Compact Zero-Knowledge Proofs of Small Hamming Weight

Ivan Damgård[1], Ji Luo[2], Sabine Oechsner[1], Peter Scholl[1(✉)], and Mark Simkin[1]

[1] Aarhus University, Aarhus, Denmark
{ivan,oechsner,peter.scholl,simkin}@cs.au.dk
[2] Tsinghua University, Beijing, China
j-luo14@mails.tsinghua.edu.cn

Abstract. We introduce a new technique that allows to give a zero-knowledge proof that a committed vector has Hamming weight bounded by a given constant. The proof has unconditional soundness and is very compact: It has size independent of the length of the committed string, and for large fields, it has size corresponding to a constant number of commitments. We show five applications of the technique that play on a common theme, namely that our proof allows us to get malicious security at small overhead compared to semi-honest security: (1) actively secure k-out-of-n OT from black-box use of 1-out-of-2 OT, (2) separable accountable ring signatures, (3) more efficient preprocessing for the TinyTable secure two-party computation protocol, (4) mixing with public verifiability, and (5) PIR with security against a malicious client.

1 Introduction

Commitments and zero-knowledge proofs are extremely important universal tools that protocol designers use to upgrade semi-honestly secure protocols to maliciously secure constructions. This follows the well known paradigm of proving you "did the right thing", without revealing any secret data. For this to be interesting, we want of course that the size of the proofs is small, in order to have small communication overhead for getting malicious security.

Generic techniques using NP reductions will of course always work, but are extremely inefficient. If proof size is the only goal to optimise for, then Succinct Non-Interactive Arguments (SNARGs) give a much better option that works in general, but it is very costly to construct a proof, though verification can sometimes be fast [5]. Moreover, soundness is only computational and requires non-falsifiable assumptions that are regarded as controversial by some. A different general approach was introduced in [23] (based on [28]), the ZKBoo protocol, which is computationally much more efficient than SNARGs and based on standard assumptions, but the proof size is much larger.

Thus a natural question is: Can we, at least for special types of statements, have both the prover and the verifier be very efficient, have unconditional soundness based on standard assumptions, and still have the size of the proof be

M. Abdalla and R. Dahab (Eds.): PKC 2018, LNCS 10770, pp. 530–560, 2018.
https://doi.org/10.1007/978-3-319-76581-5_18

much smaller than that of the statement? Where, of course, we would like that the statements we can prove are useful to get malicious security in meaningful applications.

More specifically, we consider an arbitrary linearly homomorphic commitment scheme that allows committing to elements in a finite field $\mathbb{F}$ (we show several examples later), and the following scenario: A prover has committed to a string $\boldsymbol{x} \in \mathbb{F}^n$, using n commitments, and claims that the Hamming weight of $\boldsymbol{x}$ is at most d. Such a statement can be proved with unconditional soundness, using the techniques based on Σ-protocols from [16], but the size of the proof would be dominated by the cost of sending a number of commitments that is linear in n.

Related Work. Efficiently proving properties of one or a small number of committed values from a public list $c_1, \ldots, c_n$ (of committed or public values) has been considered in several works. Brands et al. [9] propose a zero-knowledge protocol for proving non-membership in a list with squareroot complexity in the size of the list. Groth [25] gives zero-knowledge arguments for algebraic statements about matrices from a list of committed matrices with sublinear communication complexity. Bayer and Groth [3] give logarithmic size zero-knowledge arguments of list membership. Groth and Kohlweiss [26] present a logarithmic size zero-knowledge proof for a list of commitments where at least one commitments opens to 0. This result was improved with respect to practical efficiency by Bootle et al. [7].

Our contributions. We present a protocol that allows the prover to show in ZK with unconditional soundness that at most d out of n commitments do not contain 0, or alternatively, that the Hamming weight of the message vector of the commitments is at most d. The communication complexity is dominated by sending $O\left(\frac{kd}{\log|\mathbb{F}|}\right)$ commitments for an error probability of $2^{-\Omega(k)}$. Thus, if the size of $\mathbb{F}$ is exponential in the security parameter, we only need a constant number of commitments, and the communication overhead is always independent of n. Since the complexity grows linearly in d, our construction is more interesting for small values of d compared to n, and particularly constant d. In addition, the protocol is public-coin hence can be made non-interactive in the random oracle model using the Fiat-Shamir paradigm [21].

We show several applications of this type of proof: Our first application is to efficient secure computation with active security. We obtain an actively secure d-out-of-n oblivious transfer (OT) protocol which makes only black-box use of 1-out-of-2 OT and hence allows the use of efficient OT extension techniques, avoiding costly public-key operations [27]. The only previously known black-box constructions (without relying on public-key assumptions like DDH or pairings) are not actively secure [34], or only realise a weaker form of approximate d-out-of-n OT [39]. Our protocol has a communication complexity of $O(nk + k^2 d)$ bits, and we show how to reduce this to $O(nk)$ in an amortized setting using recent advances in homomorphic commitments based on OT and error-correcting

codes [13]. This gives constant overhead when the sender's strings are of length $\Omega(k)$, for arbitrary d.[1]

Second, we construct a separable accountable ring signature scheme. A ring signature scheme allows to generate signatures proving that someone in a given set of parties signed the message without revealing the identity of the signer. Accountability means that the signer can dynamically choose a trusted party who will be able to compute her (the signer's) identity from the signature, whereas no one else can. Separability means that members of the set are not required to use the same type of key or signature algorithm, but can rather use different types keys like El Gamal and RSA keys. In our case, the only requirement we impose on the public key of each participant is that there exists a Σ-protocol for showing knowledge of the corresponding secret key. Note that accountable ring signatures imply group signatures where the trusted party is chosen and fixed at key generation time. We first construct a separable ring signature using the standard OR-proof technique from [16], and then add accountability using our compact proofs. Compared to doing the OR-proof only, the involved overhead is very small: it is additive and independent of the number of parties.

Third, we also show how to apply our compact proof to generate preprocessing data for the TinyTable secure computation protocol [18]. This can give a concrete reduction in communication complexity of around a factor of two, compared with previous approaches [30], depending on the sizes of the lookup table gates used in the circuit.

Fourth, we show how to upgrade the "shuffle in public" paradigm by Adida and Wikström [1] so that the publicly verifiable proof that the shuffle is correctly formed has size $O(n)$ (where n is the number of ciphertexts to be shuffled). More precisely, [1] shows how to make a quite efficient use-once obfuscation of a shuffle operation that can then be applied later to applying a secret permutation to a set of ciphertexts. We also show a special purpose MPC protocol that a set of parties can use to efficiently generate both the obfuscation and the proof.

Finally, we show how to upgrade a standard single-server PIR protocol to be secure against a malicious client with overhead a factor $o(1)$. This protocol can be based on any additively homomorphic encryption scheme.

2 Preliminaries

2.1 Definition of Commitment Schemes

We will consider two types of linearly homomorphic commitment schemes that allow us to commit to elements in a finite field $\mathbb{F}$.

Type 1 commitments. This type of commitment scheme consists of two algorithms KeyGen and Commit. We assume for now that $\mathbb{F}$ is a prime field of order q for some prime q, and will consider extension fields later in Sect. 3.1.

[1] One could also obtain constant overhead with generic secure two-party computation techniques [29], but this would be prohibitively expensive.

KeyGen is run by a prover P and takes as input 1^k, where k is the security parameter, and outputs a public key pk that is sent to the verifier V. We assume that the verifier can convince himself that pk is valid, i.e., it is a possible output from KeyGen. This can be a direct check or via an interactive protocol, but we will not be concerned with the details of this.

Commit is run by P and takes as input $x \in \mathbb{F}$ and randomness $r \in H$, and outputs a commitment $\mathsf{Commit}_{\mathrm{pk}}(x, r) \in G$ (to the verifier), where G, H are finite groups. To open a commitment c, P sends x, r to V who checks that $c = \mathsf{Commit}_{\mathrm{pk}}(x, r)$ and accepts or rejects accordingly.

We assume the commitment scheme is:

Perfectly binding. For any valid public key pk, x is uniquely determined from $\mathsf{Commit}_{\mathrm{pk}}(x, r)$.

Computationally hiding. Consider the following experiment: Run $\mathsf{KeyGen}(1^k)$ to get pk, give it to a probabilistic polynomial time adversary A who chooses two elements $x_0, x_1 \in \mathbb{F}$ and gets $\mathsf{Commit}_{\mathrm{pk}}(x_b, r)$ where b is either 0 or 1. A outputs a guess bit b'. For all such A, we require its advantage

$$|\Pr[b' = 1 \mid b = 0] - \Pr[b' = 1 \mid b = 1]|$$

to be negligible in k.

Homomorphic. We write the group operations in G and H additively and note that since $\mathbb{F}$ is a prime field, we can think of $u \in \mathbb{F}$ as an integer and hence, e.g., $ur \in H$ is well defined. We then require Commit to be a homomorphism in the sense that

$$u\mathsf{Commit}_{\mathrm{pk}}(x, r) + v\mathsf{Commit}_{\mathrm{pk}}(y, s) = \mathsf{Commit}_{\mathrm{pk}}(ux + vy, ur + vs)$$

for all $x, y, u, v \in \mathbb{F}$ and $r, s \in H$.

q-invertible. Note that, since q is the order of $\mathbb{F}$, qc is a commitment to 0 for any commitment c (by the homomorphic property). In addition, we require that qc can be "explained" as a commitment to 0, even given only c. More precisely, there exists a polynomial time computable function $f_0 : G \mapsto H$ such that for any commitment $c \in G$, we have $qc = \mathsf{Commit}(0, f_0(c))$.

The q-inversion property was defined (with minor differences) in [15]. Note that if H is a vector space over $\mathbb{F}$, then the property is trivially satisfied, we can set $f_0(c) = 0$.

Type 2 commitments. This type of scheme is defined by an algorithm Verify and an ideal functionality $\mathsf{F}_{\mathrm{Com}}$, which we assume is available to prover P and verifier V. The parties initially agree on the field $\mathbb{F}$ and a statistical security parameter k, both are sent to $\mathsf{F}_{\mathrm{Com}}$ once and for all. $\mathsf{F}_{\mathrm{Com}}$ then sends a global, private verification key, sk, to V. To commit to a field element x, P sends x to $\mathsf{F}_{\mathrm{Com}}$ which then returns a bit string $\boldsymbol{m}_x$ to P and also sends a string $\boldsymbol{k}_x$ to V. To open, P sends $x, \boldsymbol{m}_x$ to V. Then V runs $\mathsf{Verify}_{\mathrm{sk}}(x, \boldsymbol{m}_x, \boldsymbol{k}_x)$ which returns accept or reject.

Intuitively, one can think of $\boldsymbol{m}_x$ as a MAC on x and $\boldsymbol{k}_x$ as a key that V uses to check the MAC. We assume the commitment scheme is:

Statistically binding. If, when opening a commitment to x, the prover sends $x', \boldsymbol{m}'_x$ and $x' \neq x$, then V accepts with negligible probability.

Perfectly hiding. For each commitment created by $\mathsf{F}_{\mathrm{Com}}$ to some x, the distribution of $\boldsymbol{k}_x$ is independent of x (and of any other value sent to $\mathsf{F}_{\mathrm{Com}}$).

