Oblivious Message Retrieval

Zeyu Liu[1], Eran Tromer[1,2]

[1]Columbia University
[2]Tel Aviv University

Abstract. Anonymous message delivery systems, such as private messaging services and privacy-preserving payment systems, need a mechanism for recipients to retrieve the messages addressed to them without leaking metadata or letting their messages be linked. Recipients could download all posted messages and scan for those addressed to them, but communication and computation costs are excessive at scale.

We show how untrusted servers can detect messages on behalf of recipients, and summarize these into a compact encrypted digest that recipients can easily decrypt. These servers operate obliviously and do not learn anything about which messages are addressed to which recipients. Privacy, soundness, and completeness hold even if everyone but the recipient is adversarial and colluding (unlike in prior schemes).

Our starting point is an asymptotically-efficient approach, using Fully Homomorphic Encryption and homomorphically-encoded Sparse Random Linear Codes. We then address the concrete performance using bespoke tailoring of lattice-based cryptographic components, alongside various algebraic and algorithmic optimizations. This reduces the digest size to a few bits per message scanned. Concretely, the servers' cost is $\sim$\$1 per million messages scanned, and the resulting digests can be decoded by recipients in under $\sim$20ms. Our schemes can thus practically attain the strongest form of receiver privacy for current applications such as privacy-preserving cryptocurrencies.

1 Introduction

End-to-end encryption of message content is well understood and widely practiced. However, metadata about which messages were sent and received by whom, and when, can yield abundant sensitive information via traffic analysis and deductions against auxiliary information. Protecting metadata is thus crucial to anonymous message delivery systems [7] such as anonymous messaging [61,20,42], privacy-preserving analytics [11], and privacy-preserving cryptocurrencies [48,8].

Yet, the problem of protecting communication metadata remains an open challenge for many applications, especially when privacy, scalability, efficiency and decentralization are all crucial. This challenge is well exemplified by metadata protection in privacy-preserving cryptocurrencies, such as Zcash [8,30] and Monero [48]. These convey digital asset transaction on a public ledger (blockchain),

while keeping the contents of every transaction hidden from all but the counterparties to the transaction (and those they elect to expose it to), using cryptographic protocols utilizing encryption and zero-knowledge proofs. Moreover, the underlying ledger is permissionless, decentralized and widely replicated, allowing anyone to send transactions over the Internet while anonymizing their IP address via standard means such as the Tor network. Supposedly, metadata leaks are thus eliminated.[1]

However, a crucial point lingers. From a receiver's perspective, a transaction pertinent to them could appear anywhere in the ledger. If the receiver has a full copy of the ledger (a "full node" in blockchain parlance), then it could scan it to identify pertinent transactions, but the requisite communication, storage and computation cost may far exceed the capabilities of recipients (e.g., already today, such ledgers are many GB in size; consider, then, wallet apps running on computationally-weak mobile devices, with little storage, using slow or expensive network connections).

How, then, can recipients efficiently *detect* which messages are pertinent to them, and *retrieve* the content of these messages? In general, we consider a bulletin board consisting of numerous messages, with arbitrary application-specific content (of fixed size). Each message is *pertinent* to a single recipient (identified by their public address) to which it was sent, and *impertinent* to other recipients. A recipient, in lieu of receiving and scanning the whole ledger (*full-scan*), may enlist the help of servers, which we call *detectors*, that will help them detect their pertinent messages and retrieve the content of these messages.

One approach is for the recipient to provide the detector with a "detection key" or "incoming viewing key", that allows the detector to check, for each bulletin board message, whether it is pertinent to that recipient [48,30]. The pertinent messages can then be stored and forwarded to the client. Unfortunately, this exposes metadata to the detector, and thus also to anyone who subverted or coerced the detector, whether in real time or retroactively. Furthermore, such long-lived detection key enables devastating deanonymization attacks.[2]

The problem of *Oblivious Message Detection (OMD)* is to perform such detection without revealing any information to the detectors about which messages are pertinent. This is done today in Zcash via the ZIP-307 "light client" protocol [27], which is essentially optimized full-scan: convert each message to a compact format which contains just enough information for the recipient to check for pertinence, and then send *all* of these compacted messages to the recipient for processing. In practice, this process can take hours even for a relatively lightly-used chain, and is recognized as a severe usability and scalability issue.

[1] In reality, today's blockchain privacy solutions suffer from assorted metadata leaks such as variability in transaction record size, ill-defined cryptographic guarantees, inadequate network-level anonymization, exposure of amounts at the interface between transaction pools, and operational mistakes. These are outside our scope.

[2] For example, an adversary who acquired a detection key can easily ascertain whether that key belongs to a given person, by simulating a transaction to that person and seeing if it matches that detection key; if so, then all past and future transaction pertinent to that recipient become linked.

Furthermore, the full task is *Oblivious Message Retrieval (OMR)*, where the recipient also gets the content (payload) of their pertinent messages. Given just detection, the recipient would still have to retrieve every detected pertinent message by some means. Naively querying the detector (or some other server) again leaks the pertinency metadata. This could be mitigated by using Private Information Retrieval, mixnets, or decoy traffic, but at high cost or/or ill-defined security. This is recognized by practitioners as an important open problem.[3]

These problems have been studied by two recent works, which made significant headway, but still carry significant drawbacks. Fuzzy Message Detection (FMD) [7] is based on inducing false-positive decoys into detection, and presents a difficult tradeoff between security (a high decoy density is needed to foil adversarial analysis) and efficiency (these decoys all entail costs). Private Signaling (PS) [44][4] provides full privacy only if a single detector serves all recipients in the system, and moreover requires either trusted hardware such as Intel SGX, or a pair of servers that are in constant communication but trusted not to collude. Furthermore, both works assume for correctness that all senders and recipients behave honestly, and are susceptible to amplified Denial-of-Service attacks, where an adversary can induce detector and/or recipient work that is disproportional to the number of messages they place on the bulletin board. They also exhibit linkability between detection queries and identities. (See further discussion below.) We thus pose the problem:

Is it feasible to achieve oblivious message retrieval/detection that is fully private, DoS-resistant, unlinkable, trustless, and practical?

1.1 Our Contributions

In this paper, we propose schemes that fulfill all of the above requirements. Our approach is based on homomorphic encryption using lattice-based cryptography.

Strong Security Definitions. We formally define the notions of Oblivious Message Retrieval and Detection. Our definitions capture natural notions of correctness and privacy, and moreover capture two important security notions that prior works failed to capture or achieve:

- Prior works are vulnerable to *amplified Denial-of-Service* attacks in the realistic threat model where there exist malicious participants (senders or recipients) in the messaging system. Our strengthened notion says that even if arbitrary system participants are adversarial and colluding, they cannot induce more errors or costs than honest participants.
- Prior works are vulnerable to *key-linkability* attacks, which tie retrieval actions to public identities (or to each other) since public keys are themselves reused or linkable. We achieve a strong notion of key unlinkability, preventing these.

[3] E.g., Zcash developers [32] deem it an "action item" that the "lightwalletd [server] learns which transactions belong to the wallet", and described the popular mitigation of using decoy fetches as "security theatre" that fails to achieve unlinkability.

[4] Private Signaling [44] is a concurrent and independent work, available only as a preprint at the time of this paper's submission.

Possibility of Compact OMR and OMD. We show that Fully Homomorphic Encryption (FHE) can be used to achieve message retrieval and detection with full privacy against any (computationally-bounded) adversary. Moreover, we show that FHE can be used to distill the full bulletin board into a *compact digest* of size that is near-linear in the number of *pertinent* messages, rather than all messages in the bulletin board.

Our approach is based on annotating messages with *clues* to their pertinence, having the detector inspect these clues using FHE and pack the pertinent messages into a compact digest using homomorphically-encoded sparse random linear codes, and having the recipient algebraically reconstruct the messages from the decrypted digest. The homomorphic packing and encoding stages are reminiscent of techniques used in batch Private Stream Search [51] and Private Information Retrieval [5,4], adapted and optimized to the OMR/OMD setting.

Practical OMR/OMD. Generic use of FHE is notoriously inefficient. We tackle this by a series of optimizations to drastically improve concrete performance. Our techniques include bespoke composition of several different lattice-based schemes (specifically PVW [54] and BFV [13,24]) and extensions thereto, utilization of SIMD-like packed operations, using a tailored Sparse Random Linear Code, optimization of multiplicative depth to avoid expensive bootstrapping, and parameter tuning.

We thereby obtain several concrete schemes, with different tradeoffs, that achieve our security notions under standard lattice hardness assumptions (Ring-LWE). Security is thus plausibly postquantum. DoS-resistance holds under an additional natural conjecture about LWE-based encryption, or using zk-SNARKs.

Implementation and Evaluation. We implemented our schemes as an open-source C++ library [50] and measured their concrete performance for a variety of parameters and in comparison to prior work. Salient observations include:

- Detector-to-recipient computation: for Bitcoin-scale parameter settings, our OMR schemes have lower detector-to-recipient communication than any other known retrieval scheme: ∼9 bits per message for retrieval, and ∼4.5 bits/msg for just detection. For even larger parameters, the amortized retrieval digest size drops below 1 bit/msg.
- Recipient's computation is faster than any other known retrieval scheme, e.g., ∼20 msec to retrieve 50 pertinent messages out of 500,000.
- Detector's cost for full retrieval is higher than in related schemes, but still quite practical at ∼0.065 sec/msg (∼$1.02 per million messages) on a small cloud VM. For just detection, our scheme is faster than any other known scheme, including those based on trusted hardware.

Thus, our schemes are attractive when recipients are limited in bandwidth or computation speed. The one drawback is a one-time cost of sending a key of size ∼129 MB to a detector.

Cryptocurrency Integration. We discuss key design points in integrating our scheme with a blockchain-based privacy-preserving cryptocurrency (exemplified by Zcash) including protocol and costs aspect. We conclude that our

Scheme		Privacy			Soundness + completeness	
		Detection	Retrieval	Assumptions	Assumptions	Overflows
Full Scan	[27]	Full		ECDH + Auth Enc	None	None
FMD1	[7]	pN-msg-anonymity, fixed $p = 2^{-i}$		PKE	Honest S&R	None
FMD2	[7]	pN-msg-anonymity, dynamic p		PKE		None
PS1	[44]	Full	N/A	TEE (SGX) + DDH + PKE		Undetected
PS2	[44]	Partitioned across detectors	N/A	Communicating non-colluding servers + Garbled Circuit + Unforgeable Sigs		Undetected
OMRt1	§5.3	Full + full-key-unlinkability		FHE		Detected
OMRp1	§6.3	Full + full-key-unlinkability		Ring-LWE	Honest S&R or Conj. 1 or zk-SNARK	Detected
OMRp2	§6.4	Full + full-key-unlinkability		Ring-LWE		Detected

Table 1: Comparison of privacy guarantees and assumptions. Here, pN-msg-anonymity means the recipient's messages are hidden among decoy (false-positive) messages which are a p fraction of the total N messages. "Partitioned across detectors" privacy means that if multiple detectors are used for scalability, then the anonymity set is just the recipients served by the same detector. "Honest S&R" means all senders are honest when generating clue for messages, and all receivers are honest when generating their clue keys. PS1 can be modified for key unlinkability (cf. Section 8). TEE means Trusted Execution Environment.

scheme is compatible with existing protocols, and that the cost of detection service would be ∼$1 of Cloud Computing per million messages scanned (i.e., similar magnitude to the total monthly transaction flow in all privacy-preserving cryptocurrencies).

2 Related Work

2.1 Message Detection

Privacy-preserving message detection and retrieval has been studied in several prior and concurrent works, discussed below. Table 1 summarizes the functionality and privacy aspects of these scheme, compared to our Oblivious Message Retrieval schemes presented in Sections 5 to 8. See also Section 9 (e.g., Table 2) for comparison of concrete performance.[5]

The closest prior work, addressing message detection in a sense similar to ours, is Fuzzy Message Detection (FMD) [7]. In Table 1 in this section, FMD1 and FMD2 refer to Figures 3 and 4 in [7], respectively. FMD provides the privacy we call *pN-msg-anonymity*: the detector can observe which messages were flagged as pertinent, but hidden among many intentional false-positives which are indistinguishable from the truly pertinent messages. These decoys are a p fraction of the N messages in the bulletin. Like other decoy-based privacy notions, pN-msg-anonymity introduces uncertainty into naive analysis, but still allows many privacy-violating deductions, especially given recurring traffic or an active adversary [41].

