Multi-server Fuzzy Message Detection

Christopher Goes[1], Yulia Khalniyazova[2], Enrique Larraia[3], and Xuyang Song[4]

[1] cwgoes@heliax.dev
[2] yulia@heliax.dev
[3] enrique@heliax.dev, elarraia@gmail.com
[4] xuyang@heliax.dev

Abstract. Fuzzy Message Detection, or FMD, outsources detection of messages to an untrusted server, Beck et. al. CCS 2021. In this paper, we extend FMD to the multi-key setting: several servers are given different detection keys, all extracted from a single secret key. Multi-key FMD allows to *combine* tests from multiple servers locally by each receiver. This allows to set high false-positive rates on the servers, while attaining low rates on the receiver side. Striking this way a better balance between privacy and efficiency. We further formalize the notion of stealth public keys in the FMD setting. Last, we provide two constructions, one with short public keys.

1 Introduction

Fuzzy message detection [9], outsources detection of messages to an untrusted server. Senders attach messages with flags f generated with a public key pk, and receivers give to the server a detection key dsk (associated to their pk) to test flags. Detection keys identify matching flags with probability 1, and non-matching flags with probability p, where p is a false-positive detection rate p associated with dsk. The so-called *fuzzy* property. Privacy is maintained provided false positive test results are indistinguishable from true positive test results. Beck et. al. refer to this property as *detection ambiguity* (DA). To be meaningful, detection ambiguity should hold against an adversarial server that knows the detection key. Depending on whether the adversary can test, via oracle queries, its own flags at (possibly) different false-positive rates, the notion can be refined into DA-CPA or DA-CCA. Receivers can configure the size of the anonymity set, among which their intended messages are hidden from the server, by appropriately choosing the rate p from a fixed set $\mathcal{P}$. The larger the rate p, the more privacy (anonymity) the receiver enjoys, but the less efficiency it gets from the outsourcing.

The core technique in [9] is to let senders publish γ bit encryptions of 1 under different public keys, and set them as the flag. Testing a flag consists in decrypting $n \leq \gamma$ ciphertexts, and checking whether all the results yield one. Thus, the false-positive rate is 2^{-n} if the server is given n secret keys. Yielding a minimum supported rate $p_{min} := 2^{-\gamma}$. Several candidates following this blueprint were proposed. A generic compiler to achieve the weaker notion

of DA-CPA, from any encryption scheme that is both key-private (IK) [11] and *detection-ambiguous*, an updgrade of IK that holds even with the knowledge of the secret keys. Finer detection rates were also demonstrated leveraging garbled circuits. Both led to blowup in flag sizes, and the fine-grained construction to slow flagging and test runtimes. An efficient construction based in ElGamal, called FMD2, achieved nearly-constant flags, flagging runtime $O_\lambda(|\log(p_{min})|)$ and test runtime $O_\lambda(|\log p|)$, where $p \geq p_{min}$ is the chosen false-positive rate. FMD2 enjoys the strongest notion of detection ambiguity (DA-CCA) in groups where the Gap-CDH problem is hard .

Application to shielded message synchronization Decentralized systems should, or most likely will, support *private* payments or token exchanges. Senders encrypt their data with a key-private asymmetric encryption scheme [11] and the resulting ciphertext (the "shielded message") is stored in a public pool. Many blockchains follow this blueprint [2, 5, 1, 4]. No information about the recipient is present in the shielded message, so receivers need a mechanism to detect their relevant messages. A naive solution is to download all the shielded messages and try to decrypt them to see if the result makes sense. This imposes large communication and computational overheads. Perhaps too much if the receiver is resource-limited, such as a wallet software running in a phone. FMD is a more efficient alternative. A detection server can discard many of the messages not addressed to the recipient, thereby reducing bandwidth and the number of trial decryptions in the wallet. The computational overhead on the wallet side is still $O(T)$, where T is the total number of messages, but the concrete efficiency dictated by the hidden constant depends on the chosen rate p.

FMD pitfalls There are two major drawbacks when considering deployment of an FMD-based synchronization of private messages.

Privacy/efficiency trade off. First and most importantly, the tension between privacy and efficiency questions the effectiveness of the entire approach. In [30] the authors explore the intricacies of such subtle equilibrium from different angles and assumptions. Perhaps the most worrying attack vector is to break receiver unlinkability [8] using FMD. Receiver unlinkability is a standard privacy notion stating that "*it should be hard to tell whether two messages are for the same recipient*". The authors of [30] show that recipient unlinkability for FMD-based solutions is broken if receivers use false-positive rates below some threshold p. They describe a simple attack vector $\mathcal{A}$ against the RU experiment, and lower-bound its success probability with a function $f(p, \#U)$. Here $\#U$ is the number of users of the system. For example, if there are roughly one million users, and $p \approx 2^{-8}$, the success probability of the attacker is greater than $f(p, \#U) \approx 0.5$, as per Fig. 1.a of [30]. The adversary strategy $\mathcal{A}$ is very simple. To infer whether two messages m_a, m_b are addressed to the same identity:

– Let $anon(m_a) = \#\{\text{identities for which } m_a \text{ has a positive FMD flag}\}$. Likewise, define $anon(m_b)$.

– If $anon(m_a) \cap anom(m_b)$ is non-empty, deem messages related. Else, deem messages unrelated.

Thus, a detection server can cluster related messages with probability 0.5 provided users set their rate to $p \approx 2^{-8}$. This is a realistic attack. The worrying bit being that a priori 2^{-8} is *not* that low, or at least it seems a natural choice of many users.

Incompatibility with stealth addresses. The other drawback is that a raw use of FMD in systems that employ Sybil identities as a privacy measure would not be efficient. Think of stealth payments. Stealth addresses are designed to anonymize the real recipient. Attaching the same FMD public key to all those addresses would de-anonimize the recipient. The trivial solution is to have one FMD public key for each stealth address. But then, the recipient must handle as many detection keys as Sybil identities it has. Yielding poor user experience, which may prevent wide adoption of the FMD-based synchronization in the long run.

1.1 Our contributions

Multi-key FMD In this paper we extend the notion of fuzzy message detection [9] to the threshold, multi-key setting. Several servers are given different detection keys, all being extracted from a single secret key. The main difference with the single-server setting is that we are now dealing with false-positive rate *functions* defined over any set of d keys. Although extending the syntax may be deemed obvious, the extraction must account for the corruption threshold. Of course, we need to reformulate the notions of fuzziness and detection ambiguity in the threshold setting. Our contributions are two novel notions, *composability* and, *unlinkable* public keys.

Composability. It captures the idea that combined server-side tests can be more efficient and accurate than individual ones. In the multi-server setting, receivers are in charge of combining server-side tests. Combination of tests should be more efficient than running the tests (in our constructions is several orders of magnitude faster). Furthermore, combined results must yield *lower* false-positive rates. This let us distinguish between the false-positive rate leaked to a fraction of potentially malicious servers, and the rate at which receivers can combine tests. Improving thereby the privacy of the receivers, while at the same time reducing the amount of trial-decryption needed. Importantly, combination is robust: we do not require a minimum number of results to combine – a single test result from one server would work (but it will not help towards lowering rates). For example, consider the stringent setting of pools with ten million messages, with roughly one thousand being addressed to the same receiver. According to reported empirical experiments, to obtain *temporal* detection ambiguity[5] the

[5] Temporal detection ambiguity, coined in [30], says that "*no adversary should be able to tell in a given time interval having access to all incoming flag ciphertexts whether a user received an incoming true-positive match*".

false-positive rate should be above $\approx 2^{-6}$ [30, Fig. 1b]. For example, with three servers and honest majority, the rate leaked to a (single) corrupted server can be set to 2^{-6}, and receivers achieve rates $\approx 2^{-18}$, after combining servers' tests. In other words, we can speed up trial-decryption by several orders of magnitude *without* compromising privacy, even against statistical analysis. In the dishonest majority setting, with two servers corrupted, the situation is less optimistic, but still trial decryption can be sped up by a factor of 4. Refer to Table 2 in Section 5 for more details.

Unlinkable (stealth) public keys. Stealth public keys share the same secret key, but it should not be possible to relate them. Stealth public keys in the context of FMD first appeared in [28]. We borrow their technique in our schemes. Our contribution is a formalization via an *address unlinkability* experiment, and a reduction to the DDH problem. The reduction is tight up to an additive factor $O(\gamma)$.

Constructions We present two schemes. The first one is (almost) a vanilla re-instantiation of the FMD2 scheme from [9] in the multi-key setting. Detection ambiguity rests on the Gap-CDH problem in the random oracle model, as we are re-using the same security analysis from [9].

Short public keys A chief limitation of the efficient scheme FMD2 from [9] is that public keys are linear in γ, i.e. $O_\lambda(|\log p_{min}|)$. Public derivation from shorter public keys reduces bandwidth and storage on wallets. This notion is reminiscent to fuzzy stealth signatures with publicly-derived public keys [25, 20, 29]. In our second scheme, to shorten key sizes, we will use multilinear polynomials, and achieve public keys of size $O_\lambda(|\log p_l|)$, where p_l is the rate leaked to any t servers. Note that in the multi-server setting $p_l \gg p_{min}$. The cost to pay is slower flagging and testing runtimes. Regarding security, we can show DA-CPA relying on a variant of the Poly assumption [16] in the non-pairing setting, that we conjecture to be as hard as the Poly assumption. We leave as an interesting open problem to achieve DA-CCA with short public keys.

Open source implementation We have implemented in Rust and benchmarked the multi-key schemes presented in this work. The codebase [3] is open sourced for the benefit of the community. See Section 5 for details. A theoretical comparison between our multi-key schemes and the single-key schemes from [9] is presented in Table 1. It takes into account flagging/testing complexity, flag size, as well as other features, such as the achieved notion of detection-ambiguity and the assumption it relies on.

