# Extended Abstract: NICE-PAKE and TEMPO Instantiations from MLWE Rerandomizable Splittable KEMs

Nouri Alnahawi[1,3,4] and Alexander Wiesmaier[1,2,3]

[1] Hochschule Darmstadt, Germany
[2] National Research Center for Applied Cybersecurity ATHENE, Germany
[3] European University of Technology, European Union
[4] University of Regensburg, Germany

**Abstract.** We propose two novel instantiations for the NICE-PAKE and TEMPO protocols, which were presented by Alnahawi et al. (ePrint:2024/1957), and Arriaga, Barbosa and Jarecki (ePrint:2025/1399) repectively. Our instantiations are not formally analyzed yet, but build upon known KEM security assumptions and well-studied PAKE designs. Therefore, we believe there is a great chance that a formal proof in the Universal Composability (UC) framework should also hold.

Our constructions combine three concepts: 1) Lattice KEMs with Splittable public keys of the form $\mathbf{As} + \mathbf{e}$ as introduced in Arriaga et al. (AC24:ABJS), Alnahawi et al. (ePrint:2024/1957) and Arriaga et al. (ePrint:2025/1399). 2) The Programmable Only Once Function (POPF) realized as a 2-round Feistel (2F) as in McQuoid, Rosulek and Roy (CCS20:MRR) and Arriaga , Barbosa and Jarecki (ePrint:2025/231). 3) Rerandomizable KEM as introduced in Duverger et al. (CCS25:DFJ+).

Similar to the aforementioned works, our goal is to eliminate the usage of the Ideal Cipher (IC) in (O)EKE-style KEM-based PQC PAKEs, the motivation of which is adequately and extensively explained in the cited literature. Our main contribution lies within the following two aspects: 1) Mitigating malicious public key generation attacks in the NICE-PAKE construction. 2) Defining a mechanism to realize the missing group operation in the 2F public key authentication step in NoIC-PAKE. Briefly put, we utilize the rerandomization procedure of (CCS25:DFJ+) to sample a second uniform MLWE sample, which is in turn used to shift the initiator's public key forming another fresh sample that yields indistinguishable from uniform. By doing so, we assume that we can enhance the security of NICE-PAKE to withstand a certain class of attacks, and reduce the computational complexity of the 2F instantiation relying on obfuscation in the OQUAKE variant of the 2F PAKE, which was introduced by Vos et al. (ePrint:2025/1343).

Obviously, we cannot ascertain the security of our proposed constructions without conducting a complete and thorough formal analysis. Hence, remaining open questions and future work include defining an indistinguishable UC simulator in the ideal UC world that is also capable of extracting adversarial password guesses. Further, we need to identify the concrete KEM properties required to prove security in UC via the common game-hopping reductionist proof approach.

**Keywords:** PAKE · PQC · MLWE · KEM

E-mail: nouri.alnahawi@h-da.de (Nouri Alnahawi), alexander.wiesmaier@h-da.de (Alexander Wiesmaier)

## 1 Progress on PQC PAKE

Research on Password Authenticated Key Exchange (PAKE) is progressing rapidly towards realizing secure constructions utilizing black-box PQC KEMs. Over the course of the last three years, multiple proposals revisited the original PAKE design of Encrypted Key Exchange (EKE) by Bellovin and Merit [Bel92]. As a result, several PQC PAKE constructions emerged, the most recent of which focus on eliminating the usage of IC to model block ciphers used in the public key authentication step. A complete overview of these so-called generic KEM PAKEs can be found in [AHMW25].

## 2 OEKE, NoIC-PAKE, NICE-PAKE and TEMPO

The original OEKE PAKE design [Bel92] revisited in OCAKE [BCP+23, AHHR24] utilizing PQC KEMs, relies on a symmetric key derived from the honest password to encrypt the initiator's KEM public key (first message), and adds a confirmation tag to the responders KEM ciphertext (second message). Due to multiple reasons, thoroughly explained in [AASA+24, AHMW25, ABJ25a], realizing the encryption of the first message as block cipher modeled as an IC results in PAKEs that cannon be proven entirely secure in the presence of a quantum adversary. On the other hand, the IC allows programming in proofs [HHKR25], which enables comfortable extraction of adversarial password guesses. Without the IC, programming based on password extraction becomes infeasible as shown in [AASA+24, HHKR25]. Thus, one would either need a simulator in the quantum IC (QIC) model [HHKR25], or a different construction that allows efficient programming similar to an IC [MRR20, ABJ25a]. We briefly review three PQC PAKEs without an IC.