[5] For reference, these tables also include *full scan*, which is the straightforward linear-communication approach where the recipient scans each message (or a relevant part thereof) in the whole bulletin board (used, e.g., in the Zcash light wallet [27]).

Another closely related work (concurrent and independent) is Private Signaling (PS) [44]. Their model assumes that *signals* (analogous to our clues) are sent directly to servers (analogous to our detectors). Servers handle these signals in a privacy-preserving way, using one of two mechanisms. The *single-server (PS1)* scheme relies on a Trusted Execution Environment (Intel SGX), running on the detector server. This is a very strong trust assumption (especially given the many past attacks on SGX [44, §2.2.1]), with total privacy failure if violated. The *two-server (PS2)* scheme instead relies on secure multiparty computation between two servers, using garbled circuit. The two servers jointly serve as a detector, and (shares of) signals are sent to both of them. Privacy holds as long as these servers communicate but do not collude nor leak their secrets. Note that in this model, if either of the servers is compromised and its data leaks, then the other server can passively deduce the protected information (the recipient of every message), even retroactively.

DoS Attacks and Key Linkability. Both FMD and PS do not provide soundness guarantees if clues, or clue keys (public addresses), are generated maliciously; they are thus subject to Denial-of-Service (DoS) attacks (see Section 7). They also let detection queries be linked to each other, as well as to clue keys, hence to public identities (see Section 8).

Retrieval. FMD originally addresses only the detection problem, but it naturally extends to retrieval by attaching the full payload for each message; we use this variant when we report the performance of FMD1 and FMD2 as retrieval schemes. PS also addresses the detection problem, but we cannot directly add payloads without breaking privacy; the PS schemes would need to be nontrivially modified to collect and send the payloads in some privacy-preserving way.

2.2 Other Works

Private Retrieval. There are also works that deal with retrieval once indices are already known, i.e., *Private Information Retrieval (PIR)* [19,36]. The *keyword PIR* variant [18,6] assumes the client knows explicit plaintext keywords to search for (similarly to PSS, discussed below). Some works [46,62,39] rely on trusted hardware, such as Intel SGX, for retrieval privacy. Others offer weak decoy-based notions of privacy [27,32]. Our model does not make any of these assumptions or relaxations.

Our construction shares some techniques with state-of-the-art batch PIR protocols [5,4], namely homomorphic accumulation of messages using FHE and linear coding. However, in our case the retrieved indices are not known a priori to the recipient, so our scheme includes encoding and homomorphic decoding of suitable clues, which are then used to guide the accumulation. Furthermore, we use a different coding technique, adapted to this setting.

Other Private Messaging Aspects. The complementary problem of maintaining *sender privacy* when posting on the bulletin board is addressed by Riposte [20], Signal's Sealed Sender [42] and its improvement [45]. Alpenhorn [38] addresses privacy-preserving connection establishment. As discussed in [7],

Alpenhorn essentially uses identity-based encryption and a trusted server to reduce that problem to the privacy-preserving message retrieval problem studied by this paper. The Vuvuzela [29] mixnet offers differential privacy for sender and receiver metadata. Rather than a bulletin board model, it assumes an online model where users to remain connected at all times (lest their messages get dropped). It employs a cluster of servers, of which at least one is assumed to be honest. Bandwidth cost is high (e.g, 30 GB/month for users and 416 TB/month for servers [29]).

Private Stream Search. Private Stream Search (PSS) was introduced by Ostrovsky and Skeith [51] and followups [21,9,25]. It allows a client to search a keyword over a database of documents and download the ones with such a keyword without revealing the keyword to the server.

In terms of techniques, our use of homomorphic accumulation and linear coding is shared also with PSS. However, in PSS, the elements being sought are plaintext words, which allows for relatively simple protocols. In OMR, conversely, the analogues are "clues" which must be randomly sampled and unlinkable, to hide the identity of the recipients. Therefore, we employ FHE to compute a complicated circuit for homomorphic decryption (and amplification) before retrieval, which creates very different cost tradeoffs, optimizations and implementation details compared to PSS. Furthermore, the past PSS works mainly focused on optimizing the communication cost, at the cost of very high server-server computation (e.g., [25], using Reed-Solomon coding, has a server perform do computation superlinear in the number of pertinent messages, for every bulletin message); we use different coding techniques to attain practicality in the the OMR setting.

3 Model and Definitions

System Model. In this section, we define the model and the problem of Oblivious Message Detection and Retrieval. The system components and their high-level properties are as follows. See also the high-level components of Fig. 1 (but disregard, for now, the detailed schemes like PVW or BFV; these will be introduced in Section 6)

A *bulletin board* (or *board* for short), denoted BB, contains N messages (e.g., blockchain transactions). Each message is sent from some sender and addressed to some recipient, whose identities are supposed to remain private.

A message is a pair (x_i, c_i) where x_i is the *payload* to convey, and c_i is a *clue* which helps notify the intended recipient (and only them) that the message is addressed to them.

We denote the payload space $\mathcal{P} = \{0,1\}^{\tilde{n}}$ for some $\tilde{n} \in \mathbb{Z}^+$, and the clue space $\mathcal{C}$ (typically ℓ number of ciphertexts in some encryption system). The whole board BB (i.e., all payloads and clues) is public. (In applications, the payloads will typically be end-to-end encrypted.)

At any time, any potential recipient p may retrieve the messages addressed to them in BB. We call these messages *pertinent* (to p), and the rest are *impertinent*.

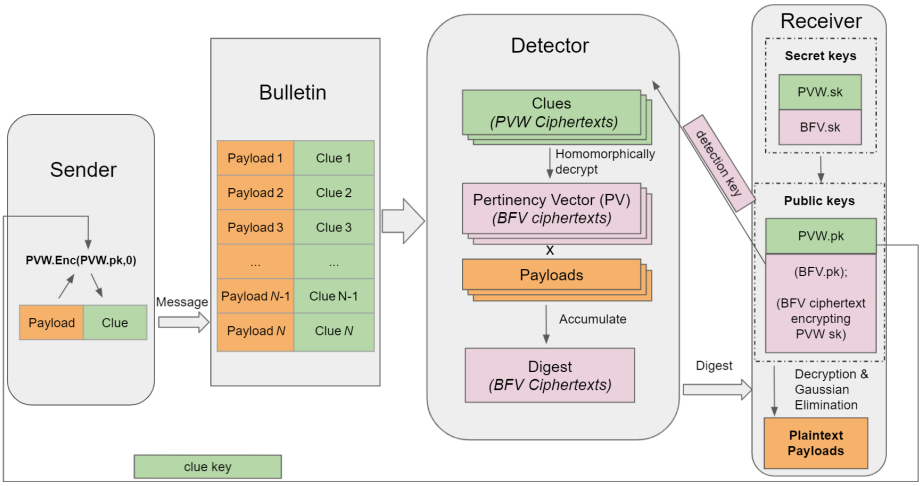


Fig. 1: System components, and major internal parts of the practical schemes.

A server, called a *detector*, helps the recipient p *detect* which message indices in BB are pertinent to them, or *retrieve* the payloads of the pertinent messages. This is done obliviously: even a malicious detector learns nothing about which messages are pertinent. The recipient gives the detector their *detection key*, and a bound $\bar{k}$ on the number of pertinent messages they expect to receive. The detector then accumulates all of the messages in BB into string M, called the *digest*, and sends it to the recipient p. The digest M should be much smaller than the board BB (ideally, proportional to $\bar{k}$).

The recipient p processes M to recover all of the pertinent messages with high probability, assuming a semi-honest detector and that the number of pertinent messages did not exceed $\bar{k}$. The *false negative rate* (probability that a pertinent message is not recovered from the digest) is denoted by ϵ_{n}. The *false positive rate* (probability that an impertinent message is output by the recovery procedure) is denoted by ϵ_{p}. Both ϵ_{n} and ϵ_{p} are small (e.g., under 10^{-9}).

There may be many detectors, and each may support many recipients. Outside our scope are application-specific aspects such as payload encryption, contact discovery and key establishment, the privacy-preserving mechanism by which messages are posted to the board, or how recipients subscribe with detectors. See related discussion in Sections 2 and 10.

Threat Model (non-DoS). We assume a computationally-bounded adversary that can read all public information, including all board messages, all public keys in the system, and all communication between the detector and the receiver. It can also honestly generate new messages (with any payload) and post them on the board, as well as honestly generate new clue keys and induce other parties to generate messages addressed to those keys. For soundness and completeness, we require the detectors, senders, and recipients to be honest but curious; they may collude by sharing information. In regard to privacy, we let all parties in the

systems be malicious and colluding (including detectors, senders, and recipients), except of course for the sender and recipient of the message(s) whose privacy is to be protected. (A stronger Denial-of-Service (DoS) threat model will be defined in Section 7.1. Key unlinkability will be defined in Section 8.)

3.1 Definitions

Oblivious Message Retrieval. We capture the notion of an Oblivious Message Retrieval (OMR) scheme as follows.

Definition 1 (Oblivious Message Retrieval (OMR)).
An Oblivious Message Retrieval scheme has the following PPT algorithms:

- $\mathsf{pp} \leftarrow \mathsf{GenParams}(1^\lambda, \epsilon_\mathrm{p}, \epsilon_\mathrm{n})$*: takes a security parameter* λ*, a false positive rate* ϵ_p *and a false negative rate* ϵ_n*, and outputs public parameters* pp*.* pp *is implicitly taken by the following algorithms.*
- $(\mathsf{sk}, \mathsf{pk} = (\mathsf{pk}_\mathsf{clue}, \mathsf{pk}_\mathsf{detect})) \leftarrow \mathsf{KeyGen}()$ *: outputs a secret key* sk *and a public key* pk *consisting of a clue key* $\mathsf{pk}_\mathsf{clue}$ *and a detection key* $\mathsf{pk}_\mathsf{detect}$*.*
- $c \leftarrow \mathsf{GenClue}(\mathsf{pk}_\mathsf{clue}, x)$ *: takes a clue key and a payload* $x \in \mathcal{P}$*, and output a clue* $c \in \mathcal{C}$*.*
- $M \leftarrow \mathsf{Retrieve}(\mathsf{BB}, \mathsf{pk}_\mathsf{detect}, \bar{k})$ *: takes as input a board* $\mathsf{BB} = \{(x_1,c_1), \ldots, (x_N,c_N)\}$ *for some size* N*, a detection key* $\mathsf{pk}_\mathsf{detect}$*, and an upper bound* $\bar{k}$ *on the number of pertinent messages addressed to that recipient; and output a digest* M*.*
- $\mathsf{PL} \leftarrow \mathsf{Decode}(M, \mathsf{sk})$ *: takes the digest* M *and corresponding secret key* sk*, and outputs either a decoded payload list* $\mathsf{PL} \subset \mathcal{P}^k$ *or an overflow indication* $\mathsf{PL} = \mathsf{overflow}$*.*

To define completeness, soundness, and privacy, we first define the notion of board generation:

> ***Definition 2 (board generation).*** *Given* pp*, and* N *which is the number of messages: Arbitrarily choose the number of recipients* $1 \le p \le N$*, and a partition of the set* $[N]$ *into* p *subsets* $S_1, \ldots, S_p$ *representing the indices of messages addressed to each party. Also arbitrarily choose unique payloads* $(x_1, \ldots, x_N)$*. For each recipient* $i \in [p]$*: generate keys* $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KeyGen}()$*, and for each* $j \in S_i$*, generate* $c_j \leftarrow \mathsf{GenClue}(\mathsf{pk}_i, x_j)$*. Then, output the board* $\mathsf{BB} = \{(x_1,c_1), \ldots, (x_N,c_N)\}$*, the set* S_1*, and* $(\mathsf{sk}_1, \mathsf{pk}_1 = (\mathsf{pk}_{\mathsf{clue}\,1}, \mathsf{pk}_{\mathsf{detect}\,1}))$*.*[6]

The scheme must satisfy the following properties:

- *(Completeness) Let* $\mathsf{pp} \leftarrow \mathsf{GenParams}(1^\lambda, \epsilon_\mathrm{p}, \epsilon_\mathrm{n})$*. Set any* $N = \mathsf{poly}(\lambda)$*, and* $0 < \bar{k} \le N$*. Let a board* BB*, a set* S *of pertinent messages, and a key pair* $(\mathsf{sk}, \mathsf{pk} = (\mathsf{pk}_\mathsf{clue}, \mathsf{pk}_\mathsf{detect}))$ *be generated as in Definition 2 for any choice of* p*, partition and payloads therein. Let* $M \leftarrow \mathsf{Retrieve}(\mathsf{BB}, \mathsf{pk}_\mathsf{detect}, \bar{k})$ *and* $\mathsf{PL} \leftarrow \mathsf{Decode}(M, \mathsf{sk})$*. Let* $k = |S|$ *(the number of pertinent messages in*

[6] That is, S_1 is the indices of messages pertinent to the recipient whose keys are $\mathsf{sk}_1, \mathsf{pk}_1$, which wlog is the first recipient.

