Abstract

Private payments in blockchain-based cryptocurrencies have been a topic of research, both academic and industrial, ever since the advent of Bitcoin. Stealth address payments were proposed as a solution to improve payment privacy for users and are, in fact, deployed in several major cryptocurrencies today. The mechanism lets users receive payments so that none of these payments are linkable to each other or the recipient. Currently known stealth address mechanisms either (1) are insecure in certain reasonable adversarial models, (2) are inefficient in practice or (3) are incompatible with many existing currencies.

In this work, we formalize the underlying cryptographic abstraction of this mechanism, namely, stealth signatures with formal game-based definitions. We show a surprising application of our notions to passwordless authentication defined in the Fast IDentity Online (FIDO) standard. We then present SPIRIT, the first efficient post-quantum secure stealth signature construction based on the NIST standardized signature and key-encapsulation schemes, Dilithium and Kyber. The basic form of SPIRIT is only secure in a weak security model, but we provide an efficiency-preserving and generic transform, which boosts the security of SPIRIT to guarantee the strongest security notion defined in this work. Compared to state-of-the-art, there is an approximately 800x improvement on the signature size while keeping signing and verification as efficient as 0.2 ms.

We extend SPIRIT with a fuzzy tracking functionality where recipients can outsource the tracking of incoming transactions to a tracking server, satisfying an anonymity notion similar to that of fuzzy message detection (FMD) recently introduced in [CCS 2021]. We also extend SPIRIT with a new fuzzy tracking framework called scalable fuzzy tracking that we introduce in this work. This new framework can be considered as a dual of FMD, in that it reduces the tracking server’s computational workload to sublinear in the number of users, as opposed to linear in FMD. Experimental results show that, for millions of users, the server only needs 3.4 ms to filter each incoming message which is a significant improvement upon the state-of-the-art.
1 Introduction

Cryptocurrencies provide support for trustless and publicly verifiable payments. The sender of a payment posts a transaction onto a public ledger called blockchain. In the most basic form, the transaction specifies the sender and receiver’s respective public keys (or addresses), and the transaction is authorized by the sender via a digital signature wrt. their public key. E-commerce [eco],
donation platforms [donc, dona, donb], gaming platforms [ega], etc., are just some of the popular use cases that are enabled by cryptocurrencies and their trustless payments. For example, donation platforms accept donations in the form of cryptocurrency payments, and to do this, a donation platform announces its addresses and users can make transactions paying to these addresses without requiring permission from any authority.

A critical weakness of the above paradigm is that it lacks reliable anonymity guarantees in its basic form. Several de-anonymisation techniques [OKH13, SO13, RH13, RS13, MSH+17] for law enforcement purposes have been demonstrated that link addresses on the blockchain to the real-world entities that own them. However, it has also led to questionable forms of censorship [frc] of users and their payments.

A mechanism known as stealth addresses [ste, vS, Tod, CM17] was developed to address these anonymity issues. For instance, the donation platform publishes a single master address, a so-called stealth address, and any user can send donations to the platform, by using a locally re-randomized version of the stealth address called one-time address. Such a one-time address is unlinkable to the stealth address for any outside observer, consequently, transactions to such a stealth address look as if they are going to random recipients (and not necessarily the donation platform). Also, with access to its master secret, the donation platform can link such a one-time address to its stealth address and further generate the corresponding one-time secret locally, on the fly. Using this one-time secret, the coins associated with the one-time address can be spent. In this case, the recipient only needs to publish its master address, and does not need to give out fresh unlinkable addresses for each potential sender. As the number of senders could well be in the hundreds or thousands (as is the case with e-commerce, donations, etc.), this mechanism leads to a scalable solution.

The stealth address scheme proposed in [vS] has in fact been deployed in many of the major currencies like Bitcoin [ste], Ethereum [umb], and Monero [vS]. The mechanism has further found direct application in privacy enhancement of payment protocols like Blitz [AMKM21]. As Monero implements stealth addresses via signature schemes, we will refer to the cryptographic abstraction of the mechanism from [vS] as stealth signatures. Thus we will henceforth use the terms addresses and public keys interchangeably.

Recent academic works [LYW+19, LLN+20] initiated the formal treatment of stealth signatures and observed that the construction of [vS] does not satisfy security under so-called key-exposures. Roughly, this means that if an adversary learns the corresponding one-time secret key for the one-time public keys that he generated, then he can learn all one-time secret keys of all one-time public keys that he generates for this particular master address.

More recent proposals of stealth signature schemes [LYW+19, LLN+20] were designed to be secure against such key-exposure attacks, with the downside that their schemes use heavy tools such as pairings [BF01] or lattice-basis-delegation [ABB10]. These are currently not compatible with any of the major cryptocurrencies that exist today. Furthermore, with the threat of quantum computers looming large, cryptocurrency payments including the pre-quantum stealth signature mechanisms of [vS, LYW+19] remain vulnerable. While a lattice-based (and thus plausibly post-quantum) construction of stealth signatures was proposed in [LLN+20], this construction relies on the aforementioned lattice basis delegation. Consequently, their scheme is most likely too inefficient for practical use\(^1\). We compare our constructions and related works in Table 1. Please refer to Appendix B for more discussion.

This work is motivated by the following two questions:

- **Can we have an efficient stealth signature scheme with security against unbounded key-exposures**,\(^1\)

---

\(^1\)As the authors of [LLN+20] point out in Section 1.1, their “public key and signature sizes are too large for practical use”.
Table 1: Comparison with Prior Works about Stealth Signatures

<table>
<thead>
<tr>
<th>Works</th>
<th>Security</th>
<th>Post-quantum</th>
<th>Size</th>
<th>Signature Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monero’s SS [vS]</td>
<td>o</td>
<td>EUFCMA</td>
<td>64 B</td>
<td>64 B</td>
</tr>
<tr>
<td>Paring-based SS [LYW+19]</td>
<td>•</td>
<td>EUFCMA</td>
<td>231 B</td>
<td>115 B</td>
</tr>
<tr>
<td>ABB10-based SS [LLN+20]</td>
<td>•</td>
<td>EUFCMA</td>
<td>3.35 GB</td>
<td>3.26 MB</td>
</tr>
<tr>
<td>[LLN+20] + NTRU (potential optimization)</td>
<td>•</td>
<td>EUFCMA</td>
<td>13.82 KB</td>
<td>13.82 KB</td>
</tr>
<tr>
<td>Appendix D</td>
<td>• 2</td>
<td>EUFCMA</td>
<td>96 B</td>
<td>64 B</td>
</tr>
<tr>
<td>Section 6.1 + Dilithium (compiler from Section 5)</td>
<td>•</td>
<td>EUFCMA</td>
<td>2.08 KB</td>
<td>2.54 KB</td>
</tr>
<tr>
<td>Section 6.1 + Falcon (compiler from Section 5)</td>
<td>•</td>
<td>EUFCMA</td>
<td>2.08 KB</td>
<td>4.09 KB</td>
</tr>
</tbody>
</table>

1 Secure against key-exposures. Our construction presented in Section 6.1 can be upgraded to w/KE according to Section 5.
2 Secure against bounded key-exposures.

Table 2: Comparison with Prior Works about Fuzzy/Private Tracking

<table>
<thead>
<tr>
<th>Works</th>
<th>Privacy</th>
<th>Assumptions</th>
<th>Post-quantum</th>
<th>Server’s Work</th>
<th>Latency/msg(^2)</th>
<th>Receiver’s Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMD(_2) [BLMG21]</td>
<td>(\rho N)-anonymity(^3)</td>
<td>Random Oracle</td>
<td>o</td>
<td>O((N))</td>
<td>933 sec</td>
<td>37.5 ms</td>
</tr>
<tr>
<td>H(_\text{TEE}) [MSS+21]</td>
<td>Full Privacy</td>
<td>Trusted Execute Environment</td>
<td>o</td>
<td>O((N))</td>
<td>228 sec</td>
<td>12 ms</td>
</tr>
<tr>
<td>H(_\text{TEE}) [MSS+21]</td>
<td>Full Privacy</td>
<td>Two Non-colluding Servers</td>
<td>o</td>
<td>O((N))</td>
<td>81.1 hour</td>
<td>1 ms</td>
</tr>
<tr>
<td>OMR(_\text{p2}) [LT21]</td>
<td>Full Privacy</td>
<td>Fully Homomorphic Encryption</td>
<td>o</td>
<td>O((N))</td>
<td>43.1 hour</td>
<td>63 ms</td>
</tr>
<tr>
<td>Section 6.2</td>
<td>(\rho N)-anonymity</td>
<td>Standard Model</td>
<td>o</td>
<td>O((N))</td>
<td>11.70 sec</td>
<td>37.5 ms</td>
</tr>
<tr>
<td>Section 6.3</td>
<td>(\rho N)-anonymity</td>
<td>Random Oracle</td>
<td>o</td>
<td>O((\rho N))</td>
<td>3.42 ms</td>
<td>37.5 ms</td>
</tr>
</tbody>
</table>

1 \(\rho\) denotes the false-positive rate and \(N\) the number of clients.
2 Calculated in a setting with \(N = 2^{128}\) users and \(M = 500,000\) messages based on the numbers from their papers.
3 Latency per message induced by the server. See more discussion in Appendix B.

that is compatible with Schnorr, ECDSA and other group based signature schemes predominantly used in currencies today?

- Can we have an efficient stealth signature scheme secure with unbounded key exposure that is post-quantum secure?

A caveat of the stealth address mechanism is that a recipient (online or offline) has to parse through a large number (hundreds of thousands per day) of transactions to identify those that send coins to one-time addresses corresponding to his master address. A workaround was proposed in [vS], where a recipient can delegate identification of incoming payments to a semi-trusted third party server called the tracking server. To do so, the recipient can generate a so-called tracking key from his secret key and provide it to the tracking server. The tracking key allows the tracking server to identify or track all incoming payments to the recipient using the tracking key, and later notify the recipient of these exact payments. On the other hand, such a tracking key should not enable the tracking server to generate one-time secrets for the concerned one-time addresses. Prior works [ADE+20, LYW+19, LLN+20] omit this important tracking functionality in their formalization of stealth signatures.

A downside of the above tracking method is that we fully give up anonymity unlinkability with regards to the tracking server who learns exactly which payments are addressed to the recipient. While there is a natural and obvious tension between the anonymity goal of unlinkability and functional goal of trackability, a recent work of Beck et al. [BLMG21] attempts to strike a balance between these notions. They introduce the concept of fuzzy message detection (FMD), where a tracking server can approximately detect messages meant for a recipient with adjustable degree of uncertainty. More
specifically, their notion of detection is fuzzy in the sense that messages meant for the recipient are always correctly identified, but there is recipient-controlled false positive rate (baked into the fuzzy tracking key) which causes messages meant for other users to be misclassified as being meant for the recipient. Thus, the tracking server cannot decide with certainty if a detected message is actually intended for the recipient or not. This mechanism makes it necessary for the sender of the message to include additional fuzzy tracking information and the tracking server possesses a fuzzy tracking key. In principal, applying their technique to enable fuzzy tracking of one-time addresses in stealth signatures is straightforward. However, relying on their schemes comes with considerable drawbacks. While their first scheme (FMD$^2$) is efficient, it relies on the pre-quantum DDH assumption. Their second scheme (FMD$_{frac}$) relies on heavy tools like garbled circuits that lead to an unacceptable size-blowup of the sender’s message. On the other hand, there are signalling detection or retrieval schemes [MSS$^++$21, LT21] for fully private tracking instead of fuzzy tracking, but all of them require linear work at the server side which doesn’t scale to thousands or millions of users. We discuss their schemes and ours in Appendix B and present a comparison in Table 2. This leads us to ask:

- Can we have a stealth signature scheme with efficient fuzzy tracking in the post-quantum setting and scalable to hundreds of thousands (or even millions) of users?

1.1 Our Contributions

We summarize our contributions below.

**Modular Framework.** We introduce SPIRIT (in Section 6.1), the first practically efficient post-quantum stealth signature scheme secure without key-exposure. Towards this goal, we consider the lattice-based Dilithium [LDK$^+20$] signature scheme, which is the winner of the NIST standardization competition and most likely candidate to be adopted into cryptocurrencies. Without changing the signature scheme in any way, we augment Dilithium with additional algorithms to obtain SPIRIT so that it now supports one-time key derivations and tracking.

Next, we show how one can generically transform (in Section 5) a stealth signature scheme that is secure without key-exposure into a scheme that is secure with unbounded key-exposure. Thus we can upgrade SPIRIT into one that is practically efficient and secure with unbounded key-exposure. Both SPIRIT and its upgrade are compatible with cryptocurrencies that would support Dilithium signature verification and no additional scripting is required.

Furthermore, we construct a stealth signature scheme (in Appendix D) that is compatible with group-based schemes like Schnorr, and ECDSA which are used in most of the currencies today. However, it only guarantees security with bounded key-exposure: It tolerates an a-priori number of one-time secret key leakage.

**Fuzzy Constructions.** We then present two fuzzy stealth signature schemes (using SPIRIT), both of which are the first efficient and post-quantum candidates.

In the first construction (in Section 6.2), we take a similar approach as FMD from [BLMG21]. But we reduce its overhead from $O(\lambda)$ to 1 bit per signal by making novel use of ciphertext compression techniques [BDGM19]. Additionally, we show how to allow finer false-positive rates without requiring heavy tools like garbled circuits as in [BLMG21].

We then present a new scalable framework for fuzzy tracking (in Section 4.4) followed by an efficient construction (in Section 6.3) in the random oracle model. This framework can be viewed as a ‘dual’ version of FMD mechanism from [BLMG21]. Intuitively, it is a trade-off between efficiency.

---

2A recent work by [ADE$^+20$] presents the construction of a re-randomized signature, which bears resemblance to the concept of stealth addresses. However, it is important to note that their proposed functionality does not offer public tracking support and is not secure against key-exposure attacks.
and usability: By limiting the users’ ability to choose false-positive rates, we are able to reduce the tracking server’s computational work to an amount which is sublinear in the total number of users. This compares very favourably with prior works, where the server needs to take a linear scan of each user’s tracking key [BLMG21, MSS+21, LT21].

**Implementation.** We implemented SPiRT, post-quantum FMD, and scalable fuzzy tracking based on Dilithium, Kyber, and Falcon with anonymized open-source code [imp]. We test them with different parameter sets on an ordinary laptop as presented in Table 3 and Table 4 (in Appendix B). Experiment results show that our stealth signature with strongest security only yields a 4.09 KB signature, while the verification time is less than 0.2 ms. Similarly, our scalable fuzzy tracking mechanism only takes 3.42 ms to filter each incoming message in the setting with millions of users.

**Application to FIDO.** As our final contribution, we surprisingly apply our stealth address notion to FIDO2 standard passwordless authentication schemes (formally defined in [BBCW21]). We show how manufacturers can implement device authenticators that not only provide post-quantum security but require limited secure memory (one master secret key), support global revocation (as defined in [HLW22]), multi-device credentials [All22], and can be used to implement asynchronous remote key generation ([FGK+20]).

## 2 Technical Overview

Let us first recall the group-based stealth signature scheme of [vS]: Given a cryptographic group $G = \langle g \rangle$ of prime order $p$, the master public key is $mpk := (g, b_0 := g^a, h_1 := g^b) \in G^3$, where $msk := (a \leftarrow Z_p, b \leftarrow Z_p)$ is the master secret key, and $mtk := a$ is the tracking key. To re-randomize $mpk$ to a one-time address (i.e., one-time public key), the sender samples a uniformly random $r \leftarrow Z_p$ and computes $opk := g^{h_1} \cdot h_1 \in G$ where $H: G \mapsto Z_p$ is a hash function modelled as a random oracle. Additionally, the sender attaches tracking information $tki := g^r \in G$ to the $opk$. To derive the corresponding one-time secret key $osk$ from $msk$, the receiver computes $osk := H(tki^r) + b \in Z_p$ with the help of $tki$. Now, $opk$ and $osk$ satisfy the discrete-log relation $opk = g^{osk}$, hence the receiver can sign (for e.g., Schnorr or ECDSA) any message with $osk$ to output a signature which can be verified with corresponding $opk$. An additional mechanism is that $mtk := a$ can be given to a tracking server for tracking: By comparing whether $opk = g^{H(tki^msk)} \cdot h_1$, the tracking server can determine whether $opk$ links to the issuer of $mtk$.

