Adaptive Oblivious Transfer with Access Control from Lattice Assumptions

Benoît Libert[1,2], San Ling[3], Fabrice Mouhartem[2], Khoa Nguyen[3], and Huaxiong Wang[3]

[1] CNRS, Laboratoire LIP, France
[2] ENS de Lyon, Laboratoire LIP (U. Lyon, CNRS, ENSL, INRIA, UCBL), France
[3] School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore

Abstract Adaptive oblivious transfer (OT) is a protocol where a sender initially commits to a database $\{M_i\}_{i=1}^N$. Then, a receiver can query the sender up to k times with private indexes $\rho_1, \ldots, \rho_k$ so as to obtain $M_{\rho_1}, \ldots, M_{\rho_k}$ and nothing else. Moreover, for each $i \in [k]$, the receiver's choice ρ_i may depend on previously obtained messages $\{M_{\rho_j}\}_{j<i}$. Oblivious transfer with access control (OT-AC) is a flavor of adaptive OT where database records are protected by distinct access control policies that specify which credentials a receiver should obtain in order to access each M_i. So far, all known OT-AC protocols only support access policies made of conjunctions or rely on *ad hoc* assumptions in pairing-friendly groups (or both). In this paper, we provide an OT-AC protocol where access policies may consist of any branching program of polynomial length, which is sufficient to realize any access policy in $\mathsf{NC1}$. The security of our protocol is proved under the Learning-with-Errors (LWE) and Short-Integer-Solution (SIS) assumptions. As a result of independent interest, we provide protocols for proving the correct evaluation of a committed branching program on a committed input.

Keywords. Lattice assumptions, standard assumptions, zero-knowledge arguments, adaptive oblivious transfer.

1 Introduction

Oblivious transfer (OT) is a central cryptographic primitive coined by Rabin [49] and extended by Even *et al.* [21]. It involves a sender S with a database of messages $M_1, \ldots, M_N$ and a receiver R with an index $\rho \in \{1, \ldots, N\}$. The protocol allows R to retrieve the ρ-th entry M_ρ from S without letting S infer anything on R's choice ρ. Moreover, R only obtains M_ρ learns nothing about $\{M_i\}_{i \neq \rho}$.

In its adaptive flavor [44], OT allows the receiver to interact k times with S to retrieve $M_{\rho_1}, \ldots, M_{\rho_k}$ in such a way that, for each $i \in \{2, \ldots, k\}$, the i-th index ρ_i may depend on the messages $M_{\rho_1}, \ldots, M_{\rho_{i-1}}$ previously obtained by R.

OT is known to be a complete building block for cryptography (see, e.g., [25]) in that, if it can be realized, then any secure multiparty computation can be. In its adaptive variant, OT is motivated by applications in privacy-preserving

access to sensitive databases (e.g., medical records or financial data) stored in encrypted form on remote servers, oblivious searches or location-based services.

As far as efficiency goes, adaptive OT protocols should be designed in such a way that, after an inevitable initialization phase with linear communication complexity in N and the security parameter λ, the complexity of each transfer is at most poly-logarithmic in N. At the same time, this asymptotic efficiency should not come at the expense of sacrificing ideal security properties. The most efficient adaptive OT protocols that satisfy the latter criterion stem from the work of Camenisch, Neven and shelat [14] and its follow-ups [28,29,30].

In its basic form, (adaptive) OT does not restrict in any way the population of users who can obtain specific records. In many sensitive databases (e.g., DNA databases or patients' medical history), however, not all users should be able to download all records: it is vital access to certain entries be conditioned on the receiver holding suitable credentials delivered by authorities. At the same time, privacy protection mandates that authorized users be able to query database records while leaking as little as possible about their interests or activities. In medical datasets, for example, the specific entries retrieved by a given doctor could reveal which disease his patients are suffering from. In financial or patent datasets, the access pattern of a company could betray its investment strategy or the invention it is developing. In order to combine user-privacy and fine-grained database security, it is thus desirable to enrich adaptive OT protocols with refined access control mechanisms in many of their natural use cases.

This motivated Camenisch, Dubovitskaya and Neven [12] to introduce a variant named *oblivious transfer with access control* (OT-AC), where each database record is protected by a different access control policy $P : \{0,1\}^* \rightarrow \{0,1\}$. Based on their attributes, users can obtain credentials generated by pre-determined authorities, which entitle them to anonymously retrieve database records of which the access policy accepts their certified attributes: in other words, the user can only download the records for which he has a valid credential Cred_x for an attribute string $x \in \{0,1\}^*$ such that $P(x) = 1$. During the transfer phase, the user demonstrates possession of a pair (Cred_x, x) and simultaneously convinces the sender that he is querying some record M_ρ associated with a policy P such that $P(x) = 1$. The only information that the database holder eventually learns is that some user retrieved some record which he was authorized to obtain.

Camenisch *et al.* formalized the OT-AC primitive and provided a construction in groups with a bilinear map [12]. While efficient, their solution "only" supports access policies consisting of conjunctions: each policy P is specified by a list of attributes that a given user should obtain a credential for in order to complete the transfer. Several subsequent works [54,13,11] considered more expressive access policies while even hiding the access policies in some cases [13,11]. Unfortunately, all of them rely on non-standard assumptions (known as "q-type assumptions") in groups with a bilinear maps. For the sake of not putting all one's eggs in the same basket, a primitive as powerful as OT-AC ought to have alternative realizations based on firmer foundations.

In this paper, we propose a solution based on lattice assumptions where ac-

cess policies consist of any branching program of width 5, which is known [6] to suffice for the realization of any access policy in $\mathsf{NC1}$. As a result of independent interest, we provide protocols for proving the correct evaluation of a committed branching program. More precisely, we give zero-knowledge arguments for demonstrating possession of a secret input $\mathbf{x} \in \{0,1\}^\kappa$ and a secret (and possibly certified) branching program BP such that $\mathsf{BP}(\mathbf{x}) = 1$.

Related Work. Oblivious transfer with adaptive queries dates back to the work of Naor and Pinkas [44], which requires $O(\log N)$ interaction rounds per transfer. Naor and Pinkas [46] also gave generic constructions of (adaptive) k-out-of-N OT from private information retrieval (PIR) [17]. The constructions of [44,46], however, are only secure in the half-simulation model, where simulation-based security is only considered for one of the two parties (receiver security being formalized in terms of a game-based definition). Moreover, the constructions of Adaptive OT from PIR [46] require a complexity $O(N^{1/2})$ at each transfer where Adaptive OT allows for $O(\log N)$ cost. Before 2007, many OT protocols (e.g., [45,3,53]) were analyzed in terms of half-simulation.

While several efficient fully simulatable protocols appeared the last 15 years (e.g., [40,48] and references therein), full simulatability remained elusive in the adaptive k-out-of-N setting [44] until the work [14] of Camenisch, Neven and shelat, who introduced the "assisted decryption" paradigm. The latter consists in having the sender obliviously decrypt a re-randomized version of one of the original ciphertexts contained in the database. This technique served as a blueprint for many subsequent protocols [28,29,30,34], including those with access control [12,13,11,1] and those presented in this paper. In the adaptive k-out-of-N setting (which we denote as $\mathcal{OT}^N_{k\times 1}$), the difficulty is to achieve full simulatability without having to transmit a $O(N)$ bits at each transfer. To our knowledge, except the oblivious-PRF-based approach of Jarecki and Liu [34], all known fully simulatable $\mathcal{OT}^N_{k\times 1}$ protocols rely on bilinear maps. [4]

A number of works introduced various forms of access control in OT. Priced OT [3] assigns variable prices to all database records. In conditional OT [19], access to a record is made contingent on the user's secret satisfying some predicate. Restricted OT [31] explicitly protects each record with an independent access policy. Still, none of these OT flavors aims at protecting the anonymity of users. The model of Coull, Green and Hohenberger [18] does consider user anonymity via stateful credentials. For the applications of OT-AC, it would nevertheless require re-issuing user credentials at each transfer.

While efficient, the initial OT-AC protocol of Camenisch *et al.* [12] relies on non-standard assumptions in groups with a bilinear map and only realizes access policies made of conjunctions. Abe *et al.* [1] gave a different protocol which they proved secure under more standard assumptions in the universal composability framework [15]. Their policies, however, remain limited to conjunctions. It was mentioned in [12,1] that disjunctions and DNF formulas can be handled by du-

[4] Several pairing-free candidates were suggested in [36] but, as pointed out in [30], they cannot achieve full simulatability in the sense of [14]. In particular, the sender can detect if the receiver fetches the same record in two distinct transfers.

plicating database entries. Unfortunately, this approach rapidly becomes prohibitively expensive in the case of (t, n)-threshold policies with $t \approx n/2$. Moreover, securing the protocol against malicious senders requires them to prove that all duplicates encrypt the same message. More expressive policies were considered by Zhang *et al.* [54] who gave a construction based on attribute-based encryption [51] that extends to access policies expressed by any Boolean formulas (and thus $\mathsf{NC1}$ circuits). Camenisch, Dubovitskaya, Neven and Zaverucha [13] generalized the OT-AC functionality so as to hide the access policies. In [11], Camenisch *et al.* gave a more efficient construction with hidden policies based on the attribute-based encryption scheme of [47]. At the expense of a proof in the generic group model, [11] improves upon the expressiveness of [13] in that its policies extend into CNF formulas. While the solutions of [13,11] both hide the access policies to users (and the successful termination of transfers to the database), their policies can only live in a proper subset of $\mathsf{NC1}$. As of now, threshold policies can only be efficiently handled by the ABE-based construction of Zhang *et al.* [54], which requires *ad hoc* assumptions in groups with a bilinear map.

Our Results and Techniques. We describe the first OT-AC protocol based on lattice assumptions. Our construction supports access policies consisting of any branching program of width 5 and polynomial length – which suffices to realize any $\mathsf{NC1}$ circuit – and prove it secure under the SIS and LWE assumptions. We thus achieve the same level of expressiveness as [54] with the benefit of relying on well-studied assumptions. In its initial version, our protocol requires the database holder to communicate $\Theta(N)$ bits to each receiver so as to prove that the database is properly encrypted. In the random oracle model, we can eliminate this burden via the Fiat-Shamir heuristic and make the initialization cost linear in the database size N, regardless of the number of users.

As a first step, we build an ordinary $\mathcal{OT}^N_{k\times 1}$ protocol (i.e., without access control) via the "assisted decryption" approach of [14]. In short, the sender encrypts all database entries using a semantically secure cryptosystem. At each transfer, the receiver gets the sender to obliviously decrypt one of the initial ciphertexts without learning which one. Security against malicious adversaries is achieved by adding zero-knowledge (ZK) or witness indistinguishable (WI) arguments that the protocol is being followed. The desired ZK or WI arguments are obtained using the techniques of [37] and we prove that this basic protocol satisfies the full simulatability definitions of [14] under the SIS and LWE assumptions. To our knowledge, this protocol is also the first $\mathcal{OT}^N_{k\times 1}$ realization to achieve the standard simulation-based security requirements under lattice assumptions.

So far, all known "beyond-conjunctions" OT-AC protocols [54,11] rely on ciphertext-policy attribute-based encryption (CP-ABE) and proceed by attaching each database record to a CP-ABE ciphertext. Our construction departs from the latter approach for two reasons. First, the only known LWE-based CP-ABE schemes are obtained by applying a universal circuit to a key-policy system, making them very inefficient. Second, the ABE-based approach requires a fully secure ABE (i.e., selective security and semi-adaptive security are insufficient) since the access policies are dictated by the environment *after* the generation of

the issuer's public key, which contains the public parameters of the underlying ABE in [54,11]. Even with the best known LWE-based ABE candidates [26], a direct adaptation of the techniques in [54,11] would incur to rely on a complexity leveraging argument [8] and a universal circuit. Instead, we take a different approach and directly prove in a zero-knowledge manner the correct evaluation of a committed branching program for a hidden input.

At a high level, our OT-AC protocol works as follows. For each $i \in [N]$, the database entry $M_i \in \{0,1\}^t$ is associated with branching program BP_i. In the initialization step, the database holder generates a Regev ciphertext $(\mathbf{a}_i, \mathbf{b}_i)$ of M_i, and issues a certificate for the pair $\big((\mathbf{a}_i, \mathbf{b}_i), \mathsf{BP}_i\big)$, using the signature scheme from [37]. At each transfer, the user U who wishes to get a record $\rho \in [N]$ must obtain a credential $\mathsf{Cred}_{\mathbf{x}}$ for an attribute string $\mathbf{x} \in \{0,1\}^\kappa$ such that $\mathsf{BP}_\rho(\mathbf{x}) = 1$. Then, U modifies $(\mathbf{a}_\rho, \mathbf{b}_\rho)$ into an encryption of $M_\rho \oplus \mu \in \{0,1\}^t$, for some random string $\mu \in \{0,1\}^t$, and re-randomizes the resulting ciphertext into a fresh encryption $(\mathbf{c}_0, \mathbf{c}_1)$ of $M_\rho \oplus \mu$. At this point, U proves that $(\mathbf{c}_0, \mathbf{c}_1)$ was obtained by transforming one of the original ciphertexts $\{(\mathbf{a}_i, \mathbf{b}_i)\}_{i=1}^N$ by arguing possession of a valid certificate for $\big((\mathbf{a}_\rho, \mathbf{b}_\rho), \mathsf{BP}_\rho\big)$ and knowledge of all randomness used in the transformation that yields $(\mathbf{c}_0, \mathbf{c}_1)$. At the same time, U proves possession of $\mathsf{Cred}_{\mathbf{x}}$ for a string $\mathbf{x}$ which is accepted by the committed BP_ρ. To demonstrate these statements in zero-knowledge, we develop recent techniques [42,39,37] for lattice-based analogues [35,41] of Stern's protocol [52].

As a crucial component of our OT-AC protocol, we need to prove knowledge of an input $\mathbf{x} = (x_0, \ldots, x_{\kappa-1})^\top \in \{0,1\}^\kappa$ satisfying a hidden BP of length L, where L and κ are polynomials in the security parameter. For each $\theta \in [L]$, we need to prove that the computation of the θ-th state

$$\eta_\theta = \pi_{\theta,0}(\eta_{\theta-1}) \cdot (1 - x_{\mathrm{var}(\theta)}) + \pi_{\theta,1}(\eta_{\theta-1}) \cdot x_{\mathrm{var}(\theta)}, \tag{1}$$

is done correctly, for permutations $\pi_{\theta,0}, \pi_{\theta,1} : [0,4] \to [0,4]$ and for integer $\mathrm{var}(\theta) \in [0, \kappa-1]$ specified by BP. To date, equations of the form (1) have not been addressed in the context of zero-knowledge proofs for lattice-based cryptography. In this work, we are not only able to handle L such equations, but also manage to do so with a reasonable asymptotic cost.

In order to compute η_θ as in (1), we have to fetch the value $x_{\mathrm{var}(\theta)}$ in the input $(x_0, \ldots, x_{\kappa-1})^\top$ and provide evidence that the searching process is conducted honestly. If we perform a naive left-to-right search in the array $x_0, \ldots, x_{\kappa-1}$, the expected complexity is $\mathcal{O}(\kappa)$ for each step, and the total complexity amounts to $\mathcal{O}(L \cdot \kappa)$. If we instead perform a dichotomic search over $x_0, \ldots, x_{\kappa-1}$, we can decrease the complexity at each step down to $\mathcal{O}(\log \kappa)$. However, in this case, we need to prove in zero-knowledge a statement "I obtained $x_{\mathrm{var}(\theta)}$ by conducting a correct dichotomic search in my secret array."

We solve this problem as follows. For each $i \in [0, \kappa-1]$, we employ a SIS-based commitment scheme [35] to commit to x_i as com_i, and prove that the committed bits are consistent with the ones involved in the credential $\mathsf{Cred}_{\mathbf{x}}$ mentioned above. Then we build a SIS-based Merkle hash tree [39] of depth $\delta_\kappa = \lceil \log \kappa \rceil$ on top of the commitments $\mathsf{com}_0, \ldots, \mathsf{com}_{\kappa-1}$. Now, for each $\theta \in [L]$, we consider the binary representation $d_{\theta,1}, \ldots, d_{\theta,\delta_\kappa}$ of $\mathrm{var}(\theta)$. We then prove knowledge of

a bit y_θ such that these conditions hold: "If one starts at the root of the tree and follows the path determined by the bits $d_{\theta,1}, \ldots, d_{\theta,\delta_\kappa}$, then one will reach the leaf associated with the commitment opened to y_θ." The idea is that, if the Merkle tree and the commitment scheme are both secure, then it should be true that $y_\theta = x_{\mathrm{var}(\theta)}$. In other words, this enables us to provably perform a "binary search" for $x_{\mathrm{var}(\theta)} = y_\theta$. Furthermore, this process can be done in zero-knowledge, by adapting the recent techniques from [39]. As a result, we obtain a protocol with communication cost just $\mathcal{O}(L \cdot \log \kappa + \kappa)$.

2 Background and Definitions

Vectors are denoted in bold lower-case letters and bold upper-case letters will denote matrices. The Euclidean and infinity norm of any vector $\mathbf{b} \in \mathbb{R}^n$ will be denoted by $\|\mathbf{b}\|$ and $\|\mathbf{b}\|_\infty$, respectively. The Euclidean norm of matrix $\mathbf{B} \in \mathbb{R}^{m \times n}$ with columns $(\mathbf{b}_i)_{i \leq n}$ is $\|\mathbf{B}\| = \max_{i \leq n} \|\mathbf{b}_i\|$. When $\mathbf{B}$ has full column-rank, we let $\widetilde{\mathbf{B}}$ denote its Gram-Schmidt orthogonalization.

When S is a finite set, we denote by $U(S)$ the uniform distribution over S, and by $x \hookleftarrow U(S)$ the action of sampling x according to this distribution. Finally for any integers A, B, N, we let $[N]$ and $[A, B]$ denote the sets $\{1, \ldots, N\}$ and $\{A, A+1, \ldots, B\}$, respectively.

2.1 Lattices

A lattice L is the set of integer linear combinations of linearly independent basis vectors $(\mathbf{b}_i)_{i \leq n}$ living in $\mathbb{R}^m$. We work with q-ary lattices, for some prime q.

