Asymptotically-Good Arithmetic Secret Sharing over $\mathbb{Z}/p^\ell\mathbb{Z}$ with Strong Multiplication and Its Applications to Efficient MPC

Ronald Cramer, Matthieu Rambaud, and Chaoping Xing

[1] CWI Amsterdam & Leiden University
[2] Telecom Paris, Institut polytechnique de Paris
[3] Nanyang Technological University, Singapore

Abstract. The current paper studies information-theoretically secure multiparty computation (MPC) over rings $\mathbb{Z}/p^\ell\mathbb{Z}$. This is a follow-up research of recent work on MPC over rings $\mathbb{Z}/p^\ell\mathbb{Z}$. In the work of [Abs+19a, TCC2019], a protocol based on the Shamir secret sharing over $\mathbb{Z}/p^\ell\mathbb{Z}$ was presented. As in the field case, one of the drawbacks of the protocol in [Abs+19a, TCC2019] is that the share size has to grow as the number of players increases. Then several MPC protocols were developed in [Abs+20, Asiacrypt 2020] in order to solve the problem of growing share size. However, the MPC protocols in [Abs+20, Asiacrypt 2020] suffer from several problems: (i) the offline multiplication gate has super-linear communication complexity; (ii) the share size is doubled for the most important case, namely over $\mathbb{Z}/2^\ell\mathbb{Z}$ due to infeasible lifting of self-orthogonal codes from fields to rings; (iii) most importantly, the BGW model could not be applied via the secret sharing given in [Abs+20, Asiacrypt 2020] due to lack of strong multiplication.
Our contribution in this paper is three fold. Firstly, we overcome all the drawbacks in [Abs+19a; Abs+20] mentioned above. Secondly, we establish an arithmetic secret sharing with strong multiplication which is the most important primitive in the BGW model. Thirdly, we generalize Reverse Multiplication Friendly Embeddings (RMFE) from fields to Galois rings. Note that RMFE has become a standard technique for amortized communication complexity in MPC (see [Cas+18, CRYPTO'18] and [DLN19, CRYPTO'19]).
To obtain our theoretical results, we use the existence of lifts of curves over rings, then use the known results stating that Riemann-Roch spaces are free modules. To make our scheme practical, we start from good algebraic geometry codes over finite fields obtained from existing computational techniques. Then we present, and implement, an efficient algorithm to Hensel-lift the generating matrix of the code, such that the multiplicative conditions are preserved over rings. Existence of this specific lift is guaranteed by the previous theory. On the other hand, a random lifting of codes over from fields to Galois rings does not preserve multiplicativity in general. (Notice that our indirect method is motivated by the fact that, following the theory instead, would require to "preprocess" the curve under a form with "smooth" equations, in particular with many variables, before lifting it. But computing on these objects over rings is out of the scope of existing research). Finally we provide efficient elementary methods for sharing and (robust) reconstruction of secrets over rings. As a result, arithmetic secret sharing over $\mathbb{Z}/p^\ell\mathbb{Z}$ with strong multiplication can be efficiently constructed and practically applied.

1 Introduction

MPC over rings $\mathbb{Z}/p^\ell\mathbb{Z}$, is a model where the computation complexity is counted in terms of elementary additions and multiplications in $\mathbb{Z}/p^\ell\mathbb{Z}$, and the communication complexity is the number of elements of $\mathbb{Z}/p^\ell\mathbb{Z}$ sent. So this model is the "closest to machine" when $p = 2$ and ℓ is the length of a machine integer. By contrast, the previous model of MPC are arithmetic circuits in $\mathbb{F}_p$. But computations modulo p are not natively done by processors. Unless $p = 2$ (MPC over binary circuits), which then comes at the cost of encoding only one bit in a whole machine integer. It appears from the literature that emulating MPC over the integers, from MPC in $\mathbb{F}_p$, incurs a substantial overhead in complexity. For instance, the protocol of [Dam+06] for bit decomposition of numbers modulo a large p, in order to perform secure comparisons, costs $\log(p)\log(\log(p))$ secure multiplications modulo p. Whereas comparisons directly between integers modulo a power of 2 are much more efficient ([Ara+18]).

1.1 Related works

In a recent line of work on efficient MPC over $\mathbb{Z}/p^\ell\mathbb{Z}$ (which is most relevant for $p = 2$), significant advances have been made in order to avoid the overhead incurred by this emulation, by redesigning basic arithmetic MPC so as to work "more directly" over the ring in question. The first published paper [Cra+18] in this line introduces the SPD$\mathbb{Z}_{2^k}$ protocol, a full redesign of the well-known SPDZ-protocol [Dam+12], the benchmark for the case of cryptographic security with dishonest majority in Beaver's preprocessing model, that works directly over the rings in question and that is essentially as efficient as the most efficient SPDZ-incarnation. See also the compilers of [DOS18; Abs+19b] from passive security over rings to active security over rings. For more discussion about practical advantages, see [Cra+18] and its follow-up [Dam+19], which also reports on applications to machine-learning that significantly outperform approaches from field-based MPC. Maliciously secure machine learning directly over the integers is now becoming the standard (e.g. [PS20]).

Closer to us is the line of work [Abs+19a; Abs+20], that aims at answering the question if information theoretically secure MPC over $\mathbb{Z}/p^\ell\mathbb{Z}$, has complexity equal to that of the one of MPC over $\mathbb{F}_p$. The answer is simple: suppose that one has the choice between two protocols with the same complexities: measured over $\mathbb{F}_p$ for the former, and over $\mathbb{Z}/p^\ell\mathbb{Z}$ for the latter. Then the latter protocol is automatically the most efficient to securely compute any function over $\mathbb{Z}/p^\ell\mathbb{Z}$, since no emulation is needed. The present paper addresses this question mainly in the plain model [CDN15, §5], denoted "BGW", that is: assuming only authenticated channels and requiring perfect security. So in particular, we have that the number of malicious corruptions is $t < n/3$, since no broadcast channel is available beyond this bound in the BGW model. [Abs+19a] considers MPC in the BGW model over rings. It adapts the protocol of [BH08], including the ASSSM mentioned above adapted from Shamir, and so inherits the amortized suboptimal $O(n\log(n))$ communication complexity of [BH08].

Let us now review the setting of Rabin&Ben-Or, denoted as "honest majority" that is: assuming a broadcast and requiring unconditional statistical security, tolerating $t < n/2$ corruptions. Until recently the best amortized communication complexity over fields was [BFO12], in $n\log(n)$, plus a term in n^2 times the depth of the circuit. This term has just been removed by [GSZ20]. However there is still a $\log(n)$ overhead remaining, even over fields. As noticed in [Abs+20] (at the beginning of §6), the main tool of [GSZ20] is the Batched Triple Sacrifice protocol of [BFO12], that checks correctness of shared Beaver triples. It runs in the offline phase, and has $O(n\log(n))$ communication complexity (to be sure, the notation ϕ in [GSZ20] stands for the log of the size of the field, it is required to be $\phi \geq \log(n)$). This Batched Triple Sacrifice was then carried over rings in [Abs+20], resulting in an overall amortized communication complexity in $O(n\log(n))$ also over rings. In addition to the above super-linear offline communication complexity, there are some other drawbacks in [Abs+20]: (i) their secret sharing schemes with (standalone) Multiplication are constructed with a double sharing, which thus doubles the size (ii) their way around this doubling uses asymptotically good families of self-dual codes, for which no practical construction is known (by contrast with good families of codes from algebraic curves / function fields, whose computation is widely studied); (iii) most importantly, the secret sharing scheme given in [Abs+20] cannot be adopted for the BGW model due to lack of Strong Multiplicativity.

1.2 Our focus

Our main focus are the two fundamental primitives for MPC in the BGW model. We also deal with the asymptotic complexity of MPC under honest majority, especially the so-far $\log(n)$ communication overhead, that hits in particular the protocols given in [Abs+20]. We are finally concerned by the computational efficiency of general reconstruction methods of linear secret sharing schemes over rings, which was not dealt with at all in previous works.

The first primitive for MPC in the BGW model is *arithmetic secret sharing with Strong multiplication* (ASSSM). Recall that such a scheme with respect to adversary bound t, guarantees both: secrecy from any t shares, and, reconstruction of the product of two shared secrets, from any list of $n-t$ products of pairwise shares of these secrets. By contrast, secret sharing with (standalone) Multiplication only, requires *all* the n pairwise products of shares for reconstruction of the product. The simplest example is the Verifiable secret sharing of BGW itself Importance of Strong multiplication is formalized in the Theorem 3 of [CDM00], as the building block of error-free MPC protocols. Namely, it is emphasized in [CDN15, p114] as the tool enabling not to restart the execution of the protocol, even under openly misbehavior of a participant. Notice that the $\log(n)$ overhead is inherited from Shamir's secret sharing, which operates in finite fields of cardinality *at least as large* as the number of players. This limitation was removed in the series of papers [CC06; Cas+09; CCX11] using algebraic geometric codes over fields. Notice that these state of the art ASSSM and constant size of shares, motivated the "MPC in the head technique" [Ish+07], see [Cas16, §5] for other

applications. In this paper we ask if the same tight size of shares is achievable over rings. We also ask if the same efficiency of constructions is achievable as over fields [Hes02; Khu04; Shu+01; SG20]. Also, much optimisation has been made for sharing/reconstruction algorithms over fields [NW17; SW99; GS99].

The second primitive are *Reverse multiplication friendly embeddings* (RMFE). They enable to emulate several circuits in parallel over small finite fields $\mathbb{F}_p$, from a single circuit over a large extension $\mathbb{F}_{p^m}$. They are introduced in [Cas+18], and are the main tool for the upper bounds of [DLN19], and of [BMN18; DLS20; CG20], to linearize the amortized communication complexity of perfectly secure MPC, over multiple instances of the same circuit (with possibly different inputs), while preserving an *optimal* corruption tolerance. Recall that a RMFE [Cas+18, Definition 1] (recalled in §5.4), is an embedding from some vector space $\mathbb{F}_q^k$ over $\mathbb{F}_q$, into some field extension $\mathbb{F}_{q^m}$, which "carries" the multiplication in $\mathbb{F}_{q^m}$ into the component-wise multiplication of vectors in $\mathbb{F}_q^k$ (the same one as for multiplicative secret sharing). The larger the ratio k/m, the better the complexity of MPC is amortized. Again, RMFE with polynomial encoding (as in Shamir secret sharing) exist up to $k \leq q+1$. And again, this limit of the field size was removed in [Cas+18, Theorem 3] with constructions from algebraic geometry coding. Namely, they achieve for any fix q, a slowly growing infinite family of parameters k, m such that the ratios k/m are lower bounded by a *constant*, which is optimal. We thus ask if the same ratios are achievable over rings, and if constructions are as efficient.

1.3 Our contributions

1.3.1 Asymptotically optimal Strong multiplication over rings

Main Theorem 1. For every p and ℓ, for any fixed even r larger than some $\hat{r}(p)$, we have a slowly growing infinite family of number of integers n, such that there exists an ASSSM over the fixed ring $\mathbb{Z}/p^\ell\mathbb{Z}$, with n shares, with constant size of shares r and t-adversary bound such that $1/3 - t/n > 0$ a constant arbitrarily close to 0 (in $O(p^{-r/2})$).

More precisely and generally: all parameters $(n, p, r(p), t)$ published in [CC06; Cas+09] over fields $\mathbb{F}_p$, *also* hold over rings. We have stronger than privacy: *uniformity* of the projection on any t shares of the space of vectors of shares of any given fixed secret. Moreover, the scheme obtained by reduction modulo p may be assumed to be asymptotically good as well. [4] Last but not least, sharing and reconstruction over $\mathbb{Z}/p^\ell\mathbb{Z}$ have the *same computational complexity* than for ASSSM over fields $\mathbb{F}_p$.

We thus close the gap between the complexity of ASSSM over fields, and over rings. Thus our result is tight, so we do not further justify why our construction makes a detour through "Galois rings" extensions. Although we hope that it will be clear from §1.4, §2.1 (and also [Abs+19a; Abs+20]) that these objects

[4] This fact is quite useful in some practical protocol applications but it is not strictly necessary for general arithmetic MPC

play the same auxiliary role over rings, as finite fields extensions do over fields. Concretely, under the hood is that $\mathbb{F}_p$ is embedded into $\mathbb{F}_{p^r}$ in order to access ASSSM/RMFE with good properties, which are then lifted over Galois ring extensions, then seen over $\mathbb{Z}/p^\ell\mathbb{Z}$. But for simplicity, we refrained from stating that the above theorem also holds for any Galois ring extension of degree r, with the same parameters $(n, p, r(p), t)$ than [CC06; Cas+09] over $\mathbb{F}_{p^r}$. We also kept simple the formula and made explicit only the case where t is close to $n/3$. To be sure, the parameters of [Cas+09] also enable smaller sizes of shares, at the cost of a lower t (descent arguments). The last claim will follow from the algorithms of Theorem 5 below (technically, because the componentwise squares of the codes constructed are included in "free codes" of dimension as small as over finite fields).

1.3.2 Optimal communication of MPC under honest majority. We remove the so-far aforementionned amortized $\log(n)$ communication overhead, which also held over fields. The bottleneck comes from the offline phase, in the subprotocol of [BFO12] checking triples, e.g., as transposed over rings in [Abs+20, §6.6]. We address it in Proposition 20 with an alternative Batched Triple Sacrifice protocol that operates on rings of fixed sizes, for any number of triples to be checked. Recall that the baseline method of [BFO12] proceeds by encoding many triples in three polynomials, then succinctly check the multiplicative relation between these polynomials. We start by replacing it by a construction over fields of fixed size, which is closely related to the strong multiplication property. So this already improves the amortized complexity of the [BFO12] protocol (as recently made insensitive to depth, by [GSZ20]). We then lift the construction over rings with the same methode as before. This results in:

Main Theorem 2. In the model of [RB89]: honest majority and assuming broadcast, then there exists a statistically secure MPC protocol with guaranteed output delivery and amortized communication complexity linear in the number of players per (both online and offline) multiplication gate.

1.3.3 Amortized complexity of MPC over rings. We construct, in §5.2 an infinite family of RMFE over rings with same *constant* asymptotic ratio as the ones of [Cas+18]. Combined with the tight complexities of general LSSS proven in Theorem 5, this enables to carry over rings the results of [Cas+18] with the same computational and communication complexities:

Main Theorem 3. In the BGW-model, there is an efficient MPC protocol for n parties secure against the maximal number of active corruptions $t < n/3$ that computes $\Omega(\log n)$ evaluations of a single circuit over $\mathbb{Z}/p^\ell\mathbb{Z}$ in parallel with an amortized communication complexity (per instance) of $O(n)$ elements of $\mathbb{Z}/p^\ell\mathbb{Z}$ per gate, and same computational complexity than in [Cas+18, Thm 1 & 2]. Combining with the Franklin-Yung paradigm [FY92], we get:

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 circuit over $\mathbb{Z}/p^\ell\mathbb{Z}$ in parallel with an amortized communication complexity (per instance) of $O(1)$ elements of $\mathbb{Z}/p^\ell\mathbb{Z}$ per gate.

1.3.4 Optimal share sizes and computability under honest majority The asymptotically good ASSSM of Theorem 1 have a fortiori standalone Multiplication. So they can be used as a replacement for the schemes constructed in [Abs+20, §4.1]. Especially for $p = 2$, recall that Multiplication of their schemes is obtained via a double sharing, which thus doubles the size of the shares (as stressed in the roadmap of [Abs+20, §3]). Our construction thus *divides their sizes of shares by* 2 *for* $p = 2$. A corollary of above, is that the active protocol presented in Section 6 of [Abs+20], which requires standalone Multiplication, now works with share sizes reduced to half, and now using *computable* families of codes, including from AG/function fields.

1.3.5 Practical computability (continued) Main Theorems 1, 2 and 3 rely on objects (ASSSM or RMFE or the related ones of Proposition 20) with nice asymptotic properties that we prove to exist. We then describe effective algorithms to construct these objects. They are sketched in §3.1 then studied more formally in §4 and in §5.4. They take as input a known family of objects over finite fields, and output "lifts" over rings of these objects. The following theorem then states correctness of these constructions. Noticeably, the proof of correctness does not suppose any knowledge of the techniques deployed to prove existence of the objects. The technicality consists in lifting the coefficients from $\mathbb{F}_p$ to $\mathbb{Z}/p^\ell\mathbb{Z}$, in a way that preserves the Strong multiplication property. However, termination of these algorithms does obviously rely on existence of such "lifts", as guaranteed by the previous results.

Theorem 4. *Starting from any ASSSM over any fixed field $\mathbb{F}_p$ considered in [CC06; Cas+09; CCX11], then, obtaining the lifts over $\mathbb{Z}/p^\ell\mathbb{Z}$ predicted by Main Theorem 1 boils down to solving ℓ instances of a* linear system *over $\mathbb{F}_p$ with $\Omega(n^6)$ coefficients.*

We have an analogous system to obtain the RMFE predicted by Main Theorem 3, from the ones of [Cas+18, Theorem 5].

The complete proof for ASSSM is in Proposition 18. These arguments apply unchanged to carry over rings the explicit construction RMFE done in [Cas+18, Theorem 3]. For sake of completeness we describe in detail the explicit algorithm to lift RMFE over rings in §5.4. We illustrate efficiency of our method in Appendix §C by lifting a strongly multiplicative secret sharing scheme over $\mathbb{F}_{16}$ for 64 players and adversary threshold $t = 13$, into a scheme over the Galois extension of degree four of $\mathbb{Z}/2^{100}\mathbb{Z}$, in a minute on a single processor.

1.3.6 Tight computational complexity of linear secret sharing schemes (LSSS) over rings Although theoretical results for error correction over rings

are shown in [Abs+19a, Construction 1 & Proposition 1], it is not yet clear in the literature if there exists *effective algorithms* for even the simple task of reconstruction of a secret with only erasures. We fill this gap by providing algorithms for sharing and reconstruction of linear secret sharing schemes (LSSS) over rings that arize from *free* codes. In particular it proves our efficiency claims in the Main Theorems above. A free code C over $\mathbb{Z}/p^\ell\mathbb{Z}$ is by definition the linear span of independent vectors with coordinates in $\mathbb{Z}/p^\ell\mathbb{Z}$, in particular it is of same dimension than its reduction $\overline{C}$ modulo p, which is a code over $\mathbb{F}_p$. See also Theorem 25 for useful equivalent properties. In particular, all the ASSSM constructed in this paper have this property, as well as the objects studied in [Abs+20] and in [Abs+19a] (which considers the specific case of Shamir secret sharing over rings). On the other hand, LSSS arising from nonfree codes have bad computational complexity, as we illustrate in Counterexample 10. We provide computational complexities that match the ones over finite fields, so which are *tight*. For simplicity the following theorem is stated over $\mathbb{Z}/p^\ell\mathbb{Z}$, but it will be clear from the proof that it obviously also holds over any Galois ring extension.

Theorem 5. *Let n, ℓ be integers, consider* a free code C *in $(\mathbb{Z}/p^\ell\mathbb{Z})^{n+1}$ and let ψ the corresponding (LSSS) with n shares in $\mathbb{Z}/p^\ell\mathbb{Z}$, such that (without loss of generality) the secret is encoded in the 0-th coordinate of codewords. Denote $\overline{C}$ the code reduced modulo p and $\overline{\psi}$ the corresponding LSSS (which is ψ modulo p) over $\mathbb{F}_p$. We have:*

- *(A) The task of computing a generating matrix of C in systematic form, from any generating matrix of C and, more generally, Gauss pivot, has same computational complexity as modulo p (as long as $n \geq \log(\ell)$).*
- *(A') Then, sharing a secret using ψ (thus of bitsize ℓ times larger) has same computational complexity than using $\overline{\psi}$. As for sharewise multiplication.*
- *(B) Let $I \subset \{1, \ldots, n\}$ be a set of $n - d(\overline{C}) + 2$ indices of shares. Then, there exists a* linear *map $\phi_I : (\mathbb{Z}/p^\ell\mathbb{Z})^{|I|} \longrightarrow \mathbb{Z}/p^\ell\mathbb{Z}$ that reconstructs the secret, with the* same complexity *than a reconstruction map $\overline{\phi_I}$ for $\overline{\psi}$. Moreover, ϕ_I can be compiled from a reconstruction map $\overline{\phi_I}$ for the LSSS modulo p, essentially for the cost of one matrix inversion in $(\mathbb{Z}/p^\ell\mathbb{Z})^{|I|\times|I|}$.*
- *(C) Let $I \subset \{1, \ldots, n\}$ be a nonempty set of indices, such that there exists a robust reconstruction algorithm $\overline{\phi}_I$ tolerating up to τ errors. [That is: on input shares of a secret s with indices in I initially obtained from $\overline{\psi}$, such that at most τ of them were arbitrarily modified, then $\overline{\phi}_I$ outputs s]. Then there exists a robust reconstruction algorithm for ψ_I tolerating up to τ errors, whose computational complexity consists in applying sequentially ℓ times $\overline{\phi}_I$.*

Notice that the matrix inversion required in (B) can be computed using the Gauss pivot of (A).

