The Whole is Less Than the Sum of Its Parts: Constructing More Efficient Lattice-Based AKEs

Rafael del Pino, Vadim Lyubashevsky, David Pointcheval

The Whole is Less than the Sum of its Parts: Constructing More Efficient Lattice-Based AKEs*

Rafael del Pino[1,2,3], Vadim Lyubashevsky[4], and David Pointcheval[3,2,1]

[1] INRIA, Paris
[2] École Normale Supérieure, Paris
[3] CNRS
[4] IBM Research Zurich

Abstract. Authenticated Key Exchange (AKE) is the backbone of internet security protocols such as TLS and IKE. A recent announcement by standardization bodies calling for a shift to quantum-resilient crypto has resulted in several AKE proposals from the research community. Because AKE can be generically constructed by combining a digital signature scheme with public key encryption (or a KEM), most of these proposals focused on optimizing the known KEMs and left the authentication part to the generic combination with digital signatures.

In this paper, we show that by simultaneously considering the secrecy and authenticity requirements of an AKE, we can construct a scheme that is more secure and with smaller communication complexity than a scheme created by a generic combination of a KEM with a signature scheme. Our improvement uses particular properties of lattice-based encryption and signature schemes and consists of two parts – the first part increases security, whereas the second reduces communication complexity.

We first observe that parameters for lattice-based encryption schemes are always set so as to avoid decryption errors, since many observations by the adversary of such failures usually leads to him recovering the secret key. But since one of the requirements of an AKE is that it be forward-secure, the public key must change every time. The intuition is therefore that one can set the parameters of the scheme so as to not care about decryption errors and everything should still remain secure. We show that this naive solution is not quite correct, but the intuition can be made to work by a small change in the scheme. Our new AKE, which now remains secure in case of decryption errors, fails to create a shared key with probability around 2^{-30}, but adds enough security that we are able to instantiate a KEM based on the NTRU assumption with rings of smaller dimension.

Our second improvement is showing that certain hash-and-sign lattice signatures can be used in "message-recovery" mode. In this mode, the signature size is doubled but this longer signature is enough to recover an even longer message – thus the signature is longer but the message does not need to be sent. This is advantageous when signing relatively long messages, such as the public keys and ciphertexts generated by a lattice-based KEM. We show how this technique reduces the communication complexity of the generic construction of our AKE by around 20%. Using a lattice-based signature in message-recovery mode is quite generic (i.e it does not depend on the structure of the message), and so it may be used in AKE constructions that use a different KEM, or even simply as a way to reduce the transmission length of a message and its digital signature.

* Supported by the European Horizon 2020 ICT Project SAFEcrypto (H2020/2014-2020 Grant Agreement ICT-644729 – SAFECrypto), the French FUI Project FUI AAP 17 – CRYPTOCOMP, and the SNSF ERC Transfer Grant CRETP2-166734 – FELICITY.

1 Introduction

Lattice-based cryptography has matured to the point that it is seen as a viable replacement to number-theoretic cryptography. There are very efficient public-key encryption schemes (and thus Key Encapsulation Mechanisms) based on the NTRU [HPS98, HPS$^+$15] and Ring-LWE problems [LPR10, DXL12, LPR13b, Pei14], as well as digital signature schemes that are also based on NTRU [DDLL13, HPS$^+$14, DLP14] and Ring-LWE [Lyu12, GLP12].

Once we have practical protocols for digital signatures and public key encryption / key encapsulation, it is clear that one can construct an authenticated key exchange (AKE) scheme, and even a forward-secure one which guarantees the key privacy after long-term authentication means are compromised. A rough outline for a simple construction is described in Figure 1, which uses a generic key encapsulation scheme and a digital signature scheme. The simple idea is that Party 1 picks an encapsulation/decapsulation key pair $(\mathcal{K}_e, \mathcal{K}_d)$, sends the encapsulation key in an authenticated way to Party 2, which in turn uses it to encapsulate a random seed k, in an authenticated message. Only Party 1 is then able to decapsulate the seed k derived into a session key sk. Authentication means are the signing keys, and their compromise or the compromise of any future or past encryption/decryption keys does not have any impact on the privacy of the session encrypted under key sk.

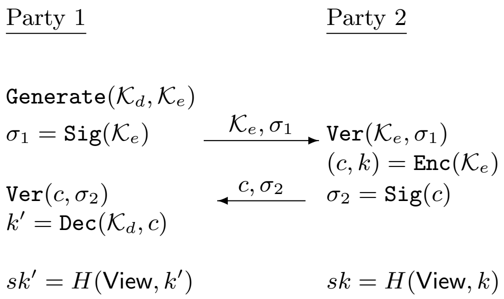

Party 1
Party 2
$\mathtt{Generate}(\mathcal{K}_d, \mathcal{K}_e)$
$\sigma_1 = \mathtt{Sig}(\mathcal{K}_e)$
$\mathcal{K}_e, \sigma_1$
$\mathtt{Ver}(\mathcal{K}_e, \sigma_1)$
$(c, k) = \mathtt{Enc}(\mathcal{K}_e)$
$\mathtt{Ver}(c, \sigma_2)$
c, σ_2
$\sigma_2 = \mathtt{Sig}(c)$
$k' = \mathtt{Dec}(\mathcal{K}_d, c)$
$sk' = H(\mathsf{View}, k')$
$sk = H(\mathsf{View}, k)$

Fig. 1. A Generic AKE Construction from a KEM and a Digital Signature

1.1 Recent Work

There has been a lot of recent work that deals with proposing optimizations of lattice-based KEMs. The works of [DXL12, Pei14, BCNS15, ADPS16] gave constructions (and instantiations) of a KEM derived from the Ring-LWE encryption scheme [LPR13a], while the work of [HPS$^+$15] optimized the parameters for a particular version of the NTRU KEM. All these papers left the authentication part of the AKE to known signature schemes and the generic composition in Figure 1. The work of Zhang et al. [ZZD$^+$15] adapted the (H)MQV [LMQ$^+$03, Kra05] discrete log-based AKE protocol to the Ring-LWE problem. But it seems that this approach leads to schemes that have larger communication complexity (for similar security levels) than the approach in Figure 1. Thus, it currently appears that the most efficient way of constructing lattice-based AKE schemes is a generic composition of a KEM with a digital signature scheme.

1.2 Our Contributions

In our work, we propose two enhancements to the generic AKE construction – allowing decapsulation errors, which increases security, and using digital signatures with message recovery, which decreases the communication.

Handling Decapsulation Errors. The security of lattice-based encryption / encapsulation schemes relies on the hardness of solving linear equations in the presence of noise. The larger the noise is (with respect to the field that we are working over), the harder it is to recover the solution. On the other hand, if the noise is too large, decryption may fail. These decryption failures are not just an inconvenience – their detection usually results in the adversary being able to recover the secret key (c.f. [HNP+03]). For this reason, stand-alone encryption schemes use parameters such that decryption failures occur with only a negligible probability.

In a forward-secure AKE, however, where the encryption keys are ephemeral, there is *intuitively* no danger of decryption failures (which will result in the users not agreeing on a shared key) since the users will restart and a fresh public key will be used in the next key-agreement attempt. The cost of a restart is an increase in the expected run-time and communication complexity of the scheme. For example, if one run of the protocol uses T of some resource and has a failure probability of ϵ, then the expected amount of this resource the complete protocol will require until it completes successfully is $T/(1-\epsilon)$. For values of ϵ that are small, this is very close to T.

A natural idea to construct such an AKE is to take a KEM that may have decapsulation failures and plug it into the prototype in Figure 1. This solution, however, is not necessarily secure. Consider an encapsulation scheme where *invalidly* formed ciphertexts immediately lead to the recovery of the *decapsulated* key[1]. The Adversary's attack would then involve intercepting the ciphertext sent by Party 2 and recovering the key k' that will be the one decapsulated by Party 1 in the event of a decapsulation error (which occurs with non-negligible probability). The Adversary and Party 1 now share a session key. While a KEM in which malformed ciphertexts can be opened by the Adversary to the decapsulated key may appear to be contrived, it does show that the protocol in Figure 1 cannot be proven to be secure when instantiated with a generic scheme with decryption failures.

Our first contribution (Section 5) is a construction of a forward-secure AKE that remains secure even when instantiated with a KEM that leads to decapsulation failures. In particular, we prove that our scheme is secure as long as recovering the *encapsulated* key is a hard problem (regardless of what happens during decapsulation) – so essentially all we need for security is for the KEM to be a one-way function. The modification of the scheme is not particularly complicated – Party 2 simply needs to apply a hash function to k and include it in his message (see the informal description in Figure 2 and the formal one in Figure 3)– but the proof contains several subtleties.

In order to instantiate the AKE, we show that a KEM based on NTRU (Section 3) very naturally fits into the requirements of our generic construction. We give a quick description of it, and leave the full details to Section 3. The decapsulation key is a polynomial $\mathbf{g}$ with small coefficients (chosen according to Table 3) in the ring $\mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$ for $q = 12289$ and $n = 512$ or 1024. The encapsulation key is $\mathbf{h} = \mathbf{f}/\mathbf{g}$, where $\mathbf{f}$ is another polynomial with small coefficients. The encapsulation procedure works by picking two polynomials $\mathbf{r}$ and $\mathbf{e}$ with small coefficients and computing the ciphertext/shared key pair $(2\mathbf{hr}+\mathbf{e}, \mathbf{e} \bmod 2)$. To decapsulate the ciphertext $\mathbf{c} = 2\mathbf{hr} + \mathbf{e}$ using the decapsulation key $\mathbf{g}$, one computes

$$\mathbf{cg} \bmod q \bmod 2/\mathbf{g} = (2\mathbf{fr} + \mathbf{ge} \bmod q) \bmod 2/\mathbf{g} = \mathbf{e} \bmod 2,$$

[1] It is simple to construct such a scheme. Suppose we have an encapsulation scheme (without decapsulation errors) with encapsulation procedure `Enc` and a decapsulation procedure where $\mathtt{Dec}(\mathcal{K}_d, 0) = 0$. We modify it to a scheme where the encapsulation procedure $\mathtt{Enc}'$ runs `Enc` to obtain (c, k) and outputs it with probability $1-\epsilon$. With probability ϵ, it outputs $(0, k)$. Notice that this new scheme is still secure (i.e. one-way) because k is still hard to recover (and actually information-theoretically hard to recover when $(0, k)$ is the output), but with probability ϵ, the decapsulated key is the constant $\mathtt{Dec}(\mathcal{K}_d, 0) = 0$.

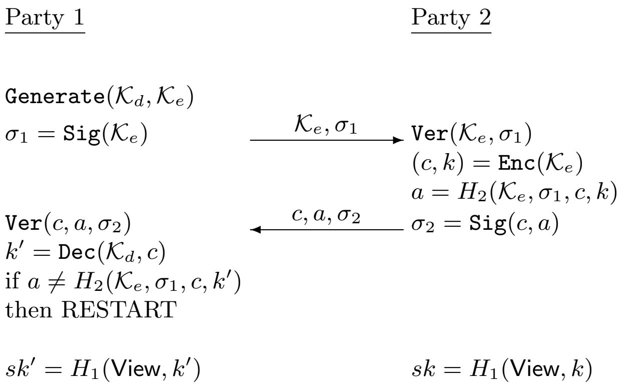

Party 1
Party 2
Generate($\mathcal{K}_d, \mathcal{K}_e$)
$\sigma_1 = \texttt{Sig}(\mathcal{K}_e)$
$\mathcal{K}_e, \sigma_1$
$\texttt{Ver}(\mathcal{K}_e, \sigma_1)$
$(c, k) = \texttt{Enc}(\mathcal{K}_e)$
$a = H_2(\mathcal{K}_e, \sigma_1, c, k)$
$\sigma_2 = \texttt{Sig}(c, a)$
c, a, σ_2
$\texttt{Ver}(c, a, \sigma_2)$
$k' = \texttt{Dec}(\mathcal{K}_d, c)$
if $a \neq H_2(\mathcal{K}_e, \sigma_1, c, k')$
then RESTART
$sk' = H_1(\mathsf{View}, k')$
$sk = H_1(\mathsf{View}, k)$

Fig. 2. Informal AKE Construction from Encapsulation and Digital Signatures with Decapsulation Errors

which is the shared key. Note that for the above equality to hold, it is crucial that $2\mathbf{fr} + \mathbf{ge} \bmod q \bmod 2 = 2\mathbf{fr} + \mathbf{ge} \bmod 2$, which happens exactly when the coefficients of $\mathbf{f}, \mathbf{r}, \mathbf{e}, \mathbf{g}$ are small enough that a reduction modulo q does not take place. If a reduction does take place, then we will end up with a decapsulation error.

Because in our construction decapsulation errors are no longer a security risk, we can set the parameters such that these failures occur a non-negligible number of times – for example with probability 2^{-30}. If a failure does occur, then the protocol can be safely restarted. We believe that 2^{-30} is a low-enough failure probability that some external, for example, networking error may have a higher probability of occurring. Table 3 shows the security gained when we instantiate our scheme such that it has decapsulation error of 2^{-30} vs. 2^{-128}. We discuss the security of our proposals later in this section.

Signatures in Message Recovery Mode. Just as for signatures based on standard number-theoretic assumptions, lattice-based signatures come in two flavors – Fiat-Shamir and hash-and-sign.[2] The improvement we present in this paper is only for hash-and-sign signature schemes – in particular for the specific parameters of the scheme presented in [DLP14]. We now give a brief description of that scheme, which combines the pre-image sampling algorithm from [GPV08] with a particular instantiation of an NTRU lattice [HHGP$^+$03].

The public key is a polynomial $\mathbf{h} \in \mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$ which is equal to $\mathbf{f}/\mathbf{g}$, where the coefficients of $\mathbf{f}$ and $\mathbf{g}$ are somewhat small.[3] The secret key, which is a basis of a particular lattice induced by $\mathbf{h}$, allows the signer to find polynomials $\mathbf{s}_1, \mathbf{s}_2$ with small coefficients such that $\mathbf{hs}_1 + \mathbf{s}_2 = H(m)$, where m is the message and H is a hash function modeled as a random oracle that maps $\{0,1\}^*$ to random elements in $\mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$.