Definition 1. *Let $m \geq n \geq 1$, a prime $q \geq 2$ and $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{u} \in \mathbb{Z}_q^n$, define $\Lambda_q(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m \mid \exists \mathbf{s} \in \mathbb{Z}_q^n \ \ s.t. \ \ \mathbf{A}^T \cdot \mathbf{s} = \mathbf{e} \bmod q\}$ as well as*

$$\Lambda_q^\perp(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m \mid \mathbf{A} \cdot \mathbf{e} = \mathbf{0}^n \bmod q\},\ \Lambda_q^{\mathbf{u}}(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m \mid \mathbf{A} \cdot \mathbf{e} = \mathbf{u} \bmod q\}.$$

For a lattice L, let $\rho_{\sigma,\mathbf{c}}(\mathbf{x}) = \exp(-\pi\|\mathbf{x} - \mathbf{c}\|^2/\sigma^2)$ for a vector $\mathbf{c} \in \mathbb{R}^m$ and a real $\sigma > 0$. The discrete Gaussian of support L, center $\mathbf{c}$ and parameter σ is $D_{L,\sigma,\mathbf{c}}(\mathbf{y}) = \rho_{\sigma,\mathbf{c}}(\mathbf{y})/\rho_{\sigma,\mathbf{c}}(L)$ for any $\mathbf{y} \in L$, where $\rho_{\sigma,\mathbf{c}}(L) = \sum_{\mathbf{x} \in L} \rho_{\sigma,\mathbf{c}}(\mathbf{x})$. The distribution centered in $\mathbf{c} = \mathbf{0}$ is denoted by $D_{L,\sigma}(\mathbf{y})$.

It is well known that one can efficiently sample from a Gaussian distribution with lattice support given a sufficiently short basis of the lattice.

Lemma 1 ([10, Le. 2.3]). *There exists a PPT algorithm* GPVSample *that takes as inputs a basis $\mathbf{B}$ of a lattice $L \subseteq \mathbb{Z}^n$ and a rational $\sigma \geq \|\widetilde{\mathbf{B}}\| \cdot \Omega(\sqrt{\log n})$, and outputs vectors $\mathbf{b} \in L$ with distribution $D_{L,\sigma}$.*

We also rely on the trapdoor generation algorithm of Alwen and Peikert [4], which refines the technique of Gentry *et al.* [23].

Lemma 2 ([4, Th. 3.2]). *There is a PPT algorithm* TrapGen *that takes as inputs* 1^n, 1^m *and an integer* $q \geq 2$ *with* $m \geq \Omega(n \log q)$, *and outputs a matrix* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ *and a basis* $\mathbf{T_A}$ *of* $\Lambda_q^\perp(\mathbf{A})$ *such that* $\mathbf{A}$ *is within statistical distance* $2^{-\Omega(n)}$ *to* $U(\mathbb{Z}_q^{n \times m})$, *and* $\|\widetilde{\mathbf{T_A}}\| \leq \mathcal{O}(\sqrt{n \log q})$.

We use the basis delegation algorithm [16] that inputs a trapdoor for $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and produces a trapdoor for any $\mathbf{B} \in \mathbb{Z}_q^{n \times m'}$ containing $\mathbf{A}$ as a submatrix. A technique from Agrawal *et al.* [2] is sometimes used in our proofs.

Lemma 3 ([16, Le. 3.2]). *There is a PPT algorithm* ExtBasis *that inputs* $\mathbf{B} \in \mathbb{Z}_q^{n \times m'}$ *whose first* m *columns span* $\mathbb{Z}_q^n$, *and a basis* $\mathbf{T_A}$ *of* $\Lambda_q^\perp(\mathbf{A})$ *where* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ *is a submatrix of* $\mathbf{B}$, *and outputs a basis* $\mathbf{T_B}$ *of* $\Lambda_q^\perp(\mathbf{B})$ *with* $\|\widetilde{\mathbf{T_B}}\| \leq \|\widetilde{\mathbf{T_A}}\|$.

Lemma 4 ([2, Th. 19]). *There is a PPT algorithm* SampleRight *that inputs* $\mathbf{A}, \mathbf{C} \in \mathbb{Z}_q^{n \times m}$, *a small-norm* $\mathbf{R} \in \mathbb{Z}^{m \times m}$, *a short basis* $\mathbf{T_C} \in \mathbb{Z}^{m \times m}$ *of* $\Lambda_q^\perp(\mathbf{C})$, *a vector* $\mathbf{u} \in \mathbb{Z}_q^n$ *and a rational* σ *such that* $\sigma \geq \|\widetilde{\mathbf{T_C}}\| \cdot \Omega(\sqrt{\log n})$, *and outputs* $\mathbf{b} \in \mathbb{Z}^{2m}$ *such that* $\left[\mathbf{A} \;\middle|\; \mathbf{A} \cdot \mathbf{R} + \mathbf{C}\right] \cdot \mathbf{b} = \mathbf{u} \bmod q$ *and with distribution statistically close to* $D_{L,\sigma}$ *where* $L = \{\mathbf{x} \in \mathbb{Z}^{2m} : \left[\mathbf{A} \;\middle|\; \mathbf{A} \cdot \mathbf{R} + \mathbf{C}\right] \cdot \mathbf{x} = \mathbf{u} \bmod q\}$.

2.2 Hardness Assumptions

Definition 2. *Let* n, m, q, β *be functions of* $\lambda \in \mathbb{N}$. *The Short Integer Solution problem* $\mathsf{SIS}_{n,m,q,\beta}$ *is, given* $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{n \times m})$, *find* $\mathbf{x} \in \Lambda_q^\perp(\mathbf{A})$ *with* $0 < \|\mathbf{x}\| \leq \beta$.

If $q \geq \sqrt{n}\beta$ and $m, \beta \leq \mathsf{poly}(n)$, then standard worst-case lattice problems with approximation factors $\gamma = \widetilde{\mathcal{O}}(\beta\sqrt{n})$ reduce to $\mathsf{SIS}_{n,m,q,\beta}$ (see, e.g., [23, Se. 9]).

Definition 3. *Let* $n, m \geq 1$, $q \geq 2$, *and let* χ *be a probability distribution on* $\mathbb{Z}$. *For* $\mathbf{s} \in \mathbb{Z}_q^n$, *let* $A_{\mathbf{s},\chi}$ *be the distribution obtained by sampling* $\mathbf{a} \hookleftarrow U(\mathbb{Z}_q^n)$ *and* $e \hookleftarrow \chi$, *and outputting* $(\mathbf{a}, \mathbf{a}^T \cdot \mathbf{s} + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. *The Learning With Errors problem* $\mathsf{LWE}_{n,q,\chi}$ *asks to distinguish* m *samples chosen according to* $\mathcal{A}_{\mathbf{s},\chi}$ *(for* $\mathbf{s} \hookleftarrow U(\mathbb{Z}_q^n)$*) and* m *samples chosen according to* $U(\mathbb{Z}_q^n \times \mathbb{Z}_q)$.

If q is a prime power, $B \geq \sqrt{n}\omega(\log n)$, $\gamma = \widetilde{\mathcal{O}}(nq/B)$, then there exists an efficient sampleable B-bounded distribution χ (i.e., χ outputs samples with norm at most B with overwhelming probability) such that $\mathsf{LWE}_{n,q,\chi}$ is as least as hard as SIVP_γ (see, e.g., [50,10]).

2.3 Adaptive Oblivious Transfer

In the syntax of [14], an adaptive k-out-of-N OT scheme $\mathcal{OT}_k^N$ is a tuple of stateful PPT algorithms $(\mathsf{S_I}, \mathsf{R_I}, \mathsf{S_T}, \mathsf{R_T})$. The sender $\mathsf{S} = (\mathsf{S_I}, \mathsf{S_T})$ consists of two interactive algorithms $\mathsf{S_I}$ and $\mathsf{S_T}$ and the receiver has a similar representation as algorithms $\mathsf{R_I}$ and $\mathsf{R_T}$. In the *initialization phase*, the sender and the receiver run interactive algorithms $\mathsf{S_I}$ and $\mathsf{R_I}$, respectively, where $\mathsf{S_I}$ takes as input messages

$M_1, \ldots, M_N$ while $\mathsf{R_I}$ has no input. This phase ends with the two algorithms $\mathsf{S_I}$ and $\mathsf{R_I}$ outputting their state information S_0 and R_0 respectively.

During the i-th *transfer*, $1 \leq i \leq k$, both parties run an interactive protocol via the $\mathsf{R_T}$ and $\mathsf{S_T}$ algorithms. The sender starts runs $\mathsf{S_T}(S_{i-1})$ to obtain its updated state information S_i while the receiver runs $\mathsf{R_T}(R_{i-1}, \rho_i)$ on input of its previous state R_{i-1} and the index $\rho_i \in \{1, \ldots, N\}$ of the message it wishes to retrieve. At the end, $\mathsf{R_T}$ outputs an updated state R_i and a message M'_{ρ_i}.

Correctness mandates that, for all $M_1, \ldots, M_N$, for all $\rho_1, \ldots, \rho_k \in [N]$ and all coin tosses ϖ of the (honestly run) algorithms, we have $M'_{\rho_i} = M_{\rho_i}$ for all i.

We consider protocols that are secure (against static corruptions) in the sense of simulation-based definitions. The security properties against a cheating sender and a cheating receiver are formalized via the "real-world/ideal-world" paradigm. The security definitions of [14] are recalled in the full paper.

2.4 Adaptive Oblivious Transfer with Access Control

Camenisch *et al.* [12] define oblivious transfer with access control (OT-AC) as a tuple of PPT algorithms/protocols $(\mathsf{ISetup}, \mathsf{Issue}, \mathsf{DBSetup}, \mathsf{Transfer})$ such that:

ISetup: takes as inputs public parameters pp specifying a set $\mathcal{P}$ of access policies and generates a key pair (PK_I, SK_I) for the issuer.

Issue: is an interactive protocol between the issuer I and a stateful user U under common input (pp, x), where x is an attribute string. The issuer I takes as inputs its key pair (PK_I, SK_I) and a user pseudonym P_{U}. The user takes as inputs its state information st_{U}. The user U outputs either an error symbol $\perp$ or a credential $\mathsf{Cred_U}$, and an updated state st'_{U}.

DBSetup: is an algorithm that takes as input the issuer's public key PK_I, a database $DB = (M_i, \mathsf{AP}_i)_{i=1}^N$ containing records M_i whose access is restricted by an access policy AP_i and outputs a database public key PK_{DB}, an encryption of the records $(ER_i)_{i=1}^N$ and a database secret key SK_{DB}.

Transfer: is a protocol between the database DB and a user U with common inputs (PK_I, PK_{DB}). DB inputs SK_{DB} and U inputs $(\rho, st_{\mathsf{U}}, ER_\rho, \mathsf{AP}_\rho)$, where $\rho \in [N]$ is a record index to which U is requesting access. The interaction ends with U outputting $\perp$ or a string $M_{\rho'}$ and an updated state st'_{U}.

We assume private communication links, so that communications between a user and the issuer are authenticated, and those between a user and the database are anonymized: otherwise, anonymizing the $\mathsf{Transfer}$ protocol is impossible.

The security definitions formalize two properties called *user anonymity* and *database security*. The former captures that the database should be unable to tell which honest user is making a query and neither can tell which records are being accessed. This should remain true even if the database colludes with corrupted users and the issuer. As for database security, the intuition is that a cheating user cannot access a record for which it does not have the required credentials, even when colluding with other dishonest users. In case the issuer is colluding with these cheating users, they cannot obtain more records from the database than they retrieve. Precise security definitions [12] are recalled in the full version of the paper.

2.5 Vector Decompositions

We will employ the decomposition technique from [41,37], which allows transforming vectors with infinity norm larger than 1 into vectors with infinity norm 1.

For any $B \in \mathbb{Z}_+$, define the number $\delta_B := \lfloor \log_2 B \rfloor + 1 = \lceil \log_2(B+1) \rceil$ and the sequence $B_1, \ldots, B_{\delta_B}$, where $B_j = \lfloor \frac{B+2^{j-1}}{2^j} \rfloor$, $\forall j \in [1, \delta_B]$. This sequence satisfies $\sum_{j=1}^{\delta_B} B_j = B$ and any integer $v \in [0, B]$ can be decomposed to $\mathsf{idec}_B(v) = (v^{(1)}, \ldots, v^{(\delta_B)})^\top \in \{0,1\}^{\delta_B}$ such that $\sum_{j=1}^{\delta_B} B_j \cdot v_j = v$. We describe this decomposition procedure in a deterministic manner as follows:

1. Set $v' := v$; For $j = 1$ to δ_B do:
   If $v' \geq B_j$ then $v^{(j)} := 1$, else $v^{(j)} := 0$; $v' := v' - B_j \cdot v^{(j)}$.
2. Output $\mathsf{idec}_B(v) = (v^{(1)}, \ldots, v^{(\delta_B)})^\top$.

For any positive integers $\mathfrak{m}, B$, we define $\mathbf{H}_{\mathfrak{m},B} := \mathbf{I}_{\mathfrak{m}} \otimes [B_1 | \ldots | B_{\delta_B}] \in \mathbb{Z}^{\mathfrak{m} \times \mathfrak{m}\delta_B}$ and the following injective functions:

(i) $\mathsf{vdec}_{\mathfrak{m},B} : [0,B]^{\mathfrak{m}} \to \{0,1\}^{\mathfrak{m}\delta_B}$ that maps vector $\mathbf{v} = (v_1, \ldots, v_{\mathfrak{m}})$ to vector $\left(\mathsf{idec}_B(v_1)^\top \| \ldots \| \mathsf{idec}_B(v_{\mathfrak{m}})^\top\right)^\top$. Note that $\mathbf{H}_{\mathfrak{m},B} \cdot \mathsf{vdec}_{\mathfrak{m},B}(\mathbf{v}) = \mathbf{v}$.

(ii) $\mathsf{vdec}'_{\mathfrak{m},B} : [-B,B]^{\mathfrak{m}} \to \{-1,0,1\}^{\mathfrak{m}\delta_B}$ that maps vector $\mathbf{w} = (w_1, \ldots, w_{\mathfrak{m}})$ to vector $\left(\sigma(w_1) \cdot \mathsf{idec}_B(|w_1|)^\top \| \ldots \| \sigma(w_{\mathfrak{m}}) \cdot \mathsf{idec}_B(|w_{\mathfrak{m}}|)^\top\right)^\top$, where for each $i = 1, \ldots, \mathfrak{m}$: $\sigma(w_i) = 0$ if $w_i = 0$; $\sigma(w_i) = -1$ if $w_i < 0$; $\sigma(w_i) = 1$ if $w_i > 0$. Note that $\mathbf{H}_{\mathfrak{m},B} \cdot \mathsf{vdec}'_{\mathfrak{m},B}(\mathbf{w}) = \mathbf{w}$.

3 Building Blocks

We will use two distinct signature schemes because one of them only needs to be secure in the sense of a weaker security notion and can be more efficient. This weaker notion is sufficient to sign the database entries and allows a better efficiency in the scheme of Section 4.

3.1 Signatures Supporting Efficient Zero-Knowledge Proofs

We use a signature scheme proposed by Libert *et al.* [37] who extended the Böhl *et al.* signature [7] in order to sign messages comprised of multiple blocks while keeping the scheme compatible with zero-knowledge proofs.

$\mathsf{Keygen}(1^\lambda, 1^{N_b})$: Given a security parameter $\lambda > 0$ and the number of blocks $N_b = \mathsf{poly}(\lambda)$, choose $n = \mathcal{O}(\lambda)$, a prime modulus $q = \widetilde{\mathcal{O}}(N \cdot n^4)$, a dimension $m = 2n\lceil \log q \rceil$; an integer $\ell = \mathsf{poly}(n)$ and Gaussian parameters $\sigma = \Omega(\sqrt{n \log q} \log n)$. Define the message space as $\mathcal{M} = (\{0,1\}^{m_I})^{N_b}$.

1. Run $\mathsf{TrapGen}(1^n, 1^m, q)$ to get $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a short basis $\mathbf{T}_{\mathbf{A}}$ of $\Lambda_q^\perp(\mathbf{A})$. This basis allows computing short vectors in $\Lambda_q^\perp(\mathbf{A})$ with a Gaussian parameter σ. Next, choose $\ell + 1$ random $\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_\ell \hookleftarrow U(\mathbb{Z}_q^{n \times m})$.

2. Choose random matrices $\mathbf{D} \hookleftarrow U(\mathbb{Z}_q^{n\times m/2})$, $\mathbf{D}_0 \hookleftarrow U(\mathbb{Z}_q^{n\times m}), \mathbf{D}_j \hookleftarrow U(\mathbb{Z}_q^{n\times m_I})$ for $j = 1, \ldots, N_b$, as well as a random vector $\mathbf{u} \hookleftarrow U(\mathbb{Z}_q^n)$.

The private signing key consists of $SK := \mathbf{T_A}$ while the public key is comprised of $PK := \big(\mathbf{A}, \{\mathbf{A}_j\}_{j=0}^{\ell}, \mathbf{D}, \{\mathbf{D}_k\}_{k=0}^{N_b}, \mathbf{u}\big)$.

Sign$\big(SK, \mathsf{Msg}\big)$**:** To sign an N_b-block $\mathsf{Msg} = (\mathfrak{m}_1, \ldots, \mathfrak{m}_{N_b}) \in (\{0,1\}^{m_I})^{N_b}$,

1. Choose $\tau \hookleftarrow U(\{0,1\}^{\ell})$. Using $SK := \mathbf{T_A}$, compute a short basis $\mathbf{T}_\tau \in \mathbb{Z}^{2m\times 2m}$ for $\Lambda_q^{\perp}(\mathbf{A}_\tau)$, where $\mathbf{A}_\tau = [\mathbf{A} \mid \mathbf{A}_0 + \sum_{j=1}^{\ell} \tau[j]\mathbf{A}_j] \in \mathbb{Z}_q^{n\times 2m}$.
2. Sample $\mathbf{r} \hookleftarrow D_{\mathbb{Z}^m,\sigma}$. Compute the vector $\mathbf{c}_M \in \mathbb{Z}_q^n$ as a chameleon hash of $(\mathfrak{m}_1, \ldots, \mathfrak{m}_{N_b})$. Namely, compute $\mathbf{c}_M = \mathbf{D}_0 \cdot \mathbf{r} + \sum_{k=1}^{N_b} \mathbf{D}_k \cdot \mathfrak{m}_k \in \mathbb{Z}_q^n$, which is used to define $\mathbf{u}_M = \mathbf{u} + \mathbf{D} \cdot \mathsf{vdec}_{n,q-1}(\mathbf{c}_M) \in \mathbb{Z}_q^n$. Using the delegated basis $\mathbf{T}_\tau \in \mathbb{Z}^{2m\times 2m}$, sample a vector $\mathbf{v} \in \mathbb{Z}^{2m}$ in $D_{\Lambda_q^{\mathbf{u}_M}(\mathbf{A}_\tau),\sigma}$.

Output the signature $sig = (\tau, \mathbf{v}, \mathbf{r}) \in \{0,1\}^{\ell} \times \mathbb{Z}^{2m} \times \mathbb{Z}^m$.