1.4 Difficulties and intuitions of the constructions

Only algebraic-geometric (AG) constructions such as in [CC06] are so far known to enable ASSSM over fields of constant sizes for an arbitrarily large number

of shares. They follow the same pattern than the scheme of Shamir, which is a particular case. First, select an algebraic curve (e.g. all the points in $\mathbb{F}_q$ plus the "point at infinity", in the case of Shamir). Second, select a "Riemann-Roch" vector space of functions (e.g. the polynomials of degree $\leq d$ in the case of Shamir: said otherwise, the space $\mathcal{L}(d\infty)$ of polynomials "vanishing at order at least d at infinity"). Then, select a particular point P_0 on the curve (e.g. the point 0 in Shamir). To share a secret s, select a function at random in the Riemann-Roch space that evaluates to s at P_0. Then evaluate it on n predefined points of the curve to obtain the shares. In what follows we will instead take a coding-based approach. This has both the advantage to make proofs which are more black box in the AG codes used, and also, our efficient methods will actually directly lift the generating matrices of such AG codes over fields.

To obtain our theoretical results over rings, we use known theorems that state the existence of lifts of curves over rings, then use the results of Judy Walker that state that Riemann Roch spaces are free modules, and also the codes deduced from the above procedure. But we will follow instead a more direct approach to compute the codes concretely. Indeed, the theoretical results require that the curve be represented with "smooth" equations, in particular with many variables. But in practice, good curves are expressed in terms of equations with two variables only. So we do not have efficient methods today to compute smooth lifts of such models in the plane that have many "non smooth" points. Let alone computing Riemann-Roch spaces of smooth models curves over rings, which is out of the scope of existing research (except Walker-Voloch, for smooth curves in the plane). So our direct approach will be to start from good codes over finite fields obtained from the method above. Then lift directly the generating matrix of the code, such that the multiplicative conditions are preserved. On the face of it, there are more constraints than variables. To be sure, we illustrate in §3.1 that lifting at random almost certainly fails to preserve the multiplicative conditions. But our theoretical results imply that a solution exists, which we are able to compute efficiently (see §3.2, and the Appendix §C for larger examples).

1.5 Roadmap

In §2 we show that LSSS derived from free codes over Galois rings have same privacy and reconstruction threshold as over the field modulo p. In §2.3 and §2.4 we present efficient sharing and reconstruction algorithms (proof of Theorem 5 (A) (A') and (B)). We show conversely in Counterexample 10 that *there does not exist a linear reconstruction map* for a large class of linear codes over rings which are not free. This is why we focus on LSSS derived from free codes. Let us mention for the hurry reader that the results in §2.3 and §2.5 are not used for the proof of Main Theorem 1.

In §3 we highlight the nontriviality of Main Theorem 1 on a toy example in §3.1, then illustrate in §3.2 how to compute a multiplicative lift of it. We then prove the Theorem in §3.3.

In §4 we describe more formally and prove *completeness* our effective construction (based on Hensel lifting). The proof relies on Theorem 12 (in §2.5) and,

also, on a hard result on AG codes: Corollary 30 that we prove in the Appendix, and that states that the componentwise square of lifts of AG codes is *also free*. We provide another example of lift in §4.1, then more optimization techniques and a larger example in Appendix C.

In §5 we prove the aforementionned applications of the theory to MPC. First with a proof of Proposition 20 (the triples sacrifice algorithm over a field/ring of constant size), then with a proof of Main Theorem 3. The proof involves RMFE over rings with same asymptotically constant rate than over fields: we also describe the effective algorithm to construct them in §5.4. In §5.5 we finally provide the efficient robust reconstruction algorithm claimed in Theorem 5 (C). The reasons for this ordering is that *robust* reconstruction is actually not needed in the primitives considered in this paper, and that its proof requires the additional linear algebra of Theorem 25 in Appendix §B.

2 LSSS from *Free* Codes have Optimal Complexity

In §2.1 we introduce Galois ring extensions, and highlight that they the same size and computational overhead over $\mathbb{Z}/p^\ell\mathbb{Z}$, than finite field extensions have over $\mathbb{F}_{p^r}$. The presentation should be self-contained, but the reader can also refer to [Abs+20; Abs+19a]. In §2.2 and §2.3 we consider general LSSS from free codes over rings, and prove the tight complexity claims (A') and (B) of Theorem 5 for sharing and reconstruction. In §2.5 we show useful results on free codes. In §2.5 we prove a useful chain of equivalences for free codes over rings. All the basics are recalled, but the reader can also refer to [Abs+20, §2-§4]

2.1 Optimal complexities in Galois rings extensions $R_\ell(r)$

2.1.1 Equal computational and sizes for elementary operations Let p be a positive prime number and $\mathbb{F}_p := \mathbb{Z}/p\mathbb{Z}$ the finite field. Then, when operating on objects with coordinates in $\mathbb{F}_p$, we say that the *computational complexity* is the number of *elementary operations in* $\mathbb{F}_p$ (where one can possibly weight differently additions, scalar multiplications and bilinear multiplications). Now, $\ell \geq 1$ denoting, an integer, the second context encountered in this paper are objects with coordinates in $\mathbb{Z}/p^\ell\mathbb{Z}$ the ring of integers modulo p^ℓ. In these cases, we say that the computation complexity is the number of *elementary operations in* $\mathbb{Z}/p^\ell\mathbb{Z}$ (where one gives the same weights as before to additions, scalar multiplications and bilinear multiplications). Likewise, the *communication (or size) complexity* is, in the first context: the number of elements in $\mathbb{F}_p$ which are sent by honest players; whereas in the second context it is the number of elements in $\mathbb{Z}/p^\ell\mathbb{Z}$ which are sent.

Galois-rings are defined as follows. Let $r \geq 1$ be a positive integer and $\overline{f(X)} \in \mathbb{F}_p[X]$ a monic irreducible polynomial of degree r. This defines the finite field extension or degree r:

$$\mathbb{F}_{p^r} = \mathbb{F}_p < \delta >:= \mathbb{F}_p[X]/\overline{f(X)}$$

which is a vector space of dimention r over $\mathbb{F}_p$ with basis $1, \delta, \ldots, \delta^{r-1}$ and multiplication rule defined by the multiplication modulo $\overline{f(X)}$. Now, consider any monic polynomial $f(X) \in \mathbb{Z}/p^\ell\mathbb{Z}$ which reduces to $\overline{f(X)}$ modulo p. Then this defines the Galois ring extension of degree r:

$$R_\ell(r) = \mathbb{Z}/p^\ell\mathbb{Z} < \Delta >:= \mathbb{Z}/p^\ell\mathbb{Z}[X]/f(X) \tag{1}$$

which is in particular equal to $\mathbb{Z}/p^\ell\mathbb{Z}$ when $r = 1$. This is a *free module* over $\mathbb{Z}/p^\ell\mathbb{Z}$ of dimention r. That is: it is isomorphic to $(\mathbb{Z}/p^\ell\mathbb{Z})^r$, with basis $1, \Delta, \ldots, \Delta^{r-1}$. Multiplication in $R_\ell(r)$ is defined by the multiplication modulo $f(X)$. Notice that an equivalent definition is to consider the unramified extension of degree r of the ring $\mathbb{Z}_p$ of p-adic integers, which is denoted $W(\mathbb{F}_{p^r})$ the "Witt ring", then reduce it modulo p^ℓ. This will be used in §3.3, and is also a useful point of view for the Hensel lifting algorithm of §4.

We say that an element $x \in R_\ell(r)$ is invertible modulo p if its reduction $\bar{x} \in \mathbb{F}_{p^r}$ is invertible. A key property of Galois rings is that an element invertible modulo p, is then also invertible in $R_\ell(r)$. Indeed, consider an arbitrary lift y of $\overline{x}^{-1}$. Then we have a formula $xy = 1 - p\lambda$ which holds in $R_\ell(r)$ for some λ. But the right hand side of the equation is invertible, of inverse $1 + p\lambda + \cdots + (p\lambda)^{\ell-1}$. From this formula we see that inversion in $R_\ell(r)$ costs essentially one inversion in $\mathbb{F}_{p^r}$, and $O(\log(\ell))$ squarings in $R_\ell(r)$.

2.1.2 Embeddings, and their equal complexities than over fields From the previous, we see that considering $\mathbb{Z}/p^\ell\mathbb{Z}$ as *embedded* in $R_\ell(r)$, multiplies by r the size (an element x is mapped to the vector $(x, 0, \ldots, 0)$ with r coordinates) by the same factor than when embedding $\mathbb{F}_p$ in $\mathbb{F}_{p^r}$. It follows from the definition (1) that the naive schoolboy multiplication algorithm in $R_\ell(r)$ has the same complexity than the one in $\mathbb{F}_{p^r}$. For large Galois rings, we have efficient multiplication algorithms, which are motivated by their usage in LWE. Hence, the references pointed in [Abs+19a, page 4] and [PC20] show that they have also the same complexity than in $\mathbb{F}_{p^r}$.

Finally, one may also need to make the converse operation, and "descend" from secret sharing schemes over $R_\ell(r)$, to secret sharing schemes over $\mathbb{Z}/p^\ell\mathbb{Z}$. The technique to do this over fields is introduced in [Cas+09, Theorem 7 & 8], and based on linear maps called "multiplication friendly embeddings (MFE)": $\mathbb{F}_{p^m} \longrightarrow \mathbb{F}_p^{2m-1}$, which have the property to bring the multiplication in $\mathbb{F}_{p^m}$, into the componentwise product in $\mathbb{F}_p^{2m-1}$. In §A.1 we show in an elementary way that the MFE of [Cas+09, Theorem 8] carry over $\mathbb{Z}/p^\ell\mathbb{Z}$ with the same parameters, and thus we have exactly the same "expansion rates" $((2m-1)/m)$.

2.2 General LSSS & ASSSM over Rings

Let R be any finite ring (including $R = \mathbb{F}_{p^r}$ or $R_\ell(r)$), and n, k be positive integers. To share a secret s in R, one samples uniformly an element $\boldsymbol{w} \in R^{k-1}$ (the randomness space) then applies a certain linear map ψ on the whole to

obtain n "shares" : $\psi(s, \boldsymbol{w}) \in R^n$. For $I \subset \{1, \dots n\}$ a set of indices, we denote $|I|$ the size of I and $\pi_I : R^n \to R^I$ the projection on these components. For any vector $\boldsymbol{x} \in R^n$, we denote for short $\boldsymbol{x}_I := \pi_I(\boldsymbol{x})$ this projection, i.e., the components of $\boldsymbol{x}$ in I, and likewise, for any linear map ψ in R^n, we denote for short $\psi_I := \pi_I \circ \psi$ the "components of ψ in I". Let $0 \leq t < n$ be a positive integer. Let $k, n \geq 1$ be integers, we say that a *linear secret sharing scheme* (LSSS) over R with n shares and randomness space R^{k-1}, is an R-linear map:

$$\begin{aligned} \psi : & R \times R^{k-1} \longrightarrow R^n \\ & (s, \boldsymbol{w}) \longrightarrow \psi(s, \boldsymbol{w}) \end{aligned}$$

We say that it has t-privacy if for any share vector, any t coordinates are independent of the secret, and it has *rec-reconstruction* if any rec coordinates of a vector of shares determine the secret s.

Definition 6. We say that a LSSS with privacy threshold t, is furthermore Arithmetic with Strong multiplication (ASSSM), if for any two secrets $s, s' \in R$, consider any sharings of them: $(s_i) = \psi(s, \boldsymbol{u})$ and $(s'_i) = \psi(s', \boldsymbol{u'})$, then for any set I of indices of size $n-t$, the data of the "sharewise" products $(\psi(s_i)\psi(s'_i))_{i \in I}$ determines uniquely ss'. Said otherwise, I is a "reconstruction set" for $\psi \times \psi$.

Notice that this a fortiori implies $n - t$ reconstruction threshold. If one replaces $n-t$ by n in the definition above, then this is the weaker *Multiplication* property.

2.3 Complexity of Sharing

From now on we specialize to a Galois ring $R := R_\ell(r)$ as defined in (1), e.g., equal to $\mathbb{Z}/p^\ell\mathbb{Z}$ when $r = 1$.

2.3.1 Systematic Form Let $C \subset R_\ell(r)^{n+1}$ be a *free* submodule of rank k, i.e., which is isomorphic to $R_\ell(r)^k$. Making a choice of $n+1$ coordinates in $R_\ell(r)^{n+1}$, we denote this a *"free code"*. Recall that this implies that the reduction $\overline{C}$ of C modulo p is a vector space of *same dimension* k. [This follows immediatly from the fact that if a square matrix with entries in $R_\ell(r)$ is invertible, then its determinant is invertible in $R_\ell(r)$, and thus by 2.1 the reduction of this determinant is invertible. See also [Wal99, Lemma 3.3]]. For the same reason, in the other direction, starting from a code $\overline{C}$ over $\mathbb{F}_{p^r}$ of dimension k, and considering any basis, then arbitrary lifts in $R_\ell(r)$ of these basis vectors generate a free code C of same rank k.

We denote that a matrix $G \in R_\ell(r)^{k \times (n+1)}$ is in *echelon form*, if for each row $i \in \{1, \dots, k\}$, there exists a column $j_i \in \{0, \dots, n\}$ containing a 1 entry on row i and 0 everywhere else. We say in particular that G is in *systematic form* if of the form $(\mathrm{Id}_k | N)$. We say that matrix G' is deduced from matrix G by "elementary row operations", if there exists a sequence of elementary row operations that transforms G into G'. Equivalently, if there exists an invertible matrix $E \in R_\ell(r)^{k \times k}$ such that $G' = EG$. Let us restate for convenience existence the

systematic form of free codes, which is used at least since Calderbank-Sloane [CS95] (see also [SAS17, §5.1.1]). We re-prove it with an explicit construction, which has same complexity than over fields, which thus proves Theorem 5 (A).

Proposition 7. *Let $G \in R_\ell(r)^{k\times(n+1)}$ be a matrix such that the rows are* free. *Then there exists a matrix in echelon form deductible from G by elementary row operations. And thus, up to reordering the $n+1$, coordinates, in systematic form.*

Proof. Consider the reduction of $\overline{G}$ in $\mathbb{F}_{p^r}$. By the Gauss pivot, there exists an invertible $k\times k$ matrix $\overline{E}$ and a matrix $\overline{G'}$ in echelon form, such that $\overline{G'} = \overline{E}\overline{G}$. Let $E \in R_\ell(r)^{k\times k}$ be an arbitrarily lift of $\overline{E}$. E being invertible (since its determinant is invertible modulo p), the matrix $G' := EG$ is deductible from G by elementary row operations. G' being a lift of G, we have furthermore, for each row i, existence of a column j_i such that the entry G_{i,j_i} is a lift of 1, and thus *invertible* in $R_\ell(r)$. Using this entry as a pivot, we anihilate all the other entries on this column j_i by elementary row operations. Finally, we divide the row i by G_{i,j_i}, thus entry (i, j_i) becomes 1. Repeating for all i yields a matrix G'' deduced from G' by elementary row operations.

2.3.2 Sharing Up to permutation of the coordinates, we may now assume $G \in R_\ell(r)^{k\times(n+1)}$ of C in *systematic form*. By Proposition 7, the (one-shot) complexity of computing this form is the same as over fields. Then, sharing a secret $s \in R_\ell(r)$ with respect to the 0-th coordinate of C, boils down to the following. First, sample a vector $\boldsymbol{w} \in R_\ell(r)^{k-1}$, uniformly at random. This has complexity $O(k)$ (or talking in bits: $O(\ell k \log_2(p))$). Then, deduce the vector of shares from the left multiplication:

$$\psi : R_\ell(r) \times R_\ell(r)^{k-1} \longrightarrow R_\ell(r)^n \tag{2}$$

$$(s, \boldsymbol{w}) \longrightarrow (s, \boldsymbol{w})G_{[1,\dots,n]}. \tag{3}$$

Where $G_{[1,\dots,n]}$ denotes the n last columns of G. The complexity claim of Theorem 5(A') then follows from the fact that $\dim(\overline{C}) = \mathrm{rk}(C) = k$, and thus that the generating matrices have the same sizes, combined with the fact pointed in §2.1, that complexity of the multiplication in $R_\ell(r)$ (by definition relatively to elementary operations in $\mathbb{Z}/p^\ell\mathbb{Z}$) is the same as the one in $\mathbb{F}_{p^r}$ (by definition relatively to $\mathbb{F}_p$).

2.4 Privacy and Reconstruction from *Free* Codes

Let us now bound privacy and reconstruction. Notice that, as over fields, t-Privacy holds in particular when for any set $\mathcal{I}_\mathcal{A}$ of t indices, the projection from the subspace of codewords with zero-th coordinate equal to 0: $C \longrightarrow C_{\{0\}} \times C_{\mathcal{I}_\mathcal{A}}$ is surjective. We now state that [CDN15, Lemma 11.71] also hold over rings:

Proposition 8. *Let C be a* free *code in $R_\ell(r)^{n+1}$ of rank k. Denote $\overline{C}$ the code over $\mathbb{F}_{p^r}$ obtained by reduction modulo p, $\overline{C}^\perp$ the dual, and $d(\overline{C})$, $d(\overline{C}^\perp)$ the minimal distances. Consider the LSSS with n shares in $R_\ell(r)$ obtained from C. Recall that rec denotes the reconstruction threshold. Then we have that:*

$$rec \leq n + 1 - (d(\overline{C}) - 1) = n - d(\overline{C}) + 2 \tag{4}$$

$$t \geq d(\overline{C}^\perp) - 2 \tag{5}$$

2.4.1 Reconstruction: Constructive proof of (4), thus of Thm 5 (B) Notice that Equation (4) is proven on a specific case in [Abs+20, Theorem 6]. But it actually holds in general. Let us take the opportunity to make a constructive proof, which will thus support our complexity claim of Thm 5 (B). We keep the notations of Equation (2).

Let $I \subset \{1, \ldots, n\}$ be a subset of $n + 1 - (d(\overline{C}) - 1)$ indices. By definition of the minimal distance, the linear map $\overline{\psi_I} : \mathbb{F}_{p^r} \times \mathbb{F}_{p^r}^{k-1} \longrightarrow \mathbb{F}_{p^r}^I$ is injective. Since it is defined over fields, it thus has a *linear left inverse*. We conclude by applying Lemma 9 to $M := R_\ell(r)^k$, $m := |I|$ and $f := \psi_I$.

Lemma 9. *Let M be a free $R_\ell(r)$-module (say of rank v) and $f : M \to R_\ell(r)^m$ be a R-linear map. Assume that the map modulo p:*

$$\overline{f} : (M \bmod p) = \mathbb{F}_{p^r}^v \longrightarrow \mathbb{F}_{p^r}^m$$

is an injection. Then f has a linear *left inverse $g : R_\ell(r)^m \longrightarrow M$. In particular, the image of f is a* free *$R_\ell(r)$-module.*

Proof. The matrix mat_f of f, of size $m \times v$ is such that, by assumption, when we reduce it modulo p, then it contains a $v \times v$ invertible minor. But then this minor in mat_f is also invertible (recall that $(1 + \lambda p)^{-1} = 1 + \sum_i \lambda^i p^i$). Inverting this minor (e.g. with Gauss pivot over $R_\ell(r)$, for efficiency), and completing with $m - v$ zero columns, yields a map $g : R_\ell(r)^m \longrightarrow R_\ell(r)^v$ such that $g \circ f = \mathrm{Id}_v$.

The last claim follow from the fact that f is in particular injective, so defines an isomorphism between M, which is free, and its image inside $R_\ell(r)^m$.

Our claim about the computational complexity then follows as previously from *linearity* of the reconstruction map ψ_I, and the fact (§2.1) that multiplications in $R_\ell(r)$ has same complexity than in $\mathbb{F}_{p^r}$.

2.4.2 Warning: Loss of Efficient Reconstruction for Non-Free Codes Recall that reconstructibility of a code means that, for any set of $d - 1$ coordinates, the map consisting in puncturing these coordinates is an injection; and that *efficient* reconstruction means that it has a *linear left-inverse*, which we denote a *retraction*, as known as the *reconstruction map*. In the following Counterexample 10 we show that, without the assumption to be free, there exists

submodules of $R_\ell(r)^{n+1}$ for which the puncturing map is an injection, but for which *there does not exists any linear retraction*. This motivates why we restrict to *free* codes in $R_\ell(r)^{n+1}$ in order to construct LSSS.

Counterexample 10. *Let C be a code in $R_\ell(r)^{n+1}$ with $\overline{d} := d(\overline{C}) \geq 2$ such that there exists a punctured $C^* \subset R_\ell(r)^{n+1-(\overline{d}-1)}$ which is* not free. *[For example*

$$C = \langle (p, p, p, 0), (1, 0, 0, 1) \rangle \in R_\ell(r)^4, \quad (\text{e.g. } R_\ell(r) := \mathbb{Z}/p^\ell\mathbb{Z} \ )$$

with $d(\overline{C}) = 2$ and injectivity in $R_\ell(r)^3$ when puncturing the last coordinate.] Then there does not exist *any linear reconstruction map, i.e., any retraction $R_\ell(r)^{n+1-(\overline{d}-1)} \longrightarrow C$.*

The proof is that, supposing such a retraction, then, composing it on the left with the puncturing map, yields a left-inverse to the inclusion $C^* \subset R_\ell(r)^{n+1-(\overline{d}-1)}$. Denoting $G^* \in R_\ell(r)^{k \times (n+1-(\overline{d}-1))}$ a generating matrix of C^* (in rows) and $L \in R_\ell(r)^{(n+1-(\overline{d}-1)) \times k}$ the matrix of this left-inverse, we would thus have by assumption $G^* L = \mathrm{Id}_k$. In particular $\overline{G^*}$ modulo p would be of maximal dimension, k, thus G^* would generate a *free* module, a contradiction.

