Amortized Complexity of Information-Theoretically Secure MPC Revisited

Ignacio Cascudo[1(✉)], Ronald Cramer[2,3], Chaoping Xing[4], and Chen Yuan[2]

[1] Aalborg University, Aalborg, Denmark
ignacio@math.aau.dk
[2] CWI Amsterdam, Amsterdam, The Netherlands
{cramer,Chen.Yuan}@cwi.nl
[3] Leiden University, Leiden, The Netherlands
cramer@math.leidenuniv.nl
[4] Nanyang Technological University, Singapore, Singapore
xingcp@ntu.edu.sg

Abstract. A fundamental and widely-applied paradigm due to Franklin and Yung (STOC 1992) on Shamir-secret-sharing based general n-player MPC shows how one may trade the *adversary threshold* t against *amortized* communication complexity, by using a so-called packed version of Shamir's scheme. For e.g. the BGW-protocol (with active security), this trade-off means that if $t + 2k - 2 < n/3$, then k *parallel* evaluations of the *same* arithmetic circuit on different inputs can be performed at the overall cost corresponding to a *single* BGW-execution.

In this paper we propose a novel paradigm for amortized MPC that offers a *different* trade-off, namely with the size of the field of the circuit which is securely computed, instead of the adversary threshold. Thus, unlike the Franklin-Yung paradigm, this leaves the adversary threshold *unchanged*. Therefore, for instance, this paradigm may yield constructions enjoying the maximal adversary threshold $\lfloor (n-1)/3 \rfloor$ in the BGW-model (secure channels, perfect security, active adversary, synchronous communication).

Our idea is to compile an MPC for a circuit over an extension field to a parallel MPC of the same circuit but with inputs defined over its *base field* and with the *same* adversary threshold. Key technical handles are our notion of *reverse multiplication-friendly embeddings* (RMFE) and our proof, by algebraic-geometric means, that these are *constant-rate*, as well as efficient auxiliary protocols for creating "subspace-randomness" with good amortized complexity. In the BGW-model, we show that the latter can be constructed by combining our tensored-up linear secret sharing with protocols based on hyper-invertible matrices á la Beerliova-Hirt (or variations thereof). Along the way, we suggest alternatives for hyper-invertible matrices with the same functionality but which can be defined over a large enough constant size field, which we believe is of independent interest.

As a demonstration of the merits of the novel paradigm, we show that, in the BGW-model and with an optimal adversary threshold $\lfloor (n-1)/3 \rfloor$,

H. Shacham and A. Boldyreva (Eds.): CRYPTO 2018, LNCS 10993, pp. 395–426, 2018.
https://doi.org/10.1007/978-3-319-96878-0_14

it is possible to securely compute a *binary circuit* with amortized complexity $O(n)$ of *bits per gate per instance*. Known results would give $n \log n$ bits instead. By combining our result with the Franklin-Yung paradigm, and assuming a *sub-optimal adversary* (i.e., an arbitrarily small $\epsilon > 0$ fraction below $1/3$), this is improved to $O(1)$ bits instead of $O(n)$.

1 Introduction

A fundamental and widely-applied paradigm due to Franklin and Yung [FY92] on Shamir-secret-sharing based general n-player MPC shows how one may trade the *adversary threshold* t against *amortized* communication complexity, by using a so-called packed version of Shamir's scheme. For e.g. the BGW-protocol [BGW88] (with active security), this trade-off means that if $t+2k-2 < n/3$, then k *parallel* evaluations of the *same* arithmetic circuit on different inputs can be performed at the overall cost corresponding to a *single* BGW-execution. In this paper we propose a novel paradigm for amortized MPC that offers a *different* trade-off, namely with the size of the field of the circuit which is securely computed, instead of the adversary threshold. In particular, unlike the Franklin-Yung paradigm, this leaves the adversary threshold *unchanged*.

We apply our paradigm in the *BGW-model*: secure channels, perfect security (privacy and correctness), active adversary, synchronous communication. Our aim is to achieve MPC that is efficient (in the amortized sense as discussed above), tolerates an adversary satisfying the maximal threshold (or close) and that evaluates *binary circuits*.

We motivate the latter choice as follows. Besides the fact that this is natural in applications to begin with, we note that many of the protocols in this model (such as [BGW88]) represent the target function to be computed as an arithmetic circuit over some finite field, and then process this circuit by distributed computing of each gate, using secret sharing. However, many of those protocols require that the size of the underlying finite field is larger the number of parties n (because of their use of Shamir's secret sharing scheme [Sha79] of some variant thereof). This means that applying those protocols to functions which are already naturally represented as a binary circuit requires to lift this circuit to a large enough extension field, wherewith the communication complexity incurs a multiplicative overhead of $\log n$. It is in these cases that our paradigm pays off by twisting this overhead into a vehicle for parallel evaluations.

Concretely, we get:

Theorem 1. *In the BGW-model, there is an efficient MPC protocol for n parties secure against the maximal number of active corruptions $\lfloor (n-1)/3 \rfloor$ that computes $\Omega(\log n)$ evaluations of a single binary circuit in parallel with an amortized communication complexity (per instance) of $O(n)$ bits per gate.*

The best known previous result of this kind and in this model is obtained from the MPC protocol by Beerliová-Trubíniová and Hirt [BH08] which communicates $O(n)$ *field elements* per gate, but requires the field size over which the arithmetic

circuit is defined to be at least $2n$, and hence the computation of a binary circuit with that protocol requires $O(n \log n)$ bits of communication per gate. Our result will be proved by applying our paradigm to Beerliova-Hirt. Note that the Franklin-Yung paradigm does not apply here as we achieve security against a maximal adversary. *Combining* our result with the Franklin-Yung paradigm, however, we get the following:

Theorem 2. *In the BGW-model, for every $\epsilon > 0$, there is an efficient MPC protocol for n parties secure against a submaximal number of active corruptions $t < (1-\epsilon)n/3$ that computes $\Omega(n \log n)$ evaluations of a single binary circuit in parallel with an amortized communication complexity (per instance) of $O(1)$ bits per gate.*

We note that, as opposed to Theorem 1, this theorem may plausibly and alternatively be argued without recourse to our novel paradigm: indeed, we could deploy the asymptotically good arithmetic secret sharing schemes from [CCCX09] (over a suitably large constant extension of $\mathbb{F}_2$ so as to get the desired adversary rate), combined with an overhaul of the complete (say) Beerliova-Hirt protocols so as to make them work over these schemes instead of Shamir's. In addition, this would use some of our present techniques to overcome the arising issue that the protocol tricks from Beerliova-Hirt involving hyper-invertible matrices (or an alternative that we discuss later on) are defined over an extension of the field of definition of the arithmetic secret sharing. Our present approach, however, in fact uses Beerliova-Hirt essentially as a *black-box*. Moreover, our approach covers *both* theorems with the same method.

As noted in [IKOS09], a complexity of $O(1)$ bits per gate can also be obtained in the non-amortized setting as long as the number of parties which provide inputs is constant, by combining the protocol from [DI06] with the aforementioned arithmetic secret sharing schemes from [CCCX09]. This would be secure against an active adversary corrupting $t = \Omega(n)$ parties, where the constant is in principle small, but this can be brought up to $t < (1-\epsilon)n/3$, for any ϵ, by using Bracha's committees technique [Bra85] as described in [DIK10]. If we remove the assumption on the number of input-providing parties, then the best result in the non-amortized setting for suboptimal adversaries is given by [DIK10], where the communication complexity per gate is $O(\text{polylog}(n))$ bits. It is an interesting question to determine if the communication complexities obtained in Theorems 1 and 2 are optimal in this model[1].

We now give a brief preview of our paradigm and its technical challenges. First we introduce the notion of reverse multiplication friendly embeddings, which provide a way to embed the ring $\mathbb{F}_2^k$ into a field $\mathbb{F}_{2^m}$ so that coordinatewise products "map" to multiplications in the extension field in a certain manner that we will explain. Furthermore we will need to construct RMFEs with $m = O(k)$, so that the degree of the extension field does not explode.

[1] Some results on lower bounds for communication complexity of gate-by-gate protocols for arithmetic circuits like ours were obtained by [DNPR16] but they do not seem to be enough to claim such optimality.

Second, using such a map as a stepping stone, we construct a compiler that transforms a secure secret-sharing based computation protocol that evaluates an arithmetic circuit over a $\mathbb{F}_{2^m}$ (for example the one by [BH08]) into a secure protocol for the same number of parties and adversary that allows for parallel evaluation of $k = \Theta(m)$ of a related boolean circuit. Several obstacles appear when constructing this compiler, as a consequence of moving back and forth between the algebraic structures and we need to solve these issues with the introduction of several subprotocols. At the same time we need to ensure that these subprotocols do not require too much communication, so that this does not offset the gains from our embedding strategy.

It turns out that these subprotocols rely on a crucial step: we need to find a way to construct sharings of random elements in prescribed $\mathbb{F}_2$-subspaces of $(\mathbb{F}_{2^m})^v$. Our third contribution is a communication-efficient protocol to accomplish that. This in turn consists of the following ideas: first, we introduce a definition of generalized linear secret sharing scheme (GLSSS), where the secret and shares belong to vector spaces over the same field; then, we cast the strategy of random generation of sharings based on hyper-invertible matrices, introduced in [BH08] (and used in [DIK10]), in this language of GLSSS. This is still not good enough for our purposes, since our GLSSS is only linear over $\mathbb{F}_2$, while the hyper-invertible matrix is defined over $\mathbb{F}_{2^m}$. So the last idea we need consists in tensoring-up our scheme, that suitably transforms our $\mathbb{F}_2$-GLSSS into a $\mathbb{F}_{2^m}$-GLSSS. Along the way, we suggest alternatives for hyper-invertible matrices with the same functionality but which can be defined over a large enough constant size field, see Remarks 3 and 4. Although there is no overall advantage to our work (quantitatively), it does mean that, in the subprotocols where it is used, "amortization kicks in faster". Moreover, we believe it is of independent interest.

1.1 Main Ideas

We describe the general idea and the technical challenges we encounter more in detail.

Reverse multiplication friendly embeddings. As mentioned above, we want to embed several instances of the computation of a binary circuit into a single computation of an arithmetic circuit over an extension field. Ideally, we would wish that the ring $\mathbb{F}_2^k$, where the sum and product are defined coordinatewise, was isomorphic (as an $\mathbb{F}_2$-algebra) to the field $\mathbb{F}_{2^k}$, with the usual finite field sum and product; or said in a different way, that there would be a map $\eta : \mathbb{F}_2^k \to \mathbb{F}_{2^k}$ satisfying both $\eta(\mathbf{x} + \mathbf{y}) = \eta(\mathbf{x}) + \eta(\mathbf{y})$ and $\eta(\mathbf{x} * \mathbf{y}) = \eta(\mathbf{x}) \cdot \eta(\mathbf{y})$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^k$ (where $*$ denotes the coordinatewise product and $\cdot$ denotes the field product). If such a η existed, then embedding k evaluations of a boolean circuit into an evaluation of a circuit over $\mathbb{F}_{2^k}$ would be trivial: just define the arithmetic circuit C' to be the same as the boolean circuit, but substituting the sum and multiplication gates in $\mathbb{F}_2$ by gates performing the same operations in $\mathbb{F}_{2^k}$; then apply η to the vectors of boolean inputs, evaluate C' and map the result back to

$\mathbb{F}_2$ with η^{-1}. Furthermore, computing C securely would not be a problem, since the parties holding the inputs would secret-share them after applying η, and the result of the secure computation of C' would be opened prior to applying η^{-1}.

Unfortunately such an η does not exist: while $\mathbb{F}_2^k$ and $\mathbb{F}_{2^k}$ are isomorphic as $\mathbb{F}_2$-vector spaces, and hence the additive homomorphic condition can be satisfied, the multiplicative structures of $\mathbb{F}_2^k$ and $\mathbb{F}_{2^k}$ are however different for every $k \geq 2$ (e.g. the former structure has zero divisors, while the latter does not).

We then need to find some alternative weaker notion that allows us to travel back and forth between these two algebraic structures in a manner that it is still amenable to the secure computation protocols we want to adapt. In order to do this we introduce the notion of reverse multiplicative friendly embedding[2].

Definition 1. *Let q be a power of a prime and $\mathbb{F}_q$ a field of q elements, let $k, n \geq 1$ be integers. A pair (ϕ, ψ) is called an $(k, m)_q$-reverse multiplication friendly embedding (RMFE for short) if $\phi : \mathbb{F}_q^k \to \mathbb{F}_{q^m}$ and $\psi : \mathbb{F}_{q^m} \to \mathbb{F}_q^k$ are two $\mathbb{F}_q$-linear maps satisfying*

$$\mathbf{x} * \mathbf{y} = \psi(\phi(\mathbf{x}) \cdot \phi(\mathbf{y}))$$

for all $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^k$.

