Additively Homomorphic UC Commitments With Optimal Amortized Overhead

Ignacio Cascudo, Ivan Damgård,
Bernardo David, Irene Giacomelli,
Jesper Buus Nielsen, Roberto Trifiletti

Aarhus University

Structure

1. Introduction
2. A general framework
3. Achieving additive homomorphism
4. Efficiency
5. Follow-up work and Open Questions

Commitment Schemes

Universal Composability

- Protocols remain secure in parallel concurrent executions and arbitrary composition.

- Commitments require setup assumptions [CF01].
- Commitments are complete [CLOS02].

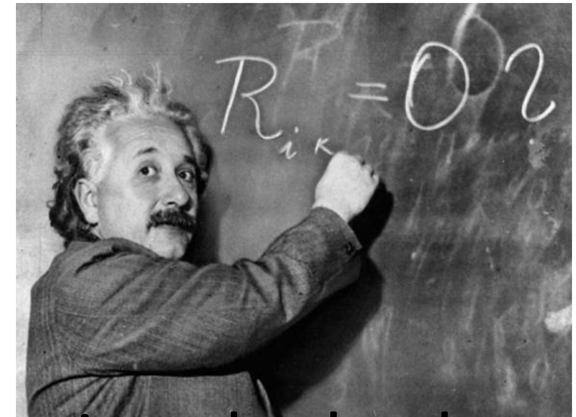
Extractability, Simulator can open if Committer is corrupt

Equivocability: Simulator can change its mind if Reciever is corrupt.

Related Works

- DDH based fast UC commitments: static security [Lindell11,BCPV13], adaptive security [DN02, DG03].
  - Use a Common Reference String (CRS).
  - High asymptotic communication and computational complexity.
- UC commitments (with optimal rate): [GIKW14] (see also [DDGN14]).
  - Use Oblivious Transfer as a setup assumption.
  - Require PRGs and Codes that are also good Linear Secret Sharing schemes.

What do we do in theory?



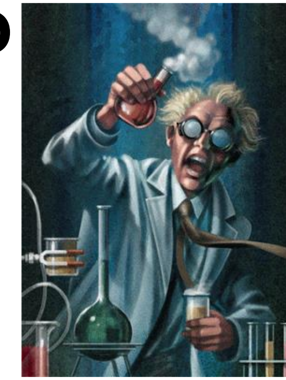
- Optimal communication
- Additively Homomorphic
- Optimal computation

- Can use any good code, no need for it to be both a good code and a good secret sharing scheme.

How do we do it?

ECC + PRG + OT

What do we do in practice?



- Online Phase:

BCH [796,256,>=121] + PRG

2 Encodings: 1.5 μs

VS.

[Lindell11,BCPV13] -> 22 exponentiations: 8250 μs

II

- Practical scheme runs 5500 times faster

Practical Trade Offs…

- No additive homomorphism.
- Then setup phase cost:

796 OTs

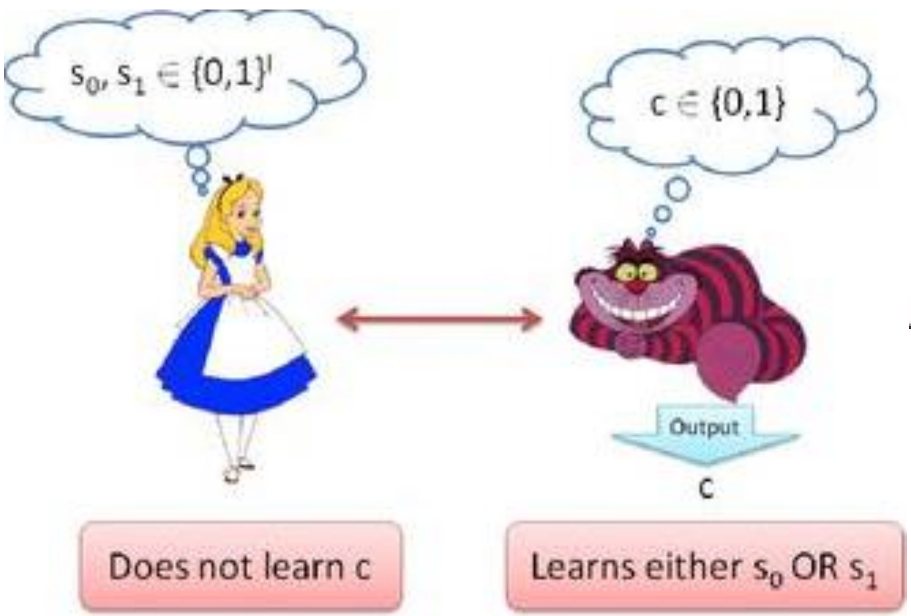
8756 exponentiations using [PVW08]