S). Then either $k > \bar{k}$ *and* $\mathsf{PL} = \mathsf{overflow}$*, or* $\Pr[x_j \in \mathsf{PL} \mid j \in S] \geq (1 - \epsilon_{\mathrm{n}} - \mathsf{negl}(\lambda))$ *for all* $j \in [N]$.

- *(Soundness) For the same quantifiers as in Completeness:* $\Pr[(x_j \in \mathsf{PL} \mid j \notin S) \vee (x \in \mathsf{PL} \mid x \notin (x_1, \ldots, x_N))] \leq (\epsilon_{\mathrm{p}} + \mathsf{negl}(\lambda))$ *for all* $j \in [N]$.
- *(Computational privacy) For any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*: let* $\mathsf{pp} \leftarrow \mathsf{GenParams}(\epsilon_{\mathrm{p}}, \epsilon_{\mathrm{n}})$*,* $(\mathsf{sk}, \mathsf{pk} = (\mathsf{pk}_{\mathsf{clue}}, \mathsf{pk}_{\mathsf{detect}})) \leftarrow \mathsf{KeyGen}()$ *and* $(\mathsf{sk}', \mathsf{pk}' = (\mathsf{pk}'_{\mathsf{clue}}, \mathsf{pk}'_{\mathsf{detect}})) \leftarrow \mathsf{KeyGen}()$*. Let the adversary choose a payload* x *and remember its state:* $(x, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pp}, \mathsf{pk}, \mathsf{pk}')$*. Let* $c \leftarrow \mathsf{GenClue}(\mathsf{pk}_{\mathsf{clue}}, x)$ *and* $c' \leftarrow \mathsf{GenClue}(\mathsf{pk}'_{\mathsf{clue}}, x)$*. Then:* $|\Pr[\mathcal{A}_2(\mathsf{st}, c) = 1] - \Pr[\mathcal{A}_2(\mathsf{st}, c') = 1]| \leq \mathsf{negl}(\lambda)$.

An OMR scheme is compact *if it moreover satisfies the following:*

- *(Compactness) An* OMR *scheme is compact if for* $\mathsf{pp} \leftarrow \mathsf{GenParams}(1^\lambda, \epsilon_{\mathrm{p}}, \epsilon_{\mathrm{n}})$*,* $(\mathsf{sk}, \mathsf{pk} = (\mathsf{pk}_{\mathsf{clue}}, \mathsf{pk}_{\mathsf{detect}})) \leftarrow \mathsf{OMR.KeyGen}()$*, for any board* $\mathsf{BB} = \{(x_1, c_1), \ldots, (x_N, c_N)\}$*, letting* $M \leftarrow \mathsf{Retrieve}(\mathsf{BB}, \mathsf{pk}_{\mathsf{detect}}, \bar{k})$*, it always holds that:* $|M| = \mathsf{poly}(\lambda, \log N) \cdot \log \epsilon_{\mathrm{p}}^{-1} \cdot \tilde{O}(\bar{k} + \epsilon_{\mathrm{p}} N)$.

Stronger definitions. The above definition assumes that the board generation is done honestly (though possibly adaptively). For maliciously generated board (e.g., clues not form $\mathsf{GenClue}$), see Section 7 for related attacks and a strengthening that prevents them. For simplicity, we also assume the payloads $(x_i)_{i \in [N]}$ are all unique (cf. Definition 2), but it can be naturally extended to the case of duplicate payloads.

Oblivious Message Detection. As a stepping stone towards OMR, we define a weaker functionality, *Oblivious Message Detection (OMD)*, in which the digest merely enables *detection* of the indices of the pertinent messages, but doesn't convey the corresponding payloads. Instead of a function $\mathsf{Retrieve}(\mathsf{BB}, \mathsf{pk}_{\mathsf{detect}}, \bar{k})$ that enables decoding the pertinent message *payloads*, OMD offers a function $\mathsf{Detect}(\mathsf{BB}, \mathsf{pk}_{\mathsf{detect}}, \bar{k})$ that enables decoding the pertinent message *indices*. Completeness then says that the index of each of the pertinent messages is included in PL with probability $\geq 1 - \epsilon_{\mathrm{n}} - \mathsf{negl}(\lambda)$. Soundness says that the index of each non-pertinent message is included in PL with probability $\leq \epsilon_{\mathrm{p}} + \mathsf{negl}(\lambda)$

4 Preliminaries

Notation. All logarithms are expressed in base 2 if not indicated otherwise. Let $[n]$ denote the set $\{1., \ldots, n\}$, and let $[n, m]$ denote the set $\{n, ..., m\}$. We use $P(\ldots; s)$ to denote a randomized algorithm P running with randomness s. Our big-O notation, when applied to computational and communication complexity, absorbs the security parameter λ and payload size $\tilde{n}$ as a constant. When we use a pseudorandom function (PRF), we assume that its range is as implied by the context (i.e., that the PRF's outputs are computationally indistinguishable from uniform over that range). We use $\langle i \rangle_{\mathsf{sk}}$ to denote the space of ciphertexts that decrypt to i under sk.

4.1 LWE Encryption

Our constructions will be optimized by using lattice-based encryption. We will use the PVW [54] variant of Regev's LWE-based encryption [57], defined as follows at a high level, and details in full version. The public parameters $\mathsf{pp}_{\mathsf{LWE}} = (n, \ell, w, q, \sigma) \leftarrow \mathsf{PVW.GenParams}(1^\lambda, \ell, q, \sigma)$ contain secret key dimension n, the number of LWE samples in the public key w, noise standard deviation σ, ciphertext modulus q, and number of bits of the plaintext modulus ℓ. ($\mathsf{pp}_{\mathsf{LWE}}$ is assumed to be implicitly taken by the other algorithms.)

For key generation $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{PVW.KeyGen}()$ and encryption $\mathsf{ct} = (\vec{a},\vec{b}) \leftarrow \mathsf{PVW.Enc}(\mathsf{pk}, \vec{m})$). Most crucial is the decryption algorithm $\vec{m} \leftarrow \mathsf{PVW.Dec}(\mathsf{sk}, ct = (\vec{a},\vec{b}))$, which will be evaluated homomorphically. It computes ℓ inner products $\vec{d} = \vec{b} - \mathsf{sk}^T\vec{a} \in \mathbb{Z}_q^\ell$ and outputs $\vec{m} \in \mathbb{Z}_2^\ell$ where $m_i = 1$ if $\lceil q/4 \rceil \leq b[i] < \lceil 3q/4 \rceil$ and $m_i = 0$ otherwise.

PVW is unconditionally correct (sound), and under the standard *Learning With Error (LWE)* hardness assumption [57,3] it fulfills the standard definitions of semantic security (IND-CPA) and key privacy.

4.2 Homomorphic Encryption

Fully Homomorphic Encryption (FHE) enables evaluation of a circuit on encrypted data, such that the resulting ciphertext (when decrypted) is the output of the circuit on the data, but the evaluator learns nothing about the data.

Formally, an (asymmetric) FHE scheme is an encryption with PPT algorithms $\mathsf{GenParams}(1^\lambda), \mathsf{KeyGen}(), \mathsf{Enc}(\mathsf{pk}, m), \mathsf{Dec}(\mathsf{sk}, c)$ fulfilling the standard definitions of semantic security, soundness, and key privacy. Moreover, it has two more PPT algorithms: $\mathsf{Eval}(\mathsf{pk}, (\mathsf{ct}_1, \ldots, \mathsf{ct}_k), C), \mathsf{Recrypt}(\mathsf{pk}, \mathsf{ct})$. $\mathsf{Recrypt}$ function is as defined in [26], i.e., it decrypts homomorphically using the encrypted secret key, thus yielding a fresh re-encryption of the original plaintext. These two new algorithms fulfill the following generalized correctness. Given a circuit C and plaintexts $(m_1, ..., m_k)$, ciphertexts $(ct_1, ..., ci_k)$ which are each either $\mathsf{ct}_i \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}, m_i)$ or $\mathsf{ct}_i \leftarrow \mathsf{FHE.Recrypt}(\mathsf{pk}, \mathsf{FHE.Enc}(\mathsf{pk}, m_i))$, and letting $\mathsf{ct}' \leftarrow \mathsf{FHE.Eval}(\mathsf{pk}, C, (\mathsf{ct}_1, ..., \mathsf{ct}_k))$: $\Pr[\mathsf{FHE.Dec}(\mathsf{sk}, ct') = C(m_1, ..., m_k)] \geq 1 - \mathsf{negl}(\lambda)$. We require an additional property for the FHE scheme:

Definition 3 (Wrong-Key Decryption). *For an FHE scheme with plaintext space $\mathbb{Z}_t$, and $t \geq 2$, letting $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda)$ and $(\mathsf{sk}', \mathsf{pk}') \leftarrow \mathsf{FHE.KeyGen}(1^\lambda)$, $\mathsf{ct} \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}, 1)$, and $m' \leftarrow \mathsf{FHE.Dec}(\mathsf{sk}', \mathsf{FHE.Recrypt}(\mathsf{pk}', ct))$, it holds that: $\Pr[m' = 1] \leq 1/t + \mathsf{negl}(\lambda)$.*

BFV Encryption. We use the Brakerski/Fan-Vercauteran homomorphic encryption scheme [13,24], which we refer to as the BFV scheme. Given a polynomial from the cyclotomic ring $R_t = \mathbb{Z}_t[X]/(X^D + 1)$, the BFV scheme encrypts it into a ciphertext consisting of two polynomials, where each polynomial is from $R_q = \mathbb{Z}_q[X]/(X^D + 1)$ where $q > t$. We refer to t, q, and D as the plaintext modulus, the ciphertext modulus, and the ring dimension, respectively. Each

ciphertext can pack D plaintext group elements $(m_1, \ldots, m_D) \in \mathbb{Z}_t^D$, and "single instruction, multiple data" (SIMD) homomorphic operations can be applied .

BFV is unconditionally correct (sound), and under the standard *Ring-LWE (RLWE)* [43,55] hardness assumption it fulfills semantic security (IND-CPA).

5 Generic OMR and OMD Using FHE

5.1 Oblivious Message Detection Using FHE

We start by constructing Oblivious Message Detection (OMD), with retrieval to be added in Section 5.2 This section's constructions will be based on general fully homomorphic encryption (FHE), and assume all the clues are generated honestly. This serves as a simple proof of theoretical possibility; Section 6 will address practicality and stronger security guarantees.

Non-compact Construction of OMD We start by showing how pertinent messages can be *detected*, with a small digest, using any key-private public-key FHE satisfying Definition 3 (e.g, FHEW/TFHE [22,16]). For simplicity, assume the plaintext space is $\mathbb{Z}_2$.[7] Our didactic starting point is a straightforward, non-compact construction; we then improve it to achieve compactness.

High-Level Idea. Each clue c_i for $i \in [N]$ consists of ℓ ciphertexts, each encrypting the constant 1 under the public key of the party this message is addressed to. The detector, serving recipient p, will use p's FHE public key pk to recrypt all the ciphertexts in all the clues: $c_{i,j} \mapsto c'_{i,j}$ for $j \in [\ell]$. Crucially, note that each such $c'_{i,j}$ will be $\langle 1 \rangle_{\mathsf{sk}}$ if the message is addressed to p, and otherwise be 1 with probability $\leq 1/2 + \mathsf{negl}(\lambda)$ when decrypted by sk, the corresponding secret key for the recipient p (by Definition 3). Thus, for each message, The detector performs an AND gate over all $c'_{i,j}$ ($j \in [\ell]$) to get PV_i (the *Pertinency Vector*) for all $i \in [N]$. Then the detector sends PV vector to the recipient. The recipient decrypts each PV_i using its FHE secret key, and if the result is 1, deems the i-ith message pertinent.

The resulting OMD algorithm $\mathsf{OMDt1}$ pseudocode and analysis is deferred to the full version. .