Taking a closer look, this approach to build stealth signature apparently can be generically decomposed to a linearly homomorphic one-way function $f : \mathcal{D} \mapsto \mathcal{M}$ where $f(x + y) = f(x) + f(y)$, and a key-exchange protocol $(KE_1, KE_2, KE_3)$, where $KE_i$ denotes the $i$-th message function:

$$c_{t1} \in C_1 \triangleleft KE_1(r_1),$$

$$(c_{t2} \in C_2, K \in K) \triangleleft KE_2(r_2, c_{t1}),$$

$$K \in K \triangleleft KE_3(r_1, c_{t2}),$$

where $r_1, r_2$ are two user’s secrets, and $K$ is the agreed-upon key. Here, $C_1, C_2$, and $K$ represent the first message space, second message space, and the key space, respectively. Now, let $mpk := (c_{t1} := KE_1(r_1), B := f(b))$ and $msk := (r_1, b), mtk := r_1$. To publish a one-time address, the sender can just compute $(c_{t2}, K) \triangleleft KE_2(r_2, c_{t1})$ and publish

$$opk := B + f(H(K)), tki := c_{t2}$$

where $H: K \mapsto \mathcal{D}$. Correspondingly,

$$osk := b + H(KE_3(r_1, tki)).$$
Since they obey the relation \( f(\text{osk}) = \text{opk} \), we can leverage this to sign and verify. The tracking mechanism still works by checking if
\[
\text{opk} = f\left( H(\text{KE}_3(\text{mtk}, \text{tki})) \right) + B.
\]
We will now adapt this blueprint to construct a stealth signature in lattice setting.

### 2.1 Spirit: Lattice-based Stealth Signature

To make our protocol both efficient and practical, we would like to use optimized NIST winners as our building blocks. In this work, we choose Dilithium as the underlying digital signature considering that it is one of most popular signature schemes in NIST [LDK+20]. We call the resulting stealth signature scheme \textit{Spirit}. Basically, it follows the above approach: In Dilithium, the public key is a Module Learning With Errors (MLWE) \[ \text{BGV12} \] sample \( t := As_1 + s_2 \) where its secret-error pair \( (s_1, s_2) \) (both chosen from a suitable short distribution) acts as the secret key. Since MLWE involves only linear operations, we have that
\[
t + t' = A(s_1 + s'_1) + s_2 + s'_2.
\]

Yet, even though adding samples is approximately linearly homomorphic, this addition will increase error rates or lengths for both \( s_1 \) and \( s_2 \). Typically, the \( s_1 \) and \( s_2 \) are generated by sampling their coefficients uniformly with absolute value at most \( \eta \) (for some small parameter \( \eta \)). The increased norm of the new secrets \( (s_1 + s'_1, s_2 + s'_2) \) will incur additional running time during signing due to the so-called “Fiat-Shamir with Abort” mechanism of Dilithium. To alleviate this issue, we only prove \textit{Spirit} to be existential unforgeable. This will give us better parameters to balance between security and efficiency. Looking ahead, we point out that \textit{Spirit} can be transformed to a strongly existentially unforgeable scheme using a generic compiler which we will introduce later.

Apart from a linearly homomorphic one-way functions, we still need a key-exchange protocol. However, this key-exchange needs some additional properties. Specifically, we need a non-interactive key-exchange (NIKE) protocol which is substantially stronger than \textit{KE} we depicted above. The starting point is that it needs to be \textit{anonymous} under chosen plaintext attacks (CPA), which means given the message \( \text{ct}_2 \), the adversary cannot link it to the \( \text{ct}_1 \) used to generate \( \text{ct}_2 \). This is for stealth signatures as we don’t want our one-time address to be linkable to the original master public address. This security notion is formalized as \textit{unlinkability}.

But anonymity under chosen plaintext attacks will not even suffice yet for our applications. We will require a stronger notion of anonymity under plaintext checking attacks (PCA). Here, the adversary is given an additional oracle which allows him to check whether a ciphertext-plaintext pair is valid or not. To see why this is necessary, consider an adversary who is trying to link some \( (\text{opk}, \text{tki}) \) to \( \text{mpk} \). Such an adversary will be able to sample \( \text{ct}_2 \leftarrow \text{C}_2, K \leftarrow \text{K} \) to generate \( (\text{opk}', \text{tki}') \), which can then be published to see if the tracking check passes. It turns out that anonymity under plaintext checking attacks is sufficient for this setting. However, we currently don’t have a simple construction satisfying anonymity under plaintext checking attacks. As a consequence, we use an even stronger key-exchange protocol which is anonymous under chosen ciphertext attacks (CCA), namely, it is ANOCCA-secure (formalized in Definition C.7). Fortunately, the recent standardized KEM by NIST, Kyber [SAB+20], can be slightly modified to be ANOCCA-secure [GMP22] and we use Kyber in the concrete instantiation. There are multiple technical details not covered in this outline, for instance, besides \( \text{tki} \), the one-time address \( \text{opk} \) itself also needs to be anonymous. We refer to Section 6.1 for detailed construction and analysis.
So far, we briefly mentioned two important security notions for stealth signatures, namely unforgeability and unlinkability (Section 4 for formalization). However, we note that we only formalize these two notions as unforgeability without key-exposure and unlinkability without key-exposure, respectively. It turns out the above approach to build stealth signatures (as well as in SPIRIT) is no longer secure if a one-time secret key $\text{osk}$ leaks: Suppose the sender learns $\text{osk}$ somehow, he can instantly recover $\text{msk}$ as

$$b := \text{osk} - H(\text{KE}_3(r_1, \text{tki})),$$

if he knows $r_1$ which is used to generate corresponding $\text{opk}$.

### 2.2 Generic Transformation: Security with Key-exposure

As mentioned above and noticed in prior works [LYW+19, NMRL16], leaking one-time secret keys is almost as bad as leaking the master secret key. This is a potential issue in current practical stealth signature schemes [vS] and it is costly to avoid. For instance, if we are willing to use techniques implying hierarchical identity based encryption (HIBE), we could have a stealth signature scheme secure with key-exposure attacks by using pairing [BF01, LYW+19], lattice basis delegation [ABB10, LLN+20], or non-black box tools [DG17]. All of above techniques are several orders of magnitude slower in computational time, or orders of magnitude larger in signature or one-time public key size.

The reason we don’t have a simple solution to this issue is that one-time secret keys are usually a linear function of the $\text{msk}$ as mentioned in [LRR+19]. Apparently we can achieve security with bounded key-exposure by adding more secrets in $\text{msk}$ where bounded key-exposure means $\text{msk}$ remains secure if the number of leaked $\text{osk}$ is smaller than some ‘a priori bound’ and we show a candidate construction in Appendix D. However, any generic-group based techniques to prevent unbounded key-exposure should imply IBE which is known to be impossible using only black-box techniques [PRV12, SGS21].

In this work, we provide a conceptually simple, generic, and powerful black-box compiler to tackle this problem in the context of stealth signatures (in Section 5): We use a short chain of signatures [Mer90] to compile any stealth signature $\text{SS}_{w/o}$ secure without key-exposure into a strong stealth signature $\text{SS}_w$ secure with unbounded key-exposure. The high level idea is to break this ‘linear’ relation between $\text{osk}$ and $\text{msk}$. Specifically, instead of generating $\text{osk}$ directly, with the help of an additional digital signature $\text{DS}$, we generate

$$\text{osk} := (\sigma_1, \text{sk}, \text{vk}),$$

where $\sigma_1 \leftarrow \text{SS}_{w/o}.\text{Sign}(\text{osk}', \text{vk})$ and $(\text{vk}, \text{sk}) \leftarrow \text{DS.\text{Gen}(\lambda)}$. Note that $\text{osk}'$ is the one-time secret key in the scheme $\text{SS}_{w/o}$. Intuitively, since $\text{osk}$ has a non-linear relation with $\text{msk}$, the adversary cannot recover $\text{msk}$ from $\text{osk}$ as $\text{SS}_{w/o}$ is unforgeable. To sign a message $m$, it runs $\sigma_2 \leftarrow \text{DS.\text{Sign}(sk, m)}$ and outputs the final signature $\sigma := (\sigma_1, \sigma_2, \text{vk})$. Similarly, to verify $\sigma$ just use $\text{opk}$ to verify the signature $\sigma_1$ on $\text{vk}$ and use $\text{vk}$ to verify the signature $\sigma_2$ on $m$. Compared to original stealth signature $\text{SS}_{w/o}$, our compiled one $\text{SS}_w$ incurs slightly larger signature size and longer verification time, but in turn is far more efficient than above HIBE-related techniques.

Additionally, we show this compiler can also leverage $\text{SS}_{w/o}$ with existential unforgeability to $\text{SS}_w$ with strong unforgeability via a small tweak: Instead of signing on $m$, we sign as $\sigma_2 \leftarrow \text{DS.\text{Sign}(sk, m||\sigma_1)}$. This prevents strong unforgeability attacks of $\text{SS}_w$ because: Assuming $\text{vk}$ in $\sigma$ is not altered, a different $\sigma'_1 \neq \sigma_1$ will lead to a forgery $(m||\sigma'_1, \sigma_2)$ of $\text{DS}$ in $\text{SS}_w$. Therefore, SPIRIT can also be leveraged in this way to be strongly unforgeable with key-exposure. This gives us the first practical post-quantum $\text{SS}_w$ secure with key-exposure.

8
2.3 Fuzzy Tracking

We will now turn to the issue that in the above constructions, the tracking mechanism will leak the users' metadata to the tracking servers, i.e., the tracking server will know exactly which $\texttt{mtk}$ belongs to which specific $(\texttt{opk}, \texttt{tki})$. As discussed above, to address this problem, Beck et al. [BLMG21] proposed a mechanism named fuzzy message detection (FMD): The server is given a fuzzy tracking key $\texttt{ftk}$ instead of $\texttt{mtk}$ to filter incoming fuzzy tracking information $\texttt{ftki}$ for its users. Here, $\texttt{ftki}$ is attached with $(\texttt{opk}, \texttt{tki})$. Specifically, for unmatched $\texttt{ftki}$ and $\texttt{ftk}$, they will be linked with probability roughly $\rho$.

Transforming their scheme to post-quantum world is non-trivial as there are still two potential obstacles in the lattice setting: First, it is not practically efficient since its $\texttt{ftki}$ is as large as $O(n \cdot |\texttt{ct}|)$-bit where $|\texttt{ct}| = \text{poly}(\lambda)$. This is highly undesirable in practice as our expectation is something like $O(\lambda) + n$. The other problem is the uniformly-ambiguous (recalled in Appendix C) encryption, as it is unclear how to extend the random oracle based approach in [BLMG21], to the lattice setting due to the presence of noise. We show that these two obstacles are related and can be resolved simultaneously. For simplicity, assume $n = 1$ for the moment. Recall that in Regev encryption with modulus $q$, the ciphertext is composed of two parts, a vector $\texttt{c}_1 \in \mathbb{Z}_q^\ell$ and a scalar $c_2 \in \mathbb{Z}_q$. The secret key is $s \in \mathbb{Z}_q^\ell$ and decryption consists of rounding after a linear operation:

$$\left\lfloor s^T \texttt{c}_1 - c_2 \right\rfloor_2 = \left\lfloor \frac{q}{2} \cdot m + e \right\rfloor_2,$$

where $e < B < \frac{q}{2}$ is a bounded error. This is not just bad for efficiency (as we need additional $n \log q$ bits to encrypt $n$ more bits), but also for security: With the correct secret key $s$, $s^T \texttt{c}_1 - c_2$ is distributed as a Gaussian around $\frac{q}{2}$ or 0; With a wrong key $s^*$, $s^* \texttt{c}_1 - c_2$ is distributed uniformly random over the entire domain $\mathbb{Z}_q$. These two cases are clearly distinguishable by an adversary.

Our solution will be to compress $c_2$ into a single bit, which doesn’t convey enough information about the distribution. Hence this idea will solve both of the above problems simultaneously. Brakerski et al. [BDGM19] introduced rate-1 packed Regev encryption which can compress each $c_2$ to just one bit but require an additional offset scalar $z \in \mathbb{Z}_q$ in the header. Thus to to encrypt $n$ bits, the ciphertext after compression is $(\texttt{c}_1, z, w_1, \ldots, w_n)$ where $w_i \in \{0, 1\}$. To make the offset $z$ statistically close to uniformly random (in our setting pseudorandom doesn’t suffice because the adversary gets the secret key), we require super-polynomial noise-modulus ratio of Learning With Errors (LWE) [Reg05] which makes the scheme slightly less efficient. This gives us a lattice-based fuzzy tracking scheme (and ambiguous encryption), and surprisingly, it doesn’t rely on heuristic assumptions like random oracles which are necessary in [BLMG21].

2.4 Scalable Fuzzy Tracking

We observe that in the above FMD style tracking, the server’s computational work is $O(N)$ with $N$ users and is not scalable when thousands (or millions) of users are using the service of the server. We provide a framework for scalable fuzzy tracking which we view as a dual version of FMD [BLMG21], where the server’s work is sublinear. In this framework, we weaken the requirement that the false-positive rate can be adaptively changed by users. Instead, it is fixed in advance in this setting. This weakening is reasonable as it was shown in [SPB21] that an adversary can mount statistical attacks if users have varying false positive rates. To circumvent such attacks it was suggested that all users have high enough false positivity rates as even a small subset of low rate users can affect unlinkability for the entire pool of users. Therefore we can fix the false positivity rate to be a high enough value for everyone. For example, as calculated in [SPB21], the false-positive rate $\rho$ is better to be as large
as \( \frac{1}{\sqrt{N}} \). In this case, we can make the server’s overhead \( O(\rho N) \) for each incoming message which was at least \( O(N) \) in prior works [BLMG21, MSS21, LT21].

We let the tracking server run FTKGen in the beginning to publish fuzzy public key fpk and secretly hold the fuzzy tracking key ftk. For each ftki received from senders, the tracking server will expand ftki to a list of size \( t \) composed of potential users’ master public keys to which ftki may belong to. The tracking server can then store (mpk, tki) to the mailbox of each candidate in this list. Crucially, the master public keys of other potential candidates should remain uncontrollable to either the sender or the server. Otherwise the sender might manipulate the chance of each key appearing in the list. This additional property is named unbiasedness. This rules out the trivial solution, where for instance the sender just sends directly a range of master public keys including the targeted mpk.

Since mpk of each user can be large, in our construction we hash mpk \( \in K \) to some small hint \( \in T \) (while making \( |T| > N \)) and use the hint to locate each user’s mailbox. Our scheme is based on the underlying IND-CPA encryption of Kyber, except that we use non-prime modulus. For instance, assuming the hint contains \( n = \log N \) bits, i.e., \( b := \text{hint} \in \{0, 1\}^n \), to generate ftki, the sender modifies the Kyber512’s ciphertext \( ct := (c_1, c_2) \) to \( ct' := (c'_1, c'_2) \) as follows:

\[
c'_1 := c_1 + \frac{q}{2} x_i, \quad c'_2 := c_2 + \frac{q}{2} y_i,
\]

where \( ct \) (and \( ct' \)) encrypts hint, as the plaintext, \( x_i \leftarrow \text{encode}_{R_0}(\text{random}) \) is a polynomial mapped from the vector \( x_i \), and \( X, y_i \in \{0, 1\}^n \leftarrow H(\delta, i) \) are outputs of a hash function \( H \) with the seed \( \delta \). Here \( i \in [t] \) denotes the i-th target mpk as the intended recipient.