While this notion has not been explicitely defined[3] in the literature to the best of our knowledge, a construction for RMFEs was introduced in another work on secure multiparty computation, more precisely on the problem of correlation extraction [BMN17], where it is used to embed a number of instances of oblivious linear evaluation over a small field into one instance of OLE over the extension field. They obtain, for every prime power q and every integer $\ell \geq 1$, a $(2^\ell, 3^\ell)_q$. This implies that we can take $m = O(k^{\log 3/\log 2}) = O(k^{1.58\cdots})$. This construction is unfortunately not enough for our purposes, so in this paper, we show the existence of RMFEs with constant rate.

Theorem 3. *For every finite prime power q, there exists a family of $(k, m)_q$-RMFE where $m = \Theta(k)$.*

We show this result using techniques from algebraic geometry. We emphasize that this is the only point where algebraic geometry is used in our protocol.

As an aside, we also show, by some elementary results on polynomial interpolation, quite practical RMFE's for moderate values of m and with a reasonable rate m/k, indicating our main results may also have some practical value.

[2] The term "reverse" refers to the fact that multiplicative friendly embeddings where defined in [CCCX09]. The notions are similar but with the roles of the ring $\mathbb{F}_q^k$ and the field $\mathbb{F}_{q^m}$ swapped. Multiplicative friendly embeddings have been studied more extensively than their reverse counterpart, as they are a special case of bilinear multiplication algorithms [CC88]. They are also special cases of arithmetic codices [CCX12] (see also [CDN15]).

[3] Our original motivation for considering this notion (unpublished work, 2014) was to improve our result on arithmetic secret sharing [CCX11] from CRYPTO 2011.

The compiler. Given a $(k, m)_2$-RMFE (ϕ, ψ), we construct an information-theoretically secure protocol compiler that transforms a secure secret-sharing based computation protocol that evaluates an arithmetic circuit over a large enough finite field $\mathbb{F}_{2^m}$ into a secure protocol for the same number of parties and adversary that allows for the simultaneous evaluation of k instances of a related boolean circuit. The compiler introduces an overhead in the communication complexity of $O(nk)$ bits per multiplication gate of the circuit (hence $O(n)$ bits per multiplication). Our compiler requires that the MPC protocol for the arithmetic circuit over the extension field satisfies a number of properties, which are quite common and are fulfilled by most Shamir-secret-sharing based protocols in this model.

The first step of the compiler is to encode the input vectors as elements in $\mathbb{F}_{2^m}$ with the map ϕ. Now one could think that we proceed by evaluating the arithmetic circuit C' over $\mathbb{F}_{2^m}$ on the encoded inputs, and then decode the result with ψ. Unfortunately this idea does not work, for several reasons; even setting aside security considerations, note that this does not ensure correct computation: it is not even true that $\psi(\phi(\mathbf{x})) = \mathbf{x}$ for all $\mathbf{x}$, hence it does not compute the identity circuit correctly. Moreover $\mathbf{x}_1 * \mathbf{x}_2 * \ldots * \mathbf{x}_\ell = \psi(\phi(\mathbf{x}_1) \cdot \phi(\mathbf{x}_2) \cdot \ldots \cdot \phi(\mathbf{x}_\ell))$ does not necessarily hold when $\ell > 2$ either.

The way to correctly compute the result is instead as follows: encode the input vectors with ϕ and evaluate the arithmetic circuit C' over $\mathbb{F}_{2^m}$ on the encoded inputs, but with the additional step that, every time a multiplication gate its processed, we apply the composition $\phi \circ \psi$ to the output of the gate. We also need to slightly adjust the gates corresponding to a NOT gate in C: in C' these gates add the vector $\phi(1, 1, \ldots, 1)$; moreover if we have random gates in C (gates that produce a random bit), we will need to create random elements $\phi(r) \in \mathbb{F}_{2^m}$; we explain later how we can do this. By doing these transformations, we have the following invariant: at each wire of the $\mathbb{F}_{2^m}$-circuit C', the corresponding value is $\phi(\mathbf{w})$, where $\mathbf{w} = (w_1, \ldots, w_k)$ is the vector containing, for $i = 1, \ldots, k$, the bit w_i that would sit in the corresponding wire of the boolean circuit C on its i-th evaluation. Indeed note that

$$(\phi \circ \psi)(\phi(\mathbf{w}) \cdot \phi(\mathbf{w}')) = \phi(\psi(\phi(\mathbf{w}) \cdot \phi(\mathbf{w}'))) = \phi(\mathbf{w} * \mathbf{w}')$$

so multiplying two encoded vectors and then applying $\phi \circ \psi$ yields an encoding of the coordinatewise product. The rest of the gates obviously preserve this invariant.

At the last step, we decode the output by applying the inverse ϕ^{-1} of ϕ (which we insist, does not coincide with ψ). It is easy to derive from the definition of RMFE that ϕ is injective and hence ϕ^{-1} indeed exists.

Additional auxiliary protocols. However, several roadblocks are introduced when we want to transform a secret-sharing based secure computation protocol π' for C' into a secure computation protocol π that computes (k instances of) C: first, we want each of the input-holding parties to secret share a value $\phi(\mathbf{x}_i)$ with the secret sharing scheme used in π', but given that as we will see the image of

ϕ is not the full $\mathbb{F}_{2^m}$, the question is: how do we ensure that a party has not shared a value outside Im ϕ instead? Note such problem will not be detected by π', as this protocol will "accept" any sharing of a value in $\mathbb{F}_{2^m}$ as long as the sharing itself is correct, so we will need to add some type of zero-knowledge proof that ensures that the shared value is in Im ϕ. The second problem is that we need some protocol that transforms a sharing of an element a into a sharing of $(\phi \circ \psi)(a)$. The composition of ϕ and ψ is a $\mathbb{F}_2$-linear map, but this does not mean that it is $\mathbb{F}_{2^m}$-linear, and therefore it is not necessarily true that the parties can locally compute sharings for the output of this function by using the $\mathbb{F}_{2^m}$-linearity of the scheme[4].

We will show that we can reduce these two issues to the following problem: construct a secure multiparty protocol that outputs a sharing of a random element in a prescribed $\mathbb{F}_2$-subspace of $(\mathbb{F}_{2^m})^v$. That is, given an $\mathbb{F}_2$-vector space $V \subseteq \mathbb{F}_{2^m}^v$ the protocol should output $([r_1], \ldots, [r_v])$ where $(r_1, \ldots, r_v)$ is uniformly random in V, and $[x]$ denotes a sharing of x with the secret sharing scheme used in the protocol for C'.

GLSSS, tensoring-up and hyper-invertible matrices. In order to generate sharings of random elements of the given subspaces, we want to use a technique introduced in [BH08], based on so-called hyper-invertible matrices. In a nutshell this technique consist in the following. Suppose we want to create sharings of one of more random elements satisfying certain relation (we specify below what kind of relations are allowed). Then each party generates a sharing of random elements of their choice satisfying the said relation. Next the hyper-invertible matrix is applied (locally by each party) to the vector containing these n sharings. This creates n new sharings, which will obbey the same relation, if the parties have been honest. Next, some of these are opened to different parties, who check that relation indeed holds. The properties of the matrix will guarantee that if all honest parties declare themselves happy with this process, then the unopened sharings are guaranteed to satisfy the same relation.

However, the type of relations that are preserved by the hyper-invertible matrices are K-linear relations, where K is a field over which the matrix is defined. Unfortunately, the construction of hyper-invertible matrices from [BH08] is based on interpolation techniques, and it requires a field K which contains at least $2n$ elements. This clearly does not fit well with the $\mathbb{F}_2$-linear relations we are dealing with. Applying an $\mathbb{F}_{2^m}$-hyper-invertible matrix to a vector of elements in V will not necessarily output a vector of elements in V.

In order to solve this, we introduce several ideas. First, we formalize the idea of "sharing secrets which are bound by K-linear relations" by the notion of K-generalized linear secret sharing scheme (GLSSS), where the space of secrets

[4] To explain this further: we can think of the elements in $\mathbb{F}_{2^m}$ as polynomials in $\mathbb{F}_2[X]$ modulo a degree-m irreducible polynomial. Now consider for example the map that sends $u = a_0 + a_1 X + \cdots + a_{m-1} X^{m-1}$ to $F(u) := a_0$. This is a $\mathbb{F}_2$-linear map (it satisfies $F(u+v) = F(u) + F(v)$), but not $\mathbb{F}_{2^m}$-linear (otherwise there would exist $\lambda \in \mathbb{F}_{2^m}$ such that $F(u) = \lambda \cdot u$, and it is easy to see that this can not happen).

and the spaces of shares are (possibly different) K-vector spaces and the secret is determined by qualified sets of shares by means of a K-linear map. Our definition has the additional advantage that we do not need to worry about how encoding of the secret is done. We also show that the hyper-invertible-matrix technique fits naturally with the notion of GLSSS, since by means of this notion we can state one lemma that captures different instances of this technique in the literature [BH08,DIK10]. However, this is still not enough for our purposes, because this lemma only works if the hyper-invertible matrix is defined over the field K.

Then we introduce the concept of interleaved secret sharing scheme. Given a GLSSS Σ, the m-fold interleaved GLSSS $\Sigma^{\times m}$ is simply the n-player scheme naturally corresponding to m independent Σ-sharings of m secrets. The reason this is useful for our problem comes from the following observation, based on arguments from multilinear algebra (which we call tensoring-up lemma): if we start by a $\mathbb{F}_2$-GLSSS Σ with space of secrets V, there is a natural way in which we can see the interleaved GLSSS $\Sigma^{\times m}$ as a $\mathbb{F}_{2^m}$-GLSSS. Moreover, even though the space of secrets of the new scheme will be V^m, we can crucially access the individual Σ-sharings of each secret in V, since these are just the components of the sharing from $\Sigma^{\times m}$. This means that the hyper-invertible matrix methodology can be applied to $\Sigma^{\times m}$, where each party will bundle together m sharings of random elements in V as a sharing of a random element in V^m, apply the matrix to the resulting sharings using the $\mathbb{F}_2^m$-linear structure given by the tensoring up lemma, and "unzip" the result again into sharings of elements in V.

Putting things together. We will show that if we are using Shamir's scheme (or any secret sharing scheme where the size of the shares is the same as that of the secret) then our subprotocols require the communication of $O(n)$ field elements per gate. Because of our results on reverse multiplication friendly embeddings, this field will have 2^m elements where $m = \Theta(k)$.

On the other hand, to compute securely the arithmetic circuit over the extension field, we can use the protocol in [BH08], which also has communication complexity of $O(n)$ field elements per gate. Altogether we communicate $O(nk)$ bits per gate of the circuit to compute securely k evaluations of the circuit, an amortized cost of $O(n)$ bits per gate. In order for this amortization to work, we need that 2^m is at least n, hence we need to compute at least $k = \Omega(\log n)$ evaluations.

Remark 1 (On the Passive Case). One may possibly be inclined to believe that the case of *passive security* admits a much simpler solution that only involves the RFME's on top of standard protocols. Indeed, after the secure computation of the product of two ϕ-encodings, the secure computation of its $(\phi \circ \psi)$-image is *linear*. However, this is an $\mathbb{F}_2$-linear map defined on the extension field, *not* a linear combination of elements in this extension field. Therefore, the usual "secure computation of linear maps is for free" rule does not hold here. In particular, we still need the same auxiliary protocols with good amortized complexity (including the tensoring-up) as in the active case. That said, the hyper-invertible matrices can

be replaced by standard privacy amplification based on error correcting codes over large enough extension fields so as to be able to handle $t < n/2$ with t maximal or arbitrarily close.

2 Abstract GLSSS and Hyper-Invertible Matrices

In our protocol application, we will have a linear secret sharing scheme Σ defined over some "small" finite field K that we wish to deploy in secure computations involving very useful randomization protocols for secret-sharings (i.e., based on hyper-invertible matrices) that, unfortunately, require their field of definition to be a *larger* extension field L. Below we explain how to treat a number of Σ-sharings as a single sharing according to a scheme Σ' that is defined over the extension field L and that has the same privacy and reconstruction properties as Σ. The rate does not change either. Furthermore, "constituent" Σ-sharings remain readily "accessible" from Σ'-sharings for our use in our protocol. Finally, L-linear operations on Σ'-sharings are easily emulated in terms of K-linear operations on constituent Σ-sharings. The way to achieve this is by exploiting basic properties of the tensor product from abstract multilinear algebra.

