Generic Forward-Secure Key Agreement Without Signatures

Cyprien de Saint Guilhem, Nigel P. Smart, and Bogdan Warinschi

Dept. Computer Science, University of Bristol, United Kingdom.

Abstract. We present a generic, yet simple and efficient transformation to obtain a forward secure authenticated key exchange protocol from a two-move passively secure unauthenticated key agreement scheme (such as standard Diffie–Hellman or Frodo or NewHope). Our construction requires only an IND-CCA public key encryption scheme (such as RSA-OAEP or a method based on ring-LWE), and a message authentication code. Particularly relevant in the context of the state-of-the-art of postquantum secure primitives, we avoid the use of digital signature schemes: practical candidate post-quantum signature schemes are less accepted (and require more bandwidth) than candidate post-quantum public key encryption schemes. An additional feature of our proposal is that it helps avoid the bad practice of using long term keys certified for encryption to produce digital signatures. We prove the security of our transformation in the random oracle model.

1 Introduction

Forward secrecy and authentication are the standard security requirements for authenticated key agreement protocols (AKA). They require that parties authenticate one another, and that the key derived remains secret to anyone but to the two parties involved at the time of the execution. Modern realizations rely on the Diffie–Hellman protocol which is unauthenticated and guarantees key secrecy only against passive adversaries. The stronger property is obtained via additional mechanisms which authenticate the two parties and ensure integrity of the conversation between them, even against active adversaries.

Numerous generic transformations in the literature show how to achieve full AKA active security from protocols with weaker guarantees [3,9,22,28,18,24] using simple mechanisms such as signatures, encryption, and MACs.

Such generic techniques are particularly appealing; on the one hand they enable a modular approach where the base protocol and the details of the transformation are designed and analyzed independently – in particular, if needed, the underlying protocol can be easily swaped out and replaced with a different mechanism. On the other it provide conceptual clarity for choices that are made, e.g. which part of the the protocol provides say, key-secrecy, and which deals with integrity/entity authentication.

In this paper we contribute to this research direction. We provide a simple generic transformation which, when applied to a certain class of passively secure key-exchange protocols, yields the most round-efficient authenticated key-agreement protocols against

This work has been supported in part by ERC Advanced Grant ERC-2015-AdG-IMPaCT.

active adversaries to date. Besides optimal round complexity, our proposal has two interesting implications which serve as further motivation for this work. The first concerns the practicalities of existing RSA certified public keys; the second concerns security of key-agreement protocols in the post-quantum world.

Consider the instantiation of the "signed Diffie-Hellman" construction which appears, for example, in the popular TLS 1.0-1.2 ciphersuite

`TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`,

using RSA signatures and elliptic curve based Diffie–Hellman. This usage is a bit of a kludge: RSA certificates in existence were issued for dual-use of RSA in both signature and encryption mode (which was needed for the earlier TLS mechanism of RSA key-transport which is still prevalent). Deploying protocols where the same keys are used for both signature and encryption would encourage a usage which is not supported by rigorous mathematical guarantees. Short of issuing new RSA keys, this type of misuse could be avoided by ensuring that existing keys are only used for encryption. We note that this is not just a theoretical concern. Attacks against deployed cryptography that reuse keys in unintended ways have been previously reported [27,19,20].

We now discuss the design of key-exchange protocols secure in the post-quantum setting. Here, a natural strategy is to consider existing designs and replace the different components with post-quantum secure versions. The underlying Diffie-Hellman constructions can be replaced by (Ring-)LWE-based variant such as NewHope [7] or Frodo [1]. For other primitives, the situation seems to be more delicate. Both for historical and technical reasons, there seems to be less confidence in proposals for post-quantum signatures th an for post-quantum encryption. Whilst lattice based encryption schemes have a strong track record, see NTRU [16] for a h istoric scheme or Ring-LWE [26,25] for more modern ones, the use of lattice based signature s chemes is less stable. Many early schemes, such as GGH [13] and NTRUSign [17,15], were eventually broken due to issues with the distribution of the signatures [12,29]; however recently more promising lattice based candidates have been proposed such as [10]. Post-quantum signature schemes based on Merkle hash trees have also had issues related to the need to maintain a large state; again recently this issue has been overcome with the introduction of state-less hash tree based [5].

Questionable dual use of RSA keys, and the relatively slow progress of post-quantum secure signature schemes, raises the question of whether one can design a passively (forward) secure unauthenticated protocol together with authentication mechanisms that rely solely on post-quantum public key encryption schemes.

Our results. We answer this question in the positive. We propose a generic transformation which bootstraps a forward secure AKA protocol out of a two-pass passively secure unauthenticated key agreement (KA) scheme which satisfies some mild additional conditions. The transformation uses an arbitrary IND-CCA public key encryption scheme and a strongly unforgeable MAC. Below we provide a sketch of our transformation, motivate its design and discuss the additional requirements on the underlying protocol.

Consider an arbitrary such protocol Π, whose execution between parties U and V is described in Figure 1 using the general syntax introduced by Bellare and Rogaway [4]. For example, Diffie–Hellman is an instantiation where U's ephemeral key is e_A and m_1 is g^{e_A}, V's ephemeral key is e_B (which can be deleted as soon as it is used to

U		**V**
$(e_A, m_1, *) \leftarrow \mathit{\Pi}(\mathsf{init}, -, -; \$_1)$		
	$\xrightarrow{m_1}$	
		$(-, m_2, \mathsf{k}_B) \leftarrow \mathit{\Pi}(\mathsf{resp}, m_1, -; \$_2)$
	$\xleftarrow{m_2}$	
$(-, \dashv, \mathsf{k}_A) \leftarrow \mathit{\Pi}(\mathsf{init}, m_1 \Vert m_2, e_A, \$_3)$		
return k_A		**return** k_B

Fig. 1: An Arbitrary Two-Round Unauthenticated Key Agreement Protocol $\mathit{\Pi}$.

derive $m_2 = g^{e_B}$ and $\mathsf{k}_B = m_1^{e_B}$), finally the computation of k_A is done by U using the equation $\mathsf{k}_A = m_2^{e_A}$. To obtain forward secrecy, the ephemeral key data is assumed to be deleted as soon as the session keys are locally computed.

We bootstrap this two round KA protocol into a fully authenticated one (which inherits the forward secrecy property). Our construction, presented in Figure 2, requires a public key encryption scheme secure under chosen ciphertext attacks, a strongly unforgeable message authentication code, and two key derivation functions H_1 and H_2 which we model as random oracles.

The protocol works by wrapping the message flows, m_1 and m_2, of the KA protocol in encryptions under the long term keys of the two parties. Interestingly, the main role played by encryption here is to authenticate the parties and ensure integrity of the messages they exchange. Indeed, one can think of the first two messages of the protocol as a challenge-response exchange where U attempts to authenticate V by sending an encryption of m_1 under the public key of V and expecting to receive the same m_1 in the next flow. Similarly, the second and third flow can be interpreted as a challenge-response where V sends m_2 to U and expects to receive a message that depends on m_2. In addition, the MAC send as the last message also ties the identities of the parties involved with this particular execution of the protocol run. The final application key is derived from the same key from $\mathit{\Pi}$, but in a way that decouples it from the MAC key and also incorporates the identities of the participants.

The last message flow and key derivation methodology also thwart an analogue of the (in)famous attack against the Needham-Schroeder protocol. A malicious V could reencrypt the first message for a third party W who would reply with its own encrypted m_2 for U; V could simply forward this message so U. Parties U and W would thus derive the same key for the underlying passively secure protocol. However, W will no longer accept the MAC as it will be on the wrong message ($U \Vert V$ as opposed to $U \Vert W$), thus thwarting the attack. In addition, since it depends on the participants' identities, the derived session key will also be different for U and W.

The essence of our transformation is that it attempts to ensure that an active adversary cannot interfere with the execution of the underlying protocol, i.e. that when a

$$\begin{array}{lcl}
U \quad (\mathsf{pk}_U, \mathsf{sk}_U) & & V \quad (\mathsf{pk}_V, \mathsf{sk}_V) \\
\hline
(e_U, m_1, *) \leftarrow \mathit{\Pi}(\mathsf{init}, -, -; \$_1) & & \\
\mathfrak{m}_1 \leftarrow \mathsf{Enc}_{\mathsf{pk}_V}(U \| m_1) & \xrightarrow{\mathfrak{m}_1} & U \| m_1 = \mathsf{Dec}_{\mathsf{sk}_V}(\mathfrak{m}_1) \\
 & & (-, m_2, \kappa_V) \leftarrow \mathit{\Pi}(\mathsf{resp}, m_1, -; \$_2) \\
m_1' \| m_2 = \mathsf{Dec}_{\mathsf{sk}_U}(\mathfrak{m}_2) & \xleftarrow{\mathfrak{m}_2} & \mathfrak{m}_2 \leftarrow \mathsf{Enc}_{\mathsf{pk}_U}(m_1 \| m_2) \\
\textbf{if } m_1' \neq m_1 \textbf{ then} & & \\
\quad \textbf{reject} & & \\
(-, \vdash, \kappa_U) \leftarrow \mathit{\Pi}(\mathsf{init}, m_1 \| m_2, e_U; \$_3) & & \\
\mathsf{k}_{U,1} \leftarrow H_1(\kappa_U) & & \\
\mathfrak{m}_3 \leftarrow \mathsf{Mac}_{\mathsf{k}_{U,1}}(U \| V) & \xrightarrow{\mathfrak{m}_3} & \mathsf{k}_{V,1} \leftarrow H_1(\kappa_V) \\
 & & \textbf{if } \mathsf{Vrfy}_{\mathsf{k}_{V,1}}(U \| V, \mathfrak{m}_3) = 0 \textbf{ then} \\
 & & \quad \textbf{reject} \\
\mathsf{k}_{U,2} \leftarrow H_2(\kappa_U \| U \| V) & & \mathsf{k}_{V,2} \leftarrow H_2(\kappa_V \| U \| V) \\
\textbf{return } \mathsf{k}_{U,2} & & \textbf{return } \mathsf{k}_{V,2}
\end{array}$$

Fig. 2: The New AKA Protocol Construction.

party accepts, it must have engaged in an execution with another honest party. Put otherwise, even an active adversary cannot force a session to accept other than by forwarding honest messages.