2.4.3 Proof of (5) of Proposition 8 This bound is proven in [Abs+20, Theorem 6]: although on a specific LSSS, the arguments apply in general. To be sure, this is the same bound as the one known over finite fields ([CDN15, Thm 11.77]). The key to prove this formula over rings is the [Abs+20, Lemma 3], that we recall here. We take the opportunity to provide a shorter and self-contained proof.

Lemma 11. *Let C be a submodule of $R_\ell(r)^n$, denote $\overline{d}^\perp$ the dual distance of the reduction $\overline{C}$ modulo p. Let $\mathcal{I}$ be a set of indices with $|\mathcal{I}| = \overline{d}^\perp - 1$. Then, projection of C on the indices in $\mathcal{I}$ is the full space $R_\ell(r)^{\mathcal{I}}$.*

Proof. By assumption, $\overline{C}_{\mathcal{I}} = \mathbb{F}_q^{\mathcal{I}}$. Hence $C_{\mathcal{I}}$ contains $|\mathcal{I}|$ vectors so that the matrix formed by them has an invertible determinant, thus has an inverse, thus these vectors generate $R_\ell(r)^{\mathcal{I}}$.

2.5 (Free) Generation from Any Lift of *Any* Basis

We did not find the following important fact in the literature:

Theorem 12. *Let C be a* free *code in $R_\ell(r)^{n+1}$. Consider the reduced code modulo p: $\overline{C} \subset \mathbb{F}_{p^r}$, and* any *basis $(\overline{e'}_i)$ of $\overline{C}$. Then C is freely* generated *by* any *lift of $(\overline{e'}_i)$ inside C.*

Proof. Let k denote the rank of C. The "freely" claim again follows from the fact that a family whose reduction modulo p is free, is itself free (the generating

matrix containing an invertible $k \times k$ determinant). Now, consider $(\overline{e'}_{i=1\ldots k})$ any lift of the $(\overline{e'}_i)$ inside C. It generates a submodule in C, which is free of rank k by the first part of the proof. But C is itself a free module of rank k. Thus this defines an injection $R_\ell(r)^k \hookrightarrow R_\ell(r)^k$, which is by assumption also an injection modulo k. Thus by Lemma 9 it has a left inverse, thus it is a bijection.

Corollary 13. *If $E \subset G$ are two lifts R^n of the same code $\overline{G}$, and G is free, then they are equal (in particular E is also free).*

Proof. Indeed E contains a lift of a basis of $\overline{G}$ which, by Theorem 12, generate the whole G. □

3 Main Theorem 1

3.1 An Arbitrary Free Lift of a Code of *Small Square* mostly *Fails* to Have a Small Square

For C a code (over a field or a ring), we denote as *componentwise square* C^{*2} the code of same length which is generated by all the products of any two codewords of C component by component. Strong multiplication of the LSSS from C thus requires that C^{*2} has large distance, thus be of small size. Let us revisit the family of [Abs+20, Example 2], and explain why they provide also counterexamples where arbitrarily lifting *fails* to yield a square code of small size.

Counterexample 14. *Let $\bar{C}$ and $\bar{D}$ be codes over $\mathbb{F}_{p^r}$ of same dimension and let us assume that $\dim \bar{D}^{*2} < \dim \bar{C}^{*2}$. Let us now build a code E over $R_\ell(r)$ with $\ell \geq 3$ and of length equal to the sum of the lengths of $\bar{C}$ and $\bar{D}$. Let $(\bar{c}_i)_i$ and $(\bar{d}_i)_i$ be bases of $\bar{C}$ and $\bar{D}$, let $(c_i)_i$ and $(d_i)_i$ be arbitrary lifts and define E the code generated by the vectors $(d_i, pc_i)_i$. Then E is free, because of dimension $\dim \bar{D} = \dim \bar{E}$, and is a lift of $\bar{E}$. Suppose by contradiction that the square E^{*2} would be free, then we would have:*

$$\dim E^{*2} \geq \dim \bar{C}^{*2} > \dim \bar{D}^{*2} = \dim \bar{E}^{*2} .$$

On the other hand if it was free, then it would be of same rank than $\bar{E}^{*2}$ by Theorem 12. So we have a contradiction. Thus E^{*2} is not free, thus it is strictly larger than some free lift of $\bar{E}^{*2}$ inside him.

3.1.1 The desirable case of small square: sparsity of solutions, if any, illustrated on a toy example

Let us now illustrate hardness of the multiplicative lifting problem on a tiny AG code. Consider the elliptic curve $y^2 + xy + y - x^3 + 1$ over

$$\mathbb{F}_{2^3} = \mathbb{F}_2 < \delta > \text{ with polynomial } \delta^3 + \delta + 1 = 0,$$

with 14 places, P_0 the place at infinity, the divisor $D_0 = 4P_0$ and the Riemann-Roch space $L(4P_0)$, with basis $\overline{\boldsymbol{e_i}}_{i=1\ldots 4}$ equal to the functions $(1, x, x^2, y)$. Let us

define the evaluation code $C(D_0)$ at the $P_1, \ldots, P_{13}$, (not at P_0, for simplicity). We compute the following generating matrix:

$$\overline{G} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \delta & \delta & \delta^2 & \delta^2 & \delta^3 & \delta^3 & \delta^4 & \delta^4 & \delta^5 & \delta^5 & \delta^6 & \delta^6 & 1 \\ \delta^2 & \delta^2 & \delta^4 & \delta^4 & \delta^6 & \delta^6 & \delta & \delta & \delta^3 & \delta^3 & \delta^5 & \delta^5 & 1 \\ 1 & \delta & 1 & \delta^2 & \delta^2 & \delta^4 & 1 & \delta^4 & \delta & \delta^2 & \delta & \delta^4 & 0 \end{bmatrix}$$

Let us consider the 10 componentwise products $\overline{\boldsymbol{e_i}} * \overline{\boldsymbol{e_j}}$, with indices (i,j) ordered as: $(1,1),(1,2),(1,3),(1,4),(2,2),(2,3)$ etc. (i.e.: j increases first). They generate by definition $C(D_0)^{*2}$. We verify that, removing $(2,2)$ and $(4,4)$ from the indices in this list, then the remaining 8 products: $\overline{\mathcal{B}} := (\overline{\boldsymbol{e_k}} * \overline{\boldsymbol{e_l}})_{(k,l)\in B}$ generate $C(D_0)^{*2}$, where B denotes the remaining indices ordered as before. In particular $\overline{\boldsymbol{e_2}} * \overline{\boldsymbol{e_2}}$ and $\overline{\boldsymbol{e_4}} * \overline{\boldsymbol{e_4}}$ decompose themselves on this basis $\overline{\mathcal{B}}$, with decomposition coefficients $(\overline{\lambda_{2,2,k,l}})_{k,l\in B}$ and $(\overline{\lambda_{4,4,k,l}})_{k,l\in B}$ given by the following 2×8 matrix, called "Reduc" in the implementation:

$$\left(transp\left(\overline{\lambda_{2,2,k,l}}, \overline{\lambda_{4,4,k,l}}\right)\right)_{(k,l)\in B} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \quad (6)$$

Then we repeated the following experiment 10^8 times: *randomly* lift the $(\overline{\boldsymbol{e_i}})_i$ modulo 2^2, to obtain vectors $(\boldsymbol{e_i})_i$ with coordinates in $R_2(3) = \mathbb{Z}/2^2\mathbb{Z} < \Delta >$. Let C_{bad} the code generated by these lifts. By Theorem 12, it is always *free*. But we observed in *all the experiments* that $\boldsymbol{e_2} * \boldsymbol{e_2}$ and $\boldsymbol{e_4} * \boldsymbol{e_4}$ *do not* anymore decompose themselves on the lifts of the previous basis of $C(D_0)^{*2}$: $\mathcal{B} := (\boldsymbol{e_k} * \boldsymbol{e_l})_{(k,l)\in B}$ —see two paragraphs later for an explanation of how this checks were done efficiently with linear algebra. So in these situations C_{bad}^{*2} *is not* a free lift of the square $C(D_0)^{*2}$, because if it were, then by Theorem 12 the lifted basis $\mathcal{B}$ *would* generate it.

3.1.2 Why solutions may likely not exist at all Let us give a feeling of why most codes with small squares are likely to have no multiplication friendly lift. We will come over these arguments in more details in §4.

Let $\overline{C}$ be a code over, say, $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ of dimension k and length n, such that the square $\overline{C}^2$ has *small* dimension, say, $3k < n$. The goal is to find a multiplication friendly lift. That is, a code C over $\mathbb{Z}/p^2\mathbb{Z}$ (namely: a free submodule of $(\mathbb{Z}/p^2\mathbb{Z})^n$) of same rank k, that lifts $\overline{C}$ modulo p^2, and such that the square C^2 is also a free lift of $\overline{C}^2$. As argued with the toy example, it follows from Theorem 12 that these requirements are equivalent to the following: let $(\boldsymbol{e_i})_i$ be any basis of C lifting a basis $(\overline{\boldsymbol{e_i}})$ of $\overline{C}$; let $\overline{\mathcal{B}}$ be *any* basis of $\overline{C^2}$; then $\overline{\mathcal{B}}$ lifts modulo p^2 to a basis of the square C^2, in particular generates the componentwise products $(\boldsymbol{e_i} * \boldsymbol{e_j})_{i,j}$. To fix ideas let us choose a basis of the form $\overline{\mathcal{B}} = (\overline{\boldsymbol{e_k}} * \overline{\boldsymbol{e_l}})_{(k,l)\in B}$ as in the toy example. Then the previous equivalent condition translates itself into the fact that the equations expressing $\overline{\boldsymbol{e_i}} * \overline{\boldsymbol{e_j}}$ on this basis:

$$\overline{\boldsymbol{e_i}} * \overline{\boldsymbol{e_j}} = \sum_{(k,l)\in B} \overline{\lambda_{i,j,k,l}}\, \overline{\boldsymbol{e_k}} * \overline{\boldsymbol{e_l}} \pmod{p} \quad (7)$$

lift modulo p^2. The number of degrees of freedom (the unknowns) are: (i) the choices of lifts for the $\overline{\boldsymbol{e_i}}$, so a total of nk coordinates to lift in $\mathbb{Z}/p^2\mathbb{Z}$; (ii) and lifts for the coefficients $\overline{\lambda_{i,j,k,l}}$: a total of $3k \times k(k+1)/2$ unknowns in $\mathbb{Z}/p^2\mathbb{Z}$. So the number of unknowns is asymptotically equivalent to (ii): $3k\times k(k+1)/2$. Whereas the number of equations is $nk(k+1)/2$ (namely: $k(k+1)/2$ vectorial equations with n coordinates in $\mathbb{Z}/p^2\mathbb{Z}$ each). Notice that $3k < n$, so that there are more constraints than variables. Finally, as will be detailed in the next paragraph, and then further in the proof of Proposition 18 notice that this *quadratic* system over a *ring* simplifies modulo p^2 to a *linear system* over the *field* $\mathbb{F}_p$. Thus, the system being *overdetermined*, then *a priori no solution is likely to exist.*

3.2 A technique to find them when they exist, illustrated on the toy example

We will formalize the general technique in §4 and justify that it returns all solutions when they exist. As a matter of fact, as will be demonstrated in general in Theorem 28, it is a remarkable property of AG codes, in particular the toy example, that they always have multiplication-friendly lifts —at least for all practical parameters. Putting this in perspective with the previous paragraph, this illustrates that AG codes seem *highly non-generic* among those with small square.

First, fix a free lift C_{bad} of $C(D_0)$ by lifting arbitrarily the basis to $(\boldsymbol{e_i}')_i$, for example by lifting the coordinates from $\mathbb{F}_2 < \delta >$ to $\mathbb{Z}/2^2\mathbb{Z} < \Delta >$ by the dummy rule: $1 \to 1$ and $\delta \to \Delta$. This gives formally the same generating matrix as $\overline{G}$, with δ replaced by Δ. With the same dummy rule, lift the decomposition coefficients $(\overline{\lambda_{2,2,k,l}})_{(k,l)\in B}$ and $(\overline{\lambda_{4,4,k,l}})_{(k,l)\in B}$ to $\lambda'_{2,2,k,l}$ and $\lambda'_{4,4,k,l}$, so that their matrix is formally the same as in (6). Now, there is *no reason* why $\boldsymbol{e'_2} * \boldsymbol{e'_2}$ and $\boldsymbol{e'_4} * \boldsymbol{e'_4}$ should decompose themselves on $\mathcal{B} := (\boldsymbol{e_k}' * \boldsymbol{e_l}')_{(k,l)\in B}$, let alone with coefficients equal to $\lambda'_{2,2,k,l}$ and $\lambda'_{4,4,k,l}$, as we illustrated with our random tests two paragraphs above. As a matter of fact, we encounter nonzero error vectors $2\boldsymbol{D_{2,2}}$ and $2\boldsymbol{D_{4,4}}$ when trying to write the decompositions in $\mathbb{Z}/2^2\mathbb{Z} < \Delta >$:

$$\boldsymbol{e'_2} * \boldsymbol{e'_2} = \sum_{(k,l)\in B} \lambda'_{2,2,k,l} \boldsymbol{e_k}' * \boldsymbol{e_l}' + 2\boldsymbol{D_{2,2}} \text{ and likewise for } \boldsymbol{e'_4} * \boldsymbol{e'_4} \tag{8}$$

Let us insist on the remarkable fact that the error vectors are multiples of 2, since the equalities (8) do hold without error term modulo 2. "Dividing" by 2, their coefficients are

$$transp(\boldsymbol{D_{2,2}}, \boldsymbol{D_{4,4}}) = \begin{bmatrix} 0 & 0 & \delta^4 & \delta^4 & \delta & \delta & 1 & 1 & \delta^5 & \delta^5 & \delta & \delta & 0 \\ 0 & 0 & \delta & \delta & 0 & 1 & \delta^2 & \delta^2 & \delta^4 & 0 & 0 & \delta^5 & 1 \end{bmatrix}$$

Which we express in $\mathbb{F}_{2^3}$ by abuse of notation (remember that an element $2x \in \mathbb{Z}/2^2\mathbb{Z} < \Delta >$ is determined by the residue $\overline{x} \in \mathbb{F}_{2^3} \mod 2$). Now, let us look for corrective terms $2\boldsymbol{f'_i}$ and $2\mu'_{i,j,k,l}$, which we need only to find modulo 2:

$$\boldsymbol{e_i} = \boldsymbol{e'_i} + 2\boldsymbol{f'_i} \text{ and } \lambda_{i,j,k,l} = \lambda'_{i,j,k,l} + 2\mu'_{i,j,k,l} \tag{9}$$

So that, replacing $\boldsymbol{e_i}'$ in (8) by the corrected $\boldsymbol{e_i}$ of (9) —where the corrective terms are treated as unknows—, simplifying and removing the terms that are multiples of 2^2 —because they vanish in $\mathbb{Z}/2^2\mathbb{Z} < \Delta >$—, we observe that all the terms remaining in the system are multiples of 2. So "dividing" the system by 2, we fall back to a *linear* system in $\mathbb{F}_{2^3}$:

(10)

$$\overline{\boldsymbol{e_2}}*\boldsymbol{f_2'}+\overline{\boldsymbol{e_2}}*\boldsymbol{f_2'}-\boldsymbol{D_{2,2}} = \sum_{(k,l)\in B} \mu'_{2,2,k,l}\overline{\boldsymbol{e_k}}*\overline{\boldsymbol{e_l}}+\overline{\lambda_{2,2,k,l}}(\overline{\boldsymbol{e_k}}*\boldsymbol{f_l'}+\overline{\boldsymbol{e_l}}*\boldsymbol{f_k'}) \quad (\text{same for } \overline{\boldsymbol{e_4}}*\overline{\boldsymbol{e_4}})$$

as could be expected from Hensel's Lemma. Solving this system for the corrective terms, we deduce the corrected basis $(\boldsymbol{e_i})_i$ defined as in (9), that define the corrected code C_{good}, whose coordinates are given in the big left-hand rotated matrix on the first formula page of the Appendix.

Likewise we deduce the corrected decomposition coefficients $(\lambda_{2,2,k,l})_{k,l\in B}$ and $(\lambda_{4,4,k,l})_{k,l\in B}$ as given in the centered right-hand formula.

We can finally check straightforwardly that, with these corrected values, then $\boldsymbol{e_2}*\boldsymbol{e_2}$ and $\boldsymbol{e_4}*\boldsymbol{e_4}$ now decompose themselves on $\mathcal{B}$ with the corrected coefficients, without anymore parasitic error vectors. So with these corrected lifts $(\boldsymbol{e_i})_i$, we have now that the square of the corrected code C_{good} is also a free lift. That is, we have succeeded in modifying the free lift C_{bad} into a *multiplication-friendly* lift C_{good}.

3.3 Proof of Main Theorem 1

3.3.1 Roadmap of the proof.

First Consider a smooth curve over $\mathbb{F}_{p^r}$ and a divisor D_0 on this curve (that is: a set of points with multiplicities), such that the degree (the sum of the multiplicities) is $\deg(D_0) < n$. Then, the curve has a lift defined over the ring $R_\ell(r)$ (provided it is given under an equivalent form where equations have no singular points). Lifting the points then applying Judy Walker's results, we have the existence of lifts of the Riemann-Roch spaces: $L(D)$ and $L(2D)$ which are *free* modules, and such that we have inclusions of products of spaces of global sections

$$L(D)^{\otimes 2} = L(2D)\,, \tag{11}$$

where the traditional notation $L(D)^{\otimes 2}$ stands for the space generated by all products fg of pairs of sections (f, g) in $L(D)$. Then, from Judy Walker's Theorem 15 below, we deduce that the evaluation codes over rings $C(D)$ and $C(2D)$, arising from evaluation of these free lifts of Riemann-Roch spaces, are also *free*. We will detail this material in the two next subsections

Next The key property of these free lifts is that they behave well with respect to inclusions and squares:

$$C(D)^{*2} \subset C(2D)\,. \tag{12}$$

Here the code $C(2D)$ is free for the same reasons, with same rank as the classical AG code $\overline{C}(2D_0)$ below modulo p. So this forces the square $C(D)^{*2}$ to stay small, contrary to the square of an arbitrary free lift, which may "spread out" too much (as seen in Counterexample 14).

Deducing the parameters By freeness of $C(D)$, Proposition 8 (5) implies that a LSSS from $C(D)$ has privacy threshold at least as large as a LSSS from the code below modulo p: $\overline{C}(D_0)$.

Likewise, by Proposition 8 (4), a LSSS from the *free* code $C(2D)$ has *full reconstruction* from any $n - d(\overline{C}(2D_0)) + 2$ shares. Thus, by inclusion (12), so does a LSSS from the subcode $C(D)^{*2}$. Said otherwise, a LSSS from $C(D)$ has *reconstruction of the product*, from a number of pairwise products of shares which is *as small* as for a LSSS from $\overline{C}(D_0)$.

For sake of completeness we review the concrete parameters of these schemes in the Appendix §A.2, examplified on the ones of [CC06].

3.3.2 Lift of curves, divisors and Riemann-Roch spaces Let us follow Walker's [Wal99] notations. Note $R = R_\ell(r)$ the (Artinian local) Galois ring, with residue ring $R/(p) = \mathbb{F}_{p^r}$. X_0 being a smooth projective curve over $\mathbb{F}_{p^r}$, then from [Ill05, Theorem 5.19 ii)] (or [SGA1, III Corollaire 7.4]), X_0 has a smooth projective lift over the ring of Witt vectors $W(\mathbb{F}_{p^r})$. Which, after reduction mod p^ℓ, yields a projective lift X over R (because these properties are preserved by base change). Also, R being local, $\mathbb{F}_{p^r}$-points of X_0 lift to R-points of X by the formal smoothness criterion (see [Wal99, Remark 4.5] or next paragraph for details). As a consequence, divisors with support on rational points (actually any divisor) lift to X —and thus also do the line bundles $\mathcal{L}_0$ arising from them.

An explicit procedure for simultaneous compatible free lifts of line bundles. By [Wal99, Lemma 4.4] we can construct lifts of divisors D_0 on X from the following recipe. First, for every rational point $P_0^{(j)}$ of X_0, fix a closed point of degree one $P^{(j)}$ of X above P_0, as described in [Wal99, Remark 4.5] (lift arbitrarily $P_0^{(j)}$ to an R-point, then choose $P^{(j)}$ inside the image).

Then we can simultaneously lift divisors D_0 and $2D_0$ on X_0 as follows. For every rational point P_0 of X_0 in the support of the line bundle D_0, let m be the valuation of D_0 at P_0 and let P be the closed point lying above P_0 as fixed earlier. Deduce from it a divisor mP, then sum over the points P_0 in the support of D_0, to obtain a lift D of D_0. Likewise for the divisor $2D$, equal to the same formal sum of R-points as in D and with twice the multiplicities. In particular, note $\mathcal{L} := \mathcal{L}(D)$ the line bundle associated to D, and likewise for $\mathcal{L}(2D)$.