Another technical aspect of our protocol application is that our Σ is not explicitly constructed as a K-linear scheme in the most standard way (e.g., from a given K-linear error correcting code $C \subset K^n$ with convenient properties) but rather implicitly. Namely, Σ will turn out to be a *K-linear "subscheme"* of a standard L-linear scheme. Concretely, Σ will correspond to several, independent instances of Shamir's scheme over L, where the secrets satisfy K-linear relations. The resulting scheme is, by all means, K-linear. But instead of complicating matters by "forcing" this into a standard formulation where the secret (and the shares) are typically encoded "systematically," we will capture it formally by giving an equivalent but "coordinate-free" version of the usual definition of (general) linear secret sharing.

2.1 An Abstract Definition of GLSSS

For nonempty sets U and $\mathcal{I}$, we let $U^{\mathcal{I}}$ denote the *indexed Cartesian product* $\Pi_{i\in\mathcal{I}}U$. For a nonempty subset $A \subset \mathcal{I}$, the natural *projection* π_A maps a tuple $u = (u_i)_{i\in\mathcal{I}} \in U^{\mathcal{I}}$ to the tuple $(u_i)_{i\in A} \in U^A$. Let K be a field.

Definition 2. *(Abstract K-GLSSS) A* general K-linear secret sharing scheme Σ *consists of the following data.*

- *A player set* $\mathcal{I} = \{1, \ldots, n\}$.
- *A finite-dimensional K-vectorspace Z, the* secret-space*, and a finite-dimensional K-vectorspace U, the* share-space.
- *A K-linear subspace $C \subset U^{\mathcal{I}}$, where the latter is considered a K-vector space in the usual way (i.e., direct sum).*
- *A surjective K-linear map $\Phi : C \longrightarrow Z$, its* defining map.

Definition 3. *Suppose $A \subset \mathcal{I}$ is nonempty. Then A is a* privacy set *if the K-linear map*

$$(\Phi, \pi_A) : C \longrightarrow Z \times \pi_A(C), \quad x \mapsto (\Phi(x), \pi_A(x))$$

is surjective. Finally, A is a reconstruction set *if, for all $x \in C$, it holds that*

$$\pi_A(x) = 0 \Rightarrow \Phi(x) = 0.$$

Remark 2. The following observations follow directly from the definition.

- (Privacy) Suppose K is finite. If we fix an arbitrary secret $z \in Z$ and select $x \in C$ uniformly random such that $\Phi(x) = z$, then for each privacy set A, it holds that the distribution of the joint shares $\pi_A(x) \in U^A$ for A does not depend on the secret z.
  We denote such a random sharing of a secret z as $[z]$, as usual.
- (Reconstruction) For each reconstruction set A, there are K-linear *reconstruction maps* $\{\rho_i : U \to Z\}_{i \in A}$, depending on A, such that for all $x \in C$, it holds that
  $$\sum_{i \in A} \rho_i(x_i) = \Phi(x).$$
  By definition, $\mathcal{I}$ is a reconstruction set.

2.2 Randomization Based on Hyper-Invertible Matrices

Hyper-invertible matrices were introduced in [BH08]. Hyper-invertible matrices provide a way for several parties to jointly generate sharings of uniformly random secrets satisfying certain relations. In [BH08], this relation consists in the fact that the uniformly random element is shared with two different sharings (Shamir sharings of different thresholds). However, different uses have been found in other protocols: in [DIK10], the parties generate sharings of uniformly random elements together with a permutation of their coordinates.

In this section, we show that those two applications of hyper-invertible matrices can both be captured under a common framework through the notion of generalized linear secret sharing schemes. Moreover, this framework will also encompass other two applications of this strategy in our protocol.

Definition 4 ([BH08]). *A matrix $M \in K^{\ell \times \ell'}$ is hyper-invertible over K if every s-by-s submatrix of M is invertible in K, for every $1 \leq s \leq \min\{\ell, \ell'\}$.*

As in [BH08], we are only interested in this work in square hyper-invertible matrices, even though the results can be generalized easily to non-square ones. If K has at least 2ℓ elements, there is the following construction of an ℓ by ℓ hyper-invertible matrix.

Lemma 1 ([BH08]). *Let K be a finite field with $|K| \geq 2\ell$. Fix $\alpha_1, \ldots, \alpha_\ell, \beta_1, \ldots, \beta_\ell$ distinct elements in K. Let $M = (m_{i,j})$ where $m_{i,j} = \prod_{k \neq j} \frac{\beta_i - \alpha_k}{\alpha_j - \alpha_k}$. Then M is a ℓ by ℓ hyper-invertible matrix over K.*

The interest of square hyper-invertible matrices for secure multiparty computation protocols arises from the property that any combination of ℓ inputs/outputs of the K-linear map induced by M are uniquely determined by, and can be written as a linear function of, the other ℓ inputs/outputs. More formally we have the following.

Lemma 2 ([BH08]). *Let $M \in K^{\ell\times\ell}$ be a square hyper-invertible matrix over K. Consider two subsets $A, B \subseteq \{1,\ldots,\ell\}$ such that $|A| + |B| = \ell$. Then there is a linear map $f_{A,B} : K^\ell \to K^\ell$ such that for every $\mathbf{x} \in K^\ell$, we have $f_{A,B}(\{x_i\}_{i\in A}, \{y_i\}_{i\in B}) = (\{x_i\}_{i\notin A}, \{y_i\}_{i\notin B})$, where $\mathbf{y} = M\mathbf{x}$.*

An important observation is that, given an K-vector space Z, we can define the action of M on vectors from Z^ℓ. Moreover, it is trivial to see that the property above still holds.

Proposition 1. *Let $M \in K^{\ell\times\ell}$ be a square hyper-invertible matrix over K and let Z be a K-linear vector space. Consider two subsets $A, B \subseteq \{1,\ldots,\ell\}$ such that $|A|+|B| = \ell$. Then there is a linear map $f_{A,B} : Z^\ell \to Z^\ell$ such that for every $\mathbf{x} \in Z^\ell$, we have $f_{A,B}(\{x_i\}_{i\in A}, \{y_i\}_{i\in B}) = (\{x_i\}_{i\notin A}, \{y_i\}_{i\notin B})$, where $\mathbf{y} = M\mathbf{x}$.*

Consider now a K-GLSSS Σ with secret space Z, share space U and player set $\mathcal{I}$. Denote a sharing of an element $z \in Z$ by $[z]$. The goal of the following protocol is to generate random sharings of a set of uniformly random elements from Z.

Protocol `RandEl`

Protocol for a set of parties $\mathcal{I} = \{1,\ldots,n'\}$. Let M be a $n' \times n'$-hyper-invertible matrix over K. Let T be an integer with $1 \le T \le n'$.

Output: Sharings $[r^1],\ldots,[r^T]$ of uniformly random elements in Z.

- For $i \in \mathcal{I}$, player i selects a uniformly random element $s^i \in Z$ and shares it among $\mathcal{I}$ with Σ.
- The players locally compute $([r^1],\ldots,[r^{|\mathcal{I}|}])^T = M\cdot([s^1],\ldots,[s^{|\mathcal{I}|}])^T$ (i.e., each party applies M to the vector of shares and interprets the resulting vector as containing shares to unknown elements $s^1,\ldots,s^{|\mathcal{I}|}$).
- For $i = T+1,\ldots,|\mathcal{I}|$ open r^i to party i. Party i checks that this is indeed a correct sharing to an element from Z and otherwise declares itself unhappy.
- Output the remaining unopened sharings $[r^1],\ldots,[r^T]$.

Proposition 2. *Suppose the active adversary has corrupted at most $t' \le (n'-1)/3$ parties from $\mathcal{I}$. Furthermore suppose Σ has t-privacy with $t \ge t'$, u-reconstruction with $u \le n' - t'$, and assume that $T \le n' - 2t'$.*

In these conditions if all honest players are happy after the execution of `RandEl`, then $[r^1],\ldots,[r^T]$ are correct sharings of uniformly random elements

$r^1, \ldots, r^T \in Z$ and the adversary has no information about these values, other than the fact that they belong to Z.

Proof. The proof essentially follows the steps of [BH08, Lemma 5].

We first consider robustness. First of all, by u-reconstruction, the shares of the (at least) $n' - t'$ honest parties uniquely determine the secrets, so cheating by the adversary by changing the shares corresponding to the corrupted parties will either be detected or not change the computation of the opened r^i. Assume that all honest players remain happy. Then all sharings opened to honest parties are valid sharings of elements r^i belong to Z. These are at least $n' - T - t'$. On the other hand the $n' - t'$ input sharings $[s^i]$ inputted by the honest parties are correct sharings of elements in Z. Note these are at least n' input/output values. By the properties of the hyper-invertible matrix the rest of inputs/outputs are a K-linear function of these values; a bit more precisely, for every honest party, her shares of the remaining inputs/outputs are a K-linear function of her shares of the honest inputs and the outputs opened by honest parties, and since the shares of honest parties fix the secrets (and Z is a K-linear vector space), the unopened secrets must be in Z.

We now consider privacy. First, by t-privacy, the shares of the adversary provide no information about the inputs provided by and the outputs opened to honest parties. The adversary does know the inputs s^i provided by corrupt parties and the outputs r^i opened to corrupt parties. But these are at most $2t'$ values. By Proposition 1, it is easy to see that these are completely independent of any set of $n' - 2t'$ other inputs/outputs, in particular the T outputted values.

This protocol generalizes the "double-sharing generation" strategy from [BH08] as well as the "generation of sharings of a random vector and a permutation of its coordinates" strategy from [DIK10]. In the first case, we can define a GLSSS that has K as space of secrets, K^2 as space of shares and where sharing in that scheme consists on independently sharing the secret with two standard Shamir secret sharing schemes of degrees d and d'. As long as $t' \leq \min d, d' < n - t'$, the conditions of the proposition are satisfied. In the second case, the secret space would be the ℓ-dimensional K-vector space $\{(\mathbf{x}, \pi(\mathbf{x})) : \mathbf{x} \in K^\ell\}$ where π is some known permutation of the coordinates of $\mathbf{x}$ and sharing means to share each coordinate individually with a K-linear scheme, and the proposition holds as long as this secret sharing scheme satisfies the privacy and reconstruction properties there.

2.3 Extending the Field of Definition of a GLSSS

Later on, we will encounter situations where our secret space is a $\mathbb{F}_2$-linear space, but not a $\mathbb{F}_{2^k}$-linear space, and hence the corresponding GLSSS is only linear over $\mathbb{F}_2$. This does not fit well with the fact that the hyper-invertible matrix will be defined over $\mathbb{F}_{2^m}$. Therefore we detail an strategy to extend the field of definition of a GLSSS.

To achieve the desired extension of the field of definition of a GLSSS, it is convenient to define *interleaved* GLSSS. Informally, the m-fold interleaved

GLSSS $\Sigma^{\times m}$ is the n-player scheme naturally corresponding to m Σ-sharings. In other words, with the notation of Sect. 2.1, a sharing in this scheme can be seen as an $m \times n$ matrix whose m rows each represent an element of C and whose n columns each represent an element of the K-vector space U^m, the share-space. We denote the K-linear subspace of $\times_{i \in \mathcal{I}} U^m$ collecting these matrices as $C^{\times m}$. The defining K-linear map $\Phi^{\times m}$ is just the "row-wise" application of Φ and the secret-space is the K-vector space Z^m. Note that the privacy sets as well as the reconstruction sets coincide with those of Σ.

A Tensoring-Up Lemma. Let L be an extension field of K of degree m. We will explain later on how $\Sigma^{\times m}$ is in fact an L-linear GLSSS in a natural and convenient way, compatible with its K-linearity as already defined. To this end we need a brief intermezzo derived from basic multilinear algebra, specifically a special case of base change in tensor products.[5] We will give an explicit lemma that defers the use of tensor products and their relevant properties to the proof.