Homomorphic. The strings $\boldsymbol{m}_x, \boldsymbol{k}_x$ created for a commitment come from finite-dimensional vector spaces G, H over $\mathbb{F}$, respectively. Furthermore, for any two commitments $(x, \boldsymbol{m}_x, \boldsymbol{k}_x)$ and $(y, \boldsymbol{m}_y, \boldsymbol{k}_y)$ and all $u, v \in \mathbb{F}$, we have that $(ux + vy, u\boldsymbol{m}_x + v\boldsymbol{m}_y, u\boldsymbol{k}_x + v\boldsymbol{k}_y)$ is a valid commitment to $ux + vy$, i.e., it can be opened to $ux + vy$ and not to any other value.

Notation. In the following, we will use $\langle x \rangle$ as a shorthand for either type of commitment, so we suppress for simplicity public key and randomness from the notation. Likewise, we will use $\langle x \rangle + \langle y \rangle = \langle x + y \rangle$ and $c\langle x \rangle = \langle cx \rangle$ for a public value $c \in \mathbb{F}$ as a shorthand for applications of the homomorphic properties as defined above.

2.2 Example Commitment Schemes

Type 1 schemes. An example of a Type 1 commitment scheme is based on El Gamal encryption with the message in the exponent. More concretely, we let KeyGen choose p, q to be primes where q divides $p-1$ and is k bits long. KeyGen also chooses random elements $g, h \in \mathbb{Z}_p^*$ of order q. We then set $\mathbb{F} = G = \mathbb{Z}_q$, $H = \{(g^r, g^x h^r) : x, r \in \mathbb{Z}\}$, $\mathrm{pk} = (p, q, g, h)$ and $\mathsf{Commit}_{\mathrm{pk}}(x, r) = (g^r, g^x h^r)$. This is well known to be hiding under the DDH assumption. Note that if a party knows the corresponding El Gamal secret key, he cannot decrypt a committed message since it is in the exponent, but he can decide if a committed value is 0 or not. We may of course do something completely similar in an elliptic curve group of order q. More generally, a commitment scheme with the right properties follows from the existence of q-one way functions as introduced in [15], which implies also constructions based on the quadratic residuosity assumption (for $\mathbb{F} = \mathbb{Z}_2$) and generalizations thereof.

Another example can be derived from Paillier encryption [37]. Here the plaintext space is $\mathbb{Z}_N$ for an RSA modulus N, which is not a field but nevertheless compatible with our main construction. See Appendix A for detailed discussion.

It seems tempting to use a somewhat homomorphic encryption scheme based on (Ring-)LWE as basis for our commitments, simply by letting a commitment to x be an encryption of x. But this does not quite fit in our model. The reason is that in this case, the randomness should not be chosen uniformly but must be small enough to avoid overflow, which, should it happen, would invalidate binding. This means the prover must convince the verifier that a commitment is well-formed. Moreover, the above Σ-protocols must be modified to work for this example and there is a limit to the number of homomorphic operations we can support. Modulo this, however, it is possible to make our main protocol work in this case as well.

Type 2 schemes. We construct Type 2 commitment schemes from UC commitments based on oblivious transfer, which can be used to implement a form of the $\mathsf{F}_{\mathrm{Com}}$ functionality. A simple example of $\mathsf{F}_{\mathrm{Com}}$ is based on information-theoretic MACs: On initialisation, $\mathsf{F}_{\mathrm{Com}}$ samples and sends a random field element $\mathsf{sk} := \alpha \in \mathbb{F}$ to the verifier, V. To commit to a message $x \in \mathbb{F}$ from P, $\mathsf{F}_{\mathrm{Com}}$ samples $\beta \in \mathbb{F}$, computes $\gamma = x \cdot \alpha + \beta$ to P, before sending $\boldsymbol{m}_x = \gamma$ to P and $\boldsymbol{k}_x = \beta$ to V. The verification algorithm simply checks that $\gamma = x \cdot \alpha + \beta$. This is unconditionally hiding and statistically binding if $|\mathbb{F}| = 2^{\Omega(k)}$, since forging an opening requires guessing the secret α. Realising this $\mathsf{F}_{\mathrm{Com}}$ functionality can be done using 1-out-of-2 correlated oblivious transfer to commit to bits [35], and repeating this k times allows committing to arbitrary field elements when $|\mathbb{F}| \leq 2^k$ (similarly to [32]). We provide more details in Sect. 4.1.

Another approach is to use recent, more efficient constructions of UC homomorphic commitment schemes [13,22], which have message space $\mathbb{F}^\ell$ for $\ell = \Omega(k)$. This has the advantage that arbitrary field elements can be committed to with $o(1)$ overhead, using only 1-out-of-2 oblivious transfer and error-correcting codes. However, because the message space is now a vector space and not a finite field, this can only be applied to our zero-knowledge proof and applications in a batch setting, where many proofs are carried out in parallel. In Appendix B, we show how to instantiate $\mathsf{F}_{\mathrm{Com}}$ in this way, and give a simpler presentation of the commitment scheme of [13] in terms of code-based information-theoretic MACs.

2.3 Auxiliary Protocols

Proof of Commitment to 0. The homomorphic property of both types of commitments implies, as is well known, that P can efficiently convince V that a commitment c contains the value 0:[2]

1. P sends $a = \langle 0 \rangle$ (using fresh randomness) to V.
2. V sends a random challenge $e \in \mathbb{F}$ to P.
3. P opens $d = a + ec$ and V checks that d was correctly opened to reveal 0.

It is easy to see that this is a Σ-protocol, i.e., it is complete, honest verifier zero-knowledge, and special sound in the sense that if any P^* can send a and answer correctly to two different challenges e, e', he must know how to open c to reveal 0. To see this for Type 1 commitments, note that we have randomness values s, s' such that $\mathsf{Commit}(0, s) = a + ec$ and $\mathsf{Commit}(0, s') = a + e'c$ which implies $\mathsf{Commit}(0, s - s') = (e - e')c$. Multiplying by $y = (e - e')^{-1}$ on both sides, we obtain $\mathsf{Commit}(0, y(s - s')) = c + tqc$ for some integer t. By the q-inversion property we can rewrite this as $\mathsf{Commit}(0, y(s - s') - tf_0(c)) = c$ as desired. The proof for Type 2 commitment is trivial and is left to the reader.

[2] This can be useful if revealing the randomness used for c might leak side information, so that we do not want to simply open c.

Proof of Multiplication. Another well-known Σ-protocol proves, for commitments $c_x = \langle x \rangle, c_y = \langle y \rangle, c_z = \langle z \rangle$, that $z = xy$:

1. P sends V two commitments $a = \langle \alpha \rangle$ and $b = \langle \alpha y \rangle$ for some random α.
2. V sends a random challenge $e \in \mathbb{F}$ to P.
3. P opens $ec_x + a$, that is, he reveals $w = ex + \alpha$. He also opens $wc_y - ec_z - b$ to reveal 0.
4. V checks that both openings are valid and that the second opening indeed reveals 0.

3 Construction of Compact Proofs of Small Hamming Weight

In this section, we assume that the size of the field $\mathbb{F}$ in the commitments is exponential in the security parameter and hence also (much) larger than n. We will explain how to get rid of this assumption in Sect. 3.1.

We consider a prover who has committed to a vector of field elements $\boldsymbol{x} = (x_1, \ldots, x_n)$ and that wants to claim that the Hamming weight of $\boldsymbol{x}$ is at most d. The idea of the protocol is the following: We first choose distinct elements $a_1, \ldots, a_n \in \mathbb{F}$, and think of a_i as the "index" of the i'th position in the committed string. The way these indices are chosen is fixed in advance, i.e., part of the protocol specification, so that both parties can compute them on their own. In particular, for a field whose characteristic is no less than n, a_i can be simply chosen as i. Now, if the Hamming weight of $\boldsymbol{x}$ is at most d, there exists a monic polynomial of degree at most d whose zeros cover the set of indices a_i where $x_i \neq 0$. The prover is thus asked to prove the existence of such a polynomial $f(x)$ by committing to its coefficients, and then convince the verifier that $\sum_{i=1}^{n} f(a_i)x_i = 0$.

However, this approach fails if used naïvely: The above equation can be easily satisfied for an adversarially chosen $f(x)$, whose zeros might not even intersect with $\{a_i\}$, since the prover knows the x_i's when he chooses $f(x)$. Therefore, to ensure soundness, the committed vector $\boldsymbol{x}$ must be randomised appropriately after the polynomial has been fixed. Multiplying each x_i by independent random values chosen by the verifier will work but requires too much communication. It is possible to resolve this by replacing independent random values with a series of values generated by a secure PRG. The drawback of this method is that it makes the soundness only computational. Below, we propose another idea that uses less randomness while still giving us unconditional soundness.

Protocol Π_{HW}: The public input is the committed vector, $\langle x_1 \rangle, \ldots, \langle x_n \rangle$.

1. The prover commits to d field elements $f_0, \ldots, f_{d-1} \in \mathbb{F}$;
2. The verifier sends a random challenge $\beta \in \mathbb{F}$;
3. Both parties compute $\langle y_i \rangle = \beta^{i-1} \langle x_i \rangle$ for $i = 1, \ldots, n$ and

$$\langle z_j \rangle = \sum_{i=1}^{n} a_i^j \langle y_i \rangle \text{ for } j = 0, \ldots, d;$$

4. The prover commits to d field elements $g_0, \ldots, g_{d-1} \in \mathbb{F}$;
5. Both parties compute
$$\langle v \rangle = \langle z_d \rangle + \sum_{j=0}^{d-1} \langle g_j \rangle;$$
6. The prover proves that $g_j = f_j z_j$ for $j = 0, \ldots, d-1$ and that $v = 0$, using the subprotocols described in the preliminaries.

The theorem below makes the informal statement that Protocol Π_{HW} is complete, sound and zero-knowledge with respect to the statement that the Hamming weight of the committed vector is at most d. For Type 1 commitments this more formally means that it is a zero-knowledge proof system for the language consisting of commitments to a vector of Hamming weight at most d. We cannot use exactly the same formalization for Type 2 commitments since here both the prover and the verifier hold private information and there is no "public" commitment. Instead, we define soundness to mean that if the vector defined by the values sent to $\mathsf{F}_{\mathrm{Com}}$ has Hamming weight greater then d, then the verifier accepts with negligible probability. Completeness means, as usual, that if prover and verifier are honest, then the verifier accepts. Likewise, zero-knowledge means, as usual, that the verifier's view of the protocol with an honest prover can be simulated with (perfectly) indistinguishable distribution. In the proof below, we first give the proof for Type 1 commitments and then state the (minor) changes needed for Type 2.

Theorem 1. *Protocol Π_{HW} is complete, sound and zero-knowledge, with respect to the statement that the Hamming weight of the committed vector $\boldsymbol{x}$ does not exceed d.*

Proof. We prove the protocol satisfies the completeness, soundness and zero-knowledge properties. We define the following two polynomials

$$f(x) = x^d + \sum_{j=0}^{d-1} f_j x^j, \quad F(x) = \sum_{i=0}^{n-1} f(a_{i+1})\, x_{i+1} x^i,$$

which will be used in the proof.