The signature of a message m is $(\mathbf{s}_1, \mathbf{s}_2)$, and the verification procedure checks that $\mathbf{s}_1, \mathbf{s}_2$ have small coefficients and that $\mathbf{hs}_1 + \mathbf{s}_2 = H(m)$. Note that because $\mathbf{s}_2$ is completely determined by $\mathbf{s}_1$ and m, there is no reason to send it. Thus the signature can be just $\mathbf{s}_1, m$. For ease of exposition of our improvement, assume that the bit-length of m is a little less than the bit-length of elements in $\mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$ (e.g. $n \log q - 256$ bits). Then rather than sending $\mathbf{s}_1, m$ as the signature, we will send $\mathbf{s}'_1, \mathbf{s}'_2$ that satisfy the equation $\mathbf{hs}'_1 + \mathbf{s}'_2 = \mathbf{t}$ where $\mathbf{t}$ is the polynomial in $\mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$ whose first coefficients are $m + F(H'(m))$ and its last $256/\lfloor \log q \rfloor$ coefficients are $H'(m)$. Here H' is a random oracle mapping $\{0,1\}^*$ to $\{0,1\}^{256}$ (thus its output fits into

[2] There are also lattice signature schemes that do not use random oracles, but those are much less practical.

[3] The distribution of $\mathbf{f}$ and $\mathbf{g}$ is different from the way the secret key is constructed for the KEM. In particular, we *do not* want $\mathbf{f}$ and $\mathbf{g}$ to be too small. Full details are provided in [DLP14].

	One-way authenticated KE		Two-way authenticated KE	
Dimension n	512	1024	512	1024
Modulus	12289	12289	12289	12289
First flow size (bits)	≈ 7200	≈ 14400	≈ 9800	≈ 19300
Second flow size (bits)	≈ 10300	≈ 19600	≈ 10300	≈ 19600
Signing key size (bits)	≈ 2100	≈ 3700	≈ 2100	≈ 3700
Verification key size (bits)	7168	14336	7168	14336

Table 1. Parameter sizes and communication length for our AKE. We consider the versions where both parties authenticate themselves and the version in which only the server is authenticated.

$256/\lfloor \log q \rfloor$ coefficients) and F is another random oracle whose output range is $n \log q - 256$ bits. Notice that if we send $\mathbf{s}_1'$ and $\mathbf{s}_2'$ as the signature, the message m can be recovered by first computing $\mathbf{t} = \mathbf{a}\mathbf{s}_1' + \mathbf{s}_2'$. Then from $\mathbf{t}$, we can recover $H'(m)$, then $F(H'(m))$, and finally m. More details (in particular how one would handle messages that are longer than $n \log q$ bits) and a full security proof are provided in Section 4.2.

The main advantage of the message recovery signature scheme is that instead of sending the message m, one can send the shorter element $\mathbf{s}_2$. Note that if the message we are signing is short, then our technique of sending $\mathbf{s}_2$ instead of a message is counterproductive and should not be used.[4] The public key and the ciphertext of a KEM, however, are polynomials that are pseudorandom over $\mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$, and so require $n \log q$ bits to represent, and their signatures would benefit from the message-recovery technique. The efficacy of the message recovery technique clearly does not depend on anything except the message size, and so it may also be appropriate to use in combination with other KEMs. In Table 2, we illustrate the savings of this technique when working over the ring $\mathbb{Z}_q[\mathbf{x}]/\langle x^{1024} + 1 \rangle$. When combining our signature scheme with our KEM, or with the Ring-LWE based KEM in [ADPS16], the savings are about 20%. Note that our complete scheme has less total communication complexity due to the fact that our NTRU KEM is a little bit more compact than the Ring-LWE one.

1.3 Putting Everything Together

Table 1 shows the communication complexity of our full AKE when instantiated with $n = 512$ and $n = 1024$ (the security of these choices will be discussed later) for the case of two-sided authentication and for the case of when only the second party needs to be authenticated (as is often the case in TLS). In addition, Section 6 describes a modification of our protocol in which the identities of the parties are hidden from a passive adversary (this is sometimes a desirable property and is an option in the IKE protocol). This anonymity property is impossible to achieve in a 2-round scheme,[5] and so a third round is required. Additionally, since maintaining anonymity requires the splitting of the key/signature pair usually sent in the first round, we

[4] We point out that this is in contrast to using message-recovery mode in other hash-and-sign signatures, such as RSA. In those cases, the signature size does not increase in message-recovery mode, and so this mode is always advantageous to use.

[5] The intuition is that the player who moves first has to send his signed message in the clear because there is no encryption key (public or private) available to him at the start of the protocol. Therefore a passive adversary can simply perform a verification procedure with that player's public verification key to see if he is indeed the sender.

	New Hope [ADPS16]		Our Scheme	
Hash-and-Sign	naive	message-recovery	naive	message-recovery
First flow size (bits)	≈ 24000	≈ 19600	≈ 23300	≈ 18900
Second flow size (bits)	≈ 25800	≈ 21400	≈ 23600	≈ 19200
Total communication (bits)	≈ 49800	≈ 41000	≈ 46900	≈ 38100

Table 2. Comparison of our paper with [ADPS16]. For the comparison to be meaningful we consider the AKE obtained by adding a Hash-and-Sign signature (either without or with message-recovery) to the scheme in [ADPS16]. We illustrate the savings of message-recovery mode by presenting the naive generic AKE construction and one that uses the digital signature in message-recovery mode.

cannot use the message-recovery technique there (but it can still be used when signing the ciphertext in the second flow), thus the total communication complexity will be somewhat larger than in our 2-round scheme.

The comparison of our AKE with $n = 1024$ to the one in [ADPS16] is given in Table 2. Because [ADPS16] only proposed a KEM, we combine it with the digital signature scheme that we use for our KEM in this paper. One can see that our AKE is slightly shorter than the one from [ADPS16] for essentially the same security level. Also, the message-recovery technique reduces the communication lengths of both AKEs by the same amount.

1.4 Computational Efficiency

We will now discuss the efficiency of our AKE. The KEM part of our scheme requires generation of small polynomials and arithmetic operations in the ring $\mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$. Rather than generating polynomials with each coefficient being independently chosen from some distribution, we follow the original NTRU way of generating such polynomials by prescribing exactly how many of each coefficient the polynomial should have. Such polynomials can be created using n random swaps within an integer array. We remark that while such an algorithm would be weak to timing attacks the permutation can be done in constant time using a sorting network, as in e.g. [BCLvV16], this incurs a small overhead resulting in a complexity of $O(n \log^2 n)$. Addition and subtraction similarly requires $O(n)$ integer operations. Multiplication and division can be done in quasi-linear time by employing the Number Theoretic Transform (i.e. FFT over a finite field) which has very efficient implementations (e.g. [LN16]). The prime $q = 12289$ was chosen such that $x^n + 1$ (for $n = 512$ and $n = 1024$) splits into n linear terms and is therefore amenable to the number theory transform. We point out that this is the same ring that was used in [DDLL13, ADPS16], and those works produced schemes that were faster than number-theoretic schemes of comparable security parameters. The KEM part of our scheme is therefore very efficient.

Lattice-based signatures that use the hash-and sign approach use a technique known as lattice pre-image sampling [GPV08, Pei10, MP12] and this generally results in digital signatures that are longer and much less efficient to compute than those generated using the Fiat-Shamir approach [Lyu12, GLP12, DDLL13]. But there has been a lot of recent work on trying to optimally (and securely) instantiate hash-and-sign signatures over polynomial rings. In [DLP14], it was shown that hash-and-sign signatures over carefully-constructed NTRU lattices that are very similar to those in [HHGP$^+$03] may be instantiated in a way such that they have signature sizes that are somewhat smaller than the most compact Fiat-Shamir NTRU-based signature [DDLL13]. Then it was shown in [LP15] that for many polynomial rings, the GPV sampling algorithm [GPV08] that is used to produce the compact signatures in [DLP14], can actually be run in time $O(n^2)$ (and in $O(n)$ space) rather than $O(n^3)$ time required for general lattices. And very recently, it was further shown that pre-image sampling can be done in quasi-linear time over rings $\mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$ using ideas from the FFT procedure [DP15].

Asymptotically, therefore, hash-and-sign signatures are as efficient as the Fiat-Shamir ones. The caveat is that the lattice pre-image sampling requires the intermediate storage of a vector of a high precision approximations to real numbers.[6] This requires roughly 300 - 700K bits of storage and so may not be a suitable option for constrained devices. One should therefore consider the situation in which the AKE is used before deciding whether using hash-and-sign signatures (and thus benefiting from their message-recovery mode) is appropriate. The most common scenario in which it would be appropriate is TLS where only the server (which is usually not a device with strong computational constraints) is being authenticated. On the other hand, if mutual authentication is required and one of the devices has resource limitations, then the hash-and-sign approach may not be appropriate and one may choose to forego the savings in the communication complexity.

1.5 Security

Obtaining the *exact* computational hardness of lattice-based schemes is an extremely difficult problem. In order to put our work into context and obtain an “apples-to-apples” comparison, we will use some security estimates from the recent work in [ADPS16]. The most efficient attacks against the KEM and signature scheme which comprise lattice-based AKEs are lattice reduction attacks.[7] The lattice attacks fall into two categories – sieving and enumeration (we discuss these in more detail in Section 3.2).

The attacks based on sieving are asymptotically more efficient, but have a very big downside in that the space complexity is essentially equivalent to the time complexity. Moreover, all known approaches to sieving have lower bounds to the required space complexity, and it would be a huge breakthrough if an algorithm were discovered that required less space. The attacks based on enumeration are less efficient time-wise, but do not require a lot of space, and thus are the attacks that are preferred in practice. We analyze our schemes with respect to both attacks following the methodology in [ADPS16].[8] All the schemes that we propose have complexity against enumeration attacks (with quantum speedup) larger than 128 bits (see Tables 3 and 4). Furthermore, all our schemes, with the exception of the digital signature component of the AKE for $n = 512$ have security larger than 128 bits against sieving attacks as well. While the signature scheme for $n = 512$ appears to have less than 128 bits of security, we point out that the sieving complexity we state follows that from [ADPS16], which uses the asymptotic complexity while leaving out the lower order terms. Those lower-order terms are in fact quite significant in practice, and put the security of the signature scheme above 128-bits. The analysis in [ADPS16] was purposefully done to be extremely optimistic in favor of the attacker, and for

[6] It was shown in [LW15] that one can do pre-image sampling without high-precision arithmetic, but the resulting vector (and thus the signature size) ends up being larger than when using sampling procedures such as [GPV08].

[7] There are also combinatorial attacks (e.g. [How07]), but the dimensions considered in this paper are too high for them to be effective.

[8] The paper [ADPS16] also discussed a “distinguishing” attack, but such an attack does not seem to be relevant in our case because the security in our AKE is based on the 1-wayness of the KEM – thus on a search, rather than a decision, problem.

that reason, that paper only proposes parameters for $n = 1024$. But they admit that there is no currently-known attack that exceeds 2^{128} for $n = 512$.

There have recently been recent attacks on NTRU encryption schemes where the modulus is much larger than the size of the secret polynomials $\mathbf{f}, \mathbf{g}$ [ABD16, CJL16]. But those attacks do not apply to schemes such as ours, where the size of these polynomials is not too far from the range in which their quotient will be uniformly-random in the ring [SS11].

1.6 Our Recommendations for Lattice-Based AKE

We presented two approaches for improving the generic construction of an AKE scheme from lattice assumptions. The first approach introduces a very small chance of protocol failure, but increases security. The second approach reduces the communication size of the flows in the AKE assuming that the public key and ciphertexts of the KEM are "large enough".

If one were to make a recommendation for parameter sizes to be used today, one should probably err on the side of caution and recommend that one use $n = 1024$. In this sense, we agree with the decision to only propose parameters for $n = 1024$ in the Ring-LWE based KEM of [ADPS16]. On the other hand, there are currently no attacks that make our scheme with $n = 512$ less than 128-bit secure. Nor are there really algorithmic directions that look promising enough to make a significant impact on this parameter range. Thus the main reason for recommending $n = 1024$ is to guard against completely surprising novel attacks – in other words, to guard against "unknown unknowns." But we believe that if cryptanalysis over the next several years intensifies and still does not reveal any novel attack ideas, it is definitely worth re-examining the possibility that the parameters for $n = 512$ are indeed secure enough.

As in [ADPS16], when we set $n = 1024$, the security against even the most optimistic attacks is way above the 128-bit threshold. In this case, we see little sense for using KEM parameters that increase security at the expense of having a 2^{-30} chance of decapsulation errors (i.e. those in column III of Table 3). Thus if one were to set $n = 1024$, then we recommend using parameters in column IV of Table 3 along with those in column II of Table 4 for the signature. And one should use the signature in message-recovery mode. The communication size of the resulting AKE is in Table 2.

If one were to use $n = 512$, then one could use the KEM parameters in column I of Table 3. In this case, an increase in 13 bits (compare to column II) of the security against sieve attacks may be worthwhile. One could then also use the parameters from column I of Table 4 for the signature. Note that the complexity against sieving attacks is not quite 128 bits, but we remind the reader that this does not take into account the lower order terms of the sieving attack, which are significant. Also, such an attack requires over 2^{85} space, which makes it impossible to mount. Because the enumeration complexity is still higher than 128 bits, and authentication is not as critical as secrecy,[9] we believe that using the parameters in column I gives a good trade-off between security and efficiency in all but the most sensitive applications.

