\mathcal{L}}$, the functionality $\mathcal{F}_{\mathsf{SOK}}$ can be defined with respect to any other functionality and signers can produce signatures on the basis of knowing an accepting input to any other functionality.

C Unilateral Witness-Authenticated Key Exchange

C.1 UWAKE Definitions

Unilateral authentication is defined in the two participant setting. The initiator Init is unauthenticated and the Responder Res is authenticated. The set of potential participants is changed to be $\mathcal{P} = \{\mathsf{Init}, \mathsf{Res}_1, \ldots, \mathsf{Res}_{n-1}\}$, authenticating with respect to the vector $\Phi = \langle \phi_{\mathsf{Init}}, \phi_{R_1}, \ldots, \phi_{R_{n-1}} \rangle$ 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 UWAKE 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, the group WAKE definition can still apply under the condition that the statement ϕ_{Init} is easy.

The goal of the adversary in the confidentiality experiment is to be able to distinguish a random key from the real session key generated by an eavesdropped protocol execution. The modification to the group WAKE experiment

$\mathcal{F}_{\mathsf{SOK}}$

The functionality $\mathcal{F}_{\mathsf{SOK}}$ is parameterized by a relation $\mathcal{R}$ and interacts with a set of parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ and adversary $\mathcal{S}$ via the following queries:

- Setup Queries from party P_i: $(\mathsf{SetUp} : \mathsf{sid})$
  - If any of the following do not hold: ignore the query
    - This is the first query for sid
    - $\mathsf{sid} = (\mathcal{R}_{\mathcal{L}}, \mathsf{sid}')$ for some sid'
  - Record (sid, P_i)
  - Output $(\mathsf{SetUp}, \mathsf{sid})$ to the adversary $\mathcal{S}$
  - Receive $(\mathsf{Algorithms} : \mathsf{sid}, \mathsf{Verify}, \mathsf{Sign}, \mathsf{Simsign}, \mathsf{Extract})$ from $\mathcal{S}$ where the following hold:
    - $\mathsf{Sign}, \mathsf{Simsign}, \mathsf{Extract}$ are descriptions for PPT TMs
    - Verify is a description of a deterministic polytime TM
  - Record $(\mathsf{sid}, P_i, \mathsf{Verify}, \mathsf{Sign}, \mathsf{Simsign}, \mathsf{Extract})$
  - Output $(\mathsf{Algorithms}, \mathsf{sid}, \mathsf{Sign}, \mathsf{Verify})$ to P_i
- Signature Generation Queries from P_i: $(\mathsf{Sign} : \mathsf{sid}, m, \phi, w)$
  - If $\mathcal{R}_{\mathcal{L}}(\phi, w) \neq 1$: ignore the query
  - Compute $\sigma \leftarrow \mathsf{Simsign}(m, \phi)$
  - If $\mathsf{Verify}(m, \phi, \sigma) = 1$
    - Output $(\mathsf{Signature}, \mathsf{sid}, m, \phi, \sigma)$ to P_i
    - Record (m, ϕ, σ)
  - Otherwise: output $(\mathsf{Completeness} - \mathsf{Error})$ to P_i and halt
- Signature Verification Queries from P_i: $(\mathsf{Verify} : \mathsf{sid}, m, \phi, \sigma)$
  - If there exists a record (m, ϕ, σ') for some σ': output $(\mathsf{Verified}, \mathsf{sid}, m, \phi, \sigma, \mathsf{Verify}(m, \phi, \sigma))$ to P_i
  - Compute $w \leftarrow \mathsf{Extract}(m, \phi, \sigma)$
  - If $\mathcal{R}_{\mathcal{L}}(\phi, w) = 1$: output $(\mathsf{Verified}, \mathsf{sid}, m, \phi, \sigma, \mathsf{Verify}(m, \phi, \sigma))$ to P_i
  - If $\mathsf{Verify}(m, \phi, \sigma) = 0$: output $(\mathsf{Verified}, \mathsf{sid}, m, \phi, \sigma, 0)$ to P_i
  - Else: output $(\mathsf{Unforgeability} - \mathsf{Error})$ to P_i and halt

Fig. 5. The signature of knowledge ideal functionality.

seen in Figure 1 is that the challenge must be an instance of Init. Otherwise, confidentiality is defined just as in Definition 1.

The goal of the adversary in the authenticity experiment should be to convince an instance of the Initiator to accept and consequently generate a session key. 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. Therefore, the change to Figure 1 is also that the challenge instance output by $\mathcal{A}$ should always be an instance of Init. Otherwise, authenticity is defined just as in Definition 3.

The simulatability experiment remains unchanged.

C.2 UWAKE Construction

Given a signature of knowledge and a key encapsulation mechanism one can construct UWAKE. 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 UWAKE, the public parameters output by SetUp are $pp_{\mathsf{UWAKE}} = (\lambda, \mathcal{R}, pp_\Sigma)$, the security parameter, the relation and the public parameters for the signature of knowledge scheme Σ. The public parameters output by SimSetUp also include the trapdoor for the signature τ_Σ. In Figure 6 we present the construction of UWAKE from KEM protocol $\mathcal{K}$ and SOK scheme Σ.

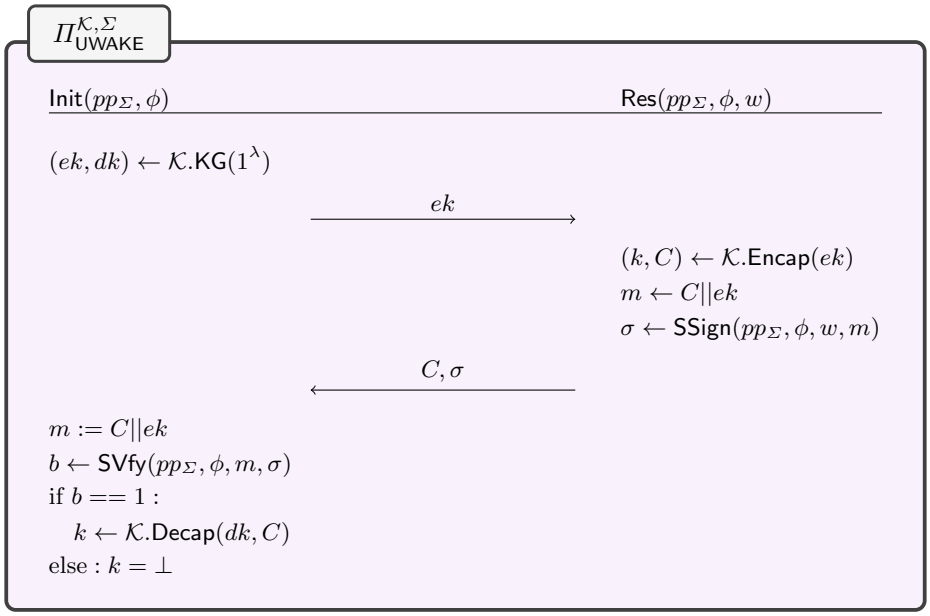

Fig. 6. A UWAKE protocol from KEM $\mathcal{K}$ and SOK Σ.

Theorem 5 ($\Pi^{\mathcal{K},\Sigma}_{\mathsf{UWAKE}}$ is secure.). *Let $\mathcal{K}$ be a correct and* KEM-CPA*-secure key encapsulation mechanism as in Definition 7 and let Σ be as-nBB-SE-secure signature of knowledge as defined in Section B.3. Then the protocol $\Pi^{\mathcal{K},\Sigma}_{\mathsf{UWAKE}}$ as seen in Figure 6 is a fully secure* UWAKE*.*

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

Confidentiality: An adversary $\mathcal{A}_C$ with nonnegligible advantage in the UWAKE confidentiality experiment implies the existence of an adversary $\mathcal{A}_K$ with non-negligible advantage in the KEM-CPA experiment seen in Figure 4. 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 $(\mathsf{ek}^*, \mathsf{C}^*, k^*_{b_{\mathsf{KEM\text{-}CPA}}})$ and must guess the real or random bit $b_{\mathsf{KEM\text{-}CPA}}$. The adversary $\mathcal{A}_K$ sets up the UWAKE as follows: generates the public parameters for the signature of knowledge $pp_\Sigma \leftarrow \Sigma.\mathsf{SSetup}(\mathcal{R})$, and sets the public parameters as $pp = (\lambda, \mathcal{R}, pp_\Sigma)$ and samples $W \leftarrow \mathcal{D}_\Phi$. Then, $\mathcal{A}_K$ runs $\mathcal{A}_C$ with input (pp, Φ, W). In response to any queries from $\mathcal{A}_C$ to $\mathsf{Execute}$ of the form $q_k = (\mathsf{Init}, k, \mathsf{Res}_i, j)$, $\mathcal{A}_K$ generates an honest transcript between Π^k_{Init} and $\Pi^j_{\mathsf{Res}_i}$ of the form $T_k = (\mathsf{ek}_k, (C_k, \sigma_k))$ with signature generated as $\sigma_k \leftarrow \Sigma.\mathsf{SSign}(pp_\Sigma, \phi_i, w_i, (ek_k || C_k))$ for the sampled w_i. $\mathcal{A}_K$ stores the generated session key as $\mathsf{sk}^k_{\mathsf{Init}}$ and $\mathsf{sk}^i_{\mathsf{Res}_j}$, sets $\mathsf{sid}^k_{\mathsf{Init}} = \mathsf{sid}^j_{\mathsf{Res}_i} = T_k$ and sets $\mathsf{acc}^k_{\mathsf{Init}} = \mathsf{acc}^j_{\mathsf{Res}_i} = \mathsf{TRUE}$. Let Q_E be the total number of queries made to $\mathsf{Execute}$ by $\mathcal{A}_C$. With probability $\frac{1}{Q_E}$, $\mathcal{A}_K$ injects as the query response $T^* = (\mathsf{ek}^*, (C^*, \sigma^*))$ for $\sigma^* \leftarrow \Sigma.\mathsf{SSign}(pp_\Sigma, \phi_j, w_j, (\mathsf{ek}^* || C^*))$. In response to any queries to Reveal of the form (Init, i), $\mathcal{A}_K$ replies with $\mathsf{sk}^i_{\mathsf{Init}}$.

If $\mathcal{A}_C$ outputs challenge (Init, i) such that $\mathsf{sid}^i_{\mathsf{Init}} \equiv T^*$ then $\mathcal{A}_K$ provides $k^*_{b_{\mathsf{KEM\text{-}CPA}}}$ as the challenge session key. In this case, when $\mathcal{A}_C$ outputs guess bit b'_{confid}, $\mathcal{A}_K$ forwards this bit as the guess $b'_{\mathsf{KEM\text{-}CPA}}$. If $\mathcal{A}_C$ outputs some other challenge transcript then $\mathcal{A}_K$ flips a bit and provides either the real key or a random key.

$\mathcal{A}_K$ 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. If T^* appears only once in the list of Q_E queries, in other words if E happened, then the probability that $\mathcal{A}_C$ selects T^* as her challenge is $\frac{1}{Q_E}$. Consequently, $\mathsf{Pr}[\mathsf{Exp}^{\mathsf{KEM\text{-}CPA}}_{\mathcal{A}_K}(\lambda)]$ is lower bounded by the following:

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

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

$\mathcal{A}_S$ assigns as the public parameters $pp \leftarrow (pp_\Sigma, \mathcal{R}, \lambda)$ and runs $\mathcal{A}_A$ with input Φ. Queries to Send and Reveal are answered by $\mathcal{A}_S$ honestly except that the signatures are generated via queries to the $\mathsf{SSimSign}$ oracle. $\mathcal{A}_A$ outputs challenge $(\mathsf{Init}, i, \mathsf{Res}_j)$ with $\mathsf{Res}_j \in \mathcal{I}(\mathsf{Init}, i)$. $\mathcal{A}_A$ is admissible and thus $\mathcal{A}_A$ was not forwarding for Π^i_{Init} and Res_j. By the definition of forwarding adversary, 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 can then (wlog) 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 $\mathcal{A}_A$ sent an $\mathsf{ek}' \neq \mathsf{ek}$, σ will not verify and Init will not accept. Therefore, $\mathcal{A}_A$ must not have replaced *only* the first message. 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: $\mathsf{SVfy}(pp_\Sigma, \phi_{\mathsf{Res}_j}, m = (C'||\mathsf{ek}), \sigma') = 1$ for ek output by Π^i_{Init}. So, $\mathcal{A}_S$ outputs $(\phi_j, m = (C'||\mathsf{ek}), \sigma')$ as the 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, if there is no extractor for $\mathcal{A}_A$ then there is no extractor for $\mathcal{A}_S$.

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

Simulatability: $\mathcal{A}_W$ with nonnegligible advantage against UWAKE simulatability implies the existence of an adversary $\mathcal{A}_S$ against the simulation experiment for Σ. Assume that $\mathcal{R}$ is such that $\mathcal{A}_W$ has a nonnegligle advantage in the simulation experiment.

On input pp^b_Σ, $\mathcal{A}_S$ constructs the public parameters for the UWAKE as $pp_b = (pp^b_\Sigma, \mathcal{R}, \lambda)$ and inputs 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. $\mathcal{A}_S$ responds to all queries to Send and Reveal honestly except $\mathcal{A}_S$ queries the oracle $\mathcal{S}^b_{pp^b_\Sigma, \tau}$ to generate any signatures. Upon receipt of the guess bit b'_Σ from $\mathcal{A}_W$, $\mathcal{A}_S$ uses this as guess bit b' for the real or random bit. The advantage of $\mathcal{A}_S$ is then non-negligible as it is greater than or equal to that of $\mathcal{A}_W$.

D Universally Composable Witness-Authenticated Key Exchange

Summary: the functionality $\mathcal{F}_{\mathsf{WAKE}}$ is realized by the functionality $s\mathcal{F}_{\mathsf{WAR}}$. The functionality $s\mathcal{F}_{\mathsf{WAR}}$ is the result of applying the split functionality transformation on $\mathcal{F}_{\mathsf{WAR}}$, a variant of $\mathcal{F}_{\mathsf{WAKE}}$ with the assumption of authenticated channels, thereby removing that assumption.

D.1 The Ideal WAKE Functionality

In the UC framework cryptographic tasks are modeled as ideal functionalities, which are essentially a trusted third party which behaves ideally with knowledge of all of the relevant inputs of each party. In this case the parties provide $\mathcal{F}_{\mathsf{WAKE}}$ with their witnesses and the functionality verifies these witnesses and then issues keys to the parties upon verification. The ideal functionality is also modeled to allow for an ideal amount of leakage to the adversary and permit ideal adversarial influence on the outcome of the exchange through adversarial influence interfaces.

The functionality $\mathcal{F}_{\mathsf{WAKE}}$ is parameterized by λ and $\mathcal{R}$ and interacts with parties in $\mathcal{P}$ along with the ideal world adversary $\mathcal{S}$. The parties in $\mathcal{P}$ interact with the functionality $\mathcal{F}_{\mathsf{WAKE}}$ via NewSession queries. Each of these queries include the session identifier $\mathsf{sid} = (\mathsf{sid}', \mathcal{R})$, the secret witness w_i, the public statement ϕ_i and the set of expected statements Φ.[16] The functionality enforces that all queried Φ are equal. Upon receipt of a NewSession query the functionality checks that the query is *as expected* meaning that the statement set Φ is consistent with all previously received queries and that there are not too many parties authenticating with respect to the same statement.[17] Modeling ideal leakage, the functionality leaks the public values in each NewSession query to the adversary and then records the query and marks it as fresh. Then the adversary $\mathcal{S}$ has access to three types of queries: Knowledge, CompromiseSession and NewKey queries. In terms of ideal leakage, the Knowledge queries leak to the adversary if the queried party has knowledge of a valid witness on record. The remaining two queries model ideal influence on the exchange.

The adversary can always internally verify any witness against the public statements, meaning that $\mathcal{S}$ can in theory do an offline attack against the exchange and recover, with complete certainty, a witness to the statement of any party. As WAKE is computed over unauthenticated and asynchronous adversarially controlled channels, it is possible for the adversary to compute a witness w_i' and impersonate party P_i authenticating with respect to ϕ_i without *formally* corrupting that party. The CompromiseSession query allows the adversary to do exactly this and declare knowledge of a witness relevant to the exchange. When the adversary queries on (P, ϕ, w), if the party P is recorded as associated to statement ϕ and the queried $(\phi, w) \in \mathcal{R}$, then the party P is marked as compromised and is treated as corrupt for the remainder of the execution. The compromised marker indicates that this party can be impersonated by $\mathcal{A}$, and therefore all partners of a compromised party in a successful exchange receive adversarially chosen outputs. If the adversary queries with an invalid witness then

[16] Note that the format of Φ has changed in an inconsequential but convenient way from the previous chapter; in the game-based definitions Φ is a vector of statements, here Φ is a set of the form $\Phi = \{(\phi_j, m_j)\}_{\sum_j m_j = n}$ where each statement-multiplicity pair in the set indicates that the querying party expects to interact with m_j parties authenticating with respect to the statement ϕ_j.