Completeness. If the Hamming weight of $\boldsymbol{x}$ does not exceed d, or equivalently, there exists d indices $a_{i_1}, \ldots, a_{i_d}$ s.t. $x_i = 0$ for all $i \notin S = \{i_1, \ldots, i_d\}$, the prover should determine the values to commit to by

$$f(x) = \prod_{j=1}^{d} \left(x - a_{i_j}\right), \quad g_j = f_j z_j,$$

and behave as the protocol requires. Computing v, we find

$$\begin{aligned} v = z_d + \sum_{j=0}^{d-1} g_j &= \sum_{i=1}^{n} a_i^d \beta^{i-1} x_i + \sum_{j=0}^{d-1} \sum_{i=1}^{n} f_j a_i^j \beta^{i-1} x_i \\ &= \sum_{i=1}^{n} \left(a_i^d + \sum_{j=0}^{d-1} f_j a_i^j \right) \beta^{i-1} x_i = \sum_{i=1}^{n} f(a_i) \beta^{i-1} x_i \\ &= \sum_{i \in S} f(a_i) \beta^{i-1} x_i + \sum_{i \notin S} f(a_i) \beta^{i-1} x_i \\ &= \sum_{i \in S} 0 \cdot \beta^{i-1} x_i + \sum_{i \notin S} f(a_i) \beta^{i-1} \cdot 0 = 0, \end{aligned}$$

and the verifier will always accept.

Soundness. Note that $v = F(\beta)$ if $g_j = f_j z_j$ for $j = 0, \ldots, d-1$. Since the degree of $f(x)$ is (exactly) d, it has at most d zeros. If there are at least $d+1$ non-zero x_i's, $F(x)$ cannot be the zero polynomial. Now that $0 \leq \deg F(x) < n$, the probability that a random field element is a zero of $F(x)$ is at most $\frac{n-1}{|\mathbb{F}|}$. For the proof to be accepted, either one of the proofs produced by the subprotocols (to prove $g_j = f_j z_j$, $v = 0$) is false and accepted (each of which occurs with probability at most $|\mathbb{F}|^{-1}$), or β happens to be a zero of $F(x)$. Hence, by union bound and by our assumption on the size of $\mathbb{F}$, the verifier will reject with probability $1 - 2^{-\Omega(k)}$.

For Type 2 commitments, the only additional event that could make the verifier accept is that the prover manages to open any of the commitments in an incorrect way. But by assumption on Type 2 commitments, this occurs with exponentially small probability.

Zero-knowledge. We define a machine T that takes two oracles $\mathcal{O}_f, \mathcal{O}_g$, each of which provides d field elements. The machine T:

1. Starts an instance of the verifier;
2. Reads d field elements from $\mathcal{O}_f$ as $f_0, \ldots, f_{d-1}$;
3. Outputs $\langle f_j \rangle$ (committed with fresh randomness);
4. Reads β from the verifier;
5. Computes $\langle y_i \rangle$ and $\langle z_j \rangle$ as described in the protocol;
6. Reads d field elements from $\mathcal{O}_g$ as $g_0, \ldots, g_{d-1}$;
7. Outputs $\langle g_j \rangle$ (committed with fresh randomness);
8. Computes $\langle v \rangle$ as described in the protocol;
9. Runs the simulators for "proving" $g_j = f_j z_j$ and $v = 0$, and outputs the transcripts.

We will use some special oracles: $\mathcal{O}_f^{\text{real}}$ provides f_j's the honest prover uses; $\mathcal{O}_f^{\text{forged}}$ provides forged f_j's, just zeros, for instance. $\mathcal{O}_g^{\text{real}}$ and $\mathcal{O}_g^{\text{forged}}$ are defined similarly.

The simulator is defined as T taking $\mathcal{O}_f^{\text{forged}}, \mathcal{O}_g^{\text{forged}}$. We employ a standard hybrid argument to show that the simulator works. Consider the following distributions:

- D_1: the transcript created by the honest prover and the verifier;
- D_2: the transcript created by the honest prover and the verifier, but with simulated transcripts for the subprotocols; or equivalently, the transcript produced by T taking $\mathcal{O}_f^{\text{real}}, \mathcal{O}_g^{\text{real}}$;
- D_3: the transcript created by T taking $\mathcal{O}_f^{\text{real}}, \mathcal{O}_g^{\text{forged}}$;
- D_4: the transcript created by T taking $\mathcal{O}_f^{\text{forged}}, \mathcal{O}_g^{\text{forged}}$, or equivalently, that produced by the simulator.

Since the subprotocols are (honest-verifier) zero-knowledge, D_1 and D_2 are indistinguishable. The difference between D_2 and D_3 is whether g_j's contain real or forged values, and D_3 and D_4, f_j's. Since the commitment scheme is hiding, D_2, D_3 and D_3, D_4 are pairs of indistinguishable distributions, which follows from the definition of hiding by a standard computational reduction.

Formally, let D be an effective distinguisher telling D_2 from D_3, we build the following adversary that tries to break the hiding property:

1. The adversary makes a commitment to $\boldsymbol{x}$ and uses it as the public input; note that since the adversary knows $\boldsymbol{x}$, it is capable of implementing $\mathcal{O}_t^{\text{real}}$ $(t = f, g)$;
2. The adversary creates an oracle $\mathcal{O}_g^{\text{challenge}}$, which:
   (a) Runs $\mathcal{O}_g^{\text{real}}$ to produce g_j^{real}'s;
   (b) Runs $\mathcal{O}_g^{\text{forged}}$ to produce g_j^{forged}'s;
   (c) Sends the two batches to the challenger, and outputs whatever the challenger outputs;
3. The adversary runs T with $\mathcal{O}_f^{\text{real}}, \mathcal{O}_g^{\text{challenge}}$ to obtain a transcript;
4. It sends the transcript to D;
5. If D says the transcript is from D_2, the adversary concludes that the commitments the call to $\mathcal{O}_g^{\text{challenge}}$ received from the challenger are those of g_j^{real}'s; otherwise, those of g_j^{forged}'s.

D sees D_2 [resp. D_3] if $\mathcal{O}_g^{\text{challenge}}$ (the adversary) was given the commitments of g_j^{real}'s [resp. g_j^{forged}'s]. Therefore, the adversary has the same advantage against the hiding property as D has against D_2 and D_3. Moreover, the adversary is also effective. Since the commitment scheme is hiding, the adversary must have negligible advantage and so must D. A similar construction proves that D_3 and D_4 are also indistinguishable.

For Type 2 commitments, the argument becomes simpler: We define the oracles to output only what the verifier sees when a Type 2 commitment is created. Further, as these commitments are perfectly hiding, the forged and the real oracles now output exactly the same distribution, so we immediately get perfect zero-knowledge.

3.1 Field Extension

The basic protocol we just described does not work for small fields: we may not be able to choose n distinct values a_i, and even if we can, the field size may be too small to guarantee a small enough soundness error. In addition, we assumed the field was prime when defining Type 1 commitments.

We can solve both problems by going to an extension field $\mathbb{K}$, which we choose as a degree t extension of $\mathbb{F}$, so that $|\mathbb{K}|$ is exponential in the security parameter k. One possible value for t is $\left\lceil \frac{k}{\log |\mathbb{F}|} \right\rceil$.

Going from $\mathbb{F}$ to its extension $\mathbb{K}$ also requires enlarging G, H. For Type 2 commitments where these are vector spaces, this can be done using the tensor product, i.e., use $G' = G \otimes \mathbb{K}$ and $H' = H \otimes \mathbb{K}$, and induce the commitment schemes accordingly. Type 1 commitments can be extended in a similar manner. The following is a concrete explanation for extending Type 1 commitments. It also applies to Type 2 commitments, which is exactly the computational way of doing tensor products.

We have to fix a basis of $\mathbb{K}$ over $\mathbb{F}$ in advance. The new sets of randomness and commitments are $G' = G^t, H' = H^t$, in which additions are induced naturally. For all $b \in \mathbb{K}$ and $r = (r_1, \ldots, r_t)^{\mathrm{T}} \in G'$, we first find the matrix M_b of endomorphism $x \mapsto bx$ of $\mathbb{K}$ under the fixed basis, and define

$$br = M_b \begin{pmatrix} r_1 \\ \vdots \\ r_t \end{pmatrix},$$

where the multiplication on the right-hand side is formal and regarding elements in $\mathbb{F}$ as integers. Scalar multiplication in H' is defined similarly. For the induction of commitment algorithm, one simply commits to $a \in \mathbb{K}$ with randomness $r \in U'$ by committing coordinatewise. That is, let the coordinates of a under the fixed basis be $(a_1, \ldots, a_t)^{\mathrm{T}}$, we define

$$\mathsf{Commit}_{\mathrm{pk}}(a, r) = (\mathsf{Commit}_{\mathrm{pk}}(a_1, r_1), \ldots, \mathsf{Commit}_{\mathrm{pk}}(a_t, r_t))^{\mathrm{T}}.$$

The newly defined $\mathsf{Commit}_{\mathrm{pk}}$ is binding, hiding and additively homomorphic thanks to the commitment scheme of $\mathbb{F}$. Moreover, it is trivial to verify that the commitment scheme is capable of performing scalar multiplication, or precisely $b\mathsf{Commit}_{\mathrm{pk}}(a, r) = \mathsf{Commit}_{\mathrm{pk}}(ba, br)$ for all $a, b \in \mathbb{K}, r \in G'$, thus linearly homomorphic. For the q-inversion property, for all $c = (c_1, \ldots, c_t)^{\mathrm{T}}$, after a series of additions and multiplication by clear-text field elements, the resulting commitment is $d = Mc$ for some integer matrix M. Note well that modulo operation cannot be performed on M in between the operations. However, should d always contain 0, it must be the case that entries of M are multiples of $|\mathbb{F}|$, therefore, by the q-inversion property of the scheme in $\mathbb{F}$, along with its (additively) homomorphic property, we obtain a similar property that allows us to "explain" commitments that should always contain 0 as 0.

If the basis starts with 1 (the field identity), when we are given the input commitments over $\mathbb{F}$ $\langle x_1\rangle, \ldots, \langle x_n\rangle$ for $x_i \in \mathbb{F}$, we can easily modify these to commitments over $\mathbb{K}$ by appending $t-1$ default commitments to 0 to each $\langle x_i\rangle$ (the randomness input used for these commitments should be deterministic so that no communication overhead is incurred). We can then execute the main protocol exactly as described using $\mathbb{K}$ instead of $\mathbb{F}$ as the base field.

By moving to $\mathbb{K}$, we now get soundness error $2^{-\Omega(k)}$, and the complexity in terms of number of commitments over $\mathbb{F}$ sent is indeed $O\left(\frac{kd}{\log|\mathbb{F}|}\right)$ as promised in the introduction.

4 Applications

4.1 Actively Secure *d*-out-of-*n* Oblivious Transfer

In a d-out-of-n OT protocol, a sender has n messages, and a receiver wishes to learn exactly d of these, without revealing to the sender which messages were chosen. We consider the *non-adaptive* setting, where the receiver's d selections are chosen all at once, and refer to this functionality as $\binom{n}{d}$-OT_k, where the sender's messages are strings of length k (the security parameter).

