Succinct Arguments from MIPs and their Efficiency Benefits

Nir Bitansky **Alessandro Chiesa**

How quickly can we verify the result of long computations?

$\exists\, w$ s.t. $M(x, w) = 1$ in $\leq T$ steps?

$$\boxed{P} \xrightarrow{\qquad w \qquad} \boxed{V}$$

proof w checkable in T time

$$L = \mathrm{Lang}(M) \in \mathrm{NP} \Rightarrow T_L = \mathrm{poly}_L(|x|)$$

Succinct Arguments for NP

A (computationally-sound) proof for NP where verifier's time complexity is independent of the time complexity T_L required to check membership in the language.

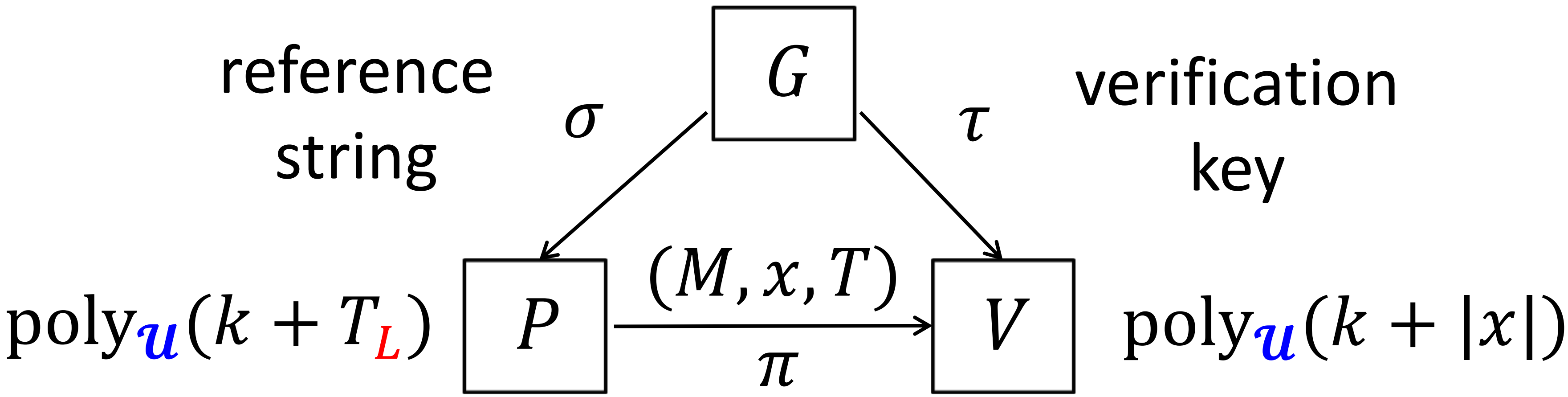
$$\exists\, w \text{ s.t. } M(x,w) = 1 \text{ in} \leq T_L \text{ steps?}$$

$$\mathrm{poly}_u(k + T_L) \quad \boxed{P} \quad \begin{array}{c} \longrightarrow \\ \longleftarrow \\ \vdots \\ \longrightarrow \end{array} \quad \boxed{V} \quad \mathrm{poly}_u(k + |x|)$$

[Kil92]
[Mic94] Exist under standard assumptions (CRHs)
[BG02]

Succinct arguments enable us to delegate ``NP''