### 2.1 NoIC-PAKE using POPF

McQuoid et al.[MRR20] introduced the idea of a POPF, which can be realized as a 2F (Fig. 1). As later shown by Januzelli, Roy, and Xu [JRX25], and also Arriaga, Barbosa and Jarecki [ABJ25a], the POPF/2F construction can be defined as an ideal functionality in the UC framework, or can be utilized in a white-box manner. In either case, the 2F allows the initiator to program an input (i.e., the public key in PAKE) into a deterministic output using the password, which can then be evaluated by the receiver in order to recover the input provided that they use the same password. In that sense, the pair $(r, M)$ in Fig. 1 corresponds to a random string $r$ and a KEM public key $M$. $k$ is the password (or a password derived value) used to program $M$ into $T$, which is in turn the modified, shifted, authenticated etc., initiator's public key.

```
$\mathsf{2F}_k(r, M)$
$R \leftarrow \mathcal{H}_1(k, r)$
$T \leftarrow M \odot R$
$t \leftarrow \mathcal{H}_2(k, T)$
$s \leftarrow r \oplus t$
**return** $(s, T)$
```

```
$\mathsf{2F}_k^{-1}(s, T)$
$t \leftarrow \mathcal{H}_2(k, T)$
$r \leftarrow s \oplus t$
$R \leftarrow \mathcal{H}_(k, r)$
$M \leftarrow T \odot R^{-1}$
**return** $(r, M)$
```

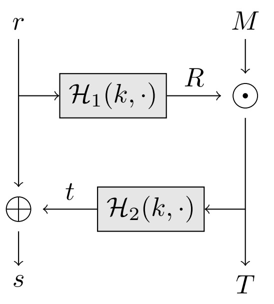

**Figure 1:** The 2-Round Fiestel (2F), where $\oplus$ is an XOR operation on bit strings, $\odot$ is a group operation in $G$, and $(.)^{-1}$ computes the inverse of a group element [ABJ25a].

To the best of our knowledge, the only existing concrete instantiation for a KEM-based PAKE utilizing the 2F is OQUAKE [VJW+25], which uses ML-KEM (aka CRYSTALS-Kyber). That is, since Arriaga et al. [ABJ25a] only defined that the PAKE public key is a group element, but did not name a specific group. Considering that ML-KEM is built from a Module Learning with Errors (MLWE) lattice PKE, the group over which the 2F operates is the ring of polynomials $R_q^k$. While defining a group operation over this group is feasible, ML-KEM public keys cannot be viewed as entirely and unconditionally uniform over the group due to a certain encoding structure explained perfectly in the original CRYSTALS-Kyber proposal [Bos18] and in the Obfuscated Key Exchange construction of Günther et al. [GSV24]. In other words, modifying an ML-KEM public key using a 2F (or an IC for that matter) may result in a value that cannot and should not be a properly generated public key [AASA+24, GRSV25].

This specific property is referred to as Public Key Uniformity [AHHR24, ABJ25a, AHMW25] and is meant to ensure that PAKE adversaries cannot distinguish real public keys following a wrong password guess [AASA+24]. To work around this issue, an algorithm to *obfuscate* public keys was proposed in [GSV24], which is utilized in the Hybrid Obfuscated Key Exchange proposal [GRSV25] and OQUAKE [VJW+25]. The idea of [GSV24] is to deterministically convert a public key into a uniform bit string representation that is (roughly speaking) the sum of the MLWE sample polynomial coefficients interpreted as a large integer. While this works perfectly, it has a small drawback, namely that not all values in the public key space can be obfuscated, which reduces this space by almost 50% and induces the probability of having to repeat key generation by a factor two. Naturally, the obfuscation routine also incurs some computational overhead, which is still manageable.

Going back to OQUAKE [VJW+25], the construction utilizes the same 2F as in Fig. 1. However, the input public key is first obfuscated into an integer representation before masking it with the hash of the randomness and the password using an XOR operation. Meaning, the group operation in [ABJ25a] is now defined as bit-wise XOR over uniform bit strings. Nevertheless, this instantiation may still suffer from the same drawbacks of the original obfuscations mechanism.