Using non-malleable encryption to protect the integrity of messages goes someway towards implementing this intuition. Ensuring that parties authenticate each other successfully is however not obvious, and in fact require additional properties on the underlying protocol $\mathit{\Pi}$. As explained above, one should think of the first two messages as a challenge-response protocol to authenticate V. Notice that for security of authentication, this requires that message m_1 of the $\mathit{\Pi}$ has sufficient entropy; otherwise, an adversary who guesses m_1 can reply with an appropriately message which encrypts m_1 and some m_2 and get U to accept.

Similarly, one should think of the second and third messages as a challenge-response protocol that authenticates U: the last message should only be computable by some party which received m_2 and derived the MAC key from it. This intuition is valid only if m_2 actually helps determine the MAC key, which is not necessarily the case. Consider a two message protocol where, if the first message of U for V is some fixed message bad, then V sets the local key to, say, 0^n. Such a protocol may still be secure against a passive adversary as an honest execution U would never send bad. Yet, the protocol obtained by applying our transformation is not actively secure since the adversary can send the encryption of bad to V. More generally, a close look shows that the problem is that the adversary can send an appropriately crafted message m_1 which coerces the key into one which can be easily guessed (even if V behaves honestly).

The above discussion shows that we need two additional properties for our transformation to work: i) that the first message of Π is unpredictable and ii) that even if the first message is an arbitrary message sent by the adversary then the key derived by V is still unpredictable.

Naturally, one can ask if further subtle attacks are possible. We show that this is not the case and provide rigorous guarantees for the above intuition. We show that if the starting protocol is an arbitrary passively secure two-message protocol and satisfies the two additional security properties informally described above, then the transformation that we propose yields a full fledged forward secure key exchange protocol with mutual authentication (in the random oracle model), under standard assumptions on the encryption and MAC scheme used in the transformation.

Related work. The first generic compilers for authenticated key exchange were by Bellare, Canetti, and Krawczyk [3] later refined by Canetti and Krawczyk [9]. These works consider adversaries of different strength, but share an interesting idea of protocol design. First construct a protocol secure in a model where links between parties are authenticated (i.e. secure against passive adversaries), and then compile it into a stronger version, secure in a world with unauthenticated links, by using special-purpose *authenticators* which authenticate the sender of each message and ensure their integrity. In particular, BCK present an authenticator that uses IND-CCA2 secure encryption and MAC schemes. However, the use of authenticator replaces every message flow of the base protocols with three flows, so starting from a two-message flow protocols one obtains a stronger protocol that requires five rounds. Unfortunately the general setting of MT-authenticators of BCK works does not immediately allows for further optimisation which reduces the number of rounds.

Katz and Young[22] consider the problem of boosting passive security to active security for *group* key exchange by first exchanging nonces between parties and then authenticating each message through signatures that involve these nonces. For the case of two parties this result in a protocol with four message flows. For this type of protocols, a less efficient compiler is the one studied by Morrissey, Smart and Warinschi [28]. They show that TLScan be regarded as TLS as the successive applications of two generic transformations which bootstrap passive security to active security.

A second line of work which is related to ours is based on the observation that key encapsulation mechanisms naturally give rise to passively secure key-exchange protocols (where one party sends the parameters of a KEM scheme, and the second party sends a KEM). There are by now several constructions of key-exchange protocols (in settings which are sometimes different from ours) which start from KEMs. For example, Boyd et al. [8] construct authenticated key exchange from KEMs, meeting the eCK stronger security requirement, and Gunther et al [14] show how to add forward security to KEMs to obtain forward security when these are used as a full-key exchange protocol that enables forward secure 0-RTT. Both transformations work in the ID-based setting, use pairings and therefore are not generic.

Perhaps the closest work with ours is that of Li *et al.* [24] who present two transformations that bootstrap AKA protocols out of passively secure ones, one based on signatures and another based on encryption. Both transformations first execute a passively secure KA protocol and then use three additional flows to perform entity authentication (and ensure the integrity of the conversation between the two parties). Just like our proposal, the encryption-based construction of [24] can serve to avoid the two issues which we have outlined above but at an increased round-complexity cost. In essence, we avoid additional communication rounds by showing how to piggy-back entity authentication on top of the passively secure protocol.

One observations which is warranted at this point is that our transformation does not achieve key-confirmation [11] (while derived keys are secret and parties authenticate each other, one party may accept without the other party actually having derived the key), whereas some other transformations do. This was not an explicit goal, afterall the notion has only recently been formalized [11].

2 Preliminaries

We first recall some standard definitions of primitives and their security notions. A comprehensive overview of this material can be found in [21], with slight modifications to suit our notation in later sections. We then recall basic notions of security for passive key agreement protocols and introduce two new formal definitions. Throughout this paper, we denote the security parameter by λ, represented in unary notation as 1^λ, and the empty string by $-$.

2.1 Standard Definitions

We recall briefly the informal descriptions of actively secure public-key encryption schemes (with the addition of multi-user security), strongly unforgeable message authentication codes as well as key derivation functions and the random oracle model.

Public-key encryption schemes. In this paper, we denote a public-key encryption scheme by a tuple $E = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ of $\mathsf{poly}(\lambda)$-time algorithms. We assume that such schemes correctly decrypt honestly encrypted ciphertexts with overwhelming probability.

The standard (single-user) active security notion for such schemes is that of *indistinguishability under chosen ciphertext attack*, denoted $\mathsf{IND\text{-}CCA}$. The experiment, also called game, for this setting gives an arbitrary adversary a randomly sampled public-key and, upon query of a left-right oracle, denoted $\mathsf{L\text{-}R}$, with two messages of identical lengths, returns the encryption of one of the two. Given access to a decryption oracle, the adversary's goal is to guess which of the two messages the oracle encrypts. The adversary may query either oracle several times, with the only restriction that it may not query the decryption oracle on any ciphertext output by the left-right oracle.

In the proof of security of our protocol, we make use of the multi-user security notion described in [2]. For n participants, the n-$\mathsf{IND\text{-}CCA}$ security experiment is very similar to the single-user setting. The difference is that the adversary is provided with n different public keys and may query the left-right oracle on any one of these keys. Whether it is the right or left message which is encrypted is still selected at random, but this choice remains consistent between all queries of the $\mathsf{L\text{-}R}$ oracle.

More formally we have

Definition 1 (Public-key Encryption Scheme). *A* public-key encryption scheme *is a tuple of probabilistic* $\mathsf{poly}(\lambda)$*-time algorithms* $E = (\mathit{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *such that:*

1. *The* setup *algorithm* Setup *takes as input the security parameter* 1^λ *and outputs a tuple of public parameters* params *required by the encryption scheme.*
   We assume for convenience that λ *is implicit in* params.
2. *The* key-generation *algorithm* KGen *takes as input the public parameters* params *and outputs a public/private key pair* $(\mathsf{pk}, \mathsf{sk})$.
   We assume for convenience that params *is implicit in either* pk *or* sk.
3. *The* encryption *algorithm* Enc *takes as input a public key* pk *and a message* m *from some message space* $\mathcal{M}$ *(specified by* params*). It outputs a ciphertext* c*, and we write this as* $c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)$.
4. *The deterministic* decryption *algorithm* Dec *takes as input a private key* sk *and a ciphertext* c*, and outputs a message* m *or a special symbol* $\perp$ *denoting failure. We write this as* $m := \mathsf{Dec}_{\mathsf{sk}}(c)$.

It is required that, except possibly with negligible probability over $(\mathsf{pk}, \mathsf{sk})$ *output by* $\mathsf{KGen}(\mathit{params})$*, we have* $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(m)) = m$ *for any valid message* m.