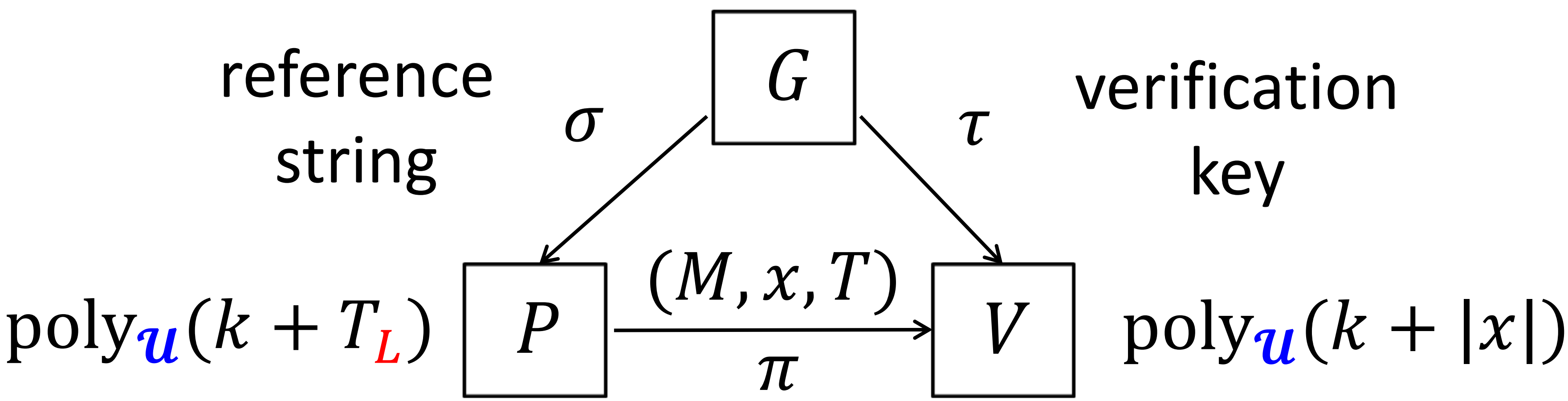
Non-Interactive Succinct Arguments (Of Knowledge) $\equiv$ SNARKs



- τ must be secret = *designated-verifier*
  τ can be published = *publicly-verifiable*

- $\mathrm{TIME}(G) = \mathrm{poly}_{\mathcal{U}}(k)$ *fully-succinct*
  $\mathrm{TIME}(G) = \mathrm{poly}_{\mathcal{U}}(k + T_L)$ *preprocessing*

Non-Interactive Succinct Arguments (Of Knowledge)
≡ SNARKs



KNOWN:

[Gentry Wichs 11] can't prove secure via black-box reduction to falsifiable assumptions (for ``hard enough NP language")

[BCCT11]
[DHF11]
[GLR11]

fully-succinct BUT designated-verifier
from extractable collision-resistant hashes

[Groth10]
[Lipmaa11]
[GGPR12]

publicly-verifiable BUT preprocessing
from knowledge of exponent assumptions

Verifier runs fast, gets strong guarantee.

BUT…

What about the prover?

The verifier might be paying the prover for his work!

ADDITIONAL GOAL:

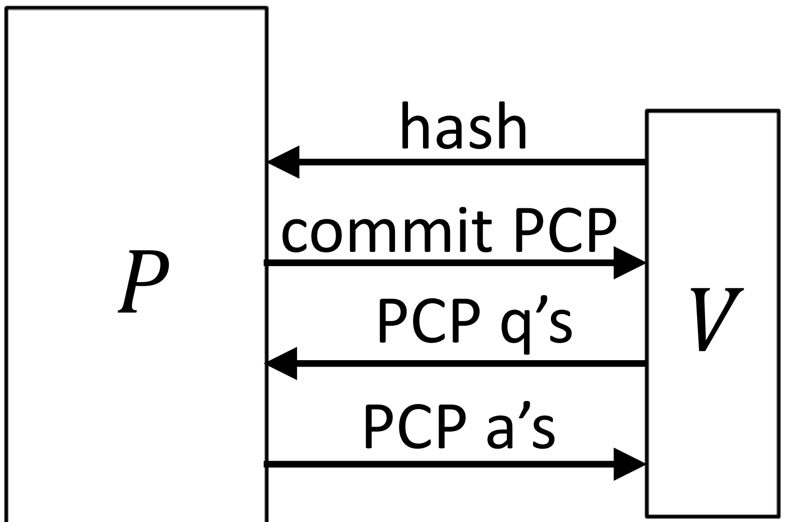
minimize prover's complexity!