Definition 5. *For our purposes, a K-algebra is a ring having the field K as a subring. Suppose R, S are K-algebras. A K-algebra morphism $R \longrightarrow S$ is a ring morphism that fixes K, i.e., it is, in particular, a K-vector space morphism.*

Lemma 3. *Let L be an extension field of degree r over K and let V be a K-vector space. Then the following hold:*

1. *Let $K^{m,m}$ denote the matrix algebra over K consisting of all m x m matrices with entries in K, with the usual addition and (K-scalar-) multiplication. Then there is a (non-unique) injective K-algebra morphism*

$$\Phi : L \longrightarrow K^{m,m}; \quad \lambda \mapsto \Phi^{(\lambda)}.$$

   In particular, the image of L is a field isomorphic to it.
2. *Each such Φ induces an L-vector space structure on V^m by defining L-scalar multiplication, for each $\lambda \in L$, as*

$$\lambda\cdot : V^m \longrightarrow V^m; \quad w \mapsto \Phi^{(\lambda)}(w),$$

   where the action of $K^{m,m}$ on V^m is the natural one, i.e., multiplication of an m x m matrix with an m-(column)vector.

   If $\lambda \in K$, it restricts to

$$\lambda\cdot : V^m \longrightarrow V^m; \quad w \mapsto \lambda \cdot w,$$

since Φ fixes K. Hence, this structure is compatible with the standard K-vector space structure on V^m, i.e., as a direct sum over V,

[5] For a treatment of abstract tensor-products aimed at a cryptographic audience, we refer to Ch. 10.9 (pp. 229–235) in [CDN15].

Proof. As to the first claim for each $\lambda \in L$, the multiplication-by-λ map on L is a K-vector space endomorphism on L (i.e., a morphism from L to L). The map Φ that sends $\lambda \in L$ to this associated morphism is clearly a K-algebra morphism from L to the K-algebra End of K-vector space endomorphisms of L. Note that Φ is injective as its domain is a field; the only possibility for its kernel is the trivial ideal (0) of L. Since L is a vector space of dimension r over K, it is clear that, once a basis is fixed, End may be given as $K^{m,m}$.

As to the second claim, an R-module M consists of an abelian group M, a ring R (with 1), together with a ring morphism mapping R to the ring of group endomorphisms of M. It is called a vector space if R is a field. In the present case, M is the direct sum V^m and R is L. By construction, (L, V^m, Φ) satisfies this condition. Finally, note that Φ maps $\lambda \in K$ to $\lambda \cdot I$, where I is the identity matrix.

As the lemma reflects one of the basic merits of tensor products, we verify it below in such terms and in more generality. By the very definition of tensor product, if M is an R-module (M and R take the role of V and K, resp.), where R is a commutative ring with 1, and if S is an extension ring (S takes the role of L), then the tensor product $S \otimes_R M$ is an R-module. By base change, we may, in fact, naturally view $S \otimes_R M$ as an S-module, compatible with the R-module structure already mentioned. Namely, for each $s \in S$, for each $r \in R$ and for each $m \in M$, define $s \cdot (r \otimes m) = (sr \otimes m)$ and extend this linearly to all of $S \otimes_R M$. If, in addition, S is free of rank r over R, then, as an R-module, the tensor product $S \otimes_R M$ is isomorphic to M^m. Since S-multiplication by a constant as defined above is, in particular, an endomorphism of R-modules, it is clear that such a map can be represented by an element of $K^{m,m}$.

Linearity over the Extension Field of the Interleaved Scheme. With the tensoring-up lemma in hand, we now explain how the m-fold interleaved GLSSS $\Sigma^{\times m}$ is L-linear, compatible with the K-linearity already pointed out. It is convenient, once again, to think of the elements of $C^{\times m}$ as matrices where each row is an element of C and where each column i collects the corresponding m shares for player i.

Take an arbitrary such matrix representing an element in $C^{\times m}$ and take an arbitrary $\lambda \in L$. Write the matrix map $\Phi^{(\lambda)}$ as $(\Phi_1, \ldots, \Phi_m)$ such that the image of $w \in U^m$ equals $(\Phi_1(w), \ldots, \Phi_m(w))$.

Then we simply replace each "column of shares" $w = (w_1, \ldots, w_m) \in U^m$ by the column $\lambda \cdot w = (\Phi_1(w), ..., \Phi_m(w))$. Since, the Φ_i are K-linear, it is immediate that application of the K-linear map $\Phi^{\times m}$ commutes with λ-multiplication. Thus, $\Psi^{\times m}$ is L-linear, compatible with its earlier mention K-linearity. Note that, for given Σ, this extension depends implicitly on the choice of a K-basis of L.

In summary:

Proposition 3. *Let L be a degree-m extension field of K and let Σ be a K-GLSSS. Then the m-fold interleaved K-GLSSS $\Sigma^{\times m}$ is naturally viewed as an L-GLSSS, compatible with its K-linearity.*

Use in Protocols. We work with this proposition as follows. Suppose we have a sharing in $\Sigma^{\times m}$, i.e., m Σ-sharings

$$([z_1], \ldots, [z_m]), \text{ with } z_1, \ldots, z_m \in Z.$$

If $\lambda \in L$, then

$$\lambda \cdot ([z_1], \ldots, [z_m]) = (\Phi_1([z_1], \ldots, [z_m]), \ldots, \Phi_r([z_1], \ldots, [z_m])),$$

which is λ times the given $\Sigma^{\times m}$-sharing, which, is, again, a $\Sigma^{\times m}$-sharing.

2.4 Alternatives for Hyper-Invertible Matrices: Same Functionality, but Constant-Size Field

We make two remarks about alternative approaches.

Remark 3. From known constructions, hyper-invertible matrices and their utility in MPC protocols appear tightly connected with polynomial evaluation codes (MDS codes) and therefore may seem to require a field of definition that grows as a linear function of n. However, we note that we found that there is an alternative coding-theoretic construction with essentially the same utility as that of hyper-invertible matrices but that allows constant-size finite fields, where the size should be large enough so as to make adversary rate 1/3 possible. In a nutshell, the argument goes as follows: given a linear code C of length $2n'$, minimum distance d and minimum distance of its dual $d^{\perp}$ it is not difficult to see that every coordinate can be written as a linear function of any set of $2n' - d + 1$ coordinates, and that any set of $d^{\perp} - 1$ coordinates of a random codeword are uniformly distributed. Suppose in addition the code has dimension n', wlog assume its in systematic form and its generator matrix is $G = (I_{n'}|M)$. If we can take $d, d^{\perp} \geq (2/3 + \epsilon)n' \geq 2t' + \epsilon n'$, then Proposition 2 still holds for $T \leq \epsilon n'$. Taking random linear codes over $\mathbb{F}_{64}$ of rate 1/2 should suffice for this purpose according to the Gilbert-Varshamov bound (and the secret sharing scheme should then be tensored-up to this field). Although there is no overall advantage to our work (quantitatively), it does mean that, in the subprotocols where it is used, "amortization kicks in faster."

Remark 4. Instead of tensoring-up the secret sharing scheme, we may have taken hyper-invertible matrices (or the alternative above) and have re-worked them to be defined over the base field, using the same technique as in tensoring-up (i.e., viewing the extension field as a matrix algebra over the base field) and making substitutions accordingly. This leads to a "block-wise" version of hyper-invertibility which is sufficient for our purposes. However, we feel that the present approach we took is more natural and leads to cleaner protocols.

3 The Protocol

In this section we detail our protocol π, that securely evaluates k instances of a binary circuit C in parallel by using a secure computation protocol π' that computes securely essentially the same arithmetic circuit defined over a extension field $\mathbb{F}_{2^m}$.

3.1 Framework

We consider a network of n parties who communicate via pairwise secure channels. Up to t of these parties are corrupted by an active adversary, where will require that $t < n/3$.

Let C be a boolean circuit consisting of input gates; computation gates which will be (unbounded fan-in) addition (XOR) gates, fan-in 2 multiplication (AND) gates and NOT gates (which we can think of as addition with the constant 1); random gates that output a uniformly random bit and an output gate. We assume that there is a single output gate for simplicity of notation only, as the generalization of our results to the case where there are more output gates is straightforward. Let c_I, c_R, c_M be the number of input, random and multiplication gates respectively.

Given a $(k, m)_2$-RMFE (ϕ, ψ), we define the following arithmetic circuit C_ϕ over the extension field $\mathbb{F}_{2^m}$: We replace the XOR and AND gates in C by gates implementing addition and multiplication in $\mathbb{F}_{2^m}$ and we replace the NOT gates by addition with the field element $\phi((1, 1, \ldots, 1)) \in \mathbb{F}_{2^m}$ (which may not coincide with the element 1 in $\mathbb{F}_{2^m}$). For consistency we replace the boolean random gates by gates which create random elements in $\mathbb{F}_{2^m}$; this does not really have too much importance, since we will entirely replace the computation of this gate by a subprotocol.

Preprocessing and player elimination. Our protocol π will have a pre-processing phase, which is independent of the inputs, and a computation phase.

In addition, π will use the player elimination framework. Player elimination, introduced in [HMP00], is a technique by which the computation (or part of it) is first divided in segments and in each segment, if at least one party has deviated from the protocol, a set of two parties is identified out of which at least one is a corrupt. The protocol then proceeds by eliminating these two parties and recompute the segment. This protocol works exactly as described in [BH08], so we refer the reader to that work for its detailed description. At every step of the protocol, we will denote by n' the number of active (not eliminated) parties, and by t', the number of active corrupted parties. Note that the invariant $t < n' - 2t'$ always holds.

It is important to mention that, as it occurs in other protocols such as [BH08], we will use player elimination in the preprocessing phase only.

Conditions on π'. We now describe the conditions that π' needs to satisfy so that we can apply our compiler and construct π. First of all, π' will be a secret-sharing based protocol and we need to make some assumptions on the underlying secret sharing scheme.

Given a secret sharing scheme with player set $\mathcal{I}$, by puncturing the scheme at a subset $A \subseteq \mathcal{I}$ we mean that we consider the secret sharing scheme where we remove the set A of parties (so the new player set is $\mathcal{I} \setminus A$ and sharing happens in the same way as in the original scheme, except that the shares that would correspond to the subset A are erased).

Definition 6. *We say that a secret sharing scheme is t-robust if there exists a polynomial-time algorithm that, when given as input all shares in a sharing $[x]$, among which at most t are erroneous, outputs x.*

We say that a secret sharing scheme on n parties is elimination-compatible t-robust, if for every $0 \leq u \leq t$, and any set of $2u$ parties, puncturing the scheme at those $2u$ parties results in a scheme on the other $n' = n - 2u$ parties which is t'-robust, where $t' = t - u$.

Remark 5. Note that a degree-t Shamir's secret sharing scheme for n parties is elimination-compatible t-robust as long as $3t + 1 \leq n$. Indeed, after player elimination, the set of possible sharings forms a Reed-Solomon code of length n' and dimension $t + 1$, and therefore minimum distance $n' - t$. There exist well known efficient algorithms that can correct any $e < (n' - t)/2$ errors. But the number of errors that can be introduced by the adversary is at most t', and as we noted above $t < n' - 2t'$ which implies $t' < (n - t)/2$.

We assume that the secure multiparty computation protocol π' to compute C_ϕ has the following features:

Assumptions on π'

- π' may have a preprocessing phase, which is independent of the inputs, and a computation phase. We allow the protocol to use player elimination in the preprocessing phase.
- π' is secure against an active adversary corrupting t parties.
- π' is a secret-sharing based secure multiparty computation protocol π' which uses a $\mathbb{F}_{2^m}$-secret sharing scheme with t-privacy and which is elimination-compatible t-robust (the sharing of an element x is denoted $[x]$)
- In the computation phase every input and intermediate computed value remain secret shared among the parties with this secret sharing scheme; the protocol creates these sharings as follows: at every addition gate, parties locally compute a sharing of the output of the gate from the sharings of the inputs using the linearity of the scheme; the same holds for addition and multiplication by known constants; multiplication gates are processed by a subprotocol `Mult` that on input $[a], [b]$ produces $[ab]$.

3.2 Result

In the rest of the section we prove the following theorem.

Theorem 4. *Assume there exists a $(k, m)_2$-reverse multiplication friendly embedding (ϕ, ψ), where $2^m \geq 2n$ and let π' be a secure multiparty computation protocol for the arithmetic circuit C_ϕ over $\mathbb{F}_{2^m}$ satisfying the assumptions above. Then there exists a multiparty computation protocol secure against*

an active adversary who corrupts at most t parties and which allows to compute k instances of the circuit C with communication complexity $cc(\pi) = cc(\pi') + (c_I + c_M + c_R) \cdot O(n)$ elements of $\mathbb{F}_{2^m}$.