[17] Optionally, the set Φ can be included in the session identifier of the protocol along with $\mathcal{R}$ to model that this set is public, fixed prior to execution and all parties agree on the set.

$\mathcal{F}_{\mathsf{WAKE}}$

The functionality $\mathcal{F}_{\mathsf{WAKE}}$ is parameterized by security parameter λ and NP relation $\mathcal{R}$, and interacts with the adversary $\mathcal{S}$ and a set of parties $\mathcal{P}$ via the following queries:

- (NewSession : $\mathsf{sid}, \mathsf{w}_i, \Phi_i, \phi_i) \leftarrow P_i$
  - Ignore the query if any of the following are true:
    - This is not the first NewSession query for party P_i
    - There does not exist a unique $m_k \in \{1, \ldots, n\}$ such that $(\phi_i, m_k) \in \Phi_i$
    - There exist m_k records of the form $(P_j, \Phi_i, \phi_i, w_j)$ for $(\phi_i, m_k) \in \Phi_i$
    - There exists a record $(P_j, \Phi_j, \phi_j, w_j)$ such that $\Phi_j \neq \Phi_i$
  - Leak (NewSession, $\mathsf{sid}, P_i, \Phi_i, \phi_i$) to the adversary $\mathcal{S}$
  - Record $(P_i, \Phi_i, \phi_i, w_i)$ and mark this record fresh
- Knowledge queries from the adversary $\mathcal{S}$: (Knowledge : sid, P_i)
  - If there exists a record (P_i, Φ, ϕ, w): output (Knldg, $\mathsf{sid}, P_i, \phi, \mathcal{R}(\phi, w)$) to $\mathcal{S}$
  - Otherwise: ignore the query
- Compromise Session Queries from the adversary $\mathcal{S}$: (CompromiseSession : sid, P, ϕ, w)
  - If any party in $\mathcal{P}$ is marked completed or there does not exist a record of the form (P, Φ, ϕ, w'): ignore the query
  - If $(\phi, w) \in \mathcal{R}$: mark the record as compromised
  - If $(\phi, w) \notin \mathcal{R}$ and the record is marked fresh: mark the record as interrupted
  - Leak (Compromised : $\mathsf{sid}, \phi, \mathcal{R}(\phi, w)$) to $\mathcal{S}$
- New Key Queries from the adversary $\mathcal{S}$: (NewKey : $\mathsf{sid}, P_i, \mathsf{sk}$)
  - Ignore the query if any of the following hold:
    - There is no record (P_i, Φ, ϕ_i, w_i)
    - This is not the first NewKey query for P_i
    - There are not m' records of the form (P', Φ, ϕ', w') for each $(\phi', m') \in \Phi$
  - Compute the boolean Succ:
    - If, for the record (P_i, Φ, ϕ_i, w_i), for all $(\phi', m') \in \Phi$ there exist m' records $\{(P_k, \Phi, \phi_k, w_k)\}_{k \leq m}$ such that for all k: $\mathcal{R}(\phi_k, w_k) = 1$ then Succ = True
    - Otherwise: Succ = False
  - If all of the following hold: output $(\mathsf{sid}, \mathsf{sk}')$ to P_i
    - The record is marked fresh
    - All parties $P \in \mathcal{P}$ are honest and all records (P, Φ, ϕ, w) are marked fresh or interrupted
    - Succ = True
    - A key sk' was already sent to party P_j for session sid
  - If any of the following is true: output $(\mathsf{sid}, \mathsf{sk})$ to P_i
    - P_i is corrupt or marked compromised
    - There $P_j \in \mathcal{P}$ which is corrupt or is marked compromised and Succ = True
  - In any other case: sample a random key $\mathsf{sk}_r \leftarrow_\$ \{0,1\}^\lambda$ and send $(\mathsf{sid}, \mathsf{sk}_r)$ to P_i
  - Mark the record $(P_i, \Phi_i, \phi_i, w_i)$ as completed

Fig. 7. The witness-authenticated key exchange ideal functionality.

that party is marked interrupted, which can be overwritten with a compromised upon a query with a valid witness. These queries are a controlled adaptive corrup-

tion mechanism for a static adversary, where the queried party does not output their entire state to the adversary so any secure erasure concerns are avoided. As such, CompromiseSession query can be submitted after a NewSession query whereas the Corrupt instructions must be submitted prior to any NewSession queries.

The NewKey queries from $\mathcal{S}$ issue keys to the queried party. In any successful session in which all parties are honest and uncompromised all parties receive the same uniformly random key from the functionality, as is ideal. Parties which are corrupted, compromised, or are participating in a successful session with a corrupt partner receive adversarially chosen keys. This models that there cannot be any guarantees made on the distribution of the keys generated by honest parties interacting with corrupt parties in a successful exchange. In all other cases the functionality samples and outputs an independent and freshly random key to issue to the queried party.

D.2 The Split Functionality

Link initialization, i.e. session authenticity, is achieved by the split authentication functionality $\mathcal{F}_{\mathsf{SA}} = s\mathcal{F}_{\mathsf{MAUTH}}$ which is the split functionality of the multi-authenticated message functionality. The multi-authenticated message functionality allows parties to exchange multiple messages over authenticated channels and can be thought of as the minimal task upon which to apply the split transformation. If parties can exchange messages over authenticated channels then they can execute arbitrary other tasks over these channels.

The functionality $\mathcal{F}_{\mathsf{SA}}$ is UC-realized by Π^{Ψ}_{SA} which can be seen in Figure 8 [10]. This protocol is proven to UC-realize the split authentication functionality in the presence of malicious, adaptive, adversaries in the bare model with no setup, given that the digital signature Ψ is EUF-CMA-secure. Protocol Π^{Ψ}_{SA} proceeds in two steps: (1) set up authenticated channels, (2) continue over these authenticated channels with a protocol that is secure under bounded concurrent composition. The code seen in Figure 8 is run by each party $P_i \in \mathcal{P}$ to initialize links and send authenticated messages.

In summary, split authentication is achieved by having all parties generate and exchange verification keys for an existentially unforgeable digital signature, and then sign each subsequent message with this verification key. Intuitively, this style of link initialization provides session authentication only the party who generated the verification key can produce verifying signatures. Additionally, the session identifier held by each party uniquely determines it's authentication set and any adversarial splitting must occur during this exchange of verification keys.

The main result of [10] relevant to this work is that, in the standalone model with no setup whatsoever, under the assumption of collision resistant hash functions and enchanced trapdoor permutations, for any PPT multiparty functionality $\mathcal{F}_{\mathsf{func}}$ there is a protocol that securely computes the split functionality $s\mathcal{F}_{\mathsf{func}}$ in the presence of static malicious adversaries. Assuming existence of enhanced

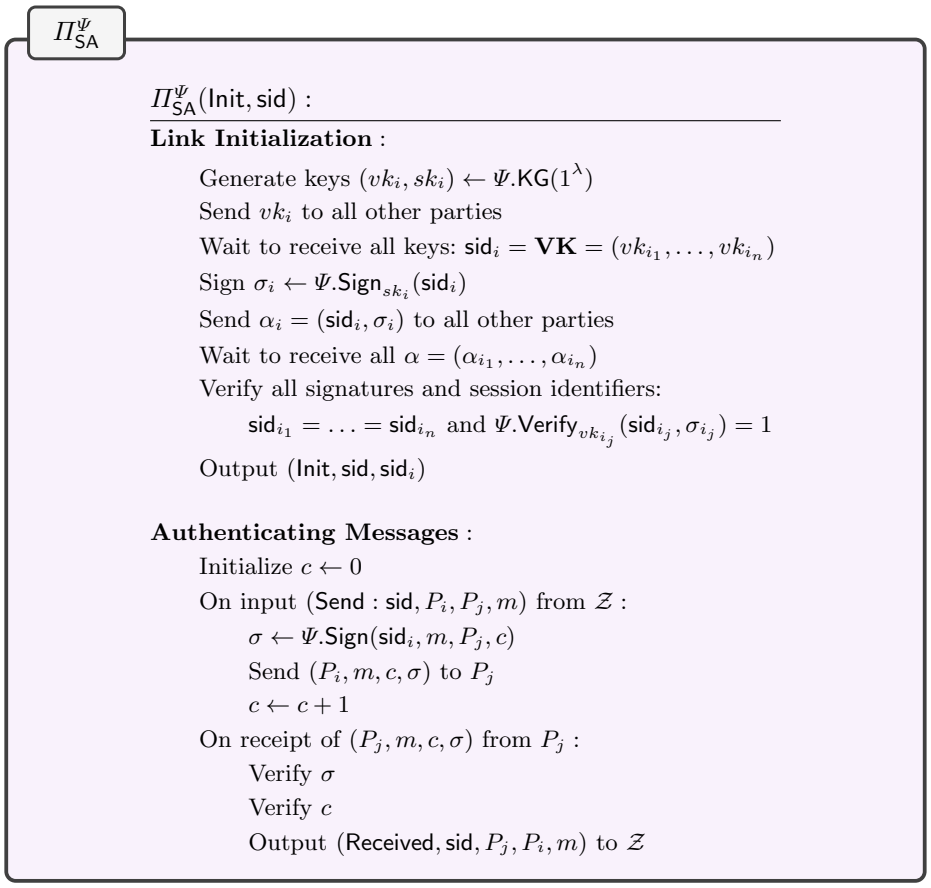

Fig. 8. The secure authentication protocol.

non-committing encryption this holds even with respect to adaptive adversaries without data erasures.

D.3 Witness Authenticated Randomness

One method of realizing $\mathcal{F}_{\mathsf{WAKE}}$ is to realize an ideal functionality for the task which assumes authenticated channels and then employ the split functionality to remove the assumption of authenticated channels. Functionality $\mathcal{F}_{\mathsf{WAR}}$, the witness-authenticated randomness functionality seen in Figure 10, is the analogue of $\mathcal{F}_{\mathsf{WAKE}}$ under the assumption of authenticated channels. The primary difference between $\mathcal{F}_{\mathsf{WAKE}}$ and $\mathcal{F}_{\mathsf{WAR}}$ is the absence of compromisation queries; the controlled adaptive corruption permitted by CompromiseSession in $\mathcal{F}_{\mathsf{WAKE}}$ is not possible in a setting with authenticated channels and is therefore not present in the definition of $\mathcal{F}_{\mathsf{WAR}}$.

The split functionality is adapted to WAKE by modifying queries to accept appropriate parameters needed to invoke instances of $\mathcal{F}_{\mathsf{WAR}}$. This split functionality $s\mathcal{F}_{\mathsf{WAR}}$ can be seen in Figure 9.

The split functionality is parameterized by security parameter λ and NP relation $\mathcal{R}$ and interacts with set of parties $\mathcal{P}$ and adversary $\mathcal{S}$. Initialization queries from parties in $\mathcal{P}$ contain the session identifier and the public parameter Φ as the set of expected statements. The functionality enforces that all queried Φ are equal; optionally, the parameter Φ can be included in the session identifier, along with the relation $\mathcal{R}$, to model that all parties agree upon the set prior to execution. Upon receipt of this Init query, the functionality leaks the party identifier and the public information to the adversary and records the query. The adversary $\mathcal{S}$ is permitted to adaptively split sessions via it's own Init interface, which include an authentication set $H \subset \mathcal{P}$ and a unique session identifier sid_H for the set. The ideal functionality checks that the set H is as expected, specifically that is disjoint or equal to all prior queried H' and has a unique session identifier. If this is the first query for H then the split functionality initializes a new instance of the functionality $\mathcal{F}^H_{\mathsf{WAR}}$. The functionality also sends a notification to the queried party, including the unique session identifier and the public information Φ. Following this, the split functionality routes messages between parties, the adversary, and the instances of $\mathcal{F}_{\mathsf{WAR}}$ through the Input and Output interfaces.

Corruption is modeled as a message from the ideal-world adversary $\mathcal{S}$ to the functionality, as in the UC framework. The split functionality records the corrupted parties, and informs instances of $\mathcal{F}_{\mathsf{WAR}}$ of the entire set of corrupted parties upon invokation. As these functionalities are secure under *static* corruption the ideal adversary $\mathcal{S}$ can only deliver corruption instructions prior to the start of the execution, specifically prior to any Init queries to $s\mathcal{F}_{\mathsf{WAR}}$.

As discussed the split functionality can be used to remove the assumption of authenticated channels from a functionality. A variant of WAKE that assumes authenticated channels could then be compiled with the split functionality to remove that assumption. The witness-authenticated randomness functionality

$s\mathcal{F}_{\mathsf{WAR}}$

The split functionality $s\mathcal{F}_{\mathsf{WAKE}}$ is parameterized by security parameter λ, NP relation $\mathcal{R}$, invokes instances of $\mathcal{F}_{\mathsf{WAR}}$, and interacts with a set of parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ and the adversary $\mathcal{S}$ via the following queries:

- Initialization:
  - $(\mathsf{Init} : \mathsf{sid}, \Phi_i)$ from a party P_i:
    - If $\Phi_i \neq \Phi_j$ for any record (P_j, Φ_j): ignore the query
    - Record (P_i, Φ_i)
    - Leak $(\mathsf{Init}, \mathsf{sid}, P_i, \Phi_i)$ to the adversary $\mathcal{S}$
  - $(\mathsf{Init} : \mathsf{sid}, P_i, H, \mathsf{sid}_H, \Phi)$ from the adversary $\mathcal{S}$:
    - If any of the following do not hold: ignore the query
      - $P_i \in H \subseteq \mathcal{P}$
      - For all $P_j \in H$: there exist records (P_j, Φ_j) and for all other $P_i \in H$ we have $\Phi_j = \Phi_i$
      - For all previously recorded $(H', \mathsf{sid}_{H'}, \Phi')$ either: (1) $H' \cap H = \emptyset$ and $\mathsf{sid}_{H'} \neq \mathsf{sid}_H$, or (2) $H = H'$, $\mathsf{sid}_{H'} = \mathsf{sid}_H$ and $\Phi = \Phi'$
    - Record $(H, \mathsf{sid}_H, \Phi)$ if it is not already recorded
    - Output $(\mathsf{Init}, \mathsf{sid}, \mathsf{sid}_H, \Phi)$ to party P_i
    - Initialize a new instance of the functionality $\mathcal{F}_{\mathsf{WAR}}$ as $\mathcal{F}_{\mathsf{WAR}}^{H,\mathsf{sid}_H}(\lambda, \mathcal{R}, \Phi)$ with session identifier sid_H, set of honest parties H and statement set Φ, denoted $\mathcal{F}_{\mathsf{WAR}}^{H}$ for ease of notation.
- Computation:
  - $(\mathsf{Input} : \mathsf{sid}, m)$ from P_i:
    - Find the set H such that $P_i \in H$ and forward m to $\mathcal{F}_H$ as if from P_i. If no such H exists then ignore the message.
  - $(\mathsf{Input} : \mathsf{sid}, H, P_j, m)$ from $\mathcal{S}$:
    - Find the record $(H, \mathsf{sid}_H, \Phi)$ and forward m to $\mathcal{F}_H$ as if coming from P_j. If with $P_j \in H$ or $\mathcal{F}_H$ not initialized then ignore the message.
  - $(\mathsf{Output} : P_i, m)$ from $\mathcal{F}_H$:
    - If $P_i \in H$: forward m to P_i
    - If $P_i \notin H$: forward m to $\mathcal{S}$
- Corruption:
  - $(\mathsf{Corrupt} : P)$ from $\mathcal{S}$
    - Record $(\mathsf{Corrupt}, P)$
    - Forward $(\mathsf{Corrupt} : P)$ to $\mathcal{F}_{\mathsf{WAR}}^{H}$ for all H

Fig. 9. The split witness authenticated randomness ideal functionality.

$\mathcal{F}_{\mathsf{WAR}}$ outputs random strings (not *called* session keys) to parties upon successful authentication with a witness to some public statement. This functionality, seen in Figure 10, is essentially WAKE under the assumption of authenticated channels.

The witness-authenticated randomness functionality is parameterized by security parameter λ, relation $\mathcal{R}$ and set of statements $\Phi = \{\phi_k, m_k\}$ and interacts with the adversary $\mathcal{S}$ and set of parties $\mathcal{P}$. Just as above, the set Φ can optionally be included in the session identifier. Just as with $\mathcal{F}_{\mathsf{WAKE}}$ parties must submit a NewSession query to $\mathcal{F}_{\mathsf{WAR}}$ detailing their statement-witness pair. The functionality, upon receipt of this query, verifies that it is *as expected*, records the query and leaks the public information to $\mathcal{S}$.

