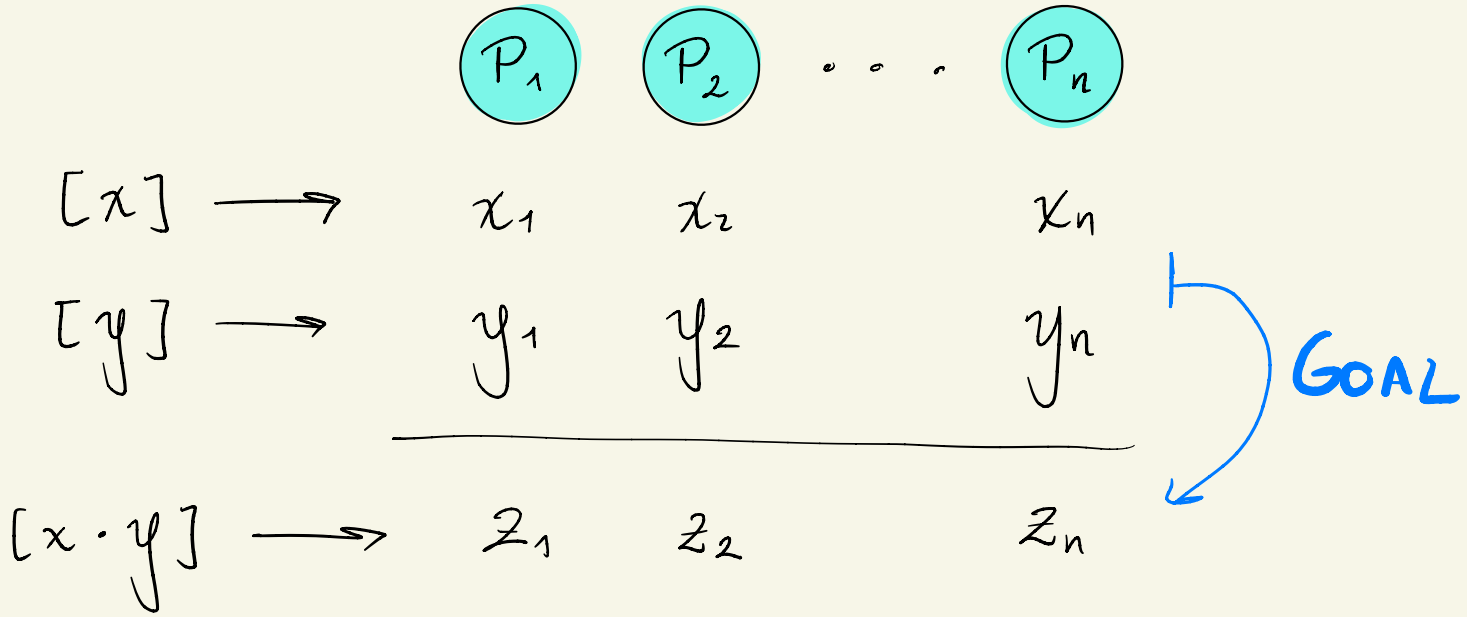
Improved Single-Round Secure Multiplication Using Regenerating Codes

Daniel Escudero – JP Morgan AI Research

* Mark Abspoel
* Ronald Cramer
* Ivan Damgård
* Chaoping Xing

Secure Multiplication



(Setting: $n = 2t+1$, passive/active security with abort)

Examples

BGW / GRR

$$\text{Comm} \approx n^2$$
$$\#\text{Rounds} = 1$$

① $[x]_t = (x_1 \cdots x_n)$

$[y]_t = (y_1 \cdots y_n)$ *

$[x \cdot y]_{2t} = (x_1 y_1 \cdots x_n y_n)$

② Each party P_i distributes shares $[x_i y_i]_t$

③ The parties compute locally $\sum_{i=1}^{2t+1} \lambda_i [x_i y_i]_t = [x \cdot y]_t$