## 2.2 NICE-PAKE

The NICE-PAKE protocol [AASA+24] attempts to eliminate the IC in a different way by splitting the KEM public key into two components (Fig. 2). These are the public seed $\rho$ used in the matrix expansion routine within the KEM KEM key generation function, and the lattice sample $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}$. The so-called $\mathbf{b}$-part is sent in the clear (without encrypting or modifying it), and the seed is masked with the hash of the password using an XOR operation. The construction succeeds in ensuring matching final keys only under usage of the same password by both parties. However, compared to the 2F method, or the original IC encryption method for that matter, this design has two main drawbacks: 1) A simulator in a proof is unable to extract adversarial password guesses because the masking cannot be programmed (in the ROM) without prior knowledge of $\rho$. 2) An adversary impersonating the initiator controls the crucial part of the public key, and may very likely send a maliciously crafted $\mathbf{b}$-part, which allows them to either guess the honest password or decapsualte any honest ciphertext. We focus here on the second issue, since the first one is not a must in BPR PAKE proofs, and requires a password extracting simulator in UC. Although NICE-PAKE addresses its limitations and proposes a validation method to identify malicious public keys, the validation incurs additional computational cost. The other mitigation strategy mentioned in NICE-PAKE is to make use of rerandomizable lattice KEMs, on which we elaborate in Sec. 4.

```
Splittable Public Key of (R)(M)LWE
$(pk, sk) \leftarrow \mathsf{KEM.KGen}(1^\lambda)$
$(\mathbf{A}, \mathbf{b}) \leftarrow pk : (\mathbf{A}, \mathbf{b} = \mathbf{As} + \mathbf{e})$
$\rho \leftarrow pk$ // Extract seed bit string
......... Knowing $\mathbf{b}$ reuse $\rho$ .........
$\mathbf{A} \leftarrow \rho$ // Deterministic expansion
$pk \leftarrow (\mathbf{A}, \mathbf{b})$
return $pk$
```

**Figure 2:** Splitting the public key of lattice KEM to obtain the fixed-length seed $\rho$ appended (or prepended) to the public key $pk$ [AASA+24, AHMW25].

## 2.3 TEMPO

The TEMPO protocol [ABJ25b] essentially addresses a different concern, namely timing attacks on the matrix expansion routine in KEM key generation, which is perfectly explained in the paper and hence we do not need to repeat it here in detail. The interesting aspect of TEMPO is that it flips the idea of NICE-PAKE (i.e, splitting a public key as in Fig. 2) to send the seed in the clear instead, and modify the public key using 2F. The seed is additionally added to the inputs of the hash functions along with the password and the randomness. Thus, the only difference to NoIC or OQUAKE on the design level is that the authenticated public key is made of three components instead of two, one of which is the seed in plain text. We make use of this idea and extend it into a new instantiation of 2F based on rerandomizable lattice KEMs in Sec. 5.

## 3 Rerandomizable Lattice KEM

Duverger et al. [DFJ+25] introduced a new KEM that mitigates MitM attacks targeting a key exchange through sanitizing the initiator's public key. The authors refer to their scheme as RKEM (as in Rerandomizable KEM) and show how it can be instantiated using MLWE (i.e., ML-KEM) with concrete parameters based on MLWE and Hint-MLWE assumptions. The idea is quite attractive and promising, since it simply defines additional algorithms compatible with the familiar KEM interface. Without going into much detail on the proven security and correctness of RKEM, we briefly review the idea.

As previously mentioned, a lattice KEM public key is basically a lattice (e.g., MLWE) sample of the form $(\mathbf{A}, \mathbf{b}) : \mathbf{b} = \mathbf{As} + \mathbf{e}$. Assuming that $\mathbf{A}$ is public, it is possible for anyone to sample a new secret $\mathbf{s}'$ and a new error term $\mathbf{e}'$ to create a fresh sample $(\mathbf{A}, \mathbf{b}') : \mathbf{b}' = \mathbf{As}' + \mathbf{e}'$. Intuitively, it is possible to obtain a fresh sample indistinguishable from uniform by simply adding $\mathbf{b}$ and $\mathbf{b}'$ together. If the secrets and errors are sampled from the same distribution and remain below the decryption threshold defined in the KEM parameters (based on the underlying assumption), it is then possible to use the new sample as a fresh independent public key for encapsulation. The decapsulation on the other hand remains correct if the initiator can combine their original secrets with the new ones. Hence, Duverger et al. generate the second set of secrets from an additional seed that allows sampling from the same secret/error distribution. As a result, the encapsulation algorithm takes as input the rerandomized public key. In turn, the decapsualtion algorithm takes as input a ciphertext, the original secret key, and the second seed used in sampling the secrets of the second public key.