Theorem 1. *The scheme* $\mathsf{OMDt1}$ *is an Oblivious Message Detection scheme, when instantiated with any fully homomorphic encryption scheme.*

Compact Construction of OMD Our next step is to achieve compactness, i.e., a digest size which depends primarily on the number of *pertinent* messages k, rather than the total number of messages N. Since k itself is a private information that should not be exposed, the digest size (and detector's computation) instead

[7] The following naturally generalizes to plaintext space $\mathbb{Z}_t$ for any prime t. For brevity, we kept this section focused on $\mathbb{Z}_2$, which suffices for its results, and added footnotes to clarify the generalization. Section 6 will use $t > 2$.

need to depend on an *upper bound* $\bar{k}$ on the number of pertinent messages k for the particular recipient (see Definition 1). The bound $\bar{k}$ is given as a parameter to the detector.[8]

The technique used here is reminiscent of that used in [51], where homomorphic operations are used to summarize "documents" according to 0/1 encryptions; in our case the the 0/1 encryptions are indirectly derived by homomorphic computation (as above), and the (implicit) "documents" are our message indices.

High-Level Idea. We start as above, with the detector recrypting these and computing PV_i ciphertexts which are $\langle 1 \rangle_{\mathsf{sk}}$ iff the i-th message is pertinent w.h.p.

Then, we create m *buckets*, for some $m > \bar{k}$ (to be fixed later). Each bucket contains an *accumulator* Acc_i containing a value in $\mathbb{Z}_N$ encrypted under pk, represented as a vector of $\lceil \log(N) \rceil$ ciphertexts of FHE that store the bit-wise binary encoding of the $\mathbb{Z}_N$ element.[9] A bucket also contains a *counter* $\mathsf{Ctr}_{i \in [m]}$, where Ctr_i contains a value in $\mathbb{Z}_{\bar{k}}$ encrypted under pk, represented using binary encoding as well. All buckets and counters initially encrypt 0.

To add the i-th message in the board, the detector computes PV_i as in $\mathsf{OMDt1}$, draws a random bucket index $\mu \in [m]$, and homomorphically adds PV_i to counter Ctr_μ in $\mathbb{Z}_{\bar{k}}$. It also homomorphically computes PV_i and adds it to the accumulator Acc_μ.

Finally, the detector sends all buckets and counters to the recipient. The recipient decrypts these and checks: if the decrypted Ctr_μ is 1, then bucket has a single pertinent message mapped to it, and the decrypted Acc_μ gives the index of that message. If any Ctr_μ decrypts to greater than 1, indicating that several pertinent messages collided in the μ-th bucket, then it outputs $\mathsf{overflow}$.

This method gives us a success rate of $\rho = \prod_{i=1}^{\bar{k}-1} \frac{(m-i)}{m}$. To achieve the desired ϵ_{n}, we amplify by repeating this C times with fresh buckets and counters and re-randomized assignment of messages to buckets, such that $(1-\rho)^C < \epsilon_{\mathrm{n}}$.

Our technique is different from bloom filter as used in [21,17]. With bloom filter, the recipient runtime can easily be $O(N)$, and for $\mathsf{polylog}(N)$ cost, there is an extra false positive rate, which can be avoided using our technique.

Gathering Partial Information. The amplification can be optimized as follows. Even when a set of buckets contains collisions and thus doesn't directly decode to give all pertinent message indices, there will likely be *some* buckets in the set that do yield useful information (i.e., the corresponding counters are $\langle 1 \rangle_{\mathsf{sk}}$). If we gather all such partial information together, we may get the full information. Then, the failure probability is $< 1 - \prod_{i=1}^{\bar{k}-1}(1 - (\frac{i}{m})^C)$.

Parameter Analysis. The scheme requires choice of m and C. If we choose $m = 10\bar{k}$, we can then choose the smallest integer C such that $1 - \prod_{i=1}^{\bar{k}-1}(1 - (\frac{i}{10\bar{k}})^C) \leq \epsilon_{\mathrm{n}}$. We can see that $C = O(\log(\bar{k}) \log(1/\epsilon_{\mathrm{n}}))$ by using union bound. We get similar results for any choice of $m > k$ that is linear in k.

[8] If the actual number of pertinent messages k exceeds the assumed bound $\bar{k}$, then retrieval may fail. The recipient can detect overflow and ask for the detection to be redone with a larger $\bar{k}$. Our scheme gives the exact number of k, as discussed below.

[9] For FHE over $\mathbb{Z}_t$, use $\lceil \log_t(N) \rceil$ ciphertexts per bucket.

Decoding k. We add a global counter for the total number of pertinent messages, Ttl, represented as $\lceil \log(N) \rceil$ ciphertexts of FHE containing the bit-wise binary representation of $k \in \mathbb{Z}_N$. The detector homomorphically sums all PV's into Ttl to let the recipient learn the number of pertinent messages, k.

We construct a compact OMD algorithm $\mathsf{OMDt2}$ using the techniques mentioned above. Pseudocode and analysis of $\mathsf{OMDt2}$ and the proof of the following theorem are deferred to the full version.

Theorem 2. *The scheme* $\mathsf{OMDt2}$ *is a compact Oblivious Message Detection scheme, when instantiated with any fully homomorphic encryption scheme* FHE *and a PRF* f*.*

This construction satisfies the asymptotic requirements of compact OMR (completeness, soundness, privacy, and compactness). Additional asymptotic optimizations are possible and deferred to the full version. The above is still far from concrete practicality (see end of Section 5), as will be addressed in Section 6.

5.2 Payload Retrieval using FHE

An OMD scheme, like the one above, lets the recipient learn the *indices* of the messages addressed to it in the board. It would then still need to retrieve the *content* (or *payload*) of those messages, and (in our motivating applications) do so privately without revealing which messages were of interest. As discussed in Section 2, retrieval with current methods *after* obtaining the indices is either inefficient or non-private.

We thus extend the above OMD scheme to Oblivious Message Retrieval, starting with a simple construction from generic FHE below, and improving it in Section 6. Our approach embeds techniques from PSS [51,21,9,25] and multi-query PIR [5,4] (see Section 2.2) into the detector's operation, without an extra round-trip to the client. Specifically, we use the encrypted 0/1 pertinency bits, derived above, to extract the pertinent payloads using homomorphic multiplication; we then compress these using homomorphically-evaluated linear codes.As discussed in Section 2.2, these techniques require substantial adaptations for efficiency in OMD/OMR. The generic OMR approach is detailed below, first as an inefficient construction based on generic FHE (as a didactic stepping stone), and then with major optimizations (Section 6).

Single Pertinent Message. Our starting point is the above $\mathsf{OMDt1}$ construction. The detector's procedure $\mathsf{OMR.Retrieve}$ first runs $\mathsf{PV} \leftarrow \mathsf{OMDt1.Detect}(D, \mathsf{pk}, \bar{k}, \epsilon_{\mathrm{n}})$, to obtain a pertinency vector PV of N ciphertexts, each encrypting 1 or 0. It then proceeds as follows.

Consider first the simple scenario where, throughout the board, there is a single message x_z that is pertinent for the recipient p (but z is initially unknown). Hence, PV has a single $\langle 1 \rangle_{\mathsf{sk}}$ ciphertext, and the rest are $\langle 0 \rangle_{\mathsf{sk}}$.

Let the payload space $\mathcal{P}$ be represented by a tuple of FHE's plaintext space (i.e., for $\mathcal{P} = \{0,1\}^{\tilde{n}}$ and the plaintext space $\mathbb{Z}_2$, we use $\tilde{n}$ ciphertexts to represent

each payload). In the following we generalize the $\mathsf{FHE.Enc}$, $\mathsf{FHE.Eval}$, and $\langle\cdot\rangle_{\mathsf{sk}}$ notation to work on such tuples in the natural way, as vector operations over $\mathbb{Z}_2$. For each $i \in [N]$, multiply x_i by PV_i homomorphically to get a ciphertext tuple x'_i. Thus, $x'_z \in \langle x_z \rangle_{\mathsf{sk}}$, and $x'_i \in \langle 0 \rangle_{\mathsf{sk}}$ for $i \in [N]$ for the rest of the ciphertext tuples ($i \neq z$). Then, homomorphically sum up all x'_i to get a ciphertext tuple M', so that $M' \in \langle x_z \rangle_{\mathsf{sk}}$. The detector sends this M' to the recipient, who decrypts it to obtain the payload x_z.

Multiple Pertinent Messages via Random Linear Coding. We generalize the above to the case of k pertinent messages, where $0 \leq k \leq \bar{k}$. The cap $\bar{k}$ is chosen by the recipient and given to the detector, but does not need to be known (or even fixed) when the board messages are generated.

The above single-message scheme fails if there are multiple pertinent messages, because the detector would output the encrypted *sum* of the k pertinent payloads, from which the individual payloads cannot be (in general) recovered. However, we can have the detector compute several encrypted combinations of the pertinent plaintexts, each one summing them with different weights, in hope of creating a linear system that the recipient can solve.

Specifically, we will choose some $m \geq \bar{k}$ and have the detector homomorphically compute encrypted payload m combinations $\mathsf{Cmb}_j \leftarrow \sum_{i\in[N]}(w_{i,j} \cdot x_i) \cdot \mathsf{PV}_i$ for $j \in [m]$, using random weights $w_{i,j}$ ($i \in [N]$, $j \in m$).[10] Letting $\mathsf{PS} \subseteq [N]$ denote the k pertinent message indices, we then have $\mathsf{Cmb}_j \in \left\langle \sum_{i\in\mathsf{PS}} w_{i,j} \cdot x_i \right\rangle_{\mathsf{sk}}$.

The combinations are realized as FHE ciphertext tuples big enough to represent $\mathcal{P}$, with element-wise operations, i.e., as a vector space over the field $\mathrm{GF}(t)$ of the plaintext space (in our case, of binary plaintext space: $\tilde{n}$ binary ciphertexts, with bitwise AND and XOR).

The OMR digest includes the aforementioned output of $\mathsf{OMDt1.Detect}$, from which the recipient can recover PS. Moreover the recipient can know the weights $w_{i,j}$ (by including the seed used to pseudorandomly generate the weights in the digest). Then the recipient can decrypt the payload combinations and recover m equations over the variables $(x_i)_{i\in\mathsf{PS}}$, of the form $\sum_{i\in\mathsf{PS}} w_{i,j} x_i = \mathsf{FHE.Dec}(\mathsf{Cmb}_j) \in \left\langle \sum_{i\in\mathsf{PS}} w_{i,j} \cdot x_i \right\rangle_{\mathsf{sk}}$ for $j \in [\bar{k}]$. If $|\mathsf{PS}| = k$ of these equations are linearly independent, then the recipient can use Gaussian elimination to recover these x_i, i.e., the pertinent messages' payloads.

For randomly-chosen weights in the field $\mathrm{GF}(t)$ (in our case $t = 2$), the probability of getting $\bar{k}$ linearly independent equations is $\prod_{i=m-\bar{k}+1}^{m}(1 - 1/t^i)$, via [58, Lemma 1]. Therefore, $m = \bar{k} + \lceil \log_t(\frac{1}{\epsilon_{\mathrm{n}}}) \rceil$ suffices.

5.3 Improved Retrieval Using SRLC

In the above attempt, the detector's computational cost for preparing the payload combinations is proportional to $\bar{k} \cdot N$. To reduce this cost, we use *Sparse*

[10] Here multiplication is in the field $\mathrm{GF}(2)$, for the plaintext space $\mathbb{Z}_2$, so the weights are just 0 or 1. In general, this works over $\mathrm{GF}(t)$ for prime t. From this point on we will require both multiplications and addition (to perform linear algebra), and thus consider the field $\mathrm{GF}(t)$ instead of the group $\mathbb{Z}_t$.

Random Linear Coding (SRLC) [34], encoded homomorphically. SRLC is widely studied and applied to data transmission [34,35,15,60].[11]

In this approach, we create weighted linear combinations as above, but we make them sparse: each message is assigned to just a few combinations (i.e., most of the weights $w_{i,j}$ are chosen as zero). We provide the following two schemes for SRLC to ensure that the resulting combinations are still linearly independent (thus solvable by the recipient) with high probability.

SRLC1: Analytically-Bounded SRLC. For SRLC1, each weight is chosen as nonzero with probability γ/m, where γ is some SRLC parameter and m is the number of combinations. To ensure that the resulting combinations are solvable for $k < \bar{k}$ variables with probability greater than $1 - \epsilon_{\mathrm{n}}$, provably $\gamma = O(\bar{k} \log^2 \bar{k} \log(\epsilon_{\mathrm{n}}^{-1}))$ and $m = O(\gamma \cdot \bar{k})$ suffice.