$\mathcal{F}_{\mathsf{WAR}}$

The functionality $\mathcal{F}_{\mathsf{WAR}}$ is parameterized by security parameter λ, an NP relation $\mathcal{R}$, and set of statements $\Phi = \{(\phi_k, m_k)\}_{k=1}^{N}$ such that $\sum_{k=1}^{N} m_k = n$, and interacts with the adversary $\mathcal{S}$ and a set of parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ via the following queries:

- New Session Queries from party P_i: (NewSession : $\mathsf{sid}, \phi_i, \mathsf{w}_i$)
  - If any of the following are true: ignore the query
    - There does not exist an $m_k \in \mathbb{N}$ such that $(\phi_i, m_k) \in \Phi$
    - There already exist m_k records of the form (P_j, ϕ_i, w_j) for $(\phi_i, m_k) \in \Phi$
  - Leak (NewSession, $\mathsf{sid}, \phi_i, P_i$) to the adversary $\mathcal{S}$
  - Record (P_i, ϕ_i, w_i)
- Has Knowledge Queries from the adversary $\mathcal{S}$: ($\mathsf{Knldg} : \mathsf{sid}, P_i$)
  - If there exists a record (P_i, ϕ_i, w_i) and $(\phi_i, w_i) \in \mathcal{R}$: leak ($\mathsf{Knldg}, \mathsf{sid}, P_i, \phi_i, 1$) to $\mathcal{S}$
  - If there exists a record (P_i, ϕ_i, w_i) and $(\phi_i, w_i) \notin \mathcal{R}$: leak ($\mathsf{Knldg}, \mathsf{sid}, P_i, \phi_i, 0$) to $\mathcal{S}$
  - Otherwise: ignore the query.
- New Key Queries from the adversary $\mathcal{S}$: (NewKey : $\mathsf{sid}, P_i, \mathsf{sk}$)
  - If any of the following are true: ignore the query
    - There is no record (P_i, ϕ_i, w_i)
    - This is not the first new key query for P_i
    - There is not a record (P_j, ϕ_j, w_j) for each $P_j \in \mathcal{P}$
  - Compute the boolean Succ:
    - If for each $(m', \phi') \in \Phi$ there are m' records $\{(P_k, \phi_k, w_k)\}_{k \leq m'}$ such that for all k: $(\phi_k, w_k) \in \mathcal{R}$ then $\mathsf{Succ} = \mathsf{True}$
    - Otherwise: $\mathsf{Succ} = \mathsf{False}$
  - If all of the following are true: output $(\mathsf{sid}, \mathsf{sk}')$ to P_i
    - All parties are honest
    - $\mathsf{Succ} = \mathsf{True}$
    - Key sk' was already output to some P_j
  - If either the following are true: output $(\mathsf{sid}, \mathsf{sk})$ to P_i
    - P_i is corrupt
    - There is some corrupt P_j and $\mathsf{Succ} = \mathsf{True}$
  - Else: sample a random key $\mathsf{sk}_r \leftarrow_{\$} \{0,1\}^{\lambda}$ and output $(\mathsf{sid}, \mathsf{sk}_r)$ to P_i
- ($\mathsf{Corrupt} : P$) from $\mathcal{S}$
  - Record ($\mathsf{Corrupt}, P$)

Fig. 10. The witness-authenticated randomness ideal functionality.

The Knldg queries submitted by the adversary leak if the queried party has knowledge of a valid witness if the party has submitted a NewSession query, just as with $\mathcal{F}_{\mathsf{WAKE}}$. Upon receipt of a NewKey query submitted by the adversary the ideal functionality determines if the parties have all successfully authenticated and then issues a key to the queried party. If the exchange was successful and all parties were honest then an equal and uniformly random key is output to those parties. If a party is corrupt or was participating in a successful exchange with a corrupt party then the queried key is forwarded to the queried party, again modeling that these keys can be arbitrarily distributed. Otherwise, a freshly sampled independently random key is output to the queried party.

Again, the CompromiseSession query is absent from $\mathcal{F}_{\mathsf{WAR}}$. This is because under the assumption of authenticated channels the adversary cannot impersonate another party participating in the exchange after the session has started. The *static* adversary is limited to Corrupt queries prior to the start of the session.

We present a protocol Π_{WAKE} which realizes the functionality $\mathcal{F}_{\mathsf{WAKE}}$ in the $s\mathcal{F}_{\mathsf{WAR}}$-hybrid model towards proving the equivalence of $\mathcal{F}_{\mathsf{WAKE}}$ and $s\mathcal{F}_{\mathsf{WAR}}$. The simplicity of the protocol Π_{WAR} is intentional; Π_{WAR} is essentially the ideal protocol $\mathsf{Ideal}_{s\mathcal{F}_{\mathsf{WAR}}}$ as defined in Section B.1. In other words, this means that the split witness-authenticated randomness functionality is enough to realize $\mathcal{F}_{\mathsf{WAKE}}$.

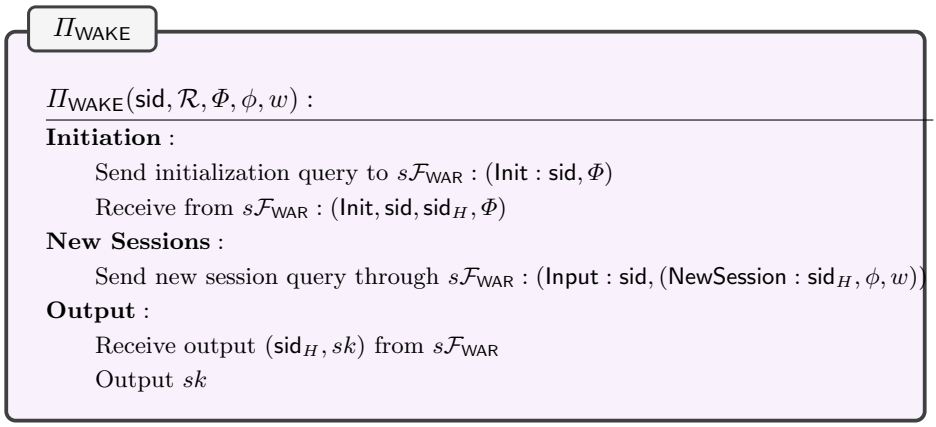

Fig. 11. A (dummy) WAKE protocol in the $s\mathcal{F}_{\mathsf{WAR}}$-hybrid model.

Π_{WAR} is a dummy protocol in which a real world party has a the same behavior as an ideal world (dummy) parties, creating and submitting queries from the inputs provided by $\mathcal{Z}$ and forwarding the output from the functionality to $\mathcal{Z}$. The protocol in Figure 11 describes the behavior of each party in $\mathcal{P}$ upon activation with inputs $(\mathsf{sid}, \mathcal{R}, \Phi, \phi, w)$ by $\mathcal{Z}$. The inputs are the session identifier, the relation, the statement set, the party's specific statement and a witness to that statement, respectively. The party first sends an intialization query to the split functionality, then a new session query. The party then waits for output from the split functionality and outputs the received key to $\mathcal{Z}$.

Theorem 6 states that this protocol, the ideal protocol $\mathsf{Ideal}_{s\mathcal{F}_{\mathsf{WAR}}}$, realizes $\mathcal{F}_{\mathsf{WAKE}}$.

Theorem 6. *Protocol Π_{WAKE} securely realizes $\mathcal{F}_{\mathsf{WAKE}}$ in the $s\mathcal{F}_{\mathsf{WAR}}$-hybrid model with respect to any NP relation $\mathcal{R}$ in the presence of static malicious adversaries.*

Proof. Consider static malicious real-world adversaries $\mathcal{A}$ corrupting an arbitrary number of parties. For every efficient $\mathcal{A}$ there exists a simulator $\mathcal{S}_{\mathsf{WAKE}}$ such that no efficient environment $\mathcal{Z}$ can distinguish between an execution involving

$(\mathcal{F}_{\mathsf{WAKE}}, \mathcal{S}_{\mathsf{WAKE}})$ from an execution involving $(\Pi_{\mathsf{WAKE}}, \mathcal{A})$ in the $s\mathcal{F}_{\mathsf{WAR}}$-hybrid model. This simulator is presented in Figure 12.

$\mathcal{S}_{\mathsf{WAKE}}$

$\mathcal{S}$ initializes $\mathcal{A}$ and reliably forwards all messages between $\mathcal{Z}$ and $\mathcal{A}$. Corruption instructions from $\mathcal{A}$ are obeyed and forwarded to $s\mathcal{F}_{\mathsf{WAR}}$ if they occur prior to any initialization queries.

- For each honest P_i:
  - Receive $(\mathsf{NewSession}, \mathsf{sid}, P_i, \Phi_i, \phi_i)$ from $\mathcal{F}_{\mathsf{WAKE}}$
  - Query $(\mathsf{Knowledge} : \mathsf{sid}, P_i)$ to $\mathcal{F}_{\mathsf{WAKE}}$ and receive $(\mathsf{Knowledge} : \mathsf{sid}, P_i, b_i)$
  - Record (P_i, b_i)
- **The Split Functionality**:
  - For each honest P_i: leak $(\mathsf{Init}, \mathsf{sid}, P_i, \Phi_i)$ to $\mathcal{A}$
  - Upon $(\mathsf{Init} : \mathsf{sid}, \Phi_j)$ from corrupt P_j: leak $(\mathsf{Init}, \mathsf{sid}, P_j, \Phi_j)$ to $\mathcal{A}$
  - Upon $(\mathsf{Init} : \mathsf{sid}, P_i, H, \mathsf{sid}_H, \Phi)$ from $\mathcal{A}$:
    - If the following conditions hold: output $(\mathsf{Init}, \mathsf{sid}, \mathsf{sid}_H, \Phi)$ to P_i and (if this is the first query for H) begin to simulate a session of $\mathcal{F}^H_{\mathsf{WAR}}(\lambda, \mathcal{R}, \Phi)$
      - $P_i \in H \subseteq \mathcal{P}$ and for all $P_k \in H$ the input Φ provided by $\mathcal{Z}$ is such that $\Phi = \Phi_k = \Phi_i$
      - The authentication set H is either disjoint from (with an unequal session identifier) or equal to (with an equal session identifier) all prior queried H'
      - Record $\mathcal{I}_H = \{P_j \notin H | P_j \text{ not corrupt according to } \mathcal{F}_{\mathsf{WAKE}}\}$
  - Upon $(\mathsf{Input} : \mathsf{sid}, m)$ from P_i: parse m as if sent directly to the internally simulated $\mathcal{F}^H_{\mathsf{WAR}}$ for $P_i \in H$
  - Upon $(\mathsf{Input} : \mathsf{sid}, H, P_j, m)$ from $\mathcal{A}$: parse m as if sent directly to the internally simulated $\mathcal{F}^H_{\mathsf{WAR}}$ from $P_j \notin H$
  - All other queries: ignore and do nothing
- **The Witness Authenticated Randomness Instance** H:
  - For honest parties P_i: leak $(\mathsf{NewSession}, \mathsf{sid}, \phi_i, P_i)$ to $\mathcal{A}$
  - Upon $(\mathsf{NewSession} : \mathsf{sid}_H, \phi_j, w_j)$ from corrupt P_j:
    - Record $(P_j, \mathsf{sid}_H, \phi_j, w_j)$ and leak $(\mathsf{NewSession}, \mathsf{sid}_H, \phi_i, P_i)$ to $\mathcal{A}$
  - Upon $(\mathsf{NewSession} : \mathsf{sid}_H, \phi_j, w_j)$ from $P_j \in \mathcal{I}_H$:
    - Query $(\mathsf{CompromiseSession} : \mathsf{sid}, \phi_j, w_j)$ to $\mathcal{F}_{\mathsf{WAKE}}$
  - Upon $(\mathsf{Knowledge} : \mathsf{sid}_H, P_i)$ from $\mathcal{A}$:
    - Query $(\mathsf{Knowledge} : \mathsf{sid}, P_i)$ to $\mathcal{F}_{\mathsf{WAKE}}$, receive $(\mathsf{Knowledge}, P_i, \phi_i, b_i)$ and leak to $\mathcal{A}$
  - Upon $(\mathsf{NewKey} : \mathsf{sid}_H, P_i, sk)$ from $\mathcal{A}$ to simulated $\mathcal{F}^H_{\mathsf{WAR}}$
    - If P_i is corrupt or $P_i \in \mathcal{I}_H$: output (sid_H, sk) to P_j
    - If P_i is honest: generate the output of simulated $\mathcal{F}^H_{\mathsf{WAR}}$ to P_i (exactly as the real functionality would) as $\bar{sk}$ and issue $(\mathsf{NewKey} : \mathsf{sid}, P_i, \bar{sk})$ to $\mathcal{F}_{\mathsf{WAKE}}$
- Corrupt NewSession Queries:
  - For all corrupt P_j: if there exists a record $(P_j, \mathsf{sid}_H, \phi_j, w_j)$ such that $(\phi_j, w_j) \in \mathcal{R}$ set $w' \leftarrow w_j$, otherwise set w' as a random w_j on record
  - Query $(\mathsf{NewSession} : \mathsf{sid}, w', \Phi, \phi)$ to $\mathcal{F}_{\mathsf{WAKE}}$

Fig. 12. The simulator for protocol Π_{WAKE}.

The simulator $\mathcal{S}_{\mathsf{WAKE}}$ must simulate: (1) the public messages sent by any honest parties according to Π_{WAKE}, exactly the two messages to the ideal split functionality $s\mathcal{F}_{\mathsf{WAR}}$ and the instance of $\mathcal{F}_{\mathsf{WAR}}$, (2) responses to queries from corrupt parties and the adversary on behalf of $s\mathcal{F}_{\mathsf{WAR}}$ and the instance of $\mathcal{F}_{\mathsf{WAR}}$, and (3) the outputs to the parties.

The environment does not learn any communication between honest parties and ideal functionalities, and only learns the view of the adversary consisting of any communications involving corrupt parties. More explicitly in the real world the environment does not learn anything about the honest parties except their inputs and outputs. All the simulator needs to produce are the query responses output to the adversary and any corrupt parties. The simulator produces query responses in exactly the same way as the split functionality in Figure 9. The simulation is straightforward as the simulator receives queries with the secrets. Honest party outputs are those output by $s\mathcal{F}_{\mathsf{WAR}}$ from $\mathcal{F}_{\mathsf{WAR}}$ in the real world and are those output by $\mathcal{F}_{\mathsf{WAKE}}$ in the ideal world. Therefore, what remains to be proven is that these output distributions are identical.

Consider a party P_i which is honest (not corrupted) in the real world while running Π_{WAKE}. This party interacts with the split functionality $s\mathcal{F}_{\mathsf{WAR}}$ and a single instance of $\mathcal{F}_{\mathsf{WAR}}^{H}$. In all other instances the party P_i is controlled by the adversary and considered corrupt: all $P_i \notin H'$ are corrupt in the execution of $\mathcal{F}_{\mathsf{WAR}}^{H'}$ as guaranteed by the split functionality. Party P_i outputs the key output to it by $\mathcal{F}_{\mathsf{WAR}}^{H}$ for $P_i \in H$ according to the protocol. If the real world session was split by $\mathcal{A}$ then every honest party P necessarily interacts with at least one adversarially controlled partner in the execution of $\mathcal{F}_{\mathsf{WAR}}$ so in the case of success that party receives an adversarially chosen key and in the case of failure that party receives a random key. Then the output of each party in Π_{WAKE} will be either adversarially chosen or random depending on the outcome of each instance. If the session was not split then a single common instance of $\mathcal{F}_{\mathsf{WAR}}$ is executed by all honest and corrupt parties and the cases for key distribution are as in the functionality $\mathcal{F}_{\mathsf{WAR}}$. Observe that the output distribution in the real world is dependent upon adversarial splitting. Therefore the output in the ideal world must also be dependent on splitting by $\mathcal{A}$.

In the ideal world an honest party P_i is simulated after the dummy party issues their query to $\mathcal{F}_{\mathsf{WAKE}}$ which is then leaked (without the secret) to the simulator. The parties necessarily engage in a single execution of $\mathcal{F}_{\mathsf{WAKE}}$ and the keys output to the parties are as in the functionality $\mathcal{F}_{\mathsf{WAKE}}$. Dependence of the output on splitting is enforced by the simulator through $\mathsf{CompromiseSession}$ queries. There are four cases, **corruption** indicates that the adversary issued corruption instructions that were followed and **splitting** indicates that the adversary split the session of $\mathcal{F}_{\mathsf{WAR}}$. In all four cases the outputs of the parties in the real world are identically distributed from those output in the ideal world. That is, that the keys as output in a session with $\mathcal{F}_{\mathsf{WAKE}}$ are identically distributed to those output when running protocol Π_{WAKE} by $s\mathcal{F}_{\mathsf{WAR}}$.

