Homomorphic Secret Sharing and Applications from Lattice-Based Assumptions

Rick

2020 年 10 月 23 日

1 Introduction

Like any other lattice workshop, we begin by introducing FHE, except that in this talk our focus is not FHE.

HSS is like FHE, but it has three key features:

- No secret key
- Needs two or more non-colluding parties to evaluate
- Boarder assumptions, better efficiencies.

Also some limitations to avoid trivial constructions:

- Security: share of x should hide x
- Size: $|x_b| \approx |x|$
- Correctness: $\mathsf{Eval}_P(x_0) + \mathsf{Eval}_P(x_1) = P(x)$

1.1 Rough Landscape of HSS

表 1: Landscape of Homomorphic Secret Sharing

Assumption	Homomorphic Ability	Reference
LWE+	Circuits	[DHRW16,BGI15,BGILT18]
DDH, Paillier, LWE	Branching Programs	[BGI16,BCGIO17,DKK18,FGJS17,BKS19]
LPN	Low-deg Polynomials	[BCGIKS19]
OWF	Simple functions	[GI14,BGI15,BGI16b]
None	Linear Functions	[Ben86]

Constructions in different categories varies dramatically.

2 Application of HSS

The goal of using HSS in the MPC setting is to reduce communication complexity.

2.1 HSS Implies Low Communication MPC

Suppose we want to evaluate $C(x,x')$. We can ge a semi-honest protocol whose communication complexity scales only with input and output size (like that in FHE).

- First do **generic MPC** to compute the HSS.Share functionality (whose complexity scales with input size not the circuit size).
- Then each party locally evaluate $\mathsf{Eval}_C(x_i)$
- Finnally each party **exchange** results to reconstruct output. (Notice that for this application, only compactness is needed, non-additive share is also fine.)

Altogether the cost is $\mathsf{poly}(\lambda)\cdot|input|+|output|$

2.2 Implications

- FHE: Succinct 2PC for circuits
  - All relies on a narrow range of assumptions
  - High concrete costs
- HSS for circuits
  - Relies on FHE anyway
- HSS for branching programs: Succinct 2PC for BP
  - New assumptions (e.g. DDH)
  - Better efficiency for certain regimes

2.3 2-Server PIR

The non-triviality here is that

- Two servers cannot communicate
- Communication must be smaller than database size

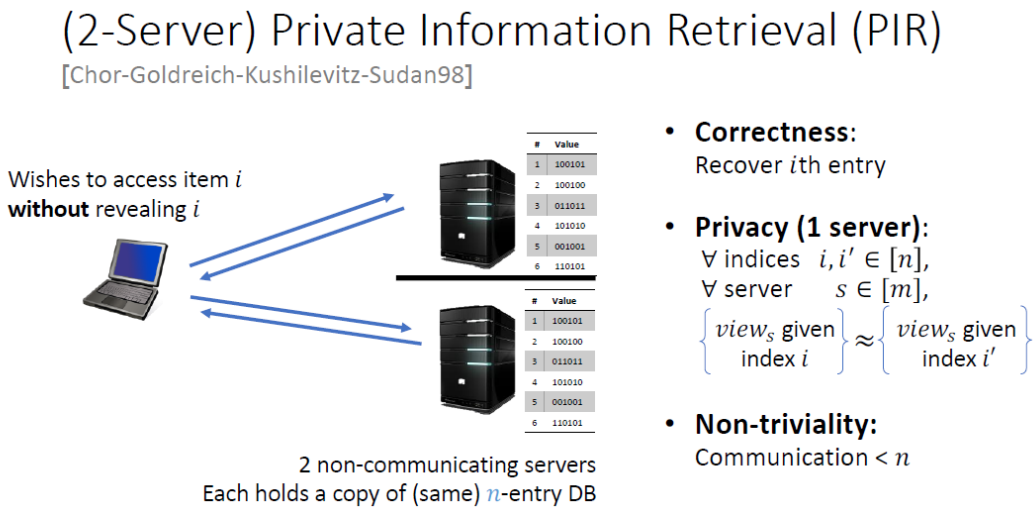


图 1: Private Information Retrival

The client formulates a private predicate P and shares it into P_0 and P_1. The shares are then distributed to the server, who then evaluates the database content on the predicate.