For \( ftki := ct' \), the tracking server decrypts \( ct' \) using the key \( sk = s \) as follows: for \( \forall j \in [t] \),

\[
\text{hint}_j \leftarrow \text{decode}_{R_0}([s^T(c'_1 - \frac{q}{2} x_j) - c'_2]_2 \oplus y_j),
\]

to get \( t \) potential hints. To argue privacy, intuitively, since \( s \) remains random to the sender, the decrypted hint for \( j \neq i \) would also be random to the sender as

\[
\text{hint}_j = \text{hint}_i \oplus (y_j \oplus y_i) \oplus \text{decode}_{R_0}([\frac{q}{2} s^T x_j]_2).
\]

However, to prove unbiasedness we mentioned above, we need to be careful because standard regularity lemma seems hard to apply with such small noise parameter and modulus in ideal lattices. Our solution is to rely on the specific structure of the corresponding cyclotomic polynomial and show that even \( s^T x_j \) is not close to a uniformly random polynomial but there’s enough entropy to make \( \text{hint}_j \) uniformly random over \( \{0, 1\}^n \) as long as \( n \) is much smaller than the degree of the polynomial.

2.5 From Stealth Addresses to FIDO

We will now describe how stealth addresses fit into FIDO-based passwordless authentication. The FIDO standard specifies ways of using device-based tokens (authenticators) for online authentication.\(^3\)

\(^3\)According to [SPB21], "concerning recipient unlinkability and temporal detection ambiguity, the false-positive rate needs to be high and there must be a large number of users in the system." This can be achieved in current blockchain systems with millions of users by setting the false-positive rate to be \( 1/\sqrt{N} \). Although the same work points out that relationship anonymity only holds when senders are hidden from the server, this is typically addressed using other techniques such as ring signatures. Furthermore, as shown in [SPB21], "...users do not employ any cover traffic due to their selfishness...", which has inspired us to propose scalable fuzzy tracking as a solution to overcome this issue.
The protocol is a simple challenge-response where the authenticator creates a signature under the server’s challenge. The server verifies the signature against a public key stored during the registration phase. Thus, the token uses different public keys per server to ensure privacy, which poses a challenge for memory-constrained devices. The common practice is to use key wrapping or a key derivation function to generate the signing key ad-hoc during the authentication process, where the server provides additional information (e.g., the ciphertext wrapping the signing keys or random value for the derivation function) for the re-computation.

Applying stealth addresses to this setting would use the same idea. The authenticator only needs to store the master secret key $\text{msk}$ and the tracking key $\text{mtk}$. During the registration phase, the server would receive a one-time public key $\text{opk}$ and send it to the token during authentication. The authenticator can then reconstruct the corresponding one-time secret key $\text{osk}$ and respond to the server’s challenge. Interestingly, due to our strong key-exposure notions, one-time keys can be leaked without compromising the unforgeability of non-leaked keys. Thus, we can use the same keys on multiple devices owned by the user, implementing the concept of FIDO multi-device credentials [All22].

Contrary to existing solutions, the public keys do not have to be generated on-token. The user platform (e.g., browser, second token) can generate the one-time public key $\text{opk}$ without $\text{msk}$ by just using the master public key $\text{mpk}$. What we just described is also called asynchronous remote key generation [FGK+20], a solution that cleverly uses two devices to solve the token loss problem. After an initialization phase, one of the authenticators is put into cold storage while registration (in the name of both) is done using the primary token. In case of loss, the authenticator in storage can create a valid response to the server’s challenge. Using stealth addresses provides the same feature without a complicated initialization step.

One problem with lost devices is that the public keys on the server side are still bound to the user’s account. A simple but bothersome solution is for the user to contact all servers it used in the past and revoke the keys. A more flexible solution was proposed by Hanzlik, Loss, and Wagner [HLW22]. With the help of a revocation key published by the user, servers can identify public keys corresponding to lost tokens. Stealth addresses provide the same feature and even improve it a bit. One can identify one-time public keys generated using the same master public key $\text{mpk}$ with the help of the tracking key $\text{mtk}$. In the FIDO scenario, a published $\text{mtk}$ can be used to globally revoke a lost/stolen token while at the same time not allowing to create forger signature as required by the notions in [HLW22].

Releasing the tracking key $\text{mtk}$ will allow everyone to link the one-time public keys, which is a privacy concern. Fuzzy tracking allows for more fine-tuned protection in the case of lost/stolen tokens. Instead of revoking public keys, servers could employ an additional policy-based mechanism to challenge authentications against public keys identified by fuzzy tracking. In other words, the user can hide the actual public key in a set of potential keys, and the server requires some additional authentication factors for those keys. Authenticating using the lost/stolen token will be impossible in such a scenario while at the same time providing extra privacy for the user that lost the device.

### 3 Preliminaries

We denote by $\lambda \in \mathbb{N}$ the security parameter and by $x \leftarrow \mathcal{A}(\text{in}; r)$ the output of the algorithm $\mathcal{A}$ on input $\text{in}$ using $r \leftarrow \{0, 1\}^*$ as its randomness. We often omit this randomness and only mention it explicitly when required. The notation $[n]$ denotes a set $\{1, \ldots, n\}$ and $x[1:n]$ denotes the sub-vector of $x$ with first $n$ elements. We consider probabilistic polynomial time (PPT) machines as efficient algorithms. Also, we use $\approx_c$ and $\approx_s$ to denote computational closeness and statistical closeness,
respectively. We defer the reader to Appendix C for assumptions and analysis tools we use in this work. Apart from this, we make use of the following cryptographic primitives.

### Digital Signatures

A digital signature scheme $\text{DS}$, formally, has a key generation algorithm $\text{KGen}(\lambda)$ that takes the security parameter $\lambda$ and outputs the verification/signing key pair $(vk, sk)$, a signing algorithm $\text{Sign}(sk, m)$ inputs a signing key and a message $m \in \{0, 1\}^*$ and outputs a signature $\sigma$, and a verification algorithm $\text{Vf}(vk, m, \sigma)$ outputs 1 if $\sigma$ is a valid signature on $m$ under the verification key $vk$, and outputs 0 otherwise. We require unforgeability, which guarantees that a PPT adversary cannot forge a fresh signature on a fresh message of its choice under a given verification key while having access to a signing oracle (that returns a valid signatures on the queried messages). Formally the notion can be captured in an experiment denoted by $\text{EUFCMA}$. Strong unforgeability refers to the case where the adversary is required to forge a fresh signature on not necessarily a fresh message. Formally the notion can be captured in an experiment denoted by $\text{sEUFCMA}$.

### Key Encapsulation Mechanism

A key encapsulation mechanism $\text{KEM}$, formally, has a key generation algorithm $\text{KGen}(\lambda)$ that takes the security parameter $\lambda$ and outputs a encaps key $ek$ and a decaps key $dk$. An encapsulation algorithm $\text{Encaps}(ek)$ inputs an encaps key and outputs a ciphertext $C$ and agreed key $K$. Finally, we have a decapsulation algorithm $\text{Decaps}(dk)$ inputs a decaps key and a ciphertext and outputs an agreed key $K$. Apart from INDCCA security, we additionally require its anonymous property which can be formally captured in Definition C.7 denoted by $\text{ANOCCA}$ and it means the adversary cannot link any ciphertext $C$ to its encaps key $ek$ even being able to access a decaps oracle. Concretely, we use Kyber [SAB+20] with the modification shown in Figure 6 of [GMP22].

# 4 Definitions of (Fuzzy) Stealth Signatures

In this section we first present our formal definitions for a stealth signature scheme, followed by how we can add-on fuzziness to the scheme. Note that stealth signatures were formalized in prior works [LYW+19, LLN+20], however our formalization of security is strictly stronger than theirs, and moreover we are the first to formalize tracking and fuzzy tracking for a stealth signature scheme. We will point out the exact differences\(^4\) in the formalism as we introduce the security notions formally.

Below we present the definition of stealth signatures, that formalizes the tracking of keys which was absent in prior works. This formalization allows for tracking to be outsourced to third-party servers.

**Definition 4.1.** A *stealth signature* (SS) scheme consists of the PPT algorithms $(\text{MKGen}, \text{OPKGen}, \text{OSKGen}, \text{Track}, \text{Sign}, \text{Vf})$ that are defined as follows.

- $(\text{mpk}, \text{msk}, \text{mtk}) \leftarrow \text{MKGen}(\lambda)$: the master key generation algorithm takes as input the security parameter $\lambda$ and outputs the master public key $\text{mpk}$, the master secret key $\text{msk}$, and the master tracking key $\text{mtk}$.
- $(\text{opk}, \text{tki}) \leftarrow \text{OPKGen}(\text{mpk})$: the one-time public key generation algorithm takes as input the master public key $\text{mpk}$, and outputs the one-time public key $\text{opk}$ and the tracking information $\text{tki}$.
- $\text{osk}/\perp \leftarrow \text{OSKGen}(\text{msk}, \text{opk}, \text{tki})$: the one-time secret key generation algorithm takes as input the master secret key $\text{msk}$, the one-time public key $\text{opk}$, and the tracking information $\text{tki}$, and outputs a one-time secret key $\text{osk}$ or a special symbol $\perp$.
- $\text{true}/\text{false} \leftarrow \text{Track}(\text{mtk}, \text{opk}, \text{tki})$: the tracking algorithm takes as input the master tracking key $\text{mtk}$, the one-time public key $\text{opk}$, and the tracking information $\text{tki}$, and outputs true or false.

\(^4\)Please refer to Definition 4.6 for details.
\[ \sigma / \bot \leftarrow \text{Sign}(osk, m) \]: the signing algorithm takes as input the one-time secret key \( osk \), and a message \( m \), and outputs a signature \( \sigma \) or a special symbol \( \bot \).

\[ \text{true/false} \leftarrow \text{Vf}(opk, m, \sigma) \]: the verification algorithm takes as input the one-time public key \( opk \), a message \( m \), and a signature \( \sigma \), and outputs \text{true} or \text{false}.

The notion of correctness is formalized below.

**Definition 4.2** (Correctness). A SS scheme \( (\text{MKGen}, \text{OPKGen}, \text{OSKGen}, \text{Track}, \text{Sign}, \text{Vf}) \) is said to be correct if for all \( \lambda \in \mathbb{N} \), all \((mpk, msk, mtk) \leftarrow \text{MKGen}(\lambda)\), all \((opk, tki) \leftarrow \text{OPKGen}(mpk)\), all \(osk \leftarrow \text{OSKGen}(msk, opk, tki)\), we have the following that hold simultaneously:

- we have \( \Pr[\text{Track}(mtk, opk, tki) = \text{true}] = 1 \)
- we have \( \Pr[\text{Vf}(opk, m, \text{Sign}(osk, m)) = \text{true}] = 1 \),

note that sometimes we don’t require perfect correctness and having correctness probability \( 1 - \text{negl}(\lambda) \) instead would suffice.

### 4.1 Security of SS Without Key Exposure

In terms of security, we first want unforgeability, which guarantees that it is infeasible for an adversary to forge a signature on a (fresh) message wrt. some one-time public key \( opk^* \) for a master public key \( mpk \). The adversary is given access to a one-time secret key generation oracle \( \text{OSKGen}^O \) using which the adversary can generate a fresh one-time secret key. However, the adversary does not get to learn the generated one-time secret keys, therefore the notion is said to be without key exposure. The adversary also has access to a signing oracle, to which it can query a signature on any message of its choice wrt. any one-time secret key that has been generated with a query to \( \text{OSKGen}^O \). The formal definition is presented below.

**Definition 4.3** (Unforgeability without key-exposure). A SS scheme \( (\text{MKGen}, \text{OPKGen}, \text{OSKGen}, \text{Track}, \text{Sign}, \text{Vf}) \) is said to be unforgeable without key exposure if there exists a negligible function \( \text{negl} \) for all \( \lambda \in \mathbb{N} \), and for all PPT adversaries \( A \) the following holds:

\[
\Pr[\text{sEUFCMA}^{A}_{w/\neg\text{ke}}(\lambda) = 1] \leq \text{negl}(\lambda)
\]

where \( \text{sEUFCMA}^{A}_{w/\neg\text{ke}} \) is defined in Figure 1.

We then want unlinkability, which guarantees that it is infeasible for an adversary to associate a one-time public key to the master public key wrt. which it was generated. The adversary is given two master public keys \( mpk_0 \) and \( mpk_1 \), while also given a challenge one-time public key \( opk_b \) and the corresponding tracking information \( tki_b \) (for \( b \in \{0, 1\} \)) generated wrt. \( mpk_b \). The adversary is given access to the \( \text{OSKGen}^O \) as before, and a signing oracle. The adversary is not given access to any of the one-time secret keys and therefore the notion is said to be without key exposure. The formal definition is presented below.

**Definition 4.4** (Unlinkability without key-exposure). A SS scheme \( (\text{MKGen}, \text{OPKGen}, \text{OSKGen}, \text{Track}, \text{Sign}, \text{Vf}) \) is said to be unlinkability without key exposure if there exists a negligible function \( \text{negl} \) for all \( \lambda \in \mathbb{N} \), and for all PPT adversaries \( A \) the following holds:

\[
\Pr[\text{UNLNK}^{A}_{w/\neg\text{ke}}(\lambda) = 1] \leq \frac{1}{2} + \text{negl}(\lambda)
\]

where \( \text{UNLNK}^{A}_{w/\neg\text{ke}} \) is defined in Figure 2.
4.2 Security of SS With Key Exposure

Prior works [LYW19, LLN20] formalized security with additionally giving adversary the one-time secret keys, i.e., the OSKGenO returns the generated osk to the adversary.

The unforgeability notion with key exposure is formalized below. Notice that our formalization exposes the one-time secret keys osk to the adversary except the key wrt. which the adversary forges the signature.

**Definition 4.5** (Unforgeability with key-exposure). A SS scheme (MKGen, OPKGen, OSKGen, Track, Sign, Vf) is said to be unforgeable with key exposure if there exists a negligible function negl for all \( \lambda \in \mathbb{N} \), and for all PPT adversaries \( A \) the following holds:

\[
\text{Pr} \left[ \text{sEUFCA}_{\text{w/o-ke}}^4(\lambda) = 1 \right] \leq \text{negl}(\lambda)
\]

where sEUFCA_{w/o-ke} is defined in Figure 3.

The notion of unlinkability with key exposure is formalized below. Similar to the case above, the OSKGenO returns the generated osk.

**Definition 4.6** (Unlinkability with key-exposure). A SS scheme (MKGen, OPKGen, OSKGen, Track, Sign, Vf) is said to be unlinkable with key exposure if there exists a negligible function negl for all
\[ \Pr \left[ \text{UNLNK}_{w}^{A}(\lambda) = 1 \right] \leq \frac{1}{2} + \text{negl}(\lambda) \]

where UNLNK_{w/o-ke} is defined in Figure 4.

**Remark.** It is worth noting that our formalization apart from the tracking functionality, is stronger than prior works in that the adversary is even given the challenge one-time secret key osk_{b}.

### 4.3 Fuzzy Stealth Signatures

We now formally incorporate the fuzzy tracking functionality into the definition of stealth signing.

**Definition 4.7 (Fuzzy Stealth Signatures).** A fuzzy stealth signatures (F-SS) scheme is a SS scheme (MKGen, OPKGen, OSKGen, Track, Sign, Vf) with additional interfaces (FTKGen, FTrack) defined below.

\[(\text{opk}, \text{tki}, \text{ftki}) \leftarrow \text{OPKGen}(\text{mpk}) \text{: overloading the interface OPKGen to output the fuzzy tracking information ftki.} \]

\[ \text{OSKGen}(b^{*}, \text{opk}, \text{tki}) \]
ftk ← FTKGen(mtk, ρ): the fuzzy tracking key generation algorithm takes as input the master tracking key mtk, and a false positivity rate ρ, and outputs a fuzzy tracking key ftk.

true/false ← FTrack(ftk, ftki): the fuzzy tracking algorithm takes as input the fuzzy tracking key ftk, the fuzzy tracking information ftki, and outputs true or false.