1.2 Related work

Oblivious Message Retrieval (OMR) The goal of OMR [22–24, 21] is to retrieve messages from the public pool in a private way. In [22] the authors present

scheme	flag complexity	test complexity	flag size	pk size	multi-key	stealth pk	DA notion (assumption)
FMD1 [9]	2γE	nE	$\gamma\mathbb{G}_q + \gamma$bits	$\gamma\mathbb{G}_q$	No	No	DA-CPA (DDH)
FMD2 [9]	$(\gamma+2)$E	$(n+2)$E	$1\mathbb{G}_q + 1\mathbb{Z}_q + \gamma$bits	$\gamma\mathbb{G}_q$	No	No	DA-CCA (Gap-CDH)
multiFMD2 (Sec. 4.1)	$(\gamma+3)$E	(n_j+2)E	$2\mathbb{G}_q + 1\mathbb{Z}_q + \gamma$bits	$\gamma\mathbb{G}_q$	Yes	Yes	DA-CCA (Gap-CDH)
polyfuzzy (Sec. 4.2)	γME + $(\gamma+3)$E	(n_j+2)E	$2\mathbb{G}_q + 1\mathbb{Z}_q + \gamma$bits	$n\mathbb{G}_q$	Yes	Yes	DA-CPA (KEA)[a]

[a] Conjectured hard. See Section 4.2 for more details.

Table 1. Comparison of the ElGamal-based schemes from [9] and ours. $\mathbb{G}_q$ denotes group elements, $\mathbb{Z}_q$ denotes scalars, E group exponentiations, and ME multi-exponentiations. For a chosen false-positive rate $p_l \in [0,1]$ leaked to the servers, we let $n = |\log p_l|$, and let $2^{-\gamma}$ be the minimum supported rate. Likewise, $n_j = |\log p_j|$ where p_j is the individual rate assigned to the j-th server. That is $n_j \leq n \leq \gamma$.

three schemes from fully homomorphic encryption (FHE). Intuitively, the idea is that during setup, the receiver generates a keypair of an FHE encryption scheme with plaintext space $\mathbb{Z}_2$. Later, a sender attaches to the message ℓ ciphertexts of 1 encrypted under the public key of the receiver (as in FMD). When the receiver wants to sync, sends a randomization of their public key to the server, who, leveraging the (fully) homomorphic property, re-encrypts and multiplies together the ℓ ciphertexts. Only re-encrypted ciphertexts for messages send to the receiver will decrypt to 1, with high probability. Using a bucketing strategy, and leveraging once again FHE, the server groups and embeds messages into compressed ciphertexts. These are sent to the receiver who can decrypt and retrieve their messages. OMR is a stronger primitive than FMD, allowing not just detecting, but also retrieving messages. This, however, comes with inherent computational and communication overheads, introduced by the use of FHE.

Private Signaling (PS) Private signalling [26, 17] is closer to the motivation of this work, as it can work in the multi-server setting. The authors of [26] present two schemes. The first one leverages trusted enclaves. The second scheme works with two non-colluding servers (but, to our knowledge, could work with more servers). The idea is that servers maintain a distributed table with one entry per (registered) receiver. The entries of the row assigned to a receiver are the locations in the public pool of the messages addressed to the receiver. When a sender publishes a message it must also send shares of the intended receiver identity, and of the location of the message in the pool, to the servers, who run a multi-party protocol(MPC) to update the table. Unlike FMD or OMR, private signalling assumes no prior communication between the sender and the receiver, such as transmission of public keys.

Comparison to our work. PS imposes a complex communication model, requiring both sender-to-server and intra-server communication, for input preparation and running of the MPC. We do assume prior communication between senders and

receivers, although in many contexts this is a realistic assumption. Specially if the private detection method is to be used in an anonymous communication system where messages are encrypted with a key-private public-key scheme. Private signalling does not suffer from the privacy-efficiency trade off that FMD does. This work aims to improve on this front, by combining the outputs of the servers, and lower bounding the filtering rate leaked to (a fraction of) them. Regarding robustness, in PS depends on the specifics of the underlying MPC scheme. In our case we get unconditional robustness, as the output of each server can be used independently.

Other approaches The Zcash light client [31] uses a proxy that retrieves the shielded transactions from the ledger and forwards compact versions (with stripped unnecessary metadata) to the receiver's wallet. This saves some communication bandwidth, but does not avoid trial-decryption. In-device smart scanning [15] exploits the receiver's transaction graph to quickly find some of the messages, and provide better user experience. But full trial-decryption is still needed to retrieve all the messages. A proposal in the Aztec network suggests using batch Private Information Retrieval to reduce communication bandwidth [7]. Last, there are also proposals based purely in TEEs [32, 19].

2 Preliminaries

Notation For any $d \in \mathbb{N}^+$, write $[d]$ for the set $\{1, \ldots, d\}$. Its power set is formed by all possible subsets of cardinality less or equal than d, and denoted with $2^{[d]}$. Throughout this work, vector of objects will be denoted in bold $\boldsymbol{v} = (v_1, \ldots, v_d)$, and for any $J \in 2^{[d]}$, we denote the subvector $(v_j)_{j \in J}$ with $\boldsymbol{v}_J$. We let $\mathbb{G}_q$ be a cyclic group of primer order q. Group elements are written with capital roman letters $G \in \mathbb{G}_q$, and scalars in lowercase $x \in \mathbb{Z}_q$.

FMD in the single-key setting Given a set of false-positive rates $\mathcal{P} := \{p \in [0, 1]\}$, an FMD scheme [9] is a tuple of algorithms $(\mathsf{KeyGen}, \mathsf{Extract}, \mathsf{Flag}, \mathsf{Detect})$ defined as follows.

- $\mathsf{KeyGen}(pp) \to (sk, pk)$: On input public parameters pp, it outputs an FMD keypair (sk, pk).
- $\mathsf{Extract}(sk, p) \to dsk$: On input a secret key sk and a false-positive rate p, it outputs a detection key dsk. Or aborts if $p \notin \mathcal{P}$.
- $\mathsf{Flag}(pk) \to f$: On input a public key pk, it outputs a flag f.
- $\mathsf{Test}(dsk, f) \to \{0, 1\}$: On input a detection key dsk and a flag f, it outputs a decision bit signaling whether f was generated using the public key associated to dsk.

FMD2 *scheme.* We take the FMD2 scheme from [9] as starting point in our constructions. It is described in Figure 1 for completeness. The public parameters pp include a fixed generator $G \in \mathbb{G}_q$ and a constant $\gamma \in \mathbb{N}^+$. The secret key are

KeyGen(pp):
1: For $i \in [\gamma]$:
2: $x_i \xleftarrow{\$} \mathbb{Z}_q, H_i \leftarrow G^{x_i}$
3: $sk \leftarrow (x_1, \ldots, x_\gamma)$
4: $pk \leftarrow (H_1 \ldots, H_\gamma)$
5: Output (sk, pk)

Extract($sk, p := 2^{-n}$):
1: $(x_1 \ldots, x_\gamma) \leftarrow sk$
2: If $n > \gamma$:
3: Output $\perp$
4: Else:
5: $dsk \leftarrow (x_1, \ldots, x_n)$
6: Output dsk

Flag(pk):
1: $(H_1 \ldots, H_\gamma) \leftarrow pk$
2: $r, z \xleftarrow{\$} \mathbb{Z}_q$
3: $U \leftarrow G^r$
4: $W \leftarrow G^z$
5: For $i \in [\gamma]$:
6: $k_i \leftarrow \mathsf{H}(U, H_i^r, W)$
7: $c_i \leftarrow k_i \oplus 1$
8: $m \leftarrow \mathsf{G}(U, c_1, \ldots, c_\gamma)$
9: $y \leftarrow (z - m)r^{-1} \mod q$
10: $f \leftarrow (U, y, c_1, \ldots, c_\gamma)$
11: Output f

Detect(dsk, f):
1: $(x_1 \ldots, x_n) \leftarrow dsk$
2: $(U, y, c_1, \ldots, c_\gamma) \leftarrow f$
3: $m \leftarrow \mathsf{G}(U, c_1 \ldots, c_\gamma)$
4: $W \leftarrow G^m U^y$
5: For $i \in [n]$:
6: $k_i \leftarrow \mathsf{H}(U, U^{x_i}, W)$
7: $b_i \leftarrow k_i \oplus c_i$
8: If $b_i = 1\ \forall i \in [n]$:
9: Output 1
10: Else:
11: Output 0

Fig. 1. The FMD2 scheme from [9]. H, G are two cryptographic hash functions.

γ scalars $sk = (x_1, \ldots, x_\gamma)$, and the public key γ points $pk = (H_1, \ldots, H_\gamma)$. The set of valid rates is $\mathcal{P} := \{2^{-n} \mid n \in [\gamma]\}$. The scheme uses hashed ElGamal to generate the flag. The ciphertext bit c_i is generated XORing 1 with the shared key bit $k_i = \mathsf{H}(U, H_i^r, W)$, where H is a cryptographic hash function with co-domain $\mathbb{Z}_2$ (modelled as a random oracle). Re-using the encryption randomness r across the γ ciphertext bits can be done because the "public keys" H_i are independent of each other [12]. The tag $W \in \mathbb{G}_q$ is a Chameleon hash [18], i.e an equivocable (Pedersen) commitment to the hashed message $m = \mathsf{H}(U, U^r, W)$ with commitment key G, U and trapdoor the encryption randomness r. It makes the ciphertext bit vector $\boldsymbol{c} = (c_1, \ldots, c_\gamma)$ non-malleable, as only the encryptor can generate the right opening $y \in \mathbb{Z}_q$. Extracting with false-positive rate 2^{-n} sets the detection key to the first n scalars of the secret key. Testing is done by decrypting the first n ciphertext bits of a flag with the n extracted scalars and checking whether they all yield one.

Hard problems Our constructions rely on three intractability assumptions in $\mathbb{G}_q$. The first two, the decisional Diffie-Hellman (DDH) assumption and the computational gap Diffie-Hellman (Gap-CDH) assumption [27, 6], are well known notions. We also use a knowledge in the exponent assumption. The computational polynomial assumption (Poly) was introduced by Groth and Maller in [16]. It says that for any $G \in \mathbb{G}_q$, given $G^{g_1(\boldsymbol{r})}, \ldots, G^{g_I(\boldsymbol{r})}$, an adversary cannot compute $G^{g(\boldsymbol{r})}$ for a polynomial g that is linearly independent of $g_1, \ldots g_I$, even if it knows $H^{g(x)}$ for $H \in \mathbb{G}_q$. In [16] it is stated in the bilinear setting, with G, H in different source groups, and claimed to hold in the generic group model. Here we state it for the non-bilinear setting, where G, H are linearly independent, but both in $\mathbb{G}_q$. We also give the two basepoints G, H to the adversary. See Figure 2.

3 Multi-key Fuzzy Message Detection

Let $\mathcal{P}_d := \{\rho : 2^{[d]} \to [0, 1]\}$ be a set of rate functions parameterized with d. The functions in $\mathcal{P}_d$ take any subset $J \in 2^{[d]}$ and map it onto $[0, 1]$.