Where do we stand?

2 Approaches for Succinct Arguments for NP

PCP-based

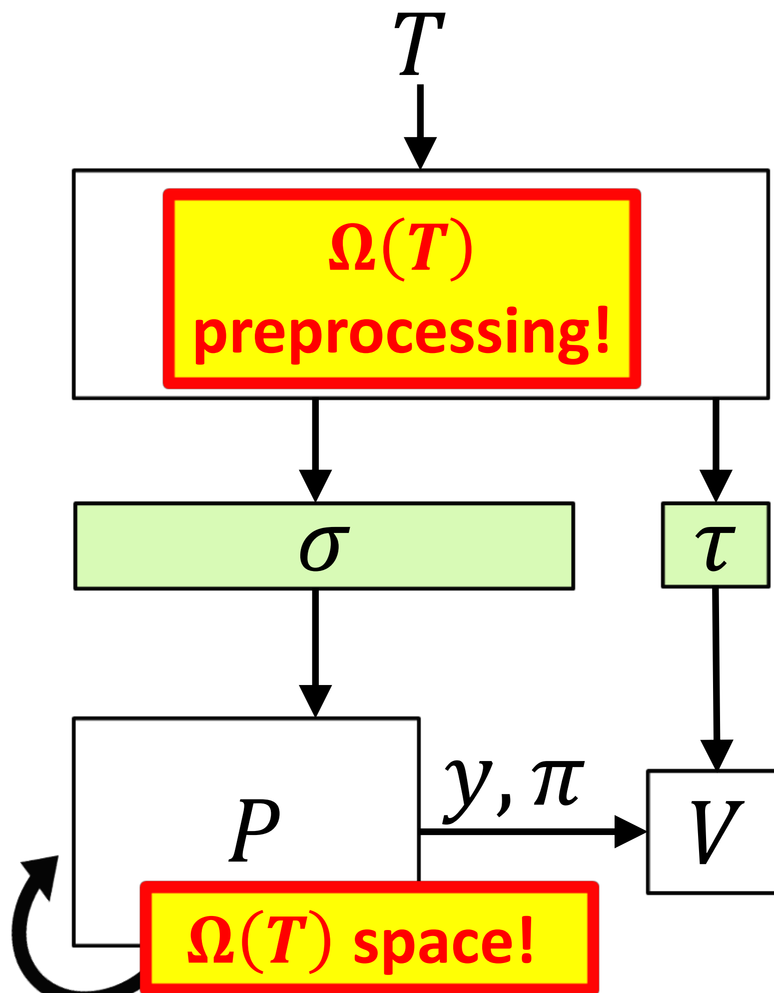
4-msg from CRH [Kil92,Mic94,BG02]
2-msg from PIR+ECRH [BCCT11,DHF11,GLR11]



w/ state-of-the-art PCPs [BCGT12]
$\widetilde{O}(T)$ time BUT need $\Omega(T)$ space!

bilinear maps + KEA

[Groth10,Lipmaa11,GGPR12]



NOT EFFICIENT ENOUGH!

For a T-time S-space RAM computation:

	preprocessing time	prover time	prover space	verifier time
[Kil92] …	$\mathrm{poly}(k)$	$T \cdot \mathrm{poly}(k)$	$\boldsymbol{T} \cdot \mathrm{poly}(k)$	$\mathrm{poly}(k)$
[GGPR12]	$\boldsymbol{T} \cdot \mathrm{poly}(k)$	$T \cdot \mathrm{poly}(k)$	$\boldsymbol{T} \cdot \mathrm{poly}(k)$	$\mathrm{poly}(k)$

QUESTIONS

Are there **COMPLEXITY-PRESERVING**

- succinct arguments from standard assumptions?
- SNARKs from reasonable assumptions?