We define the notion of correctness below. We borrow the notion of fuzziness from [BLMG21] and adapt the same for the stealth signature setting. Intuitively, the correctness of fuzzy tracking says that with a probability ρ, the fuzzy tracking algorithm returns true for a mismatched fuzzy tracking key and a one-time public key. For a correctly matched fuzzy tracking key and a one-time public key, the tracking algorithm always returns true.

**Definition 4.8 (Correctness for fuzzy tracking).** A F-SS scheme (MKGen, OPKGen, OSKGen, Track, Sign, Vf, FTKGen, FTrack) is said to be correct if the original SS scheme is correct and if for all λ ∈ N, all ρ ∈ (0, 1] such that log2 ρ ∈ Z, all (mpk, msk, mtk) ← MKGen(λ), all (opk, tki, ftki) ← OPKGen(mpk), all osk ← OSKGen(msk, opk, tki), all ftk ← FTKGen(mtk, ρ), we have the following that holds simultaneously:

- Pr[FTrack(ftk, ftki) = true] = 1
- and for any ftki′ /∈ SUPP(OPKGen(mpk)), we have Pr[FTrack(ftk, ftki′) = true] = ρ.

The unforgeability notion remains the same as in Figure 3, as the adversary in this notion already has access to the master tracking key.

Unlinkability with fuzzy tracking ensures that it is computationally infeasible for an adversary, given two fuzzy tracking keys that both return either true or false when tracking a challenge one-time public key (opkb, ftki) simultaneously, to associate (opkb, ftki) with the correct tracking key (either ftk0 or ftk1). The adversary is said to violate the notion if it can guess correctly the association non-negligibly more than 1/2.

**Definition 4.9 (Unlinkability with key-exposure and fuzzy tracking).** A F-SS scheme (MKGen, OPKGen, OSKGen, Track, Sign, Vf, FTKGen, FTrack) is said to be unlinkable with key-exposure and fuzzy tracking if there exists a negligible function negl for all λ ∈ N, all ρ ∈ (0, 1] such that log2 ρ ∈ Z, and for all PPT adversaries A the following holds:

\[ \Pr[\text{UNLNK}_{\text{fw-ke}}^A(\lambda, \rho) = 1] \leq \frac{1}{2} + \text{negl}(\lambda) \]

where UNLNK_{fw-ke} is defined in Figure 5.

### 4.4 Scalable Fuzzy Tracking

We now formalize the functionality, correctness, and security of fuzzy scalable stealth signatures as follows.

**Definition 4.10 (Fuzzy Scalable Stealth Signatures).** A fuzzy scalable stealth signature (F-SSS) is a SS scheme (MKGen, OPKGen, OSKGen, Track, Sign, Vf) with additional interfaces (FTKGen, FTrack) and a modified OPKGen defined below.

(fpk, ftk) ← FTKGen(ρ, N): the fuzzy tracking key generation algorithm takes as input a false positivity rate ρ, and the number of total users N, and outputs a fuzzy tracking key ftk and fuzzy public key fpk. The algorithm is run by the tracking server ahead of time.
if there exists a negligible function $\negl(\lambda)$.

**Definition 4.11** (Correctness for fuzzy scalable stealth signatures). A F-SSS scheme $(\text{MKGen}, \text{OPKGen}, \text{OSKGen}, \text{Track}, \text{Sign}, \text{Vf}, \text{FTKGen}, \text{FTrack})$ is said to be correct if the original SS scheme is correct and if for all $\lambda \in \mathbb{N}$, any integer $N$, all $\rho \in (0, 1]$ such that $\frac{1}{\log_2 N}$, all $(\text{mpk}, \text{msk}, \text{mtk}) \leftarrow \text{MKGen}(\lambda)$, all $(\text{mpk}, \text{tki}) \leftarrow \text{OPKGen}(\text{mpk})$, all $\text{osk} \leftarrow \text{OSKGen}(\text{mpk}, \text{opk}, \text{tki})$, all $(\text{fpk}, \text{ftk}) \leftarrow \text{FTKGen}(\lambda, \rho, N)$, we have the following that holds simultaneously:

- $\Pr[\text{mpk} \in \text{FTrack}(\text{ftk}, \text{ftki})] = 1$
- and for any $\text{mpk}' \neq \text{mpk}$, we have
  \[
  \Pr[\text{mpk}' \in \text{FTrack}(\text{ftk}, \text{ftki})] = \rho.
  \]

Crucially, we omit $\text{opk}$ in $\text{FTrack}$ as $\text{ftki}$ is already associated with $\text{opk}$ and we still have the regular $\text{Track}$ algorithm that works with $\text{tk}$, $\text{opk}$ and $\text{tki}$ for $\text{FTrack}$. The correctness definition above ‘ties’ together the keys $\text{ftk}$, $\text{mpk}$ and $\text{mtk}$, and $(\text{opk}, \text{tki}, \text{ftki}) \leftarrow \text{OPKGen}(\text{mpk})$ by requiring that $\text{FTrack}(\text{ftk}, \text{ftki})$ always returns 1.

**Definition 4.12** (Unlinkability with key-exposure and fuzzy scalable tracking). A F-SSS scheme $(\text{MKGen}, \text{OPKGen}, \text{OSKGen}, \text{Track}, \text{Sign}, \text{Vf}, \text{FTKGen}, \text{FTrack})$ is said to be unlinked with key-exposure and fuzzy scalable tracking if there exists a negligible function $\negl$ for all $\lambda \in \mathbb{N}$, any integer $N$, all $\rho \in (0, 1]$ such that $\frac{1}{\log_2 N}$, and for all PPT adversaries $\mathcal{A}$ the following holds:

\[
\Pr[\text{UNLNK}^{\mathcal{A}}_{\lambda, \text{ke}}(\lambda, \rho, N) = 1] \leq \frac{1}{2} + \negl(\lambda)
\]
Theorem 5.1 (informal) We provide our black-box compiler below to upgrade an SS we have a black-box compiler leveraging UNIUBS where secure.

\[ \text{sEUFCMA} \]

UNLNK_{ke}^{\text{w}}(\lambda, \rho, N)

\begin{align*}
\text{OK}_0 & := \text{OK}_1 := [] \\
(\text{mpk}_0, \text{msk}_0, \text{mtk}_0) & \leftarrow \text{MKGen}(\lambda) \\
(\text{mpk}_1, \text{msk}_1, \text{mtk}_1) & \leftarrow \text{MKGen}(\lambda) \\
(\text{fpk}, \text{ftk}) & \leftarrow \text{FTrack}(\text{ftk}, \text{ftki}) \\
b & \in \{0, 1\} \\
(\text{opk}_1, \text{tki}_1, \text{ftki}_1) & \leftarrow \text{OPKGen}(\text{mpk}_1, \text{fpk}) \\
\text{osk}_1 & \leftarrow \text{OSKGen}(\text{msk}_1, \text{opk}_1, \text{tki}_1) \\
\text{list} & \leftarrow \text{FTrack}(\text{ftk}, \text{ftki}_1) \\
\text{if} \ \text{mpk}_0 \in \text{list} \land \text{mpk}_1 \in \text{list} & \\
\text{b'} & \leftarrow \text{A}_{\text{FTrack}}(\text{ftk}, \text{mpk}_1, \text{tki}_1, \text{ftki}_1, \text{osk}_1) \\
\text{else} & \text{b'} \leftarrow \{0, 1\} \\
l_0 & := (b = b') \\
\text{return} l_0
\end{align*}

\[ \text{UNIUBS}_{\text{f}}(\lambda, \rho, N) \]

\begin{align*}
\text{(fpk, ftk)} & \leftarrow \text{FTrack}(\rho, N) \\
(\text{st}_{A}, \text{ftki}, i, j, \text{mpk}) & \leftarrow \text{A}_{1}(\text{fpk}) \\
\text{list} & \leftarrow \text{FTrack}(\text{ftk}, \text{ftki}) \\
b & \in \{0, 1\} \\
\text{if} \ \text{list}[i] \neq \text{mpk} \lor i = j \lor \text{mpk} \not\in \mathcal{K} & \\
b' & \leftarrow \{0, 1\} \\
\text{else} & \text{v}^0 := \text{list}[j], \text{v}^1 \leftarrow \{\mathcal{K}\} \\
b' & := \text{A}_{2}(\text{st}_{A}, \text{v}^1) \\
\text{return} b \overset{\text{b}}{=} b' \\
\text{OSKGenO}(\text{b}^*, \text{opk}, \text{tki}) & \leftarrow \text{OSKGen}(\text{msk}_1, \cdot, \text{opk}, \text{tki}) \\
\text{OK}_0 & := \text{OK}_0 \cup \{(\text{opk}, \text{ask})\} \\
\text{return} \text{ok}
\end{align*}

Figure 6: Experiments for unlinkability and uniformly unbiasedness of F-SSS with Key-Exposure.

where UNLNK_{ke}^{\text{w}} is defined in Figure 6. Note that, similar to prior works, we only consider the semi-honest server in the definition.

Definition 4.13 (Unbiasedness for fuzzy scalable tracking). A F-SSS scheme (MKGen, OPKGen, OSKGen, Track, Sign, Vf, FTrack, FTK) is said to be unbiased by senders if there exists a negligible function \(\text{negl}\), for all \(\lambda \in \mathbb{N}\), and for all PPT adversaries \(\mathcal{A}\) the following holds:

\[
\Pr\left[\text{UNIUBS}_{\text{f}}(\lambda, \rho, N) = 1\right] \leq \frac{1}{2} + \text{negl}(\lambda),
\]

where the experiment UNIUBS_{\text{f}} is defined in Figure 6 where list[i] denotes the \(i\)-th item of the list and \(\mathcal{K}\) denotes the master public key space.

5 Generic Transformation To Get Security With Key Exposure

We provide our black-box compiler below to upgrade an SS_{w/o} without key-exposure to an SS_{w} with key-exposure.

Suppose we have a digital signature scheme DS which is strongly unforgeable sEUFCMA. Then we have a black-box compiler leveraging SS to stronger version as shown in Figure 7. Basically, the compiler transforms any SS_{w/o} with EUFCMA_{w/o} and UNLNK_{w/o} security (without key-exposure) into an SS_{w} with sEUFCMA_{w} and UNLNK_{w} security (with key-exposure).

It is easy to see that correctness always holds as long as SS_{w/o} and DS are correct. The security of unforgeability and unlinkability for SS_{w} are captured informally in the following theorem. The formal theorem and security proofs are deferred to Appendix E.

Theorem 5.1 (informal). The stealth signature SS_{w} constructed in Section 5 is secure in sEUFCMA_{w} and UNLNK_{w} experiments if SS_{w/o} is EUFCMA_{w/o} secure, UNLNK_{w/o} secure, and DS is sEUFCMA secure.
We first describe Spirit 6 Spirit: Lattice based (Fuzzy) Stealth Signature

![Figure 8: A generic transformation to lift SS_{w/0} to SS_w.](Image)

![Figure 7: A generic transformation to lift SS_{w/0} to SS_w.](Image)

6 Spirit: Lattice based (Fuzzy) Stealth Signature

We first describe Spirit and later show we can make it fuzzy.
6.1 Lattice-based Stealth Signature

We use an ANOCCA-secure key exchange KEM (Kyber) [SAB+20] and an EUFCMA-secure signature (Dilithium) to construct an SS scheme with existential unforgeability without key-exposure and unlinkability without key-exposure in random oracle model. We require a common reference string unlinkability without key-exposure ANOCCA. We use an concrete security levels in Table 3.

EUFCMA of in [LDK] such that Az setting), the hardness of SelfTargetMSIS security. Without using the forking lemma (since it is not tight and not applicable in the quantum MSIS is mainly dominated by hardness but won’t harm the running time. To see this, in Dilithium denoted by "Fiat-Shamir with Aborts" approach [Lyu09, Lyu12]. It is based on MLWE, MSIS and SelfTargetMSIS assumptions with ring $R_q := \mathbb{Z}_q[X]/(X^m + 1)$. Moreover, for secrets $s \leftarrow S^t_q$, its each coefficient of the vector is an element of $R_q$ with small coefficients of size at most $\eta$. In its optimized construction, there are some useful supporting algorithms which we described as follows:

- **ExpandA(crs)**: The function maps a uniform seed crs to a matrix $A \in R_q^{k \times \ell}$.
- **ExpandS(K)**: The function used for generating the secret vectors in key generation, maps a seed $K$ to $(s_1, s_2) \in S^{\ell}_q \times S^{\ell}_q$.
- **power2Round(r, d)**: The function is the straightforward bit-wise way to break up an element $r := r_1 \cdot 2^d + r_0$ where $r_0 = r \mod 2^d$ and $r_1 = (r - r_0)/2^d$.
- **HighBits_q(r, \alpha)**: The function select an $\alpha$ that is a divisor of $q - 1$ and write $r = r_1 \cdot \alpha + r_0$ in the same way as before then returns $r_1$.
- **MakeHint_q(z, r, \alpha)**: The function runs $r_1 \leftarrow \text{HighBits}(r, \alpha)$ and $v_1 \leftarrow \text{HighBits}(r + z, \alpha)$, then returns $r_1 \neq v_1$.

**Correctness.** Since $t' = t + As'_1 + s'_2 = A(s'_1 + s_1) + (s_2 + s'_2)$, it is easy to see we have $1 - \text{negl}(\lambda)$ correctness as long as underlying KEM and Dil have $1 - \text{negl}(\lambda)$ correctness.

Notably, $s'_1 + s_1$ and $s_2 + s'_2$ have approximately doubled norms, which results in doubled $\beta$ in signatures. This will require additional iterations in the Sign algorithm, as the number of repetitions is roughly $2^{256 \beta(\frac{1}{\delta} + \frac{1}{\beta})}$, where $\gamma_1 \approx 2\gamma_2$ [LDK+20]. However, besides having doubled $\beta$, we can also increase $\gamma_1$ and $\gamma_2$ to $2\gamma_1$ and $2\gamma_2$, respectively. This adjustment slightly lowers the SelfTargetMSIS hardness but won’t harm the running time. To see this, in Dilithium’s proof, the reduction’s advantage is mainly dominated by MSIS$_{k, \ell, \gamma_1}$ for sEUFCMA security, but SelfTargetMSIS$_{k, \ell + 1, 2\gamma_2}$ for EUFCMA security. Without using the forking lemma (since it is not tight and not applicable in the quantum setting), the hardness of SelfTargetMSIS mainly comes from finding short vectors $(\|\|_\infty \leq 2\gamma_2) z, u'$ such that $Az + u' = t'$ and amounts to the MSIS problem (refer to Section 6.2.1 and Appendix C.3 in [LDK+20] for details). Therefore, doubling $\gamma_2$ in our SPIRIT construction provides the reduction of EUFCMA$_{\omega/\alpha \rightarrow \text{ke}}$ with roughly the same advantage as that of sEUFCMA in Dil. We present the concrete security levels in Table 3.
We provide a lattice-based construction for fuzzy tracking in standard model. Basically, it is packed Regev encryption (denoted as pRgv) with ciphertext compression [BDGM19]. This gives us the first post-quantum ambiguous encryption without relying on random oracles.

**Packed Regev (compressed).** For a more detailed description and analysis, please refer to Appendix G. In short terms, the packed Regev scheme pRgv is a lattice-based linearly homomorphic encryption that has an additional property that allows for ciphertext compression. This unique feature enables the representation of a ciphertext encrypting \( n \) bits with a size of only \( n + O(\lambda) \) bits. This reduced ciphertext size contributes to the both asymptotic and concrete efficiency of the scheme.