Naor and Pinkas [34] showed how to construct $\binom{n}{d}$-OT in a black-box manner from $O(d \log n)$ instances of $\binom{2}{1}$-OT, however, their protocol is only secure in a half-simulation paradigm, and is vulnerable to selective failure attacks against a corrupt sender [12]. Another construction by Shankar et al. [41] uses only $O(n)$ 1-out-of-2 OTs, and an elegant mechanism based on secret-sharing to prevent the receiver from learning more than d messages. However, this is also not fully secure against a corrupt sender. The only known actively secure protocols are based on specific assumptions like DDH or pairings [12,24] and require $\Omega(n)$ public-key operations. These are inherently less efficient than constructions based on $\binom{2}{1}$-OT as they cannot make use of efficient OT extension techniques, which reduce the number of public key operations needed for OT to $O(k)$ (independent of the total number of OTs) [27].

We show how to use the proof of Hamming weight from Sect. 3 to construct an actively secure protocol for $\binom{n}{d}$-OT_k, which makes only black-box use of $\binom{2}{1}$-OT_k and symmetric primitives. The communication cost of the basic protocol is $O(kn+k^2d)$, or can be reduced to an amortized cost of $O(kn)$ in a batch setting, which is optimal up to a constant factor.

The Commitment Scheme. Our construction uses a specific form of Type 2 homomorphic commitment scheme defined by the functionality $\mathsf{F}_{\mathrm{Com}}$ below. Note that this is identical to the aBit functionality from [35] (optimized in [36]), only here we use it as a commitment scheme instead of for two-party computation. $\mathsf{F}_{\mathrm{Com}}$ can be efficiently implemented using black-box access to k oblivious transfers in a setup phase and a pseudorandom generator.

Functionality $\mathsf{F}_{\mathrm{Com}}$

On initialisation with the security parameter k, the functionality samples a random field element $\alpha \in \mathbb{F}_{2^k}$ and sends it to P_R.
On receiving a message $x \in \mathbb{F}_{2^k}$ from P_S:

1. Sample $\beta \in \mathbb{F}_{2^k}$ at random.
2. Send β to P_R and $\gamma := \alpha \cdot x + \beta$ to P_S.

To commit to a message x, the sender P_S sends x to $\mathsf{F}_{\mathrm{Com}}$. The verification algorithm for the receiver, P_R, takes as input a message x_i and the verification information $(\gamma_i, \alpha, \beta_i)$, then simply checks that $\gamma_i = \alpha \cdot x_i + \beta_i$.

The scheme is perfectly hiding, since the verifier's data α, β_i is uniformly random and independent of the sender's messages. The scheme is statistically binding, because opening to $x_i' \neq x_i$ requires coming up with $\gamma_i' = \alpha \cdot x_i' + \beta_i$, hence $\gamma_i' - \gamma_i = \alpha \cdot (x_i' - x_i)$, but this requires guessing α so happens with probability at most 2^{-k}. The scheme is also *linearly homomorphic* over $\mathbb{F}_{2^k}$, since if $f : \mathbb{F}_{2^k}^n \to \mathbb{F}_{2^k}$ is a linear map, then

$$f(\alpha \cdot x_1 + \beta_1, \ldots, \alpha \cdot x_n + \beta_n) = \alpha \cdot f(x_1, \ldots, x_n) + f(\beta_1, \ldots, \beta_n),$$

so applying f to the commitment and opening information results in a valid commitment to $f(x_1, \ldots, x_n)$.

The functionality $\mathsf{F}_{\mathrm{Com}}$ can be implemented using 1-out-of-2 string-OT, as shown in previous works for messages in $\{0,1\}$. To commit to a bit $x \in \{0,1\}$, the parties perform an OT where P_A is the sender with inputs $(\beta, \beta + \alpha)$, for randomly sampled $\alpha, \beta \in \mathbb{F}_{2^k}$, and P_B inputs the choice bit x. P_B receives $\gamma = \beta + x \cdot \alpha$, as required. To obtain active security, a consistency check is needed to ensure that the correct inputs are provided to the OTs. This can be done with only a small, constant overhead [36] using techniques based on OT extension [31,35].

We can extend the above to commit to arbitrary field elements instead of just bits using the homomorphic property, as follows. To commit to the field element $x \in \mathbb{F}_{2^k}$, first write x as $\sum_{i=1}^{k} x_i \cdot X^{k-1}$, for $x_i \in \mathbb{F}_2$, where the vector $(1, X, \ldots, X^{k-1})$ defines a basis of $\mathbb{F}_{2^k}$ over $\mathbb{F}_2$. Then, P_B commits to the individual bits x_i, obtaining commitments $\langle x_i \rangle$, and both parties then compute the commitment $\langle x \rangle = \sum_{i=1}^{k} \langle x_i \rangle \cdot X^{i-1}$.

Efficiency. Using the protocol from [36] (based on [35]), after a setup phase consisting of $O(k)$ OTs, the cost of committing to a bit is that of sending $O(k)$ bits, plus some computation with a PRG. To commit to an arbitrary field element we require k bit commitments, which gives a communication cost of $O(k^2)$.

***d*-out-of-*n* OT Protocol.** We now show how to realise $\binom{n}{d}$-OT_k using this commitment scheme, and applying the zero-knowledge proof of Hamming weight from Sect. 3. The idea is for the OT receiver to commit to a selection vector $(x_1, \ldots, x_n) \in \{0,1\}^n$ defining its d choices, and prove that at most d of these

are non-zero. Then, we use a hash function to convert the commitments to the x_i's into 1-out-of-2 OTs, where the second message in each OT is one of the sender's inputs. The zero-knowledge proof ensures that the receiver learns at most d of these inputs.

We use the definition of a *correlation robust* hash function $H : \mathbb{F}_{2^k} \to \{0,1\}^k$, which satisfies the following security property:

Definition 1 [27]. *Let $n = \mathsf{poly}(k)$ and $t_1, \ldots, t_n, \alpha$ be uniformly sampled from $\{0,1\}^k$. Then, H is* correlation robust *if the distribution*

$$(t_1, \ldots, t_n, H(t_1 \oplus \alpha), \ldots, H(t_n \oplus \alpha))$$

is computationally indistinguishable from the uniform distribution on $2nk$ bits.

Protocol: The receiver, P_R, has d choices $c_1, \ldots, c_d \in [n]$. The sender, P_S, inputs strings $y_1, \ldots, y_n \in \{0,1\}^k$.

1. P_R defines $\boldsymbol{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ to be the weight-d selection vector defined by P_R's choices.
2. The parties initialise $\mathsf{F}_{\mathrm{Com}}$, where P_S acts as receiver and obtains $\alpha \in \mathbb{F}_{2^k}$.
3. P_R commits to x_i using $\mathsf{F}_{\mathrm{Com}}$, for $i = 1, \ldots, n$, and receives γ_i. P_S receives the commitments β_i.
4. P_R proves that $w_H(\boldsymbol{x}) \leq d$ using Π_{HW}.
5. P_S sends to P_R the values

$$z_i = H(\beta_i + \alpha) \oplus y_i$$

6. P_R outputs $y_i = z_i \oplus H(\gamma_i)$, for the values where $x_i = 1$.

Theorem 2. *If H satisfies the correlation robustness property then the protocol above securely realises the $\binom{n}{d}$-OT functionality in the $\mathsf{F}_{\mathrm{Com}}$-hybrid model.*

Proof. We first consider the simpler case of a corrupt sender, P_S^*. The simulator, $\mathcal{S}$, sends random field elements α, β_i to simulate the outputs of $\mathsf{F}_{\mathrm{Com}}$ to P_S^*. $\mathcal{S}$ then runs the zero-knowledge simulator from Π_{HW}. Next, $\mathcal{S}$ receives the values z_i from P_S^* and recovers $y_i = z_i \oplus H(\beta_i + \alpha)$, for $i = 1, \ldots, n$. Finally, $\mathcal{S}$ sends the sender's inputs $y_1, \ldots, y_n$ to the $\binom{n}{d}$-OT functionality. It is easy to see that the simulation is identically distributed to the view of P_S^* in the real protocol, because the α, β_i values are sampled identically to the real protocol, and the zero-knowledge simulator for Π_{HW} is perfect when used with Type 2 commitments.

When the receiver, P_R^*, is corrupted, the simulator $\mathcal{S}$ proceeds as follows. First, $\mathcal{S}$ receives the bits $x_1, \ldots, x_n$ as the receiver's input to $\mathsf{F}_{\mathrm{Com}}$, and sends back random field elements $\gamma_1, \ldots, \gamma_n$. $\mathcal{S}$ simulates the verifier's messages in Π_{HW} with uniformly random values, and aborts if the proof fails. If the proof succeeds, then by the soundness property of Π_{HW} it holds that $w_H(\boldsymbol{x}) \leq d$, so $\mathcal{S}$ extracts d choices $c_1, \ldots, c_d \in \{1, \ldots, n\}$ from the non-zero entries of $\boldsymbol{x}$ (if $w_H(\boldsymbol{x}) < d$ then $\mathcal{S}$ chooses arbitrary indices for the last $d - w_H(\boldsymbol{x})$ choices). $\mathcal{S}$ sends $c_1, \ldots, c_d$ to $\binom{n}{d}$-OT, and receives back the strings $y_{c_1}, \ldots, y_{c_d}$. In the final

step, for the indices i where $x_i = 1$, $\mathcal{S}$ sends $z_i = H(\gamma_i) \oplus y_i$, and for all i where $x_i = 0$, $\mathcal{S}$ samples z_i uniformly from $\{0,1\}^k$.

Up until the final step, the simulation for a corrupt receiver is perfect, because the γ_i values are identically distributed to those output by $\mathsf{F}_{\mathrm{Com}}$, and Π_{HW} (and its subprotocols) are public-coin, so the verifier's messages are uniformly random. Regarding the z_i values, first note that whenever $x_i = 1$, P_R^* obtains the correct output y_i in both the real and simulated executions. When $x_i = 0$, the z_i's sent in the protocol are computationally indistinguishable from the simulated random values, by the correlation robustness property. More formally, suppose there exists an environment $\mathcal{Z}$ and an adversary $\mathcal{A}$, who corrupts the receiver, such that $\mathcal{Z}$ distinguishes the real and ideal executions. We construct a distinguisher D for the correlation robust function, as follows:

- D receives a correlation robustness challenge $(t_1, \ldots, t_n, u_1, \ldots, u_n)$.
- D invokes $\mathcal{Z}$ with the corrupt receiver, $\mathcal{A}$, starting a simulated execution of the d-out-of-n OT protocol. D receives the sender's inputs $y_1, \ldots, y_n$, chosen by $\mathcal{Z}$.
- Instead of sampling γ_i at random, D sends $t_1, \ldots, t_n$ to simulate these values sent to $\mathcal{A}$.
- For the indices i where $x_i = 0$, D lets $z_i = u_i \oplus y_i$. The rest of the execution is simulated the same way as $\mathcal{S}$.
- After the execution, D outputs the same as $\mathcal{Z}$.

Note that if the $u_1, \ldots, u_n$ values from the challenge are uniformly random, then the view of $\mathcal{Z}$ is identical to the view in the previous simulation. On the other hand, if $u_1, \ldots, u_n$ are computed as $H(t_i \oplus \alpha)$, for some random $\alpha \in \{0,1\}^k$, then $z_i = u_i \oplus y_i$ (where $x_i = 0$) is distributed the same as in the real protocol, so the view of $\mathcal{Z}$ is identical to the real execution. Therefore, D breaks the correlation robustness property of H with exactly the same probability that $\mathcal{Z}$ distinguishes the real and ideal executions.