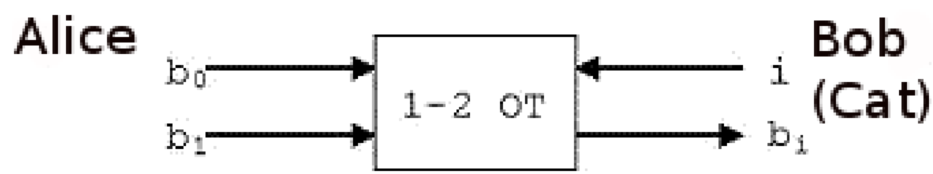
398 [Lindell11,BCPV13] commitments

Building Blocks

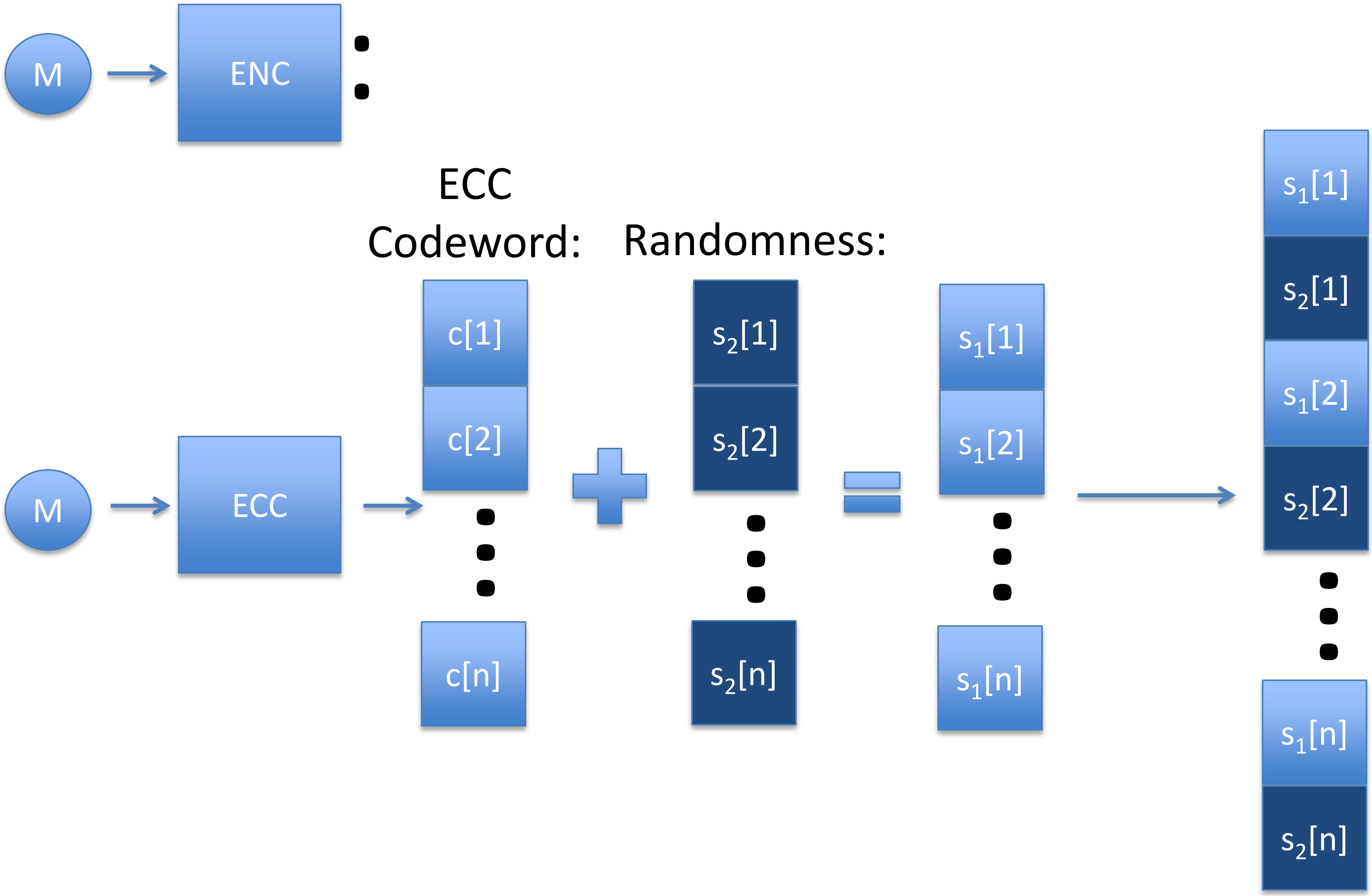
- Error correcting codes:
  - Linear-time encodable codes [GI01,GI02,GI03,GI05,Spi96,DI14].
- UC Oblivious Transfer:
  - Any UC Oblivious Transfer protocol, e.g. [PVW08]
- Pseudorandom Generator:
  - Linear-time PRG, e.g. [VZ12]