DN07

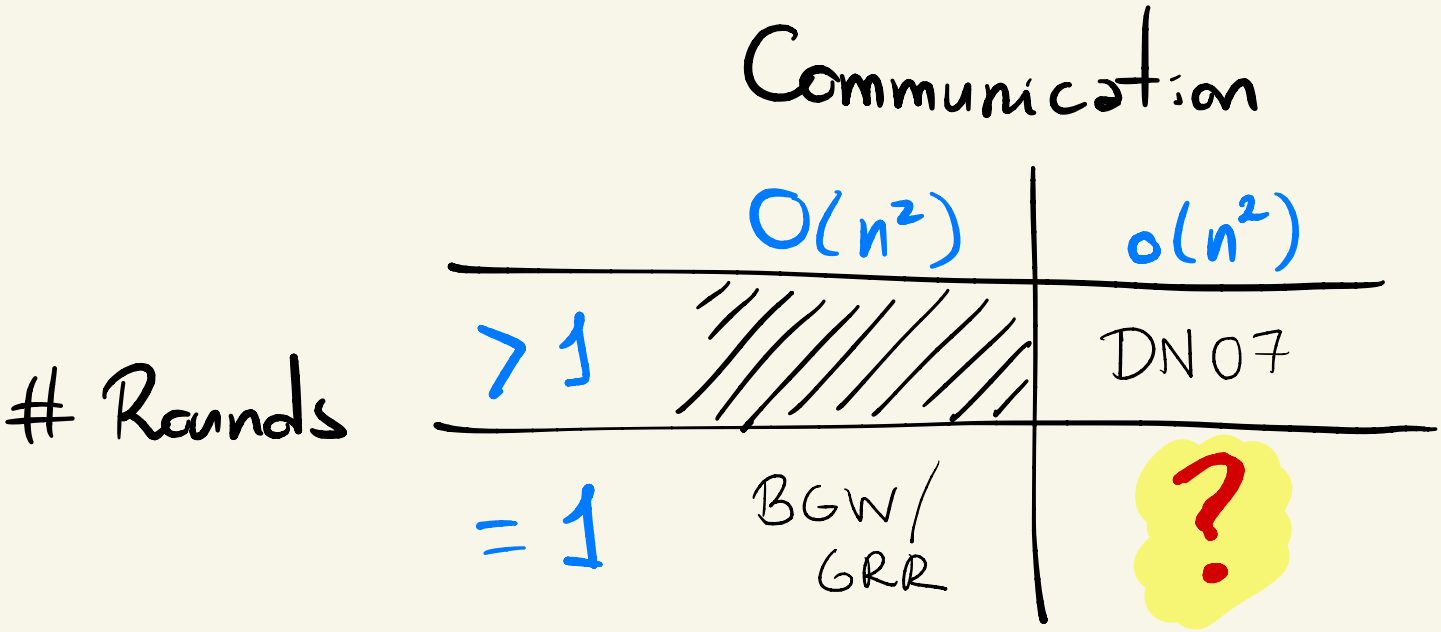
(Assume preprocessed $[r]_t, [r]_{2t}$)

Comm $\approx n$
#Rounds = 2

1) $[x]_t = (x_1 \cdots x_n)$
$[y]_t = (y_1 \cdots y_n)$ *
$[x \cdot y]_{2t} = (x_1 y_1 \cdots x_n y_n)$

2) Locally compute $[x \cdot y]_{2t} - [r]_{2t} = [e]_{2t}$, and each party P_i sends its share e_i to P_1

3) P_1 reconstructs e, and sends this to all parties, who locally compute $[x \cdot y]_t = e + [r]_t$



Can there exist one-round secure multiplication protocols with sub-quadratic communication complexity?

→ Possibly with preprocessing

Not if openings are required

↳ Each party must hear from at least $t+1$ parties

In general? NOT KNOWN!!