Verify$\big(PK, \mathsf{Msg}, sig\big)$**:** Given $\mathsf{Msg} = (\mathfrak{m}_1, \ldots, \mathfrak{m}_{N_b}) \in (\{0,1\}^{m_I})^{N_b}$ and $sig = (\tau, \mathbf{v}, \mathbf{r}) \in \{0,1\}^{\ell} \times \mathbb{Z}^{2m} \times \mathbb{Z}^m$, return 1 if $\|\mathbf{v}\| < \sigma\sqrt{2m}$, $\|\mathbf{r}\| < \sigma\sqrt{m}$ and

$$\mathbf{A}_\tau \cdot \mathbf{v} = \mathbf{u} + \mathbf{D} \cdot \mathsf{vdec}_{n,q-1}(\mathbf{D}_0 \cdot \mathbf{r} + \sum_{k=1}^{N_b} \mathbf{D}_k \cdot \mathfrak{m}_k) \bmod q. \tag{2}$$

3.2 A Simpler Variant with Bounded-Message Security and Security Against Non-Adaptive Chosen-Message Attacks

We consider a stateful variant of the scheme in Section 3.1 where a bound $Q \in \mathsf{poly}(n)$ on the number of signed messages is fixed at key generation time. In the context of $\mathcal{OT}_{k\times 1}^N$, this is sufficient and leads to efficiency improvements. In the modified scheme hereunder, the string $\tau \in \{0,1\}^{\ell}$ is an ℓ-bit counter maintained by the signer to keep track of the number of previously signed messages. This simplified variant resembles the SIS-based signature scheme of Böhl *et al.* [7].

In this version, the message space is $\{0,1\}^{n\lceil \log q\rceil}$ so that vectors of $\mathbb{Z}_q^n$ can be signed by first decomposing them using $\mathsf{vdec}_{n,q-1}(.)$.

Keygen$(1^\lambda, 1^Q)$**:** Given $\lambda > 0$ and the maximal number $Q \in \mathsf{poly}(\lambda)$ of signatures, choose $n = \mathcal{O}(\lambda)$, a prime $q = \widetilde{\mathcal{O}}(Q \cdot n^4)$, $m = 2n\lceil \log q\rceil$, an integer $\ell = \lceil \log Q\rceil$ and Gaussian parameters $\sigma = \Omega(\sqrt{n \log q} \log n)$. The message space is $\{0,1\}^{m_d}$, for some $m_d \in \mathsf{poly}(\lambda)$ with $m_d \geq m$.

1. Run $\mathsf{TrapGen}(1^n, 1^m, q)$ to get $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and a short basis $\mathbf{T_A}$ of $\Lambda_q^{\perp}(\mathbf{A})$, which allows sampling short vectors in $\Lambda_q^{\perp}(\mathbf{A})$ with a Gaussian parameter σ. Next, choose $\ell + 1$ random $\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_\ell \hookleftarrow U(\mathbb{Z}_q^{n\times m})$.
2. Choose $\mathbf{D} \hookleftarrow U(\mathbb{Z}_q^{n\times m_d})$ as well as a random vector $\mathbf{u} \hookleftarrow U(\mathbb{Z}_q^n)$.

The counter τ is initialized to $\tau = 0$. The private key consists of $SK := \mathbf{T_A}$ and the public key is $PK := \big(\mathbf{A},\ \{\mathbf{A}_j\}_{j=0}^{\ell},\ \mathbf{D},\ \mathbf{u}\big)$.

Sign$\big(SK, \tau, \mathfrak{m}\big)$**:** To sign a message $\mathfrak{m} \in \{0,1\}^{m_d}$,

1. Increment the counter by setting $\tau := \tau + 1$ and interpret it as a string $\tau \in \{0,1\}^\ell$. Then, using $SK := \mathbf{T}_\mathbf{A}$, compute a short delegated basis $\mathbf{T}_\tau \in \mathbb{Z}^{2m \times 2m}$ for the matrix $\mathbf{A}_\tau = [\mathbf{A} \mid \mathbf{A}_0 + \sum_{j=1}^{\ell} \tau[j]\mathbf{A}_j] \in \mathbb{Z}_q^{n \times 2m}$.
2. Compute the vector $\mathbf{u}_M = \mathbf{u} + \mathbf{D} \cdot \mathfrak{m} \in \mathbb{Z}_q^n$. Then, using the delegated basis $\mathbf{T}_\tau \in \mathbb{Z}^{2m \times 2m}$, sample a short vector $\mathbf{v} \in \mathbb{Z}^{2m}$ in $D_{\Lambda_q^{\mathbf{u}_M}(\mathbf{A}_\tau),\sigma}$.

Output the signature $sig = (\tau, \mathbf{v}) \in \{0,1\}^\ell \times \mathbb{Z}^{2m}$.

Verify$\big(PK, \mathfrak{m}, sig\big)$**:** Given PK, $\mathfrak{m} \in \{0,1\}^{m_d}$ and a signature $sig = (\tau, \mathbf{v}) \in \{0,1\}^\ell \times \mathbb{Z}^{2m}$, return 1 if $\|\mathbf{v}\| < \sigma\sqrt{2m}$ and $\mathbf{A}_\tau \cdot \mathbf{v} = \mathbf{u} + \mathbf{D} \cdot \mathfrak{m} \bmod q$.

For our purposes, the scheme only needs to satisfy a notion of bounded-message security under non-adaptive chosen-message attack.

Theorem 1. *The scheme is bounded message secure under non-adaptive chosen-message attacks if the* SIS *assumption holds.* (The proof is given in the full version of the paper.)

4 A Fully Simulatable Adaptive OT Protocol

Our basic $\mathcal{OT}_{k\times 1}^N$ protocol builds on the "assisted decryption" technique [14]. The databases holder encrypts all entries using a multi-bit variant [48] of Regev's cryptosystem [50] and proves the well-formedness of its public key and all ciphertexts. In addition, all ciphertexts are signed using a signature scheme. At each transfer, the receiver statistically re-randomizes a blinded version of the desired ciphertext, where the blinding is done via the additive homomorphism of Regev. Then, the receiver provides a witness indistinguishable (WI) argument that the modified ciphertext (which is submitted for oblivious decryption) is a transformation of one of the original ciphertexts by arguing knowledge of a signature on this hidden ciphertext. In response, the sender obliviously decrypts the modified ciphertext and argues in zero-knowledge that the response is correct.

Adapting the technique of [14] to the lattice setting requires the following building blocks: (i) A signature scheme allowing to sign ciphertexts while remaining compatible with ZK proofs; (ii) A ZK protocol allowing to prove knowledge of a signature on some hidden ciphertext which belongs to a public set and was transformed into a given ciphertext; (iii) A protocol for proving the correct decryption of a ciphertext; (iv) A method of statistically re-randomizing an LWE-encrypted ciphertext in a way that enables oblivious decryption. The first three ingredients can be obtained from [37]. Since component (i) only needs to be secure against random-message attacks as long as the adversary obtains at most N signatures, we use the simplified SIS-based signature scheme of Section 3.2. The statistical re-randomization of Regev ciphertexts is handled via the noise flooding technique [5], which consists in drowning the initial noise with a super-polynomially larger noise. While recent results [20] provide potentially more efficient alternatives, we chose the flooding technique for simplicity because it does not require the use of FHE (and also because the known multi-bit version [32] of the GSW FHE [24] incurs an *ad hoc* circular security assumption).

Our scheme works with security parameter λ, modulus q, lattice dimensions $n = \mathcal{O}(\lambda)$ and $m = 2n\lceil \log q \rceil$. Let $B_\chi = \widetilde{\mathcal{O}}(\sqrt{n})$, and let χ be a B_χ-bounded distribution. We also define an integer B as a randomization parameter such that $B = n^{\omega(1)} \cdot (m+1)B_\chi$ and $B + (m+1)B_\chi \leq q/5$ (to ensure decryption correctness). Our basic $\mathcal{OT}^N_{k\times 1}$ protocol goes as follows.

Initialization$\big(\mathsf{S_I}(1^\lambda, \mathrm{DB}), \mathsf{R_I}(1^\lambda)\big)$**:** In this protocol, the sender $\mathsf{S_I}$ has a database $\mathrm{DB} = (M_1, \ldots, M_N)$ of N messages, where $M_i \in \{0,1\}^t$ for each $i \in [N]$, for some $t \in \mathsf{poly}(\lambda)$. It interacts with the receiver $\mathsf{R_I}$ as follows.

1. Generate a key pair for the signature scheme of Section 3.2 in order to sign $Q = N$ messages of length $m_d = (n+t)\cdot\lceil \log q \rceil$ each. This key pair consists of $SK_{sig} = \mathbf{T_A} \in \mathbb{Z}^{m\times m}$ and $PK_{sig} := \big(\mathbf{A}, \{\mathbf{A}_j\}_{j=0}^{\ell}, \mathbf{D}, \mathbf{u}\big)$, where $\ell = \log N$ and $\mathbf{A}, \mathbf{A}_0, \ldots, \mathbf{A}_\ell \in U(\mathbb{Z}_q^{n\times m})$, $\mathbf{D} \in U(\mathbb{Z}_q^{n\times m_d})$. The counter is initialized to $\tau = 0$.
2. Choose $\mathbf{S} \hookleftarrow \chi^{n\times t}$ that will serve as a secret key for an LWE-based encryption scheme. Then, sample $\mathbf{F} \hookleftarrow U(\mathbb{Z}_q^{n\times m})$, $\mathbf{E} \hookleftarrow \chi^{m\times t}$ and compute
$$\mathbf{P} = [\mathbf{p}_1 | \ldots | \mathbf{p}_t] = \mathbf{F}^\top \cdot \mathbf{S} + \mathbf{E} \ \in \mathbb{Z}_q^{m\times t}, \tag{3}$$
so that $(\mathbf{F}, \mathbf{P}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^{m\times t}$ forms a public key for a t-bit variant of Regev's encryption scheme [50].
3. Sample vectors $\mathbf{a}_1, \ldots, \mathbf{a}_N \hookleftarrow U(\mathbb{Z}_q^n)$ and $\mathbf{x}_1, \ldots, \mathbf{x}_N \hookleftarrow \chi^t$ to compute
$$(\mathbf{a}_i, \mathbf{b}_i) = \big(\mathbf{a}_i,\ \mathbf{S}^\top \cdot \mathbf{a}_i + \mathbf{x}_i + M_i \cdot \lfloor q/2 \rfloor\big) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^t \qquad \forall i \in [N]. \tag{4}$$
4. For each $i \in [N]$, generate a signature $(\tau_i, \mathbf{v}_i) \leftarrow \mathsf{Sign}(SK_{sig}, \tau, \mathfrak{m}_i)$ on the decomposition $\mathfrak{m}_i = \mathsf{vdec}_{n+t,q-1}(\mathbf{a}_i^\top | \mathbf{b}_i^\top)^\top \in \{0,1\}^{m_d}$.
5. $\mathsf{S_I}$ sends $R_0 = \big(PK_{sig},\ (\mathbf{F}, \mathbf{P}),\ \{(\mathbf{a}_i, \mathbf{b}_i), (\tau_i, \mathbf{v}_i)\}_{i=1}^N\big)$ to $\mathsf{R_I}$ and interactively proves knowledge of small-norm $\mathbf{S} \in \mathbb{Z}^{n\times t}$, $\mathbf{E} \in \mathbb{Z}^{m\times t}$, short vectors $\{\mathbf{x}_i\}_{i=1}^N$ and t-bit messages $\{M_i\}_{i=1}^N$, for which (3) and (4) hold. To this end, $\mathsf{S_I}$ plays the role of the prover in the ZK argument system described in Section 6.3. If the argument of knowledge does not verify or if there exists $i \in [N]$ such that $(\tau_i, \mathbf{v}_i)$ is an invalid signature on the message $\mathfrak{m}_i = \mathsf{vdec}_{n+t,q-1}(\mathbf{a}_i^\top | \mathbf{b}_i^\top)^\top$ w.r.t. PK_{sig}, then $\mathsf{R_I}$ aborts.
6. Finally $\mathsf{S_I}$ defines $S_0 = \big((\mathbf{S}, \mathbf{E}), (\mathbf{F}, \mathbf{P}), PK_{sig}\big)$, which it keeps to itself.

Transfer$\big(\mathsf{S_T}(S_{i-1}), \mathsf{R_T}(R_{i-1}, \rho_i)\big)$**:** At the i-th transfer, the receiver $\mathsf{R_T}$ has state R_{i-1} and an index $\rho_i \in [1, N]$. It interacts as follows with the sender $\mathsf{S_T}$ that has state S_{i-1} in order to obtain M_{ρ_i} from DB.

1. $\mathsf{R_T}$ samples vectors $\mathbf{e} \hookleftarrow U(\{-1, 0, 1\}^m)$, $\mu \hookleftarrow U(\{0,1\}^t)$ and a random $\nu \hookleftarrow U([-B, B]^t)$ to compute
$$(\mathbf{c}_0, \mathbf{c}_1) = \big(\mathbf{a}_{\rho_i} + \mathbf{F}\cdot\mathbf{e},\ \mathbf{b}_{\rho_i} + \mathbf{P}^\top \cdot \mathbf{e} + \mu \cdot \lfloor q/2 \rfloor + \nu\big) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^t, \tag{5}$$
which is a re-randomization of $(\mathbf{a}_{\rho_i}, \mathbf{b}_{\rho_i} + \mu\cdot\lfloor q/2 \rfloor)$. The ciphertext $(\mathbf{c}_0, \mathbf{c}_1)$ is sent to $\mathsf{S_T}$. In addition, $\mathsf{R_T}$ provides an interactive WI argument that $(\mathbf{c}_0, \mathbf{c}_1)$ is indeed a transformation of $(\mathbf{a}_{\rho_i}, \mathbf{b}_{\rho_i})$ for some $\rho_i \in [N]$, and $\mathsf{R_T}$ knows a signature on $\mathfrak{m} = \mathsf{vdec}_{n+1,q-1}(\mathbf{a}_{\rho_i}^\top | \mathbf{b}_{\rho_i}^\top)^\top \in \{0,1\}^{m_d}$. To this end, $\mathsf{R_T}$ runs the prover in the ZK argument system in Section 6.5.

2. If the argument of step 1 verifies, $\mathsf{S_T}$ uses $\mathbf{S}$ to decrypt $(\mathbf{c}_0, \mathbf{c}_1) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^t$ and obtain $M' = \lfloor (\mathbf{c}_1 - \mathbf{S}^\top \cdot \mathbf{c}_0)/(q/2) \rceil \in \{0,1\}^t$, which is sent back to $\mathsf{R_T}$. In addition, $\mathsf{S_T}$ provides a zero-knowledge argument of knowledge of vector $\mathbf{y} = \mathbf{c}_1 - \mathbf{S}^\top \cdot \mathbf{c}_0 - M' \cdot \lfloor q/2 \rfloor \in \mathbb{Z}^t$ of norm $\|\mathbf{y}\|_\infty \leq q/5$ and small-norm matrices $\mathbf{E} \in \mathbb{Z}^{m \times t}$, $\mathbf{S} \in \mathbb{Z}^{n \times t}$ satisfying (modulo q)

$$\mathbf{P} = \mathbf{F}^\top \cdot \mathbf{S} + \mathbf{E} \ , \qquad \mathbf{c}_0^\top \cdot \mathbf{S} + \mathbf{y}^\top = \mathbf{c}_1^\top - {M'}^\top \cdot \lfloor q/2 \rfloor. \tag{6}$$

   To this end, $\mathsf{S_T}$ runs the prover in the ZK argument system in Section 6.4.
3. If the ZK argument produced by $\mathsf{S_T}$ does not properly verify at step 2, $\mathsf{R_T}$ halts and outputs $\perp$. Otherwise, $\mathsf{R_T}$ recalls the random string $\mu \in \{0,1\}^t$ that was chosen at step 1 and computes $M_{\rho_i} = M' \oplus \mu$. The transfer ends with $\mathsf{S_T}$ and $\mathsf{R_T}$ outputting $S_i = S_{i-1}$ and $R_i = R_{i-1}$, respectively.

In the initialization phase, the sender has to repeat step 5 with each receiver to prove that $\{(\mathbf{a}_i, \mathbf{b}_i)\}_{i=1}^N$ are well-formed. Using the Fiat-Shamir heuristic [22], we can decrease this initialization cost from $O(N \cdot U)$ to $O(N)$ (regardless of the number of users U) by making the proof non-interactive. This modification also reduces each transfer to 5 communication rounds since, even in the transfer phase, the sender's ZK arguments can be non-interactive and the receiver's arguments only need to be WI, which is preserved when the basic ZK protocol (which has a ternary challenge space) is repeated $\omega(\log n)$ times in parallel. Details are given in the full version of the paper.

The security of the above $\mathcal{OT}_{k\times 1}^N$ protocol against static corruptions is proved in the full version of the paper under the SIS and LWE assumptions.

5 OT with Access Control for Branching Programs

In this section, we extend our protocol of Section 4 into a protocol where database entries can be protected by access control policies consisting of branching programs. In a nutshell, the construction goes as follows.

When the database is set up, the sender signs (a binary representation of) each database entry $(\mathbf{a}_i, \mathbf{b}_i)$ together with a hash value $\mathbf{h}_{\mathsf{BP},i} \in \mathbb{Z}_q^n$ of the corresponding branching program. For each possessed attribute $\mathbf{x} \in \{0,1\}^\kappa$, the user U obtains a credential $\mathsf{Cred}_{\mathsf{U},\mathbf{x}}$ from the issuer.

If U has a credential $\mathsf{Cred}_{\mathsf{U},\mathbf{x}}$ for an attribute $\mathbf{x}$ satisfying the ρ-th branching program, U can re-randomize $(\mathbf{a}_\rho, \mathbf{b}_\rho)$ into $(\mathbf{c}_0, \mathbf{c}_1)$, which is given to the sender, while proving that: (i) He knows a signature $(\tau, \mathbf{v})$ on some message $(\mathbf{a}_\rho, \mathbf{b}_\rho, \mathbf{h}_{\mathsf{BP},\rho})$ such that $(\mathbf{c}_0, \mathbf{c}_1)$ is a re-randomization of $(\mathbf{a}_\rho, \mathbf{b}_\rho)$; (ii) The corresponding $\mathbf{h}_{\mathsf{BP},\rho}$ is the hash value of (the binary representation of) a branching program BP_ρ that accepts an attribute $\mathbf{x} \in \{0,1\}^\kappa$ for which he has a valid credential $\mathsf{Cred}_{\mathsf{U},\mathbf{x}}$ (i.e., $\mathsf{BP}_\rho(\mathbf{x}) = 1$).

While statement (i) can be proved as in Section 4, handling (ii) requires a method of proving the possession of a (committed) branching program BP and a (committed) input $\mathbf{x} \in \{0,1\}^\kappa$ such that $\mathsf{BP}(\mathbf{x}) = 1$ while demonstrating possession of a credential for $\mathbf{x}$.

Recall that a branching program BP of length L, input space $\{0,1\}^\kappa$ and width 5 is specified by L tuples of the form $(\mathrm{var}(\theta), \pi_{\theta,0}, \pi_{\theta,1})$ where

- $\mathrm{var} : [L] \to [0, \kappa-1]$ is a function that associates the θ-th tuple with the coordinate $x_{\mathrm{var}(\theta)} \in \{0,1\}$ of the input $\mathbf{x} = (x_0, \ldots, x_{\kappa-1})^\top$.
- $\pi_{\theta,0}, \pi_{\theta,1} : \{0,1,2,3,4\} \to \{0,1,2,3,4\}$ are permutations that determine the θ-th step of the evaluation.