A technique is to split the complex function "DB" into $\mathsf{DB} = \cup_i \mathsf{DB}_i$. Since the result of HSS.Eval is additive shares of output, each server can send back the sum of i results since the client will add them up anyway.

Communication Complexity:

Client -> Server blow-up of share

Server -> Client rate 1

2.3.1 State of affair of PIR

In the following discussion we assume DB size is n.

表 2: State of Affair of PIR

	Statistical Privacy	Computational Privacy
2+ Servers	Slightly $n^{o(1)}$	From OWF $(\lambda+2)\log n$
1 Server	Impossible	From PKE $\mathsf{poly}(\lambda)+\log^2 n$

2.4 Other HSS Applications and Developments

- Secure computation for RAM programs. [DS17,GKW18,BKKO18] Actually this paradigm uses the two-server PIR as a sub-protocol.
- Proving statements split across verifiers. [BBCGI19] This work may serves as compiler to convert passive security to active security.
- Worst-case to average-case reductions in complexity theory [BGILT18]
- Connection to locality-preserving hash [BDGIKK]

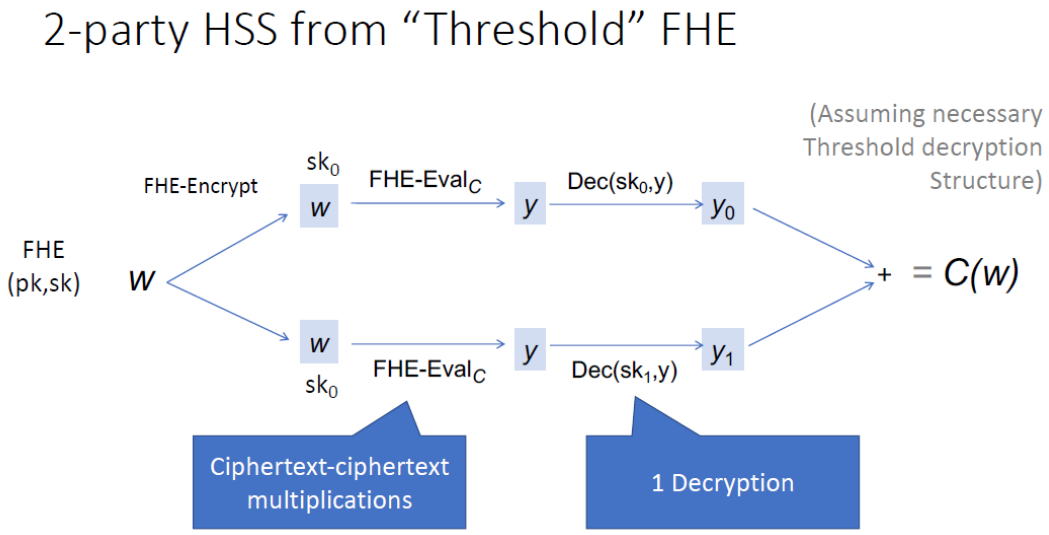
The last two points seem not so much related to cryptography.

3 HSS from Lattice

Different paradigms for implying HSS under lattice assumptions.

3.1 2-Party HSS from "Threshold" FHE

If the FHE has the "threshold" property that partial decryption of ciphertext under shares of secret key forms shares of the encryped message, we can get a HSS scheme immediately.