Why is this useful

* Minimizing the number of rounds is highly relevant in high-latency scenarios

* One-round secure multiplication is useful in several constructions (e.g. Fluid MPC)

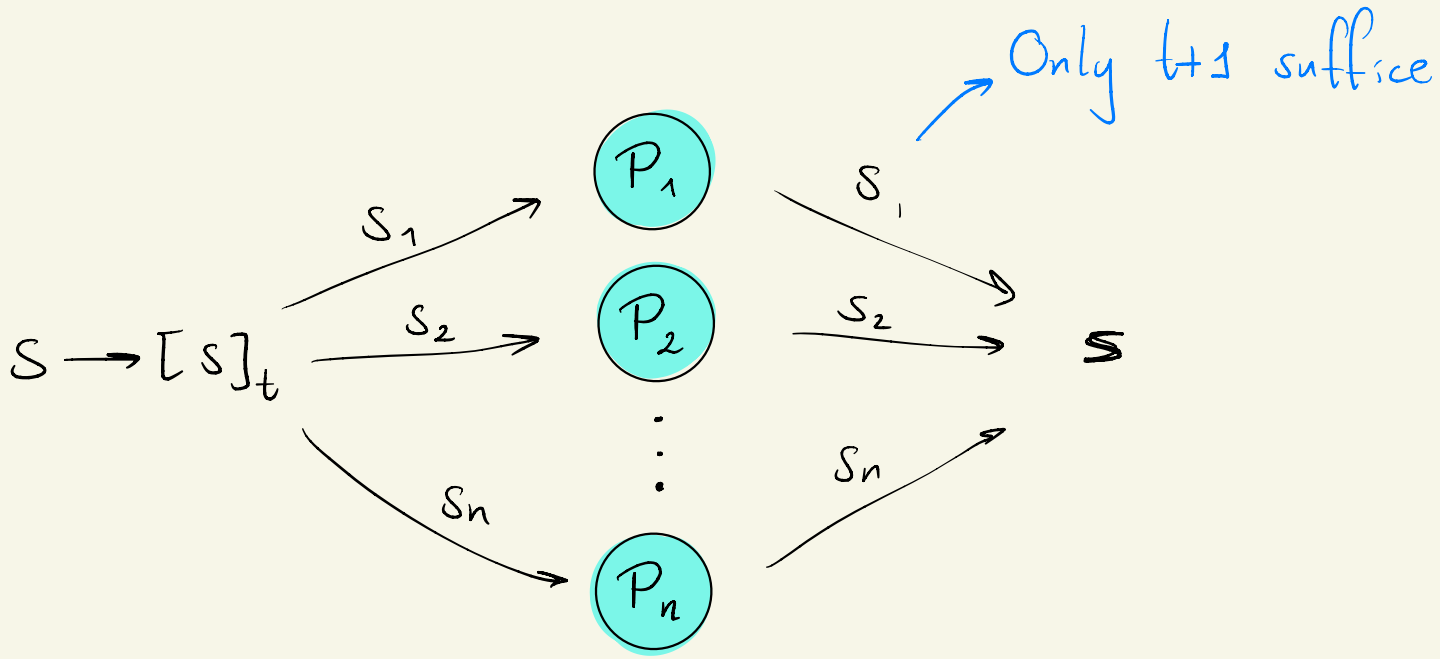
  ↳ Hence, it is useful to study the minimum communication required for 1-round multiplication