For an arbitrary public-key encryption scheme $E = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ and $\mathsf{poly}(\lambda)$-time adversary $\mathcal{A}$, we assume that the challenger simulates a set $\mathcal{U}$ of n participants to the adversary. For a bit $b \in \{0, 1\}$ we then define the n-IND-CCA-b experiment in Figure 3. We denote $\mathcal{A}$'s advantage in the n-IND-CCA game as

$$\mathbf{Adv}^{n\text{-}\mathsf{IND}\text{-}\mathsf{CCA}}_{\mathcal{A},E}(\lambda) = \left|\Pr\left[\mathbf{Exp}^{n\text{-}\mathsf{IND}\text{-}\mathsf{CCA}\text{-}0}_{\mathcal{A},E}(\lambda) = 1\right] - \Pr\left[\mathbf{Exp}^{n\text{-}\mathsf{IND}\text{-}\mathsf{CCA}\text{-}1}_{\mathcal{A},E}(\lambda) = 1\right]\right|$$

1. $\mathsf{Setup}(1^\lambda)$ is run to obtain params.
2. For each $U \in \mathcal{U}$, $\mathsf{KGen}(\mathsf{params})$ is run to obtain keys $(\mathsf{pk}_U, \mathsf{sk}_U)$.
3. The adversary $\mathcal{A}$ is given $\{\mathsf{pk}_U\}_{U \in \mathcal{U}}$ and access to several oracles: $\mathsf{L}\text{-}\mathsf{R}_{\mathsf{pk}_U}(\cdot,\cdot)$ and $\mathsf{Dec}_{\mathsf{sk}_U}(\cdot)$ for each $U \in \mathcal{U}$. The $\mathsf{L}\text{-}\mathsf{R}$ oracles take as input two valid messages m_0 and m_1 and returns the encryption of m_b under the public key pk_U.
4. $\mathcal{A}$ may query the decryption oracles whenever it wishes, but may never submit a ciphertext output by $\mathsf{L}\text{-}\mathsf{R}_{\mathsf{pk}_U}$ to the corresponding decryption oracle $\mathsf{Dec}_{\mathsf{sk}_U}$.
5. Finally, $\mathcal{A}$ outputs a guess bit b'. We define the output of the experiment to be that guess b'.

Fig. 3: The n-IND-CCA-b Security Experiment $\mathbf{Exp}^{n\text{-}\mathsf{IND}\text{-}\mathsf{CCA}\text{-}b}_{\mathcal{A},E}(\lambda)$.

Definition 2 (n-CCA-Security). *A public-key encryption scheme given by* $E = (\mathit{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *is said to have* polynomially-secure indistinguishible encryptions under a chosen-ciphertext attack *(or is* n-CCA-secure*) if for all probabilistic* $\mathsf{poly}(\lambda)$*-time adversaries* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\mathbf{\mathit{Adv}}^{n\text{-}\mathit{IND}\text{-}\mathit{CCA}}_{\mathcal{A},E}(\lambda) \leq \mathsf{negl}(\lambda).$$

We note that security in the multi-user setting and the single user setting are related by the following theorem of [2].

Theorem 1. *Let $E = (\mathit{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme and n_P be an integer polynomial in the security parameter. Then*

$$\mathbf{\mathit{Adv}}^{n\text{-}\mathit{IND\text{-}CCA}}_{\mathcal{A},E}(\lambda) \leq n \cdot \mathbf{\mathit{Adv}}^{1\text{-}\mathit{IND\text{-}CCA}}_{\mathcal{A},E}(\lambda).$$

Unforgeable message authentication codes. Message authentication codes (MACs) are symmetric key primitives that allow parties sharing a secret key k to authenticate and verify messages, thus detecting eventual modification of their content. A MAC is a triple of $\mathsf{poly}(\lambda)$-time algorithms $M = (\mathsf{KGen}, \mathsf{Mac}, \mathsf{Vrfy})$ such that, given a message and a key, Mac produces a tag, and such that, given a message, a tag and a key, Vrfy verifies that the tag corresponds to the message.

The security experiment for strong unforgeability, denoted $\mathsf{MAC\text{-}sFORGE}$, generates a random key and gives the adversary access to a Mac oracle whilst recording pairs of queried messages and the tag that was returned for each. The goal of the adversary is to output a message and a tag such that the verify algorithms accepts this tag and such that this tag was never produced by the Mac oracle for this message.

More formally we have

Definition 3 (Message Authentication Code). *A* message authentication code *(MAC) is a triple of probabilistic* $\mathsf{poly}(\lambda)$*-time algorithms* $(\mathsf{KGen}, \mathit{Mac}, \mathit{Vrfy})$ *such that:*

1. *The* key-generation *algorithm* KGen *takes as input the security parameter* 1^λ *and outputs a key* k.
2. *The* tag-generation *algorithm* Mac *takes as input a key* k *and a message* $m \in \{0,1\}^*$ *and outputs a tag* t. *We write this as* $t \leftarrow \mathit{Mac}_{\mathsf{k}}(m)$.
3. *The deterministic* verification *algorithm* Vrfy *takes as input a key* k, *a message* m *and a tag* t. *It outputs a bit* b, *with* $b = 1$ *meaning* valid *and* $b = 0$ *meaning* $\mathit{invalid}$. *We write this as* $b := \mathit{Vrfy}_{\mathsf{k}}(m, t)$.

It is required that, for every key k *output by* $\mathsf{KGen}(1^\lambda)$ *and every* $m \in \{0,1\}^*$, *it holds that* $\mathit{Vrfy}_{\mathsf{k}}(m, \mathit{Mac}_{\mathsf{k}}(m)) = 1$.

1. A key k is generated by running $\mathsf{KGen}(1^\lambda)$.
2. The adversary $\mathcal{A}$ is given input 1^λ and oracle access to $\mathsf{Mac}_{\mathsf{k}}(\cdot)$. The adversary eventually outputs (m, t). Let $\mathcal{Q}$ denote the set of pairs of messages queried to the Mac oracle together with their responses.
3. $\mathcal{A}$ $\mathsf{succeeds}$ if and only if (1) $\mathsf{Vrfy}_{\mathsf{k}}(m, t) = 1$ and (2) $(m, t) \notin \mathcal{Q}$. In that case the output of the experiment is defined to be 1.

Fig. 4: The $\mathsf{MAC\text{-}sFORGE}$ Security Experiment $\mathbf{Exp}^{\mathsf{MAC\text{-}sFORGE}}_{\mathcal{A},M}(\lambda)$.

The standard security requirement for MACs is that a $\mathsf{poly}(\lambda)$-time adversary must not be able to create new valid tags on messages which have not been honestly authenticated previously. A stronger notion asks that the adversary must not be able to forge a new tag on messages *even if* such messages have been authenticated before. This notion is captured in the experiment described in Figure 4 for a MAC M and arbitrary $\mathsf{poly}(\lambda)$-time adversary $\mathcal{A}$. We denote $\mathcal{A}$'s advantage in the $\mathsf{MAC\text{-}sFORGE}$ security game as

$$\mathbf{Adv}_{\mathcal{A},M}^{\mathsf{MAC\text{-}sFORGE}}(\lambda) = \Pr\left[\mathbf{Exp}_{\mathcal{A},M}^{\mathsf{MAC\text{-}sFORGE}}(\lambda) = 1\right]$$

Definition 4 (Strong Unforgeability). *A message authentication code* $M = (\mathsf{KGen}, \mathit{Mac}, \mathit{Vrfy})$ *is* strongly unforgeably under an adaptive chosen-message attack*, or* strongly secure*, if for all probabilistic* $\mathsf{poly}(\lambda)$*-time adversaries* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\mathbf{Adv}_{\mathcal{A},M}^{\mathit{MAC\text{-}sFORGE}}(\lambda) \leq \mathsf{negl}(\lambda)\,.$$

Key derivation functions and the random oracle model. In cryptographic schemes such as key agreement protocols, the secret information that is exchanged often cannot be used "out of the box" to achieve other goals such as encryption or authentication. Instead, we must use a method to transfer the high entropy of the key agreement session key into a format that is more suitable. This is acheived by making use of *key derivation functions* (KDFs) which are functions with high min-entropy, i.e. an adversary has a negligible chance of correctly guessing the output computed from a given input. While in practice great care must be given to the instanciation of such a KDF, we will make use here of the *random oracle model* and assume that the KDFs we use sample their output uniformly at random from a given space. We will use two independent random oracles which we will denote by H_1 and H_2.

2.2 Passively Secure Unauthenticated Key Agreement Protocol

First, we formalise what we mean by a (simple) unauthenticated key agreement protocol and what it means for such a protocol to be passively secure. Informally we consider a protocol passively secure if an adversary cannot determine the session key from seeing a transcript. We make no usage of long term keys at this stage, as we are focusing on unauthenticated protocols. In a later section we will discuss the model for fully actively secure, and authenticated, key agreement.

Informally, a key agreement protocol is a set of instructions, executed by two parties involved in a conversation, which leads to both of them computing identical session keys. These keys are then usually used to authenticate or encrypt further communication. The most basic security notion expected of such a protocol is that an adversary who has access to the transcript of a conversation is incapable of obtaining any information regarding the final session key. Our formalisation below is inspired by the original definition of such protocols by Bellare and Rogaway [4].

