Succinct Arguments From Linear Interactive Proofs

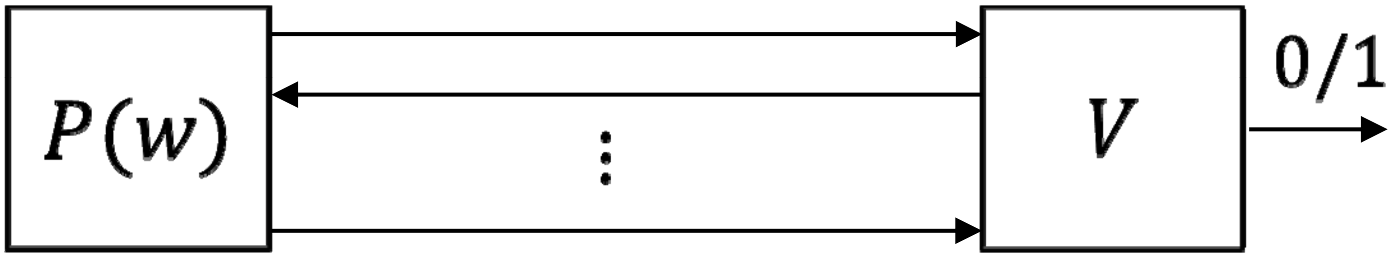
Nir Bitansky **Alessandro Chiesa** Yuval Ishai

Rafail Ostrovsky Omer Paneth

Verifying NP Computations Fast

∃ w s.t. $M(x, w)$ accepts in at most T steps?

statement
(M, x, T)