Claim 1: In the case of **(corruption, splitting)** *the distributions of outputs between the real and ideal worlds are identical.*

If the adversary issued corruption instructions to $\mathcal{F}_{\mathsf{WAKE}}$ these parties are corrupted according to $\mathcal{F}_{\mathsf{WAKE}}$ and to all instances of $\mathcal{F}_{\mathsf{WAR}}$. If the adversary split the session of $\mathcal{F}_{\mathsf{WAR}}$, each individual copy of $\mathcal{F}_{\mathsf{WAR}}$ contains additional corrupt parties which are considered honest to $\mathcal{F}_{\mathsf{WAKE}}$. Each individual session of $\mathcal{F}_{\mathsf{WAR}}$ contains corrupted parties, therefore if a session is successful then honest parties receive adversarially chosen keys, otherwise the parties receive independently random keys.

If there is at least one successful copy of $\mathcal{F}_{\mathsf{WAR}}$ then every adversarially controlled party issued a query with a valid witness, therefore the simulator will have issued a NewSession query for each corrupted party. Then, $\mathcal{F}_{\mathsf{WAKE}}$ will issue adversarially chosen keys to honest parties as well because they participated in a successful exchange with at least one corrupt partner. The keys that are queried to $\mathcal{F}_{\mathsf{WAKE}}$ by the simulator are exactly those keys held by the simulated parties, so $\mathcal{F}_{\mathsf{WAKE}}$ outputs to the dummy parties the same keys held by the simulated parties and thus the distributions are equal.

If all copies of $\mathcal{F}_{\mathsf{WAR}}$ fail then every simulated party receives an independently random output. In this case the simulator will have queried either valid or invalid witnesses for each corrupt party. If all of the issued queries for the corrupt parties are valid (meaning that the impersonated parties were the fault in each execution of $\mathcal{F}_{\mathsf{WAR}}$) then the session of $\mathcal{F}_{\mathsf{WAKE}}$ succeeds and outputs adversarially chosen keys. The queried keys are exactly the keys generated in the simulation so the distributions are identical. Otherwise, $\mathcal{F}_{\mathsf{WAKE}}$ fails and the outputs of $\mathcal{F}_{\mathsf{WAKE}}$ are uniformly random, distributed just as in the simulation and real world.

Claim 2: In the case of **(corruption, no splitting)** *the distributions of outputs between the real and ideal worlds are identical.*

If the adversary issued corruption instructions to $\mathcal{F}_{\mathsf{WAKE}}$ and did not split the session of $\mathcal{F}_{\mathsf{WAR}}$ then the single copy of $\mathcal{F}_{\mathsf{WAR}}$ contains at least one corrupted party. If the session of $\mathcal{F}_{\mathsf{WAR}}$ is successful then the adversary issued queries with valid witnesses on behalf of all corrupted parties, the simulated parties output adversarially chosen keys, the simulator issues NewSession queries to $\mathcal{F}_{\mathsf{WAKE}}$ containing valid witnesses on behalf of all corrupted parties, the session of $\mathcal{F}_{\mathsf{WAKE}}$ is successful and the honest parties receive adversarially chosen keys as well. The keys queried to $\mathcal{F}_{\mathsf{WAKE}}$ are exactly those output by the simulated parties, so the distributions are equal.

If the session of $\mathcal{F}_{\mathsf{WAR}}$ fails then the parties all receive independently random keys, the simulator received at least one query from $\mathcal{A}$ which did not verify against the relation and therefore the simulator issued a NewSesssion query to $\mathcal{F}_{\mathsf{WAKE}}$ with an invalid witness. The session of $\mathcal{F}_{\mathsf{WAKE}}$ fails and the parties receive independently random outputs, distributed exactly as in the simulation and the real world.

Claim 3: In the case of **(no corruption, splitting)** *the distributions of outputs between the real and ideal worlds are identical.*

If the adversary did not issue corruption instructions but did split the functionality then each instance of $\mathcal{F}_{\mathsf{WAR}}$ is executed with at least one corrupt participant. The functionality $\mathcal{F}_{\mathsf{WAKE}}$ is running in the all honest case. This is the

case in which the simulator must influence $\mathcal{F}_{\mathsf{WAKE}}$ to allow adversarially chosen outputs.

In the case that at least one instance of $\mathcal{F}_{\mathsf{WAR}}$ is successful then every impersonated party in the execution issued a query with a valid witness and the simulator queries CompromiseSession on behalf of each impersonated party with a valid witness so these parties are all compromised by $\mathcal{F}_{\mathsf{WAKE}}$. In any case the simulated parties receive adversarially chosen keys from $\mathcal{F}_{\mathsf{WAR}}$ and the honest parties receive the same adversarially chosen keys from $\mathcal{F}_{\mathsf{WAKE}}$.

In the case that all instances of $\mathcal{F}_{\mathsf{WAR}}$ failed all simulated honest parties received uniformly random outputs from their instance of $\mathcal{F}_{\mathsf{WAR}}$. There are two cases (a) either the simulator queried CompromiseSession with all invalid witnesses or (b) there was at least one valid witness queried. In case (a) all parties are marked as interrupted by $\mathcal{F}_{\mathsf{WAKE}}$ and receive uniformly random outputs. In case (b) then at least one party is marked as compromised by $\mathcal{F}_{\mathsf{WAKE}}$ and receives adversarially chosen output, while all partners either receive adversarially chosen outputs if $\mathcal{F}_{\mathsf{WAKE}}$ succeeds or uniformly random outputs if $\mathcal{F}_{\mathsf{WAKE}}$ fails. The outputs queried to $\mathcal{F}_{\mathsf{WAKE}}$ are those output by the simulated $\mathcal{F}_{\mathsf{WAR}}$, so the parties all receive from $\mathcal{F}_{\mathsf{WAKE}}$ either the queried key which is uniformly random or a freshly sampled uniformly random key.

Claim 4: In the case of **(no corruption, no splitting)** *the distributions of outputs between the real and ideal worlds are identical.*

In the case that the adversary did not issue any corruption instructions to $\mathcal{F}_{\mathsf{WAKE}}$ and did not split the session of $\mathcal{F}_{\mathsf{WAR}}$ both functionalities are running in the all honest case. In the ideal world the dummy parties issue their queries to $\mathcal{F}_{\mathsf{WAKE}}$ which leaks all but the secret to the simulator. The simulator issues Knowledge queries to $\mathcal{F}_{\mathsf{WAKE}}$ to determine the success or failure of the simulated $\mathcal{F}_{\mathsf{WAR}}$ and leaks everything necessary for a simulation of $\mathcal{F}_{\mathsf{WAR}}$ to $\mathcal{A}$. The simulated $\mathcal{F}_{\mathsf{WAR}}$ exchange is successful iff the $\mathcal{F}_{\mathsf{WAKE}}$ is successful. The functionalities behave the same way in both cases, thus their outputs are distributed identically.

Table 3 details the output of the honest parties and corrupt parties in both Π_{WAR} and $\mathcal{F}_{\mathsf{WAKE}}$ in the cases of corruption, splitting and success of the exchange. In the case of splitting there can be some instances of $\mathcal{F}_{\mathsf{WAR}}$ which are successful and others which fail. In terms of notation: $\mathcal{A}$ indicates that the adversary determines the output to the party ie the functionality forwards the queried key, $(\$)^{|H|}$ indicates that there are H uniformly random keys output, $\$$ indicates that a single random key is output to all of the parties, $\emptyset$ indicates that there are no such parties, and $(\$)^{|H|}$ / $\mathcal{A}$ is a special case which depends on if parties are marked as corrupt or interrupted. Notably, even in the case that the adversary selects the keys and the functionality forwards the queried keys those queried keys are in fact independently random keys. So even in this case the distributions are identical with simulator $\mathcal{S}_{2\text{-}\mathsf{WAKE}}$.

D.4 sl-bb-WAKE achieves UC security

Let Ω be a fully sl-bb-WAKE with straightline black-box extraction, meaning that Ω satisfies confidentiality (Definition 1), simulatability (Definition 2) and

Output Distributions						
Corrupt	Split	Succ	Π_{WAR}		$\mathcal{F}_{\mathsf{WAR}}$	
			H	C	H	C
T	T	T	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$
		F	$(\$)^{\lvert H\rvert}$		$(\$)^{\lvert H\rvert}$ / $\mathcal{A}$	
T	F	T	$\mathcal{A}$		$\mathcal{A}$	
T	F	F	$(\$)^{\lvert H\rvert}$		$(\$)^{\lvert H\rvert}$	
F	T	T	$\mathcal{A}$	$\emptyset$	$\mathcal{A}$	$\emptyset$
		F	$(\$)^{\lvert H\rvert}$		$(\$)^{\lvert H\rvert}$ / $\mathcal{A}$	
F	F	T	$\$$		$\$$	
F	F	F	$(\$)^{\lvert H\rvert}$		$(\$)^{\lvert H\rvert}$	

Table 3. Table detailing outputs in proof of Theorem 6.

straightline black-box authenticity (Definition 3). By definition Ω has setup algorithm SetUp and simulator $\mathcal{S}_\Omega = (\mathsf{SimSetUp}, \mathsf{Sim})$ consisting of algorithms which output the simulated setup and simulated messages, respectively.

Consider the protocol $\Pi^{\Omega}_{\mathsf{UC\text{-}W}}$ in Figure 13. Each party P in the execution receives input from $\mathcal{Z}$ and queries $\mathcal{F}_{\mathsf{CRS}}$ to receive the public parameters pp for Ω, as the distribution of the CRS is set to be the distribution of $\Omega.\mathsf{SetUp}(1^\lambda, \mathcal{R})$. Party P then executes protocol Ω exactly as described. When Ω finalizes the parties have either accepted or rejected the session, which is reflected in the party's internal variable acc_P. In the case that P has accepted and has generated a key sk_P according to Ω then P outputs this session key. In the case that P has not accepted it outputs a random value r.

The protocol is executed over an *unauthenticated, asynchronous network*, therefore the adversary is permitted to arbitrarily delay messages, or inject and deliver entirely unrelated messages. This is modeled by requiring the ITIs that are functioning as channels in the network leak all messages to the adversary, only deliver messages upon adversarial instruction, and allow these messages to be replaced, duplicated, reordered, dropped, etc, which is referred to as the *unauthenticated setting* in Theorem 1.

Theorem 7. *If Ω is an* sl-bb-WAKE *protocol with straightline black-box extraction then protocol $\Pi^{\Omega}_{\mathsf{UC\text{-}W}}$ UC-realizes $\mathcal{F}_{\mathsf{WAKE}}$ in the $\mathcal{F}_{\mathsf{CRS}}$-hybrid model in the presence of static, malicious adversaries in the unauthenticated setting.*

Proof. Let Ω be an sl-bb-WAKE protocol between set of parties $\mathcal{P}$. For every PPT adversary $\mathcal{A}$ there exists a PPT simulator $\mathcal{S}_{\mathsf{UC\text{-}W}}$ such that no PPT environment $\mathcal{Z}$ can distinguish between a real world interaction with $(\mathcal{A}, \Pi^{\Omega}_{\mathsf{UC\text{-}W}})$ and an ideal world interaction with $(\mathcal{S}_{\mathsf{UC\text{-}W}}, \mathcal{F}_{\mathsf{WAKE}})$. This simulator is described in Figure 14.

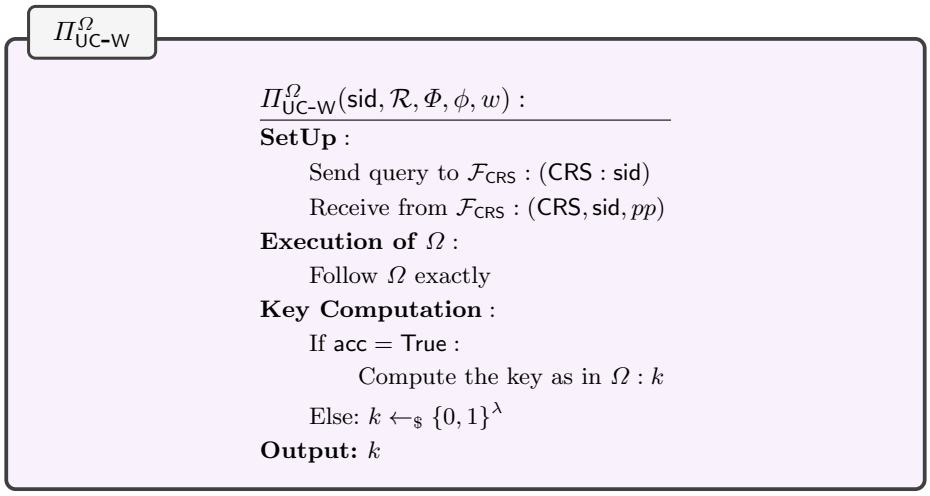

Fig. 13. A UC-secure WAKE protocol from sl-bb-WAKE Ω.

The simulator $\mathcal{S}_{\mathsf{UC\text{-}W}}$ internally invokes the real world adversary $\mathcal{A}$ and simulates a copy of the uncorrupted parties internally, along with a copy of the ideal functionality $\mathcal{F}_{\mathsf{CRS}}$. The simulator $\mathcal{S}_{\mathsf{UC\text{-}W}}$ forwards messages between $\mathcal{Z}$ and $\mathcal{A}$ reliably. As the system is in the unauthenticated setting the adversary may choose to drop or deliver arbitrary messages throughout the protocol in both worlds. What proceeds is a sequence of indistinguishable games starting in the real world and ending in the ideal world.

Game $\mathcal{G}_0$: Real Protocol

This is the real world. The protocol $\Pi^{\Omega}_{\mathsf{UC\text{-}W}}$ is run by the environment $\mathcal{Z}$ with party set $\mathcal{P}$ and adversary $\mathcal{A}$, all having direct access to the ideal functionality $\mathcal{F}_{\mathsf{CRS}}$.

Game $\mathcal{G}_1$: Ideal Grouping

Parties $\mathcal{P}$, functionality $\mathcal{F}_{\mathsf{CRS}}$ and $\mathcal{A}$ are grouped into a single machine called the simulator $\mathcal{S}_{\mathsf{UC\text{-}W}}$. An ideal functionality node and dummy party nodes are introduced. The dummy party nodes receive input from $\mathcal{Z}$ and issue queries to $\mathcal{F}_{\mathsf{WAKE}}$ from those inputs, which are leaked in their entirety to $\mathcal{S}_{\mathsf{UC\text{-}W}}$. The simulator simulates all honest parties with knowledge of these complete queries while the adversary controls any corrupt parties.

$\mathcal{G}_1$ is indistinguishable from $\mathcal{G}_0$: This grouping and labeling does not change the view of the environment. The protocol messages, along with the outputs, are generated identically and thus are identically distributed in both games.

Game $\mathcal{G}_2$: Record Keeping, Simulated $\mathcal{F}_{\mathsf{CRS}}$, Knowledge Queries

$\mathcal{S}_{\mathsf{UC\text{-}W}}$

$\mathcal{S}_{\mathsf{UC\text{-}W}}$ initializes the real world adversary $\mathcal{A}$ and reliably forwards all messages between $\mathcal{Z}$ and $\mathcal{A}$. Corruption instructions from $\mathcal{A}$ are forwarded to $\mathcal{F}_{\mathsf{WAKE}}$ and obeyed by $\mathcal{S}_{\mathsf{UC\text{-}W}}$ if they are issued prior to receipt of any $\mathsf{NewSession}$ query leakage.

- **CRS Functionality:**
  - Set the distribution $\mathcal{D} := \Omega.\mathsf{SimSetUp}(1^\lambda, \mathcal{R})$
  - Upon receipt of the first query $(\mathsf{CRS} : \mathsf{sid})$ from corrupt party P: Parse $\mathsf{sid} = (\mathsf{sid}', \mathcal{R})$, sample $(pp, \tau) \leftarrow \Omega.\mathsf{SimSetUp}(1^\lambda, \mathcal{R})$, and send a delayed public output $(\mathsf{CRS}, \mathsf{sid}, pp)$ to P
  - Upon all subsequent queries $(\mathsf{CRS} : \mathsf{sid})$ from corrupt party P: Parse $\mathsf{sid} = (\mathsf{sid}', \mathcal{R})$ and send a delayed public output $(\mathsf{CRS}, \mathsf{sid}, pp)$ to P
  - For each honest party P_i: send a delayed public output $(\mathsf{CRS}, \mathsf{sid}, pp)$ to P_i
- **Simulating** Ω: query $(\mathsf{Knowledge} : \mathsf{sid}, P_i)$ to $\mathcal{F}_{\mathsf{WAKE}}$ and receive $(\mathsf{Knldg}, \mathsf{sid}, P_i, b_i)$ for each honest P_i
  - For all P_i such that $b_i = 1$: use the CRS trapdoor τ and the simulator algorithm Sim from Ω to generate all messages on behalf of P_i exactly as in Ω
  - For all P_i such that $b_i = 0$: set $sk_i \leftarrow \{0,1\}^\lambda$
  - Keys sk_i are computed according to Ω
- **Adversarial Queries:**
  - If there is some honest P_i such that $\mathsf{acc}_{P_i} = \mathsf{True}$ then for all corrupt P_j
    - Extract a witness from the transcript of the accepting P_i: $w'_j \leftarrow \mathcal{E}(pp, \tau, \mathsf{trans}_{P_i}, \phi_j)$
    - Query $(\mathsf{NewSession} : \mathsf{sid}, w'_j, \Phi, \phi_j)$ to $\mathcal{F}_{\mathsf{WAKE}}$
  - Else if the parties terminated but no honest party accepted then for each corrupt P_j
    - Sample $w'_j \leftarrow_\$ \{0,1\}^{|\phi_j|}$
    - Query $(\mathsf{NewSession}, \mathsf{sid}, w'_j, \Phi, \phi_j)$ to $\mathcal{F}_{\mathsf{WAKE}}$
  - Else: halt the simulation
- **New Key Queries:** For all honest P_i
  - Query $(\mathsf{NewKey} : \mathsf{sid}, P_i, sk_i)$ to $\mathcal{F}_{\mathsf{WAKE}}$

Fig. 14. The simulator for protocol $\Pi^{\Omega}_{\mathsf{UC\text{-}W}}$.