Efficiency. The main cost of the protocol is the initial n calls to $\mathsf{F}_{\mathrm{Com}}$ to commit to the x_i bits, followed by running the proof Π_{HW}, which requires committing to $O(d)$ additional field elements. Since committing to a bit costs $O(k)$ bits of communication, and a field element $O(k^2)$, we get an overall communication complexity of $O(nk + k^2 d)$.

Reducing Communication with Amortization. In a batch setting, where two parties wish to perform multiple, parallel instances of $\binom{n}{d}$-OT_k, for the same values of d and n, we can reduce the amortized communication cost to $O(nk)$, which is optimal up to a constant factor. Instead of using the commitment scheme $\mathsf{F}_{\mathrm{Com}}$, we make use of recent advances homomorphic commitments based on OT and error-correcting codes [13,22]. These allow to commit to a vector of ℓ field elements with $o(1)$ communication overhead, for $\ell = \Omega(k)$. When performing many parallel executions of our protocol, this means steps 1–4 can be done with only $O(nk)$ amortized communication, instead of $O(nk + k^2 d)$. However, now we

have a problem in the final step where the sender hashes the commitments to transfer its inputs, because this is not compatible with the schemes of [13,22]. To get around this, the receiver will also commit to $x_1, \ldots, x_n$ using $\mathsf{F}_{\mathrm{Com}}$, and then prove that the two sets of commitments contain the same values. This can be shown by opening a (masked) random linear combination of the $\mathsf{F}_{\mathrm{Com}}$ commitments, then opening the same linear combination with the code-based commitments and checking that these give the same value. We give more details on this protocol and how to instantiate a Type 2 commitment scheme with code-based commitments in Appendix B.

4.2 Separable Accountable Ring Signatures

Ring signatures [40] enable a member of a group to leak information on behalf of the group without compromising its own identity. More precisely, ring signatures allow a signer to dynamically choose a group of potential signers and then sign a message on behalf of this group. A verifier can verify that the signature was indeed created by one of the group members, but cannot learn who the signer is. In [42], Xu and Yung introduce the notion of accountable ring signatures, where, in addition to a regular ring signature scheme, the signer can dynamically pick a trusted entity, called the opener, and, in addition to signing anonymously on behalf of the group, prove that this entity can revoke the signers anonymity. Accountable ring signatures imply traditional group signatures [7].

Since the members of a ring signatures are chosen dynamically, realistically speaking we can not always assume that all members use the same signing algorithm or even have the same type of public keys. Ideally, we would like to have a ring signature scheme, where we can sign on behalf of a group even if all members use different signing algorithms and different types of keys. This issue of *separability* has been first considered in the context of identity escrow [33] and later also in the context of group signatures [10,11]. Here, to the best of our knowledge, we provide the first construction of accountable ring signatures that achieves such separability. The only assumption we make on the public keys of the group members is that there exists a Σ-protocol for proving knowledge of the corresponding secret key.

Assume there are n parties $P_1, \ldots, P_n$, each holding a key pair $(\mathrm{pk}_i, \mathrm{sk}_i)$. Furthermore, assume that for each key pair, there is a Σ-protocol Σ_i to prove knowledge of the secret key sk_i corresponding to pk_i. Using an OR-proof [16] over all Σ_i, it is straightforward to prove knowledge of one of the secret keys while not revealing its own identity. Combining such an OR-proof with the Fiat-Shamir heuristic, we immediately get a separable ring signature scheme. To construct a separable accountable ring signature scheme, we additionally need to ensure that the designated opener, who has the key pair $(\mathrm{pk}_{\mathsf{op}}, \mathrm{td}_{\mathsf{op}})$, can extract the signer's identity from the Σ-protocol's transcript. Our main idea here is to "encode" the signer's identity into the protocol's challenge values and then use our compact proofs to prove that this has been done correctly. More concretely, recall that when an honest prover P_j does the OR-proof, there will be a Σ-protocol instance executed for each of the n parties. These will all be

simulated executions, except the j'th one. Now, we will interpret all challenges $e_1, \ldots, e_n$ in the Σ-protocols as commitments, and exploit the fact that P_j can choose all the simulated challenges as he likes, only e_j will be random. We can therefore instruct P_j to pick e_i where $i \neq j$ to be homomorphic commitments to 0. This means that $e_1, \ldots, e_n$, when seen as commitments, will represent a vector of Hamming weight at most 1, so P_j will prove this fact using our compact proof.

Assume we are using computationally hiding, perfectly binding, commitments. A (polynomial time) verifier cannot distinguish commitments to random bit strings from commitments to 0. Therefore, by the properties of Σ-protocols, the verifier cannot distinguish a simulated from a real transcript. The opener, who possesses a trapdoor $\mathrm{td}_{\mathsf{op}}$, can break the hiding property of the commitment scheme. That is, the opener can use $\mathrm{td}_{\mathsf{op}}$ to check whether a commitment contains a specific message, e.g. 0, or not. This is the case if, for example, the commitment scheme is actually a public-key encryption scheme. To identify a signer, the opener can open all challenge commitments and find the commitment to a non-zero value.

We will now describe our separable accountable ring signature scheme in the form of a group identification scheme with revocable anonymity. Combining this identification scheme with Fiat-Shamir then gives us our desired signature scheme. For a full formal definition of accountable ring signatures we refer the reader to [7].

Group identification scheme with revocable anonymity: Let $\mathtt{Encode}$ be a bijective function that maps elements from the commitment's message, randomness, and commitment space to bit strings. Let $\mathtt{Decode}$ be the inverse of $\mathtt{Encode}$. Let P_j be the prover and $\{P_1, \ldots, P_n\}$ the group. Let $\left(\mathrm{pk}_{\mathsf{op}}, \mathrm{td}_{\mathsf{op}}\right)$ be the opener's key pair for a perfectly binding, computationally hiding commitment scheme, where $\mathrm{td}_{\mathsf{op}}$ can be used to break the hiding property of a commitment.

Membership protocol

1. For $i \neq j$, the prover chooses uniformly random values r_i, computes $c_i = \mathsf{Commit}_{\mathrm{pk}_{\mathsf{op}}}(0, r_i)$, and encodes it as $e_i = \mathtt{Encode}(c_i)$. Next, for each e_i, the prover uses the simulator Σ_i to obtain transcripts (a_i, e_i, z_i). Finally, the prover chooses a random a_j according to Σ_j and sends $(a_1, \ldots, a_n)$ to the verifier.
2. The verifier chooses a random $x \neq 0$ and r and sends the challenge $e = \mathtt{Encode}(x, r)$ to the prover.
3. The prover computes $(x, r) = \mathtt{Decode}(e)$, picks commitment c_j such that $\sum_{i=1}^{n} c_i = \mathsf{Commit}_{\mathrm{pk}_{\mathsf{op}}}(x, r)$, and computes proof π for $w_H((c_1, \ldots, c_n)) \leq 1$ using Π_{HW}. Knowing a_j and $e_j = \mathtt{Encode}(c_j)$, the prover computes z_j honestly according to Σ_j. It sends $(c_1, \ldots, c_n)$, $(z_1, \ldots, z_n)$, and π to the verifier.
4. The verifier checks the validity of π, it checks that $\sum_{i=1}^{n} c_i = \mathsf{Commit}_{\mathrm{pk}_{\mathsf{op}}}(x, r)$, and finally it checks that for $1 \leq i \leq n$ each transcript (a_i, e_i, z_i) is an accepting transcript for Σ_i.

Anonymity Revocation. Given the transcript $\{(a_i, e_i, z_i)\}_{1 \leq i \leq n}$ of an invocation of the membership protocol described above, the opener can, for each i, compute $c_i = \mathtt{Decode}(e_i)$ and using his trapdoor $\mathrm{td}_{\mathsf{op}}$ it can reveal which commitment c_j is to a value not equal 0.

There are two things to note at this point. First, since the commitment scheme is *perfectly* binding, even an computationally unbounded opener can not open any of the commitments to any value other than the actually committed one. Secondly, the opener only needs to be able to distinguish commitments to 0 from commitments to any other value. In particular, this is a weaker requirement that recovering the exact committed message.

Security: In the following we provide an informal description of the security properties of accountable ring signatures. We sketch why our construction is secure according to these properties. The formal security definitions can be found in [7].

Full Unforgeability. From a high-level perspective, this property encompasses two security requirements. First, a corrupted opener cannot falsely accuse any member of a group of creating a signature. Second, no coalition of corrupted members in a ring can create an signature on behalf of an honest member.

Proof (sketch). Let $\sigma = (a_1, \ldots, a_n, e_1, \ldots, e_n, z_1, \ldots, z_n)$ be a valid signature created the adversary. Due to the (special) soundness of Π_{HW} we know that at most one commitment from $e_1, \ldots, e_n$ is not a commitment to 0. Let i be the index of the commitment that is not equal to 0 and j be the index of an honest member. Assume the opener accuses P_j of being the signer and consider the two following cases: If $i \neq j$, then the commitment c_j is a commitment to 0 and thus a malicious opener, who successfully accuses P_j would immediately contradict the binding property of the commitment scheme. In the case of $i = j$, the adversary successfully signed on behalf of an honest member P_j, which would contradict the (special) soundness of Σ_j.

Anonymity. This property ensures that nobody but the opener can reveal the identity of the ring member that created a signature. The anonymity property has to hold even when the secret keys of all members are revealed.

Proof (sketch). This property directly follows from the hiding property of the commitment scheme and the witness indistinguishability of the OR-proof construction.

Traceability. This property guarantees that the opener can always identify the signer and that the opener can provide a publicly verifiable proof thereof.

Proof (sketch). Let $\sigma = (a_1, \ldots, a_n, e_1, \ldots, e_n, z_1, \ldots, z_n)$ be a valid signature created by the adversary. Consider the following cases. If $\mathsf{HW}(c_1, \ldots, c_n) > 1$, then the adversary can be used to break the soundness property of Π_{HW}. In the case of $\mathsf{HW}(c_1, \ldots, c_n) = 1$, let i be the index of the commitment not equal to 0

and let P_j be the member that is accused by the opener. In this case either P_j was indeed the signer or we can use the adversary to break the soundness of Σ_j.

Tracing Soundness. This soundness property ensures that even if all members in a group and the opener are fully corrupt, the opener can still not accuse two different members of the ring.

Proof (sketch). This directly follows from the soundness of Π_{HW}.

4.3 More Efficient Preprocessing for the TinyTable Protocol

TinyTable [18] is a secure two-party computation protocol based on a 'gate scrambling' technique. It evaluates a circuit by expressing every non-linear gate with its truth table, and using a scrambled version of this table to perform the secure computation. This leads to a protocol in the preprocessing model with a very efficient online phase, where each non-linear gate requires just one message to be sent from each party, and linear gates can be evaluated without interaction. For small tables such as two-input AND gates, [18] showed to efficiently implement the preprocessing phase based on TinyOT [35], but for larger tables (such as representations of the S-boxes in 3-DES or AES) this approach does not scale well. Keller et al. [30] recently presented a more efficient approach to creating the masked tables using multiplication triples over a finite field of characteristic two. For the case of secure computation of AES, this gives a preprocessing phase that is almost as efficient as the best 2-party computation protocols based on garbled circuits, but with the benefits of the high throughput available in the TinyTable online phase.