We make use of this idea in two ways. For NICE-PAKE, since the matrix **A** can only be expanded from the correct seed obtained under the correct password. It follows, that a possibly malicious **b**-part can be sanitized a la RKEM, but only under knowledge of the correct password. As for TEMPO, the **A**-part seed is sent in the clear anyway, so it suffices to rerandomize the **b**-part inside the 2F instead of obfuscating it and XOR-ing it.

## 4 NICE-PAKE:RE

In the following, we describe the modified NICE-PAKE protocol (Fig. 3) utilizing the Rerandomized KEM (RKEM). The main obstacle in achieving a sanitized NICE-PAKE lies within finding a secure mechanism to communicate the second (rerandomization) seed to the initiator, so that they can decpasualte correctly (if they are honest and know the correct password). That is, they must not be able to decpasualte if they are malicious or do not know the correct password, even if they can try all possible passwords in an offline dictionary attack. Similarly, a malicious receiver must not be able to craft a cipehertext that decpsualtes correctly under non-matching passwords, nor are they allowed to try all possible encaspualtions following an offline dictionary attack.

| **Alice** | | **Bob** |
|---|---|---|
| *Password* $\pi$ | | *Password* $\pi'$ |
| $sk_a, pk_a \leftarrow \mathsf{RKEM.KGen}(1^\kappa)$ | | |
| $(\rho_a, \mathbf{b}) \leftarrow \mathsf{Split}(pk_a)$ | | |
| $z_a \leftarrow \rho_a \oplus \mathcal{H}_1(\pi)$ | | |
| | $\xrightarrow{z_a, \mathbf{b}}$ | |
| | | $\rho'_a \leftarrow z_a \oplus \mathcal{H}_1(\pi')$ |
| | | $pk'_a \leftarrow \mathsf{Split}^{-1}(\rho'_a, \mathbf{b})$ |
| | | $\rho_b \leftarrow\$ \mathcal{X}_{seed}$ |
| | | $pk_b \leftarrow \mathsf{RKEM.RAND}(\rho_b, pk'_a)$ |
| | | $(C_b, K') \leftarrow \mathsf{RKEM.Encap}(pk_b)$ |
| | | $z_b \leftarrow \rho_b \oplus \mathcal{H}_2(\pi')$ |
| | $\xleftarrow{C_b, z_b}$ | |
| $\rho'_b \leftarrow z_b \oplus \mathcal{H}_2(\pi)$ | | |
| $K \leftarrow \mathsf{RKEM.Decap}(sk_a, C_b, \rho'_b)$ | | |

**Figure 3:** NICE-PAKE:RE using RKEM. Note that RKEM defines the output of Encap as $(s, C)$ where $s$ is a secret value and similarly for Decap as $p$, which are used later in a KDF to obtain a final key. For ease of exposition we assume the KEM key $K$ $(K')$ is obtained directly from both algorithms, as an FO KEM applies an internal KDF-like routine to output the KEM key.

### Informal Analysis

Intuitively, the passive security should be feasible to show in a straight forward manner. Hence we discuss the cases where either Alice or Bob are malicious. Note that implicit authentication means honest parties will not directly discover if they do not end up with matching final keys. that is, they can only find it out upon exchanging payload messages encrypted with whatever key the derived in the PAKE session. This implication is necessary to assess whether the proposed construction makes sense to begin with.

**Malicious Bob.** We start with Bob, as they cannot control the KEM.KGen routine. Clearly, a malicious Bob wants to either guess Alice's password directly, trick her into decpasulating the same KEM key through a malicious ciphertext, or guess her password later by trial decrypting exchanged payload messages based on an offline dictionary attack. Note also that Bob commits to one single password guess and one ciphertext in their message, which they can no longer change adaptively at any point of time in a given session. Assuming Bob guesses the password correctly from the start, it is clear they will obtain matching final keys with Alice; something which cannot be prevented by any means. Hence, we move on to see what happens when they guess wrong.