The ideal functionality now keeps records for the queries received as seen in the description in Figure 7. The functionality responds to these knowledge queries as described in Figure 7. The simulator simulates $\mathcal{F}_{\mathsf{CRS}}$ exactly as described in Figure 14 but instead uses the distribution $\mathcal{D} = \Omega.\mathsf{SimSetUp}(1^\lambda, rel)$ which outputs pp along with trapdoor τ. The simulator issues a knowledge query to $\mathcal{F}_{\mathsf{WAKE}}$ for each honest party and receives output bit b_i. The simulator then uses the simulator $\Omega.\mathsf{Sim}$ to generate messages on behalf of honest parties P_i with $b_i = 1$ instead of honestly generating messages with the witnesses leaked by $\mathcal{F}_{\mathsf{WAKE}}$. When $b_i = 0$ the simulator sets $sk_i \leftarrow_\$ \{0,1\}^\lambda$ and halts on behalf of that party (stops simulating that party).

$\mathcal{G}_2$ is indistinguishable from $\mathcal{G}_1$: The public parameters and transcript messages generated by $\mathcal{S}_{\mathsf{UC\text{-}W}}$ are generated using the simulator for Ω, meaning

that they are indistinguishable from those output by SetUp and messages sent by real parties. This is due to the simulatability (zero knowledge) property of WAKE.

Any environment $\mathcal{Z}$ that could distinguish between $\mathcal{G}_2$ and $\mathcal{G}_1$ would imply an adversary $\mathcal{B}$ against the simulatability of Ω. The environment sees the entire transcript of the protocol, Adversary $\mathcal{B}$ in the experiment in Figure 1 receives the public parameters pp_b and initializes $\mathcal{Z}$, then emulates the simulator in $\mathcal{G}_2$ using pp_b as the public parameters output by the simulated $\mathcal{F}_{\mathsf{CRS}}$. Adversary $\mathcal{B}$ issues a SetKeys query for each honest party using the leaked associated statement and witness. Then, $\mathcal{B}$ engages in the protocol Ω with $\mathcal{A}$ controlling the corrupted parties and uses his Send^b oracle to construct the messages on behalf of the honest parties. The view of $\mathcal{Z}$ in when $b = 1$ corresponds exactly to the view of $\mathcal{Z}$ in $\mathcal{G}_1$ and otherwise to the view of $\mathcal{Z}$ in $\mathcal{G}_2$. Assume that $\mathcal{Z}$ outputs the bit $\bar{b} = 0$ for a guess of $\mathcal{G}_1$. Then the end of the protocol $\mathcal{B}$ issues a Reveal query for each honest party and outputs the keys to $\mathcal{Z}$ and outputs whichever bit $\mathcal{Z}$ does. If $\mathcal{Z}$ has a non-negligible advantage in distinguishing the games then $\mathcal{B}$ has a non-negligible advantage in breaking the simulatability of Ω.

Game $\mathcal{G}_3$: Ideal Functionality Stops Leaking Secrets, Simulator Submits NewSession Queries

The ideal functionality $\mathcal{F}_{\mathsf{WAKE}}$ no longer forwards the queries submitted by the dummy parties in their entirety, no longer leaking the witnesses to $\mathcal{S}_{\mathsf{UC\text{-}W}}$. The simulator submits NewSession queries on behalf of the corrupt parties in the case that an honest party accepts the session, using the witnesses extracted from the transcript of the accepting honest party. In the case that extraction fails the simulator outputs $\mathsf{error} - E$ and halts.

$\mathcal{G}_3$ is indistinguishable from $\mathcal{G}_2$: The environment cannot see the queries by $\mathcal{S}_{\mathsf{UC\text{-}W}}$ or leakage to $\mathcal{S}_{\mathsf{UC\text{-}W}}$ so there is no change in the view of the environment in the case that the error $\mathsf{error} - E$ does not happen.

Let E be the event in which the extractor fails to output a witness for some corrupt P' from the transcript of an honest and accepting party P: trans_P. This is the only case in which the extractor is run by the simulator. By definition the failure of the extractor violates the authenticity requirement seen in Definition 3. Let $\Phi, \mathcal{R}$ be the set of statements and relation for which event E happens with nonnegligible probability. An adversary $\mathcal{B}$ in the authenticity game invokes the adversary $(\mathcal{A}, \mathcal{Z})$ providing input Φ and emulates the simulator in the above game, providing the public parameters pp. The adversary $\mathcal{B}$ replies to all messages using the Send oracle. When event E happens with accepting party P and corrupt P' adversary $\mathcal{B}$ outputs challenge $(P, 0, P')$. As P' is corrupt and $\mathcal{B}$ provided all messages sent by P' to the oracle as output by $\mathcal{A}$, P' is in the impersonation set of P. The extractor in the authenticity game will also fail, and $\mathcal{B}$ will win the experiment. This adversary $\mathcal{B}$ has advantage equal to the probability of

event E. Therefore, event E happens with negligible probability under the authenticity of Ω.

Game $\mathcal{G}_4$: Functionality Stops Forwarding Keys in All Honest Case

The ideal functionality $\mathcal{F}_{\mathsf{WAKE}}$ no longer forwards the keys sent from $\mathcal{S}_{\mathsf{UC\text{-}W}}$ to the dummy parties to then be output to $\mathcal{Z}$. The functionality now samples the keys issued to these parties when all parties are honest, in both the cases of success and failure.

$\mathcal{G}_4$ is indistinguishable from $\mathcal{G}_3$: In the case of failure the honest parties did not accept. The keys output by the functionality to the dummy parties are uniformly and independently random keys, distributed exactly as those generated by honest parties participating in the protocol $\Pi^{\Omega}_{\mathsf{UC\text{-}W}}$ in the case of failure. In the case of success the keys output by $\mathcal{F}_{\mathsf{WAKE}}$ were: in $\mathcal{G}_3$ the real keys output by accepting parties in an execution of Ω and, in $\mathcal{G}_4$ a random string of appropriate size. These two games are indistinguishable in the case of success because any $\mathcal{Z}$ which can distinguish the keys would be an adversary against the confidentiality of Ω as in Definition 1.

An environment $\mathcal{Z}$ which can distinguish between the two games can be used to build an environment $\mathcal{B}$ against the confidentiality experiment in Figure 1 of Ω. Let $(\Phi, \mathcal{R}, W)$ be the inputs provided by $\mathcal{Z}$ on which it has non-negligible advantage in distinguishing the games. Assume that all statements in Φ are satisfied by the inputs provided by $\mathcal{Z}$ otherwise the simulator would halt on behalf of some party in both games. Adversary $\mathcal{B}$ receives the public parameters pp, statement set Φ and witnesses W, then emulates the simulator of $\mathcal{G}_3$ using the public parameters pp. As the parties are all honest $\mathcal{B}$ makes a single query to $\mathsf{Execute} \rightarrow T$ for the transcript of Ω and outputs $(P, 0)$ as the challenge for some $P \in \mathcal{P}$. Adversary $\mathcal{B}$ receives key k_b and provides $\mathcal{Z}$ with (T, k_b) to complete the view of $\mathcal{Z}$. If $b = 1$ then the view of $\mathcal{Z}$ is that $\mathcal{G}_3$ and if $b = 0$ then the view of $\mathcal{Z}$ is that in $\mathcal{G}_4$. Assume $\mathcal{Z}$ outputs a guess bit $\bar{b}$ which is 1 if the guess is $\mathcal{G}_3$. Adversary $\mathcal{B}$ outputs this same bit. The advantage of $\mathcal{B}$ in the confidentiality game is equal to that of $\mathcal{Z}$ in distinguishing between the games.

Game $\mathcal{G}_5$: Simulator Submits NewKey Queries, Functionality Manages Keys in All Cases

The simulator formally submits NewKey queries for all honest parties as detailed in Figure 14 using the key generated by the protocol. The functionality forwards these keys as output in the case that the party is corrupt or the party had a corrupt partner in a successful key exchange. In all other (not all honest) cases the functionality outputs a uniformly random string with length of the security parameter as the key.

$\mathcal{G}_5$ is indistinguishable from $\mathcal{G}_4$: The environment does not see the queries. The functionality was already forwarding the keys in the case of a corrupt

party or a partner of a corrupt party participating in a successful exchange. In the case of failure with corrupt partners, the honest parties already output random keys which were then queried by $\mathcal{S}_{\mathsf{UC\text{-}W}}$ and forwarded by $\mathcal{F}_{\mathsf{WAKE}}$. Therefore the output is distributed the same.

SEQUENCE OF GAMES							
	Game	$\mathcal{F}_{\mathsf{WAKE}}$				$\mathcal{S}_{\mathsf{UC\text{-}W}}$	Property Used
		NewSession	KnIdg	CompSes	NewKey		
	$\mathcal{G}_0$	N/A: this is the real protocol					
	$\mathcal{G}_1$	forwards queries to $\mathcal{S}$			forwards outputs to $\mathcal{P}$	sends messages for honest parties	N/A
	$\mathcal{G}_2$	record keeping	responds to queries			simulates $\mathcal{F}_{\mathsf{CRS}}$ & simulates parties with Sim	$\mathcal{F}_{\mathsf{WAKE}}$ (simulatability)
	$\mathcal{G}_3$	no longer leaks witnesses				submits corrupt NewSession queries	$\mathcal{F}_{\mathsf{WAKE}}$ (authenticity)
	$\mathcal{G}_4$				follows instructions for keys in all honest case		$\mathcal{F}_{\mathsf{WAKE}}$ (confidentiality)
	$\mathcal{G}_5$			follows instructions for keys in all cases		submits formal key queries	N/A

Table 4. Summary of hybrids for Theorem 1.

E Proof of Our Witness-Authentication Compiler

Theorem 8. *If Π is a passively secure* WAKE *protocol satisfying confidentiality only and Σ is a* nBB-s-SE*-secure* SOK *then Π^*, the output of running compiler $\mathcal{C}^{\Sigma}_{\mathsf{WAKE}}$ on Π as presented in Figure 2, is an* nBB*-fully secure witness-authenticated key exchange protocol.*

Theorem 8 has been split into Lemmas 1, 2 and 3 which assert the confidentiality, authenticity and simulatability of the compiled protocol, respectively. Together, these lemmas imply the theorem.

Lemma 1. *The compiled protocol Π^* as shown in Figure 2 satisfies confidentiality (Definition 1).*

Proof. The confidentiality of Π^* reduces to the confidentiality of the underlying scheme Π. An adversary $\mathcal{A}_{\Pi^*}$ against the confidentiality of the compiled protocol Π^* implies an adversary $\mathcal{A}_\Pi$ against the the confidentiality of the passively secure protocol Π. Let $\mathcal{R}$, Φ, $\mathcal{D}_\Phi$ be such that the adversary $\mathcal{A}_{\Pi^*}$ has a nonnegligable advantage in the confidentiality experiment seen in Figure 1. In Figure 15 is the adversary $\mathcal{A}_\Pi$ against Π with the same parameters. The advantage of $\mathcal{A}_\Pi$ in the confidentiality game for protocol Π is equal to that of $\mathcal{A}_{\Pi^*}$ in the confidentiality game for protocol Π^*.

$\mathcal{A}_\Pi$

$\mathcal{A}_\Pi^{\mathsf{Execute}_\Pi, \mathsf{Reveal}_\Pi}(pp_\Pi, \Phi, W)$

Parse pp_Π which contains $1^\lambda, \mathcal{R}$

$pp_\Sigma \leftarrow \Sigma.\mathsf{SSetup}(1^\lambda, R)$

$pp_{\Pi^*} \leftarrow (pp_\Pi, pp_\Sigma)$

Run $\mathcal{A}_{\Pi^*}(pp_{\Pi^*}, \Phi, W)$:

- $\mathsf{Execute}_{\Pi^*}(P_{i_1}, j_1, \ldots, P_{i_k}, j_k)$ called by $\mathcal{A}_{\Pi^*}$:
  - $T \leftarrow \mathsf{Execute}_\Pi(P_{i_1}, j_1, \ldots, P_{i_k}, j_k)$
  - For each participant $\Pi^j_{P_i}$ sample $\mathsf{nonce}^j_i \leftarrow_\$ \{0,1\}^*$
  - For each message m sent by $\Pi^j_{P_i}$ in round k in T :
    - $m^* \leftarrow (P_i || k || m || \mathsf{nonces}^j_{P_i})$
    - $\sigma^* \leftarrow \Sigma.\mathsf{SSign}(pp_\Sigma, \phi_i, w_i, m^*)$
  - Return the resulting transcript T^*
- $\mathsf{Reveal}(P, j)$ called by $\mathcal{A}_{\Pi^*}$:
  - Return $\mathsf{Reveal}_\Pi(P, j)$
- Challenge (P, i) output by $\mathcal{A}_{\Pi^*}$:
  - Output (P, i) and receive key k_b

Run $\mathcal{A}_{\Pi^*}(k_b)$:

- Respond to any queries as above
- Bit b' output by$\mathcal{A}_{\Pi^*}$:
  - Output b'

Fig. 15. Reduction from the confidentiality of $\Pi^* = \mathcal{C}^\Sigma_{\mathsf{WAKE}}(\Pi)$ to that of Π.

Lemma 2. *The compiled protocol Π^* as shown in Figure 2 satisfies authenticity with non-black-box extraction (Definition 3).*

Proof. The authenticity of Π^* reduces to the strong simulation extractability of Σ with non-black-box extractor as in Appendix B.3.

An adversary $\mathcal{A}_{\Pi^*}$ against the authenticity experiment seen in Figure 1 implies the existence of an adversary $\mathcal{A}_\Sigma$ against the simulation-extractability of the signature, more specifically if no extractor $\mathcal{E}_{\mathcal{A}_{\Pi^*}}$ running on the view of $\mathcal{A}_{\Pi^*}$ can efficiently output a witness to the statement associated to the impersonated party P' then no extractor $\mathcal{E}_{\mathcal{A}_\Sigma}$ can efficiently output a witness to the statement associated to the forgery output by $\mathcal{A}_\Sigma$. Let $\mathcal{R}, \Phi$ be a relation and statement vector, respectively, with respect to which $\mathcal{A}_{\Pi^*}$ has nonnegligible advantage in the experiment seen in Figure 1, then $\mathcal{A}_\Sigma$ presented in Figure 16 has nonnegligible advantage in the extraction experiment for the signature with respect to the same relation $\mathcal{R}$.

The adversary $\mathcal{A}_\Sigma$, in the case that the challenge instance Π_P^i selected by $\mathcal{A}_{\Pi^*}$ accepts and that the party P' was impersonated to the challenge instance ie $P' \in \mathcal{I}(P, i)$, outputs a triple $(\phi_{P'}, m^*, \sigma^*)$ for message $m = (m^*, \sigma^*)$ appearing in sid_P^i but not $\mathsf{sid}_{P'}^j$ for j such that $\mathsf{nonces}_P^i = \mathsf{nonces}_{P'}^j$. This message necessarily exists because $\mathcal{A}_{\Pi^*}$ is necessarily not forwarding on the instance Π_P^i and, in order for $P' \in \mathcal{I}(P, i)$ there must have been a call to $\mathsf{Send}(P, i, m)$ without m being output by a corresponding call to $\mathsf{Send}(\bar{P}, \bar{j}, \cdot)$ with $\phi_{\bar{P}} = \phi_{P'}$. If that message were the nonce of round 1 then Π_P^i would reject. Thus, m must be a message exchanged in a different round and therefore contains a signature. The signature appearing in m is not a signature generated as $\mathsf{SSimSign}(\phi_{P'}, m)$ because it was not output as a query to Send for a party authenticating with respect to $\phi_{P'}$ and thus does not appear in the recorded set of queries. Additionally, in order for $\mathsf{acc}_P^i = 1$ that signature must verify with respect to the statement $\phi_{P'}$. We can claim that either the returned triple is a valid challenge for the simulation-extractability game (with non-negligible probability), or $\mathcal{A}_{\Pi^*}$ is not admissible.