SRLC2: Empirically-Bounded SRLC. In this scheme, we uniformly choose exactly γ nonzero weights among the m combinations. The choice of γ and m is done by an empirical estimation procedure (in the absence of tight analytical bounds), and we prove that the estimation error is negligible.

Compact OMR using SRLC. We can construct a compact OMR scheme using SRLC, following the intuition of Sections 5.2 and 5.3. The resulting algorithm OMRt1 pseudocode is defered to the full version. The following theorem is proven, and complexity analyzed, in the full version:

Theorem 3. *The scheme* OMRt1 *is a Oblivious Message Retrieval scheme, when instantiated with any fully homomorphic encryption scheme* FHE*, PRFs* f*, and SRLC scheme* SRLC*. Moreover, when instantiated with* SRLC1*,* OMRt1 *is also compact.*

(Im)practicality. The above establishes the asymptotic existence of compact OMR (assuming existence of FHE), but it is still impractical, e.g., due to the cost of the Recrypt algorithm in state-of-the-art FHE schemes [22,16,13,24,14]. Moreover, the clues are large (e.g, FHEW/TFHE needs 512 bytes per each of the $\sim$20 plaintext bits required in Section 5.1).

6 Practical OMR

We proceed to introduce optimizations that improve communication and computation costs to practical levels, as well as security improvements. See Section 9 below for implementation and quantitative evaluation.

Our starting point is the generic FHE construction of Section 5, generalized from plaintext space $\mathbb{Z}_2$ to $\mathbb{Z}_t$ for prime $t \geq 2$ (as discussed in footnotes whenever pertinent). The generalization immediately gives us improved concrete bounds on false positive rate, and in the SRLC-based retrieval. We then proceed to modify the scheme as follows.

[11] State-of-the-art batch-PIR [5,4] uses different coding techniques, which rely on the client knowing the pertinent indices a priori.

6.1 PVW Clue Ciphertext

Instead of generating clues using encryption under an FHE scheme, we use a lighter-weight encryption scheme which can still be homomorphically decrypted and processed by the detector. Specifically, we use the PVW scheme [54], a variant of the Regev05 LWE-based encryption scheme [57]. See Section 4.1 for details. PVW decryption can be cheaply evaluated under FHE (we discuss in more detail later), and moreover its ciphertext size grows slowly when multiple bits are encrypted (to encrypt ℓ bits, clue size using PVW grows with $O(n+\ell)$, compared to $O(n \cdot \ell)$ in the original Regev05 scheme [57], where n is a predetermined LWE parameter to be discussed below). Choice of PVW parametersdeferred to the full version. The switch to PVW maintains the OMD/OMR privacy requirement, since PVW is $\mathsf{IND\text{–}CPA}$ and key-private by [28] and [54, Lemma 7.4].

6.2 BFV Leveled Homomorphic Encryption

In the detection and retrieval algorithm, we also replace FHE with leveled HE, i.e., homomorphic encryption restricted to evaluation of arithmetic circuits with predetermined multiplicative depth. This suffices since, as shown below, the multiplicative depth can be kept low and moreover, after the switch to PVW encryption, we do not need $\mathsf{Recrypt}$. Specifically, we use the BFV [13,24] scheme (see Section 4.2).

The detection key now contains the BFV public key, and the PVW secret key $\mathsf{PVW.sk}$ encrypted under BFV public key. The detector uses these to homomorphically decrypt the clue, resulting in PV_i which are $\langle 0 \rangle^{\mathsf{BFV}}_{\mathsf{sk}}$ or $\langle 1 \rangle^{\mathsf{BFV}}_{\mathsf{sk}}$. It then proceed to process these into a digest as in Section 5, using BFV homomorphic operations, all with plaintext space $\mathrm{GF}(t)$.

One advantage of this choice is that BFV supports "single instruction, multiple data" (SIMD) operations: each BFV ciphertext has D *slots*, each of which can convey an element of $\mathbb{Z}_t$, for some plaintext modulus t and a parameter D. When computing PV, we can thus operate on D separate messages $\{(x_i,c_i)\}$ in parallel via SIMD (see Section 4.2).

Specifically, the detector performs the following steps to homomorphically perform $\mathsf{PVW.Dec}$ (described intuitively here, and precisely in the full version). Note that for simplicity when decrypting homomorphically using BFV, we redefine our clues to be PVW encryptions of 0 instead of 1 as above (i.e. $\mathsf{PVW.Dec}(\mathsf{sk}_{p'}, \mathsf{GenClue}(\mathsf{pk}_p, \cdot)) = 0$ iff $p = p'$ w.h.p.).

- **Inner Product ($\mathsf{InnerProd}$).** The detector performs the first step of $\mathsf{PVW.Dec}$: inner product of the clue's PVW ciphertext with $\mathsf{PVW.sk}$ (that is provided by the recipient under BFV encryption).
- **Range checking ($\mathsf{RangeCheck}$)** For a range $[-r, r]$ and plaintext element $u \in \mathbb{Z}_t$, this maps u to 0 if $u \in [-r, r]$, otherwise to 1. We implement this homomorphically using [33, Equation 2] as follows, using the Paterson-Stockmeyer algorithm [53] to minimize multiplicative depth. To evaluate the function $f(x)$ where $f_r(x) = 0$ for $t - r \leq x \leq r$, and $f(x) = 1$ otherwise, we can evaluate

the following polynomial over $\mathbb{Z}_t$:

$$\mathsf{RC}_r(X) = f_r(0) - \sum_{i=0}^{t-1} X^i \sum_{a=0}^{t-1} f_r(a) a^{t-1-i} \quad . \tag{1}$$

We thus calculate $u' \leftarrow \mathsf{RC}_r(u)$ homomorphically, so $u' \in \langle 1 \rangle_{\mathsf{sk}}^{\mathsf{BFV}}$ iff $u \in [-r, r]$ (which is the case for pertinent messages with high probability). An exact implementation of PVW.Dec would require $r = t/4$ (by definition of PVW in Section 4.1), but this can be relaxed to reduce ϵ_{p}, since the error distribution is Gaussian.

– PV **Unpacking (PVUnpack).** Because we used SIMD evaluation, the above steps result in a single ciphertext ct whose D slots encode $u' \in \{0,1\}$ values of D different messages. This already suffices as a digest, but to maintain a clean interface and allow further improvements below, we proceed to unpack ct's slots into separate D ciphertexts $(\mathsf{PV}_i)_{i \in [D]}$.

Between RangeCheck and PVUnpack, we flip the sign of u' ($u' \leftarrow 1 - u'$) so that after all these steps, we get PV_i ciphertexts encrypting 1 in all slots for the pertinent messages, and 0 in all slots for impertinent messages with some false positive rate. Similarly to Section 5, we proceed as follows.

Soundness Amplification. Analogously to in Section 5.1, we reduce this false positive rate by having the sender encrypt ℓ 0's to the recipient's clue key (in this case: all in a single ℓ-bit PVW ciphertext). Note that here, the probability of a clue ciphertext decrypting to 0 (so that the range-checked output is $\mathsf{PV} \in \langle 1 \rangle_{\mathsf{sk}}^{\mathsf{BFV}}$) for impertinent messages is $p = 1 - (2r+1)/q \geq 1/2$, rather than $1/2$ as in the $\mathbb{Z}_2$ case. Therefore, using ℓ ciphertexts, we get ϵ_{p} of p^ℓ.

Applying all of the above, we obtain and OMD scheme analogous to OMDt2, but based on PVW clues and BFV scheme instead of generic FHE, thereby improving efficiency and size. The same technique applies toOMDt1, for more efficient *compact* detection.

Finally, compress the detection digest by one of two means.

Deterministic Digest Compression. Each BFV ciphertext can pack D elements of $\mathbb{Z}_t$, each of which can represent $\lfloor \log(t) \rfloor$ bits of plaintext information. We pack the single-bit pertinency indicators into these bits, homomorphically. This makes near-optimal use of the plaintext space; and in terms of digest size, utilization remains high: roughly **4.5 bit/msg**, for the representative parameters of Section 9. The recipient can sum up all these bits to get the exact number of pertinent messages, and thus robustly detects overflow.

Randomized Digest Compression. Alternatively, we can use the bucket-based method in Section 5.1 to achieve an compact digest, and compress D accumulators into one ciphertext to fully utilize all D slots in each ciphertext. In particular, we can achieve an amortized digest size of **less than 1 bit/msg**, including retrieval, for sufficiently large $N \gg \bar{k}$. See performance in Section 9.

For this methodwe could also use the summation of individual bucket counters to get k, but this may overflow if the number of pertinent messages is

huge, since each slot is computed homomorphically in $\mathbb{Z}_t$ (e.g. if $k \gtrsim tD$, where $tD > 10^{12}$ for typical parameters).[12]

OMD via Deterministic Digest Compression. We can use deterministic digest compression to construct a non-compact OMD, which we call OMDp1. The advantage of this method is that it does not need PVUnpack, and therefore greatly reduces the detector running time.

6.3 A Practical OMR scheme

Fig. 1 portrays the high-level components of the resulting scheme, and their invocation of different encryption schemes.

By combining all of the above detection optimizations, and adding message *retrieval* analogously to Section 5.3, we have a practical OMR scheme OMRp1. Here we use the aforementioned *Deterministic Digest Compression*, which is simpler than compact (randomized) detection, and offers better concrete digest size and detection time for some parameter choices (cf. Section 9). The resulting pseudocode is deferred to the full version. The following theorem is proven, and complexity analyzed, deferred to the full version:

Theorem 4. *The scheme* OMRp1 *is an* OMR *scheme, assuming security of* PVW *encryption (Section 4.2), security of* BFV *leveled HE (Section 4.2), when instantiated with PRF f and an SRLC scheme* SRLC*.*

6.4 A Practical Compact OMR Scheme

While OMRp1 above is practically efficient for many parameters of interest (cf. Section 9), its asymptotic digest size is still $O(N)$. An alternative approach achieves compactness, i.e., a digest size that grows only mildly with N when $\bar{k}$ is fixed and ϵ_{p} is small (cf. Definition 1). This can be achieved by using the *Randomized Digest Compression* approach of Section 6.2. The resulting practical and compact OMR algorithm, OMRp2, and the pseudocode is deferred to the full version. The following theorem is proven, and complexity analyzed, details deferred to the full version:

Theorem 5. *The scheme* OMRp2 *in* OMRp2 *is a* OMR *scheme for $N < D \cdot t/2$, assuming security of* LWE *encryption (Section 4.2) and security of* BFV *leveled HE (Section 4.2), when instantiated with PRF f and an SRLC scheme* SRLC*. Moreover when instantiated with* SRLC1*,* OMRp2 *is also compact.*

6.5 Additional Properties

Additional modifications improve the practicality of our algorithm. The following points are salient; further discussion is deferred to the full version.

[12] If such parameters are exceeded, we can avoid undetected decoding failures using a global counter that represents values in $[N]$ without overflow.

Streaming Updates. The detector can break up $\mathsf{OMR.Retrieve}$ into several phases and some of the phases can be done on-the-fly without receiving $\bar{k}$ from the recipient. With this process, we can greatly reduce the latency during retrieval. We defer the details to the full version. Security requires seed secrecy. For details see the the paper's full version.

Handling Overflows. Retrieval may fail in case the number k of pertinent messages overflows the bound $\bar{k}$. The possibility is inherent, since information-theoretically, a compact digest for a given $\bar{k}$ cannot represent $k \gg \bar{k}$ messages. When an overflow occurs, the recipient can robustly *detect* it and sends another retrieval query to the detector, with a larger bound $\bar{k}$. To prevent the linkability of the two queries, the recipient can issue the second query from a fresh anonymized network connection, and using unlinkable keys (see Section 8).

Detection Key Size Reduction. The detection key includes the BFV ciphertexts $\mathsf{ct}_{\mathsf{pvwSK}}$ encrypting $\mathsf{sk}_{\mathsf{pvw}}$, and the BFV public keys (including encryption key, relinearization key, and rotation keys as in [37, §5.6]). Their size is $O(1)$, but concretely quite large (cf. Section 9). We reduce it as follows.