$\underline{\mathsf{Poly}(1^\lambda)}$:

1: $G, H \xleftarrow{\$} \mathbb{G}_q$
2: $\boldsymbol{r} \xleftarrow{\$} (\mathbb{Z}_q^*)^m$
3: $(G^a, g(X_1, \ldots, X_m)) \leftarrow \mathcal{A}(G, H)^{\mathcal{O}^1_{G,\boldsymbol{r}}, \mathcal{O}^2_{H,\boldsymbol{r}}}$
4: return 1 if $a = g(\boldsymbol{r})$ and $g \notin span\{\mathcal{Q}_1\}$

$\underline{\mathcal{O}^1_{G,\boldsymbol{r}}(g_i)}$:

1: on initialization $\mathcal{Q}_1 \leftarrow \emptyset$
2: assert $g_i \in \mathbb{Z}_q[X_1, \ldots, X_m]$
3: assert $deg(g_i) \leq d$
4: $\mathcal{Q}_1 \leftarrow \mathcal{Q}_1 \cup \{g_i\}$
5: return $G^{g_i(\boldsymbol{r})}$

$\underline{\mathcal{O}^2_{H,\boldsymbol{r}}(h_i)}$:

1: assert $h_i \in \mathbb{Z}_q[X_1, \ldots, X_m]$
2: assert $deg(h_i) \leq d$
3: return $H^{h_i(\boldsymbol{r})}$

Fig. 2. The (d, m)-Poly assumption [16] for a (single) group $\mathbb{G}_q$.

For any $d \in \mathbb{N}^+$ a d-multi key fuzzy message detection scheme Π has the following algorithms.

$\mathsf{Setup}(1^\lambda) \to pp$. On input a security parameter λ, it outputs public parameters pp. They include the number of extracted detection keys $d \in \mathbb{N}^+$, and the set of rate functions $\mathcal{P}_d$. If not specified, the public parameters are an implicit input to the algorithms below.

$\mathsf{SKeyGen}(pp) \to sk$: It outputs a secret key sk.

$\mathsf{PKeyGen}(sk, \tau) \to pk$. On input a secret key sk and a public address tag τ, it outputs a public key pk.

$\mathsf{Extract}(sk, t, \rho) \to \boldsymbol{dsk}$: On input a secret key sk and corruption threshold $t \in [d]$, it outputs d detection keys $\boldsymbol{dsk} = (dsk_1, \ldots, dsk_d)$. Or it aborts if $\rho \notin \mathcal{P}_d$.

$\mathsf{Flag}(pk) \to f$: On input a public key pk, it outputs a flag f.

$\mathsf{Test}(\boldsymbol{dsk}_J, f) \to (st_J, b)$: On input a vector of detection keys $\boldsymbol{dsk}_J$ for $J \in 2^{[d]}$, and a flag f, it outputs a test state st_J and a decision bit b.

$\mathsf{Combine}(st_{J_1}, \ldots, st_{J_\ell}) \to \{0, 1\}$ On ℓ input test states st_{J_i} it outputs a decision bit.

Recovering single-key FMD. For $d = 1$ the rate function ρ is constant over the proper subsets of $2^{[d]}$, which is just the singleton $\{1\}$. If we fix $t = d = 1$, restrict the space of addresses to just one, and obviate the combine algorithm (which is only meaningful in the multi-key setting), we recover the syntax of single-key FMD [9] (see Section 2). By applying these restrictions to any multi-key scheme Π we say we obtain the *projected* single-key scheme Π'.

3.1 Known properties reformulated in the multi-key setting

Correctness A multi-key FMD scheme Π is correct if it always detects flags generated with the right public key. For all rate functions $\rho \in \mathcal{P}_d$, all subsets

$J \in 2^{[d]}$, all $t \in [d]$, and all address tags τ it holds:

$$\mathbf{Pr}\left[\Pi.\mathsf{Test}(\boldsymbol{dsk}_J, f) = (st_J, 1) \;\middle|\; \begin{array}{l} pp \leftarrow \Pi.\mathsf{Setup}(1^\lambda) \\ sk \leftarrow \Pi.\mathsf{SKeyGen}(pp) \\ pk \leftarrow \Pi.\mathsf{PKeyGen}(pp, sk, \tau) \\ f \leftarrow \Pi.\mathsf{Flag}(pp, pk) \\ \boldsymbol{dsk} \leftarrow \Pi.\mathsf{Extract}(pp, sk, t, \rho) \end{array}\right] = 1.$$

Fuzziness A multi-key FMD scheme is *fuzzy* if any subset of detection keys test flags with false-positives as prescribed by the rate function $\rho \in \mathcal{P}_d$.

Definition 1 (Fuzziness). *For $d \in \mathbb{N}^+$, a d-multi key FMD scheme Π is* fuzzy *if for all rate functions $\rho \in \mathcal{P}_d$, all subsets $J \in 2^{[d]}$, all $t \in [d]$, and all address tags τ, it holds:*

$$\left|\mathbf{Pr}\left[\Pi.\mathsf{Test}(\boldsymbol{dsk}_J, f) = (st_J, 1) \;\middle|\; \begin{array}{l} pp \leftarrow \Pi.\mathsf{Setup}(1^\lambda) \\ sk_i \leftarrow \Pi.\mathsf{SKeyGen}(pp),\ i \in \{0,1\} \\ pk_0 \leftarrow \Pi.\mathsf{PKeyGen}(pp, sk_0, \tau) \\ f \leftarrow \Pi.\mathsf{Flag}(pp, pk_0) \\ \boldsymbol{dsk} \leftarrow \Pi.\mathsf{Extract}(pp, sk_1, t, \rho) \end{array}\right] - \rho(J)\right| \leq \mathsf{negl}(\lambda),$$

Threshold detection ambiguity As in the single-key case, we require false and true positives to be indistinguishable. In the CCA version of the experiment the adversary can test arbitrary flags, except the challenge, via oracle calls. For fixed $d \in \mathbb{N}^+, t \in [d]$, and $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CCA}\}$ the experiment (t,d)-DA-ATK is against PPT stateful adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

$\underline{(t,d)\text{-DA-ATK}(\Pi, \mathcal{A}, \lambda)}$:

$pp \leftarrow \Pi.\mathsf{Setup}(1^\lambda)$
$(z, \rho, J) \leftarrow \mathcal{A}_1(pp)$ ▷ $J := \{j_1, \ldots, j_t\} \in 2^{[d]}$ and $\rho \in \mathcal{P}_d$
For $i \in \{0,1\}$:
  $sk_i \leftarrow \Pi.\mathsf{SKeyGen}(pp)$
  $\tau_i \leftarrow \{0,1\}^*$
  $pk_i \leftarrow \Pi.\mathsf{PKeyGen}(sk_i, \tau_i)$
$\boldsymbol{dsk}^{(i)} \leftarrow \Pi.\mathsf{Extract}(sk_i, t, \rho),\ \forall i \in \{0,1\}$
$b \xleftarrow{\$} \{0,1\}$
$f^* \leftarrow \Pi.\mathsf{Flag}(pk_b)$
if $\Pi.\mathsf{Test}(dsk_j^{(0)}, f^*) = \Pi.\mathsf{Test}(dsk_j^{(1)}, f^*)\ \forall j \in J$, then:
  $b' \leftarrow \mathcal{A}_2^{\mathcal{O}_{sk_0, sk_1, J}}(pk_0, pk_1, \boldsymbol{dsk}_J^{(0)}, \boldsymbol{dsk}_J^{(1)}, f^*, z)$
else:
  $b' \xleftarrow{\$} \{0,1\}$
output $b = b'$

$\underline{\mathcal{O}_{sk_0,sk_1,J}(id', \rho', f'):}$
1: if $\mathsf{ATK} = \mathsf{CPA}$ return $\perp$
2: if $id' \notin \{0,1\}$, or $\rho' \notin \mathcal{P}_d$, or $f' = f^*$ then:
3: return $\perp$
4: else:
5: $\boldsymbol{dsk}^{(id')} \leftarrow \Pi.\mathsf{Extract}(sk_{id'}, t, \rho')$
6: $(st_j, b_j) \leftarrow \Pi.\mathsf{Test}(dsk_j^{(id')}, f')\ \forall j \in J$
7: output $(st_j, b_j)\ \forall j \in J$

The advantage of $\mathcal{A}$ against a d-multi key FMD scheme Π with up to t corruptions is defined as:

$$\mathbf{Adv}^{(t,d)\text{-DA-ATK}}_{\Pi,\mathcal{A}}(\lambda) := \left| \mathbf{Pr}\left[1 \leftarrow (t,d)\text{-DA-ATK}(\Pi, \mathcal{A}, \lambda)\right] - \frac{1}{2} \right|.$$

The main difference with respect the experiment DA-ATK from [9] is that, in the multi-key setting, detection ambiguity should hold against any coalition of t colluding servers. As in the single-key experiment DA-ATK, our formulation only captures selective adversaries, but not adaptive ones. $\mathcal{A}$ must specify the rate function ρ *and* the corrupted set J of detection keys before it sees the challenge public keys. In the single-key setting, selective security is agnostic to J (i.e. hard-coded to $J := \{1\}$).

Adaptive security. In [9] it is proved that adaptive security is implied by selective security. This is also true in the multi-key setting for any $d \in O(\log \lambda)$ (and if $|\mathcal{P}_d|$ is polynomial in λ). A selective adversary can guess the subset $J \in 2^{[d]}$ (and rate ρ), chosen by an adaptive adversary, with probability $1/p(\lambda)$ for some polynomial p.

3.2 New properties

Composability A scheme Π is *composable* if it is possible to combine the state of ℓ tests and obtain the same decision as if running the test with all the involved detection keys together.

Definition 2 (Composability). *For any $d \in \mathbb{N}^+$, a d-multi key FMD scheme Π is* composable *if for any honestly generated flag f, any $t \in [d]$, any rate function $\rho \in \mathcal{P}_d$, and any ℓ subsets $J_1, \ldots J_\ell \in 2^{[d]}$, it holds:*

$$\mathbf{Pr}\left[\Pi.\mathsf{Combine}(st_{J_1}, \ldots, st_{J_\ell}) = b_K \;\middle|\; \begin{array}{l} pp \leftarrow \Pi.\mathsf{Setup}(1^\lambda) \\ sk \leftarrow \Pi.\mathsf{SKeyGen}(pp, \gamma) \\ \boldsymbol{dsk} \leftarrow \Pi.\mathsf{Extract}(pp, sk, t, \rho) \\ (b_i, st_{J_i}) \leftarrow \Pi.\mathsf{Test}(pp, \boldsymbol{dsk}_{J_i}, f)\ \forall i \in [\ell] \\ K = \cup_{1 \le i \le \ell} J_i \\ (st_K, b_K) \leftarrow \Pi.\mathsf{Test}(pp, \boldsymbol{dsk}_K, f) \end{array} \right] = 1.$$

Combine is meant to be run by receivers. To be useful, composing the outputs of ℓ tests should be faster than running any test individually.