3.3 The General Structure of π

The general idea of the construction has been explained in the introduction: in the first step, for $i = 1, \ldots, n$, the i-th party, who has an input $\mathbf{x}_i = (x_i^{(1)}, ..., x_i^{(k)}) \in \mathbb{F}_2^k$, creates a sharing of $[\phi(\mathbf{x}_i)] \in \mathbb{F}_{2^m}$. A subprotocol `CorrInput` will ensure that this sharing is well constructed (in particular, it hides an element from Im ϕ). Then the parties execute π' on inputs $[\phi(\mathbf{x}_1)], \ldots, [\phi(\mathbf{x}_n)]$, but every time that there is a multiplication gate, the output of that gate, say $[a]$, will be re-encoded by applying a sub-protocol `ReEncode` that creates $[\phi(\psi(a))]$ from $[a]$. Therefore, at the end of the computation with π', the parties obtain $\phi(\mathbf{y})$, for $\mathbf{y} = (y^{(1)}, ..., y^{(k)})$. Here each $y^{(j)}$ is the output of the evaluation of C on $(x_1^{(j)}, ..., x_N^{(j)})$. Every party can now apply ϕ^{-1} to recover $\mathbf{y}$.

An additional detail is that for random gates we need to create sharings of uniformly random elements in Im ϕ. As we will see, this is exactly the main step in `CorrInput` too.

We explain the subprotocols in the following lines.

3.4 Auxiliary Protocols

We will now describe the subprotocols needed in π. We recall that since we use player elimination, at a given point of the protocol there will be n' active parties out of which t' are corrupted, where $n' = n - 2u$ and $t' = t - u$ for some $0 \leq u \leq t$, and that $t < n' - 2t'$ always holds. So we describe our protocols taking that into account (for the sake of notation the active parties are indexed by $1, \ldots, n'$). In particular, it will be understood that the secret sharing scheme at a given point of the protocol is the original secret sharing scheme punctured on $2u$ parties.

We start with the public reconstruction protocol `ReconsPubl`. One possibility could of course be simply to have every party send their share to each other, after which every party clearly can reconstruct, since the scheme is t'-robust. However, this incurs in a communication complexity of $\Theta(n^2)$ elements of the field. The following idea comes originally from [DN07] and allows to amortize the reconstruction, so that the communication complexity is still $\Theta(n^2)$ but $\Omega(n)$ sharings are simultaneously reconstructed.

Protocol `ReconsPubl` (from [DN07])

Input: $[a_1], [a_2], \ldots, [a_{n'-2t'}]$.
Output: All parties obtain $a_1, a_2, \ldots, a_{n'-2t'}$.
Fix $\beta_1, \ldots, \beta_{n'} \in \mathbb{F}_{2^m}$ pairwise distinct.

– Call $u_j := \sum_{i=1}^{n'-2t'} a_i \beta_j^i$. For all j, parties locally compute $[u_j] = \sum_{i=1}^{n'-2t'} [a_i] \beta_j^i$.

- For all i, all parties send their shares of u_i to P_i.
- For all i, P_i applies the robust reconstruction algorithm of the secret sharing scheme to obtain u_i.
- For all i, j, P_i sends u_i to P_j.
- For all j, P_j applies an standard error decoding algorithm for Reed-Solomon codes to recover $a_1, \ldots, a_{n'-2t'}$ from the values $\widetilde{u}_1, \ldots, \widetilde{u}_{n'}$ received in the previous step (using that $\widetilde{u}_i \neq u_i$ for at most t' values).

Remark 6. `ReconsPubl` allows to perfectly reconstruct $n' - 2t' = \Omega(n)$ sharings by communicating $2n'(n'-1) = O(n^2)$ elements of the field in total, an amortized cost of $O(n)$ elements of the field per reconstructed sharing.

As it has been mentioned before, both the subprotocols `CorrInput` and `ReEncode` need to use sharings of uniformly random elements in certain $\mathbb{F}_2$-subspaces. These will be generated in the preprocessing phase with the help of hyper-invertible matrices, by using the techniques introduced in Sect. 2.

We describe more explicitely how this works. Let $V \subseteq \mathbb{F}_{2^m}^v$ be a $\mathbb{F}_2$-subspace (in our protocols we will only encounter the cases $v = 1$, $v = 2$, but here we treat the problem more generally). The protocol `RandElSub`(V) generates sharings of uniformly random elements in V. Here, by a sharing of an element $u = (u_1, \ldots, u_v) \in V$ we refer to the generalized linear secret sharing scheme that consists in that each coordinate u_j is shared with the secret sharing scheme used in the protocol. This is an $\mathbb{F}_2$-generalized linear secret sharing scheme where the secret space is V and the share spaces are $\mathbb{F}_{2^m}^v$. We will call this secret sharing scheme Σ, but by abuse of notation we write $[u] = ([u_1], \ldots, [u_v])$.

We need a $n' \times n'$-hyper-invertible matrix M over some finite field. Since $|\mathbb{F}_{2^m}| \geq 2n'$ by assumption, we know how to construct such matrices over $\mathbb{F}_{2^m}$, by Lemma 1. However, the GLSSS described above is only linear over $\mathbb{F}_2$, so in order to apply the hyper-invertible matrix we need to tensor-up this GLSSS to a $\mathbb{F}_{2^m}$-linear one, using the techniques from Sect. 2. Recall that this will create the interleaved $\mathbb{F}_{2^m}$-GLSSS $\Sigma^{\times m}$ where the secrets are in V^m and each share is in $(\mathbb{F}_{2^m})^m$, to which we apply the hyper-invertible matrix technique. However, note that the interleaved scheme still allows to access easily the sharings of the individual elements in V. Namely the scheme has secrets $\mathbf{u} = (u_1, \cdots, u_m)$ where $u_j = (u_{j,1}, \ldots, u_{j,v}) \in V$ for $j = 1, \ldots, m$ and the sharing of $\mathbf{u}$ consists of independent sharings of all $u_{j,\ell} \in \mathbb{F}_{2^m}$ with the scheme used by π'. We abuse once more notation and denote $[\mathbf{u}] := ([u_1], \cdots, [u_m])$ where in turn $[u_i] = ([u_{i,1}], \ldots, [u_{i,v}])$.

The protocol is as follows.

Protocol `RandElSub`(V)

Parameter: Let T be an integer with $1 \leq T \leq n' - 2t'$.

Output: Sharings $[r^i_{j,\ell}]$ of elements $r^i_{j,\ell} \in \mathbb{F}_{2^m}$, $i = 1, \ldots, T$, $j = 1, \ldots, m$, $\ell = 1, \ldots, v$, where $r^i_j = (r^i_{j,1}, \ldots, r^i_{j,v})$ are uniformly random elements from V.

Let $M \in \mathbb{F}_{2^m}^{n' \times n'}$ be a hyper-invertible matrix.

- For $i = 1, \ldots, n'$, P_i selects m uniformly random elements $s^i_1, \cdots, s^i_m \in V$ and creates a sharing $[\mathbf{s}^i] := ([s^i_1], \cdots, [s^i_m])$ with the interleaved secret sharing scheme $\Sigma^{\times m}$, where in turn $[s^i_j] := ([s^i_{j,1}], [s^i_{j,2}], ..., [s^i_{j,v}])$.
- Players locally compute $([\mathbf{r}^1], \ldots, [\mathbf{r}^{n'}]) = M([\mathbf{s}^1], \ldots, [\mathbf{s}^{n'}])$. Note that the entries of M are in $\mathbb{F}_{2^m}$ and that M acts on $[\mathbf{s}^i]$ as explained in Section 2, using the fact that $\Sigma^{\times m}$ is a $\mathbb{F}_{2^m} - GLSSS$.
- For $i = T + 1, \ldots, n'$, every party P_j sends its share of $[\mathbf{r}^i]$ to P_i. Note that $[\mathbf{r}^i]$ can always be parsed as $([r^i_1], \cdots, [r^i_m])$, so what P_j sends is her shares of m values shared with Σ. P_i verifies that the values received indeed are valid sharings of values $(r^i_1, \cdots, r^i_m)$, and that $r^i_1, \ldots, r^i_m \in V$. If any check fails, P_i gets unhappy.
- The remaining T sharings $[\mathbf{r}^1], \ldots, [\mathbf{r}^T]$ are outputted. Note that $[\mathbf{r}^i] = ([r^i_1], [r^i_2], \ldots, [r^i_m])$ where $r^i_j = (r^i_{j,1}, r^i_{j,2}, \ldots, r^i_{j,v}) \in V$ and $r^i_j = ([r^i_{j,1}], [r^i_{j,2}], \ldots, [r^i_{j,v}])$

Proposition 4. *If all honest players are happy after the execution of* `RandElSub`*, then* $[\mathbf{r}^1], \ldots, [\mathbf{r}^T]$ *are* $\Sigma^{\times m}$*-sharings of uniformly random vectors* $\mathbf{r}^1, \ldots, \mathbf{r}^T \in V^m$*, i.e.,* `RandElSub` *produces* Σ*-sharings of* mT *uniformly random values* $r^i_j \in V$*,* $j = 1, \ldots, m, i = 1, \ldots, T$*, about which the adversary learns no information (other than the fact that they are elements from* V*). The total communication complexity of* `RandElSub` *is* $(2n' - T)(n' - 1)mv$ *field elements, which if* $T = n' - 2t' = \Theta(n)$ *yields an amortized cost of* $O(nv)$ *field elements per sharing of an element in* V*.*

The proof of this result consists in noticing that this protocol is `RandEl` from Sect. 2 applied to the $\mathbb{F}_{2^m}$-GLSSS $\Sigma^{\times m}$.

We will handle the case where parties declare themselves unhappy by means of the player elimination technique. For the moment we assume that enough sharings of random elements in the appropriate subspaces have been generated.

Next, we describe the protocol `CorrInput` which takes as input $[a]$ (where $a \in \mathbb{F}_{2^m}$) and whose goal is verifying that $a \in \mathrm{Im}\ \phi$. For this the parties take a sharing $[r]$ of a uniformly random element $r \in \mathrm{Im}\ \phi$, that have been generated by the protocol `RandElSub`(Im ϕ). Then they can use it to locally compute $[a + r]$,

open this sharing and verify that $a + r \in \text{Im } \phi$, and since Im ϕ is a $\mathbb{F}_2$-vector subspace, $r, a + r \in \text{Im } \phi$ imply that $a \in \text{Im}\,\phi$. Moreover, since r is uniformly random in Im ϕ, the opened value $a + r$ gives no additional information on a.

Protocol `CorrInput`

Input: $[a]$.
Output: Accept if $a \in \text{Im}\,\phi$. Reject otherwise.

- Take the next unused sharing $[r]$ produced by `RandElSub`(Im ϕ).
- Compute $[a + r] = [a] + [r]$ locally.
- Use `ReconsPubl` to open $[a + r]$. Let b be the opened value.
- Accept if $b \in \text{Im}\,\phi$. Reject otherwise.

It is quite straightforward that this protocol is secure. Note that all honest parties will receive the same output, because `ReconsPubl` will output the same value to all of them. Moreover, notice that if $[a]$ is a correct sharing, then `ReconsPubl` will succeed reconstructing $a + r$ even if malicious parties communicate false shares because `ReconsPubl` is robust.

Finally, we consider the protocol `ReEncode`, whose goal is to construct $[\phi(\psi(a))]$ from $[a]$, where $a \in \mathbb{F}_{2^m}$. We remark first that the composition $\phi \circ \psi : \mathbb{F}_{2^m} \to \mathbb{F}_{2^m}$ is an $\mathbb{F}_2$-linear map, but not an $\mathbb{F}_{2^m}$-linear map. Therefore we cannot use the $\mathbb{F}_{2^m}$-linearity of the secret sharing scheme to have parties locally compute $[\phi(\psi(a))]$ given $[a]$. Instead, we use a randomization technique, as in the case of `CorrInput`. Define the set

$$W = \{(x, \phi(\psi(x))) : x \in \mathbb{F}_{2^m}\} \subseteq (\mathbb{F}_{2^m})^2.$$

This is an $\mathbb{F}_2$-subspace of $(\mathbb{F}_{2^m})^2$. The parties will have called `RandElSub` on W in the preprocessing phase in order to create (at least) c_M sharings of random elements in W. They take a unused such sharing $[\mathbf{r}] = ([r], [\phi(\psi(r))])$. Then they can use it to locally compute $[a + r]$, open this value and then compute $[\phi(\psi(a))] = [\phi(\psi(a + r))] - [\phi(\psi(r))]$, where $[\phi(\psi(a + r))]$ is some default sharing of the public element $\phi(\psi(a + r))$, which can be computed from the opened information $a + r$. Note that this opened value $a + r$ gives no information about a, since r is uniform in $\mathbb{F}_{2^m}$.