This Work

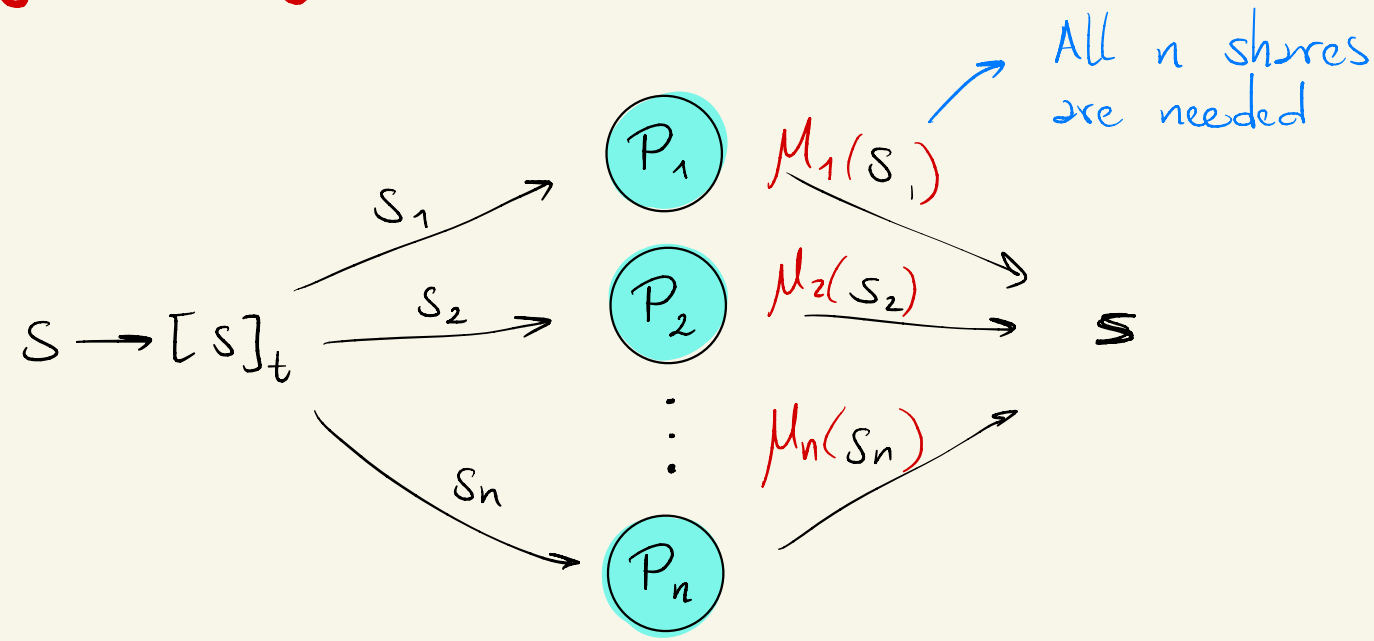
We make use of regenerating codes in order to design a 1-round secure multiplication protocol for many multiplications, having sub-quadratic communication complexity per product

* More concretely, we present an MPC protocol with the following characteristics
  - → Active security
  - → $t<n/2$ (maximal honest majority)
  - → Evaluates $\approx \log(n)$ copies of a d-layer circuit with:
    * $d + O(1)$ rounds
    * $o(n^2)$ communication per instance

First application of regenerating codes to MPC



Regenerating Codes



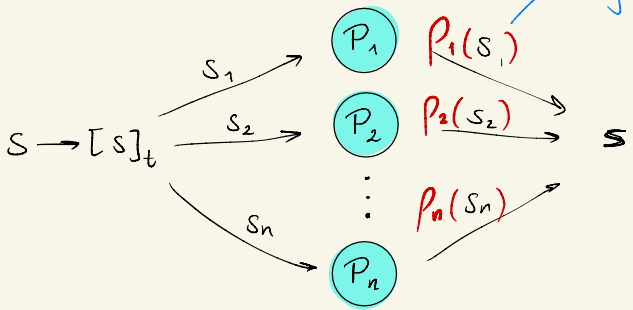
Secret / Share space $\mathbb{F}_{p^m}$, with $t + p^{m-1} < n \Rightarrow m = \Omega(\log(n))$

Compression function $\mu_i : \mathbb{F}_{p^m} \to \mathbb{F}_p$

How can they be useful for MPC?

Main limitation Requires field $\mathbb{F}_{p^m}$ with $m = \Omega(\log n)$

* Most computations tasks happen over a structure $\mathbb{F}_{p^c}$ of fixed size (in terms of n)
* Solution: embed this structure $\mathbb{F}_{p^c}$ into $\mathbb{F}_{p^m}$, and use regenerating codes to avoid the overhead in m
* Downside: This can already be avoided without the need of regenerating codes:



$$\rho_i(x) = \pi(\lambda_i \cdot x)$$

Projection into $\mathbb{F}_{p^c}$ — Lagrange coefficient

$$\Rightarrow \sum_{i=1}^{t+1} \rho_i(s_i) = \pi\left(\sum_{i=1}^{t+1} \lambda_i \cdot s_i\right) = \pi(s) = s$$

Observation: Unlike the previous compressing functions ρ_i, regenerating codes enable reconstruction of the "full" s (in contrast to $\pi(s)$)

We use regenerating codes to optimize secure multiplication over $\mathbb{F}_q^m$, and we use reverse multiplication-friendly embeddings (RMFEs) to turn this into $\approx m$ multiplications over $\mathbb{F}_q$.