We show how to further reduce the cost of the preprocessing phase, by combining our compact proof of Hamming weight with secret-shared finite field multiplications. Our approach requires just one multiplication triple per lookup table, whereas the previous method [30] needs at least $\log_2 N - 1$ triples for a table of size N (albeit over a smaller field). Our method concretely reduces the amount of communication needed for the preprocessing by around a factor of two, for lookup tables of size 32–64.

TinyTable Background. TinyTable uses linearly homomorphic, information-theoretic MACs to authenticate secret-shared data between the two parties.[3] The MACs are identical to our commitments produced by $\mathsf{F}_{\mathrm{Com}}$ in Sect. 4.1: the MACs on a shared value $x = x_1 + x_2$ are of the form $\gamma_{x_1} = x_1 \cdot \alpha_2 + \beta_{x_1}$ and $\gamma_{x_2} = x_2 \cdot \alpha_1 + \beta_{x_2}$, where P_A holds $(x_1, \gamma_{x_1}, \beta_{x_2}, \alpha_1)$ and P_B holds $(x_2, \gamma_{x_2}, \beta_{x_1}, \alpha_2)$. We use the notation $\langle x_1 \rangle_A$ and $\langle x_2 \rangle_B$ to denote these committed values held by P_A and P_B.

The goal of the preprocessing phase is to produce, for a public lookup table $T = (T[0], \ldots, T[n-1])$, the values:

$$\left(\langle s_i \rangle_i, \langle v_0^i \rangle_i, \ldots, \langle v_{n-1}^i \rangle_i\right)_{i \in \{A,B\}}$$

[3] TinyTable can also be extended to the multi-party setting [30], but here we focus on the two-party case.

where v_j^A, v_j^B are random shares that sum to $T[j \oplus s_A \oplus s_B]$, and s_A, s_B are random strings of length $\ell = \log_2 n$.

In [30], it was shown that it is enough for the parties to produce these values for the simple table where $T[0] = 1$ and $T[j] = 0$ for all $j > 0$. In other words, if the above shares satisfy $v_s^A + v_s^B = 1$ (where $s = s_A \oplus s_B$ is represented as an integer in $\{0, \ldots, n-1\}$), and $v_j^A + v_j^B = 0$ for all $j \neq s$, the parties can locally convert these shares into a scrambled table for *any* lookup table T of size n.

Preprocessing Protocol. We now show how to compute the above preprocessing data, using the Type 2 commitment scheme from Sect. 4.1 based on $\mathsf{F}_{\mathrm{Com}}$, and our proof of Hamming weight.

Additional Tools. Our protocol also requires the parties to be able to bit decompose committed values, and multiply secret-shared, committed values. Bit decomposition of a committed value $\langle x \rangle$, for $x \in \mathbb{F}_{2^k}$, can be done by first committing to the bits $\langle x_1 \rangle, \ldots, \langle x_k \rangle$, then opening $\langle x \rangle + \sum_i \langle x_i \rangle X^{i-1}$ and checking that this equals zero.

To produce a secret-sharing of the product of two committed values, where each value is held by a different party, we use a multiplication triple and Beaver's technique [4]. The current, most efficient methods of generating multiplication triples are based on oblivious transfer with the MASCOT [32] or Tiny-OLE [19] protocols. Note that these protocols create information-theoretic MACs on shares of the triples, but these MACs have the same form as the commitments produced by $\mathsf{F}_{\mathrm{Com}}$, so we can use them for our purpose.

With these building blocks, our protocol for preprocessing a masked lookup table of size n is as follows. We assume that $\mathsf{F}_{\mathrm{Com}}$ operates over the field $\mathbb{F}_{2^{2n}}$ and fix $(1, X, \ldots, X^{2n-1})$ as a basis over $\mathbb{F}_2$ of this field.

Protocol Π_{Prep}:

1. P_A samples a random, weight-one vector $(a_1, \ldots, a_n) \in \mathbb{F}_2^n$, and P_B samples $(b_1, \ldots, b_n)$ in the same way.
2. Both parties commit to the components of their vectors using $\mathsf{F}_{\mathrm{Com}}$, obtaining $\langle a_1 \rangle_A, \ldots, \langle a_n \rangle_A$ and $\langle b_1 \rangle_B, \ldots, \langle b_n \rangle_B$.
3. Compute $\sum_{i=1}^n \langle a_i \rangle_A$ and $\sum_i \langle b_i \rangle_B$ and check that these both open to 1.
4. Run Π_{HW} twice to prove that $w_H(\boldsymbol{a}) \leq 1$ and $w_H(\boldsymbol{b}) \leq 1$.
5. Let $\langle a \rangle_A = \sum_{i=1}^n \langle a_i \rangle_A \cdot X^{i-1}$ and $\langle b \rangle_B = \sum_{i=1}^n \langle b_i \rangle_B \cdot X^{i-1}$.
6. Using a random multiplication triple over $\mathbb{F}_{2^{2n}}$, compute commitments $\langle c^A \rangle_A$ and $\langle c^B \rangle_B$, such that $c^A + c^B = a \cdot b$.
7. For $j \in \{A, B\}$, bit decompose $\langle c^j \rangle_j$ to obtain $\langle c_1^j \rangle_j, \ldots, \langle c_{2n}^j \rangle_j$.
8. For $j \in \{A, B\}$, P_j outputs $(\langle c_1^j \rangle_j + \langle c_{n+1}^j \rangle_j, \ldots, \langle c_n^j \rangle_j + \langle c_{2n}^j \rangle_j)$.

Correctness and security. First note that the check that $\sum_i a_i = \sum_i b_i = 1$ rules out these vectors being all zero, therefore after Π_{HW} we know that they must have weight one. This means we can write the corresponding field elements as $a = X^r$ and $b = X^s$, where r and s represent the position of the one in each

party's random vector. Viewing these as elements of the larger field $\mathbb{F}_{2^{2n}}$, the product computed in step 6 then satisfies $c = X^{r+s}$, and has freshly random shares and MACs from the multiplication triple. The bit decomposition and computation in steps 7–8 then ensure that the output contains a one in position $r + s \pmod{n}$, and is zero elsewhere, as required.

Comparison with Other Approaches. The main cost in our protocol is that of generating one multiplication triple over $\mathbb{F}_{2^{2n}}$. In contrast, the protocol of [30] requires at least $\log_2 n - 1$ triples over a smaller field (depending on the table size, n). For example, if working over $\mathbb{F}_{2^{40}}$, [30] needs 4 triples for a table of size 32, but this increases to 7 triples when $n = 128$ and 11 when $n = 256$. We compare the communication complexity of our protocol with [30] in Table 1. The cost describes the total communication needed to generate enough triples for one masked table of size n, when using either the MASCOT [32] or TinyOLE [19] protocols for triple generation. For small tables of sizes 32–64, our protocol reduces the communication cost by around a factor of 2 compared with previous work. The reduction in communication seems more significant with TinyOLE, since MASCOT scales as $O(n^2)$ if n is the bit-length of the field, whereas TinyOLE is $O(n)$.

Table 1. Communication complexity, in kbits, of our protocol and the previous protocol when instantiated using MASCOT or TinyOLE to generate triples.

Protocol	$n = 32$	64	128	256	
[30]	279.0	348.8	488.3	767.4	MASCOT
Ours	139.3	360.4	917.8	2612	MASCOT
[30]	225.0	281.3	393.8	618.8	TinyOLE
Ours	90.0	180.0	360.0	720.0	TinyOLE

4.4 Shuffling in Public

Suppose that n parties wish to run a protocol in which each party inputs a message and the output is a (secret) permutation of the messages. This is called a *shuffle*. Of course, this shuffle could be executed by a trusted party. In absence of a trusted party, a *mixnet* [14] can be used. A mixnet consists of a number of servers and takes n ciphertexts as input. Each server permutes the ciphertexts, re-encrypts them, and hands them to the next server. If at least one server is honest, then the resulting permutation is unknown to an adversary. In addition, each server provides a *proof of correct shuffle* (e.g. [2,20]). Hence, each server needs to verify the correctness of all previous shuffles before applying its own, and only consider the correct shuffles.

In [1], Adida and Wikström presented a new approach to this problem: They show how to construct an obfuscated program for shuffling a set of n ciphertexts.

The obfuscated program P_π depends on a permutation π on n elements, but π should remain computationally hidden even given P_π. Obfuscating the shuffle has the advantage that it can be precomputed. Hence the parties only need to publish their encrypted messages and then compute the shuffle locally, while correctness of the shuffle can be verified in advance. Furthermore, the protocols enjoy public verifiability, i.e. the obfuscated program can be published together with a correctness proof that can be publicly verified.

The idea is that one takes ciphertexts $c_1, \ldots, c_n$ as input, generated in some appropriate cryptosystem, and processes them using P_π locally. If the shuffle is a re-encryption shuffle, then the output will be a re-encryption of the permuted messages to ciphertexts $c'_1, \ldots, c'_n$. If we let $m_1, \ldots, m_n$ and $m'_1, \ldots, m'_n$ denote the corresponding plaintexts, then the guarantee is that $m'_i = m_{\pi(i)}$ for $i = 1, \ldots, n$. The result can then be used for further computation. To obtain the messages, the parties can e.g. execute a distributed decryption protocol. In case of a decryption shuffle, the shuffle outputs the permuted messages directly.

The program constructed in [1] represents the shuffle as a permutation matrix. The obfuscated program has hence size roughly $O(n^2)$ ciphertexts and the correctness proof, using standard techniques as suggested by the authors, is of the same size. The program can only be used once, but on the other hand it is reasonably efficient and can be based on cryptosystems with only rather weak homomorphic properties. The authors propose three construction: The first one is a generic obfuscator for any somewhat homomorphic encryption (SHE) scheme allowing one multiplication and many additions. Such a scheme exists e.g. based on lattices (e.g. [8]) and pairings, e.g. the Boneh, Goh and Nissim cryptosystem [6]. However, the obfuscated program consists of double encryptions and hence distributed decryption with active security is expensive. The other two constructions avoid this problem by focussing on specific encryption schemes: the BGN cryptosystem for a decryption shuffle and Paillier encryption [37] (with some twists) for a re-encryption shuffle. Of course, one could also use fully homomorphic encryption, represent the permutation using only $O(n)$ ciphertexts and compute the permutations "inside" the encryption, but this would be completely impractical with current state of the art.

Another protocol for shuffling in public was proposed by Parampalli et al. [38]. The protocol computes an obfuscated re-encryption shuffle based on the Damgård-Jurik cryptosystem [17]. By using a permutation network to represent the shuffle, they could reduce the size of the obfuscated shuffle to $O(n \log n)$. The public proof of correctness has size $O(n \log n)$ using standard techniques. Due to the use of permutation networks, however, the resulting distribution over permutations may be biased, depending on the network that was used.

In the following, we will show how our techniques can be used to reduce the size of the public proof for the [1] BGN decryption shuffle to $O(n)$. Furthermore, we sketch an MPC protocol that outputs an obfuscated decryption shuffle together with a correctness proof.

Revisiting the BGN decryption shuffle. The obfuscated program P_π as constructed in [1] uses a public key pk for an SHE scheme as mentioned

above and consists of a matrix of ciphertexts $P_\pi = \{E_{\mathrm{pk}}(\Pi_{i,j})\}_{i,j=1\ldots n}$, where $\{\Pi_{i,j}\}_{i,j=1,\ldots,n}$ is the permutation matrix corresponding to π. It is now clear that one can apply π to a set of ciphertexts by multiplying the vector of input ciphertexts by the matrix P_π.