As mentioned in Section 1.4, the hash-and-sign signatures that we propose may not be suitable if the device doing the signing is extrememly limited in computational resources. In this case, we would recommend combining our KEM with a Fiat-Shamir signature such as BLISS (perhaps adapted to $n = 1024$) [DDLL13]. And if one does not want to perform any high precision arithmetic or Gaussian sampling, then one can combine our KEM with the Fiat-Shamir scheme in [GLP12] which only requires sampling from the uniform distribution. Also, if one uses the AKE in a way that is not completely forward-secure, i.e. the KEM public key does not change with each interaction, then our security reduction in case of decapsulation errors no

[9] If an attack on the signature scheme were discovered, the scheme could be changed. Whereas an attack on the KEM would reveal all previous secret communication.

longer holds. In this case, one should use our KEM with the parameters in columns II and IV of Table 3.

We also mention that the modulus of the rings we use in the paper – 12289 – was chosen for efficiency purposes. It is the smallest integer q such that the polynomial $\mathbf{x}^n + 1$, for $n = 512$ or 1024, can be factored into linear factors with integer coefficients modulo q. Such a factorization, combined with the fact that n is a power of 2, allows one to perform multiplication and division operations in quasi-linear time using the Number Theory Transform. If one did not care about optimizing the efficiency of the AKE, then we could have chosen a smaller q and then performed multiplication using other techniques (e.g. Karatsuba's multiplication). The advantage of choosing a smaller modulus is that the ciphertext, which is of length $\approx n \log q$ would be shorter. Of course, if we use our schemes with a different modulus, we would have to change the distribution of all the variables in order to maintain similar security and decapsulation error. So while one cannot choose this modulus to be too small, it seems that something on the order of 2^{11} would be possible. Compared to using the current modulus of 12289, this could result in a savings of approximately $3n$ bits in the length of the KEM public key and ciphertext. Whether this is something worth doing would depend on the scenario in which the AKE is employed. The main reason that we only use 12289 in this paper is to obtain a fair comparison to [ADPS16], whose scheme also used this modulus.

1.7 Paper Organization

In Section 2, we state the formal security definitions of a digital signature, a KEM, and an AKE. In Section 3, we describe our KEM and discuss its security for the concrete parameters in Table 3. In Section 4, we present the construction and security proof for using a lattice-based hash-and-sign signature in message-recovery mode. In Section 5, we then present a generic construction of an AKE scheme from a KEM and a digital signature scheme, where the KEM may result in decapsulation errors. In Section 6, we also discuss a protocol in which the identity of the participants is hidden from a passive eavesdropper.

1.8 Acknowledgements

We thank Léo Ducas for very helpful discussions related to lattice reduction algorithms and to [ADPS16].

2 Cryptographic Preliminaries

In this section, we describe the formalisms and security models of authenticated key exchange and its underlying building blocks.

2.1 Digital Signatures

A *digital signature scheme* Σ consists of 3 (randomized) algorithms – `SigKeyGen`, `Sig`, and `Ver`. The key generation algorithm, `SigKeyGen`, takes as input a string whose length is the security parameter, and outputs a secret signing key $\mathcal{K}_s$ and a public verification key $\mathcal{K}_v$. The signing algorithm `Sig` takes as input a signing key $\mathcal{K}_s$ and a message m, and outputs a signature σ. In order to get a unified notation for both "signature with appendix", where the signature is appended to the unmodified message, and "signature with message-recovery", where the message is embedded in the signature itself, we assume that the signature σ contains (either explicitly or implicitly) the message. The verification algorithm `Ver` takes the verification key $\mathcal{K}_v$ and a signature σ, and outputs either a message m or $\perp$ (for failure, or invalid signature).

The *correctness* of the digital signature scheme is satisfied if for all $(\mathcal{K}_s, \mathcal{K}_v)$ pairs output by the key generation algorithm, and all messages m in the message space, we have $\texttt{Ver}(\mathcal{K}_v, \texttt{Sig}(\mathcal{K}_s, m)) = m$. On the other hand, the security of a digital signature scheme is usually defined via the standard *existential unforgeability against adaptive chosen message attacks* [GMR88]: Upon getting the public verification key $\mathcal{K}_v$, the adversary is allowed to query a signing oracle and then attempts to produce a signature of a new message.

In our generic AKE construction, a slightly stronger security notion will be required, the so-called *strong unforgeability* (a.k.a. non-malleability) against adaptive chosen message attacks [SPMLS02, ADR02]: Upon getting the public verification key $\mathcal{K}_v$, the adversary is allowed to query a signing oracle and then attempts to produce a new signature (possibly on an already signed message).

The quality of an adversary $\mathcal{A}$ is measured by its success probability, denoted $\mathsf{Succ}_{\Sigma}^{\mathsf{suf-cma}}(\mathcal{A})$, in forging a new valid signature, and we denote by $\mathsf{Succ}_{\Sigma}^{\mathsf{suf-cma}}(t)$ the best success probability an adversary can get within time t.

2.2 Key Encapsulation Mechanism (KEM)

A *key encapsulation mechanism* $\mathcal{KEM}$ [Sho01] is defined via 3 (randomized) algorithms – $\texttt{KEMKeyGen}$, $\texttt{Enc}$, and $\texttt{Dec}$. The key generation algorithm, $\texttt{KEMKeyGen}$, takes as input a string whose length is the security parameter, and outputs a secret decapsulation key $\mathcal{K}_d$ and a public encapsulation key $\mathcal{K}_e$. The encapsulation algorithm $\texttt{Enc}$ takes as input the public key $\mathcal{K}_e$ and outputs an ordered pair (c, k), where c is a ciphertext that encapsulates the key k. The decapsulation algorithm $\texttt{Dec}$ takes as input the secret key $\mathcal{K}_d$ and c and outputs a key k'.

The key encapsulation algorithm is *correct* if for all $(\mathcal{K}_d, \mathcal{K}_e)$ key pairs produced by $\texttt{KEMKeyGen}$ and (c, k) produced by $\texttt{Enc}(\mathcal{K}_e)$, we have $\texttt{Dec}(\mathcal{K}_d, c) = k$. But this may not always be the case: we then say that $\mathcal{KEM}$ is ε-correct (or is an ε-KEM) if

$$\Pr_{\mathcal{K}_d, \mathcal{K}_e, c, k}[\texttt{Dec}(\mathcal{K}_d, c) = k] \geq \varepsilon,$$

where all the input variables follow the distribution probability $(\mathcal{K}_d, \mathcal{K}_e) \leftarrow \texttt{KEMKeyGen}(1^\lambda)$ and $(c, k) \leftarrow \texttt{Enc}(\mathcal{K}_e)$. We will say that such a key encapsulation c, that can be decapsulated to the expected key k, is *valid*, or by abuse of notation, the pair (c, k) will be said to be *valid*.

On the other hand, the security of a key encapsulation mechanism is usually defined via the classical *indistinguishability* of the encapsulated key k: Upon getting a pair (c, k) where k is either the actual key encapsulated in c or a random key, independent from the key encapsulated in c, the adversary attempts to guess whether this is the actual or a random key.

However, in our application, a much weaker security notion will be enough, the so-called *one-wayness* property: Upon getting the encapsulation c of the key k, the adversary has to output k. The quality of an adversary $\mathcal{A}$ is measured by its success probability in outputting k:

$$\mathsf{Succ}_{\mathcal{KEM}}^{\mathsf{ow}}(\mathcal{A}) = \Pr[\mathcal{A}(\mathcal{K}_e, c) = k | (\mathcal{K}_d, \mathcal{K}_e) \leftarrow \texttt{KEMKeyGen}(1^\lambda), (c, k) \leftarrow \texttt{Enc}(\mathcal{K}_e)].$$

As above, we denote by $\mathsf{Succ}_{\mathcal{KEM}}^{\mathsf{ow}}(t)$ the best success probability an adversary can get within time t.

We stress that in the above definition with possible invalid ciphertexts (a.k.a. decryption failures), we only consider the hardness of recovering the encapsulated key, and do not say anything about the decapsulated key (if they are not the same), as exploited in the counter-example in the previous section.

An additional property is the *checkability* of the encapsulated key, when for $(\mathcal{K}_e, c)$ and a candidate k, it is easy to check whether k is the actual encapsulated key. This is a classical result that from a one-way KEM, one gets an indistinguishable KEM by simply using $H(k)$ as the new encapsulted key, when H is modeled as a random oracle [BR93]. Checkability allows for a tight reduction between the two security notions in the random oracle model.

2.3 Authenticated Key Exchange (AKE)

An *authenticated key exchange* $\mathcal{AKE}$ [BR94,BCK98] is an interactive protocol between 2 entities that hold authentication means (a common secret key, a common password, or private keys for a public-key digital signature scheme) and that eventually agree on a common random ephemeral session key sk.

The *correctness* of an authenticated key exchange protocol is satisfied if when the two players interact with their intended partners (the players with whom they want to agree on a session key), they end up terminating the protocol in possession of the same session key. But as in the above KEM, this may not always be the case: we then say that $\mathcal{AKE}$ is ε-correct (or is an ε-AKE) if honest players agree on the same session key with probability greater than ε. In case of failure, we run the protocol again: it just needs not to fail too often, and perhaps even a 10% failure rate would not be a major issue in practice.

On the other hand, the security of an authenticated key exchange protocol means that no adversary should be able to distinguish the actual session key agreed on by an honest player from a random bitstring. Let us thus define more formally the security game. An honest player P has several instances P^i that can run concurrent executions of the protocol and the adversary has access to the following oracles to interact with the various instances of the different honest players:

- $\mathsf{Send}(P^i, m)$ makes the message m to be sent to the i^{th}-instance P^i of the player P, and the output is the message that P generates from such a message m. A special message "Start" is used to ask the instance P^i to initiate a new execution of the protocol. This models the fact that the adversary controls the network and any communication routed through it. In particular, he can forward, replay, block, or alter the messages;
- $\mathsf{Reveal}(P^i)$, if P^i has terminated and holds a session key, this key is output. This models the later misuse of the session key which might be leaked if used with a weak cryptographic system;
- $\mathsf{Test}(P^i)$, if P^i is *fresh*, one flips a bit b: if $b = 1$, the session key sk held by P^i is sent back, otherwise a truly random key is sent. However, if P^i is not fresh, the actual value of sk (possibly $\perp$, if not yet defined) is returned. This query can be asked once only, and it characterizes the randomness of *fresh* session keys.

The adversary eventually has to guess the bit b. We stress that we are using the find-then-guess flavor of indistinguishability [BDJR97] only, where a unique session key can be tested. But it is well-known that using hybrids, this security notion implies the real-or-random definition, where the adversary could test many session keys, which thus guarantees randomness and independence of session keys that are fresh. The restriction to fresh keys, or fresh instances, is to avoid trivial attacks: an instance P^i is *fresh* if

- P^i holds a key;
- neither P^i nor its partner have been asked a Reveal-query.

The latter restriction thus excludes situations where the adversary has already asked for that key and therefore already knows it. For this case, the notion of *partnership* is also important, and is defined by matching sessions: two instances are partners if their views of the protocol are the same.

Basically, two partners should terminate with the same session key with reasonable probability (ε-correctness) or abort. On the other hand, for all the fresh instances/keys, the adversary should not be able to distinguish the actual session keys from random bitstrings. Hence the adversary $\mathcal{A}$ outputs its guess b' for the bit b involved in the Test-query. The quality of the adversary $\mathcal{A}$ is measured by its advantage in guessing b: $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{AKE}}(\mathcal{A}) = |\ \Pr[b' = 1|b = 1] - \Pr[b' = 1|b = 0]\ |$. As above, we denote by $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{AKE}}(t)$ the best advantage an adversary can get within time t.

2.4 Forward-Secure AKE

An enhanced security notion for AKE and ε-AKE is *forward secrecy*, which means that the above indistinguishability of the actual session key for a random bitstring should hold for previous sessions, even after the corruption of the long-term authentication means. In the security game, the adversary has access to an additional oracle:

- $\mathsf{Corrupt}(P)$ makes the long-term authentication means of the player P to be given to the adversary.

Then, the *forward-secure freshness* property is slightly different: an instance P^i is *fs-fresh* if

- P^i holds a key;
- none of P^i nor its partner have been asked for a Reveal-query;
- none of P^i nor its partner have been asked for a $\mathsf{Corrupt}$-query before terminating the AKE protocol execution.

The quality of the adversary $\mathcal{A}$ is then measured by the advantage $\mathsf{Adv}^{\mathsf{fs-ind}}_{\mathcal{AKE}}(\mathcal{A}) = |\ \Pr[b' = 1|b = 1] - \Pr[b' = 1|b = 0]\ |$. As above, we denote by $\mathsf{Adv}^{\mathsf{fs-ind}}_{\mathcal{AKE}}(t)$ the best advantage an adversary can get against the forward-secure indistinguishability security game within time t.

2.5 Identity Hiding AKE

In the context of the "post-specified peer" model [CK02], in which the two parties do not know the identity of their partner before exchanging keys but learn it during the protocol, it can be natural to require *identity protection* from the key exchange protocol. This is an identity hiding notion, where parties should only learn each other's identity if they are interacting with their intended partner. The anonymity of the key exchange can be defined against either passive or active attackers and requires that an adversary should not be able, after either observing an exchange or interacting with an honest player, to distinguish the identity of an honest player in the protocol from any other identity. We specify the security game in a more formal way in Section 6.

3 KEM from NTRU

In this section, we instantiate a KEM based on the hardness of the NTRU problem.

The NTRU problem deals with finding short solutions to polynomial equations over certain rings. Some of the more "popular" rings to use are $\mathbb{Z}[\mathbf{x}]/\langle \mathbf{x}^n - 1\rangle$ where n is a prime integer and $\mathbb{Z}[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$ where n is a power of 2. Starting from the seminal work of [HPS98] where the NTRU problem was first defined, there have been many different flavors of the problem mostly differing on the underlying distributions from which the keys and randomness are generated.

Elements in $\mathbb{Z}[\mathbf{x}]/\langle \mathbf{x}^n \pm 1\rangle$ are represented as polynomials of degree at most $n - 1$ and reduction modulo an odd q maps the coefficients into the range $[-(q-1)/2, (q-1)/2]$. We also define the norm of an element in $\mathbb{Z}[\mathbf{x}]/\langle \mathbf{x}^n \pm 1\rangle$ to simply be the ℓ_2-norm of the vector formed by its coefficients. In all variants of the NTRU problem, there are some subsets D_e, D_f of $\mathbb{Z}[\mathbf{x}]/\langle \mathbf{x}^n \pm 1\rangle$ that consist of polynomials with coefficients of small norms. Furthermore, the polynomials from the set D_f are also invertible in both $\mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n \pm 1\rangle$ and $\mathbb{Z}_2[\mathbf{x}]/\langle \mathbf{x}^n \pm 1\rangle$.