Proof of (11) This formula well known for curves over fields. Let us justify that it also holds over rings. The reason is that, by smoothness of the lift of the curve, this guarantees that, in a small enough neighborhood U of P, we also have a *uniformizer* denoted t_U (see [Wal, Proposition 4.9]). Thus, as long as U does not

contain the other points of the support of D, we have:

$$\mathcal{L}_U = t_U^{-m}\mathcal{O}_U \ . \tag{13}$$

Thus $t_U^{-m}t_U^{-m} \in \mathcal{L}_U(2D)$, hence the claimed inclusion of products of global sections (11).

3.3.3 Deducing AG codes by Evaluation of Global Sections. For any divisor D on X, we denote as the "Riemann-Roch space" $\Gamma(X,\mathcal{L})$ the space of global sections. In the rest of the paper it is denoted instead $L(D)$. By the argument above [Wal99, Theorem 4.7], f is a *free* R-module that reduces modulo p to $\Gamma(X_0,\mathcal{L}_0)$. With slightly narrower conditions on the degree, then have the following compatibilities, as wrapped-up in [Wal99, Theorem 5.5]:

Theorem 15 (Lifts of Riemann-Roch spaces and AG codes). *Consider n rational points $\mathcal{P}_0 = \left(P_0^{(j)}\right)_{j=1\ldots n}$ on X_0, D_0 a divisor of degree:*

$$2g-2 < \deg D_0 < n$$

with associated line bundle $\mathcal{L}_0$, and the injective evaluation map γ_0 yielding an algebraic geometry code $\overline{C}$ in $\mathbb{F}_{p^r}^n$. Then this data lifts to objects over R: X,$\mathcal{P}$ and D, with associated line bundle $\mathcal{L}$, yielding an evaluation code C, such that we have the following commutative diagram:

$$\begin{array}{ccccc}
\Gamma(X,\mathcal{L}) & \twoheadrightarrow & \Gamma(X,\mathcal{L})\otimes_R \mathbb{F}_{p^r} & \xrightarrow{\sim} & \Gamma(X_0,\mathcal{L}_0) \\
\downarrow{\scriptstyle eval} & & & & \downarrow{\scriptstyle eval} \\
\oplus_j \Gamma(P^{(j)},\mathcal{L}|_{P^{(j)}}) & & & & \oplus_j \Gamma(P_0^{(j)},\mathcal{L}_0|_{P_0^{(j)}}) \\
\cong\downarrow{\scriptstyle \gamma} & & & & \cong\downarrow{\scriptstyle \gamma_0} \\
R^n & & \xrightarrow{\bullet\otimes_R\mathbb{F}_{p^r}} & & \mathbb{F}_{p^r}^n
\end{array} \tag{14}$$

Where: - the top left horizontal arrow and the bottom horizontal arrow are tensorisation by $\otimes_R\mathbb{F}_{p^r}$ - the top right isomorphism is constructed canonically as in the proofs of [Wal99, Lemma 4.6 & proof of Th 4.7]

- the top vertical arrows are the canonical restriction maps - the bottom left vertical arrow is a collection of arbitrary isomorphisms for all j:

$$\gamma_j : \Gamma(P^{(j)},\mathcal{L}|_{P^{(j)}}) \longrightarrow A$$

that reduce to γ_0 by tensorisation by $\otimes_R\mathbb{F}_{p^r}$ (and if not, then redefine γ_0 accordingly without changing the code in $\mathbb{F}_{p^r}^n$).

Notice that the name "evaluation maps" of the top vertical arrows is abusive in general (because of poles, etc: see the first example of §4.1), but they do play this role..

In conclusion, as explained in [Wal99], the code $C(D)$ (likewise $C(2D)$) is free because it is the image of a free module: $\Gamma(X, \mathcal{L})$, under the evaluation map which is an injection modulo (p), and thus its image is a *free* submodule of $R_\ell(r)^n$ by Lemma 9.

4 Proof of Theorem 4 and another example

The main result of this section, formalized in Proposition 18 and described in its proof, is that we can effectively deduce a multiplication-friendly lift $C_{\ell+1} \in R_{\ell+1}(r)^n$, *whenever it exists*, from a multiplication-friendly lift $C_\ell \in R_\ell(r)^n$, from the Hensel lifting technique. Namely, by solving a *linear* system over $\mathbb{F}_{p^r}$ of size $O(n^3) \times O(n^3)$, in particular in *polynomial time*. All the ideas and arguments have been actually given in §3.1, especially in the discussion around formula (7). Another numerical example is described in §4.1. Then in the Appendix §C.1 we provide optimization techniques [5]. , then a larger example in §C.2.

The key heuristic (unexplained) observation is that, for AG codes $\overline{C}$ satisfying the criterion of Theorem 28 of the Appendix, then we have the following apparent *stronger property* than Theorem 28: *any* multiplicative lift C_ℓ of $\overline{C}$ over a given $R_\ell(r)$, has itself a multiplicative lift over $R_{\ell+1}(r)$ (whereas what is proven in Main Theorem 1 / Corollary 30 is only existence of such a multiplicative lift for the *base code* $\overline{C}$). Thus in practice we *can lift* $\overline{C}$ *sequentially in* L *steps into* $R_L(r)^n$, *each of these steps involving only one linear system (of same size in each step)*, so that the overall complexity is *linear in* L.

Lemma 16. *Let* C *be a free code in* $R_\ell(r)^n$. *Then* C^{*2} *is also free —i.e.* C *is a multiplication-friendly lift— if and only if there exists a basis* $(\boldsymbol{e_i})_i$ *of* C, *and a set* B *of unordered couples of indices* (k, l) *of cardinality* $\dim C^{*2}$, *such that the elementary products* $(\boldsymbol{e_k} * \boldsymbol{e_l})_{(k,l)\in B}$ *form a basis of* C^{*2}. *Namely, if and only if there exists coefficients* $\lambda_{i,j,k,l}$ *in* $R_\ell(r)$ *such that the following equalities in* $R_\ell(r)^n$ *hold:*

$$\boldsymbol{e_i} * \boldsymbol{e_j} = \sum_{k,l} \lambda_{i,j,k,l}\, \boldsymbol{e_k} * \boldsymbol{e_l} \quad \text{for all } i \leq j \tag{15}$$

Although all the arguments were given in §3.1, let us give a formal proof of Lemma 16:

Proof. Let $(\overline{\boldsymbol{e_i}})_i$ be *any* basis of $\overline{C}$. Let us chose $(\boldsymbol{e_i})_i$ *any* lifts of the $(\overline{\boldsymbol{e_i}})_i$ in C. By Theorem 25 (ii), C being a free lift, they form basis of C. The family $(\overline{\boldsymbol{e_i}} * \overline{\boldsymbol{e_j}})_{(i,j)}$ is a generating set of $\overline{C}^{*2}$. Extract from it a basis $(\overline{\boldsymbol{e_k}} * \overline{\boldsymbol{e_l}})_{(k,l)\in B}$ of $\overline{C}^{*2}$. The family $(\boldsymbol{e_k} * \boldsymbol{e_l})_{(k,l)\in B}$ is contained in C^{*2} and is a lift of a basis of $\overline{C}^{*2}$. But C^{*2} is by asumption free, thus Theorem 25 (ii) implies that $(\boldsymbol{e_k} * \boldsymbol{e_l})_{(k,l)\in B}$ is a basis of C^{*2}.

[5] A reviewer also makes the powerful observation that Hensel lifting enables to double the precision at every step, thus that our computations could be much sped up.

As for the explicit equivalent conditions (15), let us simply call $\lambda_{i,j,k,l}$ the coefficients of the decomposition of a given $(\boldsymbol{e_i} * \boldsymbol{e_j})_{i,j}$ over the basis $(\boldsymbol{e_k} * \boldsymbol{e_l})_{(k,l)\in B}$. □

Lemma 17. *Let C_ℓ be a multiplication-friendly lift $R_\ell(r)^n$. Suppose that there exists a multiplication-friendly lift $C_{\ell+1}$ of C_ℓ in $R_{\ell+1}(r)^n$ —i.e. such that the square $C_{\ell+1}^{*2}$ is also a free lift of C_ℓ^{*2}. Consider a basis $(\boldsymbol{e_i})_i$ of C_ℓ^{*2}, along with $(\boldsymbol{e_k} * \boldsymbol{e_l})_{(k,l)\in B}$ a basis of C_ℓ^{*2} and the explicit decomposition* (15)*, as granted by Lemma 16.*

Then $C_{\ell+1}$ has a basis $(\widetilde{\boldsymbol{e_i}})_i$ formed of lifts *of the $\boldsymbol{e_i}$, such that there exists* lifts $\widetilde{\lambda_{i,j,k,l}}$ *of the coefficients $\lambda_{i,j,k,l}$, making all the equalities* (15) lift *over $R_{\ell+1}(r)^n$:*

$$\widetilde{\boldsymbol{e_i}} * \widetilde{\boldsymbol{e_j}} = \sum_{k,l} \widetilde{\lambda_{i,j,k,l}}\, \widetilde{\boldsymbol{e_k}} * \widetilde{\boldsymbol{e_l}} \quad \text{for all } i \leq j \tag{16}$$

Proposition 18. *Let C_ℓ be a free lift in $R_\ell(r)^n$ such that the square C_ℓ^{*2} is also a free lift. Let us consider any basis $(\boldsymbol{e_i})_i$ of C_ℓ, along with the explicit decomposition* (15) *granted by Lemma 16. Then finding —if any— a multiplication-friendly lift $C_{\ell+1}$ of C_ℓ in $R_{\ell+1}(r)^n$, falls back to solving* a linear *of size $O(n^3) \times O(n^3)$.*

More precisely, Lemma 17 states that multiplication friendly lifts $C_{\ell+1}$ are in one-to-one bijection with lifts in $R_{\ell+1}(r)^n$ of the explicit decomposition (15). Practically, the system returns all possible lifts of this decomposition, from which we deduce *all possible* multiplication friendly lifts $C_{\ell+1}$.

Proof. To simplify the notations let us make the proof for $\ell = 1$, with thus $C_1 = \overline{C}$, so we are looking for a lift $C := C_2$ in $R_2(r)^n$ such that C^{*2} is a free lift (of $\overline{C}^{*2}$). The technique for higher levels ℓ proceeds identically. By assumption we are given $(\overline{\boldsymbol{e_i}})_i$ a basis of $\overline{C}$, and a basis $(\overline{\boldsymbol{e_k}} * \overline{\boldsymbol{e_l}})_{(k,l)\in B}$ of $\overline{C}^{*2}$ along with the following (quadratic) equalities

$$\overline{\boldsymbol{e_i}} * \overline{\boldsymbol{e_j}} = \sum_{k,l} \overline{\lambda_{i,j,k,l}}\, \overline{\boldsymbol{e_k}} * \overline{\boldsymbol{e_l}} \quad \text{for all } (i,j) \tag{17}$$

in $\mathbb{F}_{p^r}$. Thanks to Lemma 17, the problem of finding C boils down to finding lifts $\boldsymbol{e_i}$ in $R_2(r)^n$ and $\lambda_{i,j,k,l}$ in $R_2(r)$, such that the equations (17) all lift to $R_2(r)^n$. Fix arbitrary lifts $\boldsymbol{e_i}'$ and $\lambda'_{i,j,k,l}$ of the $\overline{\boldsymbol{e_i}}$ and the $\overline{\lambda_{i,j,k,l}}$, over/in $R_2(r)$. We obtain error terms $pD_{i,j}$ when writing the system in $R_2(r)$:

$$\boldsymbol{e_i}' * \boldsymbol{e_j}' = \sum_{k,l} \lambda'_{i,j,k,l} \boldsymbol{e_k}' * \boldsymbol{e_l}' + pD_{i,j} \text{ for all } (i,j) \tag{18}$$

Solving the system means finding correct lifts $\boldsymbol{e_i}''$ and $\lambda''_{i,j,k,l}$ that anihilate the error terms $pD_{i,j}$. But $e''i$ and $\lambda''_{i,j,k,l}$ can always be deduced from $\boldsymbol{e_i}'$ and

$\lambda'_{i,j,k,l}$, by adding corrective terms $p\boldsymbol{f_i'}$ and $p\mu'_{i,j,k,l}$, which we need only to find modulo p:

$$\boldsymbol{e_i}'' = e'i + p\boldsymbol{f_i'} \text{ and } \lambda''_{i,j,k,l} = \lambda'_{i,j,k,l} + p\mu'_{i,j,k,l} \tag{19}$$

So that, replacing $\boldsymbol{e_i}'$ in (18) by the corrected $\boldsymbol{e_i}$" of (19) (where the corrective terms are treated as unknows), simplifying and moding out the terms that are multiples of p^2, we observe (Hensel's trick) that all the terms remaining in the system are multiplies of p. So simplifying by p, we fall back to the following *linear* system in $\mathbb{F}_{p^r}$:

$$\overline{\boldsymbol{e_i}} * \boldsymbol{f_j'} + \overline{\boldsymbol{e_j}} * \boldsymbol{f_i'} - D_{i,j} = \sum_{k,l} \mu'_{i,j,k,l} \overline{\boldsymbol{e_k}} * \overline{\boldsymbol{e_l}} + \overline{\lambda_{i,j,k,l}} (\overline{\boldsymbol{e_k}} * \boldsymbol{f_l'} + \overline{\boldsymbol{e_l}} * \boldsymbol{f_k'}) \; \forall i \leq j \tag{20}$$

Finally, as for the size of the system, each vectorial equation for (i,j) expands in n scalar (quadratic) equations, so a total of $nk(k+1)/2$. The lifts of the $(\boldsymbol{e_i})_i$ are n unknowns and the lifts of $\lambda_{i,j,k,l}$ are $k(k+1).\dim(C^{*2})$ unknowns. □

Observation 19. *For all AG codes that we tried —such as those of Corollary 30, for which a multiplicative lift exists—, then* every solution *of the system* mod p^i *(i.e. a multiplicative lift in $R_i(r)^n$), lifts to a solution* mod p^{i+1}

Thanks to this unexplained fact, we need only solving the system ℓ times to find a multiplicative lift of $\overline{C}$ in $R_\ell(r)$, hence the overall strategy runs in *polynomial time in n*, and *linear in ℓ*.

4.1 Example of a multiplication friendly lift modulo 2^{100}

Here we illustrate efficiency of our method by lifting a strongly multiplicative secret sharing scheme over $\mathbb{F}_{16}$ for 64 players and adversary threshold $t = 13$, into a scheme over $\mathbb{Z}/2^{100}\mathbb{Z}$, in a minute on a single processor. The Magma program used to compute the two following examples (among others) is in Appendix §G. In §C.1 we detail some optimization that we did, then in §C.2 we illustrate a case with a larger adversary threshold.

Let X_0 be the "Hermitian" plane curve over $\mathbb{F}_{16}$ defined by equation $f(x,T) = T^4+T-x^{4+1}$. Then it is well known that this curve has genus $g = 4(4-1)/2 = 6$ and $n+1 := |X_0(\mathbb{F}_{16})| = 1+4^3 = 65$ rational points (which reaches the Hasse-Weil upper-bound). Let us note these points $P_0, \ldots, P_n$ (so $n = 64$), consider the divisor $D_0 = 25P_0$, whose Riemann-Roch space $L(D_0)$ is of dimension 20. We define the algebraic geometry code $\overline{C}$ of length $n+1$ defined as evaluations elements of $L(D_0)$ on all the rational points of X_0, *including the support* $\{P_0\}$ of D_0.

So with the notations of [CC06, §3], we allow in addition to evaluate at Q. We did this to make a little gain on the adversary bound. To be sure, solutions to overcome the problem with evaluating at the support of D_0 are well known: see the details in Remark 2.

For the sake of illustration notice that, with $t = 13$, we have $\deg D_0 = 2g + t$ so that the condition $39 = 3t < n - 4g = 40$ of [CC06, Proposition 2] is satisfied, thus from $\overline{C}$ we can deduce a secret sharing scheme with strong multiplication for adversary bound $t = 13$.

Before going on, we compute the square code $\overline{C}^2$ and the (a priori larger) AG code associated to $L(2D_0)$, and check that both are equal. This means that equality actually holds in (12), which is actually stated by Theorem 28 (and proven in §D.4) since we have here $\deg D_0 \geq 2g + 1$. By the Riemann-Roch formula we have that $\overline{C(2D_0)}$ is of dimension $2.25 + 1 - 6 = 45$. From the generating set $(\overline{\boldsymbol{e_i}} * \overline{\boldsymbol{e_j}})_{i \leq j}$ of $\overline{C}^2$ we extract a basis $(\overline{\boldsymbol{e_k}} * \overline{\boldsymbol{e_l}})_{(k,l) \in B}$. We now look at the matrix expressing the $(\overline{\boldsymbol{e_i}} * \overline{\boldsymbol{e_j}})_{i \leq j}$ in terms of this basis (with the previous notations, this is the matrix of the coefficients $\overline{\lambda_{i,j,k,l}}$). It has $(\dim(C)(\dim(C)+1))/2 = 210$ lines (all ordered pairs $i \leq j$). Obviously the lines where the index (i, j) belongs to B contain a single coefficient, equal to one. And obviously these coefficients will remain equal to one in every lift mod p^ℓ so we can remove these $n \dim \overline{C}^2 = 64 \times 45$ relations (and the corresponding variables) from the system from now on.

After some optimizations described in Appendix §C.1, we end up with a system (20) of 10725 equations with 3305 unknowns but, surprisingly, of (still) very large kernel: dimension 83 (dimension 200 before applying the trick). We solve it in one second on a single processor.

Finally we repeat the operation, following the pattern of the proof of Proposition 18: we reinject the solution (the lifted vectors $\boldsymbol{e_i}$ and coefficients $\lambda_{i,j,k,l}$) in a system mod 2^3 (as in (16)), which is a multiple of 2^2 after simplification, thus falls back to a system mod 2 after "division by 2^2". Note the general fact that the matrix of the new system obtained is exactly the same as the initial one (20), because the coefficients depend only on the values modulo 2 of $\overline{\boldsymbol{e_i}}$ and $\overline{\lambda_{i,j,k,l}}$. To which we find again a solution (the mysterious lucky heuristic) —in one second as expected— then repeat exactly 97 times (always the lucky heuristic) to reach a multiplication friendly lift over $R_{100}(4)$.

5 Applications to MPC

5.1 Proof of Main Theorem 2

Proposition 20. *For any fixed p and ℓ, consider any fix number n of players, and choose any fixed even integer r such that $p^r \geq 64$, and security parameter κ. Then there exists a slowly growing* infinite *sequence of integers N such that: for any set of N triples a_i, b_i, c_i in $\mathbb{Z}/p^\ell\mathbb{Z}$ (resp. in $\mathbb{F}_p$), which are shared between the players using any linear secret sharing scheme, then there exists a protocol that has the following properties*

- *The protocol consumes an additional number of triples, which is asymptotically $N(1 + 2p^{-r/2})$, that are opened (so cannot be used anymore);*

– *Either all triples considered are correct: $a_i b_i = c_i$ then it outputs* true*, or at least one is incorrect, then it outputs* false *except with probability* $O(p^{-(r\kappa-1)/2}(1+4p^{-\kappa/2}))$

– *The communication complexity is* $nr(N+2\kappa^2)$ *of elements of* $\mathbb{Z}/p^\ell\mathbb{Z}$ *(resp. of* $\mathbb{F}_p$*) sent, the computational complexity is* $O(N)$ *linear operations in* $\mathbb{Z}/p^\ell\mathbb{Z}$ *(resp. in* $\mathbb{F}_p$*) per player.*

For simplicity we prove it over finite fields. Then the same methods to lift it over rings as in Main Theorem 1 apply. Consider the finite field extension $\mathbb{F}_{p^r}$ and an optimal family of algebraic curves over $\mathbb{F}_{p^r}$ with genera slowly growing to infinity. The best existing asymptotic ratio of the number of rational points divided by the genus, is denoted $A(p^r)$ "the Ihara constant". When r is *even* then Ihara showed existence of infinitely many curves with slowly growing genera such that it matches the upper-bound bound of Drinfeld-Vladuts: $A(p^r) = p^{r/2} - 1$. Recall that this bound is one order of magnitude lower than the Weil upper bound (which is relevant only for finite genera). Fix a curve $\mathcal{C}$ in this family, with genus $\mathfrak{g}$, such that it has at least $2(N+2\mathfrak{g}-1)$ points (which is possible for all N large enough since $p^r \geq 64$). Consider a fixed set of points $P_1, \ldots, P_N$ on this curve, and G a point of degree $N+2\mathfrak{g}-1$ (existence is guaranteed by [Sti09, Theorem 5.2.10 c)]). Then there exists an interpolation formula with coefficients linear in the a_i (resp. the b_i), that builds rational functions f (resp. g) in the Riemann Roch space $\mathcal{L}(G)$, such that they take the values a_i (resp. b_i) at the points $P_1, \ldots, P_N$. [The technique for this is as in Lagrange's interpolation formula: one considers for every point P_i a fixed public function χ_i that vanishes at all the P_j for $j \neq i$ but not at P_i. Existence of χ_i is guaranteed by a consequence of the Riemann-Roch formula: $\ell(G - \sum_{i\neq j} P_j) - \ell(G - \sum_j P_j) > 0$. Then, the function f is deduced as the linear combination $\sum_i (a_i/\chi_i(P_i))f_i$]. The players can thus obtain a secret sharing of coefficients of f seen as a linear combination of the public χ_i's (same for g). Define $h = fg$ in $\mathcal{L}(2G)$. Consider the remaining points of the curve: $P_{N+1}, \ldots, P_{2(N+2\mathfrak{g}-1)}$. Players sacrifice $N+2\mathfrak{g}-1$ auxiliary triples (possibly incorrect), in order to compute with the Beaver passively-secure protocol (so possibly incorrectly) secret sharings of the products $\widetilde{c_i} = f(P_i)g(P_i)$ at all those remaining points. At this point, if all triples are correct and no cheating occured, then we should have $h(P_i) = c_i$ for all $i = 1 \ldots N$ and $h(P_i) = \widetilde{c_i}$ for all $i = N+1 \ldots 2(N+2\mathfrak{g}-1)$ As above, players compute a secret sharing of the unique function $\widetilde{h}$ in $\mathcal{L}(2G)$ such that $\widetilde{h}(P_i) = c_i$ for all $i = 1 \ldots N$ and $\widetilde{h}(P_i) = \widetilde{c_i}$ (namely they locally compute a secret sharing of the linear coefficients of $\widetilde{h}$ along the public χ_i). Then they sample a random secret shared challenge value $\lambda \in \mathbb{F}_{p^{r\kappa}}$, compute locally secret sharings of the evaluations $f(\lambda), g(\lambda)$ and $\widetilde{h}(\lambda)$, compute a (possibly false) secret sharing of the product $f(\lambda)g(\lambda)$ by sacrificing $2\kappa^2$ triples (multiplication in $\mathbb{F}_{p^{r\kappa}}/\mathbb{F}_{p^r}$ being done with the schoolboy algorithm), then perform a equal-to-zero check on $f(\lambda)g(\lambda) - \widetilde{h}(\lambda)$. If it passes, then they return accept.