On input $\mathbf{x} = (x_0, \ldots, x_{\kappa-1})^\top$, BP computes its output as follows. For each bit $b \in \{0,1\}$, let $\bar{b}$ denote the bit $1-b$. Let η_θ denote the state of computation at step θ. The initial state is $\eta_0 = 0$ and, for $\theta \in [1, L]$, the state η_θ is computed as

$$\eta_\theta = \pi_{\theta, x_{\mathrm{var}(\theta)}}(\eta_{\theta-1}) = \pi_{\theta,0}(\eta_{\theta-1}) \cdot \bar{x}_{\mathrm{var}(\theta)} + \pi_{\theta,1}(\eta_{\theta-1}) \cdot x_{\mathrm{var}(\theta)}.$$

Finally, the output of evaluation is $\mathsf{BP}(\mathbf{x}) = 1$ if $\eta_L = 0$, otherwise $\mathsf{BP}(\mathbf{x}) = 0$.

We now let $\delta_\kappa = \lceil \log_2 \kappa \rceil$ and note that each integer in $[0, \kappa-1]$ can be determined by δ_κ bits. In particular, for each $\theta \in [L]$, let $d_{\theta,1}, \ldots, d_{\theta,\delta_\kappa}$ be the bits representing $\mathrm{var}(\theta)$. Then, we consider the following representation of BP:

$$\begin{aligned}\mathbf{z}_{\mathsf{BP}} = \big(&d_{1,1}, \ldots, d_{1,\delta_\kappa}, \ldots, d_{L,1}, \ldots, d_{L,\delta_\kappa}, \pi_{1,0}(0), \ldots, \pi_{1,0}(4), \pi_{1,1}(0), \ldots, \\ &\pi_{1,1}(4), \ldots, \pi_{L,0}(0), \ldots, \pi_{L,0}(4), \pi_{L,1}(0), \ldots, \pi_{L,1}(4)\big)^\top \in [0,4]^\zeta, \end{aligned} \tag{7}$$

where $\zeta = L(\delta_\kappa + 10)$.

5.1 The OT-AC Protocol

We assume public parameters pp consisting of a modulus q, integers n, m such that $m = 2n\lceil \log q \rceil$, a public matrix $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times m}$, the maximal length $L \in \mathsf{poly}(n)$ of branching programs and their desired input length $\kappa \in \mathsf{poly}(n)$.

$\mathsf{ISetup}(\mathsf{pp})$: Given public parameters $\mathsf{pp} = \{q, n, m, \bar{\mathbf{A}}, L, \kappa\}$, generate a key pair $(PK_I, SK_I) \leftarrow \mathsf{Keygen}(\mathsf{pp}, 1)$ for the signature scheme in Section 3.1 in order to sign single-block messages (i.e., $N_b = 1$) of length $m_I = n \cdot \lceil \log q \rceil + \kappa$. Letting $\ell_I = \mathcal{O}(n)$, this key pair contains $SK_I = \mathbf{T}_{\mathbf{A}_I} \in \mathbb{Z}^{m \times m}$ and

$$PK_I := \big(\mathbf{A}_I,\ \{\mathbf{A}_{I,j}\}_{j=0}^{\ell_I},\ \mathbf{D}_I,\ \{\mathbf{D}_{I,0}, \mathbf{D}_{I,1}\},\ \mathbf{u}_I\big).$$

$\mathsf{Issue}\big(\mathsf{I}(\mathsf{pp}, SK_I, PK_I, P_\mathsf{U}, \mathbf{x}) \leftrightarrow \mathsf{U}(\mathsf{pp}, \mathbf{x}, st_\mathsf{U})\big)$: On common input $\mathbf{x} \in \{0,1\}^\kappa$, the issuer I and the user U interact in the following way:

1. If $st_\mathsf{U} = \emptyset$, U creates a pseudonym $P_\mathsf{U} = \bar{\mathbf{A}} \cdot \mathbf{e}_\mathsf{U} \in \mathbb{Z}_q^n$, for a randomly chosen $\mathbf{e}_\mathsf{U} \hookleftarrow U(\{0,1\}^m)$, which is sent to I. It sets $st_\mathsf{U} = (\mathbf{e}_\mathsf{U}, P_\mathsf{U}, 0, \emptyset, \emptyset)$. Otherwise, U parses its state st_U as $(\mathbf{e}_\mathsf{U}, P_\mathsf{U}, f_{DB}, C_\mathsf{U}, \mathsf{Cred}_\mathsf{U})$.
2. The issuer I defines the message $\mathfrak{m}_{\mathsf{U},\mathbf{x}} = (\mathsf{vdec}_{n,q-1}(P_\mathsf{U})^\top | \mathbf{x}^\top)^\top \in \{0,1\}^{m_I}$. Then, it runs the signing algorithm of Section 3.1 to obtain and return $\mathsf{cert}_{\mathsf{U},\mathbf{x}} = \big(\tau_\mathsf{U}, \mathbf{v}_\mathsf{U}, \mathbf{r}_\mathsf{U}\big) \leftarrow \mathsf{Sign}(SK_I, \mathfrak{m}_{\mathsf{U},\mathbf{x}}) \in \{0,1\}^{\ell_I} \times \mathbb{Z}^{2m} \times \mathbb{Z}^m$, which binds U's pseudonym P_U to the attribute string $\mathbf{x} \in \{0,1\}^\kappa$.

3. U checks that $\mathsf{cert}_{\mathsf{U},\mathbf{x}}$ satisfies (2) and that $\|\mathbf{v}_{\mathsf{U}}\| \leq \sigma\sqrt{2m}, \mathbf{r}_{\mathsf{U}} \leq \sigma\sqrt{m}$. If so, U sets $C_{\mathsf{U}} := C_{\mathsf{U}} \cup \{\mathbf{x}\}$, $\mathsf{Cred}_{\mathsf{U}} := \mathsf{Cred}_{\mathsf{U}} \cup \{\mathsf{cert}_{\mathsf{U},\mathbf{x}}\}$ and updates its state $st_{\mathsf{U}} = (\mathbf{e}_{\mathsf{U}}, P_{\mathsf{U}}, f_{DB}, C_{\mathsf{U}}, \mathsf{Cred}_{\mathsf{U}})$. If $\mathsf{cert}_{\mathsf{U},\mathbf{x}}$ does not properly verify, U aborts the interaction and leaves st_{U} unchanged.

$\mathsf{DBSetup}\big(PK_I, \mathrm{DB} = \{(M_i, \mathsf{BP}_i)\}_{i=1}^N\big)$: The sender DB has $\mathrm{DB} = \{(M_i, \mathsf{BP}_i)\}_{i=1}^N$ which is a database of N pairs made of a message $M_i \in \{0,1\}^t$ and a policy realized by a length-L branching program $\mathsf{BP}_i = \{\mathrm{var}_i(\theta), \pi_{i,\theta,0}, \pi_{i,\theta,1}\}_{\theta=1}^L$.

1. Choose a random matrix $\mathbf{A}_{\mathrm{HBP}} \hookleftarrow U\big(\mathbb{Z}_q^{n\times\zeta}\big)$ which will be used to hash the description of branching programs.
2. Generate a key pair for the signature scheme of Section 3.2 in order to sign $Q = N$ messages of length $m_d = (2n+t)\cdot\lceil\log q\rceil$ each. This key pair consists of $SK_{sig} = \mathbf{T}_{\mathbf{A}} \in \mathbb{Z}^{m\times m}$ and $PK_{sig} := \big(\mathbf{A}, \{\mathbf{A}_j\}_{j=0}^{\ell}, \mathbf{D}, \mathbf{u}\big)$, where $\ell = \lceil\log N\rceil$ and $\mathbf{A}, \mathbf{A}_0, \ldots, \mathbf{A}_\ell \in U(\mathbb{Z}_q^{n\times m})$, $\mathbf{D} \in U(\mathbb{Z}_q^{n\times m_d})$ with $m = 2n\lceil\log q\rceil$, $m_d = (2n+t)\lceil\log q\rceil$. The counter is initialized to $\tau = 0$.
3. Sample $\mathbf{S} \hookleftarrow \chi^{n\times t}$ which will serve as a secret key for an LWE-based encryption scheme. Then, sample $\mathbf{F} \hookleftarrow U(\mathbb{Z}_q^{n\times m})$, $\mathbf{E} \hookleftarrow \chi^{m\times t}$ to compute
$$\mathbf{P} = [\mathbf{p}_1|\ldots|\mathbf{p}_t] = \mathbf{F}^\top \cdot \mathbf{S} + \mathbf{E} \ \in \mathbb{Z}_q^{m\times t} \tag{8}$$
so that $(\mathbf{F}, \mathbf{P})$ forms a public key for a t-bit variant of Regev's system.
4. Sample vectors $\mathbf{a}_1, \ldots, \mathbf{a}_N \hookleftarrow U(\mathbb{Z}_q^n)$ and $\mathbf{x}_1, \ldots, \mathbf{x}_N \hookleftarrow \chi^t$ to compute
$$(\mathbf{a}_i, \mathbf{b}_i) = \big(\mathbf{a}_i,\ \mathbf{a}_i^\top \cdot \mathbf{S} + \mathbf{x}_i + M_i \cdot \lfloor q/2\rfloor\big) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^t \qquad \forall i \in [N] \tag{9}$$
5. For each $i = 1$ to N, $(\mathbf{a}_i, \mathbf{b}_i)$ is bound to BP_i as follows.
   a. Let $\mathbf{z}_{\mathsf{BP},i} \in [0,4]^\zeta$ be the binary representation of the branching program. Compute its digest $\mathbf{h}_{\mathsf{BP},i} = \mathbf{A}_{\mathrm{HBP}} \cdot \mathbf{z}_{\mathsf{BP},i} \in \mathbb{Z}_q^n$.
   b. Using SK_{sig}, generate a signature $(\tau_i, \mathbf{v}_i) \leftarrow \mathsf{Sign}(SK_{sig}, \tau, \mathfrak{m}_i)$ on the message $\mathfrak{m}_i = \mathsf{vdec}_{2n+t,q-1}(\mathbf{a}_i|\mathbf{b}_i|\mathbf{h}_{\mathsf{BP},i}) \in \{0,1\}^{m_d}$ obtained by decomposing $(\mathbf{a}_i^\top|\mathbf{b}_i^\top|\mathbf{h}_{\mathsf{BP},i}^\top)^\top \in \mathbb{Z}_q^{2n+t}$.
6. The database's public key is defined as $PK_{\mathrm{DB}} = \big(PK_{sig},\ (\mathbf{F}, \mathbf{P}),\ \mathbf{A}_{\mathrm{HBP}}\big)$ while the encrypted database is $\{ER_i = \big(\mathbf{a}_i, \mathbf{b}_i, (\tau_i, \mathbf{v}_i)\big), \mathsf{BP}_i\}_{i=1}^N$. The sender DB outputs $\big(PK_{\mathrm{DB}}, \{ER_i, \mathsf{BP}_i\}_{i=1}^N\big)$ and keeps $SK_{\mathrm{DB}} = \big(SK_{sig}, \mathbf{S}\big)$.

$\mathsf{Transfer}\big(\mathsf{DB}(SK_{\mathrm{DB}}, PK_{\mathrm{DB}}, PK_I), \mathsf{U}(\rho, st_{\mathsf{U}}, PK_I, PK_{\mathrm{DB}}, ER_\rho, \mathsf{BP}_\rho)\big)$: Given an index $\rho \in [N]$, a record $ER_\rho = \big(\mathbf{a}_\rho, \mathbf{b}_\rho, (\tau_\rho, \mathbf{v}_\rho)\big)$ and a policy BP_ρ, the user U parses st_{U} as $(\mathbf{e}_{\mathsf{U}}, P_{\mathsf{U}}, f_{DB}, C_{\mathsf{U}}, \mathsf{Cred}_{\mathsf{U}})$. If C_{U} does not contain any $\mathbf{x} \in \{0,1\}^\kappa$ s.t. $\mathsf{BP}_\rho(\mathbf{x}) = 1$ and $\mathsf{Cred}_{\mathsf{U}}$ contains the corresponding $\mathsf{cert}_{\mathsf{U},\mathbf{x}}$, U outputs $\perp$. Otherwise, he selects such a pair $(\mathbf{x}, \mathsf{cert}_{\mathsf{U},\mathbf{x}})$ and interacts with DB:

1. If $f_{DB} = 0$, U interacts with DB for the first time and requires DB to prove knowledge of small-norm $\mathbf{S} \in \mathbb{Z}^{n\times t}$, $\mathbf{E} \in \mathbb{Z}^{m\times t}$, $\{\mathbf{x}_i\}_{i=1}^N$ and t-bit messages $\{M_i\}_{i=1}^N$ satisfying (8)-(9). To do this, DB uses the ZK argument in Section 6.3. If there exists $i \in [N]$ such that $(\tau_i, \mathbf{v}_i)$ is an invalid signature on $\mathsf{vdec}_{2n+t,q-1}(\mathbf{a}_i^\top|\mathbf{b}_i^\top|\mathbf{h}_{\mathsf{BP},i}^\top)^\top$ or if the ZK argument does not verify, U aborts. Otherwise, U updates st_{U} and sets $f_{DB} = 1$.

2. U re-randomizes the pair $(\mathbf{a}_\rho, \mathbf{b}_\rho)$ contained in ER_ρ. It samples vectors $\mathbf{e} \hookleftarrow U(\{-1,0,1\}^m)$, $\mu \hookleftarrow U(\{0,1\}^t)$ and $\nu \hookleftarrow U([-B,B]^t)$ to compute

$$(\mathbf{c}_0, \mathbf{c}_1) = \big(\mathbf{a}_\rho + \mathbf{F}\cdot\mathbf{e},\ \mathbf{b}_\rho + \mathbf{P}^\top\cdot\mathbf{e} + \mu\cdot\lfloor q/2\rfloor + \nu\big) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^t, \qquad (10)$$

which is sent to DB as a re-randomization of $(\mathbf{a}_\rho, \mathbf{b}_\rho + \mu\cdot\lfloor q/2\rfloor)$. Then, U provides an interactive WI argument that $(\mathbf{c}_0, \mathbf{c}_1)$ is a re-randomization of some $(\mathbf{a}_\rho, \mathbf{b}_\rho)$ associated with a policy BP_ρ for which U has a credential $\mathsf{cert}_{\mathsf{U},x}$ for some $\mathbf{x} \in \{0,1\}^\kappa$ such that $\mathsf{BP}_\rho(\mathbf{x}) = 1$. In addition, U demonstrates possession of: (i) a preimage $\mathbf{z}_{\mathsf{BP},\rho} \in [0,4]^\zeta$ of $\mathbf{h}_{\mathsf{BP},\rho} = \mathbf{A}_{\mathrm{HBP}}\cdot\mathbf{z}_{\mathsf{BP},\rho} \in \mathbb{Z}_q^n$; (ii) a credential $\mathsf{Cred}_{\mathsf{U},\mathbf{x}}$ for the corresponding $\mathbf{x} \in \{0,1\}^\kappa$ and the private key $\mathbf{e}_\mathsf{U} \in \{0,1\}^m$ for the pseudonym P_U to which $\mathbf{x}$ is bound; (iii) the coins leading to the randomization of some $(\mathbf{a}_\rho, \mathbf{b}_\rho)$. Then entire step is conducted by arguing knowledge of

$$\begin{cases} \mathbf{e}_\mathsf{U} \in \{0,1\}^m, \mathfrak{m}_{\mathsf{U},\mathbf{x}} \in \{0,1\}^{m_I},\ \mathbf{x} \in \{0,1\}^\kappa,\ \widehat{\mathfrak{m}}_{\mathsf{U},\mathbf{x}} \in \{0,1\}^{m/2} \\ \tau_\mathsf{U} \in \{0,1\}^{\ell_I},\ \mathbf{v}_\mathsf{U} = (\mathbf{v}_{\mathsf{U},1}^\top | \mathbf{v}_{\mathsf{U},2}^\top)^\top \in [-\beta,\beta]^{2m},\ \mathbf{r}_\mathsf{U} \in [-\beta,\beta]^m \\ \qquad\qquad \text{// signature on } \mathfrak{m}_{\mathsf{U},\mathbf{x}} = (\mathsf{vdec}_{n,q-1}(P_\mathsf{U})^\top | \mathbf{x}^\top)^\top \\ \mathbf{z}_{\mathsf{BP},\rho} \in [0,4]^\zeta \qquad \text{// representation of } \mathsf{BP}_\rho \\ \mathfrak{m} \in \{0,1\}^{m_d},\ \tau \in \{0,1\}^\ell,\ \mathbf{v} = (\mathbf{v}_1^\top | \mathbf{v}_2^\top)^\top \in \mathbb{Z}^{2m} \\ \qquad\qquad \text{// signature on } \mathfrak{m} = \mathsf{vdec}_{2n+t,q-1}(\mathbf{a}_i^\top | \mathbf{b}_i^\top | \mathbf{h}_{\mathsf{BP},\rho}^\top)^\top \\ \mathbf{e} \in \{-1,0,1\}^t,\ \mu \in \{0,1\}^t,\ \nu \in [-B,B]^t, \\ \qquad\qquad \text{// coins allowing the re-randomization of } (\mathbf{a}_\rho, \mathbf{b}_\rho) \end{cases}$$

satisfying the relations (modulo q)

$$\begin{cases} \mathbf{H}_{2n+t,q-1}\cdot\mathfrak{m} + \left[\begin{array}{c|c|c|c} \mathbf{F} & & & \\ \hline \mathbf{P}^\top & \mathbf{I}_t\cdot\lfloor q/2\rfloor & \mathbf{I}_t & \\ \hline & & & -\mathbf{A}_{\mathrm{HBP}} \end{array}\right] \cdot \left[\begin{array}{c} \mathbf{e} \\ \hline \mu \\ \hline \nu \\ \hline \mathbf{z}_{\mathsf{BP},\rho} \end{array}\right] = \left[\begin{array}{c} \mathbf{c}_0 \\ \hline \mathbf{c}_1 \\ \hline \mathbf{0}^n \end{array}\right] \\ \qquad\qquad \text{// (recall that } (\mathbf{a}_\rho^\top | \mathbf{b}_\rho^\top | \mathbf{h}_{\mathsf{BP},\rho}^\top)^\top = \mathbf{H}_{2n+t,q-1}\cdot\mathfrak{m}) \\ \mathbf{A}\cdot\mathbf{v}_1 + \mathbf{A}_0\cdot\mathbf{v}_2 + \sum_{j=1}^{\ell}\mathbf{A}_j\cdot(\tau[j]\cdot\mathbf{v}_2) - \mathbf{D}\cdot\mathfrak{m} = \mathbf{u} \\ \mathbf{A}_I\cdot\mathbf{v}_{\mathsf{U},1} + \mathbf{A}_{I,0}\cdot\mathbf{v}_{\mathsf{U},2} + \sum_{j=1}^{\ell_I}\mathbf{A}_{I,j}\cdot(\tau_\mathsf{U}[j]\cdot\mathbf{v}_{\mathsf{U},2}) - \mathbf{D}_I\cdot\widehat{\mathfrak{m}}_{\mathsf{U},\mathbf{x}} = \mathbf{u}_I \\ \mathbf{D}_{I,0}\cdot\mathbf{r}_\mathsf{U} + \mathbf{D}_{I,1}\cdot\mathfrak{m}_{\mathsf{U},\mathbf{x}} - \mathbf{H}_{n,q-1}\cdot\widehat{\mathfrak{m}}_{\mathsf{U},\mathbf{x}} = \mathbf{0} \\ \left[\begin{array}{c|c} \mathbf{H}_{n,q-1} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{I}_\kappa \end{array}\right]\cdot\mathfrak{m}_{\mathsf{U},\mathbf{x}} + \left[\begin{array}{c} -\bar{\mathbf{A}} \\ \mathbf{0} \end{array}\right]\cdot\mathbf{e}_\mathsf{U} + \left[\begin{array}{c} \mathbf{0} \\ -\mathbf{I}_\kappa \end{array}\right]\cdot\mathbf{x} = \mathbf{0} \end{cases} \qquad (11)$$

and such that $\mathbf{z}_{\mathsf{BP},\rho} \in [0,4]^\zeta$ encodes BP_ρ such that $\mathsf{BP}_\rho(\mathbf{x}) = 1$. This is done by running the argument system described in Section 6.6.