**Security Analysis.** We prove the construction of SPIRIT in Figure 8 is existential unforgeable and unlinkable without key exposure, and is secure in EUFCMA\(_{w/o-ke}\) and UNLNK\(_{w/o-ke}\) experiment, respectively. For security of EUFCMA\(_{w/o-ke}\), we prove this in two steps. First, we show it is unforgeable without key exposure under no-message attacks (NMA), i.e., the adversary cannot query \( \text{Sign}(\cdot) \), and we refer the corresponding experiment to UFNMA\(_{w/o-ke}\); Next, we show a reduction from UFNMA\(_{w/o-ke}\) to EUFCMA\(_{w/o-ke}\). Since \( \text{Dil} \) does not rely on the lower parts of public key \( t_0 \) to be secret, so for simplicity, we assume the one-time public key \( \text{opk} \) is \( t' \) instead of \( t'_1 \). Also, we assume \( \text{crs} := A \) directly and is publicly known.

**Lemma 6.1 (informal).** SPIRIT in Figure 8 is unforgeable without key exposure under no-message attacks if SelfTargetMSIS and MLWE assumptions hold.

Then we have the following theorems to show the construction is unforgeable and unlinkable. The formal statement and analysis of the above lemma and the following theorem is deferred to Appendix F.

**Theorem 6.2 (informal).** SPIRIT in Figure 8 is existential unforgeable and unlinkable without key exposures if it is UFNMA\(_{w/o-ke}\) and the KEM used is ANOCCA secure.

### 6.2 Lattice-based Fuzzy Stealth Signature

We provide a lattice-based construction for fuzzy tracking in standard model.
Since IND-CPA and IKCPA security (recalled in Appendix C) of pRgv are discussed in prior works already, we focus on its ambiguous security and we show it is actually Uniformly-ambiguous (recalled in Appendix C) with super-poly noise-modulus ratio. The formal statement and proof of the lemma below is deferred to Appendix G.

**Lemma 6.3.** Packed Regev encryption pRgv with ciphertext compression shown in Figure 16 satisfies Definition C.8 and is uniformly-ambiguous UNIAMB-secure when $\frac{\lambda \rho}{\eta}$ is negl($\lambda$).

**The modulus.** To argue uniformly-ambiguous security, we need super-polynomial noise-to-modulus ratio (e.g., 60-bit modulus in our case) which is usually assumed in homomorphic encryption related works. This is a somewhat stronger assumption since it assumes the lattice problem BDD or GapSVP is hard even with super-polynomial approximation factor [Reg05].

**Construction.** We then provide a lattice-based fuzzy stealth signature in Figure 9, which is composed of a standard stealth signature SS and a compressed packed Regev encryption pRgv shown above. Basically, it use the same framework as FMD$_1$ presented in [BLMG21].

**Correctness.** We provide the correctness analysis in Appendix G.

Now we consider the false-positive rate $\rho$ when using different fuzzy tracking key. Since $c_1$ looks uniformly random due to LWE assumption, $s^Tc_1$ is uniformly random over $\mathbb{Z}_q$ by Leftover Hash Lemma as inner product is a strong randomness extractor$^5$. This implies $[s^Tc_1 + z]$ is uniformly random over $\{0, 1\}$ and FTrack returns true with probability $2^{-t} = \rho$.

**Security Analysis.** Formal statement and corresponding proof of the following theorem are deferred to Appendix G.

**Theorem 6.4 (informal).** The fuzzy stealth signature constructed in Figure 9 is unlinkable with key-exposure and fuzzy tracking if the underlying stealth signature is UNLNAK$_{w=k}$ and pRgv is UNIAMB and IKCPA secure.

We also provide an approach to extend it to finer false-positive rate as shown in Appendix G.

### 6.3 Scalable Lattice-based Fuzzy Tracking

As discussed in Section 2.4, we limit the user’s ability to choose false-positive rate and provide a new framework of fuzzy tracking which is substantially more scalable than prior works [BLMG21, MSS+21, LT21]. Please refer to Section 4.4 for functionality and security definitions.

**Construction.** We describe the detailed construction in Figure 10, where $\{0,1\}^{2m} \leftarrow H(k \in \{0,1\}^3, i \in [t])$ is a hash function with the seed $k$ and $H_n : \{0,1\}^{||mpk||} \rightarrow \{0,1\}^n$ is another hash function mapping mpk to a hint which is used to locate mpk’s mailbox in server’s storage. Since it is based on Module-LWE assumption, $R_q$ denotes the ring $\mathbb{Z}_q[X] / \langle X^m + 1 \rangle$, and $\text{encode}_{R_q} : \{0,1\}^m \rightarrow \mathbb{Z}_q[X] / \langle X^m + 1 \rangle$ is a function mapping binary strings to the ring elements with binary coefficients; Similarly, $\text{decode}_{R_q}$ is the reverse operation to map back to binary string. Basically, it is a variant of the underlying IND-CPA encryption of Kyber with non-prime modulus because we need $\mathbb{Z}_2$ to be a subgroup of $\mathbb{Z}_q$ in correctness and security analysis. Though we lose the advantage of NTT multiplications, we can still mitigate this by using Karatsuba and Toom-Cook algorithms.

**Correctness.** It is clear to see that the targeted mpk must have hint := hint’ = $H_n$(mpk) appears in list with probability 1: For the targeted index $i \in [t]$, we have $c^i_j = A^T r + e_1$ which is the same as standard ciphertext header. The decryption will output hint directly as long as $q > 4B$. Now we focus on the other case where mpk’ $\neq$ mpk.

Firstly, considering hint’ $\in$ list, it is decrypted as

$$[s^Tc^i_j - c_2]_2 \oplus y’ = [c’ + \frac{q}{2}w + s_1(x^i - x^j) + y’]_2 \oplus y’,$$

$^5$We only use it for correctness (or fuzziness), not for security.
where $s_1$ is the first ring element of $s$. $\text{hint}_j$ is uniformly random over $\{0, 1\}^n$ after rounding $\lfloor \cdot \rfloor_2$ as $y' \oplus y'$ are outputs of the random oracle $H$. Then, for any $mpk' \neq mpk$, $\Pr[H_n(mpk') = \text{hint}_j] = \frac{1}{2^m}$ since $H_n$ is a random oracle, and

$$\Pr[H_n(mpk') \in \text{list}] = \sum_{j=1}^{t} \Pr[H_n(mpk') = \text{hint}_j] = \frac{t}{2^m} = \rho.$$

**Security Analysis.** The formal theorem statements and proof of the following theorems are deferred to Appendix H.

**Theorem 6.5 (informal).** The fuzzy scalable stealth signature constructed in Figure 10 is unlinkable with key-exposure and fuzzy tracking if the underlying stealth signature is UNLNK$_{\text{uni}}$ and MLWE holds. It is also unbiased and satisfying UNIUBS$_{\text{f}}$ defined in Definition 4.13 if $n \leq \frac{m}{2}$ where $m$ is a power of 2 and $B_q$ is a centered binomial distribution.
7 Conclusion

In this work, we have presented a novel and practical approach to address post-quantum secure stealth addresses. Along the way, we demonstrate its potential applications, such as privacy-preserving payments and passwordless authentication schemes like FIDO. We have also introduced a generic method to transform standard security without key-exposure resistance into a robust security solution capable of withstanding key-exposure attacks. Additionally, we have explored post-quantum fuzzy message detection (fuzzy tracking) and proposed two potential constructions. Future work and open problems include exploring the integration of our approach into RingCT-like frameworks [EZS+19, ESZ22] and investigating methods to reduce the signature size based on our current results.

Acknowledgement

Nico Döttling is funded by the European Union (ERC, LACONIC, 101041207). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References


[domb] How to donate crypto.

[donc] Why donate bitcoin, ethereum, nfts and other cryptocurrencies to charity.

[eco] Cryptocurrency and e-commerce.

[ega] Cryptocurrency and online gaming.


[fre] Digital currency donations for freedom convoy evading seizure by authorities.


[imp] https://github.com/sihangpu/SPIRIT.


[ste] Untraceable transactions which can contain a secure message are inevitable. 2011.


[umb] Umbra: Privacy preserving stealth payments.

[use] How many people own and use bitcoin?


A Discussions about Quantum Random Oracles

Since our goal is a practically efficient construction, we mainly focus on post-quantum security in the classical random oracle setting which is in the same spirit as in related works such as [LLN+20, EZS+19, ESZ22], etc. However, our protocols are highly likely to be secure even in the QROM setting. In more detail, for unforgeability, we can either follow the same strategy in Section 4.5 of [KLS18] to argue the EUF-CMA security in the QROM setting or apply the lifting theorem 1.1 and 1.2 from [YZ21] to establish the reduction from EUF-NMA in classical random oracle model to EUF-CMA in the QROM setting; for unlinkability, according to [GMP22], our adapted Kyber is already ANO-CCA secure in the QROM setting, and we only program the random oracle in a non-adaptive way as in the unforgeability game, thus the security in QROM is supposed to be preserved. We will add this discussion to the paper.

B Performance Analysis

We present the performance result in Table 3 and Table 4.

**Implementation.** We implement the **SPIRIT**, post-quantum FMD, and scalable fuzzy tracking schemes in C, and the open-source code of our proof-of-concept implementation can be found at [imp]. Specifically, we choose the anonymized variant of Kyber [SAB+22] to instantiate the KEM for building **SPIRIT**: We replace the original FO transform of Kyber with the one suggested in [GMP22], which makes Kyber ANOCCA-secure.

To instantiate the generic transformation in Section 5 to make **SPIRIT** secure against key-exposure attacks, we consider using Dilithium or Falcon [PFH+22] as the additional digital signature. Combining **SPIRIT** with Dilithium results in better efficiency but a slightly larger signature size; on the other hand, combining **SPIRIT** with Falcon leads to the most compact signatures but significantly longer key generation time.

For **SPIRIT** in Section 6.1, similar to Dilithium, we denote the scheme with three security levels as **SPIRIT**₂, **SPIRIT**₃, and **SPIRIT**₅. Parameters are the same as Dilithium’s, except that our β, γ₁, γ₂ are doubled.

For post-quantum FMD in Section 6.2, to achieve 104-bit computational security and 40-bit statistical security, we choose q = 2⁶⁰, ℓ = 2304, and χ = B₉ is a binomial distribution with parameter η = 3.

For Scalable Fuzzy Tracking in Section 6.3, to attain 115-bit security and negligible failure probability, we choose q = 4096; other parameters are the same as Kyber512, specifically, we have m = 256, η = 3, ℓ = 2.

**Environment.** We run the implementation on a standard laptop: Macbook Air (M1 2020) with 8GB RAM and a 2.1 GHz CPU (Turbo 3.2 GHz). It is important to note that our implementation is based on the reference implementation of Dilithium, Kyber, and Falcon, without using AES or AVX optimization. We perform each test 10,000 times to calculate the average running time. For post-quantum FMD, we run tests 100 times to obtain the average running time.

Experimental results demonstrate that Falcon512+**SPIRIT**₂ provides the smallest signature size (4.09 KB) for security against key-exposures with a decent hardness level (114-bit security). Additionally, Scalable Fuzzy Tracking offers the smallest communication cost (800 Bytes) and server’s computational overhead (3.4 ms) for millions of clients.

**Prior Works.** We provide comparison tables with prior works in Table 1 and Table 2.

In Table 1, we compare our group-based stealth signature (Appendix D), **SPIRIT**₂ (Section 6.1), **SPIRIT**₂+Dilithium2, and **SPIRIT**₂+Falcon512 with previous works. It is important to note
that [LLN+20] is a theoretical work without concrete parameters. We estimate the parameters based on the information provided in the paper. For a more in-depth analysis of the estimated parameters, please refer to the original text.

If we aim to improve their work with the recent advancements in NTRU, it is worth mentioning that the techniques used in [LLN+20] are derived from [ABB10], which implies HIBE. Combining it with NTRU could potentially enhance its efficiency. However, it is likely to have parameters similar to the state-of-the-art NTRU-based HIBE [ZMS+21]. Thus, we estimate the parameters here based on [ZMS+21] for 80-bit security, as they only provide two levels of security (80-bit or 160-bit).

In Table 2, we compare our Post-quantum FMD (Section 6.2) and Scalable Fuzzy Tracking (Section 6.3) with previous works on message detection or retrieval. All these works assume a semi-honest server, except for ΠTEE, which also considers a malicious server. Note that for security, ρ needs to be as large as $\frac{1}{\sqrt{M}}$ as calculated in [SPB21]. Additionally, some prior works consider fuzzy schemes ([BLMG21] and ours) as ρM-anonymity, where M is the total number of messages. However, this is not accurate due to statistical attacks as shown in [SPB21]: even with only one message (M = 1), some extent of anonymity is maintained if N is large.

Regarding the server’s workload, we compare the results for a single server with a single thread, as all works (except for ΠCC) support distributed servers or parallelized threads. [BLMG21] requires running their $\text{test}$ functionality for each recipient’s detection key for each incoming message. Other schemes with full privacy inherently demand $O(N)$ work from the server; otherwise, information leakage will occur.

Latency per message is dominated by the server’s computational time. Assuming there are $N = 2^{20}$ clients (a reasonable assumption for cryptocurrencies [use]) and setting the false-positive

<table>
<thead>
<tr>
<th>Scheme</th>
<th>w/KE</th>
<th>$\text{sec}$</th>
<th>$\text{sec}_{\text{opk}}$</th>
<th>Signature</th>
<th>PKEGen</th>
<th>Track</th>
<th>OSKEGen</th>
<th>Sign</th>
<th>Vf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Post-quantum FMD</td>
<td>104</td>
<td>94</td>
<td>$2^{20}$</td>
<td>$2^{-10}$</td>
<td>345.6 KB</td>
<td>17.2 KB</td>
<td>108.8 ms</td>
<td>75.13 ms</td>
<td>11.74 sec</td>
</tr>
<tr>
<td>Post-quantum FMD</td>
<td>104</td>
<td>94</td>
<td>$2^{20}$</td>
<td>$2^{-15}$</td>
<td>518.4 KB</td>
<td>17.2 KB</td>
<td>124.3 ms</td>
<td>74.64 ms</td>
<td>4.772 hour</td>
</tr>
<tr>
<td>Scalable Fuzzy Tracking</td>
<td>115</td>
<td>104</td>
<td>$2^{20}$</td>
<td>$2^{-10}$</td>
<td>580 B</td>
<td>800 B</td>
<td>0.01 ms</td>
<td>0.0348 ms</td>
<td>3.424 ms</td>
</tr>
<tr>
<td>Scalable Fuzzy Tracking</td>
<td>115</td>
<td>104</td>
<td>$2^{20}$</td>
<td>$2^{-15}$</td>
<td>800 B</td>
<td>800 B</td>
<td>0.01 ms</td>
<td>0.0149 ms</td>
<td>108.77 ms</td>
</tr>
</tbody>
</table>

1. Post-quantum FMD is based on Dilithium2 and anonyzied Kyber512. Spirit3 on Dilithium3 and Kyber768, and Spirit5 on Dilithium5 and Kyber1024. Falcon+Spirit indicates the use of an additional Falcon as DS in the generic transformation, while Dilithium+Spirit signifies the use of an additional Dilithium as DS in the generic transformation.

2. $\text{sec}_{\text{opk}}$ represents the hardness of Classical Core-SVP, while $\text{sec}$ denotes the hardness of Quantum Core-SVP.
rate $\rho = 2^{-10}$ for [BLMG21] and ours. The numbers of other works are taken directly from their papers. With an assumption of $10 \sim 20$ messages per second (e.g., Bitcoin or Ethereum), only our scheme is practical with many users.

To compute each recipient’s computational time, we assume a total of $M = 500,000$ messages.