Upon a wrong password guess, Bob will obtain a seed s.t. $\rho_a \neq \rho'_a$ and thus end up with a different matrix s.t. $\mathbf{A} \neq \mathbf{A}'$. It follows, that any rerandomized key they might compute will use a different lattice matrix with probability $\frac{|\mathcal{D}|-1}{|\mathcal{D}|}$ where $\mathcal{D}$ is the password dictionary. Since Alice's $\mathbf{b}$-part is uniform under standard lattice assumptions, and assuming Bob cannot break search MLWE, they have no distinguishing advantage over the $\mathbf{A}$-part in $\mathbf{b}$, which was proven in [ABJŠ25, AASA+24, ABJ25b]. It follows, the addition of a uniform sample and any value is also indistinguishable from uniform over the same domain (here the MLWE ring). Meaning, regardless of how many $\mathbf{A}$-parts Bob may try, their advantage in distinguishing whether $pk_b$ is an MLWE sample should be no better than breaking search MLWE as well. Hence, we may assume that Bob has no distinguishing advantage in guessing Alice's password upon receiving the first message.

Bob settles for a random matrix and computes $pk_b$, or not, but in any case they have to commit to a ciphertext and a masked seed $z_b$. In any case, Alice will obtain $\rho_b$ using her password, and decapsualte under her own secret and the rerandomization seed she unmasked. Thus, Alice will decapsualte incorrectly with overwhelming probability, even if Bob had not sanitized to begin with. Additionally, any malicious ciphertext will not decapsualte correctly to the same key, since an FO MLWE KEM (e.g., ML-KEM) satisfies strong collision freeness (SCFR-CCA). The only remaining option for Bob is to try all possible rerandomizations with different $\mathbf{A}$-parts under all possible values for $\rho_b$. That is, to obtain the corresponding $pk_b$ for each password guess and encapsulate repeatedly to prepare a list of possible KEM keys. However, KEM decapsulation is deterministic, and by the time Bob receives an encrypted payload from Alice, she most probably had already decapsulated a random KEM key via the internal KEM FO, which is indistinguishable.

**Malicious Alice.** In this case, the $\mathbf{b}$-part can be chosen at will by Alice, where she is naturally not bound to compute it honestly, which makes the analysis rather tricky. Nevertheless, any malicious $\mathbf{b}$ will be sanitized on Bob's side, which renders Alice unable to decaspualte unless she obtains $\rho_b$ under the correct password. However, a challenge may arise in ensuring that any unmasking of the second seed $\rho_b$ yields a value suitable for sampling secrets from the pre-defined MLWE distribution.

Alice can prepare the password dictionary prior to a session, and compute all possible values for $\rho_b$ to try all possible decapsualtions. Upon decpsualting $C_b$, Alice can leverage the FO re-encryption step to check whether she decpsualted correctly. However, assuming that Bob encapsualted using a public key under a non-matching password, the ciphertext should never decrypt correctly. That is, since all possible permutations of $pk_b$ will not match to Alice's secret. Even if she tries all possible combination, she can no longer change her original KEM secret key and the lattice matrix that she committed to in $\mathbf{b}$. It follows that all possible decapsulations (and re-encryptions) will yield values indistinguishable form random.

Still, we note here that this assumption requires rigorous treatment, as the standard KEM anonymity property may not be directly applicable. As seen in the original NICE-PAKE [AASA+24], the authors formulated the notion of A-Part-Secrecy for Splittable KEMs (A-SEC-CCA), which was not sufficient to rule out attacks leveraging a (very) malicious **b** without additional measures. We hence assume that a similar property adapted to the rerandomized setting is required to prove security, which will probably incorporate the Hint-MLWE assumption used in the RKEM construction in [DFJ+25].

## 5 TEMPO:RE

In the following, we describe the modified TEMPO protocol (Fig. 4) utilizing the Rerandomized KEM (RKEM). Here, the task seems rather less tricky, since utilizing the 2F to program outputs rules out maliciously crafted public keys by Alice from ever being used as such by an honest Bob. Here we notice that we need to either slightly adjust RKEM to allow obtaining the original public key by simply subtracting the second one from the randomization, or adjust TEMPO to unsplit the randomized public key and use it in the encapsulation. However, these variants only differ syntactically and do not affect the correctness of the protocol or the KEM. In the following we assume the existence of an inverse routine $\mathsf{RKEM.RAND}^{-1}$ that recovers the original public key.