3. If the ZK argument of step 2 verifies, DB decrypts $(\mathbf{c}_0, \mathbf{c}_1) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^t$ to obtain $M' = \lfloor(\mathbf{c}_1 - \mathbf{S}^\top\cdot\mathbf{c}_0)/(q/2)\rceil \in \{0,1\}^t$, which is returned to U. Then, DB argues knowledge of $\mathbf{y} = \mathbf{c}_1 - \mathbf{S}^\top\cdot\mathbf{c}_0 - M'\cdot\lfloor q/2\rfloor \in \mathbb{Z}^t$ of norm $\|\mathbf{y}\|_\infty \leq q/5$ and small-norm $\mathbf{E} \in \mathbb{Z}^{m\times t}$, $\mathbf{S} \in \mathbb{Z}^{n\times t}$ satisfying (modulo q)

$$\mathbf{P} = \mathbf{F}^\top\cdot\mathbf{S} + \mathbf{E}\ , \qquad \mathbf{c}_0^\top\cdot\mathbf{S} + \mathbf{y}^\top = \mathbf{c}_1^\top - {M'}^\top\cdot\lfloor q/2\rfloor.$$

To this end, DB uses the ZK argument system of Section 6.4.

4. If the ZK argument produced by DB does not verify, U outputs $\perp$. Otherwise, U recalls the string $\mu \in \{0,1\}^t$ and outputs $M_{\rho_i} = M' \oplus \mu$.

Like our construction of Section 4, the above protocol requires the DB to repeat a ZK proof of communication complexity $\Omega(N)$ with each user U during the initialization phase. By applying the Fiat-Shamir heuristic as shown in the full version of the paper, the cost of the initialization phase can be made independent of the number of users: the sender can publicize $\big(PK_{\mathrm{DB}}, \{ER_i, \mathsf{BP}_i\}_{i=1}^N\big)$ along with a with a universally verifiable non-interactive proof of well-formedness.

The security of the above protocol against static corruptions is proved in the full version of the paper, under the SIS and LWE assumptions.

6 Our Zero-Knowledge Arguments of Knowledge

This section provides all the zero-knowledge arguments of knowledge (ZKAoK) used as building blocks in our two adaptive OT schemes. Our argument systems operate in the framework of Stern's protocol [52], which was originally introduced in the context of code-based cryptography but has been developed [41,42,39,37,38].

In Section 6.1, we first recall Stern's protocol in a generalized, abstract manner suggested in [37]. Then, using various transformations, we will demonstrate that all the required ZKAoKs can be obtained from this abstract protocol. Our basic strategy and techniques are summarized in Section 6.2, while the details of the protocols are given in the next subsections. In particular, our treatment of hidden branching programs in Section 6.6 is rather sophisticated as it requires to handle a number of secret objects nested together via branching programs, commitments, encryptions, signatures and Merkle trees. This protocol introduces new techniques and insights of which we provide the intuition hereafter.

6.1 Abstracting Stern's Protocol

Let K, D, q be positive integers with $D \geq K$ and $q \geq 2$, and let VALID be a subset of $\mathbb{Z}^D$. Suppose that $\mathcal{S}$ is a finite set such that every $\phi \in \mathcal{S}$ can be associated with a permutation Γ_ϕ of D elements satisfying the following conditions:

$$\begin{cases} \mathbf{w} \in \mathsf{VALID} \iff \Gamma_\phi(\mathbf{w}) \in \mathsf{VALID}, \\ \text{If } \mathbf{w} \in \mathsf{VALID} \text{ and } \phi \text{ is uniform in } \mathcal{S}, \text{ then } \Gamma_\phi(\mathbf{w}) \text{ is uniform in } \mathsf{VALID}. \end{cases} \tag{12}$$

We aim to construct a statistical ZKAoK for the following abstract relation:

$$\mathrm{R}_{\mathrm{abstract}} = \big\{ \big((\mathbf{M}, \mathbf{v}), \mathbf{w} \big) \in \mathbb{Z}_q^{K \times D} \times \mathbb{Z}_q^D \times \mathsf{VALID} : \mathbf{M} \cdot \mathbf{w} = \mathbf{v} \bmod q. \big\}$$

Stern's original protocol corresponds to the case $\mathsf{VALID} = \{\mathbf{w} \in \{0,1\}^D : \mathsf{wt}(\mathbf{w}) = k\}$, where $\mathsf{wt}(\cdot)$ denotes the Hamming weight and $k < D$ is a given integer, $\mathcal{S} = \mathcal{S}_D$ is the set of all permutations of D elements and $\Gamma_\phi(\mathbf{w}) = \phi(\mathbf{w})$.

The conditions in (12) play a crucial role in proving in ZK that $\mathbf{w} \in \mathsf{VALID}$. To this end, the prover samples a random $\phi \hookleftarrow U(\mathcal{S})$ and lets the verifier check that $\Gamma_\phi(\mathbf{w}) \in \mathsf{VALID}$ without learning any additional information about $\mathbf{w}$ due to the randomness of ϕ. Furthermore, to prove in a zero-knowledge manner that the linear equation is satisfied, the prover samples a masking vector $\mathbf{r}_w \hookleftarrow U(\mathbb{Z}_q^D)$, and convinces the verifier instead that $\mathbf{M} \cdot (\mathbf{w} + \mathbf{r}_w) = \mathbf{M} \cdot \mathbf{r}_w + \mathbf{v} \bmod q$.

The interaction between prover $\mathcal{P}$ and verifier $\mathcal{V}$ is described in Figure 1. The protocol uses a statistically hiding and computationally binding string commitment scheme COM (e.g., the SIS-based scheme from [35]).

1. **Commitment:** Prover samples $\mathbf{r}_w \leftarrow U(\mathbb{Z}_q^D)$, $\phi \leftarrow U(\mathcal{S})$ and randomness ρ_1, ρ_2, ρ_3 for COM. Then he sends $\mathrm{CMT} = \big(C_1, C_2, C_3\big)$ to the verifier, where

$$C_1 = \mathsf{COM}(\phi, \mathbf{M} \cdot \mathbf{r}_w \bmod q; \rho_1), \;\; C_2 = \mathsf{COM}(\Gamma_\phi(\mathbf{r}_w); \rho_2),$$
$$C_3 = \mathsf{COM}(\Gamma_\phi(\mathbf{w} + \mathbf{r}_w \bmod q); \rho_3).$$

2. **Challenge:** The verifier sends a challenge $Ch \leftarrow U(\{1, 2, 3\})$ to the prover.
3. **Response:** Depending on Ch, the prover sends RSP computed as follows:
   - $Ch = 1$: Let $\mathbf{t}_w = \Gamma_\phi(\mathbf{w})$, $\mathbf{t}_r = \Gamma_\phi(\mathbf{r}_w)$, and $\mathrm{RSP} = (\mathbf{t}_w, \mathbf{t}_r, \rho_2, \rho_3)$.
   - $Ch = 2$: Let $\phi_2 = \phi$, $\mathbf{w}_2 = \mathbf{w} + \mathbf{r}_w \bmod q$, and $\mathrm{RSP} = (\phi_2, \mathbf{w}_2, \rho_1, \rho_3)$.
   - $Ch = 3$: Let $\phi_3 = \phi$, $\mathbf{w}_3 = \mathbf{r}_w$, and $\mathrm{RSP} = (\phi_3, \mathbf{w}_3, \rho_1, \rho_2)$.

Verification: Receiving RSP, the verifier proceeds as follows:

- $Ch = 1$: Check that $\mathbf{t}_w \in \mathsf{VALID}$, $C_2 = \mathsf{COM}(\mathbf{t}_r; \rho_2)$, $C_3 = \mathsf{COM}(\mathbf{t}_w + \mathbf{t}_r \bmod q; \rho_3)$.
- $Ch = 2$: Check that $C_1 = \mathsf{COM}(\phi_2, \mathbf{M} \cdot \mathbf{w}_2 - \mathbf{v} \bmod q; \rho_1)$, $C_3 = \mathsf{COM}(\Gamma_{\phi_2}(\mathbf{w}_2); \rho_3)$.
- $Ch = 3$: Check that $C_1 = \mathsf{COM}(\phi_3, \mathbf{M} \cdot \mathbf{w}_3; \rho_1)$, $\;C_2 = \mathsf{COM}(\Gamma_{\phi_3}(\mathbf{w}_3); \rho_2)$.

In each case, the verifier outputs 1 if and only if all the conditions hold.

Figure 1: Stern-like ZKAoK for the relation $\mathrm{R}_{\mathrm{abstract}}$.

Theorem 2. *The protocol in Figure 1 is a statistical* ZKAoK *with perfect completeness, soundness error* $2/3$*, and communication cost* $\mathcal{O}(D \log q)$*. Namely:*

- *There exists a polynomial-time simulator that, on input* $(\mathbf{M}, \mathbf{v})$*, outputs an accepted transcript statistically close to that produced by the real prover.*
- *There exists a polynomial-time knowledge extractor that, on input a commitment* CMT *and* 3 *valid responses* $(\mathrm{RSP}_1, \mathrm{RSP}_2, \mathrm{RSP}_3)$ *to all* 3 *possible values of the challenge* Ch*, outputs* $\mathbf{w}' \in \mathsf{VALID}$ *such that* $\mathbf{M} \cdot \mathbf{w}' = \mathbf{v} \bmod q$*.*

The proof of the theorem relies on standard simulation and extraction techniques for Stern-like protocols [35,41,37]. It is given in the full version of the paper.

6.2 Our Strategy and Basic Techniques, In a Nutshell

Before going into the details of our protocols, we first summarize our governing strategy and the techniques that will be used in the next subsections.

In each protocol, we prove knowledge of (possibly one-dimensional) integer vectors $\{\mathbf{w}_i\}_i$ that have various constraints (e.g., smallness, special arrangements of coordinates, or correlation with one another) and satisfy a system

$$\left\{ \sum_i \mathbf{M}_{i,j} \cdot \mathbf{w}_i = \mathbf{v}_j \right\}_j, \tag{13}$$

where $\{\mathbf{M}_{i,j}\}_{i,j}$, $\{\mathbf{v}_j\}_j$ are public matrices (which are possibly zero or identity matrices) and vectors. Our strategy consists in transforming this entire system into one equivalent equation $\mathbf{M} \cdot \mathbf{w} = \mathbf{v}$, where matrix $\mathbf{M}$ and vector $\mathbf{v}$ are public, while the constraints of the secret vector $\mathbf{w}$ capture those of witnesses $\{\mathbf{w}_i\}_i$ and they are provable in zero-knowledge via random permutations. For this purpose, the Stern-like protocol from Section 6.1 comes in handy.

A typical transformation step is of the form $\mathbf{w}_i \to \bar{\mathbf{w}}_i$, where there exists public matrix $\mathbf{P}_{i,j}$ such that $\mathbf{P}_{i,j} \cdot \bar{\mathbf{w}}_i = \mathbf{w}_i$. This subsumes the decomposition and extension mechanisms which first appeared in [41].

- **Decomposition:** Used when $\mathbf{w}_i$ has infinity norm bound larger than 1 and we want to work more conveniently with $\bar{\mathbf{w}}_i$ whose norm bound is exactly 1. In this case, $\mathbf{P}_{i,j}$ is a decomposition matrix (see Section 2.5).
- **Extension:** Used when we insert "dummy" coordinates to $\mathbf{w}_i$ to obtain $\bar{\mathbf{w}}_i$ whose coordinates are somewhat balanced. In this case, $\mathbf{P}_{i,j}$ is a $\{0,1\}$-matrix with zero-columns corresponding to positions of insertions.

Such a step transforms the term $\mathbf{M}_{i,j} \cdot \mathbf{w}_i$ into $\overline{\mathbf{M}}_{i,j} \cdot \bar{\mathbf{w}}_i$, where $\overline{\mathbf{M}}_{i,j} = \mathbf{M}_{i,j} \cdot \mathbf{P}_{i,j}$ is a public matrix. Also, using the commutativity property of addition, we often group together secret vectors having the same constraints.

After a number of transformations, we will reach a system equivalent to (13):

$$\begin{cases} \mathbf{M}'_{1,1} \cdot \mathbf{w}'_1 + \mathbf{M}'_{1,2} \cdot \mathbf{w}'_2 + \cdots + \mathbf{M}'_{1,k} \cdot \mathbf{w}'_k = \mathbf{v}_1, \\ \qquad\qquad\qquad\qquad \vdots \\ \mathbf{M}'_{t,1} \cdot \mathbf{w}'_1 + \mathbf{M}'_{t,2} \cdot \mathbf{w}'_2 + \cdots + \mathbf{M}'_{t,k} \cdot \mathbf{w}'_k = \mathbf{v}_t, \end{cases} \tag{14}$$

where integers t, k and matrices $\mathbf{M}'_{i,j}$ are public. Defining

$$\mathbf{M} = \left(\begin{array}{c|c|c|c} \mathbf{M}'_{1,1} & \mathbf{M}'_{1,2} & \cdots & \mathbf{M}'_{1,k} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline \mathbf{M}'_{t,1} & \mathbf{M}'_{t,2} & \cdots & \mathbf{M}'_{t,k} \end{array}\right); \quad \mathbf{w} = \begin{pmatrix} \mathbf{w}'_1 \\ \vdots \\ \mathbf{w}'_k \end{pmatrix}; \quad \mathbf{v} = \begin{pmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_t \end{pmatrix},$$

we obtain the unified equation $\mathbf{M} \cdot \mathbf{w} = \mathbf{v} \bmod q$. At this stage, we will use a properly defined composition of random permutations to prove the constraints of $\mathbf{w}$. We remark that the crucial aspect of the above process is in fact the manipulation of witness vectors, while the transformations of public matrices/vectors just follow accordingly. To ease the presentation of the next subsections, we will thus only focus on the secret vectors.

In the process, we will employ various extending and permuting techniques which require introducing some notations. The most frequently used ones are given in Table 1. Some of these techniques appeared (in slightly different forms) in previous works [41,42,39,37,38]. The last three parts of the table summarizes newly-introduced techniques that will enable the treatment of secret-and-correlated objects involved in the evaluation of hidden branching programs.

In particular, the technique of the last row will be used for proving knowledge of an integer $z = x \cdot y$ for some $(x, y) \in [0, 4] \times \{0, 1\}$ satisfying other relations.

6.3 Protocol 1

Let n, m, q, N, t, B_χ be the parameters defined in Section 4. The protocol allows the prover to prove knowledge of LWE secrets and the well-formedness of ciphertexts. It is summarized as follows.

Common input: $\mathbf{F} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{P} \in \mathbb{Z}_q^{m \times t}$; $\{\mathbf{a}_i \in \mathbb{Z}_q^n,\ \mathbf{b}_i \in \mathbb{Z}_q^t\}_{i=1}^N$.

Prover's goal is to prove knowledge of $\mathbf{S} \in [-B_\chi, B_\chi]^{n \times t}$, $\mathbf{E} \in [-B_\chi, B_\chi]^{m \times t}$, $\{\mathbf{x}_i \in [-B_\chi, B_\chi]^t, M_i \in \{0,1\}^t\}_{i=1}^N$ such that the following equations hold:

$$\begin{cases} \mathbf{F}^\top \cdot \mathbf{S} + \mathbf{E} = \mathbf{P} \bmod q \\ \forall i \in [N] : \mathbf{S}^\top \cdot \mathbf{a}_i + \mathbf{x}_i + \lfloor q/2 \rfloor \cdot M_i = \mathbf{b}_i \bmod q. \end{cases} \tag{15}$$

For each $j \in [t]$, let $\mathbf{p}_j, \mathbf{s}_j, \mathbf{e}_j$ be the j-th column of matrices $\mathbf{P}, \mathbf{S}, \mathbf{E}$, respectively. For each $(i, j) \in [N] \times [t]$, let $\mathbf{b}_i[j], \mathbf{x}_i[j], M_i[j]$ denote the j-th coordinate of vectors $\mathbf{b}_i, \mathbf{x}_i, M_i$, respectively. Then, observe that (15) can be rewritten as:

$$\begin{cases} \forall j \in [t] : \mathbf{F}^\top \cdot \mathbf{s}_j + \mathbf{I}_m \cdot \mathbf{e}_j = \mathbf{p}_j \bmod q \\ \forall (i, j) \in [N] \times [t] : \mathbf{a}_i^\top \cdot \mathbf{s}_j + 1 \cdot \mathbf{x}_i[j] + \lfloor q/2 \rfloor \cdot M_i[j] = \mathbf{b}_i[j] \bmod q. \end{cases} \tag{16}$$

Then, we form the following vectors:

$$\mathbf{w}_1 = \left(\mathbf{s}_1^\top \mid \ldots \mid \mathbf{s}_t^\top \mid \mathbf{e}_1^\top \mid \ldots \mid \mathbf{e}_t^\top \mid (\mathbf{x}_1[1], \ldots, \mathbf{x}_N[t])\right)^\top \in [-B_\chi, B_\chi]^{(n+m+N)t};$$
$$\mathbf{w}_2 = (M_1[1], \ldots, M_N[t])^\top \in \{0,1\}^{Nt}.$$

Next, we run $\mathsf{vdec}'_{(n+m+N)t, B_\chi}$ to decompose $\mathbf{w}_1$ into $\bar{\mathbf{w}}_1$ and then extend $\bar{\mathbf{w}}_1$ to $\mathbf{w}_1^* \in \mathsf{B}^3_{(n+m+N)t\delta_{B_\chi}}$. We also extend $\mathbf{w}_2$ into $\mathbf{w}_2^* \in \mathsf{B}^2_{Nt}$ and we then form $\mathbf{w} = ((\mathbf{w}_1^*)^\top \mid (\mathbf{w}_2^*)^\top)^\top \in \{-1, 0, 1\}^D$, where $D = 3(n + m + N)t\delta_{B_\chi} + 2Nt$.