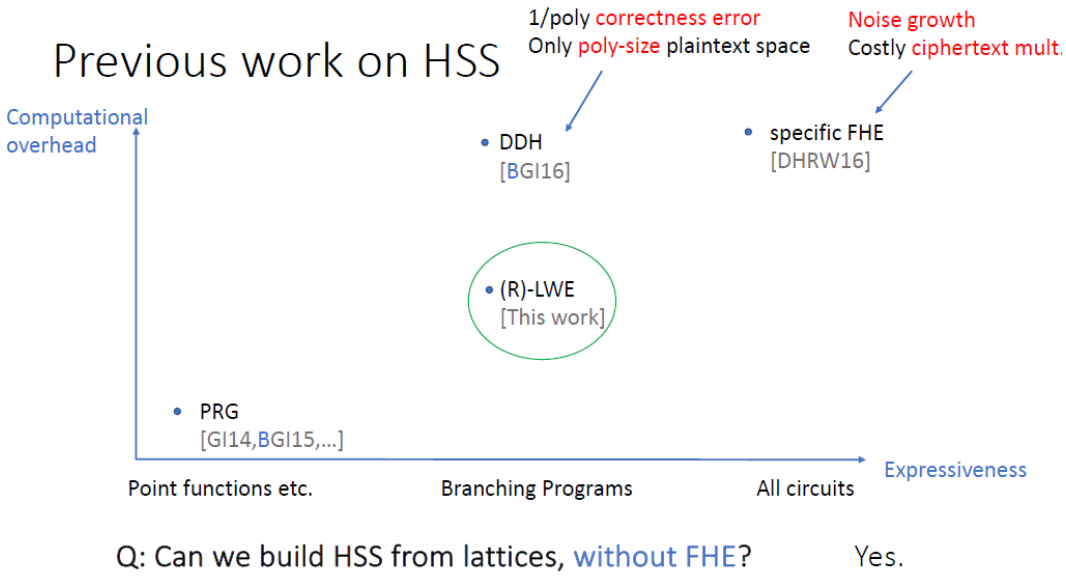


图 2: Landscape of Homomorphic Secret Sharing

3.2 Previous work on HSS

One curious question is that in [BGI16], why we are restricted to inverse polynomial error? Should a parallel repetition amplifies correctness, why did the author not use it? Or is it just a stupid notational constraint?

3.3 Branching Program

First we argue BP is strong enough for a list of functions:

- Multiplication of n n-bit numbers
- Many numerical / statistical calculations
- FHE Decryption
- Finite automata
- Min L2-distance from list of length-n vectors
- Undirected graph connectivity

- Streaming algorithms

It is also useful to formulate BP in a memory model.