Oblivious Transfer





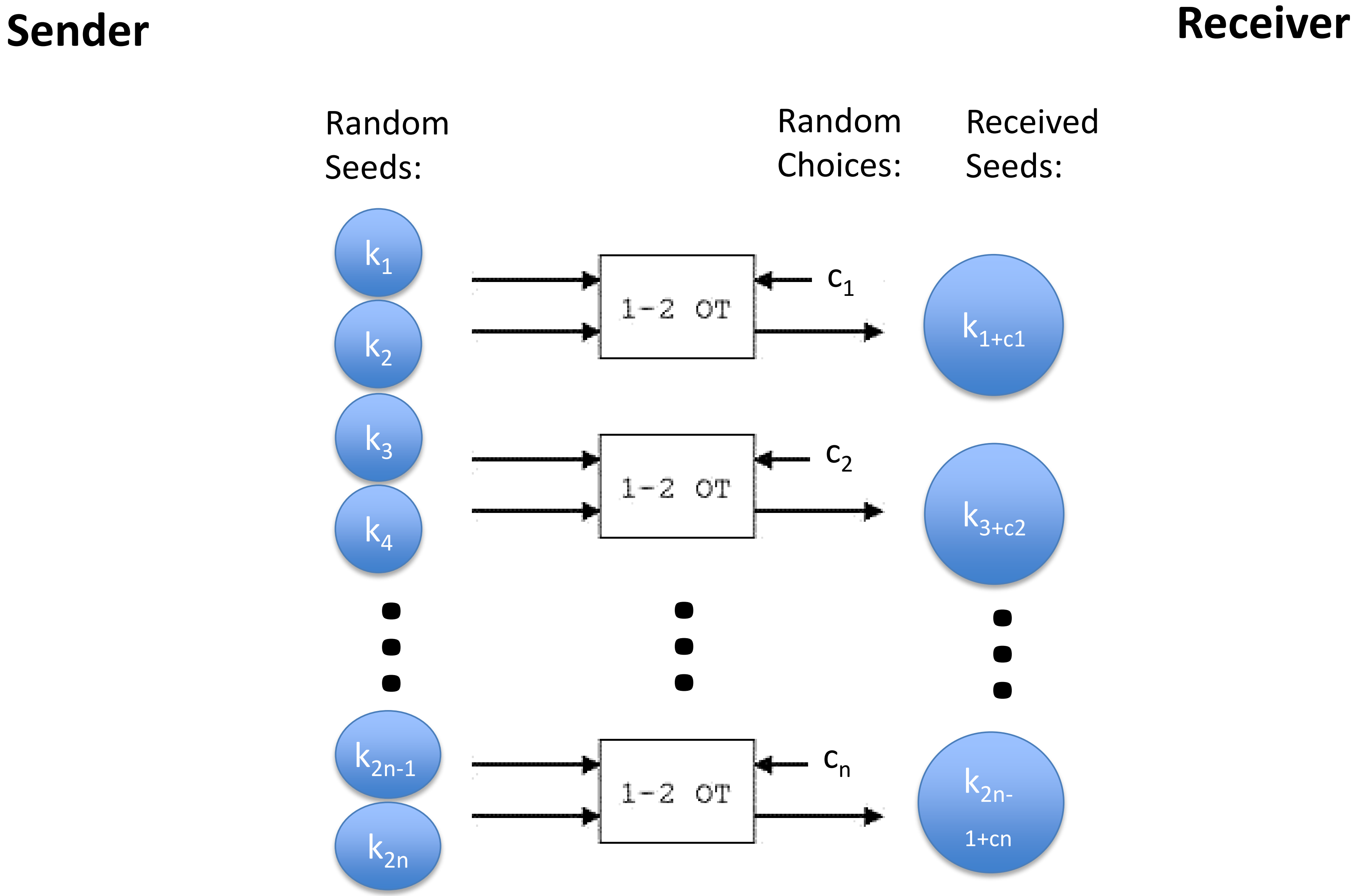
Encoding Scheme



General Framework

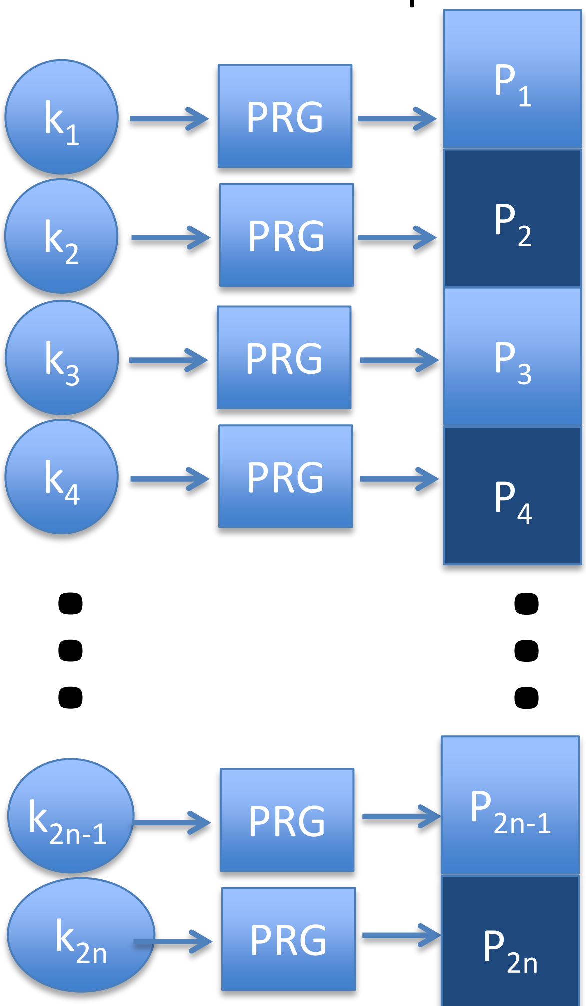
- Setup phase:
  - Independent from the inputs
  - Constant number of OTs for unbounded number of commitments.
  - Constant communication complexity.
- Commitment/Open Phases:
  - Linear communication complexity (in size of string committed to).
  - Only require a PRG and the encoding scheme.
  - Non interactive.