An obvious question from a practical point of view is of course who produces P_π in the first place, and how do we know it is correctly formed? In [1], it is suggested that P_π is produced by some secure multiparty protocol and that this protocol would also produce a zero-knowledge proof that anyone can verify that P_π is correctly formed. For this, they used existing techniques for proving correctness of shuffles, basically doing such a proof for each row (column) of the matrix. This means that the proof would typically have size $O(n^2)$. Using our techniques we can improve this to $O(n)$ as we now explain:

First, we can observe that the BGN cryptosystem can be seen as an unconditionally binding and homomorphic commitment scheme based on which our protocol can run. The proof then consists of two parts: First, show that in each column and each row, the sum of all entries is 1. This can be done by computing the product of ciphertexts across each column and row of P_π and prove using standard methods that each such product contains 1. Second, we use our protocol to show that the weight of each row is at most 1. Combined with the first step, we obtain now that each column and each row has weight exactly 1. These proofs can be made non-interactive using Fiat-Shamir paradigm and will clearly imply that the matrix underlying P_π is indeed a permutation matrix.

Finally, we sketch how to generate the obfuscated program and proof of correctness in a multiparty protocol. The BGN cryptosystem uses a group of order $N = q_1 q_2$ where q_1, q_2 are primes. Therefore it is convenient to use an MPC protocol based on linear secret sharing modulo N. This will mean that given a secret-shared representation of a message m, which we will denote $[m]$, it is easy using standard methods to securely generate an encryption $E_{\mathrm{pk}}(m)$ where pk is the BGN public key. It is therefore sufficient to generate secret shared values corresponding to a permutation matrix $[\Pi_{i,j}]$. This can be done, for instance, if each party (verifiably) secret shares his own permutation matrix, and then we multiply these using standard matrix multiplication. Generating the proof of correctness is standard for the most part, by simply emulating the prover's algorithm. Whenever the original prover would output a commitment, we will have a secret-shared representation of the same value, which we can convert to a BGN encryption (commitment) as we go. One slightly non-standard detail is that given the i'th row $\{[\Pi_{i,j}]\}_{j=1,\ldots,n}$, we want to show it has weight at most 1 and for this we need a secret shared representation of the (unique) index j_0 where $\Pi_{i,j_0} = 1$. But this we can get easily by forming the row $[1], [2], \ldots, [n]$ and computing the inner product with the row $\{[\Pi_{i,j}]\}_{j=1,\ldots,n}$.

4.5 PIR for Malicious Users

Consider a very simple folklore PIR protocol based on additively homomorphic encryption, e.g. Paillier, where a user wishes to retrieve single elements. Assume that the database holds elements $d_1, \ldots, d_n$. To retrieve a data element j from

the database, the user could send ciphertexts $c_1, \ldots, c_n$ to the database of which at most one contains a non-zero message, namely j. The database can then compute a new ciphertext $d = \sum_{i=1}^{n} c_i d_i$ corresponding to the selected element and return d to the user. Finally, the user can decrypt d to obtain the selected element d_j.

It is easy to see that this protocol has passive security. To achieve security against a malicious user, one can add our protocol (interactive or non-interactive) to prove that the user's first message to the database is well-formed.

Note that using fully homomorphic encryption, one can get an incomparable solution where the client sends only a single ciphertext containing the index of the entry he wants (j). The server can now compute, "inside the encryption", a ciphertext that contains d_j and send it back to the client. This requires much less communication but cannot be implemented based on only additively homomorphic encryption, and has a very large computational overhead compared to the more standard solution (note that in any solution the server must touch all entries in the database, or the scheme is not secure).

Acknowledgements. This work has been supported by the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No. 669255 (MPCPRO); the European Union's Horizon 2020 research and innovation programme under grant agreement No. 731583 (SODA); and the Danish Independent Research Council under Grant-ID DFF–6108-00169 (FoCC).

A Considerations for Paillier Construction

In Paillier construction [37] where the clear text space $\mathbb{Z}_N$ is not a field, some properties we employ in the construction might not hold. On a field, a polynomial of degree d has at most d zeros, while on a general ring, this is not true. For the special case $\mathbb{Z}_N$ where $N = pq$ is the product of two distinct primes p, q, we resort to the factorisation assumption.

Factorisation Assumption. Let $N = pq$ where p, q are distinct, uniformly random primes of length $\Omega(k)$. For all probabilistic polynomial time adversary A, $\Pr[A(N) = p \text{ or } A(N) = q]$ is negligible in k.

It is well known that if RSA is secure, the above assumption holds. We need two tweaks in the proof for the soundness of Π_{HW} instantiated with Paillier commitment schemes.

Malicious $f(x)$. In the protocol the prover selects a monic polynomial $f(x)$ of degree d. We say such a polynomial is malicious if it has at least $d + 1$ distinct zeros on the index set $\{a_i\}$. The factorisaton of N can be reduced to finding a malicious polynomial, therefore, the probability that a cheating prover succeeds commiting to a malicious $f(x)$ is negligible.

Proof (sketch). Observe that monic linear polynomials cannot be malicious. Let $f(x)$ be malicious, of degree $d > 1$ and $x_0, \ldots, x_d$ its $d+1$ known, distinct roots. By division with remainder, we have $f(x) = (x - x_0)g(x)$ for some monic polynomial $g(x)$ of degree $d-1$. Consider $\gcd(N, x_j - x_0)$ $(j = 1, \ldots, d)$, if one of them is not 1, it must be p or q, giving the factorisation of N. Otherwise, $x_j - x_0$ $(j = 1, \ldots, d)$ are invertible in $\mathbb{Z}_N$. Substituting $x_1, \ldots, x_d$ into the equation of division, we conclude that $g(x)$ is malicious, of degree $d-1$ and $x_1, \ldots, x_d$ are its d distinct zeros. We then continue this process with $g(x)$. However, the process must stop before reaching linear polynomials by the observation, finding either p or q.

Weak $F(x)$. The protocol verifies $f(a_i)x_i = 0$ with a "checksum" polynomial $F(x)$ whose coefficients are $f(a_i)x_i$, where we exploit the property that $F(x)$ has at most n zeros if $F(x) \neq 0$. In $\mathbb{Z}_N$, $F(x)$ of degree n could have at most $\max\{pn, qn\}$ distinct roots. This bound still guarantees asymptotic soundness, but is a great sacrifice of the concrete soundness error. By assuming the hardness of factorisation, we can prove a better bound. We define a polynomial $F(x)$ on $\mathbb{Z}_N$ of degree n to be weak, if it has more than n^2 distinct zeros. We shall show that with negligible probability, the $F(x)$ used in the protocol is weak.

Proof. By Chinese Remainder Theorem, $\mathbb{Z}_N = \mathbb{Z}_p \times \mathbb{Z}_q$. For $x_0 \in \mathbb{Z}_N$, write $x_0 = (y_0, z_0)$ by this decomposition, where $y_0 \in \mathbb{Z}_p, z_0 \in \mathbb{Z}_q$. We can naturally regard $F(x)$ as polynomial $F_p(x)$ on $\mathbb{Z}_p$ or $F_q(x)$ in $\mathbb{Z}_q$ by keeping only the relevant component (coefficients modulo the corresponding prime). It is trivial to verify that $F(x_0) = (F_p(y_0), F_q(z_0))$ and that $F(x_0) = 0$ is equivalent to $F_p(y_0) = 0$ and $F_q(z_0) = 0$. If neither $F_p(x)$ nor $F_q(x)$ is the zero polynomial, both of them have at most n distinct roots. In such case, $F(x)$ has at most n^2 roots as the set of roots of $F(x)$ is exactly the Cartesian product of the sets of roots of $F_p(x)$ and $F_q(x)$. Otherwise, suppose $F_p(x)$ is the zero polynomial, the coefficients of $F(x)$ are multiples of p, while at least one of them is not a multiple of N. Computing the greatest common divisor of the coefficients of $F(x)$ gives p, factorising N. Similar argument applies to the case $F_q(x)$ is zero. Note that the prover is able to find the coefficients of $F(x)$ himself, therefore the $F(x)$ used in the protocol is weak with negligible probability.

Combining the two tweaks ensures the soundness of the instantiation of Π_{HW} with Paillier commitment schemes. Complete and zero-knowledge properties follow by the general proof presented in the text.

It is also noticeable that the method for extension does not work with $\mathbb{Z}_N$. Therefore, N must be large enough for the construction to be sound, which is, after all, true for practical scenarios.

B Details on Code-Based Homomorphic Commitments

In this section we provide more details on instantiating our protocols using recent, UC-secure homomorphic commitment schemes, and using this to reduce the cost of batch d-out-of-n OT.

B.1 The Type 2 Commitment Scheme

We now show how to instantiate Type 2 commitments with efficient, rate-1 homomorphic commitment schemes based on 1-out-of-2 OT and error-correcting codes. The commitment functionality, $\mathsf{F}^*_{\mathrm{Com}}$, is given below. We first show how this gives a Type 2 commitment scheme where the message space is $\mathbb{F}^\ell$ instead of $\mathbb{F}$, and then discuss how existing homomorphic commitment schemes [13,22] can be used to realise this functionality.

Functionality $\mathsf{F}^*_{\mathrm{Com}}$

Parameters: $\mathbb{F}$, a finite field; ℓ, the message length; C, an $[m, \ell, s]$ linear code over $\mathbb{F}$, where s is the security parameter.
On initialisation with the public parameters, the functionality samples a random $\alpha = (\alpha_1, \ldots, \alpha_m) \in \{0,1\}^m$ and sends it to P_A.
On receiving a message $x \in \mathbb{F}^\ell$ from P_B:

1. Sample $\beta \in \mathbb{F}^m$ at random.
2. Send β to P_A and $\gamma := \alpha * C(x) + \beta$ to P_B, where $*$ denotes component-wise product.

Leakage: If P_B is corrupt, the adversary may send any number of key queries of the form $(\texttt{guess}, i, b_i)$. If $b_i = \alpha_i$ then send $(\texttt{success})$ to the adversary, otherwise send $(\texttt{abort})$ to all parties and terminate.

To verify a commitment to x with the opening information (α, β, γ), P_A checks that $\gamma = \alpha * C(x) + \beta$.

Clearly, the scheme is unconditionally hiding as with $\mathsf{F}_{\mathrm{Com}}$. To see the statistical binding property, notice that to forge an opening of a commitment to x, P_B must come up with $x' \neq x$ and $\gamma' \in \mathbb{F}^m$ such that $\gamma' = \alpha * C(x') + \beta$. We then define $\delta := \gamma - \gamma' = \alpha * C(x - x')$, by linearity of the code. Since $x \neq x'$ and C has minimum distance s, the Hamming weight of $C(x - x')$ is at least s, so coming up with such a δ requires guessing at least s bits of α, with probability $\leq 2^{-s}$. Note that including the key queries in $\mathsf{F}^*_{\mathrm{Com}}$ does not change the overall success probability, since for each query a single bit of α can be guessed only with probability $1/2$, and the functionality aborts if any query fails.