An extractor $\mathcal{E}_\Sigma$ which can output a witness to the forged signature implies an extractor $\mathcal{E}_{\mathcal{A}_{\Pi^*}}$ which can also output a witness on input the view $\mathsf{view}_{\mathcal{A}_{\Pi^*}}$. There is a polynomial time transformation from the view of $\mathcal{A}_{\Pi^*}$ to that of $\mathcal{A}_\Sigma$, which is called T. This extractor can also be seen in Figure 16. Thus, if there is no extractor for $\mathcal{A}_{\Pi^*}$ then there is no extractor for $\mathcal{A}_\Sigma$.

Lemma 3. *The compiled protocol Π^* in Figure 2 satisfies simulatability (Definition 2).*

Proof. The simulatability of Π^* reduces to the perfect simulatability of the signature Σ, as defined in Appendix B.3. An adversary $\mathcal{A}_{\Pi^*}$ against the WAKE simulatability implies an adversary $\mathcal{A}_\Sigma$ against the perfect simulatability of the signature of knowledge Σ. This reduction can be found in Figure 18.

A simulator, the $\mathsf{SimSetup}$ and $\mathsf{SimSend}$ algorithms for Π^*, are presented in Figure 17.

F Three-Round WAKE for Groups

This section shows how to instantiate the compiler presented in Section 3 on a passively secure WAKE protocol to obtain a three-round protocol for group WAKE.

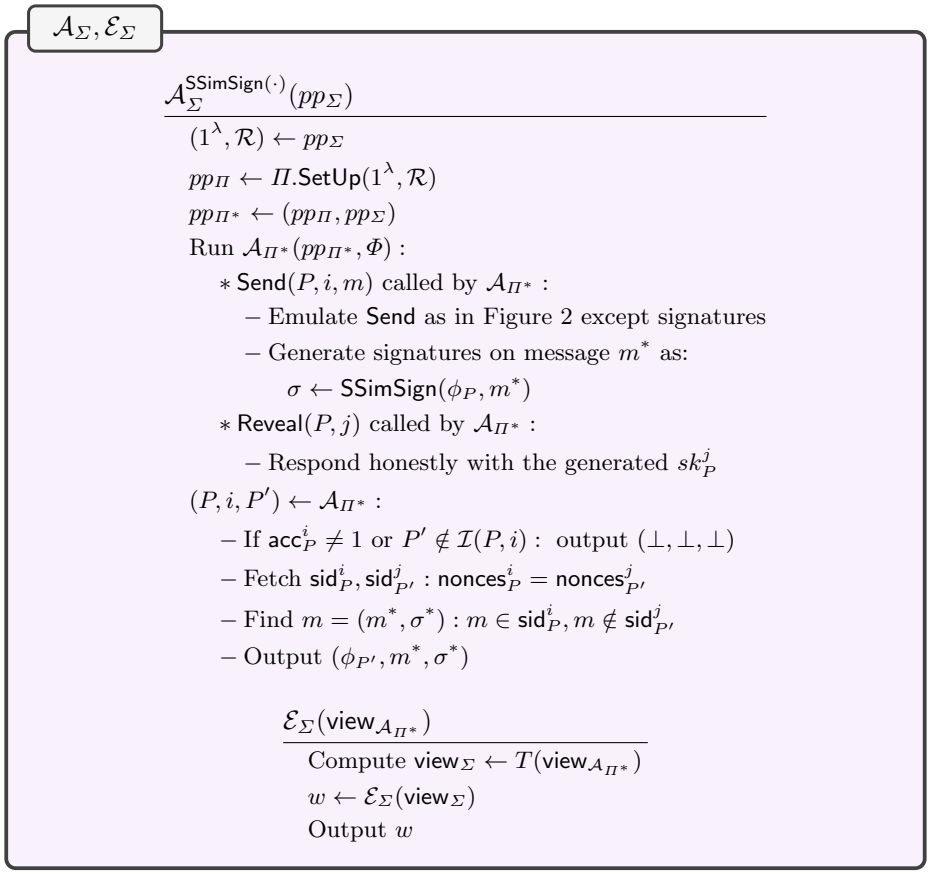

Fig. 16. Reduction from the nBB-auth of $\Pi^* = \mathcal{C}^\Sigma_{\mathsf{WAKE}}(\Pi)$ to sim of Σ.

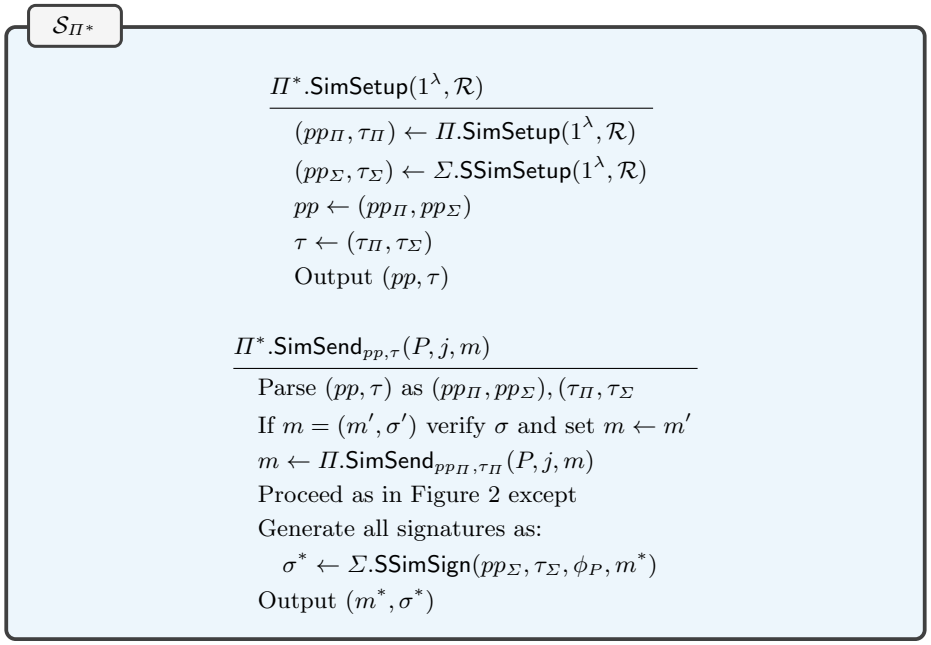

Fig. 17. The simulator for the WAKE protocol $\Pi^* = \mathcal{C}^{\Sigma}_{\mathsf{WAKE}}(\Pi)$.

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 $x, y \in \mathbb{Z}_q^*$, $z \in \mathbb{Z}_q^*$. Consider an infinite 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:

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

The DDH assumption states that for all PPT $\mathcal{A}$, the advantage $\mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_\lambda,\mathcal{A}}(\lambda)$ is negligible.[18]

Presented in Figure 19 is Π_{BDKE}, a passively secure protocol for (standard) group key exchange. This protocol was constructed by Burmester and Desmedt [16] and was adapted and proven secure under the Decisional Diffie-Hellman assumption by Katz and Yung [31]. Running the compiler seen in Figure 2 on Π_{BDKE} yields a three round, fully secure WAKE protocol. This is formally stated in Theorem 1.

The participant set is denoted $\mathcal{P} = \{P_i\}_{i=1}^n$ with participants indexed mod n, such that $P_n = P_0$ and $P_{n+1} = P_1$. The inputs $(\mathbb{G}, g, q)$ are generated beforehand but can also be generated by a single player at the expense of an additional round.

[18] The variant of DDH described in [31] excludes the possibility that $z = xy$ for simplicity, but this modificiation is mentioned to be of essentially no consequence.

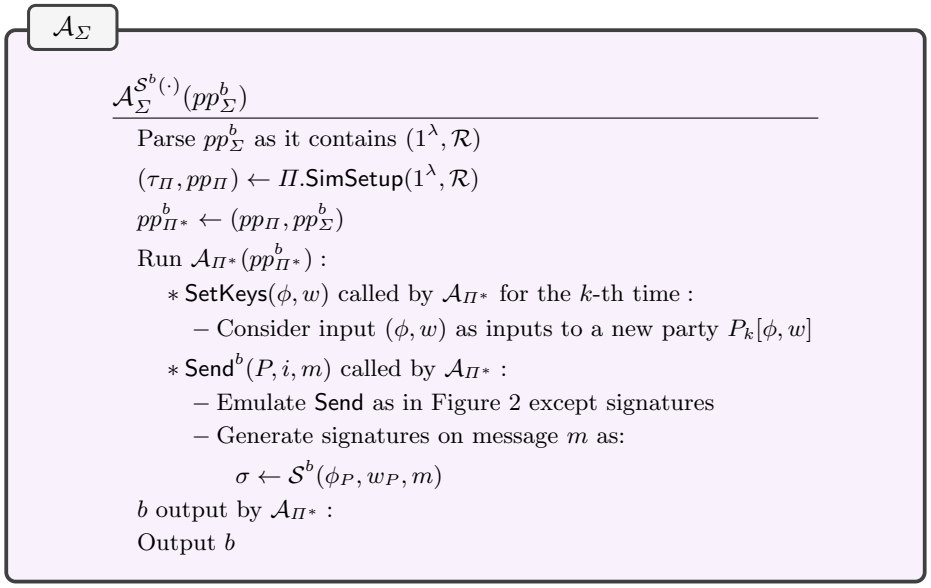

Fig. 18. Reduction from the simulatability of $\Pi^* = \mathcal{C}^\Sigma_{\mathsf{WAKE}}(\Pi)$ to that of Σ.

The communication style is referred to as broadcast 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, sending a message to the entire group via these links is referred to as broadcasting.

The session key generated by the protocol in Fig. 19 is $\mathsf{sk} = g^{x_1x_2+x_2x_3+\cdots+x_nx_1}$ and is common to all participants. Π_{BDKE} achieves passive security for group key exchanges; the protocol is secure against an eavesdropping adversary in a real-or-random confidentiality experiment. Passive security, along with forward security, is proven in [31].

As the definition of passively secure key exchange coincides with passively secure WAKE, the reader is pointed to the definition of confidentiality seen in Definition 1 which, with an empty statement set and witness set, is essentially the confidentiality experiment for standard (group) key exchange. More explicitly, the adversary is given the parameters of the key exchange, an oracle which generates transcripts to the key exchange and an oracle which reveals keys corresponding to the transcripts. The adversary outputs a challenge transcript and is given either the real key or a random string. In the case that the adversary guesses correctly if the key is real or random then the experiment outputs 1 and the adversary wins. The exchange is considered confidential, ie passively secure, if for all PPT adversaries their advantage in the experiment is negligible.

Theorem 9 (Confidentiality of Π_{BDKE} [31]). *The group key exchange protocol Π_{BDKE} seen in Figure 19 satisfies confidentiality under the* DDH *assumption.*

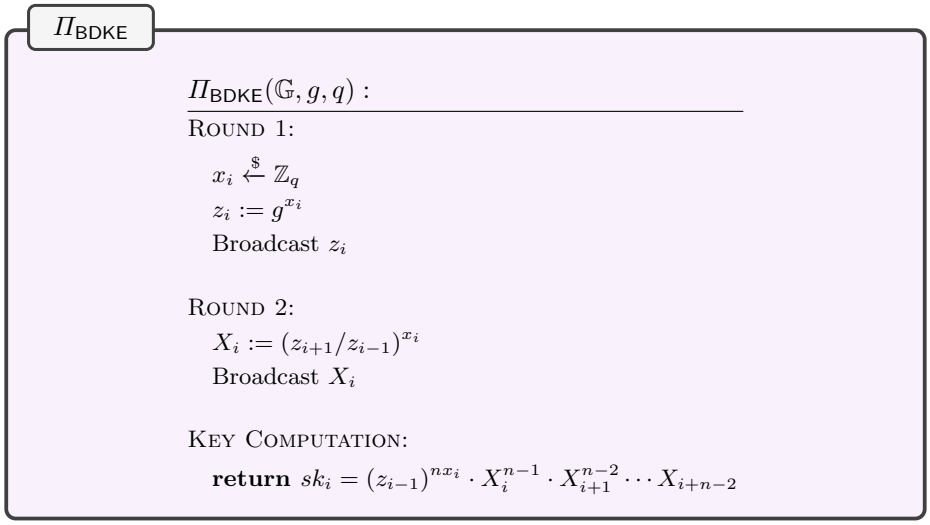

Fig. 19. The Burmester-Desmedt passively secure group key exchange protocol.

For concreteness, let Σ be the signature of knowledge detailed in [30]. We apply the compiler detailed in Section 3 to Π_{BDKE}, using Σ 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 $\mathcal{C}^{\Sigma}_{\mathsf{WAKE}}$ (Figure 2) on Π_{BDKE} (Figure 19) with any correct, simulatable and* nBB-s-SE*-secure signature of knowledge Σ yields a three round fully secure* WAKE*.*

G Two Round WAKE Security Proof

Theorem 10. *For any NP relation $\mathcal{R}$ protocol $\Pi_{\text{2-WAKE}}$ securely UC-realizes $s\mathcal{F}_{\mathsf{WAR}}$ in the $(\mathcal{F}_{\mathsf{CRS}}, \mathcal{F}_{\mathsf{SOK}})$-hybrid model in the presence of malicious adaptive adversaries in the unauthenticated asynchronous setting.*

Proof. Consider an adversary $\mathcal{A}$ corrupting an arbitrary number of parties and adaptively splitting sessions. There exists for each efficient adversary $\mathcal{A}$ a simulator $\mathcal{S}_{\text{2-WAKE}}$ such that no efficient environment $\mathcal{Z}$ can distinguish between an execution involving $(\Pi_{\text{2-WAKE}}, \mathcal{A})$ from one involving $(s\mathcal{F}_{\mathsf{WAR}}, \mathcal{S}_{\text{2-WAKE}})$. The simulator is presented in Figure 20.

The simulator $\mathcal{S}_{\text{2-WAKE}}$ must simulate three things: (1) $\mathcal{F}_{\mathsf{SOK}}$ and $\mathcal{F}_{\mathsf{CRS}}$, (2) the messages sent by any honest simulated parties in $\Pi_{\text{2-WAKE}}$, (3) queries to both $s\mathcal{F}_{\mathsf{WAR}}$ and $\mathcal{F}_{\mathsf{WAR}}$ so that these functionalities issue the appropriate outputs. The proof proceeds with a sequence of indistinguishable games as illustrated in Table 5, starting in the real world and ending in the ideal world with simulator $\mathcal{S}_{\text{2-WAKE}}$.

Following this is an argument that the errors of the $\mathcal{F}_{\mathsf{SOK}}$ happen with the same probability in both the real and ideal worlds.

Game $\mathcal{G}_0$: Real Protocol

This is the real world. The protocol $\Pi_{\text{2-WAKE}}$ is initiated by the environment $\mathcal{Z}$ with party set $\mathcal{P} = \{P_1, \ldots, P_n\}$ and (wlog dummy) adversary $\mathcal{A}$, all having direct access to the ideal functionalities $\mathcal{F}_{\mathsf{SOK}}$ and $\mathcal{F}_{\mathsf{CRS}}$.

Game $\mathcal{G}_1$: Ideal Grouping & Splitting Queries

Group the adversary $\mathcal{A}$ and the parties $\mathcal{P}$ into a single machine referred to as the simulator $\mathcal{S}_{\text{2-WAKE}}$. Add in dummy party and functionality nodes such that each channel between the environment and the simulated party is instead linked first to the corresponding dummy party, which then has a channel to the split functionality, then to the simulator. Upon activation with inputs $(\mathsf{sid}, \mathcal{R}, \Phi, \phi, w)$ from $\mathcal{Z}$ the honest dummy parties issue Init queries to $s\mathcal{F}_{\mathsf{WAR}}$ as $(\mathsf{Init} : \mathsf{sid}, \Phi)$ which are forwarded exactly as queried to $\mathcal{S}_{\text{2-WAKE}}$. The simulator begins with round 1 messages from $\Pi_{\text{2-WAKE}}$, sending all messages through $\mathcal{A}$ and each simulated party P receives a set of messages $\{(P', z_{P'}^{(i)})\}_{P' \neq P}$. For each simulated party P the simulator constructs the string of verification keys as $\mathsf{vk}_P = (P_1 || z_1^{(P)} || P_2 || z_2^{(P)} || \cdots || P_n || z_n^{(P)})$ with parties ordered by their party identity. These strings define the session identifiers for the split session. The simulator then queries a split to $s\mathcal{F}_{\mathsf{WAR}}$ with the vk_P received by each party as $(\mathsf{Init} : \mathsf{sid}, P, H, \mathsf{vk}_P, \Phi)$ where H is defined to be the set of other honest parties which have an equal set of verification keys. The simulator then receives the leakage of $(\mathsf{NewSession} : \mathsf{sid}, \phi, P)$ for each honest party P and continues to simulate the honest parties. When keys are generated the functionality forwards these keys to the dummy parties.