Definition 5 (Unauthenticated Key Agreement Protocol). *An* unauthenticated key agreement protocol *is a pair of probabilisitc* $\mathsf{poly}(\lambda)$*-time algorithms* (Setup, Π) *such that:*

1. *The* setup *algorithm* Setup *takes as input the security parameter* 1^λ *and outputs a tuple of public parameters,* params*, required by the key agreement protocol. Amongst other information* params *specifies a message space* $\mathcal{M}$ *and a key space* $\mathcal{K}$*. We assume for convenience that* λ *is implicit in* params*.*
2. *The* protocol function Π *is a function that dictates which messages the participating entities should compute and send to one another. Its input and output are of the form* $(\epsilon', m, \delta, \kappa) \leftarrow \Pi(\mathsf{params}, \rho, \tau, \epsilon; \$)$ *where the inputs are defined by:*
   - params *are the system parameters.*
   - $\rho \in \{\mathsf{init}, \mathsf{resp}\}$ *is the role of the entity running the function.*
   - $\tau \in \{0,1\}^*$ *is a transcript of the conversation so far.*
   - $\epsilon \in \{0,1\}^* \cup \{\perp\}$ *is ephemeral state information which needs to be passed from one party's invocation of* Π *to the next.*
   - $\$$ *is some randomness.*

   And the outputs of Π *are given by*
   - $\epsilon' \in \{0,1\}^* \cup \{\perp\}$ *is updated state ephemeral information, if any.*
   - $m \in \mathcal{M} \cup \{\perp, \dashv\}$ *is the next message to be sent in the conversation, where* $\dashv$ *signifies that no further message needs to be sent.*
   - $\delta \in \{\mathsf{accept}, \mathsf{reject}, *\}$ *indicates* U*'s decision in the current conversation. The symbol* $*$ *signifies a decision has not yet been made. If* $\delta = \mathsf{reject}$ *is returned then* ϵ' *and* m *are set to* $\perp$ *and* κ *must be equal to* $*$*.*
   - $\kappa \in \mathcal{K} \cup \{*\}$ *is the secret session key computed, where* $*$ *denotes that it has not been computed yet.*

We often abuse notation and use the symbol Π to denote both the protocol function and the entire protocol (Setup, Π) and we assume that params is made implicit in the use of Π. See Figure 1 for a two round example; which will be the focus of this paper.

An unauthenciated key agreement protocol is said to be *correct* if when the messages are relayed faithfully, i.e. unmodified and in the correct order, between two participants, then they both accept and compute identical session keys, except with negligible probability over the randomness used in the algorithms.

In practice one defines a specific key agreement protocol by defining how each new input message is responded to, given the current player state ϵ. We implicitly assume that if the input state is $\perp$, then the output state and message are also $\perp$ and δ will be reject.

For such unauthenticated key agreement protocol the best security guarantee we can obtain is that of passive security. Such a protocol is said to be passively secure if a single session of the protocol does not leak any information regarding the computed session key to an arbitrary $\mathsf{poly}(\lambda)$-time adversary $\mathcal{A}$ that only eavesdrops on the conversation. For an unauthenticated key agreement protocol Π and an adversary $\mathcal{A}$, this is formalised in the $\mathsf{EAV\text{-}KA}$ experiment described in Figure 5. We denote $\mathcal{A}$'s advantage in the $\mathsf{EAV\text{-}KA}$ game as $\mathbf{Adv}^{\mathsf{EAV\text{-}KA}}_{\mathcal{A},\Pi}(\lambda) = \left|\frac{1}{2} - \Pr\left[\mathbf{Exp}^{\mathsf{EAV\text{-}KA}}_{\mathcal{A},\Pi}(\lambda) = 1\right]\right|$.

Definition 6 (Passive KA Security). *A key agreement protocol* Π *is* passively secure in the presence of an eavesdropper *if for all probabilistic* $\mathsf{poly}(\lambda)$*-time adversaries* $\mathcal{A}$*, the following conditions hold.*

1. Two parties holding 1^λ execute protocol Π with one another. This results in a transcript tran of the entire conversation, and a key κ output by each of the parties.
2. A uniform bit $b \in \{0,1\}$ is chosen. If $b = 0$, set $\hat{\kappa} := \kappa$, and if $b = 1$ then sample $\hat{\kappa} \leftarrow_{\$} \mathcal{K}$ uniformly at random.
3. $\mathcal{A}$ is given tran and $\hat{\kappa}$, and outputs a guess bit b'.
4. The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise.

Fig. 5: The $\mathsf{EAV\text{-}KA}$ Security Experiment $\mathbf{Exp}^{\mathsf{EAV\text{-}KA}}_{\mathcal{A},\Pi}(\lambda)$.

1. *If messages are relayed faithfully by a benign adversary between two participant oracles, then both oracles accept holding identical session keys, and each participant's key is distributed uniformly at random over $\mathcal{K}$.*
2. *There exists a negligible function* $\mathsf{negl}(\lambda)$ *such that* $\mathbf{Adv}^{\mathsf{EAV\text{-}KA}}_{\mathcal{A},\Pi}(\lambda) \leq \mathsf{negl}(\lambda)$.

It is an easy exercise to see that our syntax captures the syntax of Diffie–Hellman, Frodo and NewHope. In addition it is another easy exercise to show that the standard unauthenticated Diffie–Hellman protocol meets our Passive KA Security defnition, assuming the Decision Diffie–Hellman problem is hard. In addition it is relatively easy to check that the proofs of security of the Frodo and NewHope key agreement schemes, given in [7,1], also imply security for our Passive KA definition.

Minor Active Security Properties We also introduce two simple active security notions relevant to KA protocols. Most well designed passive KA schemes are implicitly understood to satisfy these two notions, but we choose to make them explicit (with the definition of two new security experiments) as we shall require them later on.

The first of these formalises the notion of the first protocol message being sufficiently "unpredictable"; i.e. the adversary is not able to guess what the first message m_1 of the transcript tran is going to be. We define the $\mathsf{M1\text{-}GUESS}$ experiment in Figure 6 and denote an arbitrary adversary $\mathcal{A}$'s advantage in that game as

$$\mathbf{Adv}^{\mathsf{M1\text{-}GUESS}}_{\mathcal{A},\Pi}(\lambda) = \Pr\left[\mathbf{Exp}^{\mathsf{M1\text{-}GUESS}}_{\mathcal{A},\Pi}(\lambda) = 1\right].$$

1. One party holding 1^λ computes $(\epsilon', m_1, *, *) \leftarrow \Pi(\mathsf{params}, \mathsf{init}, \emptyset, \bot; \$)$.
2. $\mathcal{A}$ is given 1^λ and params and outputs a guess message m_1'.
3. The output of the experiment is defined to be 1 if $m_1' = m_1$, and 0 otherwise.

Fig. 6: The $\mathsf{M1\text{-}GUESS}$ Security Experiment $\mathbf{Exp}^{\mathsf{M1\text{-}GUESS}}_{\mathcal{A},\Pi}(\lambda)$.

The second security notion models the property that an adversary should not be able to obtain information about the final key κ even if it may choose the first protocol message. This definition applies only to two-messages KA protocols. To this intent, we define the experiment KEY-FORCE in Figure 7 and denote an arbitrary adversary $\mathcal{A}$'s advantage as

$$\mathbf{Adv}_{\mathcal{A},\Pi}^{\mathsf{KEY\text{-}FORCE}}(\lambda) = \Pr\left[\mathbf{Exp}_{\mathcal{A},\Pi}^{\mathsf{KEY\text{-}FORCE}}(\lambda) = 1\right]$$

1. The challenger sets 1^λ and runs Setup to obain params.
2. $\mathcal{A}$ is given 1^λ and params and ouputs a first message m_1.
3. If $m_1 \notin \mathcal{M}$ the experiment outputs 0. Otherwise, the challenger computes $(\bot, m_2, \delta, \kappa_0) \leftarrow \Pi(\mathsf{params}, \mathsf{resp}, \{m_1\}, \bot; \$)$, together with sampling $\kappa_1 \leftarrow_{\$} \mathcal{K}$, from the KE key space.
4. A bit $b \leftarrow_{\$} \{0, 1\}$ is chosen uniformly at random.
5. $\mathcal{A}$ is given κ_b and returns a guess $\tilde{b}$.
6. The experiment outputs 1 if and only if $\tilde{b} = b$, and 0 otherwise.

Fig. 7: The KEY-FORCE Security Experiment $\mathbf{Exp}_{\mathcal{A},\Pi}^{\mathsf{KEY\text{-}FORCE}}(\lambda)$.

3 Forward-secure Authenticated Key Agreement Protocols and Security Model

In this section we recall the active security model and definitions used to caracterise secure Authenticated Key Agreement (AKA) protocols. The material we present here is a slight reformulation of Kudla's BJM and mBJM models [23] which were themselves an elaboration of Bellare and Rogaways's original model [4] and of Blake-Wilson *et al.*'s formulation for the public-key setting [6]. In particular, we add the appropriate elements so that forward secrecy is captured by our model.

An AKA protcol is no different in its formal structure from a simpler KA protocol in that an AKA protocol Π or Σ consists of a tuple $(\mathsf{Setup}, \mathsf{KGen}, \Pi(\text{or } \Sigma))$ of $\mathsf{poly}(\lambda)$-time algorithms which are used by each party to obtain long-term keying information and execute sessions of the protocol to establish shared session keys. The only formal difference lies in the security conditions that an AKA protocol satisfies.

Most significanlty AKA security is an *active* notion, and therefore the execution environment for the security experiment is much more complex. We first recall the components of this environment before describing the security notions of *secure mutual authentication*, *session key secrecy* and *forward secrecy*.