First, all of the aforementioned components are RLWE ciphertexts of the form $(\vec{a}, \vec{b})$, generated by the recipient who knows the corresponding RLWE secret key, and who can thus choose a pseudorandom $\vec{a}$ that is represented as a short PRG seed, thereby halving the ciphertext size. Second, we pack the $n \cdot \ell$ elements of $\mathsf{ct}_{\mathsf{pvwSK}}$ into just ℓ BFV ciphertexts, and involve homomorphic rotation to compute the inner products. Third, the detection key size is dominated by the rotation keys in used for the homomorphic evaluation of slot rotations, and most of them can be generated for a low multiplicative level.

7 Denial-of-Service Resistance

Thus far we have assumed, as in prior works, that all clues in the board are generated honestly by the prescribed $\mathsf{GenClue}$ algorithm, using honestly-generated clue keys. However, clues may be generated maliciously, especially if anyone is allowed to add messages to the board in reality.

In a *Denial of Service (DoS) attack* on an OMR or OMD scheme, the adversary can maliciously generate any of the clues in board messages, in an attempt to induce false positives or false negatives in the subsequent detection/retrieval. The adversary could simply create pertinent messages for some recipient, thus trying to induce an overflow for that recipient (by exceeding their $\bar{k}$); this is inevitable and handled in Section 6.5. But the bigger danger is *amplified DoS*: even a single maliciously-crafted clue could cause catastrophic failure (e.g., causing false negative or positives for *many* recipients). Furthermore, the adversary may also take the role of a recipient in the system, and publish a maliciously-crafted clue key, thereby inducing honest senders to unwittingly generate harmful clues.

Attacks. $\mathsf{OMRt1}$, instantiated with FHEW/TFHE, is susceptible to a *wildcard ciphertext* attack, where a malicious sender populates the clue with ciphertexts that decrypt to 1 (hence detected as pertinent) for most recipients, causing

overflows. FMD [7] is also vulnerable to wildcard ciphertexts, as also observed by [40, commit `e19b99112e`]. PS [44] is likewise vulnerable to wildcard ciphertexts (in the single-server version). Moreover, in the PS model, signals are sent to servers rather than placed on the board, opens an additional DoS attack vector. For instance, a malicious sender can send many pertinent-looking signals to a server to overflow the message accumulator for recipients.

7.1 Modeling DoS Resistance

DoS Threat Model. In the DoS threat model, the computationally bounded adversary has all the power as defined in Section 3. Additionally, it is allowed to generate any (perhaps malformed) clues and post them on the board, as well as generate any (perhaps malformed) clue keys for other senders to use. Thus, for correctness and soundness, we assume only that the detector is honest but curious. Other parties are malicious. As before, for privacy, everyone but the message's sender and recipient are assumed malicious and colluding.

DoS Resistance Definition. We introduce a formal definition of DoS resistance, strengthening the OMR definition of Section 3.1. Recall that Definition 1assumes that the clues in the board are honestly generated, using honestly generated clue keys. In that case, there is a natural ground-truth notion of pertinent messages, defined by which clue key $\mathsf{pk}_{\mathsf{clue}}$ each clue was generated for; hence soundness and completeness are defined in reference to that ground truth.

Now, however, clues may be maliciously generated and may not obviously correspond to any specific clue key (e.g., consider the wildcard ciphertext above). We thus require the existence of an *indicator* predicate $\mathcal{I}(c,\mathsf{pk}_{\mathsf{clue}})$ that serves as a ground truth for whether a given clue c is pertinent to a given user specified by their clue key $\mathsf{pk}_{\mathsf{clue}}$. This predicate should give the natural answer for honestly-generated clues and arbitrary but determined answer for otherwise-generated clues (i.e., not claiming more than one honest recipient, except with small probability) and we call this property *collision resistance*. The stronger completeness and soundness are defined w.r.t such an indicator.

Soundness and completeness are then redefined w.r.t the indicator $\mathcal{I}$, as below. Note that to facilitate tight analysis, the completeness (false negative rate) bound ϵ_{n} in the definition is broken up into two components: the rate ϵ_{i} at which the indicator fails to detect truly pertinent messages (which may be non-negligible because a indicator with high thresholds may help achieve collision resistance), and the rate $\epsilon_{\mathrm{n}} - \epsilon_{\mathrm{i}}$ at which the scheme fails to retrieve messages flagged by the indicator (which may be on-negligible because of error sources in the concrete scheme).

Definition 4 (DoS-resistant OMR). *Let* OMR *be an OMR scheme for error rates* $\epsilon_{\mathrm{n}}, \epsilon_{\mathrm{p}}$ *(as in Definition 1). An* indicator *with an* indicator false negative rate $\epsilon_{\mathrm{i}} \leq \epsilon_{\mathrm{n}}$ *for* OMR *is a function* $b \leftarrow \mathcal{I}(\mathsf{pp}, x, c, \mathsf{pk}_{\mathsf{clue}}, \mathsf{sk})$ *on a public parameter* pp*, a message* (x, c)*, a clue key* $\mathsf{pk}_{\mathsf{clue}}$*, and its corresponding secret key* sk*, outputs* $b \in \{0,1\}$*, such that:*

– (Indicator completeness) *For* $\mathsf{pp} \leftarrow \mathsf{GenParams}(1^\lambda, \epsilon_\mathrm{p}, \epsilon_\mathrm{n})$*, honest-generated key pair* $(\mathsf{sk}, \mathsf{pk} = (\mathsf{pk}_{\mathsf{clue}}, \cdot)) \leftarrow \mathsf{KeyGen}()$*, for any payload* x*, and honest-generated clue* $c \leftarrow \mathsf{OMR.GenClue}(\mathsf{pk}_{\mathsf{clue}}, x)$*, it holds that:*

$$\Pr[\mathcal{I}(\mathsf{pp}, x, c, \mathsf{pk}_{\mathsf{clue}}, \mathsf{sk}) = 1] \geq 1 - \epsilon_\mathrm{i} - \mathsf{negl}(\lambda) \ .$$

– (Collision resistance) *For any PPT adversary* $\mathcal{A}$*, let* $\mathsf{pp} \leftarrow \mathsf{GenParams}(1^\lambda, \epsilon_\mathrm{p}, \epsilon_\mathrm{n})$*, two honest-generated key pairs* $(\mathsf{sk}, \mathsf{pk} = (\mathsf{pk}_{\mathsf{clue}}, \cdot)) \leftarrow \mathsf{OMR.KeyGen}()$ *and* $(\mathsf{sk}', \mathsf{pk}' = (\mathsf{pk}'_{\mathsf{clue}}, \cdot)) \leftarrow \mathsf{OMR.KeyGen}()$*, and adversarially-generated* $(x,c) \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{pk}')$*, for* $b \leftarrow \mathcal{I}(\mathsf{pp}, x, c, \mathsf{pk}_{\mathsf{clue}}, \mathsf{sk})$ *and* $b' \leftarrow \mathcal{I}(\mathsf{pp}, x, c, \mathsf{pk}'_{\mathsf{clue}}, \mathsf{sk}')$*:*

$$\Pr[b = 1 \ \wedge \ b' = 1] \leq \epsilon_\mathrm{p} + \mathsf{negl}(\lambda) \ .$$

An OMR scheme OMR *is* DoS-resistant *for* ϵ_n *and* ϵ_p *if there exists an indicator* $\mathcal{I}$ *with an indicator false negative rate* ϵ_i *for* OMR *such that for any PPT adversary* $\mathcal{A}$*, for* $\mathsf{pp} \leftarrow \mathsf{GenParams}(1^\lambda, \epsilon_\mathrm{p}, \epsilon_\mathrm{n})$*,* $(\mathsf{sk}, \mathsf{pk} = (\mathsf{pk}_{\mathsf{clue}}, \mathsf{pk}_{\mathsf{detect}})) \leftarrow \mathsf{OMR.KeyGen}()$*, and adversarially-generated board* $\mathsf{BB} \leftarrow \mathcal{A}(\mathsf{pp},\mathsf{pk})$ *where* $\mathsf{BB} = ((x_1,c_1), \ldots, (x_N,c_N))$ *and* $(x_i)_{i\in[N]}$ *are unique, for any* $0 < \bar{k} \leq N$*, letting* $M \leftarrow \mathsf{Retrieve}(D, \mathsf{pk}_{\mathsf{detect}}, \bar{k})$*,* $\mathsf{PL} \leftarrow \mathsf{Decode}(M, \mathsf{sk})$*:*

– (DoS-completeness) *Let* $k = \sum_{i=0}^{N} \mathcal{I}(\mathsf{pp}, x_i, c_i, \mathsf{pk}_{\mathsf{clue}}, \mathsf{sk})$*. Then either* $k > \bar{k}$ *and* $\mathsf{PL} = \mathsf{overflow}$*, or* $\Pr[x_j \in \mathsf{PL} \mid \mathcal{I}(\mathsf{pp}, x, c, \mathsf{pk}_{\mathsf{clue}}, \mathsf{sk}) = 1] \geq 1 - (\epsilon_\mathrm{n} - \epsilon_\mathrm{i}) - \mathsf{negl}(\lambda)$ *for all* $j \in [N]$*.*

– (DoS-soundness) $\Pr[x_j \in \mathsf{PL} \mid \mathcal{I}(\mathsf{pp}, x, c, \mathsf{pk}_{\mathsf{clue}}, \mathsf{sk}) = 0] \leq \mathsf{negl}(\lambda)$ *for all* $j \in [N]$*.*

Note that DoS-completeness implies the (weaker) completeness of Definition 1 with the same false negative rate ϵ_n, and DoS-soundness implies the (weaker) soundness of Definition 1 with false positive rate $\epsilon_\mathrm{p} + \epsilon_\mathrm{n}$. Completeness is trivial, and soundness is given by the following lemma (proof deferred to the full version):

Lemma 1. *Any tuple of algorithms* OMR *that is* ϵ_p*-DoS-sound by Definition 4 is* $(\epsilon_\mathrm{p} + \epsilon_\mathrm{n})$*-sound by Definition 1.*

The proof of Lemma 1 is deferred to the full version.
The Oblivious Message *Detection* definition is analogously strengthened for DoS.

7.2 Attaining DoS-resistant OMR

The OMRp1 scheme of Section 6 already satisfies DoS resistance (for $\epsilon_\mathrm{p} = \mathsf{poly}(\lambda)$, with a minor change, under the natural computational assumption stated in Conjecture 1 below). Intuitively, this is because PVW encryption has the property that a ciphertext that decrypts to 0 can be generated by adding up columns of the public key, but if the ciphertext is generated in any *other* way (e.g., from a different public key, or "out of the blue"), then its decryption is close to uniformly random.

Patched OMRp1. The exception to the above intuition is trivial ciphertexts (i.e., ciphertext $(\vec{a} = 0^n, \vec{b} \in \mathbb{Z}_q^\ell)$), so we redefine the clue space as $\{(\vec{a},\vec{b}) \in \mathbb{Z}_q^{n+\ell} : \vec{a} \neq 0^n\}$. Accordingly, we change $\mathsf{OMRp1.Retrieve}$ to reject clues where $\vec{a} = 0^n$ and $\mathsf{OMRp1.GenClue}$ such that if generates a clue with $\vec{a} = 0^n$, it retries with fresh randomness (and aborts after λ attempts).

Using the patch above, our $\mathsf{OMRp1}$ scheme is DoS resistance under a natural conjecture about LWE-based encryption.

Theorem 6. *For any $\epsilon_{\mathrm{p}} = \mathsf{poly}(\lambda)$, $\mathsf{OMRp1}$ (patched as above) is a DoS-resistant Oblivious Message Retrieval scheme, when instantiated with any PRF f, assuming the hardness of Ring-LWE and Conjecture 1 below.*

To prove our constructions fulfill this stronger definition, we first define an indicator in a natural way: given the secret key sk and a clue $c = (\vec{u}_a, \vec{u}_b) \in \mathcal{C}$, the indicator computes $\vec{u} = \vec{u}_b - \mathsf{sk}^T \vec{u}_a$ using the secret key and outputs 1 iff the result in range $[-r, r]$. This gives use completeness directly. The delicate part is to prove the indicator collision-resistance property. Intuitively, no matter what strategy is adopted by a PPT adversary, given two public keys it should be hard to generate a "snake-eye" ciphertext that decrypts to 0 under both keys, except with trivial probability (e.g., encrypting 0 to the first key and hoping to succeed for the second). The requisite property is formalized in the following definition[13] and conjecture:

Definition 5 (snake-eye-resistant encryption). *An encryption scheme* (KeyGen, Enc,Dec) *is* δ-snake-eye-resistant *if the following holds: for any PPT algorithm $\mathcal{A}$, for keys $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $(\mathsf{sk}', \mathsf{pk}') \leftarrow \mathsf{KeyGen}(1^\lambda)$, for ciphertext $c \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{pk}')$, it holds that* $\Pr[\mathsf{Dec}(sk,c) = 0 \ \wedge \ \mathsf{Dec}(sk',c) = 0] \leq \delta + \mathsf{negl}(\lambda)$.