Receiver rate. Tests always return a decision bit. Anyone with a detection key can test flags (e.g. a single detection server). However, we require that testing with more detection keys decreases the false-positive rate. Thus, the family of rate functions $\mathcal{P}_d$ should be monotonically decreasing in the following sense.

Definition 3 (Monotonically decreasing rate functions). *A rate function* $\rho : 2^{[d]} \to \{0,1\}$ *is* monotonically decreasing *if for any* $J_1, J_2 \in 2^{[d]}$ *it holds*

$$\rho(J_1 \cup J_2) < \min\{\rho(J_1), \rho(J_2)\}.$$

It pays off to combine server-side tests with monotonically decreasing rates. The composition always yields strictly lower false-positive rates. The lowest possible rate is $\rho([d])$, which we call the *receiver detection rate*. Namely, the rate after combining tests from all servers' detection keys.

Rate leaked to servers. For a corruption threshold $t \in [d]$, we would like to quantify the testing ability of up to t corrupted servers. The natural way is to take the minimum false-positive rate of any $\ell \leq t$ detection keys.

Definition 4 (Leaked rate function). *Given a false-positive rate function* $\rho : 2^{[d]} \to \{0,1\}$. *The* leaked rate *for* ρ *is the function* $r_\rho : [d] \to \{0,1\}$ *given by*

$$r_\rho(\ell) := \min_{J \in 2^{[d]}, |J| \leq \ell} \{\rho(J)\}.$$

It is not difficult to see that r_ρ is also a monotonically decreasing function (in the standard sense). Therefore, for any set J of at most t keys, we have that

$$\rho(J) \geq r_\rho(|J|) \geq r_\rho(t).$$

We see $r_\rho(t)$ as the *leaked detection rate* a receiver is willing to give away to up to t colluding servers, when extracting with rate function ρ.

Remark 1. We emphasize that composability allows receivers test at low rates. As low as $\rho([d])$. While keeping relatively high false-positive rates $r_\rho(t) > \rho([d])$ on the detection servers. Thereby, improving user privacy. This feature is unique to the multi-key setting. It cannot be achieved with a single server.

Address unlinkability Being able to randomize public keys is useful because receivers can create Sybil identities, or stealth public keys, all controlled by the *same* secret key. The same set of detection keys can be used to test messages sent to any two stealth public keys. The set of stealth public keys associated to a secret key should be unlinkable. For fixed γ, we define the address unlinkability experiment AU against PPT adversaries $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$.

$\underline{\mathsf{AU}(\Pi, \mathcal{A}, \lambda)}$:

$pp \leftarrow \Pi.\mathsf{Setup}(1^\lambda)$
$(z_1, \tau_\alpha, \tau_\beta) \leftarrow \mathcal{A}_1(pp)$
$sk_i \leftarrow \Pi.\mathsf{SKeyGen}(pp)\ \forall i \in \{0,1\}$
$c \xleftarrow{\$} \{0,1\}$
$pk_0 \leftarrow \Pi.\mathsf{PKeyGen}(sk_c, \tau_\alpha)$
$b \xleftarrow{\$} \{0,1\}$
if $b = 0$
    $pk_1 \leftarrow \Pi.\mathsf{PKeyGen}(sk_c, \tau_\beta)$
else:
    $pk_1 \leftarrow \Pi.\mathsf{PKeyGen}(sk_{c+1}, \tau_\beta)$
$b' \leftarrow \mathcal{A}_2(z_1, pk_0, pk_1)$
output $b = b'$

The advantage of $\mathcal{A}$ against a multi-key FMD scheme Π is defined as:

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

Address unlinkability is very similar to recipient unlinkability [8, 30]. Secret keys play the role of receivers, and address tags the role of messages. We emphasize that address unlinkability does not aim to protect against malicious detection servers. It protects against senders that are trying to link public keys. This justifies not giving detection keys to $\mathcal{A}$.

Stealth public keys were first introduced in [28] under the name *diversification*. However, their technique is not compatible with compact public keys. They had to choose between either having short public keys or being able to randomize them.

4 Constructions

4.1 Converting Beck et.al. FMD2 scheme into multi-key

We describe our multi-key scheme multiFMD2 in detail highlighting the differences with the single-key FMD2 from [9]. Pseudocode of the algorithms are given in Figure 3.

Multi-key extraction For false-positive rate 2^{-n}, FMD2 [9] sets the detection key to the first n secret scalars of the secret key $\boldsymbol{x} = (x_1, \ldots, x_\gamma)$. To extract multiple detection keys, our idea is to see a detection key as an *ordered subset* of the γ secret scalars, along with their index positions they occupy in the secret key vector. Specifically, the j-th part contains n_j scalars, where

$$n_j = \begin{cases} \lfloor n/t \rfloor \text{ if } j = 1, \ldots, d-1, \\ n - (t-1)\lfloor n/t \rfloor \text{ if } j = d. \end{cases} \tag{1}$$

For each split $(n_1, \ldots, n_d)$, the j-th detection key is assigned n_j consecutive scalars of $\boldsymbol{x}$. Namely:

$$dsk_j = (I_j, \boldsymbol{x}_j) \text{ where } \begin{cases} I_j = (i_0 := 1 + \sum_{k<j} n_k, i_0 + 1, \ldots, i_0 + n_j), \\ \boldsymbol{x}_j = (x_i)_{i \in I_j}. \end{cases} \tag{2}$$

Each split gives raise to the rate function $\rho(\{j_1, \ldots, j_\ell\}) = 2^{-\sum_{k=1}^{\ell} n_{j_k}}$, which is the false-positive rate assigned to detection keys $dsk_{j_1}, \ldots, dsk_{j_\ell}$.

Stealth public keys A public key in FMD2 is a vector of γ points $H_i = G^{x_i}$. We bound the public key to a public address tag τ, using a different group generator for each τ:

$$pk = (G_\tau, H_1, \ldots, H_\gamma) \text{ such that } H_i = G_\tau^{x_i}.$$

For example, we may set $G_\tau \xleftarrow{\$} \mathbb{G}_q$, or $G = \mathsf{Hash}(\tau)$ for some collision-resistant hash mapping to $\mathbb{G}_q$. As we shall see, stealth public keys defined this way are unlinkable.

Flagging with stealth keys The generator G_τ identifies the public key pk. To avoid leaking information about pk when testing flags, each flag operation uses, for the chameleon hash, an ephemeral generator $G_{\mathsf{ch}} = G_\tau^s$, with fresh random scalar s. The ephemeral generator G_{ch} is included in the flag f. The knowledge of s is needed to generate the opening y of the commitment w to the collision $m = \mathsf{G}(U, c_1, \ldots, c_\gamma)$, which now is set to $y := (z - m)sr^{-1} \mod q$. [6]

Testing flags If testing with ℓ detection keys $\boldsymbol{dsk}_J$, we modify the FMD2 algorithm in two aspects. First, recall that detection keys defined in (2) are index-aware. For all $j \in J$, the indices I_j embedded in the detection key dsk_j are used to select the correct ciphertext bit that will be decrypted with the i-th scalar of $\boldsymbol{x}_j$. Second, the generator g_{ch} for the chameleon hash is retrieved from the flag f.

Combining tests The split $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d\}$ of the secret key sk defined in Eq. (2) is disjoint, so different detection keys $dsk_{j_1}, \ldots, dsk_{j_\ell}$ decrypt *different* subset of the ciphertext bits of a given flag. Therefore, ℓ tests can be combined by simply taking the AND of the individual decision bits $b_{j_1}, \ldots, b_{j_\ell}$. This also works when combining tests done with non-singleton subsets of detection keys. Note that the rate function ρ is monotonically decreasing as per Defn. 3, so the combined test yields strictly lower false-positive rates than any of the individual tests.

Detection ambiguity In the random oracle model, the scheme multiFMD2 described in Fig. 3 has detection ambiguity when instantiated over a group where the Gap-CDH problem is hard.

[6] Stealth public keys were first described in Penumbra [28], under the name *diversification*. However, no formal security analysis was provided.

Theorem 1 (Restatement of Thm. 13 from [9]). *Let* multiFMD2 *the multi-key FMD scheme of Fig. 3. For any PPT adversary $\mathcal{A}$ against the (t, d)-*DA-CCA *experiment, making q_{H}, q_{G} queries to the random oracles H and G, respectively, there exists a PPT algorithms $\mathcal{B}_1, \mathcal{B}_2$ such that:*

$$\mathbf{Adv}^{(t,d)\text{-DA-CCA}}_{\mathsf{multiFMD2},\mathcal{A}}(\lambda) \leq \frac{q_{\mathsf{G}}}{q} + \mathbf{Adv}^{\mathsf{CDH}}_{\mathcal{B}_1}(\lambda) + q_{\mathsf{H}} \cdot \mathbf{Adv}^{\mathsf{Gap\text{-}CDH}}_{\mathcal{B}_2}(\lambda)$$

See Appendix A for the proof.