Yes and Yes

RESULTS

Theorem 1

MIP + FHE $\Rightarrow$ complexity-preserving 4-msg succinct argument

[not public coin]

new tool:
succinct function commitment

Theorem 2

MIP + FHE w/ knowledge $\Rightarrow$ complexity-preserving SNARK

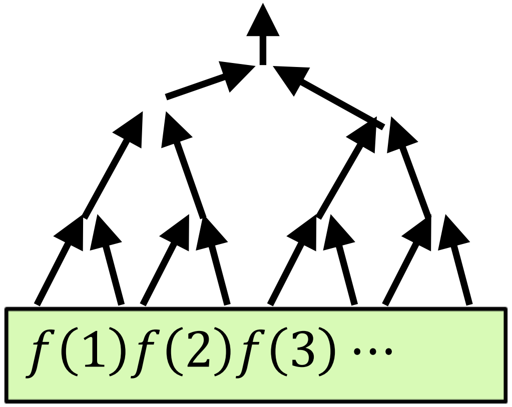
[designated verifier]

new (non-standard) assumption:
FHE with extractable homomorphism

Why do MIPs pop up here?

The Role of MIPs

What is the problem with PCP+CRH?



Let $f(i)$ compute ***i*-th bit** of PCP.

Committing to PCP requires

|PCP|=$\Omega(T)$ evaluations of f.

How to compute all these evaluations?

naively: $\Omega(T^2)$ time

[BCGT12]: $\tilde{O}(T)$ time via FFT methods BUT $\Omega(T)$ space

BUT: verifier asks only $q \stackrel{\text{def}}{=} \text{polylog}(T)$ evaluations!

Can we save on evaluations when committing?

If so, we may hope for better efficiency…

we treat f as a string because
Merkle trees are a <u>succinct STRING commitment</u>

ALTERNATIVE: treat f as a function

More concretely:

STEP 1: give a time-and-space-efficient construction in a model where the verifier sends one query to each of q ~~identical~~ functions $\equiv$ **MIP**

STEP 2: implement model in a <u>complexity-preserving</u> way

just as good: **not-necessarily-identical**

CHALLENGES

1. sufficiently-efficient MIP construction?
2. how to implement MIP model (w/ ONE prover)?

Essentially-Optimal MIPs

Thm: ∃ a 1-round MIP where to check that a T-time S-space RAM M accepts (x, w) for some w,
(i) the MIP verifier runs in time $\tilde{O}(|x|)$
(ii) each MIP prover runs in time $\tilde{O}(T)$ & space $\tilde{O}(S)$

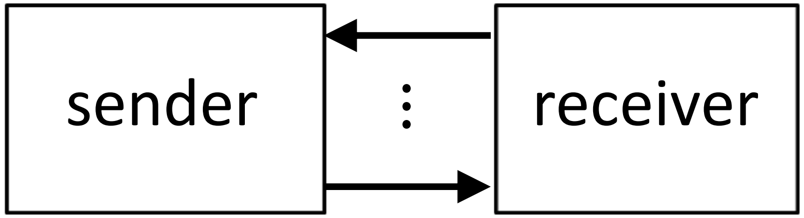
NOTE: PCPs with the above efficiency not known!

Tackled first challenge. ✓

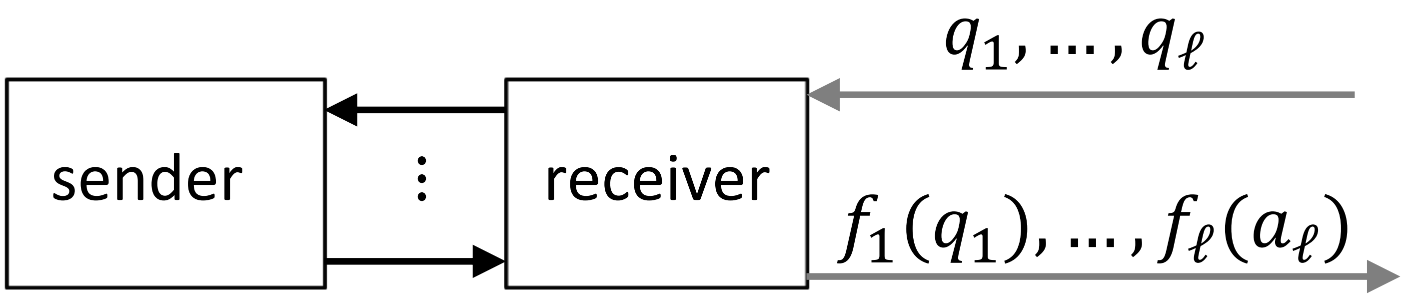
Succinct Function Commitment (SFC)