$\mathcal{G}_1$ is indistinguishable from $\mathcal{G}_0$: The inputs are forwarded to the simulator as is, and the outputs are forwarded from the functionality nodes directly to the dummy parties thus the inputs, transcript and outputs remain the same. The grouping and labeling does not change the view of the environment, and the environment does not know how man instances of $\mathcal{F}_{\mathsf{WAR}}$ are invoked.

Game $\mathcal{G}_2$: Record Keeping by $s\mathcal{F}_{\mathsf{WAR}}$ & New Key Queries

The functionality node for $s\mathcal{F}_{\mathsf{WAR}}$ now keeps records of the Init queries made by the parties and performs the required checks when queried prior to initiating new instances of $\mathcal{F}_{\mathsf{WAR}}$ as detailed in Figure 9. The simulator now issues NewKey queries for all parties in H to each instance of $\mathcal{F}_{\mathsf{WAR}}^{H}$. The keys as queried are output to the queried parties.

$\mathcal{G}_2$ is indistinguishable from $\mathcal{G}_1$: Internal record keeping is not visible to the environment. The same keys are output to the parties.

Game $\mathcal{G}_3$: Record Keeping by $\mathcal{F}_{\mathsf{WAR}}$ & Simulated $\mathcal{F}_{\mathsf{SOK}}, \mathcal{F}_{\mathsf{CRS}}$

The functionality nodes for $\mathcal{F}_{\mathsf{WAR}}$ now keep records of the NewSession queries made by the parties, as detailed in Figure 10. The simulator $\mathcal{S}_{\text{2-WAKE}}$ now simulates the ideal functionality $\mathcal{F}_{\mathsf{CRS}}$, so this node is added to the simulator grouping mentioned in $\mathcal{G}_1$. $\mathcal{S}_{\text{2-WAKE}}$ first samples from the distribution $(\mathbb{G}, g, q) \leftarrow \mathcal{D}$ over groups, as described, and responds to each query from corrupt P_j with the delayed public output $(\mathsf{CRS}, \mathsf{sid}, r)$. As the outputs are delayed and public, the simulator must simulate generating these outputs to the honest parties as well just as the ideal functionality would. The simulator $\mathcal{S}_{\text{2-WAKE}}$ now simulates the ideal signature of knowledge functionality $\mathcal{F}_{\mathsf{SOK}}$. $\mathcal{S}_{\text{2-WAKE}}$ first generates an output $(\mathsf{SetUp}, \mathsf{sid}' = (\mathsf{sid}, \mathcal{R}'))$ for the relation $\mathcal{R}' = (\mathcal{R} \wedge \mathcal{R}^g_{\mathsf{DLP}})$ to $\mathcal{A}$ and receives in response the algorithms. $\mathcal{S}_{\text{2-WAKE}}$ records and responds to all SetUp queries from corrupt P_j with $(\mathsf{Algorithms}, \mathsf{sid}', \mathsf{Sign}, \mathsf{Verify})$ as well as responds to any signature and verification queries honestly, just as $\mathcal{F}_{\mathsf{SOK}}$ would. In the case of any errors $\mathcal{S}_{\text{2-WAKE}}$ halts the simulation of $\mathcal{F}_{\mathsf{SOK}}$.

$\mathcal{G}_3$ is indistinguishable from $\mathcal{G}_2$: The environment does not see any internal records of the functionalities. The simulator's output for the simulated $\mathcal{F}_{\mathsf{CRS}}$ is identically distributed to that of the ideal functionality, as the distribution is public. The simulator internally runs $\mathcal{F}_{\mathsf{SOK}}$ so it's behavior is indistinguishable from the actual functionality. The environment does not see any queries or responses to the actual or simulated $\mathcal{F}_{\mathsf{SOK}}$.

Game $\mathcal{G}_4$: Knowledge Queries

The ideal functionalities $\mathcal{F}_{\mathsf{WAR}}$ now respond to any Knowledge queries from $\mathcal{S}$ by checking the record for the queried P and outputting for the witness on record $b = \mathcal{R}(\phi, w)$. These queries are, as all other communication is, routed through the split functionality.

$\mathcal{G}_4$ is indistinguishable from $\mathcal{G}_3$: The knowledge queries are neither seen nor detected by the environment, and the output is not used in any meaningful way by the simulator in this hybrid.

Note: The following four games ($\mathcal{G}_5$ through $\mathcal{G}_7$) apply to the instances of $\mathcal{F}_{\mathsf{WAR}}$ where all of the participating parties are honest. This means that the adversary must have decided to not split the sessions and that there is only one authentication set $H = \mathcal{P}$ detected by the simulator.

Game $\mathcal{G}_5$: $\mathcal{F}_{\mathsf{WAR}}$ Chooses Keys in the Event of Success

In the event that all parties in H are honest and the exchange was successful, thus $\mathsf{Succ} = \mathsf{True}$ meaning that the records in $\mathcal{F}^H_{\mathsf{WAR}}$ are such that $(\phi_i, w_i) \in \mathcal{R}$ for all P_i. Then, for the first NewKey query the functionality $\mathcal{F}_{\mathsf{WAR}}$ samples a key at random from $\{0,1\}^\lambda$ and outputs this key to all queried parties, just as outlined in Figure 10, instead of forwarding the key queried by the $\mathcal{S}_{\text{2-WAKE}}$. The key queried was the key generated on behalf of the simulated party, which in the case of success is the real key as would be output in the protocol by the parties in H.

$\mathcal{G}_5$ is indistinguishable from $\mathcal{G}_4$: This indistinguishability holds under the DDH assumption in $\mathcal{G}$.

Any environment $\mathcal{Z}$ that can distinguish between $\mathcal{G}_4$ and $\mathcal{G}_5$ with nonneglible advantage can be used to build an adversary $\mathcal{B}$ breaking the DDH assumption in $\mathbb{G}$. By the passive security of the Burmester-Desmedt group key exchange, in order to construct such an adversary $\mathcal{B}$ it suffices to construct a $\mathcal{D}$ with nonnegligible advantage against the passive security (confidentiality) of Π_{BDKE} seen in Figure 19.
The environment receives $\mathsf{trans}^{\mathcal{P}}_{\Pi_{\mathsf{2\text{-}WAKE}}}$ and the output keys, with

$$\mathsf{trans}^{\mathcal{P}}_{\Pi_{\mathsf{2\text{-}WAKE}}} = \{(P_i, z_i, \phi_i), (P_i, Z_i, \sigma_i)\}_{P_i \in \mathcal{P}}$$

The environment does not see any $\mathcal{F}_{\mathsf{SOK}}$ queries and responses but does see the public output of $\mathcal{F}_{\mathsf{CRS}}$ queries as $(\mathbb{G}, g, q)$.
On input $\mathsf{CRS} = (\mathbb{G}, g, q)$ (with empty $\Phi, W, \mathcal{R}$) the distinguisher $\mathcal{D}$ first queries $\mathsf{Execute}(0) \leftarrow T'$ receiving a transcript $T' = (z'_i), (Z'_i)_{P_i \in \mathcal{P}}$ between honest parties in protocol Π_{BDKE} and then emulates $\mathcal{G}_4$ with some minor modifications to the simulator $\mathcal{S}_{\mathsf{2\text{-}WAKE}}$: (1) in round 1 the simulator uses z'_i as the round one message, sending instead (P_i, z'_i), (2) in round 2 the simulator sends Z'_i from T' instead of Z_i. As the signature of knowledge is simulated without knowledge of any witness in $\mathcal{G}_4$, the simulator does not need to know the discrete logarithm $x'_i = \mathsf{log}_g(z_i)$ to generate these signatures. Once $\mathcal{S}_{\mathsf{2\text{-}WAKE}}$ has completed the two rounds $\mathcal{D}$ outputs $(1, 1)$ and receives back the challenge $\bar{sk}$. Then, for each party P_i the simulator queries $(\mathsf{NewKey} : \mathsf{sid}, P_i, \bar{sk})$. In the case that the challenge $\bar{sk}$ is the real key then the environment's view will be that in $\mathcal{G}_4$, otherwise the environment's view will be that in $\mathcal{G}_5$. Assuming $\mathcal{Z}$ outputs $\bar{b} = 1$ when it has determined to be in $\mathcal{G}_4$ then $\mathcal{D}$ forwards the guess bit and outputs $b' = \bar{b}$ as his own output. The advantage of $\mathcal{D}$ in the confidentiality experiment is equal to that of $\mathcal{Z}$ in distinguishing the games. If $\mathcal{Z}$ has non-negligible advantage in the confidentiality game then there exists an adversary against DDH in $\mathbb{G}$.

Game $\mathcal{G}_6$: $\mathcal{F}_{\mathsf{WAR}}$ Chooses Keys in the Event of Failure

In the event that all parties in $\mathcal{P}$ are honest and there is some P_i with corresponding record (P_i, ϕ_i, w_i) such that $(\phi_i, w_i) \notin \mathcal{R}$ then $\mathsf{Succ} = \mathsf{False}$. In this case the functionality $\mathcal{F}_{\mathsf{WAR}}$ independently samples new keys to output to the queried parties instead of forwarding the queried keys as outlined in Figure 10.

$\mathcal{G}_6$ is indistinguishable from $\mathcal{G}_5$: The outputs are identically distributed. In $\mathcal{G}_5$, in the event of failure, the simulated parties do not accept and output a uniformly random key which the simulator then queries to the functionality. This key is then output to the parties (and then to the environment). In $\mathcal{G}_6$ upon receipt of each new key query the functionality samples a new independent and uniformly random key to output to the parties.

Game $\mathcal{G}_7$: $\mathcal{F}_{\mathsf{WAR}}$ No Longer Leaks Secrets

In the event that all parties in $\mathcal{P}$ are honest the instances of $\mathcal{F}_{\mathsf{WAR}}$ no longer forward the witnesses included in the NewSession queries by the dummy parties. The functionality instead forwards only $(\mathsf{NewSession}, \mathsf{sid}, P_i, \Phi_i)$ to $\mathcal{S}_{2\text{-}\mathsf{WAKE}}$ as detailed in the description found in Figure 10. As $\mathcal{S}_{2\text{-}\mathsf{WAKE}}$ no longer receives the witnesses from the functionality it uses knowledge queries $(\mathsf{Knldg} : \mathsf{sid}_H, P_i)$ to determine the response bit b_i indicating if the party has knowledge of a valid witness prior to simulating any signatures on their behalf. If $b_i = 0$ then the simulator halts on behalf of that party, just as an honest party would if they could not generate a verifying signature. Otherwise, $\mathcal{S}_{2\text{-}\mathsf{WAKE}}$ generates verifying signatures using the Simsign algorithm and ensures that these signatures verify using Verify, otherwise halting with an error on behalf of $\mathcal{F}_{\mathsf{SOK}}$.

$\mathcal{G}_7$ is indistinguishable from $\mathcal{G}_6$: the knowledge queries provide the simulator with the same information used in the previous game to determine if signatures should verify. The signatures output are independent of the actual witness, and the bit returned by the knowledge query is merely used to determine if the signature should verify. Any verifying signature generated in the previous hybrid would also verify in this hybrid. In both games the simulated honest parties output veryfing signatures iff the corresponding dummy party had queried the functionality with a valid witness.

Note: All of the above games handle the cases when all parties are honest and no man in the middle attack happened. At this point the functionalities and the simulator manage all of the inputs and outputs, message routing, and the protocol transcript in the all honest no splitting case in a way that is consistent with Figure 10. In this case the proof is completed. The following games ($\mathcal{G}_8$ through $\mathcal{G}_{10}$) modify the behavior of the simulator $\mathcal{S}_{2\text{-}\mathsf{WAKE}}$ and the functionality $\mathcal{F}_{\mathsf{WAR}}$ in the case that the instance of $\mathcal{F}_{\mathsf{WAR}}$ is running with some corrupt parties. This means that either the adversary issued corruption queries prior to any Init queries or that $\mathcal{A}$ split the sessions.

Game $\mathcal{G}_8$: $\mathcal{S}_{2\text{-}\mathsf{WAKE}}$ Sets Corrupt Inputs as Extracted Witnesses

For each authentication set H, the simulator $\mathcal{S}_{2\text{-}\mathsf{WAKE}}$ extracts from the signatures received from corrupt parties $P_j \notin H$ witnesses w'_j with the Extract algorithm. After verifying that the relation holds with respect to the extracted signature, the simulator submits for all $P_j \notin H$ a NewSession query to $\mathcal{F}^H_{\mathsf{WAR}}$ via an input query to the split functionality on behalf of that party in order to set the corrupt witness: $(\mathsf{Input} : \mathsf{sid}, H, P_j, (\mathsf{NewSession} : \mathsf{sid}_H, \phi_j, w'_j))$.

$\mathcal{G}_8$ is indistinguishable from $\mathcal{G}_7$: The environment does not see the queries submitted to the ideal functionalities, and do not see the internal records of the functionalities.

Game $\mathcal{G}_9$: $\mathcal{F}_{\mathsf{WAR}}$ Follows All NewKey Instructions

In this game each instance of $\mathcal{F}_{\mathsf{WAR}}$ follows all of the instructions in Figure 10 for sampling and distributing keys to all parties in all cases, including the only remaining case: that the NewKey query is for an honest party interacting with at least one corrupt party and the exchange fails, meaning that there is at least one P_i such that $(\phi_i, w_i) \notin \mathcal{R}$ for the recorded (P_i, ϕ_i, w_i). In this case, upon receipt of each NewKey query, the functionality samples a fresh and independent random key to output to the queried party.

$\mathcal{G}_9$ is indistinguishable from $\mathcal{G}_8$: The outputs are identically distributed as, in $\mathcal{G}_8$ when the simulated parties do not accept and the exchange is a failure the simulator queries the functionality with an independent and uniformly random key which is then forwarded to the parties as output. In $\mathcal{G}_9$ upon receipt of each new key query the functionality samples a new uniformly random key and outputs this key to the queried party.

Game $\mathcal{G}_{10}$: $\mathcal{F}_{\mathsf{WAR}}$ No Longer Leaks Witnesses

The functionality $\mathcal{F}_{\mathsf{WAR}}$ no longer forwards witnesses to $\mathcal{S}_{\text{2-WAKE}}$ upon receipt of NewSession queries, instead forwarding only $(\mathsf{NewSession}, \mathsf{sid}, P_i, \phi_i)$ as in Figure 10. As $\mathcal{S}_{\text{2-WAKE}}$ no longer receives witnesses from the functionality it queries $(\mathsf{Knowledge} : \mathsf{sid}_H, P_i)$ to determine the response bit b_i indicating the validity of the witness input by P_i. If $b_i = 0$ then the simulator halts on behalf of that party, just as an honest party would if they could not generate a verifying signature. Otherwise, $\mathcal{S}_{\text{2-WAKE}}$ generates all signatures using the Simsign algorithm and ensures that these signatures verify using Verify, otherwise halting with an error on behalf of $\mathcal{F}_{\mathsf{SOK}}$.

$\mathcal{G}_{10}$ is indistinguishable from $\mathcal{G}_9$: The honest parties still have signatures that verify based on the validity of the witness input by $\mathcal{Z}$, thus the view of the environment does not change.

The games are indistinguishable in the event that the simulator does not output an unforgeability-error or an incompleteness-error. Let $\mathsf{E}^R_{\mathsf{uf}}$ and $\mathsf{E}^R_{\mathsf{inc}}$ be the events that in the real world $\mathcal{F}_{\mathsf{SOK}}$ outputs an unforgeability-error and an incompleteness-error respectively. Let $\mathsf{E}^I_{\mathsf{uf}}$ and $\mathsf{E}^I_{\mathsf{inc}}$ be these events but for the simulated $\mathcal{F}_{\mathsf{SOK}}$ in the ideal world.

Event $\mathsf{E}^R_{\mathsf{inc}}$ occurs when the functionality $\mathcal{F}_{\mathsf{SOK}}$ receives a signature query $(\mathsf{Sign} : \mathsf{sid}, m, \phi, w)$ and computes $\sigma \leftarrow \mathsf{Simsign}(m, \phi)$ but $\mathsf{Verify}(m, \phi, \sigma) \neq 1$. Event $\mathsf{E}^I_{\mathsf{inc}}$ occurs when the simulated $\mathcal{F}_{\mathsf{SOK}}$ does the same, either (1) when responding to an adversarial signing query or (2) when generating the round 2 messages. Case (1) is exactly when $\mathcal{F}_{\mathsf{SOK}}$ would generate an error. As the simulator perfectly simulates $\mathcal{F}_{\mathsf{SOK}}$ the simulated functionality would output an incompleteness-error with the same probability that the real functionality would output an incompleteness-error .