Challenges

Recall that we wish to keep the number of rounds equal to **1**

* The use of RMFEs one additional round to "re-encode"

  We introduce a novel "encoding" method that removes the need of having an extra round

* Broadcast is needed in every round to ensure consistency (else, a selective-failure attack is possible)

  We make a novel use of function-dependent preprocessing and show that this alleviates this issue, allowing the consistency check to be pushed to the end

Contributions to the theory of Regenerating Codes

* We provide a novel characterization of regenerating codes in terms of certain properties of the dual code
* We generalize the theory above to the case of Galois Rings, and present our protocol in this setting
  ↳ This in particular includes the case $\mathbb{Z}/2^k\mathbb{Z}$

Roadmap

* MPC protocol for computation over Galois rings

* Characterization and existence of regenerating codes over Galois rings

Regenerating Codes over Galois Rings

Let $S = GR(p^k, l)$ and $R = GR(p^k, m \cdot l)$

↳ R is an extension of S of degree m

Let $C \subseteq R^{n+2}$ be an R-submodule

Galois Ring

$GR(p^k, l)$: polynomials over $\mathbb{Z}_{p^k}$ modulo some irreducible polynomial of degree l

Definition (Linear repair)

C has linear repair over S if there exist S-linear maps $\phi_i : R \to S$ and scalars $z_i \in R$ for $i = 1, \ldots, n$ such that, for every $x \in C$:

$$x_0 = \sum_{i=1}^{n} \phi_i(x_i) \cdot z_i$$

One of our results:

There exist a repairing/regenerating code of R over S, assuming $p^{\ell \cdot (m-1)} \leq n - t$

We can naturally use this as a secret-sharing scheme

The repairing property enables efficient 1-round reconstruction:

If $[x] = (x_1, \ldots x_n)$, then

① Each party P_i sends $\phi_i(x_i)$ ($\in S$) to all parties

② The parties compute $x = \sum_{i=1}^{n} \phi_i(x_i) \cdot z_i$

Beaver-Based Multiplication

Given $[x]$ and $[y]$, compute $[x \cdot y]$:

0. (Preprocessing) get a triple $([a], [b], [c = a \cdot b])$
1. Reconstruct $d \leftarrow [x] - [a]$ and $e \leftarrow [y] - [b]$
2. Compute locally $[x \cdot y] \leftarrow d \cdot [b] + e \cdot [a] + [c] + d \cdot e$

Communication costs: $\begin{cases} O(n^2) \text{ elements of } S, \text{ or} \\ O(n^2/m) \text{ elements of } R \end{cases}$

$\approx \log(n)$

→ Sub n^2 communication in one round!!

PROBLEM: We want MPC over S, not over R.

* Since S can be embedded in R, we can use MPC over R to get MPC over S
  ↳ But efficiency benefits are lost! ($\approx n^2$ elements of S)

Not a new problem!

EXAMPLE: Shamir secret-sharing does not work over $\mathbb{F}_2$, so a field extension $\mathbb{F}_{2^m}$ ($m \approx \log(n)$) is typically used instead

* Traditionally, the embedding $\mathbb{F}_2 \hookrightarrow \mathbb{F}_{2^m}$ is used to enable MPC over $\mathbb{F}_2$ from MPC over $\mathbb{F}_{2^m}$
* At first glance, this is wasteful!

Solution from [CCXY18]

Intuition: $\mathbb{F}_{2^m} \cong (\mathbb{F}_2)^m$, so we can hope to use MPC over $\mathbb{F}_{2^m}$ to evaluate m circuits over $\mathbb{F}_2$

Problem: This is a vector isomorphism, but we need a "RING homomorphism"

Solution: Reverse Multiplication Friendly Embeddings (RMFEs): $\mathbb{F}_2$-homomorphism $\psi : \mathbb{F}_{2^m} \to \mathbb{F}_2^d$ and $\phi : \mathbb{F}_2^d \to \mathbb{F}_{2^m}$ s.t:

$$\forall \vec{x}, \vec{y} \in S^d : \quad \vec{x} * \vec{y} = \psi(\phi(\vec{x}) \cdot \phi(\vec{y}))$$