given T-time S-space functions $(f_1, \ldots, f_\ell): A \to A$,

commit



decommit



time: $\ell \cdot \boldsymbol{T} \cdot \mathbf{poly}(\boldsymbol{k})$ $\quad \ell \cdot \mathbf{log}\,|\boldsymbol{A}| \cdot \mathbf{poly}(\boldsymbol{k})$

space: $\ell \cdot \boldsymbol{S} \cdot \mathbf{poly}(\boldsymbol{k})$ $\quad \ell \cdot \mathbf{log}\,|\boldsymbol{A}| \cdot \mathbf{poly}(\boldsymbol{k})$

Succinct <u>Function</u> Commitment (SFC)

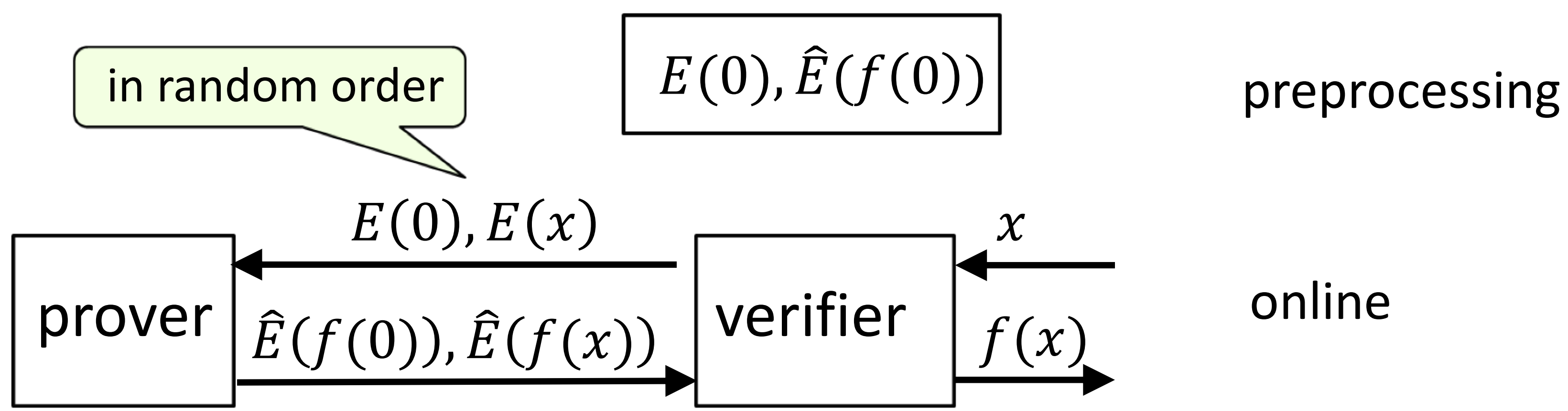
- [Ishai, Kushilevitz, Ostrovsky, CCC ‘07]
  linear hom. enc. $\Rightarrow$ SFC for linear functions

Succinct Function Commitment (SFC)

Thm: FHE $\Rightarrow$ 4-msg SFC for ANY polytime function

IDEA:

STEP 1: start from the delegation scheme of [CKV10]…

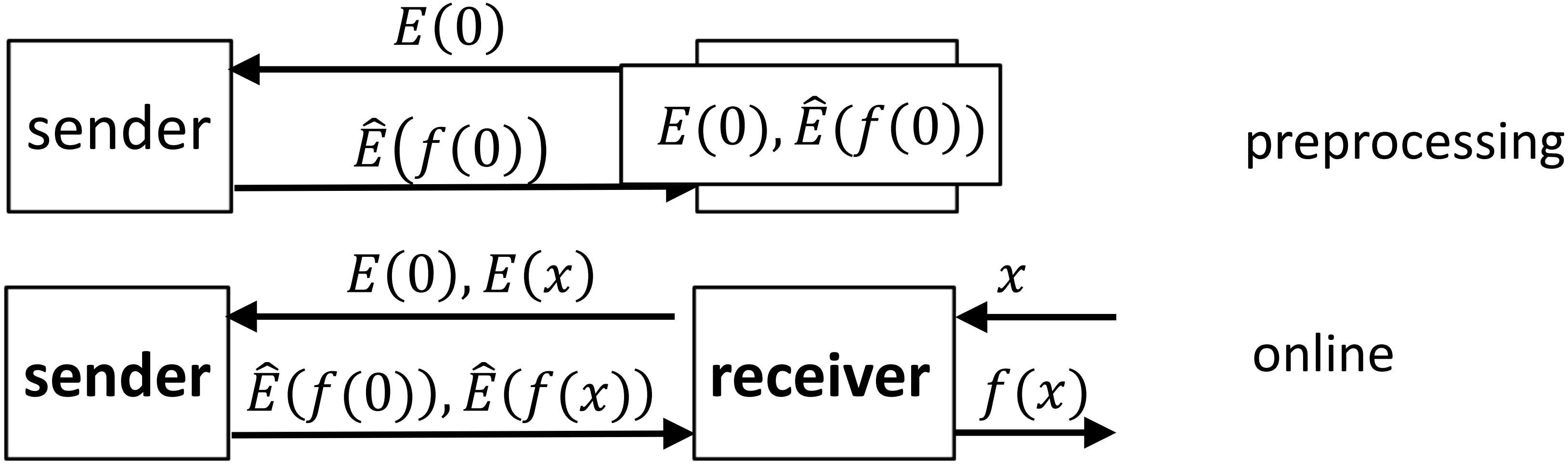


Succinct Function Commitment (SFC)

Thm: FHE ⇒ 4-msg SFC for ANY polytime function

IDEA:

STEP 1: … and "delegate" its preprocessing phase

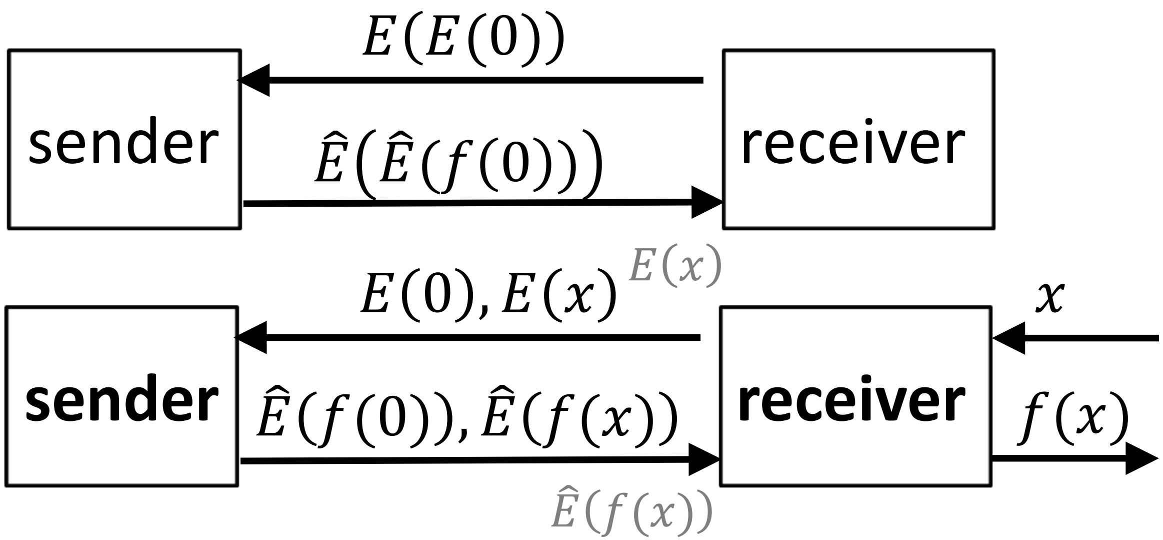


Succinct Function Commitment (SFC)