Observe that relations (16) can be transformed into *one* equivalent equation of the form $\mathbf{M} \cdot \mathbf{w} = \mathbf{v} \bmod q$, where $\mathbf{M}$ and $\mathbf{v}$ are built from the common input.

Having performed the above unification, we now define VALID as the set of all vectors $\mathbf{t} = (\mathbf{t}_1^\top \mid \mathbf{t}_2^\top)^\top \in \{-1, 0, 1\}^D$, where $\mathbf{t}_1 \in \mathsf{B}^3_{(n+m+N)t\delta_{B_\chi}}$ and $\mathbf{t}_2 \in \mathsf{B}^2_{Nt}$. Clearly, our vector $\mathbf{w}$ belongs to the set VALID.

Next, we specify the set $\mathcal{S}$ and permutations of D elements $\{\Gamma_\phi : \phi \in \mathcal{S}\}$, for which the conditions in (12) hold.

- $\mathcal{S} := \mathcal{S}_{3(n+m+N)t\delta_{B_\chi}} \times \mathcal{S}_{2Nt}$.

Notation	Meaning/Property/Usage/Technique
$\mathsf{B}^2_{\mathtt{m}}$	– The set of vectors in $\{0,1\}^{2\mathtt{m}}$ with Hamming weight $\mathtt{m}$. – $\forall \phi \in \mathcal{S}_{2\mathtt{m}}, \mathbf{x}' \in \mathbb{Z}^{2\mathtt{m}} : \mathbf{x}' \in \mathsf{B}^2_{\mathtt{m}} \Leftrightarrow \phi(\mathbf{x}') \in \mathsf{B}^2_{\mathtt{m}}$. – To prove $\mathbf{x} \in \{0,1\}^{\mathtt{m}}$: Extend $\mathbf{x}$ to $\mathbf{x}' \in \mathsf{B}^2_{\mathtt{m}}$, then permute $\mathbf{x}'$.
$\mathsf{B}^3_{\mathtt{m}}$	– The set of vectors in $\{-1,0,1\}^{3\mathtt{m}}$ that have exactly $\mathtt{m}$ coordinates equal to j, for every $j \in \{-1,0,1\}$. – $\forall \phi \in \mathcal{S}_{3\mathtt{m}}, \mathbf{x}' \in \mathbb{Z}^{3\mathtt{m}} : \mathbf{x}' \in \mathsf{B}^3_{\mathtt{m}} \Leftrightarrow \phi(\mathbf{x}') \in \mathsf{B}^3_{\mathtt{m}}$. – To prove $\mathbf{x} \in \{-1,0,1\}^{\mathtt{m}}$: Extend $\mathbf{x}$ to $\mathbf{x}' \in \mathsf{B}^3_{\mathtt{m}}$, then permute $\mathbf{x}'$.
$\mathsf{ext}_2(\cdot)$ and $T_2[\cdot](\cdot)$	– For $c \in \{0,1\} : \mathsf{ext}_2(c) = (\bar{c}, c)^\top \in \{0,1\}^2$. – For $b \in \{0,1\}$ and $\mathbf{x} = (x_0, x_1)^\top \in \mathbb{Z}^2$: $T_2[b](\mathbf{x}) = (x_b, x_{\bar{b}})^\top$. – Property: $\mathbf{x} = \mathsf{ext}_2(c) \Leftrightarrow T_2[b](\mathbf{x}) = \mathsf{ext}_2(c \oplus b)$. – To prove $c \in \{0,1\}$ simultaneously satisfies many relations: Extend it to $\mathbf{x} = \mathsf{ext}_2(c)$, then permute and use the *same* b at all appearances.
$\mathsf{expand}(\cdot,\cdot)$ and $T_{\mathsf{exp}}[\cdot,\cdot](\cdot)$	– For $c \in \{0,1\}$ and $\mathbf{x} \in \mathbb{Z}^{\mathtt{m}}$: $\mathsf{expand}(c, \mathbf{x}) = (\bar{c} \cdot \mathbf{x}^\top \mid c \cdot \mathbf{x}^\top)^\top \in \mathbb{Z}^{2\mathtt{m}}$. – For $b \in \{0,1\}, \phi \in \mathcal{S}_m$, $\mathbf{v} = \begin{pmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \end{pmatrix} \in \mathbb{Z}^{2\mathtt{m}}$: $T_{\mathsf{exp}}[b,\phi](\mathbf{v}) = \begin{pmatrix} \phi(\mathbf{v}_b) \\ \phi(\mathbf{v}_{\bar{b}}) \end{pmatrix}$. – Property: $\mathbf{v} = \mathsf{expand}(c, \mathbf{x}) \Leftrightarrow T_{\mathsf{exp}}[b,\phi](\mathbf{v}) = \mathsf{expand}(c \oplus b, \phi(\mathbf{x}))$.
$[\cdot]_5$	For $k \in \mathbb{Z}$: $[k]_5$ denotes the integer $t \in \{0,1,2,3,4\}$, s.t. $t = k \bmod 5$.
$\mathsf{ext}_5(\cdot)$ and $T_5[\cdot](\cdot)$	– For $x \in [0,4] : \mathsf{ext}_5(x) = ([x+4]_5, [x+3]_5, [x+2]_5, [x+1]_5, x)^\top \in [0,4]^5$. – For $c \in [0,4]$ and $\mathbf{v} = (v_0, v_1, v_2, v_3, v_4)^\top \in \mathbb{Z}^5$: $T_5[c](\mathbf{v}) = \left(v_{[-c]_5}, v_{[-c+1]_5}, v_{[-c+2]_5}, v_{[-c+3]_5}, v_{[-c+4]_5}\right)^\top$. – Property: $\mathbf{v} = \mathsf{ext}_5(x) \Leftrightarrow T_5[c](\mathbf{v}) = \mathsf{ext}_5(x + c \bmod 5)$. – To prove $x \in [0,4]$ simultaneously satisfies many relations: Extend it to $\mathbf{v} = \mathsf{ext}_5(x)$, then permute and use the *same* c at all appearances.
unit_x	– $\forall x \in [0,4]$: unit_x is the 5-dim unit vector $(v_0, \ldots, v_4)^\top$ with $v_x = 1$. – For $c \in [0,4], \mathbf{v} \in \mathbb{Z}^5$: $\mathbf{v} = \mathsf{unit}_x \Leftrightarrow T_5[c](\mathbf{v}) = \mathsf{unit}_{x+c \bmod 5}$. $\rightarrow$ Allow proving $\mathbf{v} = \mathsf{unit}_x$ for some $x \in [0,4]$ satisfying other relations.
$\mathsf{ext}_{5\times 2}(\cdot,\cdot)$ and $T_{5\times 2}[\cdot,\cdot](\cdot)$	– For $x \in [0,4]$ and $y \in \{0,1\}$: $\mathsf{ext}_{5\times 2}(x,y) = ([x+4]_5 \cdot \bar{y}, [x+4]_5 \cdot y, [x+3]_5 \cdot \bar{y}, [x+3]_5 \cdot y, [x+2]_5 \cdot \bar{y}, [x+2]_5 \cdot y, [x+1]_5 \cdot \bar{y}, [x+1]_5 \cdot y, x \cdot \bar{y}, x \cdot y)^\top \in [0,4]^{10}$ – For $(c,b) \in [0,4] \times \{0,1\}$ and $\mathbf{v} = (v_{0,0}, v_{0,1}, \ldots, v_{4,0}, v_{4,1})^\top \in \mathbb{Z}^{10}$: $T_{5\times 2}[c,b](\mathbf{v}) = \big(v_{[-c]_5,b}, v_{[-c]_5,\bar{b}}, v_{[-c+1]_5,b}, v_{[-c+1]_5,\bar{b}}, v_{[-c+2]_5,b}, v_{[-c+2]_5,\bar{b}}, v_{[-c+3]_5,b}, v_{[-c+3]_5,\bar{b}}, v_{[-c+4]_5,b}, v_{[-c+4]_5,\bar{b}}\big)^\top$. – Property: $\mathbf{v} = \mathsf{ext}_{5\times 2}(x,y) \Leftrightarrow T_{5\times 2}[c,b](\mathbf{v}) = \mathsf{ext}_{5\times 2}(x+c \bmod 5, y \oplus b)$. $\rightarrow$ Allow proving $z = x \cdot y$ for some $(x,y) \in [0,4] \times \{0,1\}$ satisfying other relations: Extend z to $\mathbf{v} = \mathsf{ext}_{5\times 2}(x,y)$, then permute and use the *same* c, b at all appearances of x, y, respectively.

Table 1. Basic notations and extending/permuting techniques used in our protocols.

– For $\phi = (\phi_1, \phi_2) \in \mathcal{S}$ and for $\mathbf{t} = (\mathbf{t}_1^\top \mid \mathbf{t}_2^\top)^\top \in \mathbb{Z}^D$, where $\mathbf{t}_1 \in \mathbb{Z}^{3(n+m+N)t\delta_{B_\chi}}$ and $\mathbf{t}_2 \in \mathbb{Z}^{2Nt}$, we define $\Gamma_\phi(\mathbf{t}) = (\phi_1(\mathbf{t}_1)^\top \mid \phi_2(\mathbf{t}_2)^\top)^\top$.

By inspection, it can be seen that the desired properties in (12) are satisfied. As a result, we can obtain the required ZKAoK by running the protocol from Section 6.1 with common input $(\mathbf{M}, \mathbf{v})$ and prover's input $\mathbf{w}$. The protocol has communication cost $\mathcal{O}(D \log q) = \widetilde{\mathcal{O}}(\lambda) \cdot \mathcal{O}(Nt)$ bits.

While this protocol has linear complexity in N, it is only used in the initialization phase, where $\Omega(N)$ bits inevitably have to be transmitted anyway.

6.4 Protocol 2

Let n, m, q, N, t, B be system parameters. The protocol allows the prover to prove knowledge of LWE secrets and the correctness of decryption.

Common input: $\mathbf{F} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{P} \in \mathbb{Z}_q^{m\times t}$; $\mathbf{c}_0 \in \mathbb{Z}_q^n$, $\mathbf{c}_1 \in \mathbb{Z}_q^t$, $M' \in \{0,1\}^t$.

Prover's goal is to prove knowledge of $\mathbf{S} \in [-B_\chi, B_\chi]^{n\times t}$, $\mathbf{E} \in [-B_\chi, B_\chi]^{m\times t}$ and $\mathbf{y} \in [-q/5, q/5]^t$ such that the following equations hold:

$$\mathbf{F}^\top \cdot \mathbf{S} + \mathbf{E} = \mathbf{P} \bmod q; \quad \mathbf{c}_0^\top \cdot \mathbf{S} + \mathbf{y}^\top = \mathbf{c}_1^\top - M'^\top \cdot \lfloor q/2 \rfloor \bmod q. \tag{17}$$

For each $j \in [t]$, let $\mathbf{p}_j, \mathbf{s}_j, \mathbf{e}_j$ be the j-th column of matrices $\mathbf{P}, \mathbf{S}, \mathbf{E}$, respectively; and let $\mathbf{y}[j], \mathbf{c}_1[j], M'[j]$ be the j-th entry of vectors $\mathbf{y}, \mathbf{c}_1, M'$, respectively. Then, observe that (17) can be re-written as:

$$\forall j \in [t] : \begin{cases} \mathbf{F}^\top \cdot \mathbf{s}_j + \mathbf{I}_m \cdot \mathbf{e}_j = \mathbf{p}_j \bmod q \\ \mathbf{c}_0^\top \cdot \mathbf{s}_j + 1 \cdot \mathbf{y}[j] = \mathbf{c}_1[j] - M'[j] \cdot \lfloor q/2 \rfloor \bmod q. \end{cases} \tag{18}$$

Next, we form vector $\mathbf{w}_1 = (\mathbf{s}_1^\top \mid \ldots \mid \mathbf{s}_t^\top \mid \mathbf{e}_1^\top \mid \ldots \mid \mathbf{e}_t^\top)^\top \in [-B_\chi, B_\chi]^{(n+m)t}$, then decompose it to $\bar{\mathbf{w}}_1 \in \{-1,0,1\}^{(n+m)t\delta_{B_\chi}}$, and extend $\bar{\mathbf{w}}_1$ to $\mathbf{w}_1^* \in \mathsf{B}^3_{(n+m)t\delta_{B_\chi}}$.

At the same time, we decompose vector $\mathbf{y} = (\mathbf{y}[1], \ldots, \mathbf{y}[t])^\top \in [-q/5, q/5]^t$ to $\bar{\mathbf{y}} \in \{-1,0,1\}^{t\delta_{q/5}}$, and then extend $\bar{\mathbf{y}}$ to $\mathbf{y}^* \in \mathsf{B}^3_{t\delta_{q/5}}$.

Defining the ternary vector $\mathbf{w} = ((\mathbf{w}_1^*)^\top \mid (\mathbf{y}^*)^\top)^\top \in \{-1,0,1\}^D$ of dimension $D = 3(n+m)t\delta_{B_\chi} + 3t\delta_{q/5}$, we finally obtain the equation $\mathbf{M} \cdot \mathbf{w} = \mathbf{v} \bmod q$, for public matrix $\mathbf{M}$ and public vector $\mathbf{v}$. Using similar arguments as in Section 6.3, we can obtain the desired zero-knowledge argument system. The protocol has communication cost $\mathcal{O}(D \log q) = \widetilde{\mathcal{O}}(\lambda) \cdot \mathcal{O}(t)$ bits.

6.5 Protocol 3

Let n, m, m_d, q, t, ℓ, B be the parameters defined in Section 4. The protocol allows the prover to argue that a given ciphertext is a correct randomization of some hidden ciphertext and that he knows a valid signature on that ciphertext. Let β be the infinity norm bound of these valid signatures.

Common input: It consists of matrices $\mathbf{F} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{P} \in \mathbb{Z}_q^{m\times t}$, $\mathbf{A}, \mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_\ell \in \mathbb{Z}_q^{n\times m}$, $\mathbf{D} \in \mathbb{Z}_q^{n\times m_d}$ and vectors $\mathbf{c}_0 \in \mathbb{Z}_q^n$, $\mathbf{c}_1 \in \mathbb{Z}_q^t$, $\mathbf{u} \in \mathbb{Z}_q^n$.

Prover's goal is to prove knowledge of $\mathfrak{m} \in \{0,1\}^{m_d}$, $\mu \in \{0,1\}^t$, $\mathbf{e} \in \{-1,0,1\}^t$, $\nu \in [-B,B]^t$, $\tau = (\tau[1],\ldots,\tau[\ell])^\top \in \{0,1\}^\ell$, $\mathbf{v}_1, \mathbf{v}_2 \in [-\beta,\beta]^m$ such that the following equations hold:

$$\begin{cases} \mathbf{A}\cdot\mathbf{v}_1 + \mathbf{A}_0\cdot\mathbf{v}_2 + \sum_{j=1}^{\ell}\mathbf{A}_j\cdot(\tau[j]\cdot\mathbf{v}_2) - \mathbf{D}\cdot\mathfrak{m} = \mathbf{u} \bmod q; \\ \mathbf{H}_{n+t,q-1}\cdot\mathfrak{m} + \begin{pmatrix}\mathbf{F}\\ \mathbf{P}^\top\end{pmatrix}\cdot\mathbf{e} + \begin{pmatrix}\mathbf{0}^{n\times t}\\ \lfloor\frac{q}{2}\rfloor\cdot\mathbf{I}_t\end{pmatrix}\cdot\mu + \begin{pmatrix}\mathbf{0}^{n\times t}\\ \mathbf{I}_t\end{pmatrix}\cdot\nu = \begin{pmatrix}\mathbf{c}_0\\ \mathbf{c}_1\end{pmatrix} \bmod q. \end{cases} \tag{19}$$

For this purpose, we perform the following transformations on the witnesses.

Decompositions. Decompose vectors $\mathbf{v}_1, \mathbf{v}_2, \nu$ to vectors $\bar{\mathbf{v}}_1 \in \{-1,0,1\}^{m\delta_\beta}$, $\bar{\mathbf{v}}_2 \in \{-1,0,1\}^{m\delta_\beta}$, $\bar{\nu} \in \{-1,0,1\}^{t\delta_B}$, respectively.

Extensions/Combinations.

- Let $\mathbf{w}_1 = (\mathfrak{m}^\top \mid \mu^\top)^\top \in \{0,1\}^{m_d+t}$ and extend it into $\mathbf{w}_1^* \in \mathsf{B}^2_{m_d+t}$.
- Let $\mathbf{w}_2 = (\bar{\mathbf{v}}_1^\top \mid \bar{\nu}^\top \mid \mathbf{e}^\top)^\top \in \{-1,0,1\}^{m\delta_\beta+t\delta_B+t}$ and extend it into the vector $\mathbf{w}_2^* \in \mathsf{B}^3_{m\delta_\beta+t\delta_B+t}$.
- Extend $\bar{\mathbf{v}}_2$ into $\mathbf{s}_0 \in \mathsf{B}^3_{m\delta_\beta}$. Then, for each $j \in [\ell]$, define $\mathbf{s}_j = \mathsf{expand}(\tau[j], \mathbf{s}_0)$. (We refer to Table 1 for details about $\mathsf{expand}(\cdot,\cdot)$.)

Now, we form vector $\mathbf{w} = \left(\mathbf{w}_1^{*\top} \mid \mathbf{w}_2^{*\top} \mid \mathbf{s}_0^\top \mid \mathbf{s}_1^\top \mid \ldots \mid \mathbf{s}_\ell^\top\right)^\top \in \{-1,0,1\}^D$, where $D = (2\ell+2)3m\delta_\beta + 3t\delta_B + 3t + 2(m_d+t)$. At this point, we observe that the equations in (19) can be equivalently transformed into $\mathbf{M}\cdot\mathbf{w} = \mathbf{v} \bmod q$, where the matrix $\mathbf{M}$ and the vector $\mathbf{v}$ are built from the public input.