$\mathsf{SKeyGen}(pp)$:
1: $x_i \xleftarrow{\$} \mathbb{Z}_q$ for $i \in [\gamma]$
2: $sk \leftarrow (x_1, \ldots, x_\gamma)$
3: Output sk

$\mathsf{PKeyGen}(sk, \tau)$:
1: $(x_1, \ldots, x_\gamma) \leftarrow sk$
2: $G_\tau \xleftarrow{\$} \mathbb{G}_q$
3: For $j \in [\gamma]$:
4: $\quad H_i \leftarrow G_\tau^{x_i}$
5: $pk \leftarrow (G_\tau, H_1, \ldots, H_\gamma)$
6: Output pk

$\mathsf{Extract}(sk, d, t, \rho := n)$:
1: $(x_1, \ldots, x_\gamma) \leftarrow sk$
2: If $(d-t)\lfloor n/t \rfloor + n > \gamma$ or $d > \gamma$:
3: $\quad$ abort $\quad \triangleright$ If $\rho \notin \mathcal{P}_d$
4: For $j \in [d]$:
5: $\quad$ If $j < d$ then:
6: $\quad\quad n_j \leftarrow \lfloor n/t \rfloor$
7: $\quad$ Else:
8: $\quad\quad n_j \leftarrow n - (t-1)\lfloor n/t \rfloor$
9: $\quad$ If $j = 1$, then:
10: $\quad\quad \mathsf{idx}_j \leftarrow 1$
11: $\quad$ Else:
12: $\quad\quad \mathsf{idx}_j \leftarrow \sum_{i=1}^{j-1} n_i$
13: $\quad\quad I_j \leftarrow \{\mathsf{idx}_j + 1, \ldots, \mathsf{idx}_j + n_j\}$
14: $\quad dsk_j \leftarrow (I_j, \boldsymbol{x}_{I_j})$
15: $\boldsymbol{dsk} \leftarrow (dsk_1, \ldots, dsk_d)$
16: Output $\boldsymbol{dsk}$

$\mathsf{Flag}(pk)$:
1: $(G_\tau, H_1 \ldots, H_\gamma) \leftarrow pk$
2: $r, s, z \xleftarrow{\$} \mathbb{Z}_q$
3: $U \leftarrow G_\tau^r$
4: $G_{\mathsf{ch}} \leftarrow G_\tau^s$
5: $W \leftarrow G_{\mathsf{ch}}^z$
6: For $i \in [\gamma]$:
7: $\quad k_i \leftarrow \mathsf{H}(U, H_i^r, W)$
8: $\quad c_i \leftarrow k_i \oplus 1$
9: $m \leftarrow \mathsf{G}(U, c_1, \ldots, c_\gamma)$
10: $y \leftarrow (z - m)sr^{-1} \mod q$
11: $f \leftarrow (G_{\mathsf{ch}}, U, y, c_1, \ldots, c_\gamma)$
12: Output f

$\mathsf{Test}(\boldsymbol{dsk}', f)$:
1: $(dsk_1, \ldots, dsk_\ell) \leftarrow \boldsymbol{dsk}'$
2: $(I_j, \boldsymbol{x}_j) \leftarrow dsk_j$ for $j \in [\ell]$
3: If $i \in \cap_j I_j$:
4: $\quad$ Assert $x_{j,i} = x_{j',i}\ \forall j \neq j'$
5: $\quad$ Else abort
6: $(I, \boldsymbol{x}) \leftarrow (\cup_j I_j, \cup_j \boldsymbol{x}_j)$
7: $(G_{\mathsf{ch}}, U, y, c_1, \ldots, c_\gamma) \leftarrow f$
8: $m \leftarrow \mathsf{G}(U, c_1 \ldots, c_\gamma)$
9: $W \leftarrow G_{\mathsf{ch}}^m \cdot U^y$
10: For $l \in \{1, \ldots, |I|\}$:
11: $\quad k_l \leftarrow \mathsf{H}(U, U^{x_l}, W)$
12: $\quad b_l \leftarrow k_l \oplus c_{i_l} \quad \triangleright$ l-th $i_l \in I$
13: If $b_l = 1\ \forall l \leq |I|$:
14: $\quad st \leftarrow (1, f)$
15: $\quad$ Output $(st, 1)$
16: Else:
17: $\quad st \leftarrow (0, f)$
18: $\quad$ Output $(st, 0)$

$\mathsf{Combine}(st_1, \ldots, st_\ell)$:
1: $(b_j, f_j) \leftarrow st_j$
2: If $f_j \neq f_{j'}$ for any $j \neq j' \in [\ell]$:
3: $\quad$ abort
4: Output $\wedge_{j=1}^{\ell} b_j$

Fig. 3. The multi-key FMD scheme multiFMD2. $d, \gamma \in \mathbb{N}^+$ are part of the public parameters. In Extract, the rate function ρ is identified with integer n, which defines the split of the secret key. H, G are two random oracles, with co-domain $\mathbb{Z}_2$ and $\mathbb{Z}_q$, respectively.

Fuzziness Recall that the rate function assigned to detection keys $dsk_{j_1}, \ldots, dsk_{j_\ell}$ is $\rho(\{j_1, \ldots, j_\ell\}) = 2^{-\sum_{k=1}^{\ell} n_{j_\ell}}$, where $(n_1, \ldots, n_d)$ is the split of the secret key indices from Eq. (1).

Theorem 2. *Let λ be the security parameter, and $\gamma \in poly(\lambda)$. The scheme* multiFMD2 *of Fig. 3 is fuzzy as per Definition 1. Thus, it holds:*

$$\left| \mathbf{Pr} \left[\mathsf{Test}(\boldsymbol{dsk}_J, f) = (st_J, 1) \;\middle|\; \begin{array}{l} pp \leftarrow \mathsf{Setup}(1^\lambda) \\ sk_i \leftarrow \mathsf{SKeyGen}(pp),\ i \in \{0,1\} \\ pk_0 \leftarrow \mathsf{PKeyGen}(sk_0, \tau) \\ f \leftarrow \mathsf{Flag}(pk_0) \\ \boldsymbol{dsk} \leftarrow \mathsf{Extract}(sk_1, t, \rho) \end{array} \right] - 2^{-\sum_{k=1}^{\ell} n_{j_k}} \right| \leq \mathsf{negl}(\lambda).$$

The proof is in Appendix B.

Address unlinkability Assuming the hardness of DDH in $\mathbb{G}_q$, public keys are unlinkable.

Theorem 3. *Let* multiFMD2 *be the scheme of Fig. 3. Let γ be the number of scalars in the secret key. For any PPT adversary $\mathcal{A}$ against the* AU *experiment, there exists a PPT algorithm $\mathcal{B}$ such that:*

$$\mathbf{Adv}^{\mathsf{AU}}_{\mathsf{multiFMD2},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{DDH}}_{\mathcal{B}}(\lambda) + \frac{\gamma + 1}{q}.$$

Proof. Denote the challenge secret keys as $sk_c = (x_1, \ldots, x_\gamma)$, and $sk_{c+1} = (y_1, \ldots, y_\gamma)$, where c is the bit randomly sampled by the challenger. We prove the theorem with three hybrids. In the last hybrid $\mathcal{H}_3$, the public key pk_1 is generated independently of the challenge bit r, and hence $\mathbf{Adv}^{\mathcal{H}_3}_{\mathcal{A}}(\lambda) = 0$.

Hybrid $\mathcal{H}_0$ The original AU experiment for multiFMD2.

Hybrid $\mathcal{H}_1$. The challenge secret key $sk_c = (x_1, \ldots, x_\gamma)$ is generated as follows:

1: $a \xleftarrow{\$} \mathbb{Z}_q$
2: For $j \in [\gamma]$:
3: $\quad x'_j \xleftarrow{\$} \mathbb{Z}_q$
4: $\quad x_j \leftarrow a x'_j$

If a can be inverted, the mapping $f_a(s_1, \ldots, s_\gamma) = (as_1, \ldots, as_\gamma)$ is a bijection over $\mathbb{Z}_q^\gamma$. Moreover, the blindings x'_j are chosen randomly. Bijections preserve the uniform distribution, so the secret key $sk_c = f_a(\boldsymbol{x}')$ is randomly distributed, as in the previous hybrid. Since q is prime, a random scalar a can be inverted if $a \neq 0$, which happens with probability $1/q$. We have that:

$$\left| \mathbf{Adv}^{\mathcal{H}_1}_{\mathcal{A}}(\lambda) - \mathbf{Adv}^{\mathcal{H}_0}_{\mathcal{A}}(\lambda) \right| \leq \frac{1}{q}.$$

Hybrid $\mathcal{H}_2$. Let $\boldsymbol{x}' = (x'_1, \ldots, x'_\gamma)$ be the blinding used to generate the secret key sk_c (from hybrid $\mathcal{H}_1$ onwards). In this hybrid, if the challenge bit is $r = 0$, the public key pk_1 is generated as follows:

1: $R \xleftarrow{\$} \mathbb{G}_q$
2: $pk_1 \leftarrow (G_1, R^{x'_1}, \ldots, R^{x'_\gamma})$

We claim that

$$\left|\mathbf{Adv}^{\mathcal{H}_2}_{\mathcal{A}}(\lambda) - \mathbf{Adv}^{\mathcal{H}_1}_{\mathcal{A}}(\lambda)\right| \leq \mathbf{Adv}^{\mathsf{DDH}}_{\mathcal{B}}(\lambda).$$

Suppose it is possible to distinguish both hybrids. We construct an algorithm $\mathcal{B}$ against the DDH experiment. Let instance $(G, A = G^a, B = G^b, C)$. $\mathcal{B}$ emulates the challenger of either hybrid as follows:

- Sample $\boldsymbol{x}', \boldsymbol{y} \in \mathbb{Z}_q^\gamma$.
- Set $pk_0 = (G, A^{x'_1}, \ldots, A^{x'_\gamma})$.
- Sample a challenge bit r.
- If $r = 0$, set $pk_1 = (B, C^{x'_1}, \ldots, C^{x'_\gamma})$.
- Else if $r = 1$, set $pk_1 = (B, B^{y_1}, \ldots, B^{y_\gamma})$.

If $C = G^{ab}$, then $\mathcal{B}$ emulates hybrid $\mathcal{H}_1$. Else $C \xleftarrow{\$} \mathbb{G}_q$, and we are in hybrid $\mathcal{H}_2$. This reduces distinguishing the hybrids to DDH.

Hybrid $\mathcal{H}_3$. Let $\boldsymbol{x}'$ be the blinding used to generate the secret key sk_c (from hybrid $\mathcal{H}_1$ onwards). In this hybrid, if the challenge bit is $r = 0$, the public key pk_1 is generated as follows:

1: $R_j \leftarrow G_1^{y_j / x'_j}$ for $j \in [\gamma]$
2: $pk_1 \leftarrow (G_1, R_1^{x'_1}, \ldots, R_\gamma^{x'_\gamma})$

R_j can be computed and is uniform provided x'_j can be inverted; that is if $x'_j \neq 0$. Therefore:

$$\left|\mathbf{Adv}^{\mathcal{H}_3}_{\mathcal{A}}(\lambda) - \mathbf{Adv}^{\mathcal{H}_2}_{\mathcal{A}}(\lambda)\right| \leq \frac{\gamma}{q}.$$

Last, in $\mathcal{H}_3$, if the challenge bit is $r = 0$, then $pk_1 = (G_1, G_1^{y_1}, \ldots, G_1^{y_\gamma})$, exactly as when $r = 1$. So $\mathbf{Adv}^{\mathcal{H}_3}_{\mathcal{A}}(\lambda) = 0$ □

4.2 Reducing the size of public keys

To obtain short public keys we use m scalars for $m < \gamma$ for the secret key $sk := (r_1, \ldots, r_m) \in (\mathbb{Z}_q^*)^m$. The corresponding public key is their encoding in "the exponent", $pk = (G_\tau, H_1, \ldots, H_m) := (G_\tau, G_\tau^{r_1}, \ldots, G_\tau^{r_m}) \in \mathbb{G}_q^{m+1}$.