### Discussion

As previously mentioned, applying RKEM to TEMPO should intuitively work for two main reasons: 1) RKEM requires a public lattice matrix, which is clearly the case as TEMPO sends the matrix expansion seed $\rho$ in plain text. 2) RKEM.RAND only adds two public keys together as group elements, which is in line with the definition of 2F in NoIC and TEMPO. That is, instead of hashing the password with randomness and then applying a group operation on the public key, we use the randomness to sample fresh secrets and add them to the public key.

It follows that one can conduct a formal analysis on the protocol in a modular way assuming that 2F fulfills the an ideal functionality as in [JRX25], or conduct the proof in white-box manner as in [ABJ25a]. Still, it remains to show whether the UC simulator is affected by this modification, and if password extraction is still as feasible as in the original proof(s). That being said, it appears that the required KEM properties will not be affected. One concern could arise regarding the uniformity of ML-KEM public keys. However, since addition in a ring (and its inverse) are defined, any resulting public key should be indistinguishable from uniform regardless of the password guess.

It remains to check whether a UC simulator can still extract adversarial password guesses from the 2F. Most probably, a monolithic proof as in [ABJ25a] will have to treat both the 2F and the RKEM routines as a white-box in order to determine the password used to program the output. On the other hand, a modular proof as in [JRX25] requires defining and ideal functionality, which seems rather infeasible for a KEM scheme. Hence, one would need to make use of the rerandomization mechanism in a way that is detached from the overall KEM, which might lead to the need to prove correctness and security again for this specific step. Apart from that, we believe that the rest of the formal analysis should not be affected by this modification is expected to proceed similar to the orignal one.

```
Alice                                   Password $\pi$                                   Bob
$(sk, pk) \leftarrow \mathsf{RKEM.KGen}(1^\kappa)$
$(t, \rho) \leftarrow \mathsf{Split}(pk)$
$r \leftarrow\!\!\$\ \{0,1\}^{|\kappa|}$
$v \leftarrow \mathsf{RKEM.RAND}(t, r, \rho)$
$\hat{v} \leftarrow \mathcal{H}(\pi, \rho, v)$
$u \leftarrow \hat{v} \oplus r$
$apk \leftarrow (u, v, \rho)$
                                        $\xrightarrow{apk}$
                                                                  $(u, v, \rho) \leftarrow apk$
                                                                  $\hat{v} \leftarrow \mathcal{H}(\pi, \rho, v)$
                                                                  $r \leftarrow \hat{v} \oplus u$
                                                                  $t \leftarrow \mathsf{RKEM.RAND}^{-1}(v, r, \rho)$
                                                                  $pk \leftarrow \mathsf{Split}^{-1}(t, \rho)$
                                                                  $(c, K) \leftarrow \mathsf{RKEM.Encap}(pk)$
                                                                  $\mathsf{tag} \leftarrow \mathcal{H}_{\mathsf{tag}}(\pi, pk, apk, c, K)$
                                        $\xleftarrow{c, \mathsf{tag}}$
$K \leftarrow \mathsf{RKEM.Decap}(sk, c)$
if $\mathsf{tag} \neq \mathcal{H}_{\mathsf{tag}}(\pi, pk, apk, c, K)$
    $\mathsf{key} \leftarrow\!\!\$\ \{0,1\}^\kappa$
else
    $\mathsf{key} \leftarrow \mathcal{H}_{\mathsf{key}}(\pi, pk, apk, c, K)$      $\mathsf{key} \leftarrow \mathcal{H}_{\mathsf{key}}(\pi, pk, apk, c, K)$
return key                                                        return key
```

**Figure 4:** TEMPO:RE using RKEM. Note that the variant in the figure defines RKEM.RAND to take input the matrix expansion seed $\rho$ and use it internally to build the sanitizing public key to obtain a randomized public key $v$.

We can thus conclude that rerandomizing the KEM public key in TEMPO may very well yield a concrete ML-KEM instantiation without resorting to obfuscation. It is also very likely that the same applies to NoIC-PAKE, even when the seed is not sent in plain text as observed in NICE-PAKE. If we suppose that this construction is secure, we still cannot estimate the performance gain (or loss) yet. Nevertheless, we believe that avoiding the reduction in the public key space is a desired outcome on its own.