Having performed the above transformations, we now define VALID as the set of all vectors $\mathbf{t} = (\mathbf{t}_1^\top \mid \mathbf{t}_2^\top \mid \mathbf{t}_{3,0}^\top \mid \mathbf{t}_{3,1}^\top \mid \ldots \mid \mathbf{t}_{3,\ell}^\top)^\top \in \{-1,0,1\}^D$ for which there exists $\tau = (\tau[1],\ldots,\tau[\ell])^\top \in \{0,1\}^\ell$ such that:

$$\mathbf{t}_1 \in \mathsf{B}^2_{m_d+t};\ \mathbf{t}_2 \in \mathsf{B}^3_{m\delta_\beta+t\delta_B+t};\ \mathbf{t}_{3,0} \in \mathsf{B}^3_{m\delta_\beta};\ \forall j \in [\ell] : \mathbf{t}_{3,j} = \mathsf{expand}(\tau[j], \mathbf{t}_{3,0}).$$

It can be seen that $\mathbf{w}$ belongs to this tailored set. Now, let us specify the set $\mathcal{S}$ and permutations of D elements $\{\Gamma_\phi : \phi \in \mathcal{S}\}$ satisfying the conditions in (12).

- $\mathcal{S} := \mathcal{S}_{2(m_d+t)} \times \mathcal{S}_{3(m\delta_\beta+t\delta_B+t)} \times \mathcal{S}_{3m\delta_\beta} \times \{0,1\}^\ell$.
- For $\phi = \left(\phi_1, \phi_2, \phi_3, (b[1],\ldots,b[\ell])^\top\right) \in \mathcal{S}$, we define the permutation Γ_ϕ that transforms vector $\mathbf{t} = (\mathbf{t}_1^\top \mid \mathbf{t}_2^\top \mid \mathbf{t}_{3,0}^\top \mid \mathbf{t}_{3,1}^T \mid \ldots \mid \mathbf{t}_{3,\ell}^\top)^\top \in \mathbb{Z}^D$ as follows:

$$\Gamma_\phi(\mathbf{t}) = \big(\phi_1(\mathbf{t}_1)^\top \mid \phi_2(\mathbf{t}_2)^\top \mid \phi_3(\mathbf{t}_{3,0})^\top \mid T_{\mathsf{exp}}[b[1],\phi_3](\mathbf{t}_{3,1})^\top \mid \ldots \mid T_{\mathsf{exp}}[b[\ell],\phi_3](\mathbf{t}_{3,\ell})^\top\big)^\top.$$

By inspection, it can be seen that the properties in (12) are indeed satisfied. As a result, we can obtain the required argument of knowledge by running the protocol from Section 6.1 with common input $(\mathbf{M},\mathbf{v})$ and prover's input $\mathbf{w}$. The protocol has communication cost $\mathcal{O}(D\log q) = \widetilde{\mathcal{O}}(\lambda)\cdot\mathcal{O}(\log N + t)$ bits.

6.6 Protocol 4: A Treatment of Hidden Branching Programs

We now present the proof system run by the user in the OT-AC system of Section 5. It allows arguing knowledge of an input $\mathbf{x} = (x_0, \ldots, x_{\kappa-1})^\top \in \{0,1\}^\kappa$ satisfying a hidden branching program $\mathsf{BP} = \{(\mathrm{var}(\theta), \pi_{\theta,0}, \pi_{\theta,1})\}_{\theta=1}^{L}$ of length for $L \in \mathsf{poly}(\lambda)$. The prover should additionally demonstrate that: (i) He has a valid credential for $\mathbf{x}$; (ii) The hashed encoding of BP is associated with some hidden ciphertext of the database (and he knows a signature guaranteeing this link); (iii) A given ciphertext is a re-randomization of that hidden ciphertext.

Recall that, at each step $\theta \in [L]$ of the evaluation of $\mathsf{BP}(\mathbf{x})$, we have to look up the value $x_{\mathrm{var}(\theta)}$ in $\mathbf{x} = (x_0, \ldots, x_{\kappa-1})^\top$ to compute the θ-th state η_θ as per

$$\eta_\theta = \pi_{\theta, x_{\mathrm{var}(\theta)}}(\eta_{\theta-1}) = \pi_{\theta,0}(\eta_{\theta-1}) \cdot \bar{x}_{\mathrm{var}(\theta)} + \pi_{\theta,1}(\eta_{\theta-1}) \cdot x_{\mathrm{var}(\theta)}. \qquad (20)$$

To prove that each step is done correctly, it is necessary to provide evidence that the corresponding search is honestly carried out without revealing $x_{\mathrm{var}(\theta)}$, $\mathrm{var}(\theta)$ nor $\{\pi_{\theta,b}\}_{b=0}^{1}$. To this end, a first idea is to perform a simple left-to-right search on $(x_0, \ldots, x_{\kappa-1})$: namely, (20) is expressed in terms of a matrix-vector relation where η_θ is encoded as a unit vector of dimension 5; $\{\pi_{\theta,b}\}_{b=0}^{1}$ are represented as permutation matrices; and $\mathbf{x}_{\mathrm{var}(\theta)} = \mathbf{M}_{\mathrm{var}(\theta)} \cdot \mathbf{x}$ is computed using a matrix $\mathbf{M}_{\mathrm{var}(\theta)} \in \{0,1\}^{\kappa \times \kappa}$ containing exactly one 1 per row. While this approach can be handled using proofs for matrix-vector relations using the techniques of [38], the expected complexity is $\mathcal{O}(\kappa)$ for each step, so that the total complexity becomes $\mathcal{O}(L\kappa)$. Fortunately, a better complexity can be achieved.

If we instead perform a dichotomic search on $\mathbf{x} = (x_0, \ldots, x_{\kappa-1})^\top$, we can reduce the complexity of each step to $\mathcal{O}(\log \kappa)$. To this end, we need to prove a statement "I performed a correct dichotomic search on my secret array $\mathbf{x}$".

In order to solve this problem, we will employ two existing lattice-based tools:

(i) A variant of the SIS-based computationally binding and statistically hiding commitment scheme from [35], which allows to commit to one-bit messages;
(ii) The SIS-based Merkle hash tree proposed in [39].

Let $\bar{\mathbf{A}} \hookleftarrow U(\mathbb{Z}_q^{n \times m})$ and $\mathbf{a}_{\mathsf{com}} \hookleftarrow U(\mathbb{Z}_q^n)$. For each $i \in [0, \kappa-1]$, we let the receiver commit to $x_i \in \{0,1\}$ as $\mathsf{com}_i = \mathbf{a}_{\mathsf{com}} \cdot x_i + \bar{\mathbf{A}} \cdot \mathbf{r}_{\mathsf{com},i}$, with $\mathbf{r}_{\mathsf{com},i} \hookleftarrow U(\{0,1\}^m)$, and reveal $\mathsf{com}_1, \ldots, \mathsf{com}_{\kappa-1}$ to the sender. We build a Merkle tree of depth $\delta_\kappa = \lceil \log \kappa \rceil$ on top of the leaves $\mathsf{com}_0, \ldots, \mathsf{com}_{\kappa-1}$ using the SIS-based hash function $h_{\bar{\mathbf{A}}} : \{0,1\}^{n\lceil \log q \rceil} \times \{0,1\}^{n\lceil \log q \rceil} \to \{0,1\}^{n\lceil \log q \rceil}$ of [39]. Our use of Merkle trees is reminiscent of [39] in that the content of the leaves is public. The Merkle tree will actually serve as a "bridge" ensuring that: (i) The same string $\mathbf{x}$ is used in all steps while enabling dichotomic searches; (ii) At each step, the prover indeed uses some coordinate of $\mathbf{x}$ (without revealing which one), the choice of which is dictated by a path in the tree determined by $\mathrm{var}(\theta)$.

Since $\{\mathsf{com}_i\}_{i=0}^{\kappa-1}$ are public, both parties can deterministically compute the root $\mathbf{u}_{\mathsf{tree}}$ of the Merkle tree. For each $\theta \in [L]$, we consider the binary representation $d_{\theta,1}, \ldots, d_{\theta,\delta_\kappa}$ of $\mathrm{var}(\theta)$, which is part of the encoding of BP defined in (7). We then prove knowledge of a bit y_θ satisfying the statement "From

the root $\mathbf{u}_{\mathsf{tree}} \in \{0,1\}^{n\lceil \log q \rceil}$ of the tree, the path determined by the bits $d_{\theta,1}, \ldots, d_{\theta,\delta_\kappa}$ leads to the leaf associated with the commitment opened to y_θ." If the Merkle tree and the commitment scheme are both secure, it should hold that $y_\theta = x_{\mathrm{var}(\theta)}$. Said otherwise, we can provably perform a "dichotomic search" for $x_{\mathrm{var}(\theta)} = y_\theta$. Moreover, the techniques from [39] can be adapted to do this in zero-knowledge manner, i.e., without revealing the path nor the reached leaf.

Now, our task can be divided into 3 steps: (i) Proving that the searches on Merkle tree yield $y_1, \ldots, y_L$; (ii) Proving that the branching program evaluates to $\mathsf{BP}(\mathbf{x}) = 1$ if $y_1, \ldots, y_L$ are used in the evaluation; (iii) Proving all the other relations mentioned above, as well as the consistency of $\{\mathsf{com}_i\}_{i=0}^{\kappa-1}$ and the fact that they open to a certified $\mathbf{x} \in \{0,1\}^\kappa$.

Thanks to dichotomic searches, the communication cost drops to $\mathcal{O}(L\delta_\kappa + \kappa)$. These steps can be treated as explained below.

The Merkle Tree Step. At each step $\theta \in [L]$, the prover proves knowledge of a path made of δ_κ nodes $\mathbf{g}_{\theta,1}, \ldots, \mathbf{g}_{\theta,\delta_\kappa} \in \{0,1\}^{n\lceil \log q \rceil}$ determined by $d_{\theta,1}, \ldots, d_{\theta,\delta_\kappa}$, as well as their siblings $\mathbf{t}_{\theta,1}, \ldots, \mathbf{t}_{\theta,\delta_\kappa} \in \{0,1\}^{n\lceil \log q \rceil}$. Also, the prover argues knowledge of an opening $(y_\theta, \mathbf{r}_\theta) \in \{0,1\} \times \{0,1\}^m$ for the commitment of which $\mathbf{g}_{\theta,\delta_\kappa}$ is a binary decomposition. As shown in [39] (and recalled in the full paper), it suffices to prove the following relations (mod q):

$$\forall \theta \in [L] \begin{cases} \bar{\mathbf{A}} \cdot \mathsf{expand}(d_{\theta,1}, \mathbf{g}_{\theta,1}) + \bar{\mathbf{A}} \cdot \mathsf{expand}(\bar{d}_{\theta,1}, \mathbf{t}_{\theta,1}) = \mathbf{H}_{n,q-1} \cdot \mathbf{u}_{\mathsf{tree}}, \\ \bar{\mathbf{A}} \cdot \mathsf{expand}(d_{\theta,2}, \mathbf{g}_{\theta,2}) + \bar{\mathbf{A}} \cdot \mathsf{expand}(\bar{d}_{\theta,2}, \mathbf{t}_{\theta,2}) \\ \qquad\qquad - \mathbf{H}_{n,q-1} \cdot \mathbf{g}_{\theta,1} = \mathbf{0}, \\ \qquad\qquad \vdots \\ \bar{\mathbf{A}} \cdot \mathsf{expand}(d_{\theta,\delta_\kappa}, \mathbf{g}_{\theta,\delta_\kappa}) + \bar{\mathbf{A}} \cdot \mathsf{expand}(\bar{d}_{\theta,\kappa}, \mathbf{t}_{\theta,\kappa}) \\ \qquad\qquad - \mathbf{H}_{n,q-1} \cdot \mathbf{g}_{\theta,\delta_\kappa - 1} = \mathbf{0}, \\ \mathbf{a}_{\mathsf{com}} \cdot y_\theta + \bar{\mathbf{A}} \cdot \mathbf{r}_\theta - \mathbf{H}_{n,q-1} \cdot \mathbf{g}_{\theta,\delta_\kappa} = \mathbf{0}, \end{cases} \tag{21}$$

where $\mathsf{expand}(\cdot,\cdot)$ is defined in Table 1.

Extending.

- For each $(\theta, i) \in [L] \times [\delta_\kappa]$: Extend $\mathbf{g}_{\theta,i}, \mathbf{t}_{\theta,i} \in \{0,1\}^{m/2}$ to $\widetilde{\mathbf{g}}_{\theta,i}, \widetilde{\mathbf{t}}_{\theta,i} \in \mathsf{B}^2_{m/2}$, respectively. Then, let $\widehat{\mathbf{g}}_{\theta,i} = \mathsf{expand}(d_{\theta,i}, \widetilde{\mathbf{g}}_{\theta,i})$ and $\widehat{\mathbf{t}}_{\theta,i} = \mathsf{expand}(\bar{d}_{\theta,i}, \widetilde{\mathbf{t}}_{\theta,i})$.
- For each $\theta \in [L]$, extend the bit y_θ into the vector $\mathbf{y}_\theta = \mathsf{ext}_2(y_\theta) \in \{0,1\}^2$.
- Let $\widetilde{\mathbf{r}} = (\mathbf{r}_1^\top \mid \ldots \mid \mathbf{r}_L^\top)^\top \in \{0,1\}^{mL}$, then extend it into the vector $\widehat{\mathbf{r}} \in \mathsf{B}^2_{mL}$.

Combining. Next, we let $D_{\mathsf{tree}} = 5mL\delta_\kappa + 2L + 2mL$ and define

$$\begin{aligned} \mathbf{w}_{\mathsf{tree}} = \big(&\widetilde{\mathbf{g}}_{1,1}^\top \mid \widehat{\mathbf{g}}_{1,1}^\top \mid \widehat{\mathbf{t}}_{1,1}^\top \mid \ldots \mid \widetilde{\mathbf{g}}_{1,\delta_\kappa}^\top \mid \widehat{\mathbf{g}}_{1,\delta_\kappa}^\top \mid \widehat{\mathbf{t}}_{1,\delta_\kappa}^\top \mid \ldots \mid \widetilde{\mathbf{g}}_{L,1}^\top \mid \widehat{\mathbf{g}}_{L,1}^\top \mid \widehat{\mathbf{t}}_{L,1}^\top \\ &\mid \ldots \mid \widetilde{\mathbf{g}}_{L,\delta_\kappa}^\top \mid \widehat{\mathbf{g}}_{L,\delta_\kappa}^\top \mid \widehat{\mathbf{t}}_{L,\delta_\kappa}^\top \mid \mathbf{y}_1^\top \mid \ldots \mid \mathbf{y}_L^\top \mid \widehat{\mathbf{r}}^\top\big)^\top \in \{0,1\}^{D_{\mathsf{tree}}}. \end{aligned} \tag{22}$$

Then, observe that, the above $L(\delta_\kappa + 1)$ equations can be combined into one:

$$\mathbf{M}_{\mathsf{tree}} \cdot \mathbf{w}_{\mathsf{tree}} = \mathbf{v}_{\mathsf{tree}} \bmod q, \tag{23}$$

where matrix $\mathbf{M}_{\mathsf{tree}}$ and vector $\mathbf{v}_{\mathsf{tree}}$ are built from the public input.

The Branching Program Step. The last three parts of Table 1 describe the vector transformations that will be used to handle the secret vectors appearing in the evaluation of BP. The following equations emulate the evaluation process. In particular, for each $\theta \in [2, L]$, we introduce an extra vector $\mathbf{e}_\theta = (c_{\theta,0}, \ldots, c_{\theta,4}) \in \{0,1\}^5$ to enable the extraction of the values $\pi_{\theta,0}(\eta_{\theta-1})$, and $\pi_{\theta,1}(\eta_{\theta-1})$.

$$\begin{cases} \pi_{1,0}(0) \cdot \bar{y}_1 + \pi_{1,1}(0) \cdot y_1 - \eta_1 = 0, & \text{// computing } \eta_1 \text{ with } \eta_0 = 0 \\ \mathbf{e}_2 - \sum_{i=0}^{4} \mathsf{unit}_i \cdot c_{2,i} = (0,0,0,0,0)^\top, & \text{// we will also prove } \mathbf{e}_2 = \mathsf{unit}_{\eta_1} \\ f_{2,0} - \sum_{i=0}^{4} \pi_{2,0}(i) \cdot c_{2,i} = 0, & \text{// meaning: } f_{2,0} = \pi_{2,0}(\eta_1) \\ f_{2,1} - \sum_{i=0}^{4} \pi_{2,1}(i) \cdot c_{2,i} = 0, & \text{// meaning: } f_{2,1} = \pi_{2,1}(\eta_1) \\ f_{2,0} \cdot \bar{y}_2 + f_{2,1} \cdot y_2 - \eta_2 = 0, & \text{// computing } \eta_2 \\ \qquad\qquad \vdots & \\ \mathbf{e}_L - \sum_{i=0}^{4} \mathsf{unit}_i \cdot c_{L,i} = (0,0,0,0,0)^\top, & \text{// we will also prove } \mathbf{e}_L = \mathsf{unit}_{\eta_{L-1}} \\ f_{L,0} - \sum_{i=0}^{4} \pi_{L,0}(i) \cdot c_{L,i} = 0, & \text{// meaning: } f_{L,0} = \pi_{L,0}(\eta_{L-1}) \\ f_{L,1} - \sum_{i=0}^{4} \pi_{L,1}(i) \cdot c_{L,i} = 0, & \text{// meaning: } f_{L,1} = \pi_{L,1}(\eta_{L-1}) \\ f_{L,0} \cdot \bar{y}_L + f_{L,1} \cdot y_L = 0. & \text{// final state } \eta_L = 0 \end{cases} \tag{24}$$

Extending.

- For each $\theta \in [L-1]$, extend $\eta_\theta \in [0,4]$ to 5-dimensional vector $\mathbf{s}_\theta = \mathsf{ext}_5(\eta_\theta)$.
- For each $(\theta, j) \in [2, L] \times \{0,1\}$, extend $f_{\theta,j} \in [0,4]$ to $\mathbf{f}_{\theta,j} = \mathsf{ext}_5(f_{\theta,j})$.
- For each $(\theta, i) \in [2, L] \times [0,4]$, extend $c_{\theta,i} \in \{0,1\}$ to $\mathbf{c}_{\theta,i} = \mathsf{ext}_2(c_{\theta,i})$.
- Extend the products $\pi_{1,0}(0) \cdot \bar{y}_1$ and $\pi_{1,1}(0) \cdot y_1$ into 10-dimensional vectors $\mathbf{h}_{1,0} = \mathsf{ext}_{5\times 2}(\pi_{1,0}(0), \bar{y}_1)$ and $\mathbf{h}_{1,1} = \mathsf{ext}_{5\times 2}(\pi_{1,1}(0), y_1)$, respectively.
- For each $\theta \in [2, L]$, extend the products $f_{\theta,0} \cdot \bar{y}_\theta$ and $f_{\theta,1} \cdot y_\theta$ into 10-dimensional vectors $\mathbf{h}_{\theta,0} = \mathsf{ext}_{5\times 2}(f_{\theta,0}, \bar{y}_\theta)$ and $\mathbf{h}_{\theta,1} = \mathsf{ext}_{5\times 2}(f_{\theta,1}, y_\theta)$.
- For $(\theta, i) \in [2, L] \times [0,4]$, extend the products $\pi_{\theta,0}(i) \cdot c_{\theta,i}$ and $\pi_{\theta,1}(i) \cdot c_{\theta,i}$ into $\mathbf{z}_{\theta,0,i} = \mathsf{ext}_{5\times 2}(\pi_{\theta,0}(i), c_{\theta,i})$ and $\mathbf{z}_{\theta,1,i} = \mathsf{ext}_{5\times 2}(\pi_{\theta,1}(i), c_{\theta,i})$, respectively.