This functionality can be realised from the commitment phase of [22] or [13]. To see this, recall that after the commitment phase in these protocols, the sender holds a committed message $\boldsymbol{x} \in \mathbb{F}_2^\ell$, and a random additive sharing of $C(\boldsymbol{x})$, where C is a linear $[m, \ell, s]$ error-correcting code. Meanwhile, for each component of $C(\boldsymbol{x})$, the receiver holds exactly one of the two shares. That is, the sender has two vectors $\boldsymbol{y}_0, \boldsymbol{y}_1 \in \mathbb{F}^m$ such that $\boldsymbol{y}_0 + \boldsymbol{y}_1 = C(\boldsymbol{x})$, whereas the receiver holds a random secret vector $(r_1, \ldots, r_n) \in \{0,1\}^m$, which is fixed once for all the

commitments. For the commitment to $\boldsymbol{x}$, the receiver knows a vector $\boldsymbol{z}$ satisfying $\boldsymbol{z}[i] = \boldsymbol{y}_{r_i}[i]$, from the 1-out-of-2 OT setup phase. Notice that:

$$\begin{aligned}\boldsymbol{z}[i] = \boldsymbol{y}_{r_i}[i] &= (\boldsymbol{y}_0 \cdot (1 + r_i) + \boldsymbol{y}_1 \cdot r_i)[i] \\ &= (\boldsymbol{y}_0 + r_i \cdot (\boldsymbol{y}_0 + \boldsymbol{y}_1))[i] \\ &= (\boldsymbol{y}_0 + r_i \cdot C(\boldsymbol{x}))[i]\end{aligned}$$

This is clearly the same form as the commitments produced by $\mathsf{F}^*_{\mathrm{Com}}$, since we have $\boldsymbol{z} = \boldsymbol{y}_0 + \boldsymbol{r} * C(\boldsymbol{x})$.

Note that $\mathsf{F}^*_{\mathrm{Com}}$ also allows a corrupt sender to attempt to guess the bits of $\boldsymbol{r}$, but aborts if any guess fails. This is needed because the consistency check in [13], used to ensure the sender inputs correct codewords, may leak a few of these bits to a cheating sender. This can be seen from the proof of Lemma 8, where the exact set of bits of $\boldsymbol{r}$ which the sender attempts to guess is defined. That proof can be applied directly to show that the commitment phase of Protocol Π_{HCOM} from [13] can be used to securely realise $\mathsf{F}^*_{\mathrm{Com}}$. Finally, we remark that although the protocol in [13] is defined over the field $\mathbb{F}_2$, it can be used to commit to vectors over any finite field with a suitable error-correcting code, and the communication complexity is still $O(m)$ field elements per commitment.

B.2 Switching Between Schemes

As we will see in the application to d-out-of-n OT in the batch setting (described in the full version of this paper), it can be useful to use the most efficient, rate-1 homomorphic commitments for the most expensive part of a protocol, before switching to another homomorphic commitment scheme that is more suited to the application. This can be done by committing to the messages with both schemes and then proving that both sets of commitments contain the same messages. With the Type 2 schemes $\mathsf{F}_{\mathrm{Com}}$ and $\mathsf{F}^*_{\mathrm{Com}}$, this proof works as follows (and the same technique can be adapted for any scheme).

Protocol $\boldsymbol{\Pi_{\mathsf{EQ}}}$: The input is two sets of committed vectors $\langle \boldsymbol{x}_1 \rangle^*, \ldots, \langle \boldsymbol{x}_n \rangle^*$ and $\{\langle y_1^i \rangle, \ldots, \langle y_n^i \rangle\}_{i=1}^{\ell}$, where $\langle \cdot \rangle^*$ denotes a commitment to an element of $\mathbb{F}^\ell$ with $\mathsf{F}^*_{\mathrm{Com}}$ and $\langle \cdot \rangle$ a commitment using $\mathsf{F}_{\mathrm{Com}}$ over $\mathbb{F}$. We prove that $\boldsymbol{x}_j[i] = y_j^i$ for all i, j.

1. The proves samples at random and commits to $\boldsymbol{r} = (r_1, \ldots, r_\ell) \in \mathbb{F}^\ell$ with both schemes, obtaining commitments $\langle \boldsymbol{r} \rangle^*, \langle r_1 \rangle, \ldots, \langle r_\ell \rangle$.
2. The verifier sends a random challenge $s \in \mathbb{F}$.
3. The prover opens $\langle \boldsymbol{a} \rangle^* = \sum_{j=1}^{n} \langle \boldsymbol{x}_j \rangle^* \cdot s^j + \langle \boldsymbol{r} \rangle^*$. Write $\boldsymbol{a} = (a_1, \ldots, a_\ell)$.
4. The prover opens $\langle b_i \rangle = \sum_{j=1}^{n} \langle y_j^i \rangle \cdot s^j + \langle r_i \rangle$, for $i = 1, \ldots \ell$.
5. The verifier checks that $a_i = b_i$ for all i.

Completeness is evident, and zero-knowledge holds because the values r_i are uniformly random and used to mask the opened values as one-time pads. To argue soundness, note that if the proof succeeds then we have $a_i = b_i$, and so

$\sum_{j=1}^{n}(\boldsymbol{x}_j[i] - y_j^i) \cdot s^j = 0$. However, if the committed inputs were not the same then there is at least one pair i, j such that $\boldsymbol{x}_j[i] \neq y_j^i$. This means that the probability of success is at most $n/|\mathbb{F}|$, since it corresponds to the degree n polynomial with coefficients $(\boldsymbol{x}_j[i] - y_j^i)_j$ having a root at s.

Finally, we remark that the communication cost of the protocol is independent of n, since it is $O(k^2\ell)$ bits, dominated by committing to the elements $r_1, \ldots, r_\ell$ (assuming $|\mathbb{F}| = 2^k$).

References

1. Adida, B., Wikström, D.: How to shuffle in public. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 555–574. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70936-7_30
2. Bayer, S., Groth, J.: Efficient zero-knowledge argument for correctness of a shuffle. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 263–280. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_17
3. Bayer, S., Groth, J.: Zero-knowledge argument for polynomial evaluation with application to blacklists. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 646–663. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_38
4. Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 420–432. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_34
5. Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., Virza, M.: SNARKs for C: verifying program executions succinctly and in zero knowledge. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 90–108. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_6
6. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30576-7_18
7. Bootle, J., Cerulli, A., Chaidos, P., Ghadafi, E., Groth, J., Petit, C.: Short accountable ring signatures based on DDH. In: Pernul, G., Ryan, P.Y.A., Weippl, E. (eds.) ESORICS 2015. LNCS, vol. 9326, pp. 243–265. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24174-6_13
8. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_29
9. Brands, S., Demuynck, L., De Decker, B.: A practical system for globally revoking the unlinkable pseudonyms of unknown users. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 400–415. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73458-1_29
10. Camenisch, J., Damgård, I.: Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 331–345. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44448-3_25

11. Camenisch, J., Michels, M.: Separability and efficiency for generic group signature schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 413–430. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_27
12. Camenisch, J., Neven, G., Shelat, A.: Simulatable adaptive oblivious transfer. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 573–590. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72540-4_33
13. Cascudo, I., Damgård, I., David, B., Döttling, N., Nielsen, J.B.: Rate-1, linear time and additively homomorphic UC commitments. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9816, pp. 179–207. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53015-3_7
14. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM **24**(2), 84–88 (1981). http://doi.acm.org/10.1145/358549.358563
15. Cramer, R., Damgård, I.: Zero-knowledge proofs for finite field arithmetic, or: can zero-knowledge be for free? In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 424–441. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0055745
16. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48658-5_19
17. Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44586-2_9
18. Damgård, I., Nielsen, J.B., Nielsen, M., Ranellucci, S.: The tinytable protocol for 2-party secure computation, or: gate-scrambling revisited. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 167–187. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_6
19. Döttling, N., Ghosh, S., Nielsen, J.B., Nilges, T., Trifiletti, R.: TinyOLE: efficient actively secure two-party computation from oblivious linear function evaluation. In: ACM Conference on Computer and Communications Security, CCS 2017 (2017)
20. Fauzi, P., Lipmaa, H., Zając, M.: A shuffle argument secure in the generic model. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 841–872. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53890-6_28
21. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_12
22. Frederiksen, T.K., Jakobsen, T.P., Nielsen, J.B., Trifiletti, R.: On the complexity of additively homomorphic UC commitments. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9562, pp. 542–565. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49096-9_23
23. Giacomelli, I., Madsen, J., Orlandi, C.: ZKBoo: faster zero-knowledge for Boolean circuits. In: USENIX Security Symposium, pp. 1069–1083. USENIX Association (2016)
24. Green, M., Hohenberger, S.: Universally composable adaptive oblivious transfer. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 179–197. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89255-7_12
25. Groth, J.: Linear algebra with sub-linear zero-knowledge arguments. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 192–208. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_12

26. Groth, J., Kohlweiss, M.: One-out-of-many proofs: or how to leak a secret and spend a coin. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 253–280. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_9
27. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 145–161. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_9
28. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Johnson, D.S., Feige, U. (eds.) 39th ACM STOC, pp. 21–30. ACM Press, June 2007
29. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer – efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_32
30. Keller, M., Orsini, E., Rotaru, D., Scholl, P., Soria-Vazquez, E., Vivek, S.: Faster secure multi-party computation of AES and DES using lookup tables. In: Gollmann, D., Miyaji, A., Kikuchi, H. (eds.) ACNS 2017. LNCS, vol. 10355, pp. 229–249. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61204-1_12
31. Keller, M., Orsini, E., Scholl, P.: Actively secure OT extension with optimal overhead. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 724–741. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_35
32. Keller, M., Orsini, E., Scholl, P.: MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016, pp. 830–842 (2016)
33. Kilian, J., Petrank, E.: Identity escrow. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 169–185. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0055727
34. Naor, M., Pinkas, B.: Computationally secure oblivious transfer. J. Cryptol. **18**(1), 1–35 (2005)
35. Nielsen, J.B., Nordholt, P.S., Orlandi, C., Burra, S.S.: A new approach to practical active-secure two-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 681–700. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_40
36. Nielsen, J.B., Schneider, T., Trifiletti, R.: Constant round maliciously secure 2PC with function-independent preprocessing using LEGO. In: 24th NDSS Symposium. The Internet Society (2017). http://eprint.iacr.org/2016/1069
37. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_16
38. Parampalli, U., Ramchen, K., Teague, V.: Efficiently shuffling in public. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 431–448. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30057-8_26
39. Rindal, P., Rosulek, M.: Improved private set intersection against malicious adversaries. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10210, pp. 235–259. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_9
40. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_32

41. Shankar, B., Srinathan, K., Rangan, C.P.: Alternative protocols for generalized oblivious transfer. In: Rao, S., Chatterjee, M., Jayanti, P., Murthy, C.S.R., Saha, S.K. (eds.) ICDCN 2008. LNCS, vol. 4904, pp. 304–309. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-77444-0_31
42. Xu, S., Yung, M.: Accountable ring signatures: a smart card approach. In: Quisquater, J.J., Paradinas, P., Deswarte, Y., El Kalam, A.A. (eds.) Smart Card Research and Advanced Applications VI. IFIPAICT, vol. 153, pp. 271–286. Springer, Boston (2004). https://doi.org/10.1007/1-4020-8147-2_18