3.1 Execution environment

In security experiments, the role of the challenger is to simulate an honest real-world participant to an arbitrary adversary. For the BJM model, and distributed protocols such as AKA ones, this implies that the challenger needs to simulate $n_P = \mathsf{poly}(\lambda)$ participants, grouped in a set $\mathcal{U}$ of identities. Furthermore, as we are dealing with an active adversary, the challenger needs to enable it to engage with the simulated participants.

This is achieved via the mean of *oracles* which represent individual sessions (or "runs") of the protocol. By $\Pi_{U,V}^s$, we denote the oracle representing the s-th run of party U believing it is interacting with party V according to protocol Π, where $s \in$

$\{1, \ldots, n_S\}$ for $n_S = \mathsf{poly}(\lambda)$. In order to maintain a coherent execution environment between all of the oracles, the challenger simulates their access to their own keying information as well as the public keying information of other parties. Furthermore, each oracle maintains a record of messages it has received and sent, of any decision it might have come to, and of any session key it might have computed.

As mentionned above, the adversary has access to the oracles that the challenger maintains. More specifically, the adversary can use the **Send** query to send an oracle a message of its choice, the **Reveal** query to ask for the session key of an oracle if it has been derived, and the **Corrupt** query to obtain the long-term public *and* private keying information of an oracle and eventually replace it with different keying information.

3.2 Secure Mutual Authentication

As is made explicit in its naming, an AKA protocol aims to provide authentication of the parties involved. Here we work with the notion of mutual authentication from [4] where both parties expect to have authenticated one another upon terminating the protocol.

To achieve this, Bellare and Rogaway present the concept of *matching conversation* that give us a criterion to decide whether two oracles have engaged in a protocol session together. This criterion first checks that either oracle believes it is engaging with the identity represented by the other and then verifies that the incoming messages of one match the outgoing messages of the other and vice versa.

For an AKA protocol Π and an arbitrary adversary $\mathcal{A}$, we define the $\mathsf{AKA\text{-}AUTH}$ experiment as follows. The challenger runs $\mathsf{Setup}(\lambda)$ to obtain params and then uses KGen to obtain the keying information for n_P parties. $\mathcal{A}$ is given params, all the plublic keys and access to the oracles via the queries described in Section 3.1. The challenger responds to $\mathcal{A}$'s queries according to the protocol until $\mathcal{A}$ outputs a session $\Pi^s_{U,V}$. The ouput of the experiment is defined to be 1 if that session has accepted, neither U nor V is corrupted and there does not exists another session with which $\Pi^s_{U,V}$ has had a matching conversation, and 0 otherwise. The advantage of $\mathcal{A}$ against the $\mathsf{AKA\text{-}AUTH}$-security of Π is defined to be the probability that the above experiment outputs 1.

The definition of $\mathsf{AKA\text{-}AUTH}$-security is therefore that for any arbitrary $\mathsf{poly}(\lambda)$-time adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\mathbf{Adv}^{\mathsf{AKA\text{-}AUTH}}_{\mathcal{A},\Pi}(\lambda) \leq \mathsf{negl}(\lambda)\,.$$

The definition also requires correctness; i.e. that if any two oracles engage in a matching conversation, then they both terminate having accepted the session.

3.3 Session Key Secrecy and Forward Secrecy

The next experiment we describe, $\mathsf{AKA\text{-}SEC}$, is used to capture the secrecy notion expected of AKA protocols, namely that the agreed key should only be known by the parties who have engaged in the session together. It is a natural combination of the active $\mathsf{AKA\text{-}AUTH}$ experiment together with the passive notion of $\mathsf{EAV\text{-}KA}$ security.

The challenger generates and simulates the execution environment to the adversary via oracles, similarly to the $\mathsf{AKA\text{-}AUTH}$ experiment, until the adversary returns a session $\Pi^s_{U,V}$ on which it wishes to be *tested.* At this moment, the challenger samples a uniformly random session key and, using a coin flip, gives to the adversary either the true session key derived by $\Pi^s_{U,V}$ or the random one. The challenger then continues to simulate the environment and answer the adversary's queries until the latter outputs its guess as to which key it was given. The $\mathsf{AKA\text{-}SEC}$ experiment outputs 1 if the adversary guesses correctly, and 0 otherwise.

In order to prevent the adversary from winning the experiment trivially, the challenger requires that the session $\Pi^s_{U,V}$ is *fresh*. This requires that this session's key has not been revealed to the adversary, that neither U nor V has been corrupted, and that there does not exist another session which has had a matching conversation with $\Pi^s_{U,V}$ and had its session key revealed. Note that this does not require the test session to have a matching partner, it is still considered fresh even if the adversary has made it accept without a matching conversation.

Furthermore, we do not require that the chosen session remains fresh after the adversary has been given its challenge key. It still may not trivially reveal the true session key, but it may corrupt either, or both, of the parties involved. This ability of the adversary enables the $\mathsf{AKA\text{-}SEC}$ notion of security to capture *forward secrecy*.

We denote an arbitrary adversary $\mathcal{A}$'s advantage in the $\mathsf{AKA\text{-}SEC}$ security game by

$$\mathbf{Adv}^{\mathsf{AKA\text{-}SEC}}_{\mathcal{A},\Pi}(\lambda) = \left| \frac{1}{2} - \Pr\left[\mathbf{Exp}^{\mathsf{AKA\text{-}SEC}}_{\mathcal{A},\Pi}(\lambda) = 1\right] \right| .$$

The complete definition of AKA-security is as follows. The protocol Π is *correct*: if messages are relayed faithfully, both parties terminate by accepting identical session keys that are correctly distributed. Protocol Π is a *secure mutual authentication protocol*. For all $\mathsf{poly}(\lambda)$-time adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\mathbf{Adv}^{\mathsf{AKA\text{-}SEC}}_{\mathcal{A},\Pi}(\lambda) \leq \mathsf{negl}(\lambda) .$$

As mentioned briefly above, the most notable characteristic of our security defintion for AKA protocols is that it captures the property known as *forward secrecy*. This property requires that the compromise of long-term secret keying information of entities does not allow an adversary to obtain any information regarding past session keys that these entities might have established.

This is captured in our model since the adversary is allowed, before it makes its final guess, to submit a **Corrupt** query on the entities that took part in the test session. With that possibility in mind, we still require that its advantage in the $\mathsf{AKA\text{-}SEC}$ experiment remains negligible. Thus, proving that an AKA protocol satisfies our deifnition of security also proves that it possesses forward secrecy, in which case we say it is a *forward secure AKA protocol*. Additionally, our definition also captures the usual security properties of AKA protocol such as session-key reveal secrecy and third-party compromise security.

4 A New AKA Protocol Construction

We now present in more detail our new construction of a secure AKA protocol. We also state the theorems that establish secrecy for keys and the level of authentication that our protocol offers.

The construction: Let $E = (\mathsf{Setup}_E, \mathsf{KGen}_E, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme. Let $M = (\mathsf{KGen}_M, \mathsf{Mac}, \mathsf{Vrfy})$ be a message authentication code such that its key space is $\mathcal{K}_M = \{0,1\}^{l(\lambda)}$ for some polynomial function l, and its KGen_M algorithm simply selects a key from $\mathcal{K}_M$ uniformly at random. Let $\Pi = (\mathsf{Setup}_\Pi, \Pi)$ be a two-round key agreement protocol and finally, let $H_1 : \{0,1\}^* \rightarrow \{0,1\}^{l(\lambda)}$ and $H_2 : \{0,1\}^* \rightarrow \{0,1\}^{h(\lambda)}$, where h is a polynomial function, be two key derivation functions. Using these elements, we construct the AKA protocol $\Sigma = (\mathsf{Setup}_\Sigma, \mathsf{KGen}_\Sigma, \Sigma)$ where:

1. Setup_Σ takes as input the security paramter 1^λ and outputs public parameters params_Σ which contain the parameters of the encryption scheme E output by $\mathsf{Setup}_E(1^\lambda)$ and the parameters of the KA protocol Π output by $\mathsf{Setup}_\Pi(1^\lambda)$.
2. KGen_Σ takes as input params_Σ and an indentifier U. It then outputs a public/private key pair for U by setting $(\mathsf{pk}_U, \mathsf{sk}_U) \leftarrow \mathsf{KGen}_E(\mathsf{params}_E)$, i.e. a normal public-key encryption scheme key pair.
3. Σ functions as specified by the protcol run described in Figure 2. The protocol works by first wrapping the message flows, m_1 and m_2, of the unauthenticated key agreement in encryptions to each party and then sending a MAC tag on the identities under a key derived from the key agreement session key using the KDF H_1. The final AKA session key is derived from the underlying agreed key and the party identities, using a different KDF H_2.

Security of our scheme: Authentication of Bob to Alice is obtained by Bob prefixing the plaintext m_1 to his response m_2 in the second message flow $\mathfrak{m}_2$. In this way Alice can verify that the message m_1' that she receives is identical to the one she sent out, i.e. m_1, and therefore Bob must have decrypted it; since only Bob has Bob's decryption key. Authentication of Alice to Bob is obtained by Alice sending a valid MAC on the identities under a key derived from the underlying unauthenticated key agreement scheme. Since only Alice can decrypt Bob's message m_2, only Alice could compute the underlying key agreement session key and therefore the associated MAC key. Notice that the these forms of authentication also imply liveness of the parties. The above intuition is formalized by the following theorem.