Setup Phase

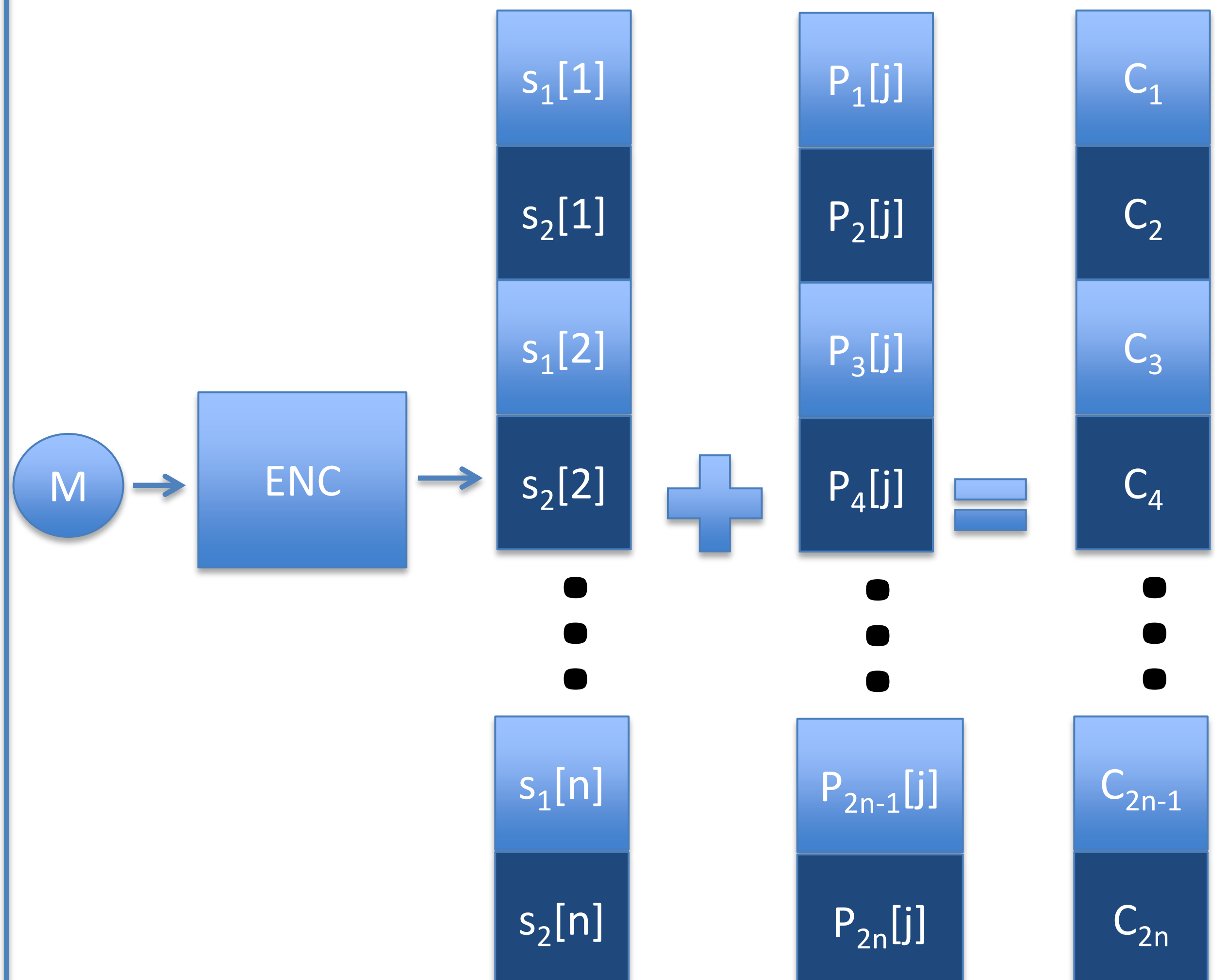

Sender
Receiver
Random Seeds:
Random Choices:
Received Seeds:
k1
k2
k3
k4
k2n-1
k2n
1-2 OT
1-2 OT
1-2 OT
c1
c2
cn
k1+c1
k3+c2
k2n-1+cn

Commit Phase (Sender) j'th com.

Once and for all,
Generate one-time pads:

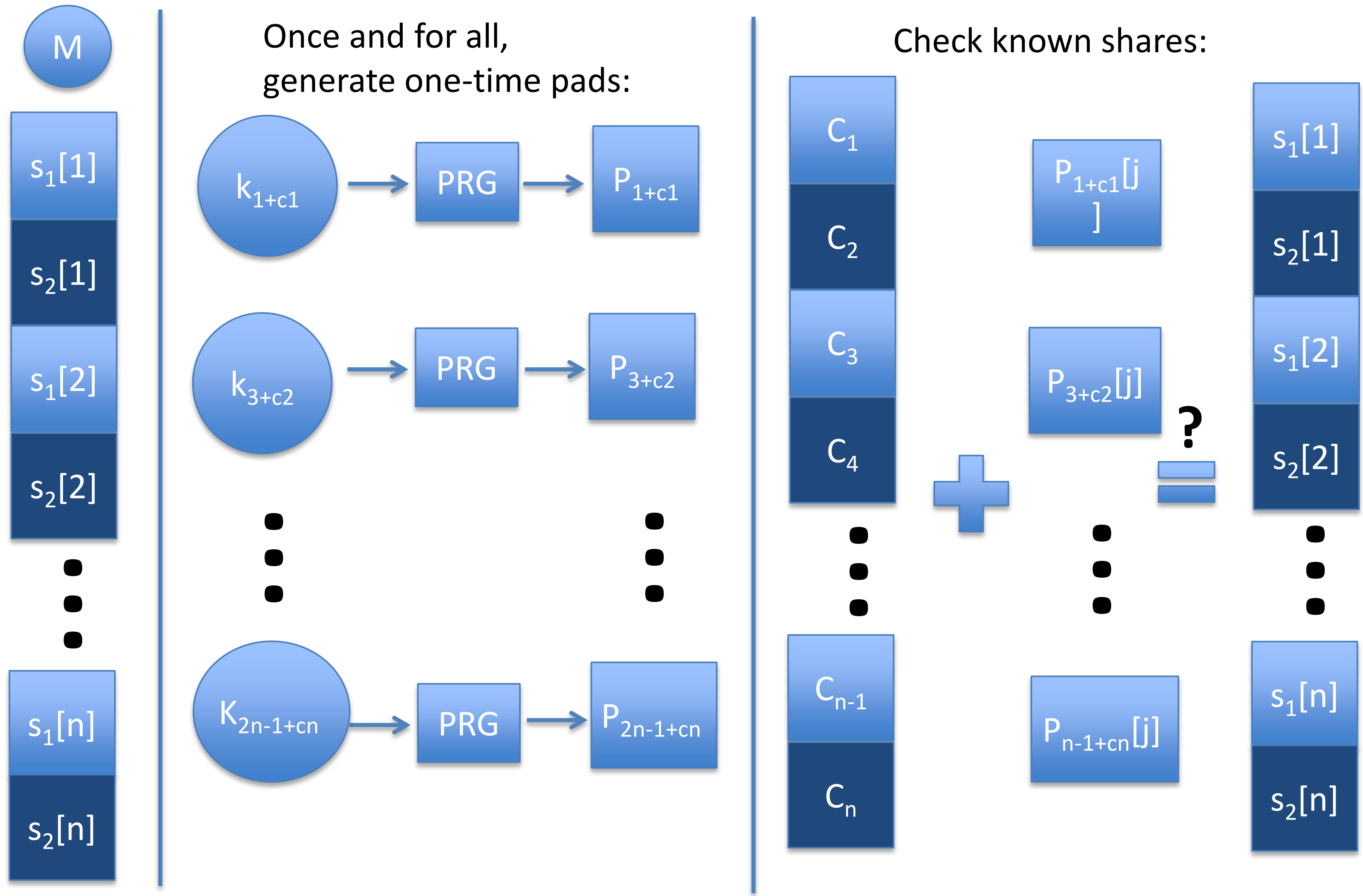


Encode messages and encrypt with single entries of one-time pads:

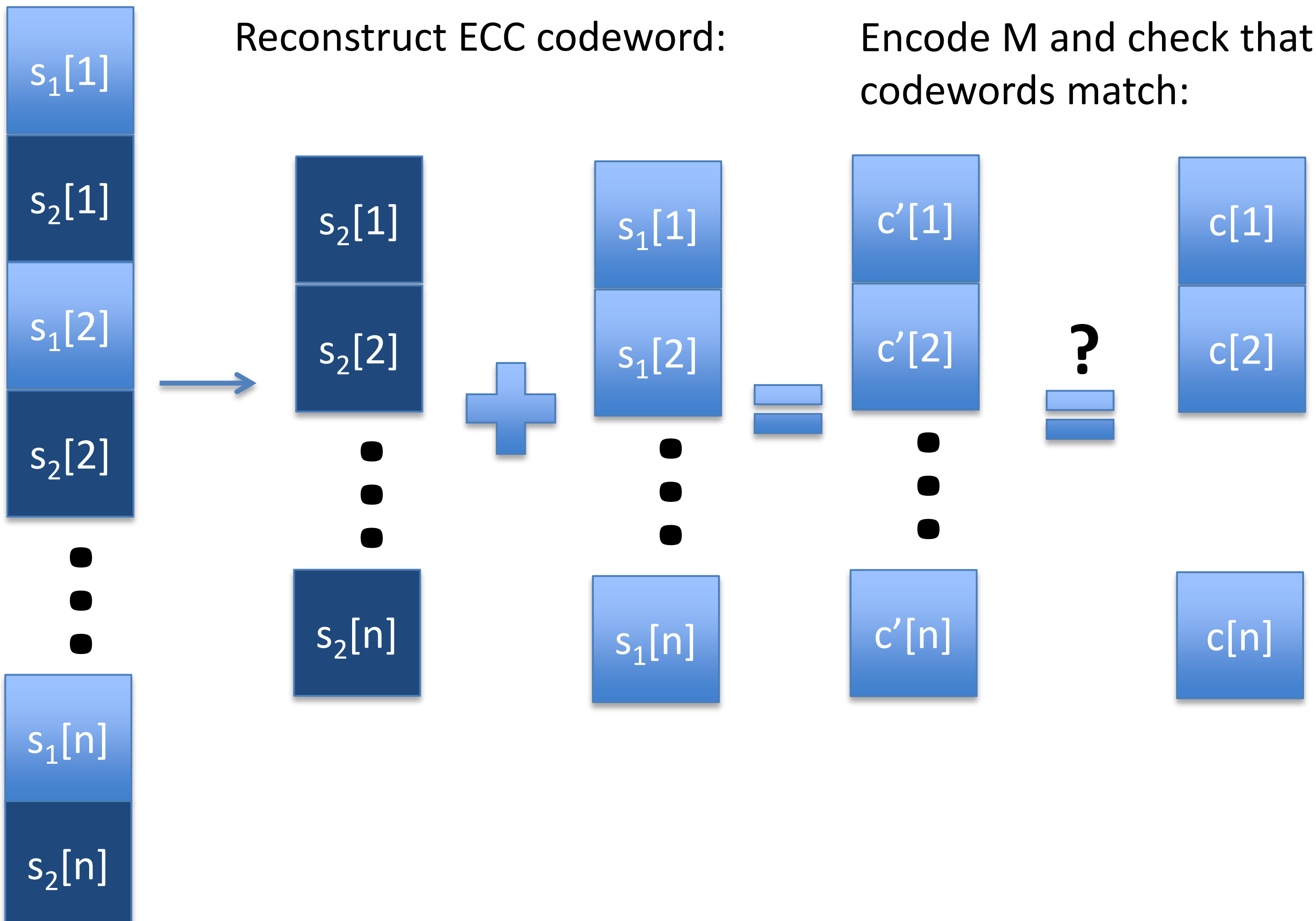


Open Phase (Receiver)

Opening Message:



Open Phase (Receiver)



Additive Homomorphism

- The encoding scheme can be seen as a LSSS.
- Use the encoding scheme to build a VSS scheme using techniques of [DDGN14] and then use MPC in the head.
- For this to work, we need to additively secret share each code-word entry into 3 additive shares.
- Made MPC in the head work for a non-threshold multiparty protocol.

Asymptotic Efficiency

- Computational complexity: O(k)
- Communication complexity: O(k)
- Round optimal (non interactive)

Scheme	Communication Complexity (in field elements)			Round Complexity		Computational Complexity		
	Commit	**Open**	**Total**	**Commit**	**Open**	**Commit**	**Open**	**Total**
Fig. 4 (homomorphic)	$\frac{2mnt}{k} + k$	m	$\frac{2mnt}{k} + k + m$	1	1	$\frac{4n(t-1)}{k}$ + 2 Enc.	1 Enc.	$\frac{4n(t-1)}{k}$ + 3 Enc.
Fig. 2 (basic)	nt	m	$m + nt$	1	1	1 Enc.	1 Enc.	2 Enc.

m=k+n(t-1)

Concrete Efficiency

- Underlying ECC: BCH [796,256,>=121]
- On average, encoding takes 0.75 μs and exponentiations on 256 bits field take 375 μs.