C Additional Preliminaries

C.1 Assumptions

Definition C.1 (Cyclotomic Polynomial). We denote by $R$ the ring $\mathbb{Z}[X]/(X^n + 1)$ and by $R_q$ the ring $\mathbb{Z}_q[X]/(X^m + 1)$, where $m = 2^{m'} - 1$ such that $X^m + 1$ is the $2m'$-th cyclotomic polynomial $\Phi_{2m'}(X)$. Moreover, we have

$$\prod_{d|m} \Phi_d(X) = X^m - 1.$$ 

Definition C.2 (Learning with Errors (LWE) [Reg05]). For a vector $s \in \mathbb{Z}_q^n$ called the secret, the LWE distribution $A_s,\chi$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ is sampled by choosing $a \in \mathbb{Z}_q^n$ uniformly at random, choosing $e \leftarrow \chi$, and outputting $(a, b = \langle a, s \rangle + e \mod q)$. Moreover, decisional-LWE $n,m,q,\chi$ is

$$\text{Adv}_{\text{lwe}} = \Pr[b = 1 | A \leftarrow R_{m \times n q}, t \leftarrow R_{m q}; b = A(A, t)] - \Pr[b = 1 | A \leftarrow R_{m \times n q}, s \leftarrow R_{m q}, e \leftarrow \chi, b = A(A, As + e)]$$

Definition C.3 (Module Learning With Errors MLWE [BGV12]). For integers $m, k$, and a probability distribution $D : R_q \rightarrow [0,1]$, we say that the advantage of algorithm $A$ in solving the decisional MLWE $m,k,D$ problem over the ring $R_q$ is

$$\text{Adv}_{\text{mlwe}} = \Pr[b = 1 | A \leftarrow R_{m \times n q}, t \leftarrow R_{m q}; b = A(A, t)] - \Pr[b = 1 | A \leftarrow R_{m \times n q}, s_1 \leftarrow D, s_2 \leftarrow D; b = A(A, As_1 + s_2)]$$

Definition C.4 (Module Short Integer Solution MSIS [Ajt98]).

$$\text{Adv}_{\text{msis}} = \Pr[0 < \|y\|_\infty < \gamma \land \|I A \cdot y\| = 0 | A \leftarrow R_{m \times n q}; y \leftarrow A(A)]$$

Definition C.5 (The SelfTargetMSIS Problem in [LDK+20]). Suppose that $H : \{0,1\}^* \rightarrow B_r$ is a cryptographic hash function. To an algorithm $A$ we associate the advantage function

$$\text{Adv}_{\text{selftargetmsis}} = \Pr[0 < \|y\|_\infty < \gamma \land H(\mu, I A) \cdot y = c | A \leftarrow R_{m \times k}; (y := \begin{bmatrix} r \\ c \end{bmatrix}, \mu) \leftarrow A^{(\cdot)(\cdot)}(A)]$$
\( K' \leftarrow \text{KEM} \cdot \text{Gen}(\lambda) \)

\( b \leftarrow \{0, 1\} \)

\( (C^*, K^*) \leftarrow \text{KEM} \cdot \text{Encaps}(ek_0) \)

\( b' \leftarrow \mathcal{A} \cdot \text{Decaps}(\cdot, \cdot)(ek_0, ek_1, C^*, K^*) \)

\( b_0 := (b = b') \)

\( \text{return } b_0 \)

Figure 11: Experiment for ANOCCA\textsubscript{KEM}(\lambda)

### C.2 Cryptographic Tools

**Definition C.6** (Binomial Distribution\[SAB^+20\]). We define the binomial distribution \( B_\eta \) as follows:

\[ (a_1, \ldots, a_\eta, b_1, \ldots, b_\eta) \leftarrow \{0, 1\}^{2\eta}, \]

and then output \( \sum_1^\eta a_i - b_i \). If we write some polynomial \( f \leftarrow B_\eta \), then each coefficient of \( f \) is sampled from \( B_\eta \).

**Definition C.7** (Anonymous KEM\[GMP22\]). A KEM is said to be anonymous under chosen-ciphertext attacks if there exists a negligible function \( \text{negl}(\lambda) \) for all \( \lambda \in \mathbb{N} \), and for all adversaries \( \mathcal{A} \) the following holds:

\[ \Pr[A \cdot \text{ANOCCA}^A_\Lambda(\lambda) = 1] \leq \frac{1}{2} + \text{negl}(\lambda) \]

where ANOCCA is defined in Figure 11. Similarly, we also define \( \text{IKCPA} \) experiment for PKE in Figure 12, which just removes access to the decryption oracle\[BBDP01\].

**Definition C.8** (Uniformly-Ambiguous Encryption\[BLMG21\]). Let \( \text{PKE} := (\text{Gen}, \text{Enc}, \text{Dec}) \) be a public-key encryption scheme for the message space \( \{0, 1\}^n \). For any \( \lambda \in \mathbb{N} \), uniformly sampled message \( m \leftarrow \{0, 1\}^n \), we say \( \text{PKE} \) is UNIAMB-secure if

\[ \text{Adv}^\text{uniamb}_\Lambda(\mathcal{A}) := \left| \Pr[\text{UNIAMB}^A_{\text{PKE}}(\lambda) = 0] - \Pr[\text{UNIAMB}^A_{\text{PKE}}(\lambda) = 1] \right| \leq \text{negl}(\lambda), \]

where the experiment \( \text{UNIAMB}^A_{\text{PKE}}(\lambda) \) is defined in Figure 12.

### C.2.1 Statistical Tools

**Definition C.9** (Statistical Distance). The statistical distance between two probability distributions \( A \) and \( B \) is

\[ \text{SD}(A, B) = \frac{1}{2} \sum_v \left| \Pr[A = v] - \Pr[B = v] \right|. \]

Recall min-entropy of a random variable \( A \) is

\[ H_\infty(A) := -\log(\max_a \Pr[A = a]), \]

then we have the following lemma.
Lemma C.1 (Leftover Hash Lemma[Ill89]). Assume a family of functions \( \{H_x: \{0,1\}^n \rightarrow \{0,1\}^m\}_{x \in X} \) is universal: \( \forall \alpha \neq \beta \in \{0,1\}^n, \Pr_{x \in X}[H_x(\alpha) = H_x(\beta)] = 2^{-m} \). Then, for any random variable \( W \),
\[
SD((H_X(W), X), (U_m, X)) \leq \epsilon,
\]
whenever \( m \leq k - 2 \log(\frac{1}{\epsilon}) + 2 \) and \( k = H_\infty(W) \).

Lemma C.2 (Rank of the Circulant Matrix[Ing56]). The rank of a circulant matrix \( C \) of order \( m \) is \( m - \ell \), where \( \ell \) is the degree of the greatest common divisor of \( X^m - 1 \) and the associated polynomial of \( C \).

## D Group-based Construction against Bounded Leakage

We provide an SS which is unforgeable and unlinkable with bounded key-exposure. The construction is shown in Figure 13, where \( G \) is a group of prime order \( p \), \( g \) is a generator, and \( H \) is a random oracle mapping from \( G \) to \( \mathbb{Z}_p \). Additionally, DS is an efficient group-based signature scheme such as ECDSA, Schnorr and others whose verification key and signing key has discrete logarithm relation, i.e., \( \text{vk} = g^{\text{sk}} \).

**Correctness.** It is clear that \( \text{opk} = g^{\text{osk}} \) as
\[
\text{opk} = \prod_{i=1}^{\ell} h_i^{H(h_i^c)} = g^{\sum_{i=1}^{\ell} x_i H(g^{-x_i})} = g^{\text{osk}}.
\]

Tracking mechanism also works since
\[
R_0 = \text{opk}^{H(h_0^c)} = \text{opk}^{H(g^{-x_0})}.
\]

**Security Analysis.** Now we analyze the security of above construction.

**Theorem D.1.** The construction in Figure 13 is (strongly) unforgeable and unlinkable with \( (n-1) \)-bounded key exposures.

**Proof.** (sketch) For unlinkability, without knowing \( x_i \) or \( r_i \), by DDH assumption, the triple \( g^r, g^{x_i}, g^{r-x_i} \) remains uniformly random over \( G \). With random oracle \( H \), \( H(g^{-x}) \) is also uniformly random over \( \mathbb{Z}_p \). Therefore, it is clear that \( \text{opk}, R, R_0 \) are uniformly random.

For unforgeability, as long as DS is (strongly) unforgeable, then SS is also (strongly) unforgeable.
<table>
<thead>
<tr>
<th>MKGen(λ)</th>
<th>OPKGen(mpk)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample G = (g) (x₀, x₁, ..., xₙ) ← $Z_p$</td>
<td>parse (g, h₀, ..., hₙ) := mpk r ← $Z_p$</td>
</tr>
<tr>
<td>mpk := (g, h₀ := g^x₀, ..., hₙ := g^xₙ)</td>
<td>opk := $\prod_{i=1}^{n} h_i^{r_i}$</td>
</tr>
<tr>
<td>msk := (x₀, ..., xₙ)</td>
<td>tki := (R := g^r, R₀ := opk^{H(h₀)})</td>
</tr>
<tr>
<td>mtk := x₀</td>
<td>return opk, tki</td>
</tr>
<tr>
<td>return mpk, msk, mtk</td>
<td>Sign(osk, m)</td>
</tr>
<tr>
<td></td>
<td>return $\sigma$ := DS.Sign(osk, m)</td>
</tr>
<tr>
<td>OSKGen(msk, opk, tki)</td>
<td>Track(mtk, opk, tki)</td>
</tr>
<tr>
<td>parse (x₀, ..., xₙ) := msk</td>
<td>parse (R, R₀) := tki</td>
</tr>
<tr>
<td>parse (R, R₀) := tki</td>
<td>parse x₀ := mtk</td>
</tr>
<tr>
<td>return ⊥ if false ← Track(x₀, ..., opk, tki)</td>
<td>return opk^{H(R₀)} &amp; R₀</td>
</tr>
<tr>
<td>osk := $\sum_{i=1}^{n} x_i \cdot H(R^{x_i})$</td>
<td></td>
</tr>
<tr>
<td>return osk</td>
<td></td>
</tr>
<tr>
<td>Vf(opk, σ, m)</td>
<td>Vf(opk, σ, m)</td>
</tr>
<tr>
<td>return DS.Vf(opk, σ, m)</td>
<td></td>
</tr>
</tbody>
</table>

Figure 13: Construction of group-based SS secure with (n − 1)-bounded key-exposure

Now we consider key-exposures. Since

\[ osk = \sum_{i=1}^{n} x_i \cdot H(h_i^r), \]

this is an equation with n variables (xᵢ) for adversaries. If the adversary learns at most n − 1 equations, then this linear system is undetermined and has at least p solutions which is exponentially large. Thus msk is hiding when there are at most (n − 1) key-exposures.

E Security Analysis of Generic Transform

Proof of Theorem 5.1 We restate the theorem here more formally for the case of unforgeability.

Theorem E.1. The stealth signature SSₜ constructed in Section 5 is secure in sEUFCMAₜ−ke experiment if SSₜ/₀ is EUFCMAₜ/₀−ke secure and DS is sEUFCMA secure. Specifically, for any $\lambda \in \mathbb{N}$, and for any PPT adversary $A$, if it succeeds in the experiment sEUFCMAₜ−ke, then there are other adversaries $B₁, B₂$ running in roughly same time such that

\[ \text{Adv}^{\text{seufcma}_{\text{t−ke}}} \left( A \right) \leq \text{Adv}^{\text{euucma}_{\text{t/₀−ke}}} \left( B₁ \right) + \text{Adv}^{\text{seufcma}} \left( B₂ \right). \]

Proof. We prove the theorem by reduction. Suppose there’s an adversary $A$ has non-negligible advantage in sEUFCMAₜ−ke, then we can construct another adversary $B$ to win the experiment EUFCMAₜ/₀−ke of SSₜ/₀ or the experiment sEUFCMA (strong unforgeability) of DS as follows. $B$ forwards mpk, mtk from the challenger in EUFCMAₜ/₀−ke to $A$. 

34
To simulate \( \text{OSKGenO}(i, \text{opk}, tk_i, \text{flag}^i) \), if \( \text{flag}^i = \text{true} \), \( B \) runs \((\text{vk}^i, \text{sk}^i) \leftarrow \text{DS.Gen}\), then queries \( \sigma_1^i \leftarrow \text{SignO}(\text{vk}^i) \) in \( \text{EUFCMA}_{w/o-ke} \) and returns \( \text{osk}^i := (\sigma_1^i, \text{vk}^i, \text{sk}^i) \) to \( A \). If \( \text{flag}^i = \text{false} \), \( B \) asks a challenger \( C' \) in \( \text{sEUFCMA} \) of DS to send a challenge verification key \( \text{vk}^i \), then queries \( \sigma_1^i \leftarrow \text{SignO}(\text{vk}^i) \) in \( \text{sEUFCMA}_{w/o-ke} \) of \( \text{SS}_{w/o} \) and stores \( \sigma_1^i \); If \( \text{OK}[i] = (\text{opk}_i^i, \cdot, \cdot) \land \text{flag}^i = \text{true} \), \( B \) signals \( C' \) to terminal the experiment and asks for its \( \text{osk}^i \) then forwards that to \( A \). To simulate \( \text{SignO}(i, m') \), \( B \) queries \( \sigma_2^i \leftarrow \text{SignO}(m'||\sigma_1^i) \) and returns \( \sigma^i := (\sigma_1^i, \sigma_2^i, \text{vk}^i) \) to \( A \).

Once \( A \) submits some valid forgery \( \sigma^i := (\sigma_1^i, \sigma_2^i, \text{vk}^i), m^i, i' \) as shown in Figure 3, \( B \) behaves in following cases:

- If \( m' \) is not appeared in \( Q \) (recall that \( Q \) is the set to record signing queries), \( B \) forwards \( \sigma_2^i, m'||\sigma_1^i \) to \( i' \)-th challenger in \( \text{sEUFCMA} \) of DS;
- If \( \text{vk}^i \) is not appeared in \( Q \), \( B \) forwards \( \sigma_1^i, \text{vk}^i \) to the challenger in \( \text{EUFCMA}_{w/o-ke} \);
- If both \( m', \text{vk}^i \) are in \( Q \), then the only case that \( \sigma^i \) is a valid forgery is either \( \sigma_1^i \) or \( \sigma_2^i \) not appeared in \( Q \). In either case, \( B \) just forwards \( \sigma_2^i, m'||\sigma_1^i \) to the challenger in \( \text{sEUFCMA} \) of DS.

This completes the proof.

We restate the theorem here for unlinkability.

**Theorem F.2.** The stealth signature \( \text{SS}_{w} \) constructed in Section 5 is secure in \( \text{UNLNK}_{w-ke} \) experiment if \( \text{SS}_{w/o} \) is \( \text{UNLNK}_{w-ke} \) secure. Specifically, for any \( \lambda \in \mathbb{N} \), and for any PPT adversary \( A \), if it succeeds in the experiment \( \text{UNLNK}_{w-ke} \), then there are other adversaries \( B \) running in roughly same time such that

\[
\text{Adv}_\lambda^{\text{UNLNK}_{w-ke}}(A) \leq \text{Adv}_\lambda^{\text{UNLNK}_{w-ke}}(B).
\]

**Proof.** Similarly, we can also prove this theorem easily by reduction. Suppose there’s an adversary \( A \) has non-negligible advantage in \( \text{UNLNK}_{w-ke} \), then we can construct another adversary \( B \) to win the experiment \( \text{UNLNK}_{w/o-ke} \) of \( \text{SS}_{w/o} \) as follows. \( B \) forwards \( \text{mpk}_i, \text{mpk}_j, \text{opk}_i, \text{tk}_i \) from the challenger in \( \text{UNLNK}_{w/o-ke} \) to \( A \). To simulate \( \text{osk}_i \), \( B \) runs \( \text{DS.Gen} \) to get \((\text{vk}, \text{sk})\), then queries the signing oracle \( \text{SignO}(\cdot, -1, \text{vk}) \) from \( \text{UNLNK}_{w/o-ke} \) to learn a signature \( \sigma_1 \) of \( \text{vk} \), then returns \( \text{osk}_i := (\sigma_1, \text{vk}, \text{sk}) \) to \( A \). To simulate \( \text{OSKGenO} \), \( B \) queries \( \text{SignO} \) and runs \( \text{DS.Gen} \) as above to generate \( \text{osk} \). Once \( A \) submits \( b' \), \( B \) simply forwards \( b' \) as its final guess. This completes the proof.

\[\blacksquare\]

## F Security Analysis of Stealth Signature Without Fuzzy Tracking

**Proof of Lemma 6.1** We restate the lemma formally here.

**Lemma F.1.** SPIRIT in Figure 8 is unforgeable without key exposure under no-message attacks. Specifically, in random oracle model, for any \( \lambda \in \mathbb{N} \), for any adversary \( A \), if Dil has parameters \( \beta, \gamma_1, \gamma_2 \), and we denote \( \text{H}^\prime \) as a random oracle can be accessed by \( A \) and \( B_2 \), then the advantage to win the game \( \text{UFNMA}_{w/o-ke}^{\lambda}(\beta) \) is

\[
\text{Adv}_{\lambda, \text{H}^\prime, \gamma_1, \gamma_2, \beta}^{\text{UFNMA}_{w/o-ke}}(A) \leq \text{Adv}_{\lambda, k, f, D}^{\text{mwe}}(B_1) + \text{Adv}_{\text{H}^\prime, k, f+1, \zeta}^{\text{selftargetmsis}}(B_2).
\]

**Proof.** Consider the experiment \( \text{EUFCMA}_{w/o-ke} \) in Figure 1 where the \( \text{SignO} \) is forbidden to access. Suppose \( A \) forges \( \sigma^* \), then we have the following claim.
Claim 1. If an adversary $A$ can forge $\sigma^*$ without accessing $\text{Sign}_O$ and assuming $\text{MLWE}_{k,\ell,D}$ assumption holds, then there is another adversary $B_2$ who solves $\text{SelfTargetMSIS}_{H,k,\ell+1,\zeta}$ in roughly same time with non-negligible probability.