Protocol `ReEncode`

Input: $[a]$.
Output: $[\phi(\psi(a))]$.
Let $W := \{(x, \phi(\psi(x))) : x \in \mathbb{F}_{2^m}\} \subseteq (\mathbb{F}_{2^m})^2$.

- Take the next unused sharing $[\mathbf{r}]$ produced by `RandElSub`(W). Parse $[\mathbf{r}]$ as $([r], [s])$, where $s = \phi(\psi(r))$.

– Compute $[a + r] = [a] + [r]$ locally.
– Use `ReconsPubl` to open $[a + r]$. Let m be the opened value.
– Compute $[w] = \phi(\psi(m)) - [s]$.
– Output $[w]$.

3.5 Final Protocol

We describe our final protocol. The preprocessing phase will generate sharings of at least $c_I + c_R$ uniformly random values in $\operatorname{Im}\phi$, and at least c_M uniformly random values in W. In order to incorporate player elimination, we split the computation of these values in $(c_I + c_M + c_R)/t$ segments. After the computation of each segment, if some party is unhappy, then all values generated in that segment are discarded and player elimination is used to identify a set of two parties containing one malicious party. These two parties are eliminated and the computation of the segment is restarted with the updated values for n' and t' and all parties resetting their status to happy.

Protocol π

Inputs: $\mathbf{x}_i = (x_i^{(1)}, ..., x_i^{(k)}) \in \mathbb{F}_2^k$, $i = 1, \ldots, N$, where each $\mathbf{x}_i$ is known to some party.

Output: All parties learn $\mathbf{y} = (y^{(1)}, \ldots, y^{(k)})$ where $y^{(j)}$ is the evaluation of circuit C on input $(x_1^{(j)}, ..., x_N^{(j)})$.

(Input-independent) preprocessing phase:

– Generation of random elements in $\mathbb{F}_2$-subspaces. The following computation is splitted in $(c_I + c_M + c_R)/t$ segments. After each segment, if some party is unhappy, discard that computation, execute player elimination and restart the segment with the new set of parties.
  - The parties run `RandElSub`($\operatorname{Im}\phi$) enough number of times to create sharings of at least $c_I + c_R$ random elements in $\operatorname{Im}\phi$.
  - The parties run `RandElSub`(W) enough number of times to create sharings of at least c_M random elements in $\operatorname{Im}\phi$.
– The parties execute the preprocessing phase of π', if there is any.

Computation phase:

– For $i = 1, \ldots, N$, the party holding input $\mathbf{x}_i$ computes $\phi(\mathbf{x}_i)$ execute the subprotocol from π' to create $[\phi(\mathbf{x}_i)]$. The parties execute `CorrInput`, using the next unused sharing produced by `RandElSub`($\operatorname{Im}\phi$).
– Parties execute the rest of the computation phase of π' on inputs $([\phi(\mathbf{x}_1)], \ldots, [\phi(\mathbf{x}_N)])$ with the following changes:

At every multiplication gate of C', after the parties execute `Mult` on inputs $([a],[b])$ and obtain $[ab]$, they apply subprotocol `ReEncode` to $[ab]$ and produce $[\phi(\psi(ab))]$. Each time `ReEncode` is called the next unused sharing produced by `RandElSub`(W) is used.
At every random gate of C' the computation of the gate by π' is ignored and instead the next unused sharing $[\phi(r)]$ produced by `RandElSub`$(\operatorname{Im}\phi)$ is used.

– Let z be the output of π' in the execution of the protocol. The output of π is $\mathbf{y} = \phi^{-1}(z)$.

We consider the communication complexity of π. It executes one instance of π', one instance of `CorrInput` per input gate and one instance of `ReEncode` per multiplication gate of the circuit. In turn, both `CorrInput` and `ReEncode` execute the public reconstruction protocol of the secret sharing scheme and both subprotocols require one fresh sharing of a random element produced by `RandElSub` (invoked on $V = \operatorname{Im}\phi$ in the case of `CorrInput` and on $V = W$ in the case of `ReEncode`). Note that we can use `RandElSub` to create these sharings of random elements in batches of size $n \log n$ with a communication complexity $O(n^2 \log n)$, which gives an amortized complexity of $O(n)$ field elements per output sharing.

Therefore the communication complexity of the protocol π is $cc(\pi) = cc(\pi') + (c_I + c_M + c_R) \cdot O(n)$ field elements.

4 Reverse Multiplicative Friendly Embeddings

In this section, we show, by algebraic geometric means, effective $(k,m)_q$-RMFE's with $m = O(k)$ for every finite field $\mathbb{F}_q$. The hidden constant is actually quite small.

But first show that if the size of the base field q is larger than $k-1$ we can construct a $(k, 2k-1)_q$-RMFE's based on some elementary results on polynomial interpolation. Chaining these together by concatenation, we then show quite practical RMFE's for moderate values of m and reasonable rate m/k. This indicates that our main results may also have some practical value.

Lemma 4. *For all $1 \le k \le q+1$, there exists a $(k, 2k-1)_q$-RMFE.*

Proof. Let $\mathbb{F}_q[X]_{\le m}$ denote the set of polynomials in $\mathbb{F}_q[X]$ of degree at most m and let ∞_{m+1} be a formal symbol such that $f(\infty_{m+1})$ is the coefficient of X^m in $f \in \mathbb{F}_q[X]_{\le m}$. Let $x_1, \ldots, x_k$ be pairwise distinct elements in $\mathbb{F}_q \cup \{\infty_k\}$ and let $\alpha \in \mathbb{F}_{q^{2k-1}}$ be such that $\mathbb{F}_{q^{2k-1}} = \mathbb{F}_q(\alpha)$.

By [CDN15, Theorems 11.13, 11.96] the maps

$$\mathcal{E}_1 : \mathbb{F}_q[X]_{\le k-1} \to \mathbb{F}_q^k; \quad f \mapsto (f(x_1), f(x_2), \ldots, f(x_k))$$

and

$$\mathcal{E}_2 : \mathbb{F}_q[X]_{\leq 2k-2} \to \mathbb{F}_{q^{2k-1}}; \quad f \mapsto f(\alpha)$$

are isomorphisms of $\mathbb{F}_q$-vector spaces.

Define also

$$\mathcal{E'}_1 : \mathbb{F}_q[X]_{\leq 2k-2} \to \mathbb{F}_q^k; \quad f \mapsto (f(x'_1), f(x'_2), \ldots, f(x'_k))$$

where $x'_i := x_i$ if $x_i \in \mathbb{F}_q$, and $x'_i := \infty_{2k-1}$ if $x_i = \infty_k$.

Now we define $\phi = \mathcal{E}_2 \circ \mathcal{E}_1^{-1}$ and $\psi = \mathcal{E'}_1 \circ \mathcal{E}_2^{-1}$ (where in the case of ϕ the composition makes sense because $\mathbb{F}_q[X]_{\leq k-1} \subseteq \mathbb{F}_q[X]_{\leq 2k-2}$). Then using that $fg(\alpha) = f(\alpha)g(\alpha)$ and $fg(x'_i) = f(x_i)g(x_i)$ for all $f, g \in \mathbb{F}_q[X]_{\leq k-1}$, it is immediate that (ϕ, ψ) is a $(k, 2k-1)_q$-RMFE.

Next, we show how to concatenate RMFEs over different finite fields.

Lemma 5. *Assume that (ϕ_1, ψ_1) is an $(k_1, m_1)_{q^{m_2}}$-RMFE and (ϕ_2, ψ_2) is an $(k_2, m_2)_q$-RMFE. Then*

$$\phi : \mathbb{F}_q^{k_1 k_2} \to \mathbb{F}_{q^{m_1 m_2}},$$

$$(\mathbf{x}_1, \ldots, \mathbf{x}_{k_1}) \mapsto (\phi_2(\mathbf{x}_1), \ldots, \phi_2(\mathbf{x}_{n_1})) \in \mathbb{F}_{q^{m_2}}^{k_1} \mapsto \phi_1(\phi_2(\mathbf{x}_1), \ldots, \phi_2(\mathbf{x}_{k_1}))$$

and

$$\psi : \mathbb{F}_{q^{m_1 m_2}} \to \mathbb{F}_q^{k_1 k_2},$$

$$\alpha \mapsto \psi_1(\alpha) = (\mathbf{u}_1, \ldots, \mathbf{u}_{k_1}) \in \mathbb{F}_{q^{m_2}}^{k_1} \mapsto (\psi_2(\mathbf{u}_1), \ldots, \psi_2(\mathbf{u}_{k_1}))$$

give an $(k_1 k_2, m_1 m_2)_q$-RMFE.

Proof. It is clear that both ϕ and ψ are $\mathbb{F}_q$-linear. For any $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^{k_1 k_2}$, we have

$$\begin{aligned}
\psi(\phi(\mathbf{x}) \cdot \phi(\mathbf{y})) &= \psi_2 \circ \psi_1(\phi_1(\phi_2(\mathbf{x}_1), \ldots, \phi_2(\mathbf{x}_{k_1})) \cdot \phi_1(\phi_2(\mathbf{y}_1), \ldots, \phi_2(\mathbf{y}_{k_1}))) \\
&= \psi_2((\phi_2(\mathbf{x}_1), \ldots, \phi_2(\mathbf{x}_{k_1})) * (\phi_2(\mathbf{y}_1), \ldots, \phi_2(\mathbf{y}_{k_1}))) \\
&= (\psi_2(\phi_2(\mathbf{x}_1) \cdot \phi_2(\mathbf{y}_1)), \ldots, \psi_2(\phi_2(\mathbf{x}_{k_1}) \cdot \phi_2(\mathbf{y}_{k_1}))) \\
&= (\mathbf{x}_1 * \mathbf{y}_1, \ldots, \mathbf{x}_{k_1} * \mathbf{y}_{k_1}) = \mathbf{x} * \mathbf{y}
\end{aligned}$$

This completes the proof.

Remark 7 ("On practical parameters"). As a consequence of applying the above two results we have the following embeddings of $\mathbb{F}_2^k$ into extensions of degree up to 325.

1. For all $r \leq 9$, there exists a $(2r, 6r-3)_2$-RMFE (obtained by concatenation of $(2,3)_2$ and $(r, 2r-1)_8$-RMFEs, both promised by Lemma 4).
2. For all $r \leq 33$, there exists a $(3r, 10r-5)_2$-RMFE (obtained by concatenation of $(3,5)_2$ and $(r, 2r-1)_{32}$-RMFEs, both promised by Lemma 4).

We now move to the asymptotic results, for which we need the methods from the theory of algebraic function fields. We will not give a detailed explanation of this area here, and refer the reader to the book by Stichtenoth [Sti09]. However, we sum up the facts that we need, ignoring some technical details.

A function field $F/\mathbb{F}_q$ is an algebraic extension of the rational function field $\mathbb{F}_q(x)$, that contains all fractions of polynomials in $\mathbb{F}_q[x]$. Associated to a function field, there is a non-negative integer $\mathfrak{g}$ called the genus, and an infinite set of "places" P, each having a degree $\deg P \in \mathbb{N}$. The number of places of a given degree is finite. The places of degree 1 are called rational places. Given a function $f \in F$ and a place P, two things can happen: either f has a pole in P, or f can be evaluated in P and the evaluation $f(P)$ can be seen as an element of the field $\mathbb{F}_{q^{\deg P}}$. If f and g do not have a pole in P then the evaluations satisfy the rules $\lambda(f(P)) = (\lambda f)(P)$ (for every $\lambda \in \mathbb{F}_q$), $f(P) + g(P) = (f+g)(P)$ and $f(P) \cdot g(P) = (f \cdot g)(P)$. Note that if P is a rational place (and f does not have a pole in P) then $f(P) \in \mathbb{F}_q$. The functions in F always have the same zeros and poles up to multiplicity (called order). An important fact of the theory of algebraic function fields is as follows: call $N_1(F)$ the number of rational places of F. Then over every finite field $\mathbb{F}_q$, there exists an infinite family of function fields $\{F_n\}$ such that their genus $\mathfrak{g}_n$ grow with n and $\lim N_1(F_n)/\mathfrak{g}_n = c_q$ with $c_q \in \mathbb{R}$, $c_q > 0$. The largest constant c_q satisfying the property above is called Ihara's constant $A(q)$ of $\mathbb{F}_q$. It is known that $0 < A(q) \leq \sqrt{q} - 1$ for every finite field $\mathbb{F}_q$. Moreover, $A(q) = \sqrt{q} - 1$ for q square and that for a prime p and any integer $a \geq 1$, $A(p^{2a+1}) \geq \frac{2(p^{a+1}-1)}{p+1+\epsilon}$ where $\epsilon = \frac{p-1}{p^a-1}$. These two results are constructive, since explicit families of function fields attaining these values are known, given in the first case by [GS95, GS96] and in the second case by [BBGS15].