5.2 Existence of lifts of RMFE over rings, with constant rate

Let p be a prime and $r,\ k,\ m \geq 1$ be positive integers. $*$ denotes the componentwise product. We adapt over rings [Cas+18, Definition 1] (where $q = p^r$).

Definition 21. A pair (ϕ, ψ) is called an $(k,m)_{p^r}$ -Reverse Multiplication Friendly Embedding (RMFE) if $\phi : R_\ell(r)^k \to R_\ell(rm)$ and $\psi : R_\ell(rm) \to R_\ell(r)^k$ are two $R_\ell(r)$-linear maps satisfying

$$x * y = \psi(\phi(x)\phi(y)) \quad \text{for all } x, y \in R_\ell(r)^k \tag{21}$$

Theorem 22. *Consider the family of "Reverse multiplication friendly embeddings" (RMFE) of [Cas+18, Theorem 5] (where $q := p^r$), then there exists a family of RMFE of $(R_\ell(r))^k$ into $R_\ell(rm)$, with k slowly growing to infinity and the* same constant *asymptotic expansion rates m/k.*

Let us review the construction over fields of [Cas+18, Lem 6 & Cor 1] that provides [Cas+18, Theorem 5], and use the tools of §3.3 to show that it lifts. We consider a smooth curve over $\mathbb{F}_q$ of genus g, with k distinct rational points denoted $P_1, P_2, \ldots, P_k$. Let G be a divisor such that $\deg G \geq k + 2g + 1$ (and for simplicity, with support outside of $\{P_1, \ldots, P_k\}$). By the Riemann-Roch formula we thus have $\dim_{\mathbb{F}q} L(G) - \dim_{\mathbb{F}q} L(G - \sum_i Pi) = k$. By §3.3, the Riemann Roch spaces in this equality lift to free modules of same rank. Consider the evaluation map $\pi : L(G) \longrightarrow \mathbb{F}_q^k : f \longrightarrow (f(P_i))$, which has kernel $L(G - \sum_i P_i)$. Then π is surjective, since $\dim_{F_q} \mathrm{Im}(\pi) = \dim_{\mathbb{F}q} L(G) - \dim_{\mathbb{F}q} L(G - \sum_i P_i) = k$. Surjectivity is preserved over rings (by the invertible determinant mod p trick).

Choose a subspace W of $L(G)$ of dimension k such that π induces an isomorphism between W and $\mathbb{F}_q^k$. Choose R a point of degree $m > 2\deg(G)$, which exists for m large enough by [Sti09, Theorem 5.2.10 c)]. For any $f \in L(G)$, we denote by $\boldsymbol{c_f}$ the evaluation vector $(f(P_i))$, and by $f(R)$ the evaluation. The previous isomorphism induces the $\mathbb{F}_q$-linear map $\phi : \pi(V) = \mathbb{F}_q^k \longrightarrow \mathbb{F}_{q^m} : \boldsymbol{c_f} \to f(R)$. Then ϕ is injective, since $\deg(R) > \deg(G)$. Thus the lift over rings is also injective, by Lemma 9.

Define the $\mathbb{F}_q$ -linear map $\tau : L(2G) \longrightarrow \mathbb{F}_{q^m} : f \to f(R)$. Then τ is injective, since $m = deg(R) > deg(2G)$, and likewise for the lift by Lemma 9. Bijectivity of $\mathrm{Im}(\tau)$ with $L(2G)$ induces the $\mathbb{F}_q$-linear map $\psi' : \mathrm{Im}(\tau) \subseteq \mathbb{F}_{q^m} \longrightarrow \mathbb{F}_q^k : f(R) \to (f(P_i))$. Then ψ' surjective (but not injective), by the same degree reason than π, and likewise for surjectivity of the lift. We extend ϕ' from $\mathrm{Im}(\tau)$ to all of $\mathbb{F}_{q^m}$ linearly, and denote the resulting map ψ.

Finally, RMFE follows from the fact that, for any $\boldsymbol{c_f}, \boldsymbol{c_g} \in \mathbb{F}_q^k$ we have:

$$\psi(\phi(\boldsymbol{c_f})\phi(\boldsymbol{c_g})) = \psi(f(R)g(R)) = \psi((f.g)(R)) = \boldsymbol{c_{fg}} = \boldsymbol{c_f} * \boldsymbol{c_g}$$

where $f, g \in W$ are uniquely determined from $\boldsymbol{c_f}$, $\boldsymbol{c_g}$ by the injectivities above. Note that $(fg)(R)$ belongs to $\mathrm{Im}(\tau)$ since $fg \in L(2G)$.

5.3 Proof of Main Theorem 3

We can now compile a protocol for a circuit over a large Galois ring $R_\ell(r)$, into a protocol for many evaluations in parallel of this circuit in $\mathbb{Z}/p^\ell\mathbb{Z}$ by casting over rings the protocols of [Cas+18]. Since we choose to restrict ourselves to the case of optimal adversary rate, we really need hyperinvertible matrices over Galois rings for any number of players (not the alternative with suboptimal adversary bound discussed in [Cas+18, §2.4]). Fortunately their construction is straightforward, see e.g. [Abs+19a]. We can thus cast the original protocol of Beerliova-Hirt over Galois rings, then compensate their bad asymptotic communication overhead by amortizing it over several instances in parallel, exactly as done in [Cas+18, Theorem 1 & 2]. Namely, the main tool are RMFE over rings with asymptotically *linear* rate, which is solved above in §5.2. Whereas the "tensoring-up" trick carries over rings without any technical difficulty.

5.4 An analogous efficient Hensel lifting for RMFE

Again we consider for simplicity only the base field $\mathbb{F}_p$, instead of $\mathbb{F}_{p^r}$. Let us make the following useful rephrasing of the definition of a *reverse multiplication embedding* (RMFE) of $\mathbb{F}_p^k$ into $\mathbb{F}_{p^m}$ Consider the field extension $\mathbb{F}_{p^m}$, equipped with its internal multiplication law. Denoting the dual over $\mathbb{F}_p$ with $*$, this law is captured by what is denoted as the *multiplication tensor* $T \in \mathbb{F}_{p^m}^* \otimes \mathbb{F}_{p^m}^* \otimes \mathbb{F}_{p^m}$ Its components $T_{i=1..m}$ are $\mathbb{F}_p$-bilinear forms from $\left(\mathbb{F}_{p^m} \times \mathbb{F}_{p^m}\right)$ to $\mathbb{F}_p$. Now fix a linear map

$$\phi : \mathbb{F}_p^k \longrightarrow \mathbb{F}_{p^m}$$

The pull back of T :

$$\phi^* T = T(\phi(.), \phi(.))$$

decomposes in $\mathbb{F}_{p^m}$ in m components which are symmetric bilinear forms

$$\phi^* T_i = T_i(\phi(.), \phi(.)) \ , i = 1..m$$

belonging by definition to the symmetric tensor space of the linear forms $S^2((\mathbb{F}_p^k)^*)$.

Definition 23. Consider the (nonintegral) algebra $\mathbb{F}_p^k$, equipped with the multiplication law component-by-component. This law is captured by what is denoted as the "multiplication tensor", belonging to $\left(\mathbb{F}_p^k\right)^* \otimes \left(\mathbb{F}_p^k\right)^* \otimes \mathbb{F}_p^k$. We say that ϕ is a reverse multiplication embedding iff these m bilinear forms $\phi^* T_i$ generate the components $(x_1^* \otimes x_1^*, ...x_k^* \otimes x_k^*)$ of the multiplication tensor.

Lifting of an algorithm ϕ modulo p^2: Suppose we are given a reverse multiplication friendly embedding ϕ, over $\mathbb{F}_p$ ($r = 1$ to make notations simple): for each $j = 1 \dots k$, we have coefficients $\lambda_{i,j}$ such that:

$$x_j^* \otimes x_j^* = \sum_{i=1}^{m} \lambda_{i,j}.\, \phi^* T_i \tag{22}$$

(it is a tensorial equality: it takes place in the space of symmetric bilinear forms of length k, so expands on coordinates as a set of $k(k+1)/2$ equations). We want to lift ϕ and the coefficients $\lambda_{i,j}$ such that the equalities (22) hold modulo p^2. (So we have $mk+mk$ unknowns and m equations, each of them taking place in a symmetric tensor space of dimension $k(k+1)/2$). Consider arbitrary lifts ϕ' and $\lambda'_{i,j}$ of ϕ and $\lambda_{i,j}$ over $\mathbb{Z}/p^2\mathbb{Z}$, we thus obtain the (tensorial) equalities modulo p^2 for $j = 1..k$:

$$x_j^* \otimes x_j^* = \sum_{i=1}^{m} \lambda'_{i,j} \phi'^* T_i + p\Delta_j$$

and we would like to eliminate the error terms $p\Delta_j$ modulo p^2 by choosing better lifts of ϕ and of $\lambda_{i,j}$:

$$\phi' + p\psi \text{ and } \lambda'_{i,j} + p\mu_{i,j} \tag{23}$$

After replacing (23) in (22) then simplification, the equation becomes the following (tensorial) *linear* equation modulo p (so with coordinates in $\mathbb{F}_p$):

$$\sum_{i=1}^{m} 2\lambda'_{i,j} T_i\big(\phi'(.), \psi(.)\big) + \mu'_{i,j} T_i\big(\phi'(), \phi'_i(.)\big) = -\Delta_j$$

where the unknowns are ψ and $\mu'_{i,j}$.

We will not state and prove that the previous systems returns all lifts modulo p^2, as it is very similar to Proposition 18, nor will we discuss again how to repeat and compute higher lifts modulo p^ℓ.

5.5 Proof of Theorem 5 (C) (robust reconstruction algorithm)

5.5.1 (C): Robust reconstruction (codes version) Here we quickly explain how to decode with errors in *any* free code, which thus generalizes the algorithm of [Abs+19a, §3.2] in Reed-Solomon codes. It uses, as a black box subroutine, any given decoding algorithm for the reduction $\overline{C}$ over the residue field $\mathbb{F}_{p^r}$, and uses it with a *linear* overhead in ℓ. [For p-adic rings we have $\ell = \infty$: the decoding algorithm will return iteratively a solution, where the error term remains on the same support and has smaller and smaller p-adic norm.].

Proposition 24 (A compiler from error-correction over fields to rings). *Let $\big(R, (p)\big)$ be a principal ideal local ring and C be a code in R^n which is a* free *code. Then we can compile any decoding algorithm $\overline{\phi}$ for the code $\overline{C}$ over the residue field (up to half of the minimum distance), into an algorithm ϕ for decoding-with-errors in C, with complexity equal to ℓ times the complexity of $\overline{\phi}$, where ℓ is such that $p^\ell = 0$.*

Let us describe informally the decoding algorithm (with justifications inline). Recall that the operation of lifting a codeword $\bar{c} \in \overline{C}$ to C can be done

efficiently, thanks to the existence of a generating matrix for C in systematic form (Proposition 7).

Let $\boldsymbol{c} \in C$ be an unknown codeword, $\boldsymbol{e} \in R^n$ an error term with weight $< \overline{d}/2$ and $\boldsymbol{u} = \boldsymbol{c} + \boldsymbol{e}$ the corrupted codeword to be decoded. Repeat the following procedure:

- Decode $\overline{\boldsymbol{u}}$ into $\overline{\boldsymbol{c}}$ and deduce $\overline{\boldsymbol{u}} - \overline{\boldsymbol{c}} = \overline{\boldsymbol{e}}$.
- Choose any lift $\boldsymbol{c_1} \in C \pmod{p^2}$ of $\overline{\boldsymbol{c}}$ and $\boldsymbol{e_1} \in (R/p^2)^n$ any lift of $\overline{\boldsymbol{e}}$ with same support.
- Compute the difference $\boldsymbol{u} - (\boldsymbol{c_1} + \boldsymbol{e_1})$ modulo $(p)^2$: it is equal to

  $$(\boldsymbol{c} - \boldsymbol{c_1}) + (\boldsymbol{e} - \boldsymbol{e_1})$$

  where the left term is by construction a codeword in $C \cap (R/p^2)^n$, and thus in pC by Theorem 25 (i) (in §B): let us call it $p\boldsymbol{c_2}$. Whereas the right term is in pR^n with at most the same support as $\overline{\boldsymbol{e}}$ (so $< \overline{d}/2$ nonzero coordinates): let us call it $p\boldsymbol{e_2}$.
- the difference computed is thus of the form $p\boldsymbol{u_1}$: dividing by p (that is: choosing any preimage under the multiplication by p) we obtain the equation modulo p

  $$\boldsymbol{u_1} = \boldsymbol{c_2} + \boldsymbol{e_2}$$

- Apply the decoding algorithm to $\boldsymbol{u_1}$, deduce $\boldsymbol{c_2}$ and $\boldsymbol{e_2}$, lift them arbitrarily in $C \mod p^3$ and in $(R/(p^3))^n$ etc.

As an alternative, we could notice that, since we obtained LSSS with *strong multiplication* in Main Theorem 1, then we can lift over Galois rings the generic decoding algorithm [CDN15, §12.5.4]

References

[Abs+19a] Mark Abspoel, Ronald Cramer, Ivan Damgård, Daniel Escudero, and Chen Yuan. "Efficient Information-Theoretic Secure Multiparty Computation over $\mathbb{Z}/p^k\mathbb{Z}$ via Galois Rings". In: *TCC*. 2019.

[Abs+19b] Mark Abspoel, Anders Dalskov, Daniel Escudero, and Ariel Nof. *An Efficient Passive-to-Active Compiler for Honest-Majority MPC over Rings*. to appear in ACNS'21. 2019.

[Abs+20] M. Abspoel, R. Cramer, C. Yuan, I. Damgard, D. Escudero, M. Rambaud, and C. Xing. "Asymptotically Good Multiplicative LSSS over Galois Rings and Applications to MPC over $\mathbb{Z}_{2^k}$". In: *Asiacrypt*. 2020.

[Ara+18] Toshinori Araki, Assi Barak, Jun Furukawa, Marcel Keller, Yehuda Lindell, Kazuma Ohara, and Hikaru Tsuchida. "Generalizing the SPDZ Compiler For Other Protocols". In: *CCS*. 2018.

[AZ18] C. Bachoc A. Couvreur and G. Zémor. "towards a function field version of Freiman's Theorem". In: *Algebraic Combinatorics* (2018).

[BFO12] Eli Ben-Sasson, Serge Fehr, and Rafail Ostrovsky. "Near-Linear Unconditionally-Secure Multiparty Computation with a Dishonest Minority". In: *CRYPTO*. 2012.

[BH08] Z. Beerliová-Trubíniová and M. Hirt. "Perfectly-Secure MPC with Linear Communication Complexity". In: *TCC*. 2008.

[BMN18] Alexander R. Block, Hemanta K. Maji, and Hai H. Nguyen. "Secure Computation with Constant Communication Overhead Using Multiplication Embeddings". In: *Indocrypt*. 2018.

[Cas+09] I. Cascudo, H. Chen, Ronald Cramer, and C. Xing. "Asymptotically Good Ideal Linear Secret Sharing with Strong Multiplication over Any Fixed Finite Field". In: *CRYPTO*. Springer, 2009.

[Cas+18] Ignacio Cascudo, Ronald Cramer, Chaoping Xing, and Chen Yuan. "Amortized Complexity of Information-Theoretically Secure MPC Revisited". In: *CRYPTO*. Springer, 2018.

[Cas16] Ignacio Cascudo. "Secret Sharing Schemes with Algebraic Properties and Applications". In: *Pursuit of the Universal*. 2016.

[CC06] Hao Chen and Ronald Cramer. "Algebraic Geometric Secret Sharing Schemes and Secure Multi-Party Computations over Small Fields". In: *CRYPTO*. Springer, 2006.

[CCX11] Ignacio Cascudo, Ronald Cramer, and Chaoping Xing. "The Torsion-Limit for Algebraic Function Fields and Its Application to Arithmetic Secret Sharing". In: *CRYPTO*. 2011.

[CDM00] Ronald Cramer, Ivan Damgård, and Ueli Maurer. "General Secure Multi-party Computation from Any Linear Secret-sharing Scheme". In: *EUROCRYPT*. 2000.

[CDN15] Ronald Cramer, Ivan Bjerre Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. New York, NY, USA: Cambridge University Press, 2015.

[CG20] Ignacio Cascudo and Jaron Skovsted Gundersen. "A Secret-Sharing Based MPC Protocol for Boolean Circuits with Good Amortized Complexity". In: *TCC*. 2020.

[Cra+18] R. Cramer, I. Damgård, D. Escudero, P. Scholl, and C. Xing. "SPD$\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for Dishonest Majority". In: (2018).

[CS95] A. R. Calderbank and N. J. A. Sloane. "Modular and p-adic cyclic codes". In: *Designs, Codes and Cryptography* (1995).

[Dam+06] Ivan Damgård, Matthias Fitzi, Eike Kiltz, Jesper Buus Nielsen, and Tomas Toft. "Unconditionally Secure Constant-Rounds Multi-party Computation for Equality, Comparison, Bits and Exponentiation". In: *TCC*. 2006.

[Dam+12] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. "Multiparty Computation from Somewhat Homomorphic Encryption". In: *Advances in Cryptology – CRYPTO 2012*. 2012.

[Dam+19] I. Damgård, D. Escudero, T. Frederiksen, M. Keller, P. Scholl, and N. Volgushev. "New Primitives for Actively-Secure MPC over Rings

with Applications to Private Machine Learning". In: *IEEE S&P*. 2019.

[DLN19] Ivan Damgård, Kasper Green Larsen, and Jesper Buus Nielsen. "Communication Lower Bounds for Statistically Secure MPC, With or Without Preprocessing". In: *CRYPTO*. 2019.

[DLS20] Anders Dalskov, Eysa Lee, and Eduardo Soria-Vazquez. "Circuit Amortization Friendly Encodingsand Their Application to Statistically Secure Multiparty Computation". In: *ASIACRYPT*. 2020.

[DOS18] Ivan Damgård, Claudio Orlandi, and Mark Simkin. "Yet Another Compiler for Active Security or: Efficient MPC Over Arbitrary Rings". In: *CRYPTO*. 2018.

[FY92] Matthew Franklin and Moti Yung. "Communication Complexity of Secure Computation (Extended Abstract)". In: *STOC*. 1992.

[GS99] V. Guruswami and M. Sudan. "Improved decoding of Reed-Solomon and algebraic-geometry codes". In: *IEEE Trans. Inf. Theory* (1999).

[GSZ20] Vipul Goyal, Yifan Song, and Chenzhi Zhu. "Guaranteed Output Delivery Comes Free in Honest Majority MPC". In: *Advances in Cryptology – CRYPTO 2020*. 2020.

[Hes02] F. Hess. "Computing Riemann—Roch Spaces in Algebraic Function Fields and Related Topics". In: *J. Symb. Comput.* (2002).

[Ill05] Luc Illusie. "Grothendieck's existence theorem in formal geometry". In: *Fundamental Algebraic Geometry: Grothendieck's FGA Explained*. Ed. by AMS. 2005.

[Ish+07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. "Zero-knowledge from Secure Multiparty Computation". In: *STOC*. ACM, 2007.

[Khu04] Kamal Khuri-Makdisi. "Linear Algebra Algorithms for Divisors on an Algebraic Curve". In: *Math. Comput.* 73.245 (2004).