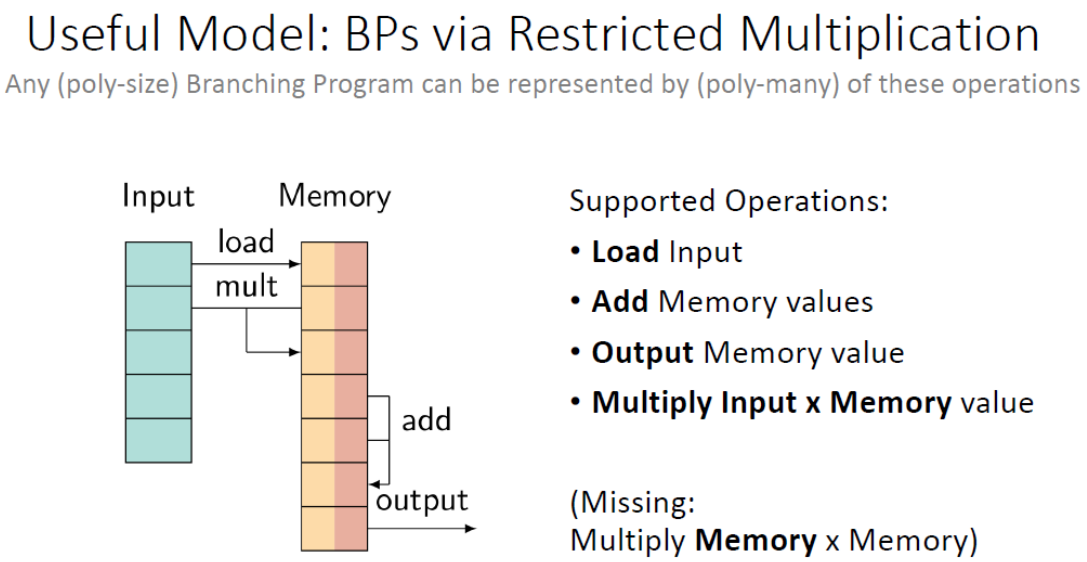


图 3: Branching Program Diagram

The [BGI16] construction suffers polynomial-sized message space, inverse polynomial error probability due to its limitations in the memory/input multiplication step.

3.4 Nearly-Linear Decryption

Consider plaintext space as $\mathbb{Z}_p$ and ciphertext space as $\mathbb{Z}_q$ where $q \gg p$. Many LWE-based encryption scheme (e.g. the one in [Regev05]) supports a linear function *LinDec* such that

$$LinDec(sk, [x]) \approx \frac{q}{p}x \mod q. \tag{1}$$

3.5 Multiplication

With the function LinDec we can then design a solution for multiplication (input/memory).

The main idea is very simple. By letting the encrypted value of x to be $\mathsf{Enc}(\mathsf{sk} \cdot x)$, we can use the memory value as secret key to decrypt input value. Using linearality, the output is noisy additive shares of $x \cdot y \cdot \mathsf{sk}$, which is almost what we want. But we need to remove the noise.

The noise removal is very simple: rounding and lifting.

It seems that the so-called "rounding" lemma and "lifting" lemma are the major contributions of this work. So I think why not try to prove them myself, since they are elementary enough and may be of use in other circumstances.

Lemma 1 (Rounding)**.** *If z_1 and z_2 are uniform shares of value z such that $z_1 + z_2 = \frac{q}{p}z + e \pmod q$ for some B-bounded noise e, then on the rounding operation $Round(x) = \lfloor \frac{p}{q}x \rceil$, we have*

$$Round(z) = Round(z_1) + Round(z_2) \pmod p \tag{2}$$

except with probability $O(B\frac{p}{q})$.

My understanding of this bound is that since the two shares are uniformly random over the correct reconstruction, we can view them as counter-diagonal lines over a 2-D plane (say z_1 being the x-coordinate and z_2 being the y-coordinate). Now the rounding operation on the left hand side correspond to rounding a number, whereas in the right hand side to mapping a $\frac{q}{p}$-by-$\frac{q}{p}$ box into its center point. The **good** boxes are those forming a straight line such that the mapped value would sum to left hand side, whereas all rest boxes would incur error. If we choose the box big enough, regions not covered by this diagonal of **good** boxes would be very small (a wild guess would be $O(B\frac{p}{q})$. In the original paper, error probability seems to be multiplied by the degree N. A wilder guess is that by applying our analysis to each coordinate independently and using union bound, we can get the same result.

Lemma 2 (Lifting)**.** *Let z_1 and z_2 be uniformly random additive secret shares of some value z over $\mathbb{Z}_p$, then except with probability $\frac{|z|}{p}$, they forms addtivie secret shares of z over $\mathbb{Z}_q$.*

I think this can also be shown rather clearly using 2-D image. Consider z_1 as the x-coordinate and z_2 as the y-coordinate as usual, then the joint distribution is uniform over all the points (x, y) such that

$$x + y = z, z \pm p, z \pm 2p, \ldots \tag{3}$$

But since we are only interested the range $[-\frac{p-1}{2}, \frac{p-1}{2}]$, only a $\frac{|z|}{p}$ fraction of them would be off the "main line", which completes the proof.