Proof. After receiving uniformly random samples $(A, t) \in R^{k+\ell} \times R^k$ and random oracle access $H'(\cdot)$ from the challenger in $\text{SelfTargetMSIS}_{H,k,\ell+1,\zeta}$, $B_2$ computes $mpk := (A, t, ek)$, mtk := $(dk, t)$ and forwards $mpk$, mtk, $H'$ to $A$. As long as the $\text{MLWE}_{k,\ell,D}$ assumption holds, $mpk$ looks indistinguishable from real public key for $A$. For $i$-th query in $\text{OSKGen}_O$, $B_2$ computes and stores $s^*_i, s^*_2$. Once $A$ submits some valid forgery $\sigma^*$ with $i^*$, meaning it finds some $(x, z, c)$ for $opk^* := t^*$ such that

$$H'\left(\mu \parallel I_k | A | t^* \parallel \begin{bmatrix} x \  z \\ c \end{bmatrix}\right) = c,$$

where $\|x\|_\infty \leq 2 \gamma_2 + 1 + 2^{d-1}\tau, \|z\|_\infty \leq \gamma_1 - 2\beta$ and $\|c\|_\infty = 1|\text{LDK} + 20|$. Then $B_2$ can retrieve $s^*_1, s^*_2$ from its storage and instantly return $y := \begin{bmatrix} x' \\ z' \ c \end{bmatrix}$, $\mu$ to the $\text{SelfTargetMSIS}_{H',k,\ell+1,\zeta}$ challenger, where $x' := x + cs^*_1$ and $z' := z + cs^*_1$. Note that $\|cs^*_1\|, \|cs^*_1\| \leq \beta$. Since we can write $t^* := t + As^*_1 + s^*_2$, it is easy to check that this is a valid solution

$$H'\left(\mu \parallel I_k | A | t \parallel \begin{bmatrix} x + cs^*_1 \  z + cs^*_2 \ c \end{bmatrix}\right) = c$$

where $\|y\|_\infty \leq \zeta$ and $\zeta := \max\{\gamma_1 - \beta, 2\gamma_2 + 1 + 2^{d-1}\tau + \beta\}$. This completes the proof to show it is secure in $\text{UFNMA}_{w/o-ke}$ experiment.

Proof of Theorem 6.2 We restate the theorem for unforgeable without key exposures formally here.

Theorem F.2. $\text{SPIRIT}$ in Figure 8 is existential unforgeable without key exposures. Specifically, for any adversary $A$, if it succeeds in the experiment $EUF\text{CMA}_{w/o-ke}$, then there is another adversary $B$ running in roughly same time such that

$$\text{Adv}_{\text{EUF\text{CMA}_{w/o-ke}}}(A) \leq \text{Adv}_{\text{UFNMA}_{w/o-ke}}(B) + \text{negl}(\lambda),$$

where we denote $H', H$ as random oracles can be accessed by $B_1$ and $A$, respectively.

Proof. Intuitively, reduction from CMA to NMA usually needs “patching” random oracles [KLS18, AFLT12]. We prove this theorem in a sequence of hybrid games as follows.

Hybrid$_0$: This is exactly the standard $\text{EUF\text{CMA}_{w/o-ke}}$ experiment. Thus we have

$$\Pr[\text{Hybrid}_0 \Rightarrow 1] = \text{Adv}_{\text{EUF\text{CMA}_{w/o-ke}}}^\text{EUF\text{CMA}_{w/o-ke}}(A).$$

Hybrid$_i$: We modify Hybrid$_0$ as follows. In $\text{OSKGen}_O(opk^i, tki^i)$, for $i$-th query, if true $\leftrightarrow \text{Track}(mtk, opk^i, tki^i)$ it only stores $s^*_1, s^*_2, t^* := As^*_1 + s^*_2 + t$, sets $\text{osk}^i := T$ and returns 1. In $\text{Sign}_O(i, m^j)$, for $j$-th query, it generates and sets $\text{osk}^j$ by $\text{msk}$ if $\text{osk}^j := T$, then return a signature.
\[ H(w_1||\mu) \]

| \text{EUFCA}_3^{\text{A}}_{w/o-ke}(\lambda) | (\text{mpk, msk, mtk}) \leftarrow \text{MKGen}(\lambda) \\
| \text{OK} := [], Q := \emptyset \\
| (m^*, \sigma^*, i^*) \leftarrow A^{\text{OSKGenO,SignO}}(\text{mpk, mtk}) \\
| (\text{opk}^* := t^*, \text{osk}^*, :) \leftarrow \text{OK}[i^*] \\
| \text{Hyb}_2 \text{ block begins} \\
| (z^*, c^*, h^*) := \sigma^* \\
| \mu^* := G(m^*||t^*) \\
| w_1^* := \text{HighBits}_q(Az^* - c^*t^*, 2\gamma_2) \\
| \text{if } H(w_1^*||\mu^*) \neq c^* \\
| \text{then return } 0 \\
| \text{Hyb}_2 \text{ block ends} \\
| b_0 := (m^*, i^*) \notin Q \\
| b_1 := \text{Vf}(\text{opk}^*, m^*, \sigma^*) = 1 \\
| b_2 := (\text{OK}[\mu^*] \neq \cdot, \cdot, \cdot) \\
| \text{return } b_0 \land b_1 \land b_2 \\

\[ \text{Figure 14: Simulated } H \text{ and } \text{EUFCA}_3^{\text{A}}_{w/o-ke}(\lambda) \text{ in } \text{Hybrid}_2 \text{ and } \text{Hybrid}_3 \]

\( \sigma^j \) by using osk\(^j\). This game only changes the time to generate osk\(^j\), thus advantage remains the same:

\[ | \Pr[\text{Hybrid}_1 \Rightarrow 1] - \Pr[\text{Hybrid}_3 \Rightarrow 1] | = 0. \]

**Hybrid\(_2\):** We update Hybrid\(_1\) by modifying Sign\(_O\)(i, m\(^j\)) in j-th query: Instead of generating \( \sigma^j \) with osk\(^j\) when needed, it just simulates \( \sigma^j \) by choosing uniformly random \( (z^j, c^j) \in S_{\gamma_1-23-1}^j \times B_7 \) and stores a key-value pair \( (\mu^j : (c^j, w_1^j)) \) where \( \mu^j \leftarrow G(m^j||t^j) \), \( w_1^j \leftarrow \text{HighBits}_q(Az^j - c^jt^j, 2\gamma_2) \), and \( G \) is a perfect random function. We also use a new random oracle \( H(w_1||\mu) \) to simulate random oracle \( H(w_1||\mu) \) in above game as shown in left part of Figure 14. Now we analyze the advantage. In our construction, Dil\(\cdot\)Sign remains unaltered, thus the resulting signature \( \sigma \) is still perfectly zero-knowledge (where the exact simulation is shown in Sign of Figure 15). Therefore the distribution of each \( \sigma \) is exactly the same as the one in Hybrid\(_1\), then we have

\[ | \Pr[\text{Hybrid}_2 \Rightarrow 1] - \Pr[\text{Hybrid}_1 \Rightarrow 1] | = 0. \]

**Hybrid\(_3\):** We modify the above game by adding an additional block in EUFCA\(_3^{\text{A}}_{w/o-ke}\) as shown in right part of Figure 14. This game only differs from the Hybrid\(_2\) if \( w_1^* = w_1^j \) and \( (m^*, \cdot, i^*) \notin Q \) \( \land \) \( b_1 \land b_2 \) (Hybrid\(_3\) return 0 and Hybrid\(_2\) return 1). However, A didn’t query Sign\(_O\)(i, m\(^*\)) before, thus \( w_1^j \) should remain hidden. And from [KLS18], it shows that Dil signature has enough min-entropy, thus the probability \( \Pr[w_1^j = w_1^i] \) is negligible, i.e.,

\[ | \Pr[\text{Hybrid}_3 \Rightarrow 1] - \Pr[\text{Hybrid}_2 \Rightarrow 1] | \leq \text{negl}(\lambda). \]

This game can be fully simulated by \( B \) against UFNMA\(_{w/o-ke}\) as follows. \( B_1 \) simulates OSKGen\(_O\), Sign\(_O\) oracles without knowing msk, and it patches \( H' \) from UFNMA\(_{w/o-ke}\) to \( H \) for generating \( \sigma^i \). Once A submits a valid signature \( \sigma^* \) and if \( H' \) works well in \( \sigma^* \), \( B \) directly forwards \( \sigma^* \) to the challenger of UFNMA\(_{w/o-ke}\). Therefore

\[ \Pr[\text{Hybrid}_1 \Rightarrow 1] = \text{Adv}_{\lambda, H', \gamma_1, \gamma_2, \beta}(B_1) \]

and we completes the proof.
Theorem F.3

We denote \( H \) as a random oracles can be accessed by \( A \) and \( \gamma_1, \gamma_2, \beta \) are parameters of the underlying DL scheme.

**Proof.** We prove the theorem in a sequence of hybrid games.

**Hybrid\(_0\):** This is the original UNLNK\(_{w→ke}\) experiment, thus we have

\[
\Pr[\text{Hybrid}_0 \Rightarrow 1] = \text{Adv}^{\text{unlink}_{w→ke}}_{\lambda, H, \gamma_1, \gamma_2, \beta}(A).
\]

**Hybrid\(_1\):** We modify the above game by changing the function \( \text{Sign}(\text{osk}, m^j) \) in UNLNK\(_{w→ke}\) experiment to the \( \text{Sign}(\text{opk}, m^j) \) without using osk in Figure 15. Specifically, it samples uniformly random \((z', c')\), programs the random oracle such that \( H(\mu^j || w_1^j) = c' \) where \( \mu^j \) is determined by \( m^j \) and \( w_1^j := \text{HighBits}(Az' - c't' + c_i t'_0, 2\gamma_2) \). Then set \( \sigma^j := (z', c', h^j) \) where \( h^j \) can be determined by \( c', t', z' \). Because of the perfectly zero-knowledge of \( \sigma^j \), the distribution of signatures in this hybrid is the same as the one in \( \text{Hybrid}_0 \), i.e.,

\[
\Pr[\text{Hybrid}_0 \Rightarrow 1] - \Pr[\text{Hybrid}_1 \Rightarrow 1] = 0.
\]

**Hybrid\(_2\):** We modify the above game by follows. Parse \( \text{mpk}_0 := (t_0, \text{ek}_0) \) and \( \text{mpk}_1 := (t_1, \text{ek}_1) \), instead of generating \( t_0, t_1 \) from msk, we sample uniformly random \((t_0, t_1) \leftarrow S_{\gamma_1 - 2\beta_1} \times B_r \) and \( t'_0 \leftarrow G(m^j || t') \). This is the original \( \text{Sign}(\text{opk}, m^j) \) in \( \text{Hyb}_1 \) and \( \text{Hyb}_2 \), and \( \text{mpk}_0 := (t_0, \text{ek}_0) \) and \( \text{mpk}_1 := (t_0, \text{ek}_1) \) where \((t_0, t_1) \leftarrow S_{\gamma_1 - 2\beta_1} \times B_r \) are uniformly sampled.

<table>
<thead>
<tr>
<th>Sign(opk^j, m^j)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \sigma^j := (z', c', h^j) )</td>
</tr>
<tr>
<td>( \mu^j \leftarrow G(m^j</td>
</tr>
<tr>
<td>( w'_1 \leftarrow \text{HighBits}(Az' - c't' + c_i t'_0, 2\gamma_2) )</td>
</tr>
</tbody>
</table>

**Figure 15:** Simulation of Sign from \( \text{Hybrid}_1 \) to \( \text{Hybrid}_2 \)

Besides, this hybrid can be fully simulated by an adversary \( B_1 \) of ANOCCA experiment. \( B_1 \) simulates the random oracle \( H \) for \( A \). Upon receiving \( \text{ek}_0, \text{ek}_1 \) and \((C_0, K_0)\) from ANOCCA experiment, \( B_1 \) sets \( \text{mpk}_0 := (t_0, \text{ek}_0) \) and \( \text{mpk}_1 := (t_0, \text{ek}_1) \) where \((t_0, t_1) \leftarrow S_{\gamma_1 - 2\beta_1} \times B_r \) are uniformly sampled.
We first recall the construction of packed Regev with ciphertext compression [BDGM19] in Figure 16.

\[ A \in \mathbb{Z}_q^{2\times n} \rightarrow \chi \]
\[ (S, E) \rightarrow \chi (\chi^{2n})^2 \]
\[ B := AS + E \]
\[ \text{return} \ pk := B, \ sk := S \]

\[ \text{pRgv.Gen}(\lambda, n) \quad \text{pRgv.Enc}(pk, m \in \{0, 1\}^n) \]
\[ (r, e_1) \rightarrow \chi (\chi^{2n})^2, c_2 \rightarrow \chi^n \]
\[ c_1 := A^* r + e_1, \]
\[ c_2 := B^* r + e_2 + \frac{q}{2} m \]
\[ z \rightarrow \mathbb{Z}_q \text{ such that } \forall i \in [n]: \]
\[ z + c_{2,i} \notin \left( \frac{q}{4} - B, \frac{q}{4} + B \right) \cup \left( \frac{3q}{4} - B, \frac{3q}{4} + B \right) \]
\[ \forall i \in [n], w_i := (z + c_{2,i})_2 \]
\[ \text{return} \ ct := (c_2, z, w_1, \ldots, w_n) \]

\[ \text{pRgv.Dec}(sk, ct) \]
\[ \text{parse} \ (c_1, z, w_1, \ldots, w_n) := ct \]
\[ \text{parse} \ (s_1, \ldots, s_n) := sk \]
\[ \forall i \in [t], m_i := [s_i^T c_1 + z]_2 \oplus w_i \]
\[ \text{return} \ m := [m_1, \ldots, m_t] \]

Figure 16: Packed Regev encryption pRgv with ciphertext compression

\[ B_1 \text{ sets } tki_b := C_b, \text{osk}_b := T, \text{opk}_b \leftarrow R_{\gamma}^{k} \text{ and sends } (mpk_0, mpk_1, tki_b, opk_b) \text{ to } A \text{ of Hybrid}_2. \]

For each query of OSKGenO(b*, opk*, tki*), B_1 queries K* \leftarrow \text{KEM}. DecapsO(b*, tki*) to check if \( \text{As}_i^1 + s_i^2 + t_{b,i} = \text{opk}^* \) where \( s_1^i, s_2^i \leftarrow \text{DilExpands}(K^*) \). If the check doesn’t pass, set \text{osk}_i^{b*} := T; Otherwise set \text{osk}_i^{b*} := \perp. For each query of \text{SignO}(b*, i, m^*), B_1 simulates the signature \( \sigma^i \) by using \text{opk}^*_i, if the corresponding \text{osk}_i^{b*} = T, otherwise return \perp. If \( i = -1 \), just simulates a signature using \text{opk}_b. \]

Then Hybrid_2 can be simulated without knowing any msk, mtik or b. Once A returns b', B_1 simply forwards b' to the challenger of ANOCCA. Thus we have

\[ \Pr[\text{Hybrid}_2 \Rightarrow 1] = \text{Adv}_{\text{anocca}}^*(B_1), \]

and this completes the proof.

\[ \square \]

### G Analysis of Post-quantum FMD

We first recall the construction of packed Regev with ciphertext compression [BDGM19] in Figure 16, where \( \chi \) is the error distribution and \( B \) is the error bound between \( z + c_{2,i} \) and \( z + s_i^T c_1 \).