Theorem 2. *If Π is $\mathsf{M1\text{-}GUESS}$-secure and $\mathsf{KEY\text{-}FORCE}$-secure, E is 2-$\mathsf{IND\text{-}CCA}$-secure and M is $\mathsf{MAC\text{-}sFORGE}$-secure, then Σ is a secure mutual authentication protocol.*

Proof. We prove correctness first and AKA-AUTH-security second.

[Matching Conversation $\Rightarrow$ Acceptance].
Suppose two oracles $\Sigma^s_{U,V}$ and $\Sigma^{s'}_{U',V'}$ have matching conversations. This first implies $U' = V$ and $V' = U$. Furthermore, the three protocol messages $\mathfrak{m}_1, \mathfrak{m}_2$ and $\mathfrak{m}_3$ are relayed faithfully.

Since V receives the same $\mathfrak{m}_1$ that U sent out, by the correctness of the encryption scheme E, V decrypts m_1 correctly and prefixes m_2 with the correct message. Hence, since U receives the same $\mathfrak{m}_2$ that V sent out, the correctness of E implies again that indeed $m_1' = m_1$ and therefore that U does not reject this session when it receives $\mathfrak{m}_2$.

As U does not receive another response from V before it terminates, we see that U accepts this session.

As U has received the same $\mathfrak{m}_2$ that V has sent out, it receives the same m_2 and therefore the correctness of Π implies that $\kappa_U = \kappa_V$. By equality of the inputs to the KDF H_1, we have that $\mathsf{k}_{U,1} = \mathsf{k}_{V,1}$. Also, since V receives the same $\mathfrak{m}_3$ that U has sent out, the correctness M and the equality of the keys imply that $\mathsf{Vrfy}_{\mathsf{k}_{V,1}}(U\|V, \mathsf{Mac}_{\mathsf{k}_{U,1}}(U\|V)) = 1$ and hence V does not reject the session when it receives $\mathfrak{m}_3$. As V does not receive another response from U before it terminates, we see that V accepts this session as well. This completes the proof of correctness.

[Acceptance ⇒ Matching Conversation].

We now prove that an arbitrary $\mathsf{poly}(\lambda)$-time adversary $\mathcal{A}$ has a negligible chance of winning the AKA-AUTH security experiment. We first observe that the following holds:

$$\mathbf{Adv}^{\mathsf{AKA\text{-}AUTH}}_{\mathcal{A},\Sigma}(\lambda) \leq \Pr\left[\mathcal{A} \text{ makes } \mathsf{init} \text{ accept}\right] + \Pr\left[\mathcal{A} \text{ makes } \mathsf{resp} \text{ accept}\right].$$

Bounding success against initiator. We proceed via a series of Games.

Game 0 is the initial AKA-AUTH security experiment with the difference that the adversary loses by default if it returns a responder session.

Game 1. The challenger first selects a random session $\Sigma^s_{U,V}$ and then runs Game 0 with the adversary. The adversary loses by default if it outputs a session different from $\Sigma^s_{U,V}$.

Game 2. The challenger runs Game 1 with the adversary but replaces the message m_1, honestly generated by $\Sigma^s_{U,V}$, by a random KA message m_1' in the rest of the protocol (including in the computation of m_2). However, $\Sigma^s_{U,V}$ still only accepts if it receives the honest m_1 in the second message flow.

We denote by G_i the event that Game i outputs 1, for which we sometimes say that an arbitrary adversary $\mathcal{A}$ "wins" Game i. Our aim is to obtain an upperbound for $\Pr[G_0]$.

Game 0 to Game 1. First, we see that $\mathcal{A}$ wins Game 1 exactly when it wins Game 0 and the challenger has guessed the output session correctly out of $n_P^2 \cdot n_S$ possibilities. We therefore have

$$\Pr[G_1] = \frac{1}{n_P^2 \cdot n_S} \cdot \Pr[G_0]. \tag{1}$$

Game 1 to Game 2. Next, we build a reduction $\mathcal{B}_1$ that uses an adversary that is able to distinguish between Game 1 and Game 2 to attack the 2-IND-CCA security of the encryption scheme E.

The reduction uses the L-R oracle given by the 2-IND-CCA challenger to encrypt either m_1 or m_1' in the protocol flows. Whenever the output of the L-R oracle is submited to a session, the reduction uses both m_1 and m_1' to compute two responses and encrypts either $m_1||m_2$ or $m_1'||m_2'$ using the L-R oracle. If the secret bit $b = 0$, then it is the honest m_1 that is used throughout the protocol thus simulating Game 1 perfectly. If the secret bit is 1, m_1' is used and Game 2 is simulated. If $\mathcal{A}$ still manages to make $\Sigma^s_{U,V}$ accept without a matching conversation, by submitting a message $\widetilde{\mathfrak{m}}_2$ containing the honest m_1, then the reduction returns $b' = 1$ to the 2-IND-CCA challenger.

Therefore we see that we have

$$\mathbf{Adv}^{\text{2-IND-CCA}}_{\mathcal{B}_1,E}(\lambda) = \left|\Pr\left[\mathbf{Exp}^{\text{2-IND-CCA-0}}_{\mathcal{B}_1,E}(\lambda) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{2-IND-CCA-1}}_{\mathcal{B}_1,E}(\lambda) = 1\right]\right|$$

$$= |\Pr[G_1] - \Pr[G_2]|$$

Bounding Game 2. Finally we see that an adversary can only win Game 2 by guessing the honest m_1 that $\Sigma^s_{U,V}$ will accept since no information regarding m_1 is contained within the message flows. Given an adversary $\mathcal{A}$ capable of winning Game 2, we can easily build a reduction $\mathcal{B}_2$ that plays the M1-GUESS experiment against the KA protocol Π. This reduction simulates Game 2 to $\mathcal{A}$ and simply returns whichever m_1 is contained within $\mathfrak{m}_2$. As the reduction is given no information regarding the message it has to guess, the adversary is equally likely to guess the correct message. We thus have

$$\mathbf{Adv}^{\mathsf{M1\text{-}GUESS}}_{\mathcal{B}_2,\Pi}(\lambda) = \Pr[G_2]$$

Advantage terms. As E is 2-IND-CCA-secure and Π is M1-GUESS-secure, there exist negligible functions such that

$$|\Pr[G_1] - \Pr[G_2]| = \mathbf{Adv}^{\mathsf{2\text{-}IND\text{-}CCA}}_{\mathcal{B}_1,E}(\lambda) \leq \mathsf{negl}(\lambda)$$
$$\text{and} \quad |\Pr[G_2]| = \mathbf{Adv}^{\mathsf{M1\text{-}GUESS}}_{\mathcal{B}_2,\Pi}(\lambda) \leq \mathsf{negl}(\lambda)\,.$$

This, combined with (1), yields

$$\Pr[G_0] = \Pr[\mathcal{A} \text{ makes init accept}] \leq \mathsf{negl}(\lambda)\,. \tag{2}$$

Bounding success against initiator. We also proceed via a series of Games.

Game 0 is the initial AKA-AUTH security experiment with the difference that the adversary loses by default if it returns an initiator session.

Game 1. The challenger first selects a random session $\Sigma^s_{U,V}$ and then runs Game 0 with the adversary. The adversary loses by default if it outputs a session different from $\Sigma^s_{U,V}$.

Game 2. In all of the message flows sent by initiator sessions of V intended for U and responder sessions of U intended for V (including $\Sigma^s_{U,V}$), the challenger replaces the honest KA protocol messages m_1 and m_2 by independently randomly sampled messages m'_1 and m'_2. Internally, the oracles still compute and verify with the honest messages and keys. The victory conditions for $\mathcal{A}$ are the same as for Game 1.

Game 3. The challenger runs Game 2 with the adversary but replaces the KA key κ_V for the chosen session $\Sigma^s_{U,V}$ derived from the honestly computed messages by a uniformly sampled key $\kappa' \leftarrow\!\!\$\, \mathcal{K}$ (independent from the random message m'_2). The session $\Sigma^s_{U,V}$ only accepts a MAC tag validly computed under κ'.

Game 4. The challenger runs Game 3 with the adversary but replaces the MAC key $\mathsf{k}_{V,1}$ derived from the uniformly sampled κ' by a uniformly sampled MAC key k'. The session $\Sigma^s_{U,V}$ only accepts a MAC tag validly computed under k'.

Game 0 to Game 1. First, we see that $\mathcal{A}$ wins Game 1 exactly when it wins Game 0 and the challenger has guessed the output session correctly out of $n_P^2 \cdot n_S$ possibilities. We therefore have

$$\Pr[G_1] = \frac{1}{n_P^2 \cdot n_S} \cdot \Pr[G_0]. \tag{3}$$

Game 1 to Game 2. In order to bound the difference between Game 1 and Game 2, we build the reduction $\mathcal{B}_3$ to attack the 2-IND-CCA security of the encryption scheme E. As for the previous cases, our aim is to build a reduction that simulates Game 1 or Game 2 perfectly when the L-R oracle's bit is 0 or 1. To this effect, $\mathcal{B}_3$ proceeds as follows.

Whenever an oracle simulating an initiator session of V with U is activated, the reduction queries the L-R oracle to send either an honestly computed m_1 or a randomly sampled m_1'. It also records both messages and the resulting ciphertext in a list L_1.