3.6 Takeaway

The takeaway of this lecture is that using plain lattice-based encryption, we can achieve evaluation of any branching program with magnitude B such that

$$B \underbrace{\ll}_{\text{lift}} p \underbrace{\ll}_{\text{round}} \frac{q}{B}. \tag{4}$$

3.7 Circular Security

Actually [ACPS09,BV11] showed that the form $\mathsf{Enc}_{\mathsf{sk}}(x \cdot \mathsf{sk})$ is secure without additional assumptions. Moreover, this means by using public key and x only, one can generate encryption of $x \cdot \mathsf{sk}$.

4 PCG from LPN

In the previous construction, we have gained additive share of some evalutation $P(x)$ from homomorphic secret shares of x. In fact, if what we want is correlated randomness, this is already good enough.

Why with correlated randomness, MPC's online phase has constant computation and communication overhead? Maybe this is because for evalutating every (arithmetic) gate each party only has to exchange two (field) elements?

4.1 Functionality of PCG

Like PRG, where a single random seed is stretched into large pseudorandom strings, a PCG extend a pair of correlated random seeds into a large string of correlated randomness. The requirement (challenge) is that s_b should leak no information of X_{1-b} beyond X_b. And the stetching process should be local.

4.2 Constructions of PCGs

- Multiparty linear correlations (PRG) in [GI99,CDI05]
- Vector OLE from LPN in [BCGI18, BCGIKRS19]

The CRYPTO19 paper [BCGIKRS19] contains a collection of results, including:

- OT from LPN
- Constant-degree polynomial evaluation from LPN
- Mutiparty bilinear relation from LPN and LWE
- Truth table from PRG
- Low-degree via HSS (from LWE, pairings, MQ)

[BCGIO17,S18] also proposed some constructions, albeit less practical.

4.3 Generic PCG Construction for "Additive" Correlation from HSS

A general result in [BCGIKRS19].

4.3.1 Additive Correlations

By additive correlation we mean additive shares of value drawn form some "master distribution", like Beaver's triple, authenticated Beaver's triple, etc.

4.3.2 Construction

If we have an HSS scheme for some class of function, we can acquire PCG from HSS.

Consider a program $P(s)$ where s is a seed. The program first expand the seed into longer pseudorandom string, and then use a sampling algorithm to sample y from a distribution R, which is then outputed.

Suppose tho party has shares of some secret s, then using HSS we can get shares of evalutaion result y.

So how do we design such program P such that efficient HSS scheme can support (log-depth hopefully)? HSS for Linear functions and point functions have efficiency even better than those for BP's. So can we get a PRG from these simple operations (Prof. Boyle showed that we can do 200 million evaluation per second)?

It seems that the construction presented in this lecture does not fit perfectly in this framework. But it should be faster than generic construction.

Today's lecture would be about

1. Vector OLE correlation
2. Oblivious transfer correlation

4.3.3 Vector OLE

VOLE is an extension to OLE, which is sort of OT's analogue in arithmetic circuits. For some finite field $\mathbb{F}$, VOLE computes the functionality:

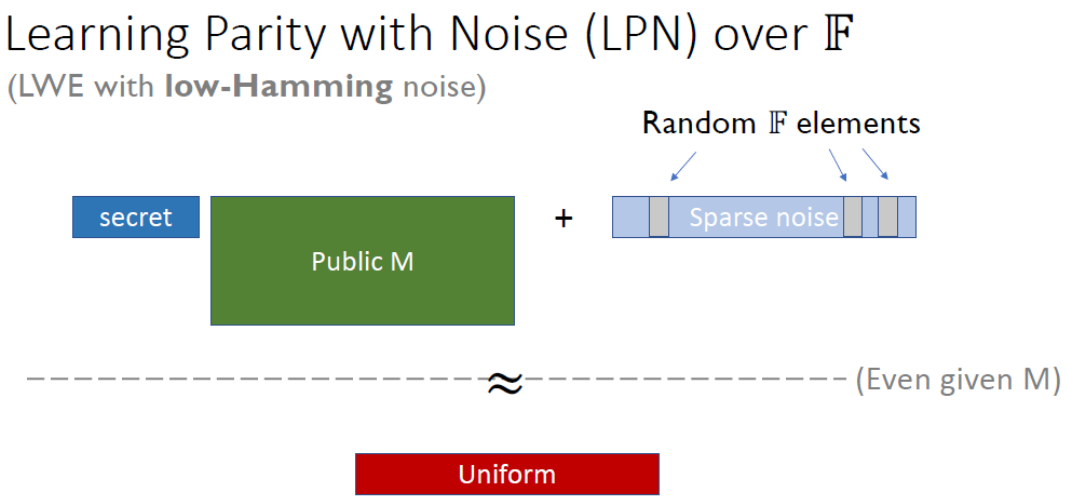
$$\mathscr{F}_{\text{VOLE}} : \mathbb{F}^{2n} \times \mathbb{F} \to \mathbb{F}^n \tag{5}$$