Conjecture 1 (Patched Regev05 is snake-eye-resistant). The Regev05 scheme [57], patched such that decryption rejects any ciphertext $(\vec{a}, b)$ where $\vec{a} = \vec{0}$ and the decryption checks the result in range $[-r, r]$ as introduced in Section 6.2, is $\frac{2r+1}{t}$-snake-eye-resistant.

Snake-eye resistance is not generically implied by semantic security of the encryption scheme, nor by key privacy. However, Conjecture 1 can be proven under the standard (homogeneous) *Short Integer Solution (SIS)* hardness assumption [1] combined with a natural generalization of the *Knowledge of Knapsack of Noisy Inner Products* assumption [10]. We defer the high level explanation of this statement to the full version. Alternatively the latter assumption can be replaced by a zk-SNARK proof [10], appended to the ciphertext, of the statement "$(\vec{a},b)$ was constructed as a linear combination of public-key vectors".[14]

Conjecture 1 implies a natural generalization to snake-eye resistance of PVW encryption, which implies indicator collision resistance, and thus Theorem 6 (see the full version for proofs).

[13] We note that for encryption schems like El Gamal, the snake-eye-resistance is trivial, as for a ciphertext to be decrypted to the same plaintext, the secret keys must be the same as the decryption funciton is a one-to-one function.

[14] Such a proof fits in 192 bytes [30, §5.4.9.2] per clue regardless of ℓ, and in Zcash it can be merged into pre-existing zk-SNARK proofs in the same transaction [30].

8 Key Unlinkability

The security notions discussed thus far, and in prior works, omit another privacy consideration: linkability of the detection and clue keys. This occurs in several senses. *Detection-key to clue-key linkability* would reveal clients' identities to the detector serving them. *Detection-key to detection-key linkability* would allow detectors to link recurring clients, facilitating traffic analysis of detection queries. *Clue-key to clue-key linkability* is a concern when recipients wish to publish several addresses that are unlinkable, but (secretly) correspond to a single secret key which controls all of them (as in Zcash's "diversified address" [30]).

Linkability in Prior Work. In FMD [7], the detection keys and clue keys are a key pair in an asymmetric encryption scheme and therefore can be linked. Both PS schemes [44] link the detection key to the clue key by having a public identifying number R_i for each recipient and used in each retrieval. In both FMD and PS, the detection key and clue key are fixed for each recipient.

Defining Unlinkability. We define, in the natural way, the notion of OMR (or OMD) that is *detection-to-clue-key-unlinkable*. We also define OMR (or OMD) that is *full-key-unlinkable*: it offers algorithms $\mathsf{pk}^*_{\mathsf{detect}} \leftarrow \mathsf{RegenDetectKey}(\mathsf{sk})$ and $\mathsf{pk}^*_{\mathsf{clue}} \leftarrow \mathsf{RegenClueKey}(\mathsf{sk})$ that generate new public keys such that cannot be linked to other ones for the same sk. See the full version for details.

Key Unlinkability in OMRp1 and OMRp2. The OMRp1 and OMRp2 schemes are detection-to-clue-key-unlinkable, by the semantic security of BFV (applied to $\mathsf{ct}_{\mathsf{lweSK}}$ in $\mathsf{pk}_{\mathsf{detect}}$). Moreover, they are full-key-unlinkable, via a key re-generation process. To resample the detection key, we generate a fresh BFV key pair (which is trivially unlinkable to prior keys) and encrypt the PVW secret key to the new BFV key (this ciphertext is also unlinkable to prior keys, by semantic security of BFV). To resample clue keys, we regenerate the PVW public keys from the PVW private key; this is indeed unlinkable. We defer the details to the full version.

Attaining Key Unlinkability for Other Schemes. Our OMRt1 scheme can be modified to achieve full-key-unlinkability similarly. We defer details to the full version. PS1 can be analogously modified to achieve detection-to-clue-key unlinkability by changing its PKE and some other modifications (deferred to the full version). It is not obvious how to achieve key-unlinkablity, for PS2, FMD1, or FMD2.

9 Performance Evaluation

We implemented the OMRp1 scheme of Section 6.3, and the OMRp2 scheme of Section 6.4 instantiated with SRLC2, in a C++ library (released as open source). We used the PALISADE library [52] for PVW encryption, and the SEAL library [47] with Intel-HEXL acceleration [12] for the BFV scheme. We benchmarked these schemes on several parameter settings, on a Google Compute Cloud `c2-standard-4` instance type (4 hyperthreads of an Intel Xeon 3.10 GHz CPU with 16GB RAM). This section reports and compares the results.

		Detection schemes				**Retrieval schemes** (including detection)				
		ZIP-307 [27,23]	PS1 [44]	PS2 [44]	OMDp1 §6.2	Zcash full scan [23]	FMD1 [7] / [40]	FMD2 [7]	OMRp1 §6.3	OMRp2 §6.4
Communication (bytes/msg)		116	$\ll 1$	$\ll 1$ + 3M s↔s	0.56	612	42	5.3	1.13	9.03
Detector computation time (sec/msg)	1 thread 2/4	N/A	0.06	0.25	0.021 0.01/0.0099	N/A	0.011 / 0.00020	0.043	0.145 0.075/0.065	0.155 0.085/0.072
Recipient computation total time (sec)	1 thread	70	$\ll 10^{-3}$	$\ll 10^{-3}$	0.005	61	2.1	0.29	0.02	0.063
Clue size (bytes)		N/A	32	32	956	N/A	68 / 64.5	318,530	956	956
Clue key size (bytes)		N/A	32	N/A	133 k	N/A	1.5 k	1 k	133 k	133 k
Detection key size (bytes)		N/A	64	920	99 M	N/A	768	512	129 M	129 M
Retrieval privacy		Full	Full	Partitioned across detectors	Full	Full	pN-msg-anonymity $p = 2^{-5}$	pN-msg-anonymity $p = 2^{-8}$	Full	Full
Env. assumptions for privacy		None	TEE (SGX)	Non-colluding servers	None	None	None	None	None	None
Env. assumptions for Soundness+completeness		None	Honest S&R	Honest S&R	None	None	Honest S&R	Honest S&R	None	None

Table 2: Comparison of cost metrics, functionality and security attributes. Costs are per message, per recipient. Notation is as in Table 1. The bulletin contains $N = 500{,}000$ messages, of which $k = \bar{k} = 50$ are pertinent to the recipient. "s↔s" means server-to-server communication. For PS1/PS2, we used the times from [44, §9.2] (Intel Xeon Platinum 8259CL), and some costs are via private communication from their authors. If FMD1/FMD2 are used just for detection, the costs are essentially unchanged except that communication is ≤ 1.

Parameters. We set the total number of messages to $N = 500{,}000$ (roughly the number of Bitcoin payments per day), and set the cap on the number of pertinent message to $\bar{k} = 50$. We set a false positive rate $\epsilon_{\mathrm{p}} = 2^{-21}$ (including decryption failure) and a false negative rate $\epsilon_{\mathrm{n}} = 2^{-30}$. The payload size is 612 bytes, as in Zcash. The internal parameter choice for PVW, BFV and other settings in OMRp1/OMRp2 are deferred to the full version of this paper.

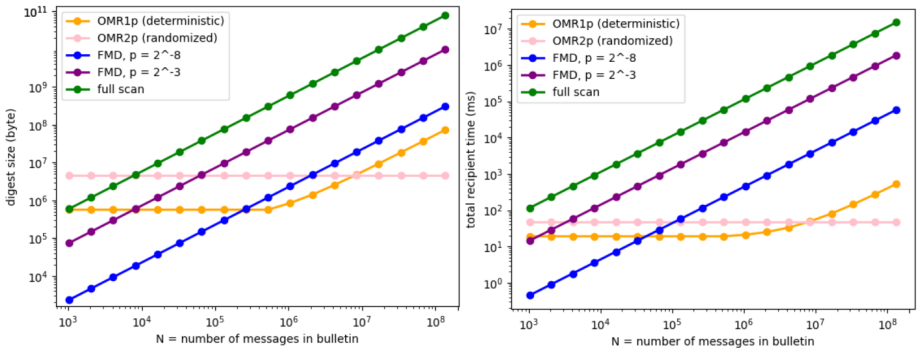
Representative Costs. Table 2 summarizes the main cost metrics and functionality/security attributes of our scheme, compared to related ones, for the above parameters. (See also Table 1 for additional functional/security attributes and the full version for asymptotic costs,).

We see that in both communication and recipient computation, OMRp1 is better than any other scheme with retrieval functionality, thereby making it attractive for recipients that are limited in bandwidth, computation speed, or energy. Furthermore, OMRp1/OMRp2 provide the strongest form of security, and under the least assumptions.

Retrieval Scaling with #Messages. Fig. 2 evaluates how the recipient's total cost of retrieval scales with increasing N, keeping $\bar{k}$ constant. As can be seen, our scheme OMRp1 outperforms all prior constructions starting at moderate bulletin sizes, in both digest size and recipient computation time.

For $N > 8 \cdot 10^6$, our compact OMR scheme OMRp2 takes the lead and achieves an amortized digest size of less than 1 bit per message. In general, the crossover point grows with k (due to the growing number of buckets in OMRp2), so OMRp2 outperforms when N is large but k is small.

The detector computation time in OMRp1/OMRp2 is worse than for FMD and full-scan. It is linear in N for all schemes, and thus follows from Table 2.



(a) Digest size vs. number of messages. (b) Total recipient computation time vs. number of messages.

Fig. 2: Retrieval cost comparison (total digest size and recipient computation) for up to $\bar{k} = 50$ pertinent messages.

Detection Key. The detection key size is 129 MB , using the techniques of Section 6.5. (Without these optimizations, the detection key size is ∼13.5 GB.) Note that the detection key can be sent to the detector via an insecure channel, authenticated by a hash. After the one-time cost of transmission, it can then be used to detect an arbitrary number of messages.

Memory Use. For simplicity, our implementation stores the detection key and all intermediate results in RAM, for a total memory use of ∼3.5 GB per thread.

Streaming Finalization Cost. Using the streaming approach of Section 6.5, we can reduce the response time to recipients For OMRp1, the finalization is only ∼ 2.1 ms/msg. It can be further reduced to ∼ 0.35 ms/msg , since we have fixed γ. Similar results hold for OMRp2.

10 Integration and Limitations

We proceed to discuss systems aspect of integrating OMR in real-world applications, and limitations of our current constructions. For concreteness, we consider integration of OMRp1 or OMRp2 with the Zcash cryptocurrency [30] to solve the problem of receiver metadata leakage [32] from its light wallet protocol [27]. This prospective integration illustrates several hurdles and how they can be resolved. We use the same scheme parameters as in Section 9.

Clue Key Distribution. In our OMR approach (as for FMD [7] and single-server PS [44]), senders need to obtain the recipient's clue key to generate clues. It is natural to consider the clue key to be an extension of the public address, shared by the same trusted channels. Zcash's Unified Addresses mechanism [31] indeed allows such data to be included with public addresses in a backwards-compatible way, and payment URIs [59,49] can be similarly extended. The clue

key size of 133 kB (PVW public key) has usability issues. Alternatively, the Unified Address can contain a URL from which the clue key can be fetched. Zcash diversified addresses [30] can be accompanied by different clue keys while preserving address unlinkability, using the full-key-unlinkability property.

Clue Embedding. Clues of size 956 bytes need to be attached to every payload. This is comparable to the roughly 1.3 kB of data already on-chain per such payment. The transaction format can be extended with a dedicated clue field, or clues can be embedded into `OP_RETURN` data or dummy output descriptions.

Detection Latency. Detectors needs to see all blockchain data, facilitating detection in several ways: in the *single-shot* model, the recipient makes a stateless query to the detector. Response latency is high: about 0.145 sec/msg (cf. Table 2). The *subscribe and finalize* model utilizes a streaming version of OMRp1, as in Section 6.5, taking 0.0005 core-sec/msg.

Detection Key. Our OMRp1 and OMRp2 schemes requires recipient to send large detection keys (∼129 MB) to detectors. Conversely, OMRt1 instantiated with TFHE reduces detection key size to ∼16 MB, but with much slower detection. Combining the best of these is an open problem.