[Mum11] David Mumford. "Varieties Defined by Quadratic Equations". In: *Questions on Algebraic Varieties*. Springer, 2011.

[NW17] Anand Kumar Narayanan and Matthew Weidner. "Subquadratic time encodable codes beating the Gilbert-Varshamov bound". In: *CoRR* (2017). URL: `http://arxiv.org/abs/1712.10052`.

[PC20] İ. Keskinkurt Paksoy and M. Cenk. *TMVP-based Multiplication for Polynomial Quotient Rings and Application to Saber on ARM Cortex-M4*. `https://eprint.iacr.org/2020/1302`. 2020.

[PS20] Arpita Patra and Ajith Suresh. "BLAZE: Blazing Fast Privacy-Preserving Machine Learning". In: *NDSSS* (2020).

[RB89] T. Rabin and M. Ben-Or. "Verifiable Secret Sharing and Multiparty Protocols with Honest Majority". In: *STOC*. 1989.

[SAS17] M. Shi, A. Alahmadi, and P. Solé. *Codes and Rings*. Academic Press, 2017.

[SG20] P.-J. Spaenlehauer and A. le Gluher. "A fast randomized geometric algorithm for computing Riemann-Roch spaces". In: *Mathematics of Computation* (2020).

[SGA1] Alexander Grothendieck. *SGA 1*. LNM 224. Springer, 1964.

[Shu+01] K. W. Shum, I. Aleshnikov, P. V. Kumar, H. Stichtenoth, and V. Deolalikar. "A low-complexity algorithm for the construction of algebraic-geometric codes better than the Gilbert-Varshamov bound". In: *IEEE Trans. Inf. Theory* (2001).

[Sta18] The Stacks project authors. *The Stacks project*. `https://stacks.math.columbia.edu`. 2018.

[Sti09] H. Stichtenoth. *Algebraic Function Fields and Codes, 2nd edition*. Springer, 2009.

[SW99] M. A. Shokrollahi and H. Wasserman. "List decoding of algebraic-geometric codes". In: *IEEE Trans. Inf. Theory* (1999).

[Wal] Judy Walker. "Algebraic geometry codes over rings". PhD thesis. Univ Illinois Champain.

[Wal99] Judy L. Walker. "Algebraic geometric codes over rings". In: *Journal of Pure and Applied Algebra* 144.1 (1999), pp. 91–110.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2\Delta^2+3\Delta+2 & \Delta+2 & \Delta^2+2\Delta & 3\Delta^2+2\Delta+2 & \Delta+3 & \Delta+1 & \Delta^2+\Delta+1 & \Delta^2+3\Delta+3 & 3\Delta^2+\Delta+1 & 3\Delta^2+3 & 3\Delta^2+3 & 1 \\ \Delta^2 & \Delta^2 & 3\Delta^2+3\Delta & 3\Delta^2+3\Delta & \Delta^2+2\Delta+1 & \Delta^2+2\Delta+1 & \Delta+2 & \Delta+2 & 2\Delta^2+3\Delta+3 & 3\Delta^2+3\Delta+3 & \Delta^2+3\Delta+1 & 1 \\ 1 & \Delta & 1 & \Delta^2 & \Delta^2 & \Delta^2+\Delta & 1 & \Delta^2+\Delta & \Delta & \Delta & \Delta^2+\Delta & 0 \end{bmatrix}$$

$$transp\Big((\lambda_{2,2,k,l})_{k,l\in B},(\lambda_{4,4,k,l})_{k,l\in B}\Big)=\begin{bmatrix}0&0&1&0&0&0&0&0\\1&0&0&1&1&1&2&0\end{bmatrix}$$

A Reminders of [CC06; Cas+09]

A.1 Carrying the "multiplication friendly embeddings" of [Cas+09] over rings

We consider any $2m-2 \leq p$. We have an isomorphism of vector spaces $\phi : \mathbb{Z}/p^\ell\mathbb{Z}[X]_{<m} \longrightarrow R_\ell(m)$ given by $X \longrightarrow \Delta$ with the notations of (1). Let us take $2m-2$ elements in $\mathbb{Z}/p^\ell\mathbb{Z}$: $\beta_1, \ldots, \beta_{2m-2}$ *that have pairwise distinct reductions modulo* p. Consider the map $\xi : \mathbb{Z}/p^\ell\mathbb{Z}[X]_{<m} \longrightarrow R_\ell(2m-1)$ given by $g \longrightarrow (g(\beta_1), \ldots, g(\beta_{2m-2}), g(\infty))$ where the last component is the coefficient of degree $m-1$ of g. Then the key property preserved is that for any polynomials g and h in $\mathbb{Z}/p^\ell\mathbb{Z}[X]_{<m}$, then the image of the map $\xi * \xi : (g,h) \longrightarrow (gh(\beta_1), \ldots, gh(\beta_{2m-2}), gh(\infty))$ *determines uniquely* gh. This is because one can interpolate the polynomial gh, of degree $2m-2$, from $2m-1$ points with distinct reductions modulo p. Indeed, one can write the Lagrange interpolation formula since the denominators, which are products of $\beta_i - \beta_j$, are invertible modulo p, and thus invertible.

A.2 Reminders on the asymptotic parameters

Recall first the tradeoff of [CC06, §5] for secret sharing in finite fields $\mathbb{F}_p$. Let us cast a secret in $\mathbb{F}_p$, into the extension $\mathbb{F}_{p^r}$ of degree r, such that

$$p^r \geq 49 \, .$$

Then for adversary threshold $1/3-\epsilon$, and for infinitely many number of players, there exists an ASSSM over $\mathbb{F}_{p^r}$ and size r of shares, such that:

$$\epsilon < \frac{4}{3(p^{r/2}-1)} \, ,$$

[CC06, §5] In particular, choosing $\widehat{r}(\epsilon) = -2\log(\epsilon)$ yields an adversary bound $1/3-\epsilon$ when ϵ is sufficiently small.

Notice that the classical bound for the dual distance of AG codes over fields is not stated explicitly in [Cas+09; CCX11]. But its parameters are well known since Goppa (recalled e.g. in [Wal99, Theorem 2.1]), and also asymptocially optimal in our regime $2g-2 < \deg D_0 < n$. Which supports the claims of [Cas+09; CCX11], and thus ours by Proposition 8.

B A chain of Equivalences for Free Codes

Theorem 25. *Let C be a code in R^n, then the following are equivalent:*

(0) C is free;
(0') C is a direct summand in R^n;
(i) the inclusion $pC \subset pR^n \cap C$ is an equality;

(ii) C is free and generated by any lift *in C of* any basis *of $\overline{C}$. In particular the rank equals dimension of the reduced:*

$$\mathrm{rk}\,(C) = \dim \overline{C}\ .$$

Proof. (0) => (0') is the first statement of [Wal99, Lemma 3.2] (notice that it is specific to Artinian rings), applied to the inclusion $C \hookrightarrow R^n$. Indeed recall that a map "splits" means that it has a left retraction. In particular it is then standard that the image of such a map, in R^n, is a direct summand.

(0') => (i) This is the second statement of [Wal99, Lemma 3.2]. Alternatively, let us prove it under the following friendlier form. *Claim:* if $t \in R$, $z \in R^n$ are such that tz belongs to C, then there exists $c \in C$ such that $tz = tc$ (which, when t is a non-zero divisor, is equivalent to $c \in C$);

Proof: write $C \oplus C' = R^n$ (internal direct sum). Suppose tz in C. Write $z = c + c'$. Thus $tz = tc + tc'$ in C. Hence, $tc' = 0$. So $tz = tc$.

(i) => (ii) Take any basis $(\overline{e_i})$ of the k-vector space C/pC and lift it arbitrarily to (e_i) in C. Then by Nakayama's Lemma [See e.g. Corollary 2.7 of Atiyah-MacDonald's Introduction to commutative algebra], the (e_i) automatically form a basis of C. But by assumption we have

$$\overline{C} := C/(pR^n \cap C) = C/pC$$

(ii) => (0) is immediate. □

C Examples of computations of lifts over rings of ASSSM

C.1 Optimizations

Remark 1. List the (i,j) for which the decomposition of $\overline{\boldsymbol{e_i}} * \overline{\boldsymbol{e_j}}$ is very simple: one single nonzero coefficient $\overline{\lambda_{i,j,k,l}}$ equal to one and the others equal to zero. Which includes, but far from exclusively, the basis vectors $\overline{\boldsymbol{e_k}} * \overline{\boldsymbol{e_l}}$ themselves. Then ask for these relations to hold modulo p^2, p^3 etc: this removes all the variables $\mu'_{i,j,k,l}$ from the system, for those "forced" relations on $\boldsymbol{e_i} * \boldsymbol{e_j}$. In practice this divides by two the dimension of the kernel while the system still yields solutions (actually one solution is enough for us, thanks to our lucky heuristic: see below) In practice, for algebraic geometry codes, this seems to make the number of equations drop from $k(k+1)/2$ to approximately $\dim\,(C^{*2}) \sim 2k$: see in the examples below.

Let us now illustrate some more optimizations on the example of §4.1:

Remark 2. With the notations of [CC06, §3], we allowed in addition to evaluate at Q. We did this to make a little gain on the adversary bound. To be sure, solutions to overcome the problem with evaluating at the support of D_0 are well known. A standard trick would have been to choose instead D_0 equal to one point of degree 25, but this wouldn't fit in our simplistic assumption of Corollary 30, that D_0 is supported on rational points. So we do otherwise and

keep $D = 25P_0$, compute t_0 a uniformizing parameter at P_0, and define the evaluation of $f \in L(D_0)$ at P_0 by:

$$(t_0^{23} f)(P_0) \ .$$

The intrinsic meaning of this formula is that we first compute the restriction (called "evaluation", in Theorem 15) of $\mathcal{L}(D_0)$ in a neighborhood of P_0: multiplying $\mathcal{L}(D_0)$ by t_0^{23} maps it to regular functions at P_0. Then we evaluate. The trick yields codes that still satisfy the conditions of Corollary 30 for the existence of finding multiplicative lifts [6]

Remark 3. To our surprise, even when removing the lines of Remark 1 from the matrix, the remaining matrix (which we call "Reduc" in the program) contains many other lines (119, we call their list "Fixed" in the program) that have also this property to have only one nonzero entry, equal to one. Looking at the global sections in $L(D_0)$ corresponding to these equalities $\overline{\boldsymbol{e_i}} * \overline{\boldsymbol{e_j}} = \overline{\boldsymbol{e_k}} * \overline{\boldsymbol{e_l}}$, we check that these equalities also hold for the underlying functions (which we already knew, since the evaluation map is injective). So, betting on the fact that this simple relations will also hold on the curve lifted over rings, we force these coefficients $\overline{\lambda_{i,j,k,l}} = 1$ to lift to 1 (and likewise the other coefficients on the line to lift to zero). Namely, in all iterations of the linear system mod 2^ℓ, we force all the corresponding $\mu'_{i,j,k,l} = 0$, and $\lambda_{i,j,k,l} = 1$ for these special lines. As described in Remark 2, it seems that we still get many solutions to the system after this trick, which yields a significant drop in the number of unknowns in the system (20) .

Also, the rest of the matrix Reduc is also hollow, ($O(1)$ nonzero coefficients per line, from the other examples we tested), thus the overall system (20) is *sparse*.

C.2 Lifting an AG code from the manyPoints.org curve of genus 3 over $\mathbb{F}_{2^5}$

Here we want a comparatively larger adversary threshold than in §4.1, so take an extension of $\mathbb{F}_2$ of larger size. Let δ be a root of the polynomial $1 + T^2 + T^5$ in $\mathbb{F}_2[T]$. Consider the plane curve X_0 over $\mathbb{F}_{2^5}$ defined as follows: let x, y be the affine coordinates, put $X := x^2 + x$, $Y := y^2 + y$ then X_0 is given by the equation:

$$X^2 + XY + \delta^3 Y^2 + Y + \delta^{26}$$

Its function field has

$$n = 64$$

[6] For the first trick, notice that closed points of arbitrary degree do lift so the corresponding divisors satisfy Corollary 30. For the second trick, consider $Z^{(0)}$ a $R_{100}(4)$-point of the lifted curve X above P_0 and t a uniformizing element as in [Wal, Proposition 4.9]. Then t^{23} is by construction the local equation of the lifted Cartier divisor D at the closed point $P^{(0)}$ in $Z^{(0)}$, and reduces to t_0.

places of degree one (i.e. a projective smooth model of X_0 would have 64 rational points). Consider the place at infinity $P_0(0, \delta, 1)$ and the divisor $D_0 = 22P_0$ (of degree $2g+16$). The Riemann Roch space $L(D_0)$ has dimension $22+1-3 = 21$, of which Magma can compute a basis E. We construct the AG code $\overline{C}$ obtained by evaluating these basis elements at all the rational points except the support P_0 (this time, avoiding the support doesn't harm the adversary bound).

Here the square $\dim \overline{C^2}$ is of dimension 42, after applying the same "special rows" trick the system contains 10584 equations and 5817 variables, and half of its rows are sparse. Surprisingly it (still) has very large kernel, of dimension 93. This time we solve it in 100 seconds and, thanks again to the lucky heuristic, need only iterating 98 times to finish with a lift mod 2^{100}.

D Stronger result: lifting to *free* codes over rings, with multiplicativity

D.1 Statement and roadmap

Main Theorem 26. Fix any prime number $p > 0$ and any integer $\ell > 0$. Write $R_\ell = \mathbb{Z}/p^\ell\mathbb{Z}$. For each integer $r > 0$, denote the degree-r Galois-ring extension of R_ℓ by $R_\ell(r)$. Then there is a fixed integer $\hat{r} = \hat{r}(p) > 0$ and a (dense) family of $R_\ell(\hat{r})$-linear codes C of unbounded length such that:

1. Denoting reduction of C modulo p (an $\mathbb{F}_{p^r}$-linear code) by $\overline{C}$, each of $\overline{C}$, $(\overline{C})^\perp$, $(\overline{C})^{*2}$ is asymptotically good.
2. Each of C, $C^\perp$, C^{*2} is free over $R_\ell(\hat{r})$, with the same dimension as its reduction. Therefore, each has the same minimum distance as its reduction. Particularly, each is asymptotically good.
3. All constructions are efficient.

We begin with an elementary result on the freeness of the dual. The real proof starts with the algebraic-geometric Theorem 28. Then, Main Theorem 26 will follow from Corollary 30, as explained at the end of the section.

D.2 Freeness of the dual code

Judy Walker proved that minimum distance of a free code was inherited from the code below (notice that equality is specific to Artinian rings, in general we have $d(C) \geq d(\overline{C})$), while Calderbank-Sloane [CS95, 3)] noticed that the dual of a free code was also free[7]. These two properties combined imply that C also has the same dual distance as $\overline{C}$.

Proposition 27. *[Wal99, Theorem 3.4] If C is a free code, then we have*

[7] Actually they didn't state this explicitly, but they were the first to establish the form of a generating matrix of $C^\perp$ over $\mathbb{Z}/p^\ell\mathbb{Z}$, which clearly generates a free module when C is free. So for convenience of the reader we instead quote the reference book, which states this property explicitly over —in particular— any Galois ring

(iii) $d(C) = d(\overline{C})$;
(iv) [SAS17, Prop 5.5] $C^\perp$ *is a free lift of* $\overline{C}^\perp$.
(thus:) $C^\perp$ *is of rank equal to the co-rank of* C *and* $C^\perp = d(\overline{C}^\perp)$.

D.3 Equality in equation (12)

Theorem 28. *Let* X_0 *be a function field of genus* g *over any finite field* $\mathbb{F}_{p^r}$, $\left(P_0^{(j)}\right)_j$ *the rational places (i.e. of degree one) of* X_0 *and* $\mathcal{P}_0 = P_0^{(1)}, \ldots, P_0^{(n)}$ *a subset of them. Consider any divisor* D_0 *on* X_0 *with support on rational places*[8], *and degree*

$$2g + 1 \leq \deg(D_0) < \frac{n}{2} .$$

Then we can construct algebraic geometry codes $C(D_0)$ *and* $C(2D_0)$ *defined by evaluation of the Riemann-Roch spaces* $L(D_0)$ *and* $L(2D_0)$ *on* $\mathcal{P}_0$, *and free lifts* $C(D)$ *and* $C(2D)$ *over* $R_\ell(r)$ *such that:*

$$C(D)^{*2} = C(2D) . \tag{24}$$

We then show in §D.3 Theorem 29 that equality in (24) holds over fields: $C(D_0)^{*2} \subset C(2D_0)$ as soon as $\deg(D_0) \geq 2g + 1$. This results from a hard theorem of Mumford, which we generalize to nonnecessarily algebraically closed fields, as proven in appendix D.6, by standard arguments. Fortunately, we also obtain an elementary proof of Theorem 29 in the interesting cases where $\deg(D_0) \geq 4g$: see §D.5. We then deduce equality over rings: (24) from the elementary Corollary 13, finishing the proof of Theorem 28.

From the elementary theory we deduce Corollary 30, which, instantiated e.g. on Garcia-Stichtenoth towers (or any other optimal family), immediatly yields Main theorem 26.

D.4 Proof of equality in (24) of Theorem 28

Firstly, *over fields*, the following theorem gives a criterion to have equality for the *reductions*:

$$C(D_0) * C(D_0') = C_0(D_0 + D_0') .$$

Theorem 29. *Let* D_0, D_0' *be two divisors of a function field* X_0 *of genus* g *over any field* K. *Suppose that* $\deg D_0 \geq 2g$ *and* $\deg D_0' \geq 2g + 1$. *Then*

$$L(D_0)L(D_0') = L(D_0 + D_0') \tag{25}$$

This theorem is deduced in Appendix D.6 from Mumford's normal generation criterion, which we extend to any field. See also the next section for more elementary proofs of Theorem 29 in particular cases.

[8] Inluding possibly points of $\mathcal{P}_0$: see Remark 2

From the inclusion in (12), and under the degree assumptions of Theorem (29), we can then apply Corollary 13 to

$$E := C(D) * C(D) \ \subset \ G := C(2D)$$

to deduce that equality (25) holds *over rings*, which proves Theorem 28.

From the properties on the distance and dual distance of free lifts stated in Theorem 25 (iii) and (iv), we can finally state:

Corollary 30. *Let X_0 be a function field of genus g over any finite field $\mathbb{F}_{p^r}$. Let D_0 be a divisor on X_0 with support on rational points (i.e. of degree one) and with degree*

$$2g + 1 \leq \deg(D_0) < \frac{n}{2} .$$

Let $L(D_0)$ be the Riemann-Roch space and $P_1, \ldots, P_n$ a collection of rational points on X_0. Define the algebraic geometry code $\overline{C}$ as the isomorphic image of $L(D_0)$ by the evaluation map on the $(P_i)_i$.

Then for any positive integer ℓ, $\overline{C}$ lifts to a free submodule C over the Galois ring $R_\ell(r)$, of same dimension and dual distance than $\overline{C}$, and such that the square *C^{*2} is also free of rank $\dim \overline{C}^{*2}$ and minimal distance $d(\overline{C}^{*2})$.*

D.5 Elementary proof of Theorem 29 in a particular case

D.5.1 $D_0 = dP_0$ supported at a rational point, with degree $d \geq 4g$

We first prove two lemmas on gaps between Riemann-Roch spaces.

Lemma 31. *With the same assumptions, for any integer*

$$v \in \left[\left\lceil \frac{d}{2} \right\rceil, \ldots, d \right]$$

Then there exists a rational function $y_0 \in L(D)$ with exactly a pole of order v at P.

Proof. Claim: for all $i' \leq \left\lceil \frac{d}{2} \right\rceil + 1$, then we have that $l(K - (D - i')) = 0$ for degree reasons. Indeed:

$$\deg(K - (D - i')) \leq 2g - 2 - d + \left\lceil \frac{d}{2} \right\rceil + 1 < 2g - 1 - 4g + \left(\frac{d}{2} + 1\right) \leq 0 .$$

From the claim it follows that for all integer $i \leq \left\lceil \frac{d}{2} \right\rceil$, we have a gap in the sequence of dimensions:

$$l(D - iP) < l(D - (i+1)P) , \tag{26}$$

thus the result. □

Proof of the theorem: Consider f_0 a function in $L(2D) = L(2dP)$. Either it is in $L(D)$, and we are done. Or it has a pole at P with order strictly larger than d:

$$w := \mathrm{ord}_P(f) > d$$

(and by definition no other pole elsewhere). In this case, Lemma 31 implies that there exist y_0, y_0' in $L(D)$ such that

$$\mathrm{ord}_P(y_0) + \mathrm{ord}_P(y_0') = w$$

and thus, up to multiplying y_0 by a constant ρ_0, we have that the function:

$$f_1 = f_0 - y_0 y_0'$$

has a pole at P strictly lower than w (and by construction no pole elsewhere). Since $y_0 y_0'$ is in $L(D)^2$, we can conclude by recursion on the order of the pole of f_1 at P.

D.6 Proof of Theorem 29: extending Mumford's normal generation criterion over any field

The following theorem is stated in [Mum11, Theorem 6] over any algebraically closed field. The goal of this section is to deduce that the theorem holds over any field, which is exactly the statement of Theorem 29 (formulated with function fields, see e.g. [AZ18, Theorem 6.1]).