Event $\mathsf{E}^R_{\mathsf{uf}}$ occurs when a party issues a verification query $(\mathsf{Verify} : \mathsf{sid}, m, \phi, \sigma)$ for (m, ϕ) which was not signed through the signing interface of $\mathcal{F}_{\mathsf{SOK}}$ (otherwise there would be a record) and the Extract algorithm does not output a valid witness but the signature verifies with the Verify algorithm. This implies that

the query is issued by an honest party on an adversarially generated signature which violates extractability. Event $\mathsf{E}_{\mathsf{uf}}^{I}$ occurs when (1) simulated parties verify adversarial signatures in the ideal world through the simulated $\mathcal{F}_{\mathsf{SOK}}$ or (2) when the simulator extracts from verifying signatures to construct new session queries on behalf of the impersonated parties for each instance of $\mathcal{F}_{\mathsf{WAR}}$. Case (1) is exactly the case in which an error occurs in the real world. In case (2) the simulator only extracts witnesses from signatures that have *already* verified through a simulated query to the simulated $\mathcal{F}_{\mathsf{SOK}}$. Any signature which triggers an error in the real world would trigger that error in the ideal world prior to the simulator running the extractor for the new session queries. As these events happen with the same probability in both worlds, the errors occuring do not effect the view of $\mathcal{Z}$ or help the environment to distinguish.

H Offline/Online Computation

The most expensive part of these WAKE protocols is the computation of the Signature of Knowledge, implemented using a Non-Interactive Proof of Knowledge of the witness. If implemented with a SNARK, a *succinct* noninteractive argument of knowledge, the proof can be constructed with small bandwidth and verification time. The ability to instantiate the protocols using SNARKs comes from the choice of non-black-box extraction, which permits succinctness.

It is well known that the bottleneck cost of such computations is the *prover time* ie the time it takes to compute the proof. The following is an optional optimization which migrates the cost of computing the SNARK to an *offline* phase prior to the participant initiating a WAKE session or being contacted for a WAKE. The optimization is shown in the two party case for simplicity.

Let Ψ be an existentially unforgeable under chosen message attack (EUF-CMA) digital signature scheme defined by the following three algorithms: key generation $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$, signature $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$ and verification $\{0,1\} \leftarrow \mathsf{Vfy}(\mathsf{vk}, m, \sigma)$.

Let the Prover P hold a witness w to the statement ϕ such that $(w, \phi) \in \mathcal{R}$. Let Σ be a nBB-s-SE-secure signature of knowledge and $\mathcal{K}$ a KEM-CPA-secure key encapsulation mechanism. During the offline phase P generates $(sk, vk) \leftarrow \Psi.\mathsf{Gen}(1^\lambda)$ as a key pair for a digital signature scheme Ψ with sk the secret signing key and vk the public verification key. Then, P uses the SOK Σ to sign vk, storing $\big((sk, vk), \sigma_1\big)$ for:

$$\sigma_1 \leftarrow \Sigma.\mathsf{SSign}(pp_\Sigma, \phi, w, vk)$$

During the online phase, any party V to whom P wants to authenticate generates and sends ek. With this key P encapsulates $(k, C) \leftarrow \mathcal{K}.\mathsf{Encap}(ek)$, sets $m := (C||ek)$ and signs $\sigma_2 \leftarrow \Psi.\mathsf{Sign}(sk, m)$. Then P sends $(C, vk, \sigma_1, \sigma_2)$ back to V for verification and decapsulation. If V verifies σ_1 and σ_2 then V computes the key $k \leftarrow \mathcal{K}.\mathsf{Decap}(dk, C)$. This is the protocol $\Pi_{\mathsf{UWAKE}}^{\mathcal{K},\Sigma}$ from Appendix B.

The offline-online optimized protocol can be seen in Figure 21 for any existentially unforgeable EUF-CMA-secure digital signature scheme Ψ. Incidentally,

the digital signature Ψ can be a one-time signature as each verification key is used to sign a single message. Intuitively, the proof of security follows from the security of Σ and Ψ.

Concerning security: the main technical issue is extraction. The Prover is guaranteed to know the witness during the offline phase during the computation of σ_1 but not upon completion of the online phase of the protocol. This is an inherent limitation of the optimization. In some applications this may be an issue eg if a party must prove storage of a certain file. One way to address this issue in practice is to add a timestamp to the offline signature, thereby mitigating the problem and guaranteeing that the Prover *knew* the witness at a relatively recent known time.

While the above discussion and Figure 21 refer only to the UWAKE setting this modification can be generalized and applied to the group-WAKE setting by having all authenticated participants execute the offline phase.

I Experimental Settings

The signature of knowledge used for instantiation [30] requires groups endowed with bilinear pairings; its signatures consist of only two elements of $\mathbb{G}_1$, one element of $\mathbb{G}_2$ and a hash. Estimateds are derived through the implementation of [30] in libsnark[19] using curve BN254 with 110 bits of security. [20] All experiments were run on Amazon EC2 c5ad.16xlarge with 128 GiB of RAM running 3.3GHz AMD EPYC 7002 series CPUs, using a single thread.

Zero-Knowledge Contingent Payment:

(a) Sudoku Solutions In the benchmarks in table, the party authenticates with respect to knowledge of a solution to a sudoku puzzle of standard size $N = 10$. The circuit complexity of this verification grows roughly with N^3. *(b) Bug Bounties:* Benchmarks are for $|C_{\text{expect}}| \approx 500K$ wires and $|C_{\text{buggy}}| \approx 10K$ wires.

Dark Pools: This scenario is introduced in Section 1. Each party authenticates on the basis that some committed value is in a certain range. The range for these benchmarks $\bar{\mathbf{B}}$ is 32 bits and the commitment $\bar{c}$ is a SHA256 hash.

Retrieval Market: This scenario is introduced in Section 1. The content identifier (CID) $\bar{h}$ of the file $\mathbf{F}$ is the the Blake3 [9] hash, similarly to with IPFS [35]. Files are broken into 256KB block sizes.[21]

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

[20] This curve has comparable runtime to BLS12-381 with 128 bits of security. More specifically, the work in [8] reports a slowdown factor of roughly $2\times$ for $\mathbb{G}_1$ operations and $3\times$ for $\mathbb{G}_2$ operations in BLS12-381 when compared to BN254.

[21] These benchmarks differ from the current implementation of IPFS which employs SHA256, but producing a (very succinct) SOK for such a SHA computation of that size is significantly more expensive. Hashing with Blake3 requires approximately 2^{19} constraints while SHA256 would require approximately 2^{27}.

$\mathcal{S}_{2\text{-WAKE}}$

$\mathcal{S}$ initializes $\mathcal{A}$ and reliably forwards all messages between $\mathcal{Z}$ and $\mathcal{A}$. Corruption instructions from $\mathcal{A}$ are obeyed if they occur prior to any initialization queries.

- **CRS Functionality:**
  - Sample $(\mathbb{G}, g, q) \leftarrow_\$ \mathcal{D}$
  - Upon queries $(\mathsf{CRS} : \mathsf{sid})$ from party P_j, send a delayed public output $(\mathsf{CRS}, \mathsf{sid}, (\mathbb{G}, g, q))$ to P_j
  - For each honest party P_i: send a delayed public output $(\mathsf{CRS}, \mathsf{sid}, (\mathbb{G}, g, q))$ to P_i
- **Signature of Knowledge Functionality:**
  - Initial Set Up:
    - Set $\mathcal{R}' = \mathcal{R} \times \mathcal{R}_{\mathsf{DLP}}^{(\mathbb{G},g)}$ and $\mathsf{sid}' = (\mathsf{sid}, \mathcal{R}')$
    - Output $(\mathsf{SetUp}, \mathsf{sid}')$ to $\mathcal{A}$ and receive $(\mathsf{Algorithms} : \mathsf{sid}', \mathsf{Verify}, \mathsf{Sign}, \mathsf{Simsign}, \mathsf{Extract})$ from $\mathcal{A}$
  - Set Up, Signing and Verification Queries: responses are simulated exactly as in the description of $\mathcal{F}_{\mathsf{SOK}}$, except the simulator records any valid witnesses submitted by corrupt parties
  - In the case that the simulated $\mathcal{F}_{\mathsf{SOK}}$ generates an error: halt the simulation of $\mathcal{F}_{\mathsf{SOK}}$
- **Round 1 Messages:** For all honest P_i, after receipt of $(\mathsf{Init}, \mathsf{sid}, P_i, \Phi_i)$ from $s\mathcal{F}_{\mathsf{WAR}}$
  - Sample $x_i \leftarrow_\$ \mathbb{Z}_q^*$ compute $z_i := g^{x_i}$. Send $(P_i||z_i)$ to all $\mathcal{P}$ and receive $(P_j||z_j^{(i)})$ for all $\mathcal{P}$ and set $\mathsf{vk}_i := P_1||z_1^{(i)}||P_2||z_2^{(i)}||\cdots||P_n||z_n^{(i)}$
- **Adversarial Init Queries:**
  - For each corrupt P_j: query $(\mathsf{Init} : \mathsf{sid}, \Phi)$ to $s\mathcal{F}_{\mathsf{WAR}}$ from P_j
  - For each honest P_i: determine the authentication set as $H_i = \{P_k \in \mathcal{P} | \mathsf{vk}_k = \mathsf{vk}_i\}$
  - For each authentication set H:
    - For any $P_i \in H$: set $\mathsf{sid}_H = \mathsf{vk}_i$
    - For each $P_i \in H$: query $(\mathsf{Init} : \mathsf{sid}, P_i, H, \mathsf{sid}_H, \Phi)$ to $s\mathcal{F}_{\mathsf{WAR}}$
- **Round 2 Messages:** For all honest P_i after receipt of $(\mathsf{NewSession}, \mathsf{sid}_H, \phi_i, P_i)$ from $\mathcal{F}_{\mathsf{WAR}}^H$ through $s\mathcal{F}_{\mathsf{WAR}}$
  - Query $(\mathsf{Knowledge} : \mathsf{sid}_H, P_i)$ to $s\mathcal{F}_{\mathsf{WAR}}$ and receive $(\mathsf{Knowledge}, P_i, \phi_i, b_i)$. If $b_i = 0$: halt
  - Compute $Z_i := (z_{i+1}^{(i)} / z_{i-1}^{(i)})^{x_i}$ and $m_i := (P_i||Z_i||\mathsf{vk}_i)$
  - Compute $\sigma_i \leftarrow \mathsf{Simsign}(m_i, \phi_i')$ for $\phi_i' = (\phi_i, z_i)$ and if $\mathsf{Verify}(m_i, \phi_i') \neq 1$ halt the simulation of $\mathcal{F}_{\mathsf{SOK}}$ and the party P_i, otherwise record (m_i, ϕ_i', σ_i) on behalf of the simulated $\mathcal{F}_{\mathsf{SOK}}$
  - Send $(P_i||Z_i||\sigma_i)$ to all $\mathcal{P}$ and receive $m_j^{(i)} = (P_j||Z_j^{(i)}||\sigma_j^{(i)})$ from each P_j
  - Verify each $\sigma_j^{(i)}$ by simulating the behavior of $\mathcal{F}_{\mathsf{SOK}}$ in a verification query. Let $\{b_j^{(i)}\}_j$ be the returned verification bits. If any $b_j^{(i)} = 0$ set $sk_i \leftarrow \{0,1\}^\lambda$
- **Key Computation:** For each honest P_i: if sk_i is not set compute sk_i using the messages that party received
- **New Session Queries:** For each H, for all $P_j \notin H$:
  - If any $b_j^{(i)} = 1$ for $P_i \in H$: Extract $\bar{w}_j^{(i)} \leftarrow \mathsf{Extract}(m_j^{(i)}, \phi_j^{(H)}, \sigma_j^{(H)})$
    - If $(\bar{w}, \phi_j^{(H)}) \in \mathcal{R}$: Query $(\mathsf{NewSession} : \mathsf{sid}_H, \phi_j^{(i)}, \bar{w}_j^{(i)})$ to $\mathcal{F}_{\mathsf{WAR}}^H$
    - Else: halt on behalf of $\mathcal{F}_{\mathsf{SOK}}$ with an $\mathsf{unforgeability\text{-}error}$
  - Query $(\mathsf{NewSession} : \mathsf{sid}_H, \phi_j^{(i)}, \bar{w}_j^{(i)})$ to $s\mathcal{F}_{\mathsf{WAR}}$ as $(\mathsf{Input} : \mathsf{sid}, H, P_j, q_j^{(H)})$
- **New Key Queries:** For all honest P_i:
  - Query $\mathcal{F}_{\mathsf{WAR}}^H$ with $(\mathsf{NewKey} : \mathsf{sid}_H, P_i, sk_i)$

Fig. 20. The simulator for protocol $\Pi_{2\text{-WAKE}}$.

SEQUENCE OF GAMES								
	Game	$s\mathcal{F}_{\mathsf{WAR}}$		$\mathcal{F}_{\mathsf{WAR}}$			$\mathcal{S}_{\mathsf{WAR}}$	Property
		Init	I/O	NewSession	KnIdg	NewKey		
	$\mathcal{G}_0$	N/A: this is the real protocol						
All Cases	$\mathcal{G}_1$	forwards queries & invokes instances of $\mathcal{F}_{\mathsf{WAR}}$	routes communication	forwards queries from dummy parties to $\mathcal{S}$		forwards outputs from $\mathcal{S}$ to dummy parties	simulates messages of honest parties & splits sessions	N/A
	$\mathcal{G}_2$	records inputs & checks queries				issues queried keys	submits NewKey queries	N/A
	$\mathcal{G}_3$			records inputs			simulates $\mathcal{F}_{\mathsf{SOK}}$ & $\mathcal{F}_{\mathsf{CRS}}$	N/A
	$\mathcal{G}_4$				verifies witness knowledge when queried		submits Knowledge queries	N/A
All Honest & No MITM	$\mathcal{G}_5$					Succ = T $\Rightarrow$ sends consistent random key		DDH
	$\mathcal{G}_6$					Succ = F $\Rightarrow$ sends independent random keys		N/A
	$\mathcal{G}_7$			no longer forwards witnesses			uses knowledge queries to learn if parties have witnesses	N/A
Some Corruptions or MITM	$\mathcal{G}_8$			corrupt inputs are set by $\mathcal{S}$			uses Extract to obtain witnesses and forwards them to $\mathcal{F}^H_{\mathsf{WAR}}$	$\mathcal{F}_{\mathsf{SOK}}$
	$\mathcal{G}_9$					Succ = F $\Rightarrow$ samples independent keys for any honest partners		N/A
	$\mathcal{G}_{10}$			no longer forwards witnesses to $\mathcal{S}$			uses knowledge queries to learn if parties have witnesses	N/A

Table 5. Summary of hybrids for Theorem 3.

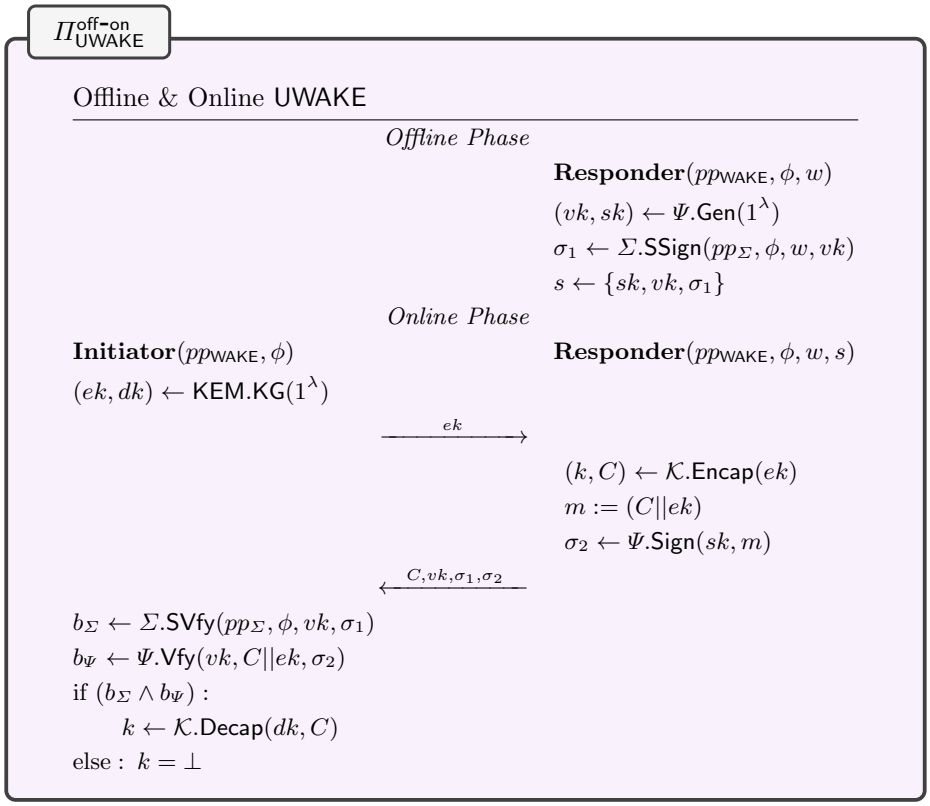

Fig. 21. The offline-online optimized protocol for UWAKE.