Whenever an oracle simulating a responder session of U with V receives a ciphertext that appears on L_1, it is able to compute an honest response, thus deriving a session key, sample a random one and send either an honest message $m_1 \| m_2$ or the random $m_1' \| m_2'$ using the L-R oracle. If the ciphertext does not appear on L_1, it can be decrypted using Dec to obtain the plaintext. If that plaintext does not appear on L_1, then, as before, $\mathcal{B}_3$ derives a key and samples a uniform m_2' before replying with either the honest or random message (unless the content of the plaintext deviates from the protocol). Whenever $\mathcal{B}_3$ replies, it records both plaintexts, the session key and the resulting ciphertext on a list L_2. If, however, the decrypted plaintext appears on L_1, then $\mathcal{B}_3$ learns the bit of the challenger oracle.

Finally, if an oracle simulating an initiator session of V with U receives a ciphertext (as a second protocol message) that appears on L_2, then it is able to check if the accompanying plaintexts match the ones computed / sampled earlier. If they do, $\mathcal{B}_3$ uses the accompanying session key to proceed. If they don't, then this ciphertext was not intended for this session and the oracle rejects. If the ciphertext received does not appear on L_2, it can be decrypted using Dec to obtain the plaintext. If that plaintext appears on L_2, $\mathcal{B}_3$ learns the bit of the challenger oracle. If it does not, $\mathcal{B}_3$ checks if the first message matches either of the plaintexts computed / sampled earlier. If it does, $\mathcal{B}_3$ learns the bit of the challenger oracle. If it doesn't, then the ciphertext is an invalid protocol message and the oracle rejects.

If the adversary wins the experiment under the same conditions as for Game 1 and 2, $\mathcal{B}_3$ returns 1 to the 2-IND-CCA-b challenger. Let $\mathcal{E}_1$ denote the event that an oracle receives a ciphertext that leads to $\mathcal{B}_3$ learning the challenger bit. We then have

$$\begin{aligned}\mathbf{Adv}^{\textsf{2-IND-CCA}}_{\mathcal{B}_3,E}(\lambda) = \Big| 0 \cdot \Pr[\mathcal{E}_1] + \Pr\left[\mathbf{Exp}^{\textsf{2-IND-CCA-0}}_{\mathcal{B}_3,E}(\lambda) = 1 \mid \neg\mathcal{E}_1\right] \cdot \Pr[\neg\mathcal{E}_1] - \\ 1 \cdot \Pr[\mathcal{E}_1] - \Pr\left[\mathbf{Exp}^{\textsf{2-IND-CCA-1}}_{\mathcal{B}_3,E}(\lambda) = 1 \mid \neg\mathcal{E}_1\right] \cdot \Pr[\neg\mathcal{E}_1] \Big|\end{aligned}$$

which implies

$$\mathbf{Adv}^{\textsf{2-IND-CCA}}_{\mathcal{B}_3,E}(\lambda) + \Pr[\mathcal{E}_1] \geq |\Pr[G_1] - \Pr[G_2]| \cdot (1 - \Pr[\mathcal{E}_1]) \,. \tag{4}$$

It is easy to see from $\mathcal{B}_3$ how one can build a reduction $\mathcal{B}_3'$ which wins the 2-IND-CCA experiment against E whenever $\mathcal{E}_1$ happens. This allows us to write

$$\mathbf{Adv}^{\textsf{2-IND-CCA}}_{\mathcal{B}_3',E}(\lambda) = \Pr[\mathcal{E}_1]. \tag{5}$$

Game 2 to Game 3. The reduction $\mathcal{B}_4$ acts as an attacker in a KEY-FORCE experiment against the KA protocol $\mathit{\Pi}$. To simulate Game 2, it chooses a session $\Sigma^s_{U,V}$ at random and runs the execution environment, sampling protocol messages at random where required, maintaining consistency. When $\Sigma^s_{U,V}$ receives a ciphertext, $\mathcal{B}_4$ decrypts it and checks whether the plaintext was sampled by an initiator oracle $\Sigma^t_{V,U}$. If it was, then $\mathcal{B}_4$ recovers the honest m_1 that was computed and returns it to the KEY-FORCE challenger. If it was not, then $\mathcal{B}_4$ returns the plaintext it decrypted. When the KEY-FORCE challenger returns a key $\widetilde{\kappa}$, $\mathcal{B}_4$ sets $\kappa_V = \widetilde{\kappa}$ and proceeds with its simulation of Game 2. If $\mathcal{A}$ wins the simulation of Game 2 according to the same conditions, $\mathcal{B}_4$ returns $\widetilde{b} = 1$ to the challenger.

It is easy to see that if the KEY-FORCE challenger returns the real key, then $\mathcal{B}_4$ simulates Game 2 perfectly, and that if the it returns a random KA key, then it is Game 3 that is simulated perfectly. We therefore have

$$\begin{aligned}\mathbf{Adv}^{\mathsf{KEY\text{-}FORCE}}_{\mathcal{B}_4,\Pi}(\lambda) &= \left|\Pr\left[\mathbf{Exp}^{\mathsf{KEY\text{-}FORCE\text{-}0}}_{\mathcal{B}_4,\Pi}(\lambda)=1\right] - \Pr\left[\mathbf{Exp}^{\mathsf{KEY\text{-}FORCE\text{-}1}}_{\mathcal{B}_4,\Pi}(\lambda)=1\right]\right| \\ &= \left|\Pr[G_2] - \Pr[G_3]\right|.\end{aligned}$$

Game 3 to Game 4. We see that, as κ' is selected uniformly at random, the derived MAC key $\mathsf{k} = H_1(\kappa')$ follows the same distribution. Therefore this is a simple re-wording and we have

$$\Pr[G_4] = \Pr[G_3]. \tag{6}$$

Bounding Game 4. Finally, we see that Game 4 is very similar to the MAC- sFORGE game against the MAC M. Indeed, the adversary has no information regarding the MAC key k' used by the oracles except perhaps a valid tag $\mathfrak{m}_3$ if it relayed the two first messages correctly. We therefore make use of a final reduction $\mathcal{B}_5$ to bound $\Pr[G_4]$.

This reduction works as follows. If the adversary presents the choosen session $\Sigma^s_{U,V}$ with a first KA message m'_1, $\mathcal{B}_5$ replies by inserting a random message m'_2 into $\widetilde{\mathfrak{m}}_2$ without deriving a session key. If a instance of V recieves the correct $m'_1\|m'_2$ that matches $\Sigma^s_{U,V}$, then it queries the MAC oracle for a tag $\widetilde{\mathfrak{m}}_3$ on $V\|U$. When $\Sigma^s_{U,V}$ receives a tag $\mathfrak{m}_3 \neq \widetilde{\mathfrak{m}}_3$, it returns $\mathfrak{m}_3$ to the MAC-sFORGE challenger.

We can see from the description above that if $\mathcal{A}$ manages to make a responder accept in Game 4 without a matching conversation, then $\mathcal{B}_5$ is able to win the MAC-sFORGE experiment. We therefore have

$$\mathbf{Adv}^{\mathsf{MAC\text{-}sFORGE}}_{\mathcal{B}_5,M}(\lambda) = \Pr[G_4].$$

Advantage terms. As E is 2-IND-CCA-secure, there exist negligible functions such that

$$\mathbf{Adv}^{\mathsf{2\text{-}IND\text{-}CCA}}_{\mathcal{B}_3,E}(\lambda) \leq \mathsf{negl}(\lambda) \quad \text{and} \quad \mathbf{Adv}^{\mathsf{2\text{-}IND\text{-}CCA}}_{\mathcal{B}'_3,E}(\lambda) \leq \mathsf{negl}(\lambda).$$

Combining this with equations (4) and (5) yields

$$\left|\Pr[G_1] - \Pr[G_2]\right| \leq \mathsf{negl}(\lambda).$$

As Π is KEY-FORCE-secure, there exists a negligible function such that

$$\left|\Pr[G_2] - \Pr[G_3]\right| = \mathbf{Adv}^{\mathsf{KEY\text{-}FORCE}}_{\mathcal{B}_4,\Pi}(\lambda) \leq \mathsf{negl}(\lambda).$$

Finally, as M is $\mathsf{MAC\text{-}sFORGE}$-secure, there exists a negligible function such that

$$|\Pr[G_4]| = \mathbf{Adv}^{\mathsf{MAC\text{-}sFORGE}}_{\mathcal{B}_5,M}(\lambda) \leq \mathsf{negl}(\lambda)\,.$$

Combining the above, together with (3) and (6), yields

$$\Pr[G_0] = \Pr[\mathcal{A} \text{ makes } \mathsf{resp} \text{ accept}] \leq \mathsf{negl}(\lambda)\,. \tag{7}$$

Final advatage statement. Combining (2) and (7) yields

$$\mathbf{Adv}^{\mathsf{AKA\text{-}AUTH}}_{\mathcal{A},\Sigma}(\lambda) \leq \mathsf{negl}(\lambda)$$

which completes the proof that Σ is a secure mutual authentication protocol.

Finally, we show that our construction yields a protocol that guarantee key secrecy.