- $m/d \to$ constant
- We can replace $\mathbb{F}_{2^m}$ by R and $\mathbb{F}_2$ by S

[CCXY18]: Cascudo et al CRYPTO 2018

In [CCXY18]:

To secret-share $\vec{x} \in \mathbb{F}_{2^m}$: Use secret-sharing $[\phi(\vec{x})]$

* Addition: $[\phi(\vec{x}+\vec{y})] \leftarrow [\phi(\vec{x})] + [\phi(\vec{y})]$ can be local

* Multiplication: we need to obtain $[\phi(\vec{x} * \vec{y})]$:

  ① Use MPC over $\mathbb{F}_{2^m}$ to multiply

  $$[\phi(\vec{x}) \cdot \phi(\vec{y})] \leftarrow [\phi(\vec{x})] \cdot [\phi(\vec{y})]$$

  ② Use MPC to apply the map $\tau = \phi \circ \psi$:

  $$[\tau(\phi(\vec{x}) \cdot \phi(\vec{y}))] \leftarrow \tau\left([\phi(\vec{x}) \cdot \phi(\vec{y})]\right)$$

$$\phi\left(\psi(\phi(\vec{x}) \cdot \phi(\vec{y}))\right) = \phi(\vec{x} * \vec{y})$$

This does not work in our current setting!

Applying the function τ adds extra rounds

Our solution

Secret-share $\vec{x} \in S^d$ differently

ψ is surjective but not injective

* Before: $[\phi(\vec{x})]$
* Now: $[x]$, where $x \in R$ is ANY element with $\psi(x) = \vec{x}$

ADDITION IS STILL LOCAL:

Given $[x]$ and $[y]$ with $\psi(x) = \vec{x}$, $\psi(y) = \vec{y}$, then $[x+y] \leftarrow [x] + [y]$ satisfies $\psi(x+y) = \vec{x} + \vec{y}$

What about (1-round) multiplication?

Assume preprocessed tuple

$$([a], [b], [\tau(a)], [\tau(b)], [\tau(a)\cdot\tau(b)])$$

Given $[x]$ and $[y]$ with $\psi(x)=\vec{x}$ and $\psi(y)=\vec{y}$, we obtain $[z]$ with $\psi(z)=\vec{x}*\vec{y}$ as follows:

① Parties open $d \leftarrow [x]-[a]$ and $e \leftarrow [y]-[b]$

② Parties compute locally

$$\underbrace{[\tau(x)\cdot\tau(y)]}_{z} \leftarrow \tau(d)[\tau(a)] + \tau(e)[\tau(b)] + [\tau(a)\tau(b)] + \tau(d)\cdot\tau(e)$$

NOTE $\psi(z) = \psi\left(\phi(\overbrace{\psi(x)}^{\vec{x}})\cdot\phi(\overbrace{\psi(y)}^{\vec{y}})\right) = \vec{x}*\vec{y}$

Notes

Our new encoding method also improves [CCXY18]:

* The extra "re-encoding" round (where τ was applied) is not needed
* The subspace check for the input phase disappears

Additional challenges that we address

Active security
↳ { correctness of openings
  { consistency of broadcast

Revisiting Repairing Codes Over Galois Rings

Theorem (Characterization of repairing ability)

C has linear repair over S if and only if there exists an S-submodule $D_0 \subseteq C^\perp$ satisfying:

1. $\pi_0(D_0) = R$
2. For every $i = 1, \dots, n$, $\pi_i(D_0) \cong p^j S$ (for some j)

Theorem (Existence of repairing/regenerating codes)

$$x = \sum_{i=1}^{n} \phi_i(x_i) \cdot z_i \quad \text{with} \quad \begin{cases} \phi_i(x_i) = \mathrm{Tr}\left(\frac{w_i x_i}{\alpha_i}\right) \\ z_i = -\alpha_i / w_0 \end{cases}$$

- Tr: Generalized trace function
- α_i: Evaluation points
- w_i: "Weights" associated with the dual RS-code


Thanks