Thm: FHE $\Rightarrow$ 4-msg SFC for ANY polytime function

IDEA:

STEP 1: … and “delegate” its preprocessing phase



~~preprocessing~~

succinct

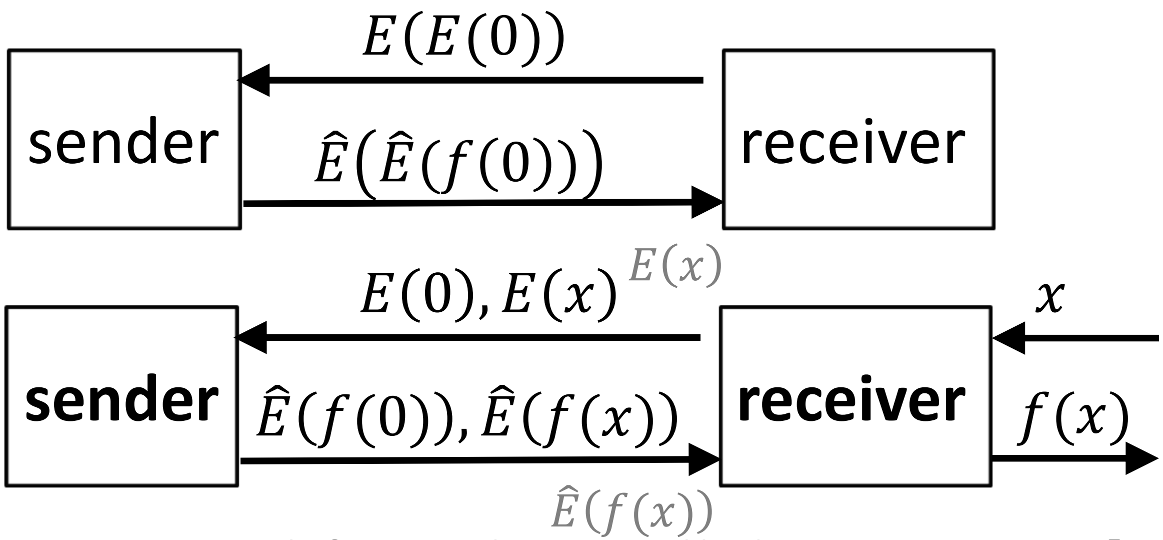
~~online~~

Succinct Function Commitment (SFC)

Thm: FHE ⇒ 4-msg SFC for ANY polytime function

IDEA:

STEP 1: … and "delegate" its preprocessing phase



~~preprocessing~~

fully succinct

~~online~~

STEP 2: amplify with parallel repetition [Hai09,CL10]

Tackled second challenge. ✓

The Role of MIPs

Thm: MIP + SFC ⇒ complexity-preserving 4-msg succinct arguments

What about SNARKs?

[Dwork et al., ’04]: *MIP + PIR unlikely to work*

Thm: MIP + FHE* ⇒ complexity-preserving SNARKs

(FHE* ≈ FHE where homomorphic ops. are extractable)

In fact, can ``squash’’ any public-coin interactive argument (and not just proofs as in [KR09])

Follow-Up

[Bitansky, Canetti, Chiesa, Tromer, EPRINT 12]

<u>any</u> SNARK $\Rightarrow$ complexity-preserving SNARK
& proof-carrying data

even if has expensive preprocessing!

Want More?

See paper for details & interesting open problems!

THANKS!

http://eprint.iacr.org/2012/461