Theorem 32. *Let X be a smooth projective curve over an algebraically closed field k. Let $\mathcal{L}$ and $\mathcal{M}$ be invertible sheaves on X, such that $\deg \mathcal{L} \geq 2g+1$ and $\deg \mathcal{M} \geq 2g$. Then the morphism*

$$\Gamma(\mathcal{L}) \otimes \Gamma(\mathcal{M}) \longrightarrow \Gamma(\mathcal{L} \otimes \mathcal{M})$$

is surjective.

Lemma 33. *Let k be a field, $k \subset K$ a field extension, X a variety over K and $f : X_K \to X$ the K-variety deduced from X by base-change. Let $\mathcal{F}$ be a sheaf of k-algebras over X (for example an invertible sheaf) and let $\mathcal{F}_K = f^{-1}\mathcal{F} \otimes_k K$ be its pull-back over X_K — where K is the constant sheaf over X_K. Then the morphism*

$$\Gamma(\mathcal{F}) \otimes_k K \longrightarrow \Gamma(\mathcal{F}_K)$$

is an isomorphism.

Let us first admit the lemma and prove Theorem 29. Let K be the algebraic closure of k. The property of being a proper —resp. smooth— morphism being stable by base change, the variety X_K is still proper and smooth over K. Mumford's Theorem 32, applied to the variety X_K and to the pulled-back sheafs $\mathcal{L}_K$ and $\mathcal{M}_K$, thus states that the morphism:

$$\Gamma(\mathcal{L}_K) \otimes \Gamma(\mathcal{M}_K) \longrightarrow \Gamma(\mathcal{L}_K \otimes \mathcal{M}_K)$$

is surjective. By the lemma, one can move out the $\otimes_k K$ from both from the right hand and left hand sides. The surjection therefore reads itself as:

$$\Gamma(\mathcal{L}) \otimes \Gamma(\mathcal{M}) \otimes_k K \longrightarrow \Gamma(\mathcal{L} \otimes \mathcal{M}) \otimes_k K.$$

But K being a k-vector space, it is faithfully flat, hence the theorem.

Let us now prove the lemma. Let $\rho_{UV} : \mathcal{F}(U) \to \mathcal{F}(V)$ the restriction morphisms of the sheaf $\mathcal{F}$. By definition, $\mathcal{F}_K$ is the sheaf associated to the following presheaf over X_K, whose sections over any open set U are the $\mathcal{F}(U) \otimes_k K$ and the restrictions equal to the $\rho_{UV,K} = \rho_{U,V} \otimes_k K : \mathcal{F}(U) \otimes_k K \to \mathcal{F}(V) \otimes_k K$.

Let us thus consider $\tilde{s}$ a section of $\mathcal{F}_K$. It consists in the data of a finite open covering $(U_i)_i$ of X_K, and of a collection of sections $\tilde{s_i} \in \mathcal{F}(U_i) \otimes_k K$ compatible between each other by the restriction maps $\rho_{UV,K}$. Explicitly, let U_i and U_j be two open sets. Let us abridge ρ_i and ρ_j the restriction morphisms $\rho_{U_i,\, U_i \cap U_j}$ and $\rho_{U_j,\, U_i \cap U_j}$. Let us express the sections under the form of finite sums of elementary tensors:

$$\tilde{s_i} = \sum_{p \in P} m_p^i \otimes \lambda_p^i$$

$$\tilde{s_j} = \sum_{q \in Q} m_q^j \otimes \lambda_q^j \ ,$$

where P and Q are finite sets of indices, the $(\lambda_p^i)_p$ and $(\lambda_q^i)_q$ are elements of the field K, and the m_p^i (resp. m_q^j) sections of $\mathcal{F}(U_i)$ (resp. $\mathcal{F}(U_j)$). The glueing condition for the open sets U_i and U_j, noted (ij), is

$$\text{(ij)} \qquad \big(\rho_i \otimes_k K\big)\big(\sum_{p \in P} m_p^i \otimes \lambda_p^i\big) = \big(\rho_j \otimes_k K\big)\big(\sum_{q \in Q} m_q^j \otimes \lambda_q^j\big) \ .$$

Let $k \subset L \subset K$ a finite extension of k, large enough to contain all the coefficients $(\lambda_p^i)_{\substack{i,j,k,\dots \\ p,q,r,\dots}}$ which show up in the previous expressions of all the sections $\tilde{s_i}$, $\tilde{s_j}$, $\tilde{s_k}$ etc. Let $(l_1, \dots, l_N)$ a basis of L over k and

$$\lambda_p^i = \lambda_{p,1}^i l_1 + \cdots + \lambda_{p,N}^i l_N,$$

$$\lambda_q^j = \lambda_{q,1}^j l_1 + \cdots + \lambda_{q,N}^j l_N$$

etc. The decompositions of each of these coefficients over the basis $(l_1, \cdots, l_N)$. L being a vector space over k —of dimension N—, every $\mathcal{F}(U) \otimes_k L$ is a direct sum of N copies of $\mathcal{F}(U)$ (by regrouping the components in $\cdot \otimes l_n$, $n = 1 \dots N$). Consequently, the set of glueing conditions $(ij)_{i,j}$ is satisfied iff the set of projections of these glueing conditions $(ij, n)_{i,j,n}$, over all the components in $(\cdot \otimes l_n)_{n=1 \dots N}$, is satisfied. For example, let us fix n, then the projection over $\cdot \otimes l_n$ of a glueing condition (ij) can be expressed as

$$\text{(ij,n)} \qquad \rho_i(\sum_{p \in P} m_p^i \otimes \lambda_{p,n}^i) = \rho_j(\sum_{q \in Q} m_q^j \otimes \lambda_{q,n}^j)$$

(Where we recall that the coefficients $(\lambda^i_{p,n})_p$, $(\lambda^j_{q,n})_q$ are in k). Consequently, the collection of the projected conditions $(ij,n)_{i,j}$, for a fixed n, defines a global section $s_n \in \mathcal{F}(X)$. But

$$\tilde{s} = s_1 \otimes_k l_1 + \cdots + s_N \otimes_k l_N \in \mathcal{F}(X) \otimes_k K,$$

which is what was to be proven.

E Alternative proof of Theorem 28 with only *formal* lifts of curves, and projective limits of codes

E.1 Lifting smooth projective curves and algebraic geometry codes over local rings

Let us furthermore assume from now on that R *is noetherian*. Let $S = \operatorname{Spec} R$ be the corresponding affine scheme, we call *smooth projective curve over* S an irreducible scheme X with a smooth projective morphism $f : X \to \operatorname{Spec} S$ of relative dimension one [f being flat, Lemma [Sta18, 0AFE] implies that $\dim X = \dim R + 1$. So $\dim X = 2$ (an "arithmetic surface") if R is a DVR, and $\dim X = 1$ if R is a local Artinian ring].

Let us fix $(A, \mathfrak{m}, \kappa)$ a local noetherian ring (although A is "morally" the complete ring $\mathbb{Z}_p$ or $W(\mathbb{F}_q)$, the assumptions for A are actually the same as for the "generic" local noetherian ring R considered throughout, the completeness assumption being not necessary until the appendix). Consider the projective system of local Artinian rings:

$$\ldots A_3 \to A_2 \to A_1 \to A_0 = \kappa\ ,$$

where $A_i = A/\mathfrak{m}^{i+1}$ and each arrow $A_{i+1} \to A_i$ is the reduction modulo $\overline{\mathfrak{m}^{i+1}}$. This corresponds to a direct system of affine schemes:

$$\operatorname{Spec} \kappa = S_0 \to S_1 \to S_2 \to S_3 \to \ldots\ ,$$

where each arrow $S_i \to S_{i+1}$ is a closed immersion defined by the ideal $\overline{\mathfrak{m}^{i+1}}$ of square zero. Let X_0 be any flat scheme over $\operatorname{Spec} \kappa = S_0$, then a *formal lift* $\mathfrak{X}$ of X_0 over this direct system is the data of flat schemes X_i over each S_i, fitting into an infinite diagram where all squares are cartesian:

$$\begin{array}{ccccccc} X_0 & \xrightarrow{j_0} & X_1 & \xrightarrow{j_1} & X_2 & \xrightarrow{j_2} & \cdots \\ \downarrow{\scriptstyle f_0} & & \downarrow{\scriptstyle f_1} & & \downarrow{\scriptstyle f_2} & & \\ S_0 & \longrightarrow & S_1 & \longrightarrow & S_2 & \longrightarrow & \cdots \end{array} \tag{27}$$

In particular the X_i form a direct system and, by base-change, the maps $j_i : X_i \to X_{i+1}$ are closed immersions. They are given locally on $\operatorname{Spec} B_{i+1}$ by the ideal $\mathfrak{m}^{i+1} B_{i+1}$ of square zero .

Our principal addition to [Wal99], which studies reduction of AG codes over rings, is that we notice the possibility to go in the other direction:

Theorem 34 (Formal lifts of curves [9] ***[SGA1, III Theorem 6.3] or [Ill05, proof of 5.19 (i)]).*** *Let $(A, \mathfrak{m})$ be a local ring and consider X_0 a smooth projective curve over $S_0 = \operatorname{Spec} \kappa$, then X_0 admits a formal lift $\mathfrak{X}$ over the direct system $S_0 \to S_1 \to S_2 \to \cdots$. Moreover $\mathfrak{X}$ is projective.*

Proof. Only the last point, about projectivity, is not stated in the references mentionned in the theorem. It is stated in the full FGA's existence theorem ([Ill05, Theorem 5.19 (ii)] or [SGA1, III Théorème 7.3]). But it can also be showed directly, as in the proof of [Ill05, Theorem 8.4.10], where a very ample sheaf on X_0 is lifted to each X_i by Nakayama.

Coming back to our generic (noetherian) local ring R and X a smooth projective curve over $S = \operatorname{Spec} R$, let us define an *$R$-point* of X as an S-morphism $Z : S \to X$.

Noting s the closed point of S and $Z(s)$ its image in X, one can prove that Z defines a regularly embedded subscheme of codimension one, which is contained in any sufficiently small affine neighborhood $U = \operatorname{Spec} B$ of $Z(s)$, and thus is a Cartier divisor (also noted Z) defined by:

$$\big\{(U, b), (1 \text{ outside of } Z(S))\big\}$$

where b is a suitable non-zero divisor in B. But actually for practical purposes (Main Theorem 26) we will only need the case where R is Artinian, whence Z is just a closed embedding to one closed point $Z(s)$: see [Wal99, Lemma 4.4], and Lemma 37 for the general case.

Thus in the situation of Theorem 34, the smooth morphisms $X_{i+1} \to S_{i+1}$ being in particular formally smooth, it is possible to lift any A_i-point Z_i on X_i to an A_{i+1}-point Z_{i+1} on X_{i+1} with compatibility relations. This boils down to [Wal99, Remark 4.5], see Proposition 39 below over general (local) rings. From here, d being any positive integer and $\mathcal{L}_i = [Z_i]^{\otimes d}$ (or $\mathcal{O}(d.Z_i)$) the line bundle class corresponding to the Cartier divisor $d.Z_i$, we immediatly deduce a lift $\mathcal{L}_{i+1} = [Z_{i+1}]^{\otimes d}$ of $\mathcal{L}_i$. [10] The key point is that *the line bundles surject to each other* in a compatible way with the projective system of rings. More precisely, considering affine open subsets where the line bundles become principal

[9] About references: of course the clearest is https://amathew.wordpress.com/2011/06/18/lifting-smooth-curves-to-characteristic-zero/ . The small missing point is that he actually doesn't prove how to obtain a compatible system of lifts in Corollary 9, he only shows that X_0/k lifts to X_1/A_1. The trick that makes it possible is [Ill05, Remark 5.10 (b)] (see also [SGA1, p61]), which boils down to the standard base change formula for modules of differentials: let B be an A-algebra and $A \to A'$ a morphism of rings (here B is an affine subset algebra of X_{i+1}, $A = A_{i+1}$ and $A' = A_i$), then $A' \otimes \Omega_{B/A} = \Omega_{A' \otimes B/A'}$. We also mention that smoothness criterion [Ill05, 5.8 (ii)] is false and should be replaced by [SGA1, §II 1.1 & 4.8]. Hartshorne's Deformation theory, Corollary 10.3 recovers the result by more machinery (T functors).

[10] Notice also that it is actually possible to lift any line bundle, by [Ill05, §5.2] (see also Lemma 11 of Akhil Matthews' blog), although for our purpose it is enough to lift points, as we just did.

fractional ideals, we see that for each i the identity map on $\mathcal{O}_{X_i}$ induces the isomorphism of line bundles:

$$\mathcal{O}_{X_i} \otimes_{\mathcal{O}_{X_{i+1}}} \mathcal{L}_{i+1} \to \mathcal{L}_i \tag{28}$$

from which we deduce in particular the isomorphisms for all i:

$$\mathcal{O}_{X_0} \otimes_{\mathcal{O}_{X_i}} \mathcal{L}_i \to \mathcal{L}_0 \tag{29}$$

Finally, starting from a line bundle $\mathcal{L}_0 = [Z_0^{(0)}]^{\otimes d}$ on X_0 along with n distinct points of degree one $Z_0^{(1)}, \ldots, Z_0^{(n)}$ outside of $Z_0^{(0)}$, defining a κ-linear evaluation code C_0, then we can lift this data to all X_i/S_i in a compatible way (the points and the line bundles embed/surject to each other), and obtain A_i-linear evaluation codes C_i of length n (see the explicit description of AG codes over Artinian rings at the beginning of [Wal99, §5]). What then remains to be shown is that these evaluation codes *reduce* to each other in a compatible way.

Theorem 35 (Projective systems of lifts of Riemann-Roch spaces and AG codes). *Consider the same situation as above: $\mathcal{L}_0$ any line bundle over X_0, n closed points $Z_0^{(j)}$, $j = 1 \ldots n$ on X_0 and the evaluation map γ_0 yielding an algebraic geometry code. Then this data lifts to every X_i, such that we have the following commutative diagram:*

$$\begin{array}{ccccc}
\Gamma(X_{i+1}, \mathcal{L}_{i+1}) & \twoheadrightarrow & \Gamma(X, \mathcal{L}_{i+1}) \otimes_{A_{i+1}} A_i & \xrightarrow{\sim} & \Gamma(X_i, \mathcal{L}_i) \\
\downarrow & & & & \downarrow \\
\oplus_j \Gamma(Z_{i+1}^{(j)}, \mathcal{L}_{i+1}|_{Z_{i+1}^{(j)}}) & & & & \oplus_j \Gamma(Z_i^{(j)}, \mathcal{L}_i|_{Z_i^{(j)}}) \\
\downarrow \gamma_{i+1} & & & & \downarrow \gamma_i \\
A_{i+1}^n & & \xrightarrow{\bullet \otimes_{A_{i+1}} A_i} & & A_i^n
\end{array} \tag{30}$$

Where: - the top left horizontal arrow and the bottom horizontal arrow are tensorisation by $\otimes_{A_{i+1}} A_i$ - the top right arrow is constructed canonically as in [Wal99, Lemma 4.6 & proof of Th 4.7]

- the top vertical arrows are the canonical restriction maps, - the bottom left vertical arrow arizes from choices of isomorphisms for all j:

$$\gamma_{i+1} : \Gamma(Z_{i+1}^{(j)}, \mathcal{L}|_{Z_{i+1}^{(j)}}) \to A_{i+1}$$

under the (recursive) condition that it induces the bottom right isomorphism γ_i by tensorisation by $\otimes_{A_{i+1}} A_i$.

Proof. The lifting of $\mathcal{L}_0$ and of the points follows from the discussion above the theorem.

The proof that the top right arrow is an isomorphism is mutatis mutandis the arguments in [Wal99, Lemma 4.6 & proof of Th 4.7].

Maybe could we also detail how to obtain such a lift of γ_i for the bottom left vertical arrow.

Corollary 36 (Good lifts of AG codes). *The codes C_i form a projective system of codes, more precisely we have surjections for all i:*

$$C_{i+1} \twoheadrightarrow C_{i+1} \oplus_{A_{i+1}} A_i \cong C_i \tag{31}$$

Moreover the codes C_i are all free of rank $\dim C_0$ and thus free lifts of C_0:

$$\pi(C_i) = C_0 \tag{32}$$

thus their projective limit $\hat{C} = \varprojlim C_i$ over $\widehat{A}$ is also a free lift of C_0.

Proof. The proof for (31) being the same as [Wal99, Theorem 5.5], let us describe it quickly.

The freeness and equality of ranks follows from [Wal99, Th 5.4], thus they are free lifts by definition, whence (32) (this is exactly the argument of [Wal99, Th 5.7]).

For the last assertion, the projective limit being an additive functor, it preserves direct summands so sends free lifts to free lifts. □

F Realizing the projective limit of codes as an AG code, thanks to the existence theorem

The following lemma states that [Wal99, Lemma 4.4] also holds over any local ring R, and that the situation is equally explicit.

Lemma 37. *An R-point is a regular immersion of codimension one. There exists a unique, well defined Cartier divisor (which we will also denote by Z) associated to Z. Furthermore let s be the closed point of S and $Z(s)$ be its (closed) image in X, then Z factors through a closed immersion in $\operatorname{Spec} \mathcal{O}_{Z(s)}$ followed by the open immersion in X. Thus there exists an affine neighborhood $U = \operatorname{Spec} B$ of $Z(s)$ and a regular element $b \in B$ such that $Z = \big\{(U, b), (1 \text{ outside of } Z(S))\big\}$.*

Proof. We firstly prove that the image of Z is contained in any (affine) neighborhood of $Z(s)$. Let $\operatorname{Spec} B$ be any affine neighborhood of the image $Z(s)$ in X, then $Z^{-1}(\operatorname{Spec} B)$ is an open subset of S containing s so is the whole S .

Let us now show that Z defines a closed immersion in $\operatorname{Spec} B$, which implies in particular that the image $Z(s)$ is a closed point. Let us restrict to $\operatorname{Spec} B$ the structural morphism $f : X \to S$, we now have the corresponding morphisms of rings $R \xrightarrow{f^\sharp} B \xrightarrow{Z^\sharp} R$ which by assumption compose to the identity of R. Thus in particular $Z^\sharp : B \to R$ is surjective.

Let us finally show the Cartier divisor description of Z. Z being an immersion, it is furthermore regular by [SGA1, II Corollaire 4.16]. In particular its ideal $I_{Z(s)} \subset \mathcal{O}_{Z(s)}$ is generated by a regular sequence. Let us remind why the codimension d —i.e. the size of this regular sequence— is one. The local ring $\mathcal{O}_{Z(s)}$ being noetherian, we have:

$$\dim R = \dim \mathcal{O}_{Z(s)}/I_{Z(s)} = \dim B - d = \dim R + 1 - d \,,$$

where the second equality follows from [Sta18, 00KW] (see also [Liu], theorem 2.5.15) . All the other closed points of X being outside of $Z(S)$, Z is defined by 1 there. Thus by [Sta18, 00NX (5)] the sheaf of ideals of Z is locally free of rank one. The claimed description of Z follows by choosing a sufficiently small open affine neighborhood $U = \operatorname{Spec} B$ of $Z(s)$ and such that a regular generator b_B of $I_{Z(s)} \subset \mathcal{O}_{Z(s)}$ is a regular element of B. □

Theorem 38 (the existence theorem [Ill05, Theorem 5.19 (ii)] or [SGA1, III Corollaire 7.4]). *Under the assumptions of Theorem 34, if furthermore $A = \widehat{A}$ is* complete *(e.g. $\mathbb{Z}_p$ or more generally $W(\mathbb{F}_q)$), there exists a smooth projective curve X over $S = \operatorname{Spec} \widehat{A}$ that lifts X_0/S_0.*

Proposition 39 (Lifts of points). *Under the assumptions of Theorem 34, let $Z_0 : A_0 \to X_0$ be a A_0-point of X_0, then there exists a compatible direct system of A_i-points of X_i lifting Z_0. Namely we have a family of A_i-points $(Z_i)_i$ such that, noting j_i the closed immersion given by Theorem 34, then the following diagrams commute:*

(33)
$$\begin{array}{ccc} X_i & \xrightarrow{j_i} & X_{i+1} \\ {\scriptstyle Z_i}\uparrow & & \uparrow{\scriptstyle Z_{i+1}} \\ S_i & \longrightarrow & S_{i+1} \end{array}$$

Proof. By induction, let us deduce Z_{i+1} assuming the existence of Z_i. Consider the composite map:

$$g_i : S_i \xrightarrow{Z_i} X_i \xrightarrow{j_i} X_{i+1} ,$$

which by Theorem 34 fits into the following commutative diagram:

(34)
$$\begin{array}{ccccc} & & & & X_{i+1} \\ & & {\scriptstyle g_i} \nearrow & {\scriptstyle ?}\nearrow & \downarrow \\ S_i & \longrightarrow & S_{i+1} & \xrightarrow{\mathrm{id}} & S_{i+1} \end{array}$$

The vertical arrow being smooth and the bottom left arrow being a closed immersion of Artinian local rings defined by an ideal of square zero, [SGA1, III Th 3.1 (iii)] provides the existence of a dotted arrow Z_{i+1}, which is indeed a R_{i+1}-point making (33) commute. □

Under the assumptions of Theorem 38 we can also lift S_0 points of X_0 to S-points of X, this time as a consequence of [SGA1, III Th 3.1 (ii)]. Indeed as noticed in the proof of Lemma 37, any affine neighborhood of the closed point of S is actually the whole S.