$$\mathscr{F}_{\text{VOLE}} : (a,b) \times c \mapsto a \cdot c + b \tag{6}$$

We define the correlation $R = \{a,b,c,d : \mathscr{F}_{\text{VOLE}}(a,b,c) = d\}$. The goal is to generate pseudorandom samples from R in an HSS manner.

4.3.4 Learning Parity with Noise

The subtitle in this slide is "LWE with **low-Hamming** noise". How insightful.



Note: Parameterized by **M** & by **noise distribution**

LPN is parameterized by distribution of A and the noise.

The LPN assumption in this work has the features:

- Over $\mathbb{F}$ bigger than $\mathbb{F}_2$
- High dimenstion k
- Low noise (noise rate is $\frac{1}{k^\varepsilon}$ for some constant ε)
- Bounded number of samples

Enlarging field seems benign for security, though I do now currently know how to prove it. Prof. Boyle argued that the choice of $\varepsilon > 1/2$, meaning that we do not currently know how to build PKE from this assumption.

The idea is to leverage linearity to reduce problem to **sparse** space. (If noise bit is zero, the operation on that bit is public and linear.)

4.3.5 Primal Construction

Start with a short VOLE correlation and expand it using M (the LPN public matrix). We get a construction that satisfies correctness but losses security, since in this case the other party's share has at most n-bit entropy, whereas we want m-bit entropy.

We stick to the notation in her slides. c, d $cx+d$ are the correlated randomness inside the seed, M is the random matrix, C, D, $Cx+D$ are the expanded results, and finally a' is the sparse noise vector.

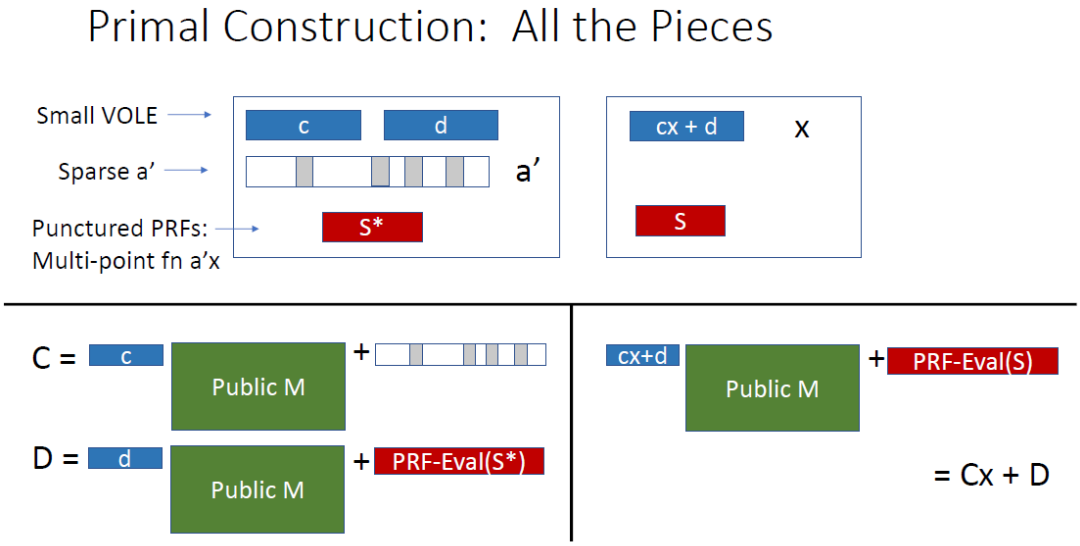
At this point I have think I have confused about the goal in this section. Based on the following content I think the construction here is to expend short correlated randomness into large Random-VOLE-like outputs, much like a reminiscent of OT extension.