Extraction We first "expand" sk into γ secret scalars $\boldsymbol{x}$, setting $x_i = h_i(sk)$. Here, h_i are public multilinear polynomials $h_i \in \mathbb{Z}_q[X_1, \ldots, X_m]$. The set $\mathcal{H} = \{h_i\}_{i \in [\gamma]}$ is chosen so that any subset of at most m polynomials are linearly independent. $\mathcal{H}$ can be generated using hyper-invertible matrices [10]. This recovers the secret key $\boldsymbol{x}$ of the multi-key scheme multiFMD2, over which we can run Extract.

SKeyGen(pp):
1: $\boldsymbol{r} \xleftarrow{\$} (\mathbb{Z}_q^*)^m$
2: $sk \leftarrow \boldsymbol{r}$
3: Output sk

PKeyGen(sk, τ):
1: $\boldsymbol{r} \leftarrow sk$
2: $G_\tau \xleftarrow{\$} \mathbb{G}_q$
3: For $h_i \in \mathcal{H}$:
4: $\quad H_i \leftarrow G_\tau^{h_i(\boldsymbol{r})}$
5: $pk \leftarrow (G_\tau, H_1, \ldots, H_m)$
6: Output pk

Extract($sk, t, \rho := n$):
1: $\boldsymbol{r} \leftarrow sk$
2: If $m \leq n$:
3: $\quad$ abort
4: $\gamma \leftarrow (d-1)\lfloor n/t \rfloor + n - (t-1)\lfloor n/t \rfloor$
5: For $h_i \in \mathcal{H}$: ▷ $|\mathcal{H}| = \gamma$
6: $\quad x_i \leftarrow h_i(\boldsymbol{r})$
7: $\boldsymbol{x} \leftarrow (x_1, \ldots, x_\gamma)$
8: $\tilde{sk} \leftarrow (\boldsymbol{x}, t)$
9: $\boldsymbol{dsk} \leftarrow$ multiFMD2.Extract($\tilde{sk}, t, n$)
10: Output $\boldsymbol{dsk}$

Flag(pk):
1: $(G_\tau, H_1, \ldots, H_m) \leftarrow pk$
2: For $h_i \in \mathcal{H}$: ▷ $|\mathcal{H}| = \gamma$
3: $\quad \boldsymbol{c}_i \leftarrow h_i$ ▷ coeffs of h_i
4: $\quad \tilde{H}_i \leftarrow G_\tau^{c_{i,0}} \prod_{j=1}^m H_j^{c_{i,j}}$
5: $\tilde{pk} \leftarrow (G_\tau, \tilde{H}_1 \ldots, \tilde{H}_\gamma)$
6: $f \leftarrow$ multiFMD2.Flag($\tilde{pk}$)
7: Output f

Test($\boldsymbol{dsk}', f$):
1: Output:
multiFMD2.Test($\boldsymbol{dsk}', f$)

Combine($\boldsymbol{st}, f$):
1: Output:
multiFMD2.Combine($\boldsymbol{st}, f$)

Fig. 4. The compact multi-key FMD scheme polyfuzzy based on multiFMD2. The degree $m \in \mathbb{N}^+$ and the γ multilinear polynomials $h_i \in \mathcal{H}$ are part of the public parameters. See Figure 3 for the description of multiFMD2.

Flagging Expanding pk onto $(G_\tau, G_\tau^{h_1(\boldsymbol{r})}, \ldots, G_\tau^{h_\gamma(\boldsymbol{r})})$ can be done *publicly*, e.g. by the sender, using the encoded evaluation point in pk. This is possible because each h_i is multilinear and public. Let $\boldsymbol{c}_i \in \mathbb{Z}_q^{m+1}$ be the coefficients of h_i. During a Flag operation, the sender first computes γ multi-exponentiations $\tilde{H}_i = G_\tau^{c_{i,0}} \prod_{j=1}^m H_j^{c_{i,j}}$. Clearly, it holds $\tilde{H}_i = G_\tau^{h_i(\boldsymbol{r})}$. The set $\mathcal{H}$ is part of the public parameters.

Tradeoffs We trade communication bandwidth with computational effort on the sender side. The public key consists now of $m+1$ points. The false-positive rate leaked to t servers must be 2^{-n} with $n \in [t, m)$. Note n is the number of evaluations $x_i = h_i(\boldsymbol{r})$ leaked to t corrupted servers. Otherwise, if $n \geq m$, since any m polynomials in $\mathcal{H}$ are linearly independent, the secret evaluation point $\boldsymbol{r}$ can be recovered by t colluding servers with Gaussian elimination using m evaluations $h_i(\boldsymbol{r})$. This is an improvement over public keys with γ points of the non-compact versions multifmdX.

The compact multi-key scheme polyfuzzy is described in Figure 4.

Fuzziness Fuzziness is not altered by the correlation between the scalars, or group elements, in the expanded secret key $(h_1(\boldsymbol{r}), \ldots, h_\gamma(\boldsymbol{r}))$, or in the expanded public key $(G_\tau, G_\tau^{h1(\boldsymbol{r})}, \ldots, G_\tau^{h_\gamma(\boldsymbol{r})})$. Thus, fuzziness holds for polyfuzzy given we have proved it for multiFMD2.

Address unlinkability Compact keys consist of m random scalars $\boldsymbol{r}$ and their encoding in the exponent. We are in the same situation as for the non-compact scheme.

Theorem 4. *Let* polyfuzzy *be the scheme of Figure 4. Let m be the size of the evaluation point $\boldsymbol{r}$ forming the compact secret key. For any PPT adversary $\mathcal{A}$ against the* AU *experiment, there exists a PPT algorithm $\mathcal{B}$ such that:*

$$\mathbf{Adv}^{\mathsf{AU}}_{\mathsf{polyfuzzy},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{DDH}}_{\mathcal{B}}(\lambda) + \frac{m+1}{q}.$$

Proof. The proof is an exact copy of the proof of Thm. 3 replacing γ by m. □

Detection ambiguity We rely on a variant of the Poly assumption [16]. In our variant, the adversary is given oracle evaluations of up to $r < m$ multilinear polynomials $\mathcal{E} \in \mathbb{Z}_q[X_1, \ldots, X_m]$ of its choice. It must ouput $G^{g(\boldsymbol{r})}$ for a polynomial g not in the span of $\mathcal{Q}_1 \cup \mathcal{E}$, where $\mathcal{Q}_1$ are the polynomials for which it gets evaluations in the exponent (by querying its first oracle). We call this assumption the computational (m, r)-rank linear polynomial assumption (RLPoly). See Figure 5. Recall the (d, m)-Poly assumption [16] from the preliminaries (cf. Fig 2).

Conjecture 1. (m, r)-RLPoly is as hard as $(1, m - r)$-Poly, for any $m \geq 2$ and $r \in [m - 1]$.

Intuition. The extra help in RLPoly, with respect the Poly experiment, are the $r \leq m$ evaluations $y_i = e_i(\boldsymbol{r})$, which allows to establish a linear system of rank r of equations on m unknowns. There are $m - r$ free variables, thus q^{m-r} solutions $\boldsymbol{r}$ to this system, all equally likely. In other words, it seems RLPoly is as hard as Poly with the number of variables reduced to $m - r$.

The $(1, 1)$-Poly assumption is closely related to the KEA1 assumption [14, 13]. So the conjecture above could also be understood as saying that $(m, m - 1)$-RLPoly is as hard as KEA1.

$\underline{(m, r)\text{-}\mathsf{RLPoly}(1^\lambda)}$:

1: $G, H \xleftarrow{\$} \mathbb{G}_q$
2: $\boldsymbol{r} \xleftarrow{\$} (\mathbb{Z}_q^*)^m$
3: $(G^a, g(X_1, \ldots, X_m)) \leftarrow \mathcal{A}(G, H)^{\mathcal{O}^1_{G,\boldsymbol{r}}, \mathcal{O}^2_{H,\boldsymbol{r}}, \mathcal{O}^3_{\boldsymbol{r}}}$
4: return 1 if $a = g(\boldsymbol{r})$ and $g \notin \mathit{span}\{\mathcal{Q}_1 \cup \mathcal{E}\}$

$\underline{\mathcal{O}^1_{G,\boldsymbol{r}}(g_i)}$:

1: on initialization $\mathcal{Q}_1 \leftarrow \emptyset$
2: assert $g_i \in \mathbb{Z}_q[X_1, \ldots, X_m]$
3: assert $\mathit{deg}(g_i) \leq 1$
4: $\mathcal{Q}_1 \leftarrow \mathcal{Q}_1 \cup \{g_i\}$
5: return $G^{g_i(\boldsymbol{r})}$

$\underline{\mathcal{O}^2_{H,\boldsymbol{r}}(h_i)}$:

1: assert $h_i \in \mathbb{Z}_q[X_1, \ldots, X_m]$
2: assert $\mathit{deg}(h_i) \leq 1$
3: return $H^{h_i(\boldsymbol{r})}$

$\underline{\mathcal{O}^3_{\boldsymbol{r}}(e_i)}$:

1: on initialization $\mathcal{E} \leftarrow \emptyset$
2: assert $|\mathcal{E}| < r$
3: assert $e_i \in \mathbb{Z}_q[X_1, \ldots, X_m]$
4: assert $\mathit{deg}(e_i) \leq 1$
5: $\mathcal{E} \leftarrow \mathcal{E} \cup \{e_i\}$
6: return $e_i(\boldsymbol{r})$

Fig. 5. The (m, r)-RLPoly assumption in $\mathbb{G}_q$.

We now show that polyfuzzy is (t,d)-DA-CPA based on the hardness of RLPoly.