A divisor G is a formal sum of places, $G = \sum c_P P$, such that $c_P \in \mathbb{Z}$ and $c_P = 0$ except for a finite number of P. We call this set of places where $c_P \neq 0$ the support of G, denoted $\mathrm{supp}(G)$. The degree of G is $\deg G := \sum c_P \deg P \in \mathbb{Z}$.

The Riemann-Roch space $\mathcal{L}(G)$ is the set of all functions in F with certain prescribed poles and zeros depending on G (together with the zero function). More precisely if $G = \sum c_P P$, every function $f \in \mathcal{L}(G)$ must have a zero of order at least $|c_P|$ in the places P with $c_P < 0$, and f can have a pole of order at most c_P in the places with $c_P > 0$. The space $\mathcal{L}(G)$ is a vector space over $\mathbb{F}_q$. Its dimension is governed by certain laws (given by the so-called Riemann-Roch theorem). A weaker version of that theorem called Riemann's theorem states that if $\deg G \geq 2\mathfrak{g} - 1$ then $\dim \mathcal{L}(G) = \deg(G) - \mathfrak{g} + 1$. On the other hand, if $\deg G < 0$, then $\dim \mathcal{L}(G) = 0$.

Given $f, g \in \mathcal{L}(G)$ its product $f \cdot g$ is in the space $\mathcal{L}(2G)$.

The following is a generalization of Lemma 4.

Lemma 6. *Let $F/\mathbb{F}_q$ be a function field of genus $\mathfrak{g}$ with k distinct rational places $P_1, P_2, \ldots, P_k$. Let G be a divisor of F such that $\mathrm{supp}(G) \cap \{P_1, \ldots, P_k\} = \emptyset$ and $\dim_{\mathbb{F}_q} \mathcal{L}(G) - \dim_{\mathbb{F}_q} \mathcal{L}(G - \sum_{i=1}^{k} P_i) = k$. If there is a place R of degree m with $m > 2\deg(G)$, then there exists an $(k, m)_q$-RMFE.*

Proof. Consider the map

$$\pi : \mathcal{L}(G) \to \mathbb{F}_q^k; \quad f \mapsto (f(P_1), \ldots, f(P_k)).$$

Then the kernel of π is $\mathcal{L}(G - \sum_{i=1}^{k} P_i)$. Since $\dim_{\mathbb{F}_q} \mathrm{Im}(\pi) = \dim_{\mathbb{F}_q} \mathcal{L}(G) - \dim_{\mathbb{F}_q} \mathcal{L}(G - \sum_{i=1}^{k} P_i) = k$, π is surjective. Choose a subspace W of $\mathcal{L}(G)$ of dimension k such that π induces an isomorphism between W and $\mathbb{F}_q^k$.

We write by $\mathbf{c}_f$ the vector $(f(P_1), \ldots, f(P_k))$, and by $f(R)$ the evaluation of f in the higher degree place R, for a function $f \in \mathcal{L}(2G)$. We now define

$$\phi : \ \pi(V) = \mathbb{F}_q^k \to \mathbb{F}_{q^m}; \quad \mathbf{c}_f \mapsto f(R) \in \mathbb{F}_{q^m}.$$

Note that the above $f \in W$ is uniquely determined by $\mathbf{c}_f$. Moreover ϕ is $\mathbb{F}_q$-linear and injective since $\deg(R) > \deg(G)$.

Define

$$\tau : \ \mathcal{L}(2G) \to \mathbb{F}_{q^m}; \quad f \mapsto f(R) \in \mathbb{F}_{q^m}.$$

Then τ is $\mathbb{F}_q$-linear and injective since $m = \deg(R) > \deg(2G)$.

Define the map

$$\psi' : \ \mathrm{Im}(\tau) \subseteq \mathbb{F}_{q^m} \to \mathbb{F}_q^k; \quad f(R) \mapsto (f(P_1), \ldots, f(P_k)) \in \mathbb{F}_q^k.$$

Note that the above $f \in \mathcal{L}(2G)$ is uniquely determined by $f(R)$. ψ is $\mathbb{F}_q$-linear and surjective (but not injective). We extend ψ' from $\mathrm{Im}(\tau)$ to all of $\mathbb{F}_{q^m}$ linearly and call the resulting map ψ. We obtain thus the pair (ϕ, ψ).

For any $\mathbf{c}_f, \mathbf{c}_g \in \mathbb{F}_q^k$ we have

$$\psi(\phi(\mathbf{c}_f) \cdot \phi(\mathbf{c}_g)) = \psi(f(R) \cdot g(R)) = \psi((f \cdot g)(R)) = \mathbf{c}_{fg} = \mathbf{c}_f * \mathbf{c}_g,$$

where $f, g \in W$ are uniquely determined from $\mathbf{c}_f, \mathbf{c}_g$ as explained above. Note that $(fg)(R)$ belongs to $\mathrm{Im}(\tau)$ since $fg \in \mathcal{L}(2G)$. We conclude that (ϕ, ψ) defined above is an $(k, m)_q$-RMFE.

Corollary 1. *Let $F/\mathbb{F}_q$ be a function field of genus $\mathfrak{g}$ with k distinct rational places and a place of degree $m \geq 2k+4\mathfrak{g}-1$. Then there exists an $(k, m)_q$-RMFE.*

Proof. We take G a divisor of degree $k + 2\mathfrak{g} - 1$ whose support is disjoint with the promised set of k rational places. Then, since both $\deg G \geq 2\mathfrak{g} - 1$ and $\deg(G - \sum_{i=1}^{k} P_i) \geq 2\mathfrak{g} - 1$ we can apply the Riemann Theorem to conclude that $\dim_{\mathbb{F}_q} \mathcal{L}(G) - \dim_{\mathbb{F}_q} \mathcal{L}(G - \sum_{i=1}^{k} P_i) = \deg(G) - \mathfrak{g} + 1 - (\deg(G) - \mathfrak{g} + 1 - k) = k$. We are then in the conditions of Lemma 6.

Proposition 5 ([Sti09], Theorem 5.2.10 (c)). *For every function field $F/\mathbb{F}_q$, and all $m \in \mathbb{N}$ with $2\mathfrak{g}+1 \leq q^{(m-1)/2}(\sqrt{q}-1)$, there exists a place in F of degree m. In particular this holds for every $m \geq 4\mathfrak{g} + 3$, regardless of q.*

This implies that the condition about the existence of the high degree place in Corollary 1 is in fact always satisfied as soon as $k \geq 2$, since any $m \geq 2k + 4\mathfrak{g} - 1$ satisfies the inequality in the proposition above.

Now we can show the main theorem of this section

Theorem 5. *There exists a family of $(k,m)_q$-RMFE with $k \to \infty$ and $m = O(k)$. More concretely*

$$\frac{m}{k} \to 2 + \frac{4}{A(q)}.$$

Proof. Take a family $\{F_\ell\}$ of function fields over $\mathbb{F}_q$ of growing genus $\mathfrak{g}_\ell \to \infty$ with $N_1(F_\ell)/\mathfrak{g}_\ell \to A(q)$. Since $N_1(F_\ell)$ is the number of distinct rational places of F_ℓ, we can take $k = N_1(F_\ell)$. Moreover we take $m = 2k + 4\mathfrak{g}_\ell - 1$. These parameters satisfy all conditions in Corollary 1 and therefore the construction above yields a $(k,m)_q$-RMFE.

For $q = 2$ a direct application of this result, together with the bound $A(2) \geq 97/376$ from [XY07] yields a family of $(k,m)_2$-RMFEs with

$$\frac{m}{k} \to 2 + \frac{4}{A(2)} \leq 2 + \frac{4 \times 376}{97} \approx 15.51.$$

4.1 An Explicit Construction over $\mathbb{F}_2$

The result above for $q = 2$ is not explicit, since the bound for $A(2)$ was attained by a non-explicit of function fields. In this section we will show an explicit construction of a family of RMFEs over $\mathbb{F}_2$ with a constant asymptotic ratio. This example also shows that, fortunately, as was the case for practical values of k, the expansion expressed by the asymptotic ratio m/k can be quite small.

Proposition 6. *There exists a constructive family of $(k,m)_{32}$-RMFE with $k \to \infty$ and $\frac{m}{k} \to 62/21$.*

Proof. This comes from applying Theorem 5, that implies the existence of a family of $(k,m)_{32}$-RMFE with $k \to \infty$ and $\frac{m}{k} \to 2 + \frac{4}{A(32)}$.

Now we use that for every prime p and every $a \geq 1$, we have $A(p^{2a+1}) \geq \frac{2(p^{a+1}-1)}{p+1+\epsilon}$ (where $\epsilon = \frac{p-1}{p^a-1}$) and that this is achieved for the explicit construction in [BBGS15]. In particular $p = 2$, $a = 2$ gives $A(32) \geq 21/5$. This means $2 + \frac{4}{A(32)} \leq 62/21$ and concludes the proof.

Corollary 2. *There exists a constructive family of $(k,m)_2$-RMFE with $k \to \infty$ and*

$$\frac{m}{k} \to 4.92...$$

Proof. Applying the concatenation in Lemma 5 to the $(3,5)_2$-RMFE (from Lemma 4) and the family of $(k_1,m_1)_{32}$-RMFE with $\frac{m_1}{k_1} \to 62/21$ provides a family of $(3k_1, 5m_1)_2$-RMFE. Note that $\frac{5m_1}{3k_1} \to 5/3 \times 62/21 = 4.92...$

5 Proof of Theorem 1

The last step towards proving Theorem 1 is how to instantiate the protocol π' that securely computes the arithmetic circuit over $\mathbb{F}_{2^m}$. We use the protocol by Beerliová-Trubíniová and Hirt [BH08].

Theorem 6 ([BH08]). *There is a protocol π' which computes an arithmetic circuit over a field $\mathbb{F}_{2^m}$, where $|\mathbb{F}_{2^m}| > 2n$, with a communication complexity of $O((c_I+c_M+c_R)\cdot n+D_M\cdot n^2+n^3)$ field elements, where D_M is the multiplicative depth of the circuit.*

The protocol π' satisfies all conditions in Sect. 3. In particular it is a secret-sharing based protocol where the secret sharing scheme used is degree t-Shamir's secret sharing scheme over $\mathbb{F}_{2^m}$.

Proof (of Theorem 2). We use Theorem 4 with a (ϕ, ψ) from the family of (k, m)-RMFEs with $m = \Theta(k)$ constructed in Sect. 4 and the protocol π' from Theorem 6. The total communication complexity is $O((c_I+c_M+c_R)\cdot n+D_M\cdot n^2+n^3)$ elements of $\mathbb{F}_{2^m}$, and therefore $O(nm)$ bits per gate of the circuit. Note that this allows to compute $k = \Theta(m)$ evaluations of the circuit and therefore the amortized complexity is $O(n)$ bits per gate.

We point out one optimization that it is possible when we combine our compiler with [BH08]. Indeed the input phase in [BH08] consists in selecting a sharing $[r]$ of a random element in $\mathbb{F}_{2^m}$ which has been generated in their preprocessing phase and opening this privately to the party P_i holding the input $a_i \in \mathbb{F}_{2^m}$, who broadcasts the difference of the random element and a_i so that the rest of the parties update their shares. If we use our compiler as described, in the next step P_i would prove $a_i \in \mathrm{Im}\ \phi$. Rather than executing these two phases, we can merge these two processes in one step: instead of using $[r]$ for a uniformly random $r \in \mathbb{F}_{2^m}$, we can have parties take $[r']$ for a uniformly random $r' \in \mathrm{Im}\,\phi$, generated in our preprocessing phase by `RandElSub`$(\mathrm{Im}\ \phi)$, then open this to P_i, and have P_i broadcast the difference of $r' - a_i$. The other parties can now verify that $r' - a_i \in \mathrm{Im}\ \phi$ and if so, update their shares accordingly.

6 Proof of Theorem 2

We combine our amortization technique with the packed secret sharing paradigm to further decrease the communication complexity in the case where the adversary is suboptimal.

The result is based on the observation that one can replace Shamir's secret sharing scheme by packed Shamir's secret sharing in the protocol from [BH08].