Scheme	Communication Complexity (in bits)			Round Complexity		Computational Complexity		
	Commit	**Open**	**Total**	**Commit**	**Open**	**Commit**	**Open**	**Total**
[BCPV13] (Fig. 6)	1024	2048	3072	1	5	10 Exp.	12 Exp.	22 Exp.
[Lin11] (Protocol 2)	1024	2560	3584	1	3	5 Exp.	$18\frac{1}{3}$ Exp.	$23\frac{1}{3}$ Exp.
Fig. 4 (homomorphic, $t = 3$)	34733	1848	36580	1	1	27 Enc.	1 Enc.	28 Enc.
Fig. 2 (basic, $t = 2$)	1592	1052	2644	1	1	1 Enc.	1 Enc.	2 Enc.

- Our basic scheme is faster than previous schemes even in the ROM.

Open Problems

- Can we get optimal rate?
- Can we get additive homomorphism in this construction without VSS?
- YES! Follow-up work [Nielsen et al. 15]: check that committer uses (almost) a code word by checking random linear combinations.
- Very small overhead, natural idea but non-trivial to prove.

Usage with garbling schemes

- Several schemes seem to need efficient homomorphic commitments
- No need for UC OT in set-up phase in this context, can use the OT’s already available.
- Seems to be the garbler’s best friend ☺

THANK YOU!

READ THE FULL PAPER:

https://eprint.iacr.org/**2014/829**