Note that apart from the header \((c_1, z)\), the payload \((w_i)\) are just \( n \) bits which is almost as succinct as DLog-based fuzzy message detection scheme FMD_2 in [BLMG21]. Specifically, the entire ciphertext is \((\ell + 1) \log q + \tau\)-bit large.

**Correctness.** We show the scheme in Figure 9 satisfies Definition 4.8 as follows. For each \( i \in [t] \), we have \([s_i^T c_1 + z]_2 \oplus w_i = 1\). Since \( c_{2,i} - s_i^T c_1 = \frac{q}{2} + e' \) where \( e' \in [-B, B] \) is some short error, we have \( c_{2,i} - e' = s_i^T c_1 + \frac{q}{2} \). Also, we choose

\[ c_{2,i} + z \notin \left( \frac{q}{4} - B, \frac{q}{4} + B \right) \cup \left( \frac{3q}{4} - B, \frac{3q}{4} + B \right), \]

thus we have \([c_{2,i} + z]_2 = [c_{2,i} + z - e']_2\), which implies \( w_i = [s_i^T c_1 + z + \frac{q}{2}]_2 = [s_i^T c_1 + z]_2 \oplus 1\). Therefore, with correct ftk, FTrack always returns true. Note that for correctness, we require \( q > 4Bn \).

**Security Analysis.** We show the scheme in Figure 9 is unlinkable with key-exposure and fuzzy tracking (Definition 4.9).

**Proof of Lemma 6.3 and Theorem 6.4** We restate the formal lemma here.
Lemma G.1. Packed Regev encryption $\mathbf{pRgv}$ with ciphertext compression shown in Figure 16 satisfies Definition C.8 and is uniformly-ambiguous UNIAMB-secure when $\frac{4B}{q}$ is $\negl(\lambda)$. Specifically, we have

$$\text{Adv}_{\lambda}^{\text{uniamb}}(A) \leq \text{Adv}_{\lambda}^{\text{lwe}}(A) + \frac{4Bn}{q},$$

where $B$ is the bound such that $||S^Tc_1 - c_2||_\infty \mod \frac{2}{q} < B$.

Proof. To see it is uniformly-ambiguous, firstly note that $c_1$ looks uniformly random due to LWE assumption, and $w_i$ is uniformly random due to $m_i$ is a uniformly random bit in the experiment in Figure 12. For $z$, the statistical distance between its distribution and uniformly random distribution over $\mathbb{Z}_q$ is $\frac{4B}{q}$. Thus as long as $\frac{4B}{q} \leq \negl(\lambda)$, we can simulate the entire ciphertext without knowing $b$ or $sk$.

We restate the theorem formally here.

Theorem G.2. The fuzzy stealth signature constructed in Figure 9 is unlinkable with key-exposure and fuzzy tracking. Specifically, for any $\lambda, n, t$ where $n \geq t$, if there is a PPT adversary $A$ has non-negligible advantage in experiment defined in Figure 5, then there exist other adversaries $B_1, B_2, B_3$ running in roughly same time such that:

$$\text{Adv}_{\lambda, n, t}^{\text{unlnk-w}}(A) \leq 2\text{Adv}_{\lambda}^{\text{uniamb}}(B_1) + p(\lambda) \cdot (4t\text{Adv}_{\lambda}^{\text{uniamb}}(B_2) + (n-t)\text{Adv}_{\lambda}^{\text{ikcpa}}(B_3)),$$

where $p(\lambda)$ is some polynomial on security parameter $\lambda$.

Proof. Combined with Lemma G.1, recall Theorem 11 and Lemma 2 in [BLMG21] to prove this via the same approach.

Extends to finer false-positive rates. We introduce an approach to achieve finer false-positive rates ($p \neq \frac{1}{2}$) in fuzzy tracking (and also FMD) schemes. As mentioned in [BLMG21], to achieve finer rates like $\frac{\lambda}{2}, \frac{\lambda}{3}$ is easy via switching the base. However, to achieve rates like $\frac{\lambda}{4}$ is still challenging without garbled circuits. We show how to achieve rate like $\frac{\lambda}{2^k}$ where $1 \leq \alpha \leq 2^k - 1$ with a small tweak but $\alpha, k$ needs to be fixed in advance. The sender instead of computing $\text{Enc}(pk_i, 1)$ for each $i \in [n]$, it computes $c_i \leftarrow \text{Enc}(pk_i, msg_i)$ where $msg_i$ is uniformly sampled via $msg_i \leftarrow \{0, 1, \ldots, \alpha\}$. The detector only accepts the ciphertext $c_i$ if and only if $\text{Dec}(sk, c_i) \leq \alpha$. It is easy to see that this satisfies correctness, fuzziness and security simultaneously and is compatible to FMD$_1, FMD_2$ in [BLMG21] and our fuzzy tracking scheme in Figure 9. Essentially, the receiver is able to ‘tune’ the false-positive rate $\rho$ via a finer step: Originally, $\rho$ can only be decreased half by half (i.e., from $\rho$ to $\frac{\rho}{2}$ each time); Now it can be decreased by a factor $\frac{\alpha}{2}$ (i.e., from $\rho$ to $\frac{\rho}{2^k}$). For example, if we choose $k = 2, \alpha = 3$, then we have rates set like $\{\frac{\lambda}{4}, \frac{\lambda}{8}, \ldots, \frac{\lambda}{2^k}\}$.

H Analysis of Scalable Fuzzy Tracking

Security Analysis. For adversaries without holding secret keys, arguments for security are the same as standard encryption. We consider the unlinkability defined in Definition 4.12, then we argue it also satisfies unbiased fuzziness defined in Definition 4.13. Intuitively, unlinkability is to make true-positive and false-positive indistinguishable from the tracking server; And unbiased fuzziness is to make the hint of each potential $mpk'$ uniformly random for the sender.

Proof of Theorem 6.5 We restate the theorem for unlinkability formally here.
Theorem H.1. The fuzzy scalable stealth signature constructed in Figure 10 is unlinkable with key-exposure and fuzzy tracking. Specifically, for any $\lambda, N, \rho$, if there is a PPT adversary $A$ has non-negligible advantage in experiment defined in Figure 6, then there exist other adversaries $B$ running in roughly same time such that:

$$\text{Adv}^{\text{unlink}}_{\lambda, N, \rho}(A) = \text{Adv}^{\text{unlink}}_{\lambda, N, \rho}(B) + \text{Adv}^{\text{mlwe}}_{c_{\rho, q}}(C).$$

**Proof.** First consider the two hybrids as follows:

- **Hybrid$_0$:** This is the standard experiment.
- **Hybrid$_1$:** This only changes $\text{ftki}_0$ to $\text{ftki}_1$ when $\text{hint}_0 \in \text{list} \land \text{hint}_1 \in \text{list}$ whereas $\text{opk}_0, \text{tki}_0$ remain unchanged.

**Claim 2.** Hybrid$_0$ and Hybrid$_1$ are computationally indistinguishable to the adversary if the decisional MLWE holds.

**Proof.** Since we maps each $\text{mpk}$ to $\text{hint}$, we only need to consider the case where $\text{hint}_0 \in \text{list} \land \text{hint}_1 \in \text{list}$ as otherwise $b' \leftarrow \{0, 1\}$ and $A_2$ will not be invoked. Without loss of generality, we assume $\text{ftki}_0 = \text{ftki}_0$ and $\text{hint}_0 = \text{list}[i]$ which implies that, for list generated from $\text{ftki}_0$ and $\forall j \in ||\text{list}||$, there is

$$w_0^j = [s^T c_1 - c_2]_2 \oplus y^j = \left[\frac{q}{2} (s_1( x^j - x^i ) ) + e' - \frac{q}{2} (w_0 + y^j) \right]_2 \oplus y^j = \left[\frac{q}{2} (s_1( x^j - x^i ) ) + e' - \frac{q}{2} (w_0) \right]_2 \oplus (y^j \oplus y^i) = \left[\frac{q}{2} (w_0 + e') + s_1( x^j - x^i ) \right]_2 \oplus y^j \oplus y^i = w_0 \oplus (y^j \oplus y^i) \oplus \left[\frac{q}{2} (s_1( x^j - x^i ) ) \right]_2,$$

where $s_1$ is the first ring element of $s$ and $\text{hint}_0 = \text{decode}_{R_k}(w_0); n$. On the other hand, if $\text{hint}_1$ (i.e., $w_1$) appears in the list with index $k$, i.e., $w_1 = \text{list}[k] = w_0^k$, then the list can also be generated from $\text{ftki}_1$ because $\forall j \in ||\text{list}||$:

$$w_1^j = w_1 \oplus (y^j \oplus y^i) \oplus \left[\frac{q}{2} (s_1( x^k - x^i ) ) \right]_2 = w_0^k \oplus (y^j \oplus y^i) \oplus \left[\frac{q}{2} (s_1( x^k - x^i ) ) \right]_2 = w_0 \oplus (y^j \oplus y^i) \oplus \left[\frac{q}{2} (s_1( x^k - x^i ) ) \right]_2 \oplus (y^j \oplus y^i) \oplus \left[\frac{q}{2} (s_1( x^k - x^i ) ) \right]_2 = w_0 \oplus (y^j \oplus y^i) \oplus \left[\frac{q}{2} (s_1( x^k - x^i ) ) \right]_2 = w_0^j,$$

which means $\text{ftki}_0$ and $\text{ftki}_1$ will generate exactly the same list. Particularly, there is

$$\frac{q}{2} (w_0 + y^j + s_1 x^j) = \frac{q}{2} (w_1 + y^j + s_1 x^j).$$

Now consider the hybrids. We have $\text{ftki}_0 := (c_1, c_2)$ and $\text{ftki}_1 := (c'_1, c'_2)$, specifically,

$$c_1 = A^T r + e_1 + \frac{q}{2} \begin{bmatrix} x^j \\ 0 \end{bmatrix}, c_2 = b^T r + e_2 + \frac{q}{2} (w_0 + y^j)$$

$$c'_1 = A^T r + e_1 + \frac{q}{2} \begin{bmatrix} x^j \\ 0 \end{bmatrix}, c'_2 = b^T r + e_2 + \frac{q}{2} (w_1 + y^j).$$
Thus, there is
\[
(c_1, c_2) \approx_c (u + \frac{q}{2} \begin{bmatrix} x^1 \\ 0 \end{bmatrix}, s^T u + \frac{q}{2} (w_0 + y^1) + \epsilon')
\]
\[
\approx_c (u', s^T u' + \frac{q}{2} (w_0 + y^1 - s^T \begin{bmatrix} x^1 \\ 0 \end{bmatrix}) + \epsilon')
\]
\[
= (u', s^T u' + \frac{q}{2} (w_1 + y^1 + s_1 x^1) + \epsilon')
\]
\[
= (u', s^T u' + \frac{q}{2} (w_1 + y^1 + s_1 x^1) + \epsilon')
\]
\[
\approx_c (c_1', c_2'),
\]
where \( \epsilon' = e_2 + e^T r - s^T e_1 \) is the small noise term and \( u' = u - \frac{q}{2} \begin{bmatrix} x^1 \\ 0 \end{bmatrix} \). Note that \( \frac{q}{2} (s_1 x^1) = \frac{q}{2} (-s_1 x^1) \mod q \). Therefore, we have shown that \( \text{ftki}_0 \) and \( \text{ftki}_1 \) are indistinguishable for the adversary.

Since \( \text{ftki}_b \) and \( \text{ftki}_{1-b} \) are indistinguishable and exchangeable for \( \mathcal{A} \) in UNILNKn-ke. Now we show that \( \mathcal{B} \) can fully simulate the UNILNKn-ke experiment as follows. Upon receiving \( \text{mpk}_b, \text{mpk}_1, \text{opk}_b, \text{tki}_b, \text{osk}_b \) from UNILNKn-ke, \( \mathcal{B} \) sample \( \text{ftk}, \text{fpk} \) then computes corresponding \( \text{ftki}_0 \) and list for \( b' \leftrightarrow \{0, 1\} \), such that \( w_0 \in \text{list} \land w_1 \in \text{list} \) where \( H(\text{mpk}_b) = \text{decode}_{\text{Rq}}(w_0); n \) and \( H(\text{mpk}_1) = \text{decode}_{\text{Rq}}(w_1); n \). Then \( \mathcal{B} \) forwards all of them to \( \mathcal{A} \) of UNILNKn-ke and \( \mathcal{A} \) cannot distinguish between \( \text{ftki}_0 \) or \( \text{ftki}_1 \) due to Claim 2. If \( \mathcal{A} \) has non-negligible advantage \( u(\lambda) \) in UNILNKn-ke, then \( \mathcal{B} \) has the same non-negligible advantage \( u(\lambda) \) in UNILNKn-ke.

We restate the theorem for UNIUBSN, formally here.

**Theorem H.2.** If there is \( n \leq \frac{m}{2} \) where \( m \) is a power of 2, and \( B_q \) is a centered binomial distribution, then the scalable fuzzy tracking constructed in Figure 10 is information theoretically unbiased and satisfies UNIUBSN defined in Definition 4.13.

**Proof.** If \( \mathcal{A} \) is able to output valid \( \text{mpk}_i \) (i.e., valid \( \text{hint}_i \) and \( w^i \)), then for him, there is

\[
w^i = w^i \oplus y^i \oplus y^i \oplus \left[ \frac{q}{2} (s_1 (x^i - x^j)) \right]_2.
\]

Note that the coefficients of \( \left[ \frac{q}{2} s_1 \right]_2 \) are uniformly random over \( \{0, 1\}^m \) because \( s_1 \leftrightarrow B_q \) where \( B_q \) is a centered binomial distribution. Moreover, since polynomial multiplication can be written as circular convolution, \( \left[ \frac{q}{2} (s_1 (x^i - x^j)) \right]_2 \) can be written as \( \text{Xs} \mod 2 \) where \( s \leftrightarrow \text{decode}_{\text{Rq}}(\left[ \frac{q}{2} s_1 \right]_2) \) and \( \text{X} \) is the circulant matrix represented by the polynomial \( x \leftrightarrow \left[ \frac{q}{2} (x^i - x^j) \right]_2 \). Specifically, the first column of \( \text{X} \) is \( \text{decode}_{\text{Rq}}(x) \) and other columns are rotational shift of the previous column. Since \( m \) is a power of 2, it only has divisors from \( 2^0 \) to \( 2^{\log_2 m} \). According to Lemma C.2 and Definition C.1, the biggest divisor of \( X^m - 1 \) is the polynomial \( \Phi_m(x) = x^\frac{m}{2} + 1 \) with degree \( \frac{m}{2} \). Thus the rank of \( \text{X} \) is at least \( m - \frac{m}{2} \) and at least a half of elements in \( \text{X} \mod 2 \) are uniformly randomly distributed. This means \( \text{hint}_j \leftrightarrow \text{decode}_{\text{Rq}}(w^j); n \) is uniformly random as long as \( n \leq \frac{m}{2} \). 

\( \square \)