Theorem 7. *There is a multiparty computation protocol for n parties that evaluates $\ell = \Theta(n)$ instances of an arithmetic circuit over $\mathbb{F}_q$ (where $q \geq 2n$) with c_I input, c_R random and c_M multiplication gates, by communicating $O(c_I n + c_R n + c_M n + D_M n^2 + n^3)$ field elements, where D_M denotes the multiplicative depth of the circuit. The protocol is secure against an active adversary corrupting $t < (n - 2\ell + 2)/3$ players.*

In order to sketch an argument for this result, we briefly describe how [BH08] works. This protocol has a preprocessing phase and a computation phase. In the computation phase, all inputs and intermediate values are shared among the network of parties using Shamir's secret sharing of degree t, denoted by $[\cdot]_t$. In order to process multiplication gates, the protocol uses the well known randomization technique due to Beaver [Bea91], which relies on auxiliary shared triplets $([a]_t, [b]_t, [c]_t)$, where a, b are random field elements and $c = ab$; these have been computed in the preprocessing phase.

The preprocessing phase uses player elimination and its goal is to generate the aforementioned triplets as well as "individual" sharings of random elements that are used in input and random gates. The crucial step in order to obtain these triplets is to be able to generate "double sharings" of random elements, more specifically one needs to generate pairs $[r]_t, [r]_{t'}$ and $[r]_t, [r]_{2t'}$ (where t' as always is the updated corruption tolerance after player elimination). This is done by means of hyper-invertible matrices in a way we have already sketched in Sect. 2. Here an important point underlying the protocol is that the product of two degree-t polynomials is a degree-$2t$ polynomial. A small detail is that at some points of the computation the parties need to generate, from a publicly known value x, the 0-degree sharing $[x]_0$. This is simply that each party defines as share the value x.

Finally, the other important point to notice regards reconstruction of secrets: throughout the protocol two reconstruction protocols are used for the secret sharing scheme: `ReconsPriv` reconstructs the secret privately towards a party, and consists on all other parties sending their shares to her. The protocol `ReconsPubl`, which we have already detailed in this paper, reconstructs a batch of secrets publicly, with amortized communication. Given a sharing $[\cdot]_d$, the secret can be reconstructed (with either protocol) t'-robustly if $d < n' - 2t'$ and t'-detectably (meaning that either the correct secret is reconstructed or the party detects the sharing is erroneous) if $d < n' - t'$. Hence t and t'-degree sharings can be robustly reconstructed and $2t'$-degree sharings can be detectably reconstructed. This is enough for the purposes of [BH08].

We describe how this would be adapted so that packed Shamir secret sharing is used instead. We recall how packed Shamir secret sharing for n parties and with secrets in $\mathbb{F}_q^\ell$ (where $\ell < n$), is defined; by assumption $\mathbb{F}_q$ has at least $n + \ell < 2n$ elements. Fix $\omega_1, \ldots, \omega_\ell$, $\alpha_1, \ldots, \alpha_n$ pairwise distinct points in $\mathbb{F}_q$. Then, for a degree $d \geq \ell - 1$, degree d-packed Shamir secret sharing works as follows: given $\mathbf{s} = (s_1, \ldots, s_\ell) \in \mathbb{F}_q^\ell$, a polynomial $f \in \mathbb{F}_q[X]$ is chosen uniformly at random among all polynomials of degree $\leq d$ with $f(\omega_j) = s_j$ for $j = 1, \ldots, \ell$. Then $[\mathbf{s}]_d^\ell$ is the vector $(f(\alpha_1), \ldots, f(\alpha_n))$ where $f(\alpha_i)$ is sent to the i-th player. This packed scheme has $(d-\ell+1)$-privacy: any set of $d-\ell+1$ shares gives no information about the secret. On the other hand, it has exactly the same reconstruction properties (even in the presence of errors) as degree d-standard Shamir. In particular, it has $d+1$-reconstruction ($d+1$ honest shares determine the secret), it is t-robust as long as $d < n - 2t$ and it has t-detectable reconstruction as long as $d < n - t$.

We can turn [BH08] into a protocol that computes ℓ parallel evaluations of an arithmetic circuit over $\mathbb{F}_q$ with $O(1)$ field elements communicated per gate by doing the following modifications. The standard Shamir sharings $[\cdot]_t$, $[\cdot]_{t'}$, $[\cdot]_0$ and $[\cdot]_{2t'}$ in [BH08] are substituted by packed Shamir sharings $[\cdot]^\ell_{t+\ell-1}$, $[\cdot]^\ell_{t'+\ell-1}$, $[\cdot]^\ell_{\ell-1}$ and $[\cdot]^\ell_{2t'+2\ell-2}$, respectively. Multiplication of secrets in $\mathbb{F}_q$ becomes now componentwise multiplication in $\mathbb{F}_q^\ell$.

One can then verify that all properties we need are still preserved: first, we have that $2t'+2\ell-2 = 2(t'+\ell-1)$, which is needed in the shared triplets generation; moreover, the main scheme is now $[\cdot]^\ell_{t+\ell-1}$, which is still t-private; furthermore, under the assumption that $t < (n-2\ell+2)/3$, we have $2t'+2\ell-2 < n'-t'$ and $t'+\ell-1 \leq t+\ell-1 < n'-2t'$, so $[\cdot]^\ell_{t+\ell-1}$, $[\cdot]^\ell_{t'+\ell-1}$ are have t'-robust reconstruction and $[\cdot]^\ell_{2t'+2\ell-2}$ has t'-detectable reconstruction; finally $[\cdot]^\ell_{\ell-1}$ is a degenerate secret sharing scheme that takes the unique polynomial of degree $\ell-1$ that interpolates the secret and generates the corresponding shares, i.e., every party can compute her share given the secret, so it plays exactly the role which is needed from $[\cdot]_0$ in the original protocol. The double sharing generation via hyper-invertible matrices still works, because it can be still captured with our notion of GLSSS. Indeed we will have a $\mathbb{F}_q$-GLSSS where the secret is now in $\mathbb{F}_q^\ell$ but each of the shares in $\mathbb{F}_q^2$ and consists of a share with $[\cdot]^\ell_d$, and another with $[\cdot]'^\ell_d$ (the protocol will need to invoke this with $d = t+\ell-1$, $d' = t'+\ell-1$ and with $d = t+\ell-1$, $d' = 2t'+2\ell-2$).

This establishes Theorem 7 given that the communication complexity of this modified protocol is the same as that of [BH08], but it computes ℓ evaluations of the arithmetic circuit under the weaker assumption that $t < (n-2\ell+2)/3$.

Now we show Theorem 2.

Proof (of Theorem 2). We describe a secure multiparty computation protocol for n parties with perfect security against an adversary corrupting $t < (n-2\ell+2)/3$ parties that computes simultaneously $k\ell$ evaluations of the binary circuit C with communication $O(k\ell)$ bits per gate of the circuit, and hence $O(1)$ bits per gate per instance. We recover the theorem by taking $\ell = \epsilon n/2$.

We briefly describe how to modify our compiler from Sect. 3 so that it works with packed Shamir secret sharing.

Take (ϕ, ψ) from a family of $(k,m)_2$-RMFE with $m = \Theta(k)$ and such that $2^m > 2n$. Define $\Phi : \mathbb{F}_2^{k\ell} \to (\mathbb{F}_{2^m})^\ell$ and $\Psi : (\mathbb{F}_{2^m})^\ell \to \mathbb{F}_2^{k\ell}$ that respectively consist in applying ϕ to each block of k coordinates of the input and ψ to each coordinate of the input. Parties now encode their vectors of inputs with Φ and provide these to the protocol π' (for example the vers and they need to prove that their inputs are in Φ. In order to do this the parties need to apply `RandElSub` to $\operatorname{Im}\Phi = (\operatorname{Im}\phi)^\ell$ in the preprocessing phase. At multiplication gates, the parties need to compute $[\Phi(\Psi(\mathbf{a}))]$ from $[\mathbf{a}]$ which can be done in similar fashion as in Sect. 3 but using random sharings generated by applying `RandElSub` to the $\mathbb{F}_2$-subspace $\mathcal{W} = \{(\mathbf{x}, \Phi(\Psi(\mathbf{x}))) : \mathbf{x} \in \mathbb{F}_{2^m}^\ell\}$ in the preprocessing phase. We also need to use that `ReconsPubl` is t'-robust as explained above.

Because the secret sharing scheme is now the packed version of Shamir's, we attain the same complexity as in our protocol, but now we are computing

$k\ell = \Theta(kn)$ evaluations of the circuit. The amortized complexity per gate per instance of the compiler is therefore $O(1)$ bits. Using this in combination with the packed version of [BH08] described above as protocol π' proves the theorem.

Acknowledgements. The work of Ronald Cramer and Chen Yuan was supported in part by ERC Advanced Grant No. 74079 (ALGSTRONGCRYPTO). Part of Chen Yuan's work was performed while he was employed at NTU in Singapore. The authors thank Martin Hirt, Ivan Damgård, Yuval Ishai, and Jesper Buus Nielsen for helpful discussions and the anonymous reviewers for their valuable comments.

References

[BBGS15] Bassa, A., Beelen, P., Garcia, A., Stichtenoth, H.: Towers of function fields over non-prime finite fields. Moscow Math. J. **15**(1), 1–29 (2015)

[Bea91] 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

[BGW88] Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing, Chicago, Illinois, USA, 2–4 May 1988, pp. 1–10 (1988)

[BH08] Beerliová-Trubíniová, Z., Hirt, M.: Perfectly-secure MPC with linear communication complexity. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 213–230. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78524-8_13

[BMN17] Block, A.R., Maji, H.K., Nguyen, H.H.: Secure computation based on leaky correlations: high resilience setting. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part II. LNCS, vol. 10402, pp. 3–32. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63715-0_1

[Bra85] Bracha, G.: An o(log n) expected rounds randomized byzantine generals protocol. In: Proceedings of the 17th Annual ACM Symposium on Theory of Computing, Providence, Rhode Island, USA, 6–8 May 1985, pp. 316–326 (1985)

[CC88] Chudnovsky, D., Chudnovsky, G.: Algebraic complexities and algebraic curves over finite fields. J. Complex. **4**, 285–316 (1988)

[CCCX09] Cascudo, I., Chen, H., Cramer, R., Xing, C.: Asymptotically good ideal linear secret sharing with strong multiplication over *Any* fixed finite field. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 466–486. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_28

[CCX11] Cascudo, I., Cramer, R., Xing, C.: The torsion-limit for algebraic function fields and its application to arithmetic secret sharing. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 685–705. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_39

[CCX12] Cascudo, I., Cramer, R., Xing, C.: The arithmetic codex. In: 2012 IEEE Information Theory Workshop, Lausanne, Switzerland, 3–7 September 2012, pp. 75–79 (2012)

[CDN15] Cramer, R., Damgård, I., Nielsen, J.B.: Secure Multiparty Computation and Secret Sharing. Cambridge University Press, Cambridge (2015)

[DI06] Damgård, I., Ishai, Y.: Scalable secure multiparty computation. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 501–520. Springer, Heidelberg (2006). https://doi.org/10.1007/11818175_30

[DIK10] Damgård, I., Ishai, Y., Krøigaard, M.: Perfectly secure multiparty computation and the computational overhead of cryptography. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 445–465. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_23

[DN07] Damgård, I., Nielsen, J.B.: Scalable and unconditionally secure multiparty computation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 572–590. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_32

[DNPR16] Damgård, I., Nielsen, J.B., Polychroniadou, A., Raskin, M.: On the communication required for unconditionally secure multiplication. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part II. LNCS, vol. 9815, pp. 459–488. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53008-5_16

[FY92] Franklin, M.K., Yung, M.: Communication complexity of secure computation (extended abstract). In: Proceedings of the 24th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, 4–6 May 1992, pp. 699–710 (1992)

[GS95] García, A., Stichtenoth, H.: A tower of Artin-Schreier extensions of function fields attaining the Drinfeld-Vlăduţ bound. Invent. Math. **121**(1), 211–222 (1995)

[GS96] Garcia, A., Stichtenoth, H.: On the asymptotic behaviour of some towers of function fields over finite fields. J. Number Theory **61**(2), 248–273 (1996)

[HMP00] Hirt, M., Maurer, U.M., Przydatek, B.: Efficient secure multi-party computation. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 143–161. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44448-3_12

[IKOS09] Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge proofs from secure multiparty computation. SIAM J. Comput. **39**(3), 1121–1152 (2009)

[Sha79] Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)

[Sti09] Stichtenoth, H.: Algebraic Function Fields and Codes. Graduate Texts in Mathematics, vol. 254, 2nd edn. Springer, Berlin (2009). https://doi.org/10.1007/978-3-540-76878-4

[XY07] Xing, C., Yeo, S.L.: Algebraic curves with many points over the binary field. J. Algebra **311**(2), 775–780 (2007)