Combining. Let $D_{\mathsf{BP}} = 150L - 130$, and form $\mathbf{w}_{\mathsf{BP}} \in [0,4]^{D_{\mathsf{BP}}}$ of the form:

$$\begin{aligned} \big(\mathbf{s}_1^\top \mid \ldots \mid \mathbf{s}_{L-1}^\top \mid \mathbf{e}_2^\top \mid \ldots \mid \mathbf{e}_L^\top \mid \mathbf{c}_{2,0}^\top \mid \ldots \mid \mathbf{c}_{L,4}^\top \mid \mathbf{z}_{2,0,0}^\top \mid \ldots \mid \mathbf{z}_{L,1,4}^\top \mid \\ \mathbf{f}_{2,0}^\top \mid \ldots \mid \mathbf{f}_{L,1}^\top \mid \mathbf{h}_{1,0}^\top \mid \mathbf{h}_{1,1}^\top \mid \mathbf{h}_{2,0}^\top \mid \mathbf{h}_{2,1}^\top \mid \ldots \mid \mathbf{h}_{L,0}^\top \mid \mathbf{h}_{L,1}^\top \big)^\top. \end{aligned} \tag{25}$$

Then, observe that the vector $\mathbf{w}_{\mathsf{BP}}$ of (25) satisfies *one* equation of the form:

$$\mathbf{M}_{\mathsf{BP}} \cdot \mathbf{w}_{\mathsf{BP}} = \mathbf{v}_{\mathsf{BP}}, \tag{26}$$

where matrix $\mathbf{M}_{\mathsf{BP}}$ and vector $\mathbf{v}_{\mathsf{BP}}$ are obtained from the common input. Note that we work with integers in $[0,4]$, which are much smaller than q. As a result,

$$\mathbf{M}_{\mathsf{BP}} \cdot \mathbf{w}_{\mathsf{BP}} = \mathbf{v}_{\mathsf{BP}} \bmod q. \tag{27}$$

Conversely, if we can prove that (27) holds for a well-formed vector $\mathbf{w}_{\mathsf{BP}}$, then that vector should also satisfy (26).

The Third Step. In the third layer, we have to prove knowledge of:

$$\begin{cases} d_{1,1},\ldots,d_{L,\delta_\kappa} \in \{0,1\}, \pi_{1,0}(0),\ldots,\pi_{L,1}(4) \in [0,4],\ \mathfrak{m} \in \{0,1\}^{m_d}, \\ \mathbf{x} = (x_0,\ldots,x_{\kappa-1})^\top \in \{0,1\}^\kappa,\ \mathfrak{m}_{\mathsf{U},\mathbf{x}} \in \{0,1\}^{\frac{m}{2}+\kappa},\ \widehat{\mathfrak{m}}_{\mathsf{U},\mathbf{x}} \in \{0,1\}^{\frac{m}{2}}, \\ \mathbf{e}_{\mathsf{U}} \in \{0,1\}^m,\ \mathbf{r}_{\mathsf{com},0},\ldots,\mathbf{r}_{\mathsf{com},\kappa-1} \in \{0,1\}^m,\ \mu \in \{0,1\}^t,\ \tau \in \{0,1\}^\ell, \\ \tau_{\mathsf{U}} \in \{0,1\}^{\ell_I}, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_{\mathsf{U},1}, \mathbf{v}_{\mathsf{U},2}, \mathbf{r}_{\mathsf{U}} \in [-\beta,\beta]^m, \mathbf{e} \in \{-1,0,1\}^t, \nu \in [-B,B]^t, \end{cases} \tag{28}$$

which satisfy the equations of (11) for $\mathbf{z}_{\mathsf{BP},\rho} = (d_{1,1},\ldots,d_{L,\delta_\kappa},\pi_{1,0}(0),\ldots,\pi_{L,1}(4))^\top$ and, $\forall i \in [0,\kappa-1]$, the bit x_i is committed in com_i with randomness $\mathbf{r}_{\mathsf{com},i}$:

$$\begin{bmatrix} \mathbf{a}_{\mathsf{com}} & & \\ & \ddots & \\ & & \mathbf{a}_{\mathsf{com}} \end{bmatrix} \cdot \mathbf{x} + \begin{bmatrix} \bar{\mathbf{A}} & & \\ & \ddots & \\ & & \bar{\mathbf{A}} \end{bmatrix} \cdot \begin{pmatrix} \mathbf{r}_{\mathsf{com},0} \\ \vdots \\ \mathbf{r}_{\mathsf{com},\kappa-1} \end{pmatrix} = \begin{pmatrix} \mathsf{com}_0 \\ \vdots \\ \mathsf{com}_{\kappa-1} \end{pmatrix} \bmod q.$$

Decomposing. We use $\mathsf{vdec}'_{m,\beta}(\cdot)$ to decompose $\mathbf{v}_1,\mathbf{v}_2,\mathbf{v}_{\mathsf{U},1},\mathbf{v}_{\mathsf{U},2},\mathbf{r}_{\mathsf{U}} \in [-\beta,\beta]^m$ into $\bar{\mathbf{v}}_1,\bar{\mathbf{v}}_2,\bar{\mathbf{v}}_{\mathsf{U},1},\bar{\mathbf{v}}_{\mathsf{U},2},\bar{\mathbf{r}}_{\mathsf{U}} \in \{-1,0,1\}^{m\delta_\beta}$, respectively. Similarly, we decompose vector $\nu \in [-B,B]^t$ into vector $\bar{\nu} = \mathsf{vdec}'_{t,B}(\nu) \in \{-1,0,1\}^{t\delta_B}$.

Extending and Combining. Next, we perform the following steps:

- For each $(\theta,i) \in [L] \times [\delta_\kappa]$, extend $d_{\theta,i}$ to $\mathbf{d}_{\theta,i} = \mathsf{ext}_2(d_{\theta,i})$.
- For each $(\theta,j,i) \in [L] \times \{0,1\} \times [0,4]$, extend $\pi_{\theta,j}(i)$ to $\mathit{\Pi}_{\theta,j,i} = \mathsf{ext}_5(\pi_{\theta,j}(i))$.
- Let $\overline{\mathbf{w}}_{3,1} = \left(\mathbf{x}^\top|\mathbf{r}^\top_{\mathsf{com},0}|\ldots|\mathbf{r}^\top_{\mathsf{com},\kappa-1}|\mathfrak{m}^\top_{\mathsf{U},\mathbf{x}}|\widehat{\mathfrak{m}}^\top_{\mathsf{U},\mathbf{x}}|\mathfrak{m}^\top \mid \mathbf{e}^\top_{\mathsf{U}}|\mu^\top\right)^\top \in \{0,1\}^{D_{3,1}}$, where $D_{3,1} = \kappa(m+2)+2m+m_d+t$. Then extend $\overline{\mathbf{w}}_{3,1}$ to $\mathbf{w}_{3,1} \in \mathsf{B}^2_{D_{3,1}}$.
- Define the vector $\overline{\mathbf{w}}_{3,2} = (\bar{\mathbf{v}}_1^\top|\bar{\mathbf{v}}^\top_{\mathsf{U},1}|\bar{\mathbf{r}}^\top_{\mathsf{U}}|\bar{\nu}^\top|\mathbf{e}^\top)^\top \in \{-1,0,1\}^{D_{3,2}}$ of dimension $D_{3,2} = 3m\delta_\beta + t(\delta_B+1)$ and extend it into $\mathbf{w}_{3,2} \in \mathsf{B}^3_{D_{3,2}}$.
- Extend $\bar{\mathbf{v}}_2$ to $\mathbf{s}_0 \in \mathsf{B}^3_{m\delta_\beta}$. Then for $j \in [\ell]$, form vector $\mathbf{s}_j = \mathsf{expand}\big(\tau[j],\mathbf{s}_0\big)$.
- Extend $\bar{\mathbf{v}}_{\mathsf{U},2}$ to $\mathbf{s}_{\mathsf{U},0} \in \mathsf{B}^3_{m\delta_\beta}$. Then for $j \in [\ell_I]$, form $\mathbf{s}_{\mathsf{U},j} = \mathsf{expand}\big(\tau_{\mathsf{U}}[j],\mathbf{s}_{\mathsf{U},0}\big)$.

Given the above transformations, let $D_3 = 2L(\delta_\kappa+25)+2D_{3,1}+3D_{3,2}+3m\delta_\beta(2\ell+1)+3m\delta_\beta(2\ell_I+1)$ and construct vector $\mathbf{w}_3 \in [-1,4]^{D_3}$ of the form:

$$\big(\mathbf{d}^\top_{1,1} \mid \ldots \mid \mathbf{d}^\top_{L,\delta_\kappa} \mid \mathit{\Pi}^\top_{1,0,0} \mid \ldots \mid \mathit{\Pi}^\top_{L,1,4} \mid \mathbf{w}^\top_{3,1} \mid \mathbf{w}^\top_{3,2} \mid \mathbf{s}^\top_0 \mid \mathbf{s}^\top_1 \mid \ldots \mid \mathbf{s}^\top_\ell \mid \mathbf{s}^\top_{\mathsf{U},0} \mid \mathbf{s}^\top_{\mathsf{U},1} \mid \ldots \mid \mathbf{s}^\top_{\mathsf{U},\ell_I} \mid \big)^\top. \tag{29}$$

Observe that the given five equations can be combined into one of the form:

$$\mathbf{M}_3 \cdot \mathbf{w}_3 = \mathbf{v}_3 \bmod q, \tag{30}$$

where matrix $\mathbf{M}_3$ and vector $\mathbf{v}_3$ can be built from the public input.

Putting Pieces Altogether. At the final stage of the process, we connect the three aforementioned steps. Indeed, all the equations involved in our process are captured by (23), (27), and (30) - which in turn can be combined into:

$$\mathbf{M} \cdot \mathbf{w} = \mathbf{v} \bmod q, \tag{31}$$

where $\mathbf{w} = (\mathbf{w}_{\mathsf{tree}}^\top \mid \mathbf{w}_{\mathsf{BP}}^\top \mid \mathbf{w}_3^\top)^\top \in [-1,4]^D$, for

$$D = D_{\mathsf{tree}} + D_{\mathsf{BP}} + D_3 = \widetilde{\mathcal{O}}(\lambda) \cdot (L \cdot \log \kappa + \kappa) + \widetilde{\mathcal{O}}(\lambda) \cdot (\log N + \lambda) + \widetilde{\mathcal{O}}(1) \cdot t.$$

The components of $\mathbf{w}$ all have constraints listed in Table 1. By construction, these blocks either belong to the special sets $\mathsf{B}_{\mathsf{m}}^2$, $\mathsf{B}_{\mathsf{m}}^3$ or they have the special forms $\mathsf{expand}(\cdot,\cdot)$, $\mathsf{ext}_2(\cdot)$, $\mathsf{ext}_5(\cdot)$, $\mathsf{ext}_{5\times 2}(\cdot,\cdot)$, which are invariant under the permutations defined in Table 1. As a result, we can specify suitable sets VALID, $\mathcal{S}$ and permutations of D elements $\{\Gamma_\phi : \phi \in \mathcal{S}\}$, for which the conditions of (12) are satisfied. The description of VALID, $\mathcal{S}$ and Γ_ϕ is detailed in the full paper.

Our desired argument system then works as follows. At the beginning of the interaction, the prover computes commitments $\mathsf{com}_0, \ldots, \mathsf{com}_{\kappa-1} \in \mathbb{Z}_q^n$ and send them once to the verifier. Both parties construct matrix $\mathbf{M}$ and vector $\mathbf{v}$ based on the public input as well as $\mathsf{com}_0, \ldots, \mathsf{com}_{\kappa-1}$, while the prover prepares vector $\mathbf{w}$, as described. Finally, they run the protocol of Section 6.1, which has communication cost $\mathcal{O}(D \log q) = \mathcal{O}(L \cdot \log \kappa + \kappa)$.

Acknowledgements

Part of this research was funded by Singapore Ministry of Education under Research Grant MOE2016-T2-2-014(S) and by the French ANR ALAMBIC project (ANR-16-CE39-0006).

References

1. M. Abe, J. Camenisch, M. Dubovitskaya, and R. Nishimaki. Universally composable adaptive oblivious transfer (with access control) from standard assumptions. *ACM Workshop on Digital Identity Management*, 2013.
2. S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. *Eurocrypt*, 2010.
3. W. Aiello, Y. Ishai, and O. Reingold. Priced oblivious transfer: How to sell digital goods. *Eurocrypt*, 2001.
4. J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. *STACS 2009*, 2009.
5. G. Asharov, A. Jain, A. Lopez-Alt, E. Tromer, V. Vaikuntanathan, and D. Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. *Eurocrypt 2012*, 2012.
6. D. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc1. *STOC'86*, 1986.
7. F. Böhl, D. Hofheinz, T. Jager, J. Koch, and C. Striecks. Confined guessing: New signatures from standard assumptions. *Journal of Cryptology*, 28(1):176–208, 2015.
8. D. Boneh and X. Boyen. Efficient selective-ID secure Identity-Based Encryption without random oracles. *Eurocrypt*, 2004.
9. X. Boyen. Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. *PKC 2010*, 2010.
10. Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé. On the classical hardness of learning with errors. *STOC*, 2013.

11. J. Camenisch, M. Dubovitskaya, R. Enderlein, and G. Neven. Oblivious transfer with hidden access control from attribute-based encryption. *SCN 2012*, 2012.
12. J. Camenisch, M. Dubovitskaya, and G. Neven. Oblivious transfer with access control. *ACM-CCS 2009*, 2009.
13. J. Camenisch, M. Dubovitskaya, G. Neven, and G. Zaverucha. Oblivious transfer with hidden access control policies. *PKC'11*, 2011.
14. J. Camenisch, G. Neven, and a. shelat. Simulatable adaptive oblivious transfer. *Eurocrypt 2007*, 2007.
15. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. *FOCS 2001*, 2001.
16. D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. *Eurocrypt*, 2010.
17. B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. *FOCS 1995*, 1995.
18. S. Coull, M. Green, and S. Hohenberger. Controlling access to an oblivious database using stateful anonymous credentials. *PKC 2009*, 2009.
19. G. Di Crescenzo, R. Ostrovsky, and S. Rajagopalan. Conditional oblivious transfer and timed-release encryption. *Eurocrypt'99*, 1999.
20. L. Ducas and D. Stehlé. Sanitization of FHE ciphertexts. *Eurocrypt 2016*, 2016.
21. S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.
22. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. *Crypto'86*. Springer, 1987.
23. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. *STOC*, 2008.
24. C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. *Crypto*, 2013.
25. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. *STOC*, 1987.
26. S. Gorbunov and D. Vinayagamurthy. Riding on asymmetry: Efficient abe for branching programs. *Asiacrypt 2015*, 2015.
27. S. D. Gordon, J. Katz, and V. Vaikuntanathan. A group signature scheme from lattice assumptions. *Asiacrypt 2010*, 2010.
28. M. Green and S. Hohenberger. Blind identity-based encryption and simulatable oblivious transfer. *Asiacrypt 2007*, 2007.
29. M. Green and S. Hohenberger. Universally composable adaptive oblivious transfer. *Asiacrypt 2008*, 2008.
30. M. Green and S. Hohenberger. Practical adaptive oblivious transfer from simple assumptions. *TCC 2011*, 2011.
31. J. Herranz. Restricted adaptive oblivious transfer. *Theoretical Computer Science*, 412(46):6498–6506, 2011.
32. R. Hiromasa, M. Abe, and T. Okamoto. Packing messages and optimizing bootstrapping in GSW-FHE. *PKC 2015*, 2015.
33. S. Hohenberger and B. Waters. Short and stateless signatures from the RSA assumption. *Crypto*, 2009.
34. S. Jarecki and X. Liu. Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. *TCC 2009*, 2009.
35. A. Kawachi, K. Tanaka, and K. Xagawa. Concurrently secure identification schemes based on the worst-case hardness of lattice problems. *Asiacrypt'08*, 2008.
36. K. Kurosawa, L. Phong, and R. Nojima. Generic fully simulatable adaptive oblivious transfer. *ACNS 2011*, 2011.

37. B. Libert, S. Ling, F. Mouhartem, K. Nguyen, and H. Wang. Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. *Asiacrypt 2016*, 2016.
38. B. Libert, S. Ling, F. Mouhartem, K. Nguyen, and H. Wang. Zero-knowledge arguments for matrix-vector relations and lattice-based group encryption. *Asiacrypt 2016*, 2016.
39. B. Libert, S. Ling, K. Nguyen, and H. Wang. Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. *Eurocrypt 2016*, 2016.
40. A. Y. Lindell. Efficient fully-simulatable oblivious transfer. *CT-RSA*, 2008.
41. S. Ling, K. Nguyen, D. Stehlé, and H. Wang. Improved zero-knowledge proofs of knowledge for the isis problem, and applications. *PKC 2013*, 2013.
42. S. Ling, K. Nguyen, and H. Wang. Group signatures from lattices: Simpler, tighter, shorter, ring-based. *PKC 2015*, 2015.
43. D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. *Eurocrypt*, 2012.
44. M. Naor and B. Pinkas. Oblivious transfer with adaptive queries. *Crypto*, 1999.
45. M. Naor and B. Pinkas. Efficient oblivious transfer protocols. *SODA*, 2001.
46. M. Naor and B. Pinkas. Computationally secure oblivious transfer. *Journal of Cryptology*, 18(1):1–35, 2005.
47. T. Nishide, K. Yoneyama, and K. Ohta. Attribute-based encryption with partially hidden encryptor-specified access structures. *ACNS'08*, 2008.
48. C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. *Crypto 2008*, 2008.
49. M. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Aiken Computation Laboratory, Harvard University, 1981.
50. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *STOC*, 2005.
51. A. Sahai and B. Waters. Fuzzy identity-based encryption. *Eurocrypt 2005*, 2005.
52. J. Stern. A new paradigm for public key identification. *IEEE Transactions on Information Theory*, 42(6):1757–1768, 1996.
53. Y. Tauman-Kalai. Smooth projective hashing and two-message oblivious transfer. *Eurocrypt*, 2005.
54. Y. Zhang, M.-H. Au, D. Wong, Q. Huang, N. Mamoulis, D. Cheung, and S.-M. Yiu. Oblivious transfer with access control: Realizing disjunction without duplication. *Pairing*, 2010.