Theorem 3. *If Π is $\mathsf{EAV\text{-}KA}$-secure, $\mathsf{M1\text{-}GUESS}$-secure and $\mathsf{KEY\text{-}FORCE}$-secure, E is 2-$\mathsf{IND\text{-}CCA}$-secure and M is $\mathsf{MAC\text{-}sFORGE}$-secure, then Σ is $\mathsf{AKA\text{-}SEC}$- secure.*

Proof. We show that the three conditions required for $\mathsf{AKA\text{-}SEC}$ security hold.

Correctness. The proof of Theorem 2 gives us that if two oracles have matching conversations, then both of them accept. We now show that they derive identical sessions keys which are uniformaly distributed.

As the two oracles have had matching conversations, the two messages m_1, m_2, of the KA protocol have been correctly received by both entities. The correctness of Π therefore implies that the keys derived are identical. Furthermore, the matching conversations also imply that the two entites agree on each other's identities. This all together implies that the inputs to the KDF H_2 are identical for each entity, therefore they derive the same session key.

Finally, as Π is passively secure, we see that the session keys κ_U and κ_V are uniformly distributed over $\mathcal{K}_\Pi$. Therefore, we have that the output of H_2, modelled as a random oracle, is also uniformly distributed over $\mathcal{K}_\Sigma$.

Σ is a secure mutual authentication protocol. This condition is proved by Theorem 2 which holds as we make the same assumptions on the security of Π, E and M.

An arbitrary adversary has negligible advantage against $\mathsf{AKA\text{-}SEC}$. We build a reduction $\mathcal{B}_6$ that uses an arbitrary $\mathsf{poly}(\lambda)$-time adversary $\mathcal{A}$ against the $\mathsf{AKA\text{-}SEC}$ security of Σ in order to win the $\mathsf{EAV\text{-}KA}$ experiment against Π.

$\mathcal{B}_6$ receives a transcript from the $\mathsf{EAV\text{-}KA}$ challenger and guesses which session $\mathcal{A}$ will return. It assumes that this will be a initiator session. In order for it to be fresh, the adversary is cannot have corrupted either participating party and must have made it accept. As shown by Theorem 2, this implies that this session must have had a matching conversation with another. Therefore, when its chosen session is first activated, $\mathcal{B}_6$ inserts the first message of the KA transcript in place of the honestly computed one.

When a session recieves this first ciphertext (and there is one as there must be a matching conversation), $\mathcal{B}_6$ replaces the second message with the one from the KA transcript. It also replaces the session key with that given by the $\mathsf{EAV\text{-}KA}$ challenger. When the second AKA protocol message is relayed to the initiator session, $\mathcal{B}_6$ once again replaces the session key with the challenger's.

When the adversary requests to be tested on the chosen session, $\mathcal{B}_6$ does not flip a coin, instead, it simply returns the challenger's session key. When $\mathcal{A}$ terminates and returns a guess bit, $\mathcal{B}_6$ returns that same bit to the $\mathsf{EAV\text{-}KA}$ challenger.

If $\mathcal{A}$ submits any query that invalidates $\mathcal{B}_6$'s chosen session, the reduction aborts the simulation and guesses the challenger's bit at random.

As with previous reductions, we see that the probability of $\mathcal{B}_6$ picking the correct test session is $1/n_P^2 \cdot n_S$, in which case, $\mathcal{B}_6$ has the same probability of success in the $\mathsf{EAV\text{-}KA}$ game as $\mathcal{A}$. If $\mathcal{B}_6$ guesses the session wrongly, then it must output a random guess which will win with probability $1/2$. Therefore we have:

$$\begin{aligned}\Pr\left[\mathbf{Exp}_{\mathcal{B}_6,\Pi}^{\mathsf{EAV\text{-}KA}}(\lambda)=1\right]-\frac{1}{2} &= \left[\frac{1}{n_P^2\cdot n_S}\cdot\Pr\left[\mathbf{Exp}_{\mathcal{A},\Sigma}^{\mathsf{AKA\text{-}SEC}}(\lambda)=1\right]\right.\\ &\qquad\left.+\left(1-\frac{1}{n_P^2\cdot n_S}\right)\cdot\frac{1}{2}\right]-\frac{1}{2}\\ &= \frac{1}{n_P^2\cdot n_S}\cdot\left(\Pr\left[\mathbf{Exp}_{\mathcal{A},\Sigma}^{\mathsf{AKA\text{-}SEC}}(\lambda)=1\right]-\frac{1}{2}\right).\end{aligned}$$

This then implies that

$$\mathbf{Adv}_{\mathcal{A},\Sigma}^{\mathsf{AKA\text{-}SEC}}(\lambda)=n_P^2\cdot n_S\cdot\mathbf{Adv}_{\mathcal{B}_6,\Pi}^{\mathsf{EAV\text{-}KA}}(\lambda).$$

We conclude the proof by stating that as Π is $\mathsf{EAV\text{-}KA}$-secure, there exists a negligible function $\mathsf{negl}(\lambda)$ which, combined with the above, gives the desired relation,

$$\mathbf{Adv}_{\mathcal{A},\Sigma}^{\mathsf{AKA\text{-}SEC}}(\lambda)\leq\mathsf{negl}(\lambda).$$

References

1. Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 327–343. USENIX Association, 2016.
2. Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 259–274. Springer, Heidelberg, May 2000.
3. Mihir Bellare, Ran Canetti, and Hugo Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract). In *30th ACM STOC*, pages 419–428. ACM Press, May 1998.
4. Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 232–249. Springer, Heidelberg, August 1994.
5. Daniel J. Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. SPHINCS: Practical stateless hash-based signatures. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 368–397. Springer, Heidelberg, April 2015.

6. Simon Blake-Wilson, Don Johnson, and Alfred Menezes. Key agreement protocols and their security analysis. In Michael Darnell, editor, *6th IMA International Conference on Cryptography and Coding*, volume 1355 of *LNCS*, pages 30–45. Springer, Heidelberg, December 1997.
7. Joppe W. Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 16*, pages 1006–1018. ACM Press, October 2016.
8. Colin Boyd, Yvonne Cliff, Juan Gonzalez Nieto, and Kenneth G. Paterson. Efficient one-round key exchange in the standard model. In Yi Mu, Willy Susilo, and Jennifer Seberry, editors, *ACISP 08*, volume 5107 of *LNCS*, pages 69–83. Springer, Heidelberg, July 2008.
9. Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 453–474. Springer, Heidelberg, May 2001.
10. Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal Gaussians. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 40–56. Springer, Heidelberg, August 2013.
11. Marc Fischlin, Felix Günther, Benedikt Schmidt, and Bogdan Warinschi. Key confirmation in key exchange: A formal treatment and implications for TLS 1.3. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 452–469, 2016.
12. Craig Gentry and Michael Szydlo. Cryptanalysis of the revised NTRU signature scheme. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 299–320. Springer, Heidelberg, April / May 2002.
13. Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 112–131. Springer, Heidelberg, August 1997.
14. Felix Günther, Britta Hale, Tibor Jager, and Sebastian Lauer. 0-RTT key exchange with full forward secrecy. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 519–548. Springer, Heidelberg, May 2017.
15. Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. NTRUSIGN: Digital signatures using the NTRU lattice. In Marc Joye, editor, *CT-RSA 2003*, volume 2612 of *LNCS*, pages 122–140. Springer, Heidelberg, April 2003.
16. Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe Buhler, editor, *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, 1998.
17. Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NSS: An NTRU lattice-based signature scheme. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 211–228. Springer, Heidelberg, May 2001.
18. Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. Generic compilers for authenticated key exchange. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 232–249. Springer, Heidelberg, December 2010.
19. Tibor Jager, Kenneth G. Paterson, and Juraj Somorovsky. One bad apple: Backwards compatibility attacks on state-of-the-art cryptography. In *NDSS 2013*. The Internet Society, February 2013.
20. Tibor Jager, Jörg Schwenk, and Juraj Somorovsky. On the security of TLS 1.3 and QUIC against weaknesses in PKCS#1 v1.5 encryption. In Indrajit Ray, Ninghui Li, and Christopher Kruegel:, editors, *ACM CCS 15*, pages 1185–1196. ACM Press, October 2015.

21. Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC Press, 2007.
22. Jonathan Katz and Moti Yung. Scalable protocols for authenticated group key exchange. *Journal of Cryptology*, 20(1):85–113, Jan 2007.
23. Caroline J. Kudla. *Special Signature Schemes and Key Agreement Protocols*. PhD thesis, Royal Holloway University of London, 2006.
24. Yong Li, Sven Schäge, Zheng Yang, Christoph Bader, and Jörg Schwenk. *New Modular Compilers for Authenticated Key Exchange*, pages 1–18. Springer International Publishing, Cham, 2014.
25. Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 319–339. Springer, Heidelberg, February 2011.
26. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Heidelberg, May 2010.
27. Nikos Mavrogiannopoulos, Frederik Vercauteren, Vesselin Velichkov, and Bart Preneel. A cross-protocol attack on the TLS protocol. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 12*, pages 62–72. ACM Press, October 2012.
28. Paul Morrissey, Nigel P. Smart, and Bogdan Warinschi. The TLS handshake protocol: A modular analysis. *Journal of Cryptology*, 23(2):187–223, April 2010.
29. Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 271–288. Springer, Heidelberg, May / June 2006.