Theorem 5. *Let* polyfuzzy *be the scheme of Figure 4 with parameters m, d. For any $t < m$, and any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against the (t,d)-DA-CPA experiment, making q_{H}, q_{G} queries to the random oracles H, G, respectively, there exists a PPT algorithm $\mathcal{B}$ against m-RLPoly such that:*

$$\mathbf{Adv}^{(t,d)\text{-DA-CPA}}_{\mathsf{polyfuzzy},\mathcal{A}}(\lambda) \leq \frac{q_{\mathsf{G}}}{q} + q_{\mathsf{H}}\mathbf{Adv}^{(m,m-1)\text{-RLPoly}}_{\mathcal{B}}(\lambda).$$

Proof. Let $\rho := n \in [t, m)$ be the rate function that $\mathcal{A}_1$ outputs. We can hop to a hybrid where the only dependence of the challenge flag f^* on challenge bit b is in the masks $D_j = U^{h_{m+j}(\boldsymbol{r})}$ for $j \in [\gamma - n]$, used to generate the bit key k_{m+j}. These hops would be identical as in the proof for FMD2 [9, Thm. 13]. The term $\frac{q_{\mathsf{G}}}{q}$ of the bound accounts for these transitions.

At this point, if $\mathcal{A}$ does not query D_j to the random oracle H for $j \in [\gamma - n]$, then the bit key $k_{n+j} = \mathsf{H}(U, D_j, W)$ looks random in $\mathcal{A}$'s view. We hop to a hybrid where the challenger uses a random point R_j as the mask D_j. In this hybrid, if the event "$\mathcal{A}$ queries any D_j" happens, we can construct an adversary $\mathcal{B}$ against the m-RLPoly experiment. Recall $\mathcal{H}$ are the public multilinear polynomials used to generate the scalars of the detection keys $(dsk_j)_{j\in[d]}$. The description of $\mathcal{B}$ is as follows:

- $\mathcal{B}$ receives as input two base points G, H. From now on, we rename them to $U := G, G_\tau := H$.
- $\mathcal{B}$ runs $\mathcal{A}_1$, which outputs the corrupted set $J \in 2^{[d]}$. $\mathcal{B}$ chooses $\mathcal{E} := \{e_1, \ldots, e_n\}$, where $e_1, \ldots, e_n$ are the subset of the polynomials in $\mathcal{H}$ used to generate the scalars of $(dsk^{(i)}_{j_k})_{j_k \in J}$ for $i = 1, 2$.
- $\mathcal{B}$ queries $e_i \in \mathcal{E}$ to its third oracle $\mathcal{O}^3_{\boldsymbol{r}}$ to get evaluations $y_1 = e_1(\boldsymbol{r}), \ldots, y_n = e_n(\boldsymbol{r})$, and uses $y_1, \ldots y_n$ as the scalars for $(dsk^{(b)}_{j_k})_{j_k \in J}$, where b is the challenge bit.
- To form the challenge public key $pk^{(b)} = (G_\tau, G_\tau^{r_1}, \ldots, G_\tau^{r_m})$, $\mathcal{B}$ queries polynomials X_i for $i = 1, \ldots m$ to its second oracle $\mathcal{O}^2_{G_\tau, \boldsymbol{r}}$
- To form the challenge flag f^*, $\mathcal{B}$ chooses $\gamma - n$ random points R_j and sets $f^* = (G_{\mathsf{ch}}, U, y, c_1, \ldots, c_\gamma)$, where $c_{n+j} = \mathsf{H}(G, R_j, W) \oplus 1$, and $c_i = \mathsf{H}(U, U^{y_i}, W) \oplus 1$, for $i \in [n]$. Recall y_i is known to $\mathcal{B}$.
- $\mathcal{B}$ runs $\mathcal{A}_2(pk_0, pk_1, \boldsymbol{dsk}^{(0)}_J, \boldsymbol{dsk}^{(1)}_J, f^*)$
- At some point, $\mathcal{A}_2$ will query $D_j = U^{h_{n+j}(\boldsymbol{r})}$ to the random oracle H for some $j \in [\gamma - n]$. So $\mathcal{B}$ picks one query Q at random, and outputs $(Q, h_{n+j}(X_1, \ldots, X_m))$. The probability that $Q = D_j$ is $1/q_{\mathsf{H}}$.

Note $\mathcal{B}$ never queries its first oracle $\mathcal{O}^1_{U,\boldsymbol{r}}$. Also, $h_{n+j} \notin \mathcal{E}$ because, by construction, any set of $n + 1 \leq m$ polynomials from $\mathcal{H}$ are linearly-independent. So $(U^{h_{n+j}(\boldsymbol{r})}, h_{n+j}(X_1, \ldots, X_m))$ is a winning output in the RLPoly experiment.

□

5 Choice of parameters, comparison, and benchmarks

The parameters are the number of servers d, the corruption threshold t, and the false-positive rate leaked to the servers $p_l = 2^{-n}$.

As commented in the introduction, safe choices of p_l should be larger than 2^{-6}, roughly. Detection ambiguity for these choices seems to hold even if (malicious) servers conduct statistical analysis, according to empirical experiments [30, Fig 1b]. We therefore set $n \leq 6$.

We consider the honest majority setting, with $t = \lfloor d/2 \rfloor$, and the dishonest majority setting with $t = d - 1$. As each server is given at least one scalar in its detection key, we can have $d \leq 13$ servers in the HM setting, and $d \leq 7$ with DM. In our benchmarks we cap $d \leq 6$. Table 2 presents the receiver rate for different configurations. Aiming for servers leaked rate 2^{-6}, the most efficient configuration in terms of receiver rate, is three servers in the honest majority setting. The receiver rate is several orders of magnitude lower than the leaked rate.

d / p_l	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
2	2^{-2}	2^{-4}	2^{-6}	2^{-8}	2^{-10}	2^{-12}
3	2^{-3}	2^{-6}	2^{-9}	2^{-12}	2^{-15}	2^{-18}
4	2^{-1}	2^{-4}	2^{-5}	2^{-8}	2^{-9}	2^{-12}
5	2^{-1}	2^{-5}	2^{-6}	2^{-10}	2^{-11}	2^{-15}
6	2^{-1}	2^{-2}	2^{-6}	2^{-7}	2^{-8}	2^{-12}

d / p_l	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
3	2^{-1}	2^{-3}	2^{-4}	2^{-6}	2^{-7}	2^{-9}
4	2^{-1}	2^{-2}	2^{-4}	2^{-5}	2^{-6}	2^{-8}
5	2^{-1}	2^{-2}	2^{-3}	2^{-5}	2^{-6}	2^{-7}
6	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-6}	2^{-7}

Table 2. Receiver rate for the different safe choices of leaked rate $p_l = 2^{-n}$ and number of servers d. **Left:** Honest majority setting with $t = \lfloor d/2 \rfloor$. **Right:** Dishonest majority setting with $t = d - 1$.

We have implemented our multi-key schemes in Rust over the Ristretto curve using the `dalek` crate. In this curve, a scalar and a compressed point take 32 bytes. In Table 3 and Figure 6 we present benchmarks run on a laptop with 12 cores intel i5 13th generation and 16GB RAM. The code is open source and publicly available [3].

scheme	Flag runtime	flag size	public key size
multiFMD2	741.63 μs	99 bytes	608 bytes
polyfuzzy	787.93 μs		256 bytes

Table 3. Flagging runtime, and sizes of flags and public keys. We set $\gamma = 18$, $d = 6$, and for the compact scheme polyfuzzy, $m = 7$. These choices account for all safe configurations (cf. Table 2). The difference in flagging is due to key expansion in polyfuzzy.

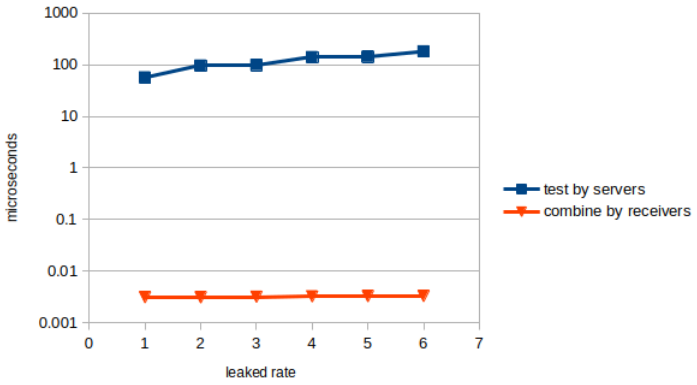

Fig. 6. Test and Combine runtimes for $d = 3$ servers in the honest majority setting ($t = 2$). Tests are run by detection servers. Combination of test results is run by receivers and is roughly five orders of magnitude faster than testing on servers.

References

1. Aztec network. https://aztec.network/
2. Monero. https://www.getmonero.org/
3. Multikey FMD. https://github.com/larraia/multikey-fmd
4. Namada blockchain. https://namada.net/
5. Zcash. https://z.cash/
6. Abdalla, M., Bellare, M., Rogaway, P.: The oracle diffie-hellman assumptions and an analysis of DHIES. In: Naccache, D. (ed.) Topics in Cryptology - CT-RSA 2001, The Cryptographer's Track at RSA Conference 2001, San Francisco, CA, USA, April 8-12, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2020, pp. 143–158. Springer (2001). https://doi.org/10.1007/3-540-45353-9_12, https://doi.org/10.1007/3-540-45353-9_12
7. Anon: View Tag Batch PIR. https://forum.aztec.network/t/proposal-view-tag-batch-pir/3193/, accessed: 2025-03-07
8. Backes, M., Kate, A., Manoharan, P., Meiser, S., Mohammadi, E.: Anoa: A framework for analyzing anonymous communication protocols. In: 2013 IEEE 26th Computer Security Foundations Symposium, New Orleans, LA, USA, June 26-28, 2013. pp. 163–178. IEEE Computer Society (2013). https://doi.org/10.1109/CSF.2013.18, https://doi.org/10.1109/CSF.2013.18
9. Beck, G., Len, J., Miers, I., Green, M.: Fuzzy message detection. In: Kim, Y., Kim, J., Vigna, G., Shi, E. (eds.) CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021. pp. 1507–1528. ACM (2021). https://doi.org/10.1145/3460120.3484545, https://doi.org/10.1145/3460120.3484545
10. Beerliová-Trubíniová, Z., Hirt, M.: Perfectly-secure MPC with linear communication complexity. In: Canetti, R. (ed.) Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008. Lecture Notes in Computer Science, vol. 4948, pp. 213–230. Springer (2008). https://doi.org/10.1007/978-3-540-78524-8_13, https://doi.org/10.1007/978-3-540-78524-8_13

11. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-privacy in public-key encryption. In: Boyd, C. (ed.) Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2248, pp. 566–582. Springer (2001). https://doi.org/10.1007/3-540-45682-1_33, https://doi.org/10.1007/3-540-45682-1_33
12. Bellare, M., Boldyreva, A., Staddon, J.: Randomness re-use in multi-recipient encryption schemeas. In: Desmedt, Y. (ed.) Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2567, pp. 85–99. Springer (2003). https://doi.org/10.1007/3-540-36288-6_7, https://doi.org/10.1007/3-540-36288-6_7
13. Bellare, M., Palacio, A.: The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In: Franklin, M.K. (ed.) Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings. Lecture Notes in Computer Science, vol. 3152, pp. 273–289. Springer (2004). https://doi.org/10.1007/978-3-540-28628-8_17, https://doi.org/10.1007/978-3-540-28628-8_17
14. Damgård, I.: Towards practical public key systems secure against chosen ciphertext attacks. In: Feigenbaum, J. (ed.) Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings. Lecture Notes in Computer Science, vol. 576, pp. 445–456. Springer (1991). https://doi.org/10.1007/3-540-46766-1_36, https://doi.org/10.1007/3-540-46766-1_36
15. Grigg Jack: DAGSync: Graph-aware Zcash wallets. https://words.str4d.xyz/dagsync-graph-aware-zcash-wallets/, accessed: 2025-03-07
16. Groth, J., Maller, M.: Snarky signatures: Minimal signatures of knowledge from simulation-extractable snarks. IACR Cryptol. ePrint Arch. p. 540 (2017), http://eprint.iacr.org/2017/540
17. Jakkamsetti, S., Liu, Z., Madathil, V.: Scalable private signaling. IACR Cryptol. ePrint Arch. p. 572 (2023), https://eprint.iacr.org/2023/572
18. Krawczyk, H., Rabin, T.: Chameleon signatures. In: Proceedings of the Network and Distributed System Security Symposium, NDSS 2000, San Diego, California, USA. The Internet Society (2000), https://www.ndss-symposium.org/ndss2000/chameleon-signatures/
19. Le, D.V., Hurtado, L.T., Ahmad, A., Minaei, M., Lee, B., Kate, A.: A tale of two trees: One writes, and other reads. Proc. Priv. Enhancing Technol. **2020**(2), 519–536 (2020). https://doi.org/10.2478/POPETS-2020-0039, https://doi.org/10.2478/popets-2020-0039
20. Liu, W., Liu, Z., Nguyen, K., Yang, G., Yu, Y.: A lattice-based key-insulated and privacy-preserving signature scheme with publicly derived public key. In: Chen, L., Li, N., Liang, K., Schneider, S.A. (eds.) Computer Security - ESORICS 2020 - 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14-18, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12309, pp. 357–377. Springer (2020). https://doi.org/10.1007/978-3-030-59013-0_18, https://doi.org/10.1007/978-3-030-59013-0_18
21. Liu, Z., Sotiraki, K., Tromer, E., Wang, Y.: Snake-eye resistant PKE from LWE for oblivious message retrieval and robust encryption. In: Fehr, S., Fouque, P.

(eds.) Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part III. Lecture Notes in Computer Science, vol. 15603, pp. 126–156. Springer (2025). https://doi.org/10.1007/978-3-031-91131-6_5, https://doi.org/10.1007/978-3-031-91131-6_5

22. Liu, Z., Tromer, E.: Oblivious message retrieval. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13507, pp. 753–783. Springer (2022). https://doi.org/10.1007/978-3-031-15802-5_26, https://doi.org/10.1007/978-3-031-15802-5_26
23. Liu, Z., Tromer, E., Wang, Y.: Group oblivious message retrieval. In: IEEE Symposium on Security and Privacy, SP 2024, San Francisco, CA, USA, May 19-23, 2024. pp. 4367–4385. IEEE (2024). https://doi.org/10.1109/SP54263.2024.00115, https://doi.org/10.1109/SP54263.2024.00115
24. Liu, Z., Tromer, E., Wang, Y.: Perfomr: Oblivious message retrieval with reduced communication and computation. In: Balzarotti, D., Xu, W. (eds.) 33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024. USENIX Association (2024), https://www.usenix.org/conference/usenixsecurity24/presentation/liu-zeyu
25. Liu, Z., Yang, G., Wong, D.S., Nguyen, K., Wang, H.: Key-insulated and privacy-preserving signature scheme with publicly derived public key. In: IEEE European Symposium on Security and Privacy, EuroS&P 2019, Stockholm, Sweden, June 17-19, 2019. pp. 215–230. IEEE (2019). https://doi.org/10.1109/EUROSP.2019.00025, https://doi.org/10.1109/EuroSP.2019.00025
26. Madathil, V., Scafuro, A., Seres, I.A., Shlomovits, O., Varlakov, D.: Private signaling. In: Butler, K.R.B., Thomas, K. (eds.) 31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022. pp. 3309–3326. USENIX Association (2022), https://www.usenix.org/conference/usenixsecurity22/presentation/madathil
27. Okamoto, T., Pointcheval, D.: The gap-problems: A new class of problems for the security of cryptographic schemes. In: Kim, K. (ed.) Public Key Cryptography, 4th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2001, Cheju Island, Korea, February 13-15, 2001, Proceedings. Lecture Notes in Computer Science, vol. 1992, pp. 104–118. Springer (2001). https://doi.org/10.1007/3-540-44586-2_8, https://doi.org/10.1007/3-540-44586-2_8
28. Penumbra Developers: Sender FMD. https://protocol.penumbra.zone/main/crypto/fmd/sender-receiver.html, accessed: 2025-02-10
29. Pu, S., Thyagarajan, S.A.K., Döttling, N., Hanzlik, L.: Post quantum fuzzy stealth signatures and applications. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023. pp. 371–385. ACM (2023). https://doi.org/10.1145/3576915.3623148, https://doi.org/10.1145/3576915.3623148
30. Seres, I.A., Pejó, B., Burcsi, P.: The effect of false positives: Why fuzzy message detection leads to fuzzy privacy guarantees? In: Eyal, I., Garay, J.A. (eds.) Financial Cryptography and Data Security - 26th International Conference, FC 2022, Grenada, May 2-6, 2022, Revised Selected Papers. Lecture Notes in Computer

Science, vol. 13411, pp. 123–148. Springer (2022). https://doi.org/10.1007/978-3-031-18283-9_7, https://doi.org/10.1007/978-3-031-18283-9_7
31. Tankersley George: ZIP 307. https://zips.z.cash/zip-0307, accessed: 2025-03-07
32. Wüst, K., Matetic, S., Schneider, M., Miers, I., Kostiainen, K., Capkun, S.: Zlite: Lightweight clients for shielded zcash transactions using trusted execution. In: Goldberg, I., Moore, T. (eds.) Financial Cryptography and Data Security - 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers. Lecture Notes in Computer Science, vol. 11598, pp. 179–198. Springer (2019). https://doi.org/10.1007/978-3-030-32101-7_12, https://doi.org/10.1007/978-3-030-32101-7_12

A Detection ambiguity of multiFMD2

Theorem 6 (Restatement of Thm. 13 from [9]). *Let* multiFMD2 *the multi-key FMD scheme of Fig. 3. For any PPT adversary $\mathcal{A}$ against the (t,d)-DA-CCA experiment, making q_{H}, q_{G} queries to the random oracles* H *and* G*, respectively, there exists a PPT algorithms $\mathcal{B}_1, \mathcal{B}_2$ such that:*

$$\mathbf{Adv}^{(t,d)\text{-DA-CCA}}_{\mathsf{multiFMD2},\mathcal{A}}(\lambda) \leq \frac{q_{\mathsf{G}}}{q} + \mathbf{Adv}^{\mathsf{CDH}}_{\mathcal{B}_1}(\lambda) + q_{\mathsf{H}} \cdot \mathbf{Adv}^{\mathsf{Gap\text{-}CDH}}_{\mathcal{B}_2}(\lambda)$$

Proof. For any $J = (j_1, \ldots, j_t) \in 2^{[d]}$ we can recover the FMD2 scheme from [9] by setting the detection key to $\tilde{dsk} = (dsk_{j_1}, \ldots, dsk_{j_t})$, and testing with $\tilde{\mathsf{Test}}(f) := \mathsf{Combine}(\mathsf{Test}(dsk_{j_1}, f), \ldots, \mathsf{Test}(dsk_{j_1}, f))$. So essentially, we can reuse the security analysis given in [9].

Formally, any adversary $\mathcal{A}$ against (t,d)-DA-CCA for multiFMD2 can be turned into an adversary $\mathcal{B}$ against the selective experiment DA-CCA from [9] for FMD2.

$\mathcal{B}$ proceed as follows:

- When $\mathcal{A}_1$ outputs rate $(\rho := n, J)$, $\mathcal{B}$ outputs $p := n$ to its own challenger.
- $\mathcal{B}$ is given two public keys $\tilde{pk}^{(i)}$, and also two detection keys $\tilde{dsk}^{(i)}$ each with n scalars $\boldsymbol{x}^{(i)}$. $\mathcal{B}$ partitions $\boldsymbol{x}^{(i)}$ appropriately to form the two vectors of t detections keys $\boldsymbol{dsk}^{(i)}_J := (dsk^{(i)}_{j_1}, \ldots dsk^{(i)}_{j_t})$ and gives $pk^{(i)}, \boldsymbol{dsk}^{(i)}_J$ to $\mathcal{A}$.
- $\mathcal{B}$ answers queries to $\mathcal{A}$'s oracle by querying its own oracle, and then outputs what $\mathcal{A}$ outputs.

In [9, Appendix F] it is proved that the advantage of $\mathcal{B}$ is upper bounded as per the theorem statement. □

B Fuzziness (proof of Theorem 2)

Proof. The proof is virtually the same as the proof of the fuzziness of FMD2 [9].

The number of secret scalars from sk_1 included in $\boldsymbol{dsk}_J = (dsk_{j_1}, \ldots, dsk_{j_\ell})$ is exactly $s = \sum_{k=1}^{\ell} n_{j_k}$. The probability that any of these scalars matches those in sk_0 is negligible as they are sampled uniformly in $\mathbb{Z}_q$. Using that H is a random

oracle, the probability that all the s decryption (bit) keys $k_j^{(0)}, k_j^{(1)}$ used in Flag and Test respectively, match is 2^{-s}. This is negligible in λ since $s \leq \gamma$ and $\gamma = poly(\lambda)$. □