Consider two parties P_0 and P_1 where each can hold its share of correlated input, and we want to design such short input such that they can (using LPN assumption) extend their input into large pseudorandom output.

Initially P_0 holds short c, d while P_1 holds x and $cx+d$. Now we add a short sampling seed to P_0's input so that it can now sample a sparse random noise a'. Notice that if we use "punctured" PRF (i.e. To puncture x, we need to provide the value of every sibling on the path of x to the root in the GGM tree) we are actually able to get succinct secret sharing of some spare value $a'x$.

The trick here is that for every coordinate i such that $a'_i \neq 0$, we can create a punctured one-bit PRF key such that it cannot evaluate on i. And then since in our application, the noise is known to some party P_0, we can give additional information to P_0 so that it can evaluate $\mathsf{PRF}(s,j)$ on every $a'_j = 0$ and get $\mathsf{PRF}(s,i) - a'_i x$ for every $a'_i \neq 0$. In this way, we created some kind of secret share of a sparse vector $a'x$ (though the shared secret is known to one party). And that is exactly what we need.

The idea is perhaps best epitmoized in Prof. Boyle's presentation slides.



4.3.6 Dual Construction

LPN has a dual view. Let

$$s^t G + e^t \approx U_m \tag{7}$$

be the description of LPN problem (of course G is public), then by multiplying the kernel of G on the right, we can get

$$s^t GH + e^t H \approx U_m H \approx U_{m-n}. \tag{8}$$

And that kind of reminds me about knapsack LPN problem. So the assertation here is that by taking a sparse noise e and shrink it by n-bits by multiplying H, what we ends up with is pseudorandom.

So we can use this notion combined with the punctured-PRF based construction, we can get a dual construction, which seems not so different.

But it turns out that we cannot get super-quadratic stretch in the primal construction, whereas we for the dual construction, we can get arbitrarily stretch.

4.4 OT Correlation

I think Prof. Boyle has confirmed our effort in the previous section can be viewed as a low-communication, field-extended version of OT extension.

Construction in this section is from CRYPTO 2019 and CCS 2019.

"Silent" OT Extension: Securely Generating Seeds

- 2-round secure seed generation protocol (building from Vector OLE)
  - Hash the vector OLE outputs to destroy unwanted correlations (similar to [IKNP03])
- Active security:
  - Lightweight PPRF consistency checks for malicious sender
    - Allows selective failure attacks — sender can guess 1 bit of LPN error
    - Assume problem is hard with 1-bit leakage
  - 10-20% overhead on top of semi-honest
- Implementation:
  - Main challenge: fast mult. by H
  - Quasi-cyclic : polynomial mult. mod $X^n - 1$
  - Security based on quasi-cyclic syndrome decoding / ring-LPN

5 Summary

Part I Simple HSS for Branching Programs from Lattices (R/LWE)

Part II Pseudorandom Correlation Generators from LPN

- First a generic construction
- Then a specific construction

6 Open Problems

- HSS Constructions
  - 3+ party HSS (beyond **pathetic**)
  - Beyond branching programs (without FHE)
  - FHE-style bootstrapping?
  - Lower bounds? Necessity of certain assumptions/tools?

- Pseudorandom correlation generators
  - Efficient constructions for further correlations (say **Garbled Circuit** correlations)
  - Understanding power of additive correlations
- New applications
- Better efficiency! (for all of the above)