The NTRU trap-door function intuitively rests on two assumptions. The first is that when one is given a polynomial $\mathbf{h} = \mathbf{f}/\mathbf{g} \bmod q$ where $\mathbf{f}, \mathbf{g} \leftarrow D_f$, it is hard to recover $\mathbf{f}$ and $\mathbf{g}$. The second assumption is that when one is given an $\mathbf{h}$ generated as above and $\mathbf{t} = 2\mathbf{hr} + \mathbf{e} \bmod q$ where $\mathbf{r}, \mathbf{e} \leftarrow D_e$, it is hard to recover $\mathbf{e} \bmod 2$.[10] When one has the trap-door $\mathbf{g}$, however, one can recover $\mathbf{e}$ by first computing $\mathbf{gt} \bmod q = 2\mathbf{fr} + \mathbf{ge}$. If the modulus q is large enough (i.e. $2\mathbf{fr} + \mathbf{ge}$ in $\mathbb{Z}[\mathbf{x}]/\langle \mathbf{x}^n \pm 1\rangle$ is equal to $2\mathbf{fr} + \mathbf{ge}$ in $\mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n \pm 1\rangle$), then the preceding is equal to $\mathbf{ge}$ in the ring $\mathbb{Z}_2[\mathbf{x}]/\langle \mathbf{x}^n \pm 1\rangle$. Since $\mathbf{g}$ has an inverse in $\mathbb{Z}_2[\mathbf{x}]/\langle \mathbf{x}^n \pm 1\rangle$, one can then divide by $\mathbf{g}$ to recover $\mathbf{e} \bmod 2$.

Definition 1. *For some ring $R = \mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n \pm 1\rangle$ and subsets of the ring D_f and D_e, generate polynomials $\mathbf{f}, \mathbf{g} \leftarrow D_f$ and $\mathbf{e}, \mathbf{r} \leftarrow D_e$. Define $\mathbf{h} = \mathbf{f}/\mathbf{g} \bmod q$ and $\mathbf{t} = 2\mathbf{hr} + \mathbf{e} \bmod q$. The NTRU$(R, D_f, D_e)$ problem asks to find $\mathbf{e} \bmod 2$ when given $\mathbf{h}$ and $\mathbf{t}$.*

The distributions of D_f and D_e will depend on the security and failure probability of our scheme. We direct the reader to Table 3 and Section 3.1. We now present a simple KEM whose one-wayness (see Section 2.2) is directly based on the NTRU(R, D_f, D_e) problem in Definition 1, and whose correctness is based on the discussion preceding the definition.

```
KEMKeyGen{  $\mathbf{f}, \mathbf{g} \xleftarrow{\$} D_f, \mathbf{h} \leftarrow \mathbf{f}/\mathbf{g} \bmod q$
            Return $(\mathcal{K}_d, \mathcal{K}_e) = (\mathbf{g}, \mathbf{h})$            }

   Enc($\mathbf{h}$){  $\mathbf{r}, \mathbf{e} \xleftarrow{\$} D_e, \mathbf{t} \leftarrow 2\mathbf{hr} + \mathbf{e} \bmod q$
            Return $(c, k) = (\mathbf{t}, \mathbf{e} \bmod 2)$            }

  Dec($\mathbf{g}, \mathbf{t}$){
            Return $k = \frac{\mathbf{gt} \bmod q \bmod 2}{\mathbf{g}} \bmod 2$            }
```

3.1 KEM Parameters

Table 3 contains our proposed parameter choices for the KEM. To explain the table, we will use the first column as a running example. The polynomial ring considered in this instantiation is $\mathbb{Z}_{12289}[\mathbf{x}]/\langle \mathbf{x}^{512} + 1\rangle$ and the secret key and randomness parameters $\mathbf{f}, \mathbf{g}, \mathbf{e}$, and $\mathbf{r}$ are chosen as random permutations of degree 512 polynomials that have 1 coefficient set to ±12 (each), 1 to ±11, 3 to ±10, etc. and 48 set to 0. The norm of the secret key and error elements $(\mathbf{f}, \mathbf{g})$ and $(\mathbf{r}, \mathbf{e})$ in this instance is approximately 93.21.

Note that for security, one should use a distribution that produces the largest vectors possible while not resulting in too many decryption failures. The most appropriate such distribution is the normal distribution (either the discrete normal or a rounded continuous).[11] Such an operation,

[10] This is somewhat different from the standard NTRU assumption in that we are going to allow the coefficients of $\mathbf{e}$ to be larger than 2, but only require $\mathbf{e} \bmod 2$ to be recovered. This is actually more related to an NTRU encryption scheme that was first introduced in [SS11] where the message was hidden in the lower order bits of the error. One could then think of our KEM as an encryption of a random message. But since the message itself is random, its randomness contributes to the noise making it larger.

[11] The paper of [ADPS16] proposes to use the binomial distribution, which is a good approximation of the normal distribution and is not too difficult to generate. It should be pointed out that the distribution does not really affect the security of the scheme – of main importance is the norm of the generated vectors.

	I	II	III	IV
Polynomial	$\mathbf{x}^{512}+1$	$\mathbf{x}^{512}+1$	$\mathbf{x}^{1024}+1$	$\mathbf{x}^{1024}+1$
Modulus	12289	12289	12289	12289
±12 coeff.	1	0	0	0
±11 coeff.	1	0	1	0
±10 coeff.	3	0	2	0
±9 coeff.	5	1	4	0
±8 coeff.	8	2	8	1
±7 coeff.	12	4	15	3
±6 coeff.	17	9	26	9
±5 coeff.	24	17	42	22
±4 coeff.	31	28	61	46
±3 coeff.	38	41	81	80
±2 coeff.	44	55	100	118
±1 coeff.	48	65	113	150
0 coeff.	48	68	118	166
σ	4.151	2.991	3.467	2.510
sk norm	≈ 93.21	≈ 67.17	≈ 110.42	≈ 79.54
bits (pk and ciphertext)	7168	7168	14336	14336
bits sk	2560	2560	5120	5120
failure prob.	$\approx 2^{-30}$	$\approx 2^{-128}$	$\approx 2^{-30}$	$\approx 2^{-128}$
block size	487	438	1026	939
sieving complexity (log #operations)	> 128	> 115	> 269	> 246
sieving space (log bits)	> 114	> 104	> 227	> 209
enumeration complexity (log #operations)	> 185	> 157	> 645	> 503

Table 3. Parameters for the NTRU KEM

though, might be somewhat more costly than simply creating a random permutation of a fixed vector. We thus fix the coefficients of our polynomials to be as close to a discrete Gaussian as possible. Concretely the number of coefficients set to an integer k is the probability that a discrete Gaussian of parameter σ outputs k multiplied by the degree of the polynomial, e.g. $512 \cdot \Pr[\mathcal{D}_{4.151} = 12] \approx 1$ so we fix the number of coefficient with value 12 to one in our first parameter set. Public keys and ciphertexts are polynomials in $\mathbb{Z}_{12289}[\mathbf{x}]/\langle \mathbf{x}^{512} + 1\rangle$ and thus need $512 \cdot \lceil \log 12289 \rceil = 7168$ bits of storage memory. On the other hand, since the coefficients of the secret key are no larger than 12 they can be stored in 5 bits, resulting in a secret key of size $512 \cdot 5 = 2560$ bits. To evaluate the failure probability of decapsulation we model the polynomial as having Gaussian coefficients, using an error analysis similar to the one of Section 5.4.1 in [MR08], we obtain the following probability of failure:

$$\Pr\left[\|2\mathbf{fr} + \mathbf{ge}\|_\infty > \frac{q-1}{2}\right] = 1 - erf\left(\frac{q-1}{\sqrt{40n\sigma^2}}\right)$$

where erf is the Gauss error function. Though we use permutations of fixed polynomials rather than Gaussians, experiments show that the error rate is close to the expected one.

3.2 Security of the KEM

The most effective attacks against the NTRU problem involve *lattice reduction*, and this is the reason why cryptographic constructions based on NTRU are usually referred to as lattice-based schemes. Consider an attack that tries to find the key $\mathbf{e}$ from the public key $\mathbf{h}$ and the ciphertext $\mathbf{t} = 2\mathbf{hr} + \mathbf{e} \bmod q$. We know that $2\mathbf{hr} + \mathbf{e} - \mathbf{t} = \mathbf{0} \bmod q$. If we define the set

$$\Lambda = \{(\mathbf{y}_1, \mathbf{y}_2, z) : \mathbf{y}_i \in \mathbb{Z}[\mathbf{x}]/\langle \mathbf{x}^n \pm 1\rangle, z \in \mathbb{Z} \ \& \ 2\mathbf{h}\mathbf{y}_1 + \mathbf{y}_2 - z\mathbf{t} = \mathbf{0} \bmod q\},$$

then it is not hard to see that it is an additive group over $\mathbb{Z}^d$ with $d = 2n+1$, and thus a lattice. Also notice that the tuple $\mathbf{v} = (\mathbf{r}, \mathbf{e}, 1)$, if seen as a vector over $\mathbb{Z}^d$, is in fact a "short vector" in this lattice. Note that we could have also tried to find the secret key $\mathbf{g}$ from the public key $\mathbf{h}$ but that problem is strictly harder (as a lattice problem) than the one we consider here.

The best known approach to solving this problem is through the use of the BKZ algorithm. This algorithm reduces a basis of the lattice by iteratively solving the shortest vector problem (SVP) in sublattices of fixed dimension β. To assess the hardness of finding the key $\mathbf{e}$ we need to compute the necessary block size β and evaluate the hardness of solving SVP in dimension β. We follow the analysis of [ADPS16]. [12] This attack tries to recover the vector $\mathbf{v} = (\mathbf{r}, \mathbf{e}, 1) \in \Lambda$ by solving a unique SVP problem in this lattice, i.e. we exploit the fact that $\mathbf{v}$ is constructed in such a way that all the other linearly-independent vectors in Λ will be much larger.

We model the behavior of BKZ using the geometric series assumption, i.e. the basis output by BKZ is such that $\|\mathbf{b}_i^*\| = \delta^{d-2i-1} \cdot \mathrm{Vol}(\Lambda)^{1/d}$ where $(\mathbf{b}_i^*)_{1 \le i \le d}$ is the Gram-Schmidt orthogonalization of the basis and $\delta = ((\pi\beta)^{1/\beta}\beta/2\pi e)^{1/2(\beta-1)}$. By construction of the BKZ algorithm we know that $\left\|\mathbf{b}_{d-\beta+1}^*\right\|$ will be the norm of the shortest vector in the lattice obtained by projecting Λ onto the last β vectors of the Gram-Schmidt basis. Thus if the norm of the projection of $\mathbf{v}$ is smaller than the expected norm of $\left\|\mathbf{b}_{d-\beta+1}^*\right\|$ this vector will be detected when using BKZ. The expected norm of the projection of $\mathbf{v}$ is $\|\mathbf{v}\| \sqrt{\beta/d}$, and using the fact that Λ has volume q^n we obtain the following condition on the block size β:

$$\|\mathbf{v}\| \sqrt{\beta/d} \le \delta^{2\beta-d-3} q^{n/d}$$

[12] It is worth noting that the attack defined as the dual attack in that paper does not apply: this attack creates a distinguisher (in our case to know whether $\mathbf{t}$ is well-formed or completely random), but it is not clear how such a distinguisher can be used to break the one-wayness of our KEM.

We obtain the block sizes given in table 1 by solving this equation (we in fact consider a slightly more general equation which relates to the one given in [ADPS16]).

We now consider the time needed to solve the SVP problem in dimension β, it is worth noting that the BKZ algorithm in fact needs to do a large number of calls to its SVP solving oracle which implies that the complexity of running BKZ can be much larger than the complexity of solving SVP in dimension β. There exist two main approaches to solving SVP:

- Enumeration: The shortest vector of the lattice is found by enumerating all the vectors in a sphere of a given radius. This approach does not need large storage space as it proceeds by backtracking down a search tree. It is however (asymptotically) slower than Sieving as it runs in time $2^{O(n^2)}$. The complexity of the enumeration algorithm depends on the quality of the input basis, as such the best known algorithm for enumeration relies on recursively applying BKZ basis reduction as a subroutine of enumeration and pruned enumeration as a subroutine of BKZ reduction. While this algorithm remains super-exponential (and thus highly impractical in high dimensions such as the one of our parameter sets III and IV) it has rather small constants in the exponents and benefits from a quadratic speedup in the quantum setting by applying the results of [Mon15]. We extrapolate the results of [Che13] to the block size given in Table 1 and halve the result to obtain our (quantum) enumeration complexity.
- Sieving: The shortest vector of the lattice is found by sampling an exponential number of lattice vectors and combining them to create increasingly shorter vectors. All known sieving algorithms need a list of at least $(4/3)^{\beta/2+o(\beta)} \approx 2^{0.2075\beta+o(\beta)}$ vectors to be successful (improving the space complexity is an open problem which seems rather unrealistic as this lower bound is already the result of a rather optimistic heuristic assumption). The best known algorithm has complexity $2^{0.292\beta}$ [Laa15, BDGL16] which can be reduced to $2^{0.262\beta}$ on a quantum computer by using Grover's algorithm [LMvdP15]. We use this complexity to obtain the parameters in Table 1.

For extremely high-security situations we would advocate the use of our third or fourth set.[13] But due to the high storage requirement that is intrinsic in the sieving attack, we believe that our first set is also secure against today's best attacks and those in the foreseeable future.

Notice that by raising the decapsulation probability from 2^{-128} to 2^{-30} (between the second and first set of parameters), we were able to gain the extra security necessary for the 128-bit security level. But since parameter set 4 already has extremely high security, we do not think that it makes too much sense to increase its security further by increasing the chance of decapsulation errors.

4 Digital Signatures from NTRU

After the KEM, the second component of our AKE is a digital signature scheme. As for number-theoretic schemes, there are two ways to construct (efficient) lattice-based signature schemes. The first approach is via the Fiat-Shamir transform of a sigma protocol. The currently most efficient such protocol is BLISS, which was proposed in [DDLL13]. The second approach, hash-and-sign, was proposed by Gentry et al. [GPV08], and its most efficient instantiation is based on the hardness of finding short vectors in NTRU lattices [DLP14].

4.1 Hash-and-sign and Message Recovery

In this section we show how to adapt the hash-and-sign scheme from [DLP14] to create a signature scheme with message recovery. What this means is that instead of sending a signature

[13] This is the same parameter set used in [ADPS16] and we achieve essentially the same security as in that paper.

and a message, one can simply send a larger signature which then allows for the entire message to be recovered. We first briefly outline the scheme from [DLP14]. In the below scheme the distribution D_f is some distribution from which secret keys are drawn and the distribution D_s is the distribution of signatures. The goal of the signer is to produce polynomials according to the distribution D_s conditioned on the message that he is signing. He is able to do that using the fact that he knows the secret NTRU keys $\mathbf{f}$ and $\mathbf{g}$.

```
SigKeyGen{ $\mathbf{f}, \mathbf{g} \xleftarrow{\$} D_f, \mathbf{h} \leftarrow \mathbf{f}/\mathbf{g} \bmod q$
           Return $(\mathcal{K}_s, \mathcal{K}_v) = ((\mathbf{f}, \mathbf{g}), \mathbf{h})$ }

Sig($(\mathbf{f}, \mathbf{g}), m$){ $\mathbf{t} \leftarrow H(m)$,
           $\mathbf{s}_1, \mathbf{s}_2 \xleftarrow{\$} D_s$ such that $\mathbf{h}\mathbf{s}_1 + \mathbf{s}_2 = \mathbf{t} \bmod q$
           Return $\sigma = (\mathbf{s}_1, m)$ }

Ver($\mathbf{h}, \sigma = (\mathbf{s}_1, m)$){ $\mathbf{t} \leftarrow H(m)$
           $\mathbf{s}_2 \leftarrow \mathbf{t} - \mathbf{h}\mathbf{s}_1 \bmod q$
           if $\|(\mathbf{s}_1, \mathbf{s}_2)\| < B$
           then accept
           else reject }
```

We now give a brief intuition about the correctness of the scheme. The correctness relies on the fact that the polynomials $\mathbf{s}_1$ and $\mathbf{s}_2$ are drawn according to a discrete Normal distribution D_s with a small standard deviation by using the trapdoor $\mathbf{f}, \mathbf{g}$ (this can be done by using e.g. [GPV08, LP15]) so for an appropriate positive value B, the probability that $\|(\mathbf{s}_1, \mathbf{s}_2)\| < B$ is overwhelming. The condition $\mathbf{s}_1\mathbf{h} + \mathbf{s}_2 = \mathbf{t}$ comes directly from the way $\mathbf{s}_1$ and $\mathbf{s}_2$ are obtained during the sampling.

We point out that as described above, the scheme needs to be stateful in order to be secure. In particular, it needs to output the same signature for the same m, and therefore store the signatures that were output. There are two simple ways to remove this requirement. The first way is for the signer to use a pseudo-random function on the message m (with an additional secret key) in order to derive the randomness that he will use to produce the signature. The second way is for the signer to pick a random string r and compute $\mathbf{t} \leftarrow H(m, r)$ instead of $H(m)$, and then send r along with the signature. This way, the signer is almost certainly assured that he will never sign the same (m, r) pair twice.

We use the security analysis of Section 3.2 to revise the security of [DLP14] and also extend the parameters to include the ring $\mathbb{Z}_{12289}[\mathbf{x}]/\langle \mathbf{x}^{1024} + 1 \rangle$. The security of the scheme is based on the fact that it is hard to recover $\mathbf{f}$ and $\mathbf{g}$ from $\mathbf{h}$, and that forging a signature implies finding short polynomials $\mathbf{s}_1, \mathbf{s}_2$ such that $\mathbf{h}\mathbf{s}_1 + \mathbf{s}_2 = \mathbf{0} \bmod q$ (see [DLP14] for more formal security statements).

Based on the way that the parameters are set, recovering $\mathbf{f}$ and $\mathbf{g}$ is harder than the corresponding problem for the KEM, and so we focus on the problem of forging signatures. As in Section 3.2, this corresponds to solving an SVP instance. However, since the polynomials $\mathbf{s}_1$ and $\mathbf{s}_2$ are much larger than the polynomials of our KEM (here the coefficients of $\mathbf{s}_1$ and $\mathbf{s}_2$ have standard deviation $1.17\sqrt{q}$ which is ≈ 50 times larger than the ones used for the KEM) the corresponding problem is no longer unique-SVP, but rather an approximate-SVP one. To solve this problem we compute the γ factor of the associated lattice, as done in [DLP14] (see Table 4). To solve SVP using the BKZ algorithm, one needs the vector $\mathbf{b}_1$ output by BKZ to be the shortest vector of the lattice, which corresponds to the condition $\delta \leq \gamma$ where

	I	II
Polynomial	$\mathbf{x}^{512}+1$	$\mathbf{x}^{1024}+1$
Modulus	12289	12289
Signing key size (bits)	≈ 2100	≈ 3700
Verification key size (bits)	6956	13912
message size (bits)	6956	13912
hash-and-sign size (bits)	≈ 11600	≈ 23300
message-recovery hash-and-sign size (bits)	≈ 9600	≈ 18900
Gamma factor	1.0041	1.0022
Block size	388	906
sieving complexity (log #operations)	> 102	> 237
sieving space (log bits)	> 85	> 216
enumeration complexity (log #operations)	> 130	> 520

Table 4. Signature parameters for $n = 512$ and 1024, $q = 12289$, and message $m \in \mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$

$\delta = ((\pi\beta)^{1/\beta}\beta/2\pi e)^{1/2(\beta-1)}$. From this equation we obtain the block size β and the analysis of Section 3.2 gives the parameters of Table 4.

4.2 Signature with Message Recovery

Instead of sending the signature $\sigma = (\mathbf{s}_1, m)$ and then letting the verification algorithm recover $\mathbf{s}_2$, it may sometimes be intuitively useful to send the signature as $\mathbf{s}_1, \mathbf{s}_2$ and let the verifier somehow recover m. This may be advantageous because $\mathbf{s}_1$ and $\mathbf{s}_2$ are drawn according to small Gaussians, and may be compressed (see Section 4.3), while m can be any polynomial in $\mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$ and so cannot be encoded in less than $n\log(q)$ bits. In this scenario, a better solution would thus be to modify $\mathbf{t}$ so that sending $\mathbf{s}_1$ and $\mathbf{s}_2$ would allow the verifier to recover m. Our signature with message recovery can be used to recover messages of up to $n\log q - 256$ bits, the scheme we define here can be used for messages $m = (m_1\|m_2)$ of arbitrary size but the second part of the message m_2 will not benefit from message recovery and thus needs to be output as part of the signature.

```
Sig((f, g), m = (m_1||m_2)){  $\mathbf{t} = (m_1 + F(H'(m)) \bmod q \| H'(m))$
                              $\mathbf{s}_1, \mathbf{s}_2 \xleftarrow{\$} D_s$ such that $\mathbf{h}\mathbf{s}_1 + \mathbf{s}_2 = \mathbf{t} \bmod q$
                              Return $\sigma = (\mathbf{s}_1, \mathbf{s}_2, m_2)$   }

Ver(h, σ = (s_1, s_2, m_2)){  $(\mathbf{t}_1\|\mathbf{t}_2) \leftarrow \mathbf{h}\mathbf{s}_1 + \mathbf{s}_2 \bmod q$
                              $m_1 \leftarrow \mathbf{t}_1 - F(\mathbf{t}_2) \bmod q$
                              if $\|(\mathbf{s}_1, \mathbf{s}_2)\| < B$ and $H'(m_1\|m_2) = \mathbf{t}_2$
                              then accept
                              else reject   }
```

While the hash function H mapped to a random element of $\mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle \simeq \mathbb{Z}_q^n$ in the previous scheme, now we want $(m_1 + F(H'(m)) \bmod q \| H'(m))$ to be a random element of

$\mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$. To achieve this, we split the message m into $(m_1 \| m_2)$ with $|m_1| = n \log q - 256$ bits (note that m_2 can be empty if m is small) and we set the hash function H' to output 256 bits. To prove that this scheme is secure we show that we can use an adversary that breaks this scheme to break the one from [DLP14].

Lemma 2. *If the hash functions F and H are modeled as random oracles and an adversary can break the message-recovery hash-and-sign scheme signature unforgeability game with advantage ε, then there is an algorithm that can break the previous hash-and-sign scheme with probability close to ε.*

Proof. We will refer to the standard scheme as scheme 1 and to the message recovery scheme as scheme 2. Assume $\mathcal{A}$ breaks unforgeability in scheme 2 with advantage ε. We construct a simulator $\mathcal{S}$ that breaks scheme 1 with probability ε. Without loss of generality we assume that $\mathcal{A}$ always queries H' on m before asking for a signature for a message m.

- On a query $H'(m)$: $\mathcal{S}$ sets $(m_1 \| m_2) = m$, queries the signing oracle to obtain $H(m)$ and $\sigma = (\mathbf{s}_1, m)$. He then sets $\mathbf{t} = (\mathbf{t}_1 \| \mathbf{t}_2) = H(m)$, $\sigma_m = (\mathbf{s}_1, \mathbf{t} - \mathbf{h}\mathbf{s}_1, m_2)$, $H'(m) \leftarrow \mathbf{t}_2$ and $F(\mathbf{t}_2) \leftarrow \mathbf{t}_1 - m_1 \bmod q$. $\mathcal{S}$ then returns $\mathbf{t}_2$.
- On a query $\mathtt{Sig}(m)$. $\mathcal{S}$ looks up (m, σ_m) in his storage and returns σ_m.
- On a query $F(z)$. $\mathcal{S}$ checks if $F(z)$ has already been defined. If not he sets $F(z)$ to be random.

When $\mathcal{A}$ produces a forgery $(m^* = (m_1^* \| m_2^*), \sigma_m^* = (\mathbf{s}_1^*, \mathbf{s}_2^*, m_2^*))$ for scheme 2 we can assume that he has already queried H' on m^* (otherwise his advantage would be close to 0). This implies that $H(m^*) = (\mathbf{t}_1^* \| \mathbf{t}_2^*)$ with $\mathbf{t}_2^* = H'(m^*)$ and $F(\mathbf{t}_2^*) = \mathbf{t}_1^* - m_1^* \bmod q$. $\mathcal{S}$ then outputs $\sigma^* = (m^*, (\mathbf{s}_1^*, m^*))$ as a forgery for scheme 1. Note that $\mathbf{t}^* = (\mathbf{t}_1^* \| \mathbf{t}_2^*) = (m_1^* + F(H'(m^*)) \bmod q \| H'(m^*)) = H(m^*)$, $\mathbf{s}_2^* = \mathbf{t}^* - \mathbf{h}\mathbf{s}_1^*$, and $\|(\mathbf{s}_1^*, \mathbf{s}_2^*)\| < B$ (since $\mathcal{A}$ produced a valid forgery). Which means that $(m^*, (\mathbf{s}_1^*, m^*))$ is a valid forgery for scheme 1. Giving a random value to $F(z)$ when it is not already set is not an issue since if $\mathcal{A}$ were to query H on a message m such that $H(m) = z$ (which is the only other way to query $F(z)$) then $\mathcal{A}$ would have found a preimage of z for H, which has negligible probability.

4.3 Compressing the Signature

The signature of the message-recovering version of the hash-and-sign scheme consists of a pair $(\mathbf{s}_1, \mathbf{s}_2) \in \mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$ each of whose coefficients follows the Gaussian distribution D_s.

Naively sending the signature would result in a communication cost of $2n \log(q)$ bits per flow. We can use the fact that the distribution is a Gaussian to reduce this cost. Indeed each coefficient of $\mathbf{s}_1$ and $\mathbf{s}_2$ is distributed according to a distribution that is very close to a one dimensional Gaussian of parameter s, the entropy of which is much lower than $\log(q)$ [DDLL13]:

$$X \leftarrow D_s,\ \mathcal{H}(X) \leq \frac{1}{s^3} + \log(\sqrt{2\pi e} s).$$

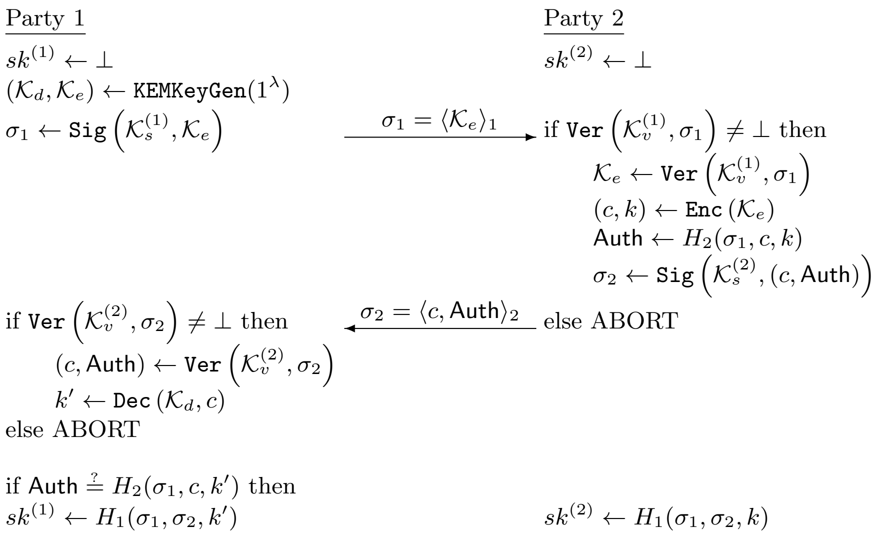
As suggested in [DDLL13], one can use Huffman coding to encode these coefficients. This gives an efficient construction for a prefix-free encoding C that guarantees an average bit size $E[|C(X)|]$ such that:

$$\mathcal{H}(X) \leq E[|C(X)|] < \mathcal{H}(X) + 1.$$

According to [DLP14] the optimal standard deviation for the coefficient of $\mathbf{s}_1$ and $\mathbf{s}_2$ is $1.17\sqrt{q} \approx 129.7$. For such a standard deviation we have $\mathcal{H}(X) \leq 9.06$ which gives the bound $E[|C(X)|] < 10.06$ meaning we need an average bit size of up to 10.06 bits per coefficient, whereas simply sending the coefficients as elements of $\mathbb{Z}_q$ would require 14 bits. In practice, however, Gaussian distributions have a Huffman encoding that is much closer to the lower

Setup:
H_1 and H_2 are two hash function onto $\{0,1\}^{\ell_1}$ and $\{0,1\}^{\ell_2}$ respectively
Each party (i) runs the signing key generation algorithm to get its own pair of keys: $\left(\mathcal{K}_s^{(i)}, \mathcal{K}_v^{(i)}\right) \leftarrow \mathtt{SigKeyGen}(1^\lambda)$

Protocol:



Party 1
$sk^{(1)} \leftarrow \bot$
$(\mathcal{K}_d, \mathcal{K}_e) \leftarrow \mathtt{KEMKeyGen}(1^\lambda)$
$\sigma_1 \leftarrow \mathtt{Sig}\left(\mathcal{K}_s^{(1)}, \mathcal{K}_e\right)$

$\xrightarrow{\sigma_1 = \langle \mathcal{K}_e \rangle_1}$

Party 2
$sk^{(2)} \leftarrow \bot$
if $\mathtt{Ver}\left(\mathcal{K}_v^{(1)}, \sigma_1\right) \neq \bot$ then
$\mathcal{K}_e \leftarrow \mathtt{Ver}\left(\mathcal{K}_v^{(1)}, \sigma_1\right)$
$(c, k) \leftarrow \mathtt{Enc}\left(\mathcal{K}_e\right)$
$\mathsf{Auth} \leftarrow H_2(\sigma_1, c, k)$
$\sigma_2 \leftarrow \mathtt{Sig}\left(\mathcal{K}_s^{(2)}, (c, \mathsf{Auth})\right)$
else ABORT

$\xleftarrow{\sigma_2 = \langle c, \mathsf{Auth} \rangle_2}$

Party 1
if $\mathtt{Ver}\left(\mathcal{K}_v^{(2)}, \sigma_2\right) \neq \bot$ then
$(c, \mathsf{Auth}) \leftarrow \mathtt{Ver}\left(\mathcal{K}_v^{(2)}, \sigma_2\right)$
$k' \leftarrow \mathtt{Dec}\left(\mathcal{K}_d, c\right)$
else ABORT

if $\mathsf{Auth} \overset{?}{=} H_2(\sigma_1, c, k')$ then
$sk^{(1)} \leftarrow H_1(\sigma_1, \sigma_2, k')$

Party 2
$sk^{(2)} \leftarrow H_1(\sigma_1, \sigma_2, k)$

Fig. 3. Generic 2-round forward-secure ε-AKE

bound. Experiments show that in our case we have $E[|C(X)|] = 9.1$ which is close to optimal. For better efficiency we do not encode the low order bits since they are statistically close to uniform.

While we are able to save space by encoding Gaussians, one must first consider the size of the message that will be signed to assess the usefulness of a message-recovery signature scheme. It would be more efficient to simply sign messages whose size is smaller than that of the polynomial $\mathbf{s}_2$ and send them along with only $\mathbf{s}_1$. To have a security of 128 bits we need the H' hash function to have 256 bits of entropy (the hash-and-sign scheme relies on the collision resistance of the hash function) which means that messages of size smaller than $n\log(q)-256$ bits can be hashed into a target vector for the message-recovery signature schemes. For larger messages we simply send the excess bits directly. For messages of bit-length less than $n * 9.1$, however, the message-recovery signature will be less efficient and therefore shouldn't be used. As an example, consider the first column of Table 4. We consider a polynomial of $\mathbb{Z}_{12289}[\mathbf{x}]/\langle \mathbf{x}^{512}+1\rangle$ as the message m. Signing m without message recovery would take space $|\mathbf{s}_1|+|m| = 512{\cdot}9.1+{\cdot}512{\cdot}\log 12289 = 11615$ bits. On the other hand message-recovery can only be used on messages of size up to $512 \cdot \log 12289 - 256$ bits so if we use it the last 256 bits of m will have to be sent along the signature, resulting in a size of $|\mathbf{s}_1| + |\mathbf{s}_2| + 256 = 2 * 512 \cdot 9.1 + 256 = 9575$ bits.

5 The Generic AKE Construction

In this section, we present a generic 2-round construction of a forward-secure ε-AKE that is built from an ε-KEM and a digital signature. As can be seen from the parameters in the previous section, by simply plugging in current lattice primitives into this construction, one already achieves a rather efficient concrete construction. The complete description is provided in Figure 3, and the security claim is the following:

Theorem 3. *The authenticated key exchange $\mathcal{AKE}$ described in Figure 3 is a forward-secure ε-AKE, when H_1 and H_2 are modeled by random oracles onto $\{0,1\}^{\ell_1}$ and $\{0,1\}^{\ell_2}$ respectively, if Σ is a secure signature scheme and $\mathcal{KEM}$ is a secure ε-KEM:*

$$\mathsf{Adv}^{\mathsf{fs-ind}}_{\mathcal{AKE}}(t) \leq n \times \mathsf{Succ}^{\mathsf{suf-cma}}_{\Sigma}(t) + 2q_s^2 q_h \times \mathsf{Succ}^{\mathsf{ow}}_{\mathcal{KEM}}(t) + \frac{q_s^2}{2^{\ell_2}},$$

where n is the number of players involved in the protocols, q_s the number of Send*-queries, and q_h the number of hash-queries. With a checkable KEM, one gets*

$$\mathsf{Adv}^{\mathsf{fs-ind}}_{\mathcal{AKE}}(t) \leq n \times \mathsf{Succ}^{\mathsf{suf-cma}}_{\Sigma}(t) + 2q_s^2 \times \mathsf{Succ}^{\mathsf{ow}}_{\mathcal{KEM}}(t') + \frac{q_s^2}{2^{\ell_2}},$$

where $t' \approx t + q_h \times t_{\mathsf{check}}$, with t_{check} the expected time to check a candidate.

Proof. The security analysis is performed with a sequence of games. It starts with the real security game, between an adversary $\mathcal{A}$ and a challenger, that models the indistinguishability of the fresh session keys, even in a forward-secure way. After small modifications that are shown not to alter much the advantage of the adversary, we have a final game in which the advantage of any adversary is trivially 0, which will allow us to bound the advantage of the adversary in the initial game.

Game $\mathbf{G}_0$: This initial game corresponds to the real attack game in which all the honest players have signing key pairs $(\mathcal{K}_s, \mathcal{K}_v)$; Send-queries are answered exactly as the honest players would do using their keys; a Corrupt-query to a player P is answered by the signing key of P; a Reveal-query to an instance P^i is answered by issuing the session key sk generated by P^i during the execution of the protocol (or $\perp$ if no session key is set); and the Test-query to a *forward-secure fresh* instance P^i is answered, after having flipped a coin b, by either the output of $\mathsf{Reveal}(P^i)$ or $sk \overset{\$}{\leftarrow} \{0,1\}^{\ell_1}$.

By definition we have :

$$\begin{aligned}\mathsf{Adv}^{\mathsf{fs-ind}}_{\mathcal{AKE}}(\mathcal{A}) &= \mathsf{Adv}_{\mathbf{G}_0}(\mathcal{A}) \\ &= |\Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0]|.\end{aligned}$$

Game $\mathbf{G}_1$: In this game, while the official signer is not corrupted, one does not check anymore the validity of the signatures on the flows, but aborts as soon as the proposed signature has not been generated by the simulation of a player.

Under the strong unforgeability of the signature scheme, this modification does not alter much the advantage of the adversary. More precisely, a difference occurs if a signature is refused in the current game (not generated by our simulation) whereas it would have been accepted in the previous game (still valid). Let us raise the event BadReject if one rejects such a valid signature:

$$\mathsf{Adv}_{\mathbf{G}_0}(\mathcal{A}) \leq \mathsf{Adv}_{\mathbf{G}_1}(\mathcal{A}) + \Pr[\mathsf{BadReject}].$$

On the other hand, in order to evaluate $\Pr[\mathsf{BadReject}]$, we interact with a challenger of the strong unforgeability security game, against chosen-message attacks: we choose a random player among the n players involved in the AKE security game, and affect him the challenge verification key, whereas the signing oracle is used to generate signatures on flows under his name. For all the other players, nothing is changed. When BadReject is raised, with probability $1/n$, the forgery is under this challenge key, hence

$$\Pr[\mathsf{BadReject}] \leq n \times \mathsf{Succ}^{\mathsf{suf-cma}}_{\Sigma}(\mathcal{A}).$$

Note that we do not have to wait until the very end of the execution, and so as soon as a forgery is detected, the signature scheme is broken. As a consequence, there are no issues with Corrupt-queries. If a Corrupt-query is asked to the chosen player, we abort the simulation, since we know that no forgery will happen anyway under his key.

Game $\mathbf{G}_2$: Now, all the executions between uncorrupted players are honestly simulated, since the adversary cannot generate flows under their names, but can just replay a previous flow. So we have to guess which session will be tested, in order to embed a specific tuple $(\mathcal{K}_d^*, \mathcal{K}_e^*, c^*, k^*)$, that has been generated as usual: $(\mathcal{K}_d^*, \mathcal{K}_e^*) \leftarrow \texttt{KEMKeyGen}(1^\lambda)$ and $(c^*, k^*) \leftarrow \texttt{Enc}(\mathcal{K}_e^*)$. For that, we have to guess the first time $\mathcal{K}_e^*$ is sent (the q_1-th Send-query) and when c^* is sent back (the q_2-th Send-query). More precisely, the tested session involves the pair $(\mathcal{K}_e^*, c^*)$, but in this session, the first flow σ_1^* is just a replay of the honest output to the q_1-th Send-query, and c^* is included in the σ_2^* output to the q_2-th Send-query.

In this game our simulator, who knows all the signing keys, generates a specific tuple $(\mathcal{K}_d^*, \mathcal{K}_e^*, c^*, k^*)$, and guesses $q_1, q_2 \xleftarrow{\$} \{1 \ldots, q_s\}$, where q_s is an upper-bound on the number of Send-queries. All the queries are answered as usual, except the q_1-th Send-query that is answered by a signature σ_1^* of $\mathcal{K}_e^*$, and the q_2-th Send-query that is answered by a signature σ_2^* of (c^*, Auth^*) if the input to that Send-query is indeed σ_1^* (and then Auth^* is computed in the appropriate way, using H_2 and k^*). If the input to that Send-query is not σ_1^*, or if the Test-query is not asked to this specific session (either to the instance that generated σ_1^* or to the instance that generated σ_2^*), then one aborts the simulation and outputs a random bit b'. With probability greater than $1/q_s^2$, the guesses are correct, and so the simulator does not abort. These guesses do not impact the view of the adversary, since the simulator still uses $\mathcal{K}_d^*$ for completing the protocol executions, we have

$$\mathsf{Adv}_{\mathbf{G}_2}(\mathcal{A}) \geq \mathsf{Adv}_{\mathbf{G}_1}(\mathcal{A})/q_s^2.$$

Game $\mathbf{G}_3$: We would like to no longer use $\mathcal{K}_d^*$ for the simulation, but it is important for detecting decryption failures. However, only the instance who first generated σ_1^* will have to use it, since any other replay of σ_1^* will be sent by the adversary, and so the adversary is playing Party 1: no Reveal or Test-query can be asked to Party 1 in case of replay, but just to Party 2, and for the latter, $sk^{(2)}$ can always be simulated without $\mathcal{K}_d^*$.

For the specific session where σ_1^* is sent for the first time, in case of Reveal or Test-query to Party 1, when $k' \neq k$ (decryption failure), even if the Auth matches, one sets $sk^{(1)} \leftarrow \perp$. This makes a difference only if $H_2(\sigma_1, c, k') = \mathsf{Auth}$ while $k' \neq k$:

$$\mathsf{Adv}_{\mathbf{G}_2}(\mathcal{A}) \leq \mathsf{Adv}_{\mathbf{G}_3}(\mathcal{A}) + \frac{1}{2^{\ell_2}}.$$

Game $\mathbf{G}_4$: Again, for the specific session where σ_1^* is sent for the first time, in case of Reveal or Test-query to Party 1, one flips a coin to decide whether $sk^{(1)} \leftarrow \perp$ (decryption failure) or $sk^{(1)} \leftarrow sk^{(2)}$ (no decryption failure, and $sk^{(2)}$ is computed using the encapsulated key known to the simulator). With probability of one-half, the guess is correct. When this guess is correct, the view of the adversary is unchanged. Let us define the flag CorrectGuess to be true when the guess is correct, and false otherwise. Of course, the simulator will not explicitly compute the value of CorrectGuess, but this Boolean is formally defined. We are now interested in

$$\begin{aligned}\mathsf{Adv}'_{\mathbf{G}_4}(\mathcal{A}) &= |\Pr_{\mathbf{G}_4}[b'=1 \wedge \mathsf{CorrectGuess} | b=1] \\ &\quad - \Pr_{\mathbf{G}_4}[b'=1 \wedge \mathsf{CorrectGuess} | b=0]| \\ &= |\Pr_{\mathbf{G}_3}[b'=1 \wedge \mathsf{CorrectGuess} | b=1] \\ &\quad - \Pr_{\mathbf{G}_3}[b'=1 \wedge \mathsf{CorrectGuess} | b=0]|,\end{aligned}$$

where in $\mathbf{G}_3$ we could have flipped a coin and defined the flag $\mathsf{CorrectGuess}$, but still using $\mathcal{K}_d^*$ to answer the queries. So the flag $\mathsf{CorrectGuess}$ has no impact on the bit b' in $\mathbf{G}_2$, then the two events are independent:

$$\mathsf{Adv}'_{\mathbf{G}_4}(\mathcal{A}) = \mathsf{Adv}_{\mathbf{G}_3}(\mathcal{A})/2.$$

Game $\mathbf{G}_5$: One can note that in the previous game, the simulator does not need $\mathcal{K}_d^*$, so we slightly modify the game by being given the tuple $(\mathcal{K}_e^*, c^*, k^*)$, that has been generated as above: $(\mathcal{K}_d^*, \mathcal{K}_e^*) \leftarrow \texttt{KEMKeyGen}(1^\lambda)$ and $(c^*, k^*) \leftarrow \texttt{Enc}(\mathcal{K}_e^*)$:

$$\mathsf{Adv}'_{\mathbf{G}_5}(\mathcal{A}) = \mathsf{Adv}'_{\mathbf{G}_4}(\mathcal{A}).$$

Game $\mathbf{G}_6$: We are now just given the tuple $(\mathcal{K}_e^*, c^*)$, and without k^* the simulator sets $\mathsf{Auth} \overset{\$}{\leftarrow} \{0,1\}^{\ell_2}$ and $sk^{(2)} \overset{\$}{\leftarrow} \{0,1\}^{\ell_1}$. The simulation of the random oracles is perfectly indistinguishable unless the adversary asks for the actual encapsulated key k^* to either H_1 or H_2. Let us denote by $\mathsf{InHQueries}$ the event that k^* is among the H-queries:

$$\mathsf{Adv}'_{\mathbf{G}_5}(\mathcal{A}) \leq \mathsf{Adv}'_{\mathbf{G}_6}(\mathcal{A}) + \Pr[\mathsf{InHQueries}].$$

In this final game, the session key of the tested session is truly random, and thus indistinguishable from the random case:

$$\mathsf{Adv}'_{\mathbf{G}_6}(\mathcal{A}) = 0.$$

If one recaps all the relations, $\mathsf{Adv}^{\mathsf{fs-ind}}_{\mathcal{AKE}}(\mathcal{A})$ is upper-bounded by

$$n \times \mathsf{Succ}^{\mathsf{suf-cma}}_{\Sigma}(\mathcal{A}) + q_s^2 \times \left(2 \times \Pr[\mathsf{InHQueries}] + \frac{1}{2^{\ell_2}}\right).$$

However, when the encapsulated key k^* of a challenge $(\mathcal{K}_e^*, c^*)$ is in a given list, one can choose it at random, and this is the correct value with probability $1/q_h$, where q_h is the global number of H-queries:

$$\Pr[\mathsf{InHQueries}] \leq q_h \times \mathsf{Succ}^{\mathsf{ow}}_{\mathcal{KEM}}(\mathcal{A}),$$

which leads to the general result

$$\mathsf{Adv}^{\mathsf{fs-ind}}_{\mathcal{AKE}}(t) \leq n \times \mathsf{Succ}^{\mathsf{suf-cma}}_{\Sigma}(t) + 2q_s^2 q_h \times \mathsf{Succ}^{\mathsf{ow}}_{\mathcal{KEM}}(t) + \frac{q_s^2}{2^{\ell_2}}.$$

But in the case a candidate for the encapsulated key can be checked (a checkable KEM), one gets:

$$\mathsf{Adv}^{\mathsf{fs-ind}}_{\mathcal{AKE}}(t) \leq n \times \mathsf{Succ}^{\mathsf{suf-cma}}_{\Sigma}(t) + 2q_s^2 \times \mathsf{Succ}^{\mathsf{ow}}_{\mathcal{KEM}}(t') + \frac{q_s^2}{2^{\ell_2}},$$

where $t' \approx t + q_h \times t_{\mathsf{check}}$, with t_{check} the expected time to check a candidate.

Setup:

H_1 and H_2 are two hash function onto $\{0,1\}^{\ell_1}$ and $\{0,1\}^{2\ell_s}$ respectively
Each party (i) runs the signing key generation algorithm to get its own pair of keys: $\left(\mathcal{K}_s^{(i)}, \mathcal{K}_v^{(i)}\right) \leftarrow \texttt{SigKeyGen}(1^\lambda)$

Protocol:

Party 1		Party 2
$sk^{(1)} \leftarrow \bot$		$sk^{(2)} \leftarrow \bot$
$(\mathcal{K}_d, \mathcal{K}_e) \leftarrow \texttt{KEMKeyGen}(1^\lambda)$	$\xrightarrow{\mathcal{K}_e}$	$(c, k) \leftarrow \texttt{Enc}\,(\mathcal{K}_e)$
		$k_1, k_2 \leftarrow H_2(\mathcal{K}_e, c, k)$
$k' \leftarrow \texttt{Dec}\,(\mathcal{K}_d, c)$	$\xleftarrow{c,\,[\sigma_2]_{k_2}}$	$[\sigma_2]_{k_2} \leftarrow \texttt{Sig}\left(\mathcal{K}_s^{(2)}, (\mathcal{K}_e, c)\right) \oplus k_2$
$k_1', k_2' \leftarrow H_2(\mathcal{K}_e, c, k')$		
if $\texttt{Ver}\left(\mathcal{K}_v^{(2)}, [\sigma_2]_{k_2} \oplus k_2', (\mathcal{K}_e, c)\right)$ then		
$\quad [\sigma_1]_{k_1'} \leftarrow \texttt{Sig}\left(\mathcal{K}_s^{(1)}, (\mathcal{K}_e, c)\right) \oplus k_1'$		
$\quad sk^{(1)} \leftarrow H_1(\mathcal{K}_e, c, k')$		
else ABORT	$\xrightarrow{[\sigma_1]_{k_1'}}$	if $\texttt{Ver}\left(\mathcal{K}_v^{(1)}, [\sigma_1]_{k_1'} \oplus k_1, (\mathcal{K}_e, c)\right)$ then
		$\quad sk^{(2)} \leftarrow H_1(\mathcal{K}_e, c, k)$
		else ABORT

Fig. 4. 3-round forward-secure ε-AKE with identity encryption

6 Adding Anonymity

In this section we consider an augmented version of our previous generic scheme that provides identity hiding, in the same vein as SIGMA-I [Kra03], the basis of IKEv2, with Identity Protection. The description is given in Figure 4. We protect the identities of both parties by hiding their signatures behind an additional encryption. Here, we simply use a one-time pad derived from the hash function H_2, but any symmetric encryption scheme could be used. This additional encryption provides semantic security of the identity of Party 1 against active attackers and semantic security of the identity of Party 2 against passive attackers. The lack of security against active attackers for Party 2 comes from the fact that an attacker can simply impersonate Party 1 for the first flow of the exchange and be able to decrypt the signature of Party 2, that can then be tested with several verification keys.

6.1 Security of the Key

We first show that this protocol (in Figure 4) achieves at least the same security level as the previous one (in Figure 3).

Theorem 4. *The authenticated key exchange $\mathcal{AKE}$ described in Figure 4 is a forward-secure ε-AKE, when H_1 and H_2 are modeled by random oracles onto $\{0,1\}^{\ell_1}$ and $\{0,1\}^{2\ell_s}$ (where ℓ_s is the bitsize of the signature space) respectively, if Σ is a secure signature scheme and $\mathcal{KEM}$ is a secure ε-KEM:*

$$\mathsf{Adv}_{\mathcal{AKE}}^{\mathsf{fs-ind}}(t) \leq (n + q_s^2) \times \mathsf{Succ}_{\Sigma}^{\mathsf{suf-cma}}(t) + 2q_s^2 q_h \times \mathsf{Succ}_{\mathcal{KEM}}^{\mathsf{ow}}(t) + \frac{q_s^2}{2^{\ell_s}}.$$

where n is the number of players involved in the protocols, q_s the number of Send*-queries, and q_h the number of hash-queries. With a checkable KEM, one gets*

$$\mathsf{Adv}_{\mathcal{AKE}}^{\mathsf{fs-ind}}(t) \leq (n + q_s^2) \times \mathsf{Succ}_{\Sigma}^{\mathsf{suf-cma}}(t) + 2q_s^2 \times \mathsf{Succ}_{\mathcal{KEM}}^{\mathsf{ow}}(t') + \frac{q_s^2}{2^{\ell_s}},$$

where $t' \approx t + q_h \times t_{\mathsf{check}}$, with t_{check} the expected time to check a candidate.

Proof. The security analysis is mostly identical to the previous one. Verifying that there was no decryption error can now be done using the masks (k_2, k_2') so the authenticator is no longer necessary, and Party 2 can no longer abort after the first flow so the first hybrid game is slightly different.

Game $\mathbf{G}_0$: This initial game corresponds to the real attack game in which all the honest players have signing key pairs $(\mathcal{K}_s, \mathcal{K}_v)$; Send-queries are answered exactly as the honest players would do using their keys; a Corrupt-query to a player P is answered by the signing key of P; a Reveal-query to an instance P^i is answered by the session key sk generated by P^i during the execution of the protocol (or $\perp$ if no session key is set); and the Test-query to a *forward-secure fresh* instance P^i is answered, after having flipped a coin b, by either the output of $\mathsf{Reveal}(P^i)$ or $sk \xleftarrow{\$} \{0,1\}^{\ell_1}$.

By definition we have :

$$\begin{aligned}\mathsf{Adv}^{\mathsf{fs-ind}}_{\mathcal{AKE}}(\mathcal{A}) &= \mathsf{Adv}_{\mathbf{G}_0}(\mathcal{A}) \\ &= |\ \Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0]\ |.\end{aligned}$$

Game $\mathbf{G}_1$: We would like to abort as soon as a flow has been modified, however doing so would alter the view of the adversary. Instead we define the BadKey flag to be raised when the first flow of the protocol has not been generated by the simulator. When playing as Party 1 we abort if the second flow of the protocol has not been generated by the simulator. When playing Party 2 we abort after the third flow if either it has not been generated by the simulator or if the BadKey flag was previously raised. As before the difference in the advantage of the attacker can be bounded by the probability of a BadReject event. By interacting with a challenger of the strong unforgeability security game we obtain like previously

$$\Pr[\mathsf{BadReject}] \le n \times \mathsf{Succ}^{\mathsf{suf-cma}}_{\Sigma}(\mathcal{A}).$$

Which gives

$$\mathsf{Adv}_{\mathbf{G}_0}(\mathcal{A}) \le \mathsf{Adv}_{\mathbf{G}_1}(\mathcal{A}) + n \times \mathsf{Succ}^{\mathsf{suf-cma}}_{\Sigma}(\mathcal{A}).$$

Game $\mathbf{G}_2$: We do as in $\mathbf{G}_2$ in the proof of Theorem 3, with a guess of the two critical flows, where to inject the challenge inpout $\mathcal{K}_e^*$ and c^*.

$$\mathsf{Adv}_{\mathbf{G}_2}(\mathcal{A}) \ge \mathsf{Adv}_{\mathbf{G}_1}(\mathcal{A})/q_s^2.$$

Game $\mathbf{G}_3$: For the session where $\mathcal{K}_e^*$ is sent for the first time, in case of Reveal or Test-query to Party 1, when $k' \neq k$ (decryption failure), even if the signature verification goes through, one sets $sk^{(1)} \leftarrow \perp$. This makes a difference if either $k_2 = k_2'$ or $[\sigma_2]_{k_2} \oplus k_2'$ is a valid forgery:

$$\mathsf{Adv}_{\mathbf{G}_2}(\mathcal{A}) \le \mathsf{Adv}_{\mathbf{G}_3}(\mathcal{A}) + \frac{1}{2^{\ell_s}} + \mathsf{Succ}^{\mathsf{suf-cma}}_{\Sigma}(t).$$

Game $\mathbf{G}_4$: We do as in $\mathbf{G}_4$ in the proof of Theorem 3, with a guess for a possible decryption failure.

$$\mathsf{Adv}'_{\mathbf{G}_4}(\mathcal{A}) = \mathsf{Adv}_{\mathbf{G}_3}(\mathcal{A})/2.$$

Game $\mathbf{G}_5$: We do as in $\mathbf{G}_5$ in the proof of Theorem 3, without the decryption key.

$$\mathsf{Adv}'_{\mathbf{G}_5}(\mathcal{A}) = \mathsf{Adv}'_{\mathbf{G}_4}(\mathcal{A}).$$

Game $\mathbf{G}_6$: We do as in $\mathbf{G}_6$ in the proof of Theorem 3, with random $(k_1, k_2) \xleftarrow{\$} \{0,1\}^{2s}$

$$\mathsf{Adv}'_{\mathbf{G}_5}(\mathcal{A}) \le \mathsf{Adv}'_{\mathbf{G}_6}(\mathcal{A}) + \Pr[\mathsf{InHQueries}].$$

We obtain the result:

$$\mathsf{Adv}^{\mathsf{fs-ind}}_{\mathcal{AKE}}(t) \leq (n + q_s^2) \times \mathsf{Succ}^{\mathsf{suf-cma}}_{\Sigma}(t) + 2q_s^2 q_h \times \mathsf{Succ}^{\mathsf{ow}}_{\mathcal{KEM}}(t) + \frac{q_s^2}{2^{\ell_s}},$$

and in the case of a checkable KEM:

$$\mathsf{Adv}^{\mathsf{fs-ind}}_{\mathcal{AKE}}(t) \leq (n + q_s^2) \times \mathsf{Succ}^{\mathsf{suf-cma}}_{\Sigma}(t) + 2q_s^2 \times \mathsf{Succ}^{\mathsf{ow}}_{\mathcal{KEM}}(t') + \frac{q_s^2}{2^{\ell_s}}.$$

6.2 Anonymity

We now consider the identity protection in the new protocol (in Figure 4). To this aim, we consider an interaction between an attacker and a challenger in which the challenger choses an identity at random between U_0 and U_1 (represented by the verification and signature key associated to each identity) and runs the key exchange protocol with the attacker. After this run the attacker should be unable to distinguish the identity used by the challenger with anything but a negligible advantage. More formally we define four security games corresponding to whether the adversary is passive or active and whether it attacks the identity of Party 1 or Party 2.

- The security game for a **passive attacker against Party 1** is as follows:
  - The adversary $\mathcal{A}$ generates signing keys $(\mathcal{K}_s^{(0)}, \mathcal{K}_v^{(0)})$, $(\mathcal{K}_s^{(1)}, \mathcal{K}_v^{(1)})$, and $(\mathcal{K}_s, \mathcal{K}_v)$ for the identities U_0, U_1, and V, respectively
  - The challenger $\mathcal{C}$ flips a bit b
  - $\mathcal{A}$ observes a simulation of an honest execution of the protocol between the user U_b acting as Party 1 and user V acting as Party 2.
  - $\mathcal{A}$ outputs its guess b' of the bit b

  The advantage of the passive adversary against Player 1 is $\mathsf{Adv}^{pass-1}_{\mathcal{AKE}}(\mathcal{A}) = |\Pr[b' = 1|b = 1] - \Pr[b' = 1|b = 0]|$.
- The security game for a **passive attacker against Party 2** is as follows:
  - The adversary $\mathcal{A}$ generates signing keys $(\mathcal{K}_s, \mathcal{K}_v)$, $(\mathcal{K}_s^{(0)}, \mathcal{K}_v^{(0)})$, and $(\mathcal{K}_s^{(1)}, \mathcal{K}_v^{(1)})$, and for the identities V, U_0, and U_1, respectively
  - The challenger $\mathcal{C}$ flips a bit b
  - $\mathcal{A}$ observes a simulation of an honest execution of the protocol between the user V acting as Party 1 and user U_b acting as Party 2.
  - $\mathcal{A}$ outputs its guess b' of the bit b

  The advantage of the passive adversary against Player 2 is $\mathsf{Adv}^{pass-2}_{\mathcal{AKE}}(\mathcal{A}) = |\Pr[b' = 1|b = 1] - \Pr[b' = 1|b = 0]|$.
- The security game for an **active attacker against Party 1** is as follows:
  - The adversary $\mathcal{A}$ generates signing keys $(\mathcal{K}_s^{(0)}, \mathcal{K}_v^{(0)})$ and $(\mathcal{K}_s^{(1)}, \mathcal{K}_v^{(1)})$ for two identities U_0 and U_1
  - The challenger $\mathcal{C}$ flips a bit b
  - $\mathcal{A}$ plays on behalf of Party 2 (but without a valid signature key) and runs the key exchange protocol with $\mathcal{C}$ who uses identity U_b for Party 1
  - $\mathcal{A}$ outputs his guess b' of the bit b

  The advantage of an active adversary against Player 1 is $\mathsf{Adv}^{act-1}_{\mathcal{AKE}}(\mathcal{A}) = |\Pr[b' = 1|b = 1] - \Pr[b' = 1|b = 0]|$.

One can note that the passive attacks against Party 1 and Party 2 are symmetric, and so similar. On the other hand while we can obtain security for an active attacker against Party 1, we cannot achieve such a notion for an attacker against Party 2. Since the protocol does not have to terminate for an attack to occur, there is no chance to hide the identity of Party 2 against active attackers when the protocol has no more than 3 flows. In our case, if $\mathcal{A}$ generates $(\mathcal{K}_e, \mathcal{K}_d)$ and plays honestly until it receives U_b's flow, it can then decrypt the signature and try the verification keys $\mathcal{K}_v^{(0)}$ and $\mathcal{K}_v^{(1)}$ to recover the bit b. $\mathcal{A}$ cannot complete the protocol, but it can guess b with overwhelming probability.

Theorem 5. *The authenticated key exchange $\mathcal{AKE}$ described in Figure 4 protects identities of both parties against passive attackers, as well as Party 1's identity against active attackers.*

Proof. Let us first consider Party 2's anonymity against a passive attacker: The sequence of games used in this proof is similar to the one used for semantic security (for Theorem 4), but it is somewhat simpler as there is no need to guess the flows nor to prove that flows are not modified, since we are against a passive adversary.

Game $\mathbf{G}_0$: This game corresponds to the real attack game in which $\mathcal{A}$ generates the signature keys for V, U_0, U_1. The simulator flips a bit b and runs the protocol between V and U_b. $\mathcal{A}$ then outputs his guess b'.

$$\mathsf{Adv}^{pass-2}_{\mathcal{AKE}}(\mathcal{A}) = \mathsf{Adv}_{\mathbf{G}_0}(\mathcal{A}) = |\ \Pr[b' = 1|b = 1] - \Pr[b' = 1|b = 0]\ |.$$

Game $\mathbf{G}_1$: When decrypting, if $k' \neq k$ (decryption failure), even if the signature verification goes through, one sets $sk^{(1)} \leftarrow \perp$. This makes a difference if either $k_2 = k'_2$ or $[\sigma_2]_{k_2} \oplus k'_2$ is a valid forgery:

$$\mathsf{Adv}_{\mathbf{G}_0}(\mathcal{A}) \leq \mathsf{Adv}_{\mathbf{G}_1}(\mathcal{A}) + \frac{1}{2^{\ell_s}} + \mathsf{Succ}^{\mathsf{suf-cma}}_{\Sigma}(t).$$

Game $\mathbf{G}_2$: Since Party 1 now aborts when $k \neq k'$ we can set $(k'_1, k'_2) = (k_1, k_2)$. In this game we set $(k_1, k_2) \overset{\$}{\leftarrow} \{0,1\}^{2\ell_s}$. This simulation of the random oracle is perfectly indistinguishable unless $\mathcal{A}$ queries the key k used by Party 2 to H_2. We denote by $\mathsf{InHQueries}$ such an event, we have:

$$\mathsf{Adv}_{\mathbf{G}_1}(\mathcal{A}) \leq \mathsf{Adv}_{\mathbf{G}_2}(\mathcal{A}) + \Pr[\mathsf{InHQueries}].$$

Given q_h the number of queries made to H_2, if k is in this list of queries one can guess which query at random and break the one-wayness of $\mathcal{KEM}$ with probability $1/q_h$.

$$\Pr[\mathsf{InHQueries}] \leq q_h \times \mathsf{Succ}^{\mathsf{ow}}_{\mathcal{KEM}}(\mathcal{A}),$$

Game $\mathbf{G}_3$: In this game we take $[\sigma_2]_{k_2} \overset{\$}{\leftarrow} \{0,1\}^{2\ell_s}$ this does not affect the advantage of the adversary since the mask k_2 was already completely random.

$$\mathsf{Adv}_{\mathbf{G}_2}(\mathcal{A}) = \mathsf{Adv}_{\mathbf{G}_3}(\mathcal{A}).$$

In this final game since the encrypted signature $[\sigma_2]_{k_2}$ is truly random, $\mathcal{A}$ has no advantage in deciding whether it belongs to U_0 or U_1:

$$\mathsf{Adv}_{\mathbf{G}_3}(\mathcal{A}) = 0$$

Using these relations we obtain the final advantage of the adversary:

$$\mathsf{Adv}^{pass-2}_{\mathcal{AKE}}(t) \leq \mathsf{Succ}^{\mathsf{suf-cma}}_{\Sigma}(t) + q_h \times \mathsf{Succ}^{\mathsf{ow}}_{\mathcal{KEM}}(t) + \frac{1}{2^{\ell_s}}.$$

We omit the proof of security for passive attackers against Party 1 as it is nearly identical.

We now consider the security against an active attacker on Party 1: In the game $\mathbf{G}_1$ in the proof of Theorem 4 we show that aborting as soon as the second flow is modified does not significantly reduce the advantage of the adversary. $\mathcal{A}$'s knowledge of the signature key of Party 1 has no incidence on this proof. This implies that an active adversary will cause an abort before the third flow. It is then direct that aborting before the third flow of the protocol prevents the attacker from distinguishing between users U_0 and U_1 as the first flow of the protocol is completely independent from the identity of Party 1.

References

ABD16. Martin Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions: Cryptanalysis of some FHE and graded encoding schemes. *Crypto*, 2016.

ADPS16. Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - a new hope. *USENIX*, 2016.

ADR02. Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 83–107. Springer, April / May 2002.

BCK98. 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.

BCLvV16. Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. NTRU prime. *IACR Cryptology ePrint Archive*, 2016:461, 2016.

BCNS15. Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 553–570, 2015.

BDGL16. Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 10–24, 2016.

BDJR97. Mihir Bellare, Anand Desai, Eric Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *38th FOCS*, pages 394–403. IEEE Computer Society Press, October 1997.

BR93. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

BR94. 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, August 1994.

Che13. Yuanmi Chen. *Lattice reduction and concrete security of fully homomorphic encryption.* PhD thesis, Universite Paris Diderot, Paris, 2013.

CJL16. Jung Hee Cheon, Jinhyuck Jeong, and Changmin Lee. An algorithm for NTRU problems and cryptanalysis of the GGH multilinear map without an encoding of zero. *IACR Cryptology ePrint Archive*, 2016.

CK02. Ran Canetti and Hugo Krawczyk. Security analysis of ike's signature-based key-exchange protocol. In *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, pages 143–161, 2002.

DDLL13. Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In *CRYPTO (1)*, pages 40–56, 2013.

DLP14. Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over NTRU lattices. In *ASIACRYPT*, pages 22–41, 2014.

DP15. Léo Ducas and Thomas Prest. A hybrid gaussian sampler for lattices over rings. *IACR Cryptology ePrint Archive*, 2015:660, 2015.

DXL12. Jintai Ding, Xiang Xie, and Xiaodong Lin. A simple provably secure key exchange scheme based on the learning with errors problem. Cryptology ePrint Archive, Report 2012/688, 2012. `http://eprint.iacr.org/`.

GLP12. Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In *CHES*, pages 530–547, 2012.

GMR88. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

HHGP$^+$03. Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. Ntrusign: Digital signatures using the ntru lattice. In *CT-RSA*, pages 122–140, 2003.

HNP$^+$03. Nick Howgrave-Graham, Phong Q. Nguyen, David Pointcheval, John Proos, Joseph H. Silverman, Ari Singer, and William Whyte. The impact of decryption failures on the security of NTRU encryption. In *CRYPTO*, pages 226–246, 2003.

How07. Nick Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In *CRYPTO*, pages 150–169, 2007.

HPS98. Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *ANTS*, pages 267–288, 1998.

HPS$^+$14. Jeffrey Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, and William Whyte. Transcript secure signatures based on modular lattices. In *PQCrypto*, pages 142–159, 2014.

HPS$^+$15. Jeffrey Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, William Whyte, and Zhenfei Zhang. Choosing parameters for ntruencrypt. *IACR Cryptology ePrint Archive*, 2015:708, 2015.

Kra03. Hugo Krawczyk. SIGMA: The "SIGn-and-MAc" approach to authenticated Diffie-Hellman and its use in the IKE protocols. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 400–425. Springer, August 2003.

Kra05. Hugo Krawczyk. HMQV: A high-performance secure diffie-hellman protocol. In *CRYPTO*, pages 546–566, 2005.

Laa15. Thijs Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 3–22, 2015.

LMQ$^+$03. Laurie Law, Alfred Menezes, Minghua Qu, Jerome A. Solinas, and Scott A. Vanstone. An efficient protocol for authenticated key agreement. *Des. Codes Cryptography*, 28(2):119–134, 2003.

LMvdP15. Thijs Laarhoven, Michele Mosca, and Joop van de Pol. Finding shortest lattice vectors faster using quantum search. *Des. Codes Cryptography*, 77(2-3):375–400, 2015.

LN16. Patrick Longa and Michael Naehrig. Speeding up the number theoretic transform for faster ideal lattice-based cryptography. *IACR Cryptology ePrint Archive*, 2016:504, 2016.

LP15. Vadim Lyubashevsky and Thomas Prest. Quadratic time, linear space algorithms for gram-schmidt orthogonalization and gaussian sampling in structured lattices. In *EUROCRYPT*, pages 789–815, 2015.

LPR10. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, pages 1–23, 2010.

LPR13a. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6):43, 2013. Preliminary version appeared in EUROCRYPT 2010.

LPR13b. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-lwe cryptography. In *EUROCRYPT*, pages 35–54, 2013.

LW15. Vadim Lyubashevsky and Daniel Wichs. Simple lattice trapdoor sampling from a broad class of distributions. In *Public-Key Cryptography - PKC*, pages 716–730, 2015.

Lyu12. Vadim Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, pages 738–755, 2012.

Mon15. Ashley Montanaro. Quantum walk speedup of backtracking algorithms. *CoRR*, abs/1509.02374, 2015.

MP12. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718, 2012.

MR08. Daniele Micciancio and Oded Regev. Lattice-based cryptography. In Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Chapter in Post-quantum Cryptography*, pages 147–191. Springer, 2008.

Pei10. Chris Peikert. An efficient and parallel gaussian sampler for lattices. In *CRYPTO*, pages 80–97, 2010.

Pei14. Chris Peikert. Lattice cryptography for the internet. In *Post-Quantum Cryptography*, pages 197–219, 2014.

Sho01. Victor Shoup. A proposal for an ISO standard for public key encryption. Cryptology ePrint Archive, Report 2001/112, 2001. http://eprint.iacr.org/2001/112.

SPMLS02. Jacques Stern, David Pointcheval, John Malone-Lee, and Nigel P. Smart. Flaws in applying proof methodologies to signature schemes. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 93–110. Springer, August 2002.

SS11. Damien Stehlé and Ron Steinfeld. Making NTRU as secure as worst-case problems over ideal lattices. In *EUROCRYPT*, pages 27–47, 2011.

ZZD$^+$15. Jiang Zhang, Zhenfeng Zhang, Jintai Ding, Michael Snook, and Özgür Dagdelen. Authenticated key exchange from ideal lattices. In *EUROCRYPT*, pages 719–751, 2015.