Lifting n on X_0 points and a line bundle (of the form $O_{X_0}(dZ_0)$), we obtain an AG code C on X the smooth projective curve of 38. One can see that C surjects in a compatible way to the projective system of Corollary 36.

Remark 4. One can also show directly that C is a free lift of C_0. Indeed, we need only show the saturation criterion of Proposition 25 (i): if a codeword w in C is a multiple of p : $w = pw_1$ then w_1 is also a codeword of C . To prove this, use that all local rings in X are UFD (because X is smooth over the regular local ring R).

Question 40. So it would be very nice to find a counterexample of code C over a non smooth curve over $\mathbb{Z}_p$ (or Witt), such that C is not saturated (= the criterion that we just checked).

G Magma program for multiplicative lifts, and examples on AG codes

```
reset
magma
ZZ:=Integers();
p:=2;

F2:=FiniteField(p);

/*
//===== Hermitian curve over F16 ========== // n+1=65
    places, genus 6, 3t<n-24=40 =>  t=13 ok => degree(D)
    =2*6+13=25
r:=4;
F<dt> := FiniteField(p^r);
q:=p^(ZZ!(r/2));
A<x,T> := AffineSpace(F,2); //doesn't like function
    fields when asking for an uniformizing parameter at a
    place
f:=T^q+T-x^(q+1);
K1:=Curve(A,f);
m:=25; //condition for projective normality: m=degree(D)
    >=2g+1 . Also, t=m-(2g+1)<(n-4g)/3
pole:=true;
Fixed:=[ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
    15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27,
    28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
    41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53,
    54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66,
    67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
    80, 81, 82, 83, 84, 85, 89, 90, 91, 92, 93, 94, 95,
    96, 97, 101, 102, 103, 104, 105, 106, 107, 108, 112,
    113, 114, 115, 116, 117, 118, 122, 123, 124, 125, 126,
```

```
    127, 131, 132, 133, 134 ]; // the indices pruned from
    the _Reduc variables_ , not from the _equations_ (
    because not basis vectors of C2)
*/
/*
//========== Genus 3 over F32 with record 64 points
    ========== // ATTENTION: first basis element of L(D
    ) is no more 1 (but fixing it also works, for a reason
    ..)
r:=5;
F<dt> := ext<F2|Polynomial([1,0,1,0,0,1])>;
A<x,y> := AffineSpace(F,2);
X:=x^2+x; Y:=y^2+y;
f:=X^2 + X*Y + dt^3*Y^2 + Y + dt^26;
K1:=Curve(A,f);
m:= 22; //equal 2g+t. with t=17 , such that 3*t<(N-1)-4g
    =51 -> t< 17 (so no gain to evaluate also at P0..)
pole:=false;
Fixed:=[ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
    15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27,
    28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
    41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53,
    54, 64, 65, 66, 78 ];
*/

//Elliptic curve over F2^3 with 14 places
r:=3;
F<dt> := FiniteField(p^r);
A<x,y> := AffineSpace(F,2);
K1 := Curve(A,y^2+x*y+y-x^3+1);
m:=4; // 2g+t avec t=2 , > 2g+2 = et t=2, verifie bien
    6=3t<(n-1)-4g=9
pole:=false;
Fixed:=[];

g:=Genus(K1); g; //
P:=Places(K1,1); #P; //
if pole eq true then n:=#P;
else n:=#P-1; end if;
D:=m*P[1];

Rlist:=[];
for k in [1..101] do
```

```
  Append(~Rlist, GaloisRing(p^(k+1), Polynomial([ZZ!
      Coefficient(MinimalPolynomial(dt,F2),i) : i in [0..r
      ]]) ) );
end for;
R4<Dt>:=Rlist[1];  // $.1^4 + $.1 + 1 for dt  //
    ResidueField(R4) eq F; //false: doesn't identify the
    two
R8:=Rlist[2];
ElementToSequence(R4!Dt);



//the evaluation code of L(D) and L(2D) over (P1 if pole)
     P2,...,P_(n+1)
AGCodes:=function(D,P, pole);
  E:=Basis(D);
  dimL:=#E; //m+1-Genus
  D2:=2*D;
  E2:=Basis(D2);
  dimL2:=#E2;
  V,phi:=RiemannRochSpace(D); //(phi^-1)(E[1]); //gives
      the decomposition of the section E[1] in the basis E
       of V:  V: (      1       0       0)
  if pole eq true then
    tpole:=UniformizingParameter(P[1]);
    tvp:=tpole^m;
    EvD := Matrix(F,dimL, n, [<i,j,Evaluate(E[i],P[j])>:
        j in [2..n], i in [1..dimL]]);
    EvD2 := Matrix(F,dimL2, n, [<i,j,Evaluate(E2[i],P[j])
        >: j in [2..n], i in [1..dimL2]]);
    for i in [1..dimL] do
      EvD[i,1]:=Evaluate(E[i]*tvp,P[1]);
    end for;
    for i in [1..dimL2] do
      EvD2[i,1]:=Evaluate(E2[i]*tvp^2,P[1]);
    end for;
  else
    EvD := Matrix(F,dimL, n, [<i,j,Evaluate(E[i],P[j+1])
        >: j in [1..n], i in [1..dimL]]);
    EvD2 := Matrix(F,dimL2, n, [<i,j,Evaluate(E2[i],P[j
        +1])>: j in [1..n], i in [1..dimL2]]);
  end if;
  return EvD, EvD2, dimL,dimL2;
end function;
```

```
EvD, EvD2, dimL,dimL2:=AGCodes(D,P,pole);


C:=RowSpace(EvD); //the code of L(D)
dimC:=Dimension(C); //of dimension dimL(D), as expected (
    here: 3)
CD2:=RowSpace(EvD2); // the code of L(D^2)


//the square of C: dimension checks.
square:=ZZ!(dimC*(dimC+1)/2);
RowsC2:=[];
for k in [1..dimL] do
  for l in [k..dimL] do
    Append(~RowsC2, [EvD[k][j]*EvD[l][j] : j in [1..n]]
        );
  end for;
end for;

MatC2:=Matrix(F,RowsC2);
C2:=RowSpace(MatC2);
dimC2:=Dimension(C2);
nbEqVec:=square-dimC2; //square; //square-dimC2; //number
     of vectorial equations

//extracting a basis of C2
Bx:=[MatC2[1]]; //
Ix:=[[1,1,1]]; //keeping track of the indexes - and of
    the ij indexes
C2:=VectorSpaceWithBasis(Bx);
ij:=0;
for i in [1..dimC] do
  for j in [i..dimC] do
    ij+:=1;
    cij:=MatC2[ij];
    if (cij notin C2) then
      Append(~Bx,cij);
      Append(~Ix,[i,j,ij]);
      C2:=VectorSpaceWithBasis(Bx); //f will kill me
    end if;
  end for;
end for;
```

```
//decomposition of vectors of C2 on this basis: binary
    only decompositions! And how sparse !!!
Reduc:=[];
ijsquare:=0;
b:=1;
for i in [1..dimC] do
  for j in [i..dimC] do
    ijsquare+:=1;
    if (i ne Ix[b][1]) or (j ne Ix[b][2]) then
      Append(~Reduc, Coordinates(C2, C2!RowsC2[ijsquare])
          );
    else
      if b lt dimC2 then
        b+:=1;
      end if;
    end if;
  end for;
end for;


[i: i in [1..#Reduc] |#[j: j in [1..dimC2] | Reduc[i][j]
    ne 0] eq 1]; //outputs Fixed


//lift of C
liftGal:=function(a,R);
  pol:=ElementToSequence(F!a, F2); //
  polZ:=[ZZ!coeff : coeff in pol];
  return R!( &+[polZ[i]*R.1^(i-1): i in [1..r] ] );
end function;
//liftGal(Dt^5,R4);


RowsCR4:=[[liftGal(EvD[i][j],R4) : j in [1..n]]: i in
    [1..dimC] ] ;


// Square of the lift of C, with basis put aside
CRSquared:=function(CR, Rlift);
  CR2:=[]; BxR:=[]; b:=1;
  for i in [1..dimL] do
    for j in [i..dimL] do
      cij:=  [CR[i][crd]*CR[j][crd] : crd in [1..n]]  ;
      if Ix[b][1] ne i or Ix[b][2] ne j then
        Append(~CR2, cij  );
```

```
        else
          Append(~BxR, Vector(cij) );
          if b lt dimC2 then
            b+:=1;
          end if;
        end if;
      end for;
    end for;
    MCR2:=Matrix(Rlift,CR2);
    return MCR2,BxR;
end function;
MC24,Bx4:=CRSquared(RowsCR4,R4);


Reduc4:=[Vector([liftGal(Reduc[i][j],R4) : j in [1..dimC2
    ]] ) : i in [1..nbEqVec]];


// Dividing by p^k
dividek:=function(b,R,k);
    pol:=ElementToSequence(R!b);
    polk:=[F2!((ZZ!coeff)/2^(k)) : coeff in pol];
    return F!SequenceToElement(polk,F);
end function;
dividek(4*(R8.1^2+R8.1^3),R8,2); //SequenceToElement([F2
    |0,0,1,1],F); // returns dt^6: he converts
    automatically to the (shorter) primitive element power
     !


//list of error terms divided by p^k for basis
    reconstruction of C2 (minus basis) in Z/p^(k+1)Z[X]/P.
errork:=function(MCR2,BxR,ReducR,Rlift,k)
    pDR:=[];
    for ij in [1..nbEqVec] do
      Append(~pDR,MCR2[ij]-&+[ReducR[ij][l]*BxR[l] : l in
          [1..dimC2]]);
    end for;
    DF:=[ Vector([dividek(pDR[ij][j],Rlift,k): j in [1..n
        ]]) : ij in [1..nbEqVec]]; //pD4[50]; DF[50];
        division seems ok
    return DF;
end function;
DF:=errork(MC24,Bx4,Reduc4,R4,1);
```

```
// ======= Multiplicative lift modulo 4 : solve V.A = W
    in vector V => constructing transp(A)
//transp(W)= DF= transp(A).transp(V) , which expresses on
     each of the n Big blocks of square-dimC2 rows the
    vectorial equations: -(ei.fj+ej.fi) + sum_base( mu_(ij
    ,b) (e_bi.e_bj) + Reduc_(ij,b) (e_bi.f_j + e_bj.fi) )
// V: firstly the lifts fi:  n Big blocks of size dimC (
    lifts of C) for each fixed coord crd. Then, (square-
    dimC2) Big blocks (coefficients mu_ij for ij fixed) of
     size dimC2 (coordinates on (e_bi.e_bj)_b for b in IxM
    )
// DF: as for A, n Big blocks (one per coordinate) of
    size square

nbVecInc:=dimC-1;// dimC-1;
tA:=ZeroMatrix(F,nbEqVec*n,nbVecInc*n+dimC2*(nbEqVec-#
    Fixed) ); // (rows: n coords times square vector
    equations, cols: n blocs of nbVecInc-1 coordinates
    plus dimC2 coeffs Reduc_ij for each couple ij)
DFvec:=Vector([F!0: j in [1..nbEqVec*n]]);
offset:=nbVecInc*n; // Place taken at the beginning of V
    by the lifts of C


populateRow:=procedure(~M,~D,i,j,ijrowA,ijcolA,crd);
//-------------POPULATE the invCrd coordinate differently
  row:=(crd-1)*nbEqVec+ijrowA ; //the line of A and D to
      be changed: inside big block number crd-1 , go fetch
       couple (i,j) number ij
  idmuij:=offset+dimC2*(ijcolA-1); //skip offset, and
      fetch the ij-th big block for coefficients mu(ij,l),
       l varying in dimC2.
  idfj:=(crd-1)*nbVecInc+j; //go in block crd and look
      for index of fj
  idfi:=(crd-1)*nbVecInc+i; //go in block crd and look
      for index of fi
  if j ne 1 then // should remove the 1st column in each
      of the n first blocks instead
    M[row,idfj]:=-EvD[i][crd];
  end if;
  if i ne 1 then
    M[row,idfi]:=M[row,idfi]-EvD[j][crd];
  end if;
  D[row]:=DF[ijrowA][crd];
```

```
  for b in [1..dimC2] do
    bi,bj:=Explode(Ix[b]); //the indexes i,j of the b-th
        basis vector ei.ej of C^2
    idbj:=(crd-1)*nbVecInc+bj;
    idbi:=(crd-1)*nbVecInc+bi;
    if bj ne 1 then
      M[row,idbj]:= M[row,idbj]+ Reduc[ijrowA][b]*EvD[bi
          ][crd] ;
    end if;
    if bi ne 1 then
      M[row,idbi]:= M[row,idbi]+ Reduc[ijrowA][b]*EvD[bj
          ][crd] ;
    end if;
    if ijrowA notin Fixed then
      M[row,idmuij+b]:= M[row,idmuij+b]+ Bx[b][crd]; //
    end if;
  end for;
end procedure;


ijrowA:=0;
ijcolA:=0;
b:=1;
for i in [1..dimC] do
  for j in [i..dimC] do
    if (i ne Ix[b][1]) or (j ne Ix[b][2]) then
      ijrowA+:=1;
      if ijrowA notin Fixed then
        ijcolA+:=1;
      end if;
      for crd in [1..n] do
        populateRow(~tA,~DFvec,i,j,ijrowA,ijcolA,crd);
      end for;
    else
      if b lt dimC2 then
        b:=b+1;
      end if;
    end if;
  end for;
end for;


//Ker:=NullspaceOfTranspose(tA);
timestart:=Cputime();
Sol,Ker:=Solution(Transpose(tA),DFvec);
```

```
print Cputime(timestart);
procedure printInfo()
print "dimC ", dimC, "equals dimL ",dimL, " : ",dimC eq
    dimL," -- length ",n;
print "dimC2 ", dimC2, " vs total number of unordered
    couples (i,j): ", square;
print "Mumford's normal generation -- ",dimC2 eq Dimension
    (CD2);
//print "indexes (k,l) of the extracted basis (ek*el) of
    C2 ", Ix;
//print "Decomposition of each nonbasis element ei*ej on
    the previous basis (ek*el) of C2 ", Reduc[[ij : ij in
     [1..square] | ij notin [Ix[j][3]: j in [1..dimC2] ]
    ]];
//print "Error terms, after lifting of this decomposition
    , for each (i,j)", pD4;
//print "Indexes (i,j) with nonzero error term after
    lifting of this decomposition ",[i: i in [1..square] |
      not IsZero(pD4[i])];
print "Size of the system: #rows of tA ",Nrows(tA),"vs #
    columns of tA ", Ncols(tA) ;
//print "lignes matrice A non nulles ",[i: i in [1..
    square*n] |  not IsZero(tA[i])];
//print "indices ij ou A non nulle ",[i: i in [1..square]
     | &or[not IsZero(tA[i+k*square]) : k in [0..n-1]] ];
/*print "RowsCR ",RowsCR;
print "RowsC2R ",RowsC2R;
print "RowsCR2 ", RowsCR2; */
//print "==== SOLUTION: ====";
//print "Corrective terms of the lifts of the code + of
    the lifts of linear relations of C2, that cancel the
    previous error terms ",Solution(Transpose(tA),DFvec);
print "dimKer ", Dimension(Ker);
end procedure;
printInfo();


//
    ============================================================

//========== Reconstructing the corrected lift from
    the solution of the linear system ===
//
    ============================================================
```

```
unfoldVects:=function(sol);
  fiVects:=[Vector(F,[sol[(crd-1)*nbVecInc+1] : crd in
      [1..n] ])];
  for i in [2..dimC] do
    Append(~fiVects,Vector(F,[sol[(crd-1)*nbVecInc+i]:
        crd in [1..n] ]) );
  end for;
  return fiVects;
end function;


unfoldReduc:=function(sol); //unfolds corrections to the
    coefficients (=stored in "Reduc")
  muVects:=[];
  ijcolA:=0;
  for rowA in [1..nbEqVec] do
    if rowA notin Fixed then
      ijcolA+:=1;
      idmuij:=offset+dimC2*(ijcolA-1);
      Append(~muVects,Vector(F,[sol[idmuij+b]: b in [1..
          dimC2] ]));
    else;
      Append(~muVects,Vector([F!0: b in [1..dimC2]]));
    end if;
  end for;
  return muVects;
end function;


liftGalk:=function(a,R,Rlift);
  pol:=ElementToSequence(R!a); //! coeff constant a
      gauche. Par contre juste Polynomial(dt^3); renvoie
      renvoie un poly de degre 2, sur F4
      ElementToSequence(dt^5, F2);
  polZ:=[ZZ!coeff : coeff in pol];
  return Rlift!( &+[polZ[i]*(Rlift.1)^(i-1): i in [1..r]
      ] );
end function;
//liftGalk(Dt^5,R4,R8);


liftCorrectC:=function(rowsC,unfoldVects,R,Rlift,k);
  CorrectC:=[Vector(Rlift, [liftGalk(rowsC[1][j],R,Rlift)
      : j in [1..n] ])];
```

```
  for i in [2..dimC] do
    Append(~CorrectC, Vector( [liftGalk(rowsC[i][j],R,
        Rlift) + p^k*liftGal(unfoldVects[i][j],Rlift ) :
        j in [1..n] ] ) );
  end for;
  return CorrectC;
end function;


liftCorrectReduc:=function(ReducR,unfoldReduc,R,Rlift,k);
  ReducRlift:=[];
  for row in [1..nbEqVec] do
    Append(~ReducRlift, Vector( [liftGalk(ReducR[row][b
        ],R,Rlift) + p^k*liftGal(unfoldReduc[row][b],Rlift
        ) : b in [1..dimC2] ] ) );
  end for;
return ReducRlift;
end function;


//Create the total error vector from the list of error
    vectors
populateD:=procedure(~D,Dlist,ijrowA,crd);
  row:=(crd-1)*nbEqVec+ijrowA ; //the line number of D to
      be changed: inside big block number crd-1 , go
      fetch couple (i,j) number ij
  D[row]:=Dlist[ijrowA][crd];
end procedure;


//
    ============================================================

//========== Recursively searching for higher lifts
    modulo higher p^\ell ==========
//
    ============================================================


findSolk:=function(solR,RowsCR,ReducR,R,Rlift,k);//,time)
    ;
  for round in [1] do
    print round;
    solRand:=solR; //+Random(Ker);
```

```
    CRlift:=liftCorrectC(RowsCR,unfoldVects(solRand),R,
        Rlift,k);
    ReducRlift:=liftCorrectReduc(ReducR,unfoldReduc(
        solRand), R, Rlift,k);
    MCRlift2,BxRlift:=CRSquared(CRlift,Rlift);
    Dk:=errork(MCRlift2,BxRlift,ReducRlift,Rlift,k+1);

    Dkvec:=Vector([F!0: j in [1..nbEqVec*n]]);
    for ijrowA in [1..nbEqVec] do
      for crd in [1..n] do
        populateD(~Dkvec,Dk,ijrowA,crd);
      end for;
    end for;

    if round in [2000*i: i in [1..2^9]] then
      print round;
    end if;

    if IsConsistent(Transpose(tA),Dkvec) then
      solk:=solRand;
      timestart:=Cputime();
      sollift:=Solution(Transpose(tA),Dkvec);
      print Cputime(timestart); //timeend:=Cputime(
          timestart);
      //timelift:=time+timeend;
      print "found at step ",k;//,sol4,"errorVec ",sol8;
      break;
    end if;
  end for;
  return sollift,CRlift, ReducRlift;//,timelift;
end function;


solR:=Sol; ReducR:=Reduc4; RowsCR:=RowsCR4; R:=R4; Rlift
   :=Rlist[2]; //time:=0;
for k in [1..99] do
  sollift,CRlift, ReducRlift:=findSolk(solR,RowsCR,ReducR
      ,R,Rlift,k);//,timelift:=      ,time);
  solR:=sollift;
  RowsCR:=CRlift;
  ReducR:=ReducRlift;
  R:=Rlift;
  Rlift:=Rlist[k+2];
  //time:=timelift;
end for;
```

```
//=========================================
//========== for the toy example ==========
//=========================================

CorrectC:=[Vector(RowsCR4[1])];
  for i in [2..dimC] do
    Append(~CorrectC, Vector( [RowsCR4[i][j] + p*liftGal
        (unfoldVects(Sol)[i][j],R4) : j in [1..n] ] )
        );
  end for;
CorrectC;

CorrectReduc:=[];
  for row in [1..nbEqVec] do
    Append(~CorrectReduc, Vector( [Reduc4[row][b] + p*
        liftGal(unfoldReduc(Sol)[row][b],R4) : b in [1..
        dimC2] ] ) );
  end for;

cpp:=function(vec1,vec2);
return Vector([ vec1[i]*vec2[i] : i in [1..n] ]);
end function;

[&+[ cpp(CorrectC[Ix[b][1]], CorrectC[Ix[b][2]])*
    CorrectReduc[i][b] : b in [1..dimC2] ]: i in [1,2]];
cpp(CorrectC[2], CorrectC[2]); cpp(CorrectC[4],
    CorrectC[4]); //The two rows of Reduc (2 = square-
    dimC2)
```