## Acknowledgments

We would like to express our thanks to Jintai Ding, Manuel Barbosa, Shuichi Katsumata and Vadim Lyubashevsky for their useful input and discussions. This research work has been partly funded by the German Federal Ministry of Education and Research and the Hessian State Ministry for Higher Education, Research and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE.

## References

[AASA$^+$24] Nouri Alnahawi, Jacob Alperin-Sheriff, Daniel Apon, Gareth T. Davies, and Alexander Wiesmaier. NICE-PAKE: On the security of KEM-based

PAKE constructions without ideal ciphers. Cryptology ePrint Archive, Paper 2024/1957, 2024.

[ABJ25a] Afonso Arriaga, Manuel Barbosa, and Stanislaw Jarecki. NoIC: PAKE from KEM without ideal ciphers. Cryptology ePrint Archive, Paper 2025/231, 2025.

[ABJ25b] Afonso Arriaga, Manuel Barbosa, and Stanislaw Jarecki. Tempo: ML-KEM to PAKE compiler resilient to timing attacks. Cryptology ePrint Archive, Paper 2025/1399, 2025.

[ABJŠ25] Afonso Arriaga, Manuel Barbosa, Stanislaw Jarecki, and Marjan Škrobot. C'est très chic: A compact password-authenticated key exchange from lattice-based kem, 2025.

[AHHR24] Nouri Alnahawi, Kathrin Hövelmanns, Andreas Hülsing, and Silvia Ritsch. Towards post-quantum secure pake - a tight security proof for ocake in the bpr model. In *Cryptology and Network Security*. Springer Nature Singapore, 2024.

[AHMW25] Nouri Alnahawi, David Haas, Erik Mauß, and Alexander Wiesmaier. SoK: PQC PAKEs - cryptographic primitives, design and security. Cryptology ePrint Archive, 2025/119, 2025.

[BCP+23] Hugo Beguinet, Céline Chevalier, David Pointcheval, Thomas Ricosset, and Mélissa Rossi. Get a cake: Generic transformations from key encaspulation mechanisms to password authenticated key exchanges. In *ACNS 2023, Kyoto, Japan, June 19–22, 2023, Proceedings, Part II*. Springer-Verlag, 2023.

[Bel92] Bellovin, S.M. and Merritt, M. Encrypted key exchange: password-based protocols secure against dictionary attacks. In *Proceedings 1992 IEEE Computer Society Symposium on Research in Security and Privacy*, 1992.

[Bos18] Bos, Joppe and Ducas, Leo and Kiltz, Eike and Lepoint, T and Lyubashevsky, Vadim and Schanck, John M. and Schwabe, Peter and Seiler, Gregor and Stehle, Damien. CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2018.

[DFJ+25] Kévin Duverger, Pierre-Alain Fouque, Charlie Jacomme, Guilhem Niot, and Cristina Onete. Subversion-resilient key-exchange in the post-quantum world. In *CCS 2025-32nd ACM Conference on Computer and Communications Security*, 2025.

[GRSV25] Felix Günther, Michael Rosenberg, Douglas Stebila, and Shannon Veitch. Hybrid obfuscated key exchange and kems. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025*, Cham, 2025. Springer Nature Switzerland.

[GSV24] Felix Günther, Douglas Stebila, and Shannon Veitch. Obfuscated key exchange. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, New York, NY, USA, 2024. Association for Computing Machinery. doi:10.1145/3658644.3690220.

[HHKR25] Kathrin Hövelmanns, Andreas Hülsing, Mikhail Kudinov, and Silvia Ritsch. CAKE requires programming - on the provable post-quantum security of (o)CAKE. Cryptology ePrint Archive, Paper 2025/458, 2025.

[JRX25] Jake Januzelli, Lawrence Roy, and Jiayu Xu. Under what conditions is encrypted key exchange actually secure? In *Advances in Cryptology – EUROCRYPT 2025*, pages 451–481, Cham, 2025. Springer Nature Switzerland.

[MRR20] Ian McQuoid, Mike Rosulek, and Lawrence Roy. Minimal symmetric pake and 1-out-of-n ot from programmable-once public functions. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020.

[VJW$^+$25] Jelle Vos, Stanislaw Jarecki, Christopher A. Wood, Cathie Yun, Steve Myers, and Yannick Sierra. A hybrid asymmetric password-authenticated key exchange in the random oracle model. Cryptology ePrint Archive, Paper 2025/1343, 2025. URL: https://eprint.iacr.org/2025/1343.