Detection Cost. The detector cost is ∼$1.02 per million payments scanned (per recipient), using commodity cloud computing.[15] This implies $0.02/month detection cost for Zcash's current shielded payments usage, or $1.66/month for the current usage rate of Monero (the highest-volume privacy-enhanced cryptocurrency). If all of Bitcoin's payments were instead done as Zcash shielded payments, detection cost would grow to $15.3/month; and even higher for massive private messaging applications such as Signal. Acceleration via GPU, FPGA or ASIC can improve costs by orders of magnitudes [56,2]. For cryptocurrency applications, the recipient can directly pay the server for this anonymously.

Acknowledgements. We are grateful to Daniele Micciancio for suggesting suitable FHE schemes for Section 5; to Ran Canetti, Oded Regev and Noah Stephens-Davidowitz for observations on Conjecture 1; to Matthew Green, Jack Grigg, Daira Hopwood, Taylor Hornby and Madarz Virza for ideas and observations regarding Zcash integration in Section 10; to István András Seres and Varun Madathil for assistance in quantitative evaluation of [44] and comparisons to our work; to Miranda Christ for excellent editorial suggestions; and to Wei Dai for assistance in generating level-specific rotation keys using SEAL library.

This material is based upon work supported by DARPA under Contract No. HR001120C0085; the U.S. Department of Energy (DOE), Office of Science, Office of Advanced Scientific Computing Research under award number DE-SC-0001234, the Columbia-IBM center for Blockchain and Data Transparency; JPMorgan Chase & Co, and LexisNexis Risk Solutions. Any opinions, views, findings and conclusions or recommendations expressed in this material are those

[15] 0.065 sec/msg (4-core, `c2-standard-4` preemptible compute instance, $0.051/hr). For finalization, 0.18 ms/msg, 4-core (non-preemptible instance, $0.168/hour with sustained use discount).Communication cost is negligible: $<\$10^{-9}$/msg egress.

of the authors and do not necessarily reflect the views of the U.S. Government, DARPA, DOE, JPMorgan Chase & Co. or its affiliates, or other sponsors.

References

1. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: ACM Symposium on Theory of Computing. p. 99–108. STOC '96, ACM (1996)
2. Al Badawi, A., Polyakov, Y., Aung, K.M.M., Veeravalli, B., Rohloff, K.: Implementation and performance evaluation of RNS variants of the bfv homomorphic encryption scheme. IEEE Transactions on Emerging Topics in Computing (2021)
3. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Journal of Mathematical Cryptology pp. 169–203 (2015)
4. Ali, A., Lepoint, T., Patel, S., Raykova, M., Schoppmann, P., Seth, K., Yeo, K.: Communication–computation trade-offs in PIR. In: (USENIX Security 21). pp. 1811–1828. USENIX (Aug 2021)
5. Angel, S., Chen, H., Laine, K., Setty, S.T.V.: PIR with compressed queries and amortized query processing. In: 2018 IEEE S&P. IEEE Computer Society Press (2018)
6. Angel, S., Setty, S.: Unobservable communication over fully untrusted infrastructure. In: (OSDI 16). pp. 551–569. USENIX (Nov 2016)
7. Beck, G., Len, J., Miers, I., Green, M.: Fuzzy message detection. The ACM Conference on Computer and Communications Security (CCS) 2021 (2021)
8. Ben Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE S&P. pp. 459–474 (2014)
9. Bethencourt, J., Song, D.X., Waters, B.: New techniques for private stream searching. ACM Trans. Inf. Syst. Secur. 12, 16:1–16:32 (2009)
10. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In: ITCS'12. p. 326–349. ACM (2012)
11. Bittau, A., Erlingsson, Ú., Maniatis, P., Mironov, I., Raghunathan, A., Lie, D., Rudominer, M., Kode, U., Tinnes, J., Seefeld, B.: Prochlo: Strong privacy for analytics in the crowd. In: SOSP. pp. 441–459 (2017)
12. Boemer, F., Kim, S., Seifu, G., de Souza, F.D., Gopal, V., et al.: Intel HEXL (release 1.2). `https://github.com/intel/hexl` (Sep 2021)
13. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: CRYPTO 2012. LNCS, Springer (Aug 19–23, 2012)
14. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: ITCS 2012. ACM (Jan 8–10, 2012)
15. Brown, S., Johnson, O., Tassi, A.: Reliability of broadcast communications under sparse random linear network coding. IEEE Transactions on Vehicular Technology 67(5), 4677–4682 (2018)
16. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: Fast fully homomorphic encryption over the torus. Journal of Cryptology pp. 34–91 (Jan 2020)
17. Choi, S.G., Dachman-Soled, D., Gordon, S.D., Liu, L., Yerukhimovich, A.: Compressed oblivious encoding for homomorphically encrypted search. CCS'21 (2021)
18. Chor, B., Gilboa, N., Naor, M.: Private information retrieval by keywords (1998), `http://ia.cr/1998/003`, appeared in the Theory of Cryptography Library.

19. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. In: 36th FOCS. pp. 41–50. IEEE Computer Society Press (Oct 23–25, 1995)
20. Corrigan-Gibbs, H., Boneh, D., Mazières, D.: Riposte: An anonymous messaging system handling millions of users. In: 2015 IEEE S&P. pp. 321–338 (2015)
21. Danezis, G., Diaz, C.: Space-efficient private search with applications to rateless codes. In: FC'07. p. 148–162. Springer (2007)
22. Ducas, L., Micciancio, D.: FHEW: Bootstrapping homomorphic encryption in less than a second. In: EUROCRYPT 2015, Part I. pp. 617–640. LNCS, Springer, Heidelberg, Germany (Apr 26–30, 2015)
23. Electric Coin Company: Zcash Rust crates. `https://github.com/zcash/librustzcash`, commit hash: 99d877e22d58610dc43021b831a28286ef353a89
24. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144 (2012), `https://ia.cr/2012/144`
25. Finiasz, M., Ramchandran, K.: Private stream search at almost the same communication cost as a regular search. In: Knudsen, L.R., Wu, H. (eds.) Selected Areas in Cryptography. pp. 372–389. Springer, Berlin, Heidelberg (2013)
26. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: ACM Symposium on Theory of Computing. p. 169–178. STOC '09, ACM (2009)
27. Grigg, J., Hopwood, D.: Zcash improvement proposal 307: Light client protocol for payment detection. `https://zips.z.cash/zip-0307` (Sep 2018)
28. Halevi, S.: A sufficient condition for key-privacy. Cryptology ePrint Archive, Report 2005/005 (2005)
29. Jelle van den Hooff, J., Lazar, D., Zaharia, M., Zeldovich, N.: Vuvuzela: Scalable private messaging resistant to traffic analysis. In: SOSP. p. 137–152. ACM (2015)
30. Hopwood, D., Bowe, S., Hornby, T., Wilcox, N.: Zcash Protocol Specification Version 2021.2.14. `https://github.com/zcash/zips/blob/master/protocol/protocol.pdf`
31. Hopwood, D., Wilcox, N., Hornby, T., Grigg, J., Bowe, S., Nuttycombe, K., Lai, Y.T.: Zcash improvement proposal 316: Unified addresses and unified viewing keys. `https://zips.z.cash/zip-0316` (Apr 2021)
32. Hornby, T.: Fixing privacy problems in the Zcash light wallet protocol. `https://defuse.ca/downloads/Fixing%20Privacy%20Problems%20in%20the%20Zcash%20Light%20Wallet%20Protocol.pdf` (Oct 2020)
33. Iliashenko, I., Nègre, C., Zucca, V.: Integer functions suitable for homomorphic encryption over finite fields. Cryptology ePrint Archive, Report 2021/1335 (2021), WAHC 2021.
34. Kaufman, T., Sudan, M.: Sparse random linear codes are locally decodable and testable. In: FOCS'07 (2007)
35. Khan, A.S., Chatzigeorgiou, I.: Improved bounds on the decoding failure probability of network coding over multi-source multi-relay networks. IEEE Communications Letters 20(10), 2035–2038 (2016)
36. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: single database, computationally-private information retrieval. In: FOCS'97 (1997)
37. Laine, K.: Simple encrypted arithmetic library 2.3.1. `https://www.microsoft.com/en-us/research/uploads/prod/2017/11/sealmanual-2-3-1.pdf`, Microsoft Research, Redmond, WA.
38. Lazar, D., Zeldovich, N.: Alpenhorn: Bootstrapping secure communication without leaking metadata. In: OSDI 16. pp. 571–586. USENIX (Nov 2016)
39. Le, D., Tengana Hurtado, L., Ahmad, A., Minaei, M., Lee, B., Kate, A.: A tale of two trees: One writes, and other reads. PETS 2020 pp. 519–536 (04 2020)

40. Lewis, S.J.: fuzzytags, https://git.openprivacy.ca/openprivacy/fuzzytags.git
41. Lewis, S.J.: Discreet log #1: Anonymity, bandwidth and Fuzzytags (Feb 2021), https://openprivacy.ca/discreet-log/01-anonymity-bandwidth-and-fuzzytags/
42. Lund, J.: Technology preview: Sealed sender for signal. https://signal.org/blog/sealed-sender/ (Oct 2018)
43. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. J. ACM (2013)
44. Madathil, V., Scafuro, A., Seres, I.A., Shlomovits, O., Varlakov, D.: Private signaling. Cryptology ePrint Archive, Report 2021/853 (20210624:145011) (2021)
45. Martiny, I., Kaptchuk, G., Aviv, A., Roche, D., Wustrow, E.: Improving signal's sealed sender. In: NDSS 2022 (01 2021), nDSS 2022
46. Matetic, S., Wüst, K., Schneider, M., Kostiainen, K., Karame, G., Capkun, S.: BITE: Bitcoin lightweight client privacy using trusted execution. In: (USENIX Security 19). pp. 783–800. USENIX (Aug 2019)
47. Microsoft SEAL (release 3.6). https://github.com/Microsoft/SEAL (Nov 2020), Microsoft Research, Redmond, WA.
48. Noether, S.: Ring signature confidential transactions for monero. IACR Cryptology ePrint Archive 2015, 1098 (2015)
49. Nuttycombe, K., Hopwood, D.: Zcash improvement proposal 321: Payment request URIs. https://zips.z.cash/zip-0321 (Aug 2020)
50. Oblivious message retrieval implementation. https://github.com/ZeyuThomasLiu/ObliviousMessageRetrieval (Dec 2021)
51. Ostrovsky, R., Skeith, W.E.: Private searching on streaming data. In: CRYPTO (2005)
52. PALISADE lattice cryptography library (release 11.2). https://palisade-crypto.org/ (Jun 2021)
53. Paterson, M., Stockmeyer, L.: On the number of nonscalar multiplications necessary to evaluate polynomials. SIAM J. Comput. pp. 60–66 (03 1973)
54. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: CRYPTO 2008. pp. 554–571. Springer (2008)
55. Player, R.: Parameter selection in lattice-based cryptography. Ph.D. thesis, Royal Holloway, University of London (2018)
56. Reagen, B., Choi, W.S., Ko, Y., Lee, V.T., Lee, H.H.S., Wei, G.Y., Brooks, D.: Cheetah: Optimizing and accelerating homomorphic encryption for private inference. In: 2021 IEEE HPCA. pp. 26–39 (2021)
57. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM (Sep 2009)
58. Salmond, D., Grant, A.J., Grivell, I., Chan, T.: On the rank of random matrices over finite fields. CoRR (2014), http://arxiv.org/abs/1404.3250
59. Schneider, N., Corallo, M.: Bitcoin improvement proposal 21: URI scheme. https://github.com/bitcoin/bips/blob/master/bip-0021.mediawiki (Jan 2012)
60. Tassi, A., Chatzigeorgiou, I., Lucani, D.: Analysis and optimization of sparse random linear network coding for reliable multicast services. IEEE Transactions on Communications pp. 285–299 (01 2016)
61. Wolinsky, D.I., Corrigan-Gibbs, H., Ford, B., Johnson, A.: Dissent in numbers: Making strong anonymity scale. In: OSDI 12. pp. 179–182. USENIX (Oct 2012)
62. Wüst, K., Matetic, S., Schneider, M., Miers, I., Kostiainen, K., Capkun, S.: Zlite: Lightweight clients for shielded Zcash transactions using trusted execution. In: Financial Cryptography and Data Security 2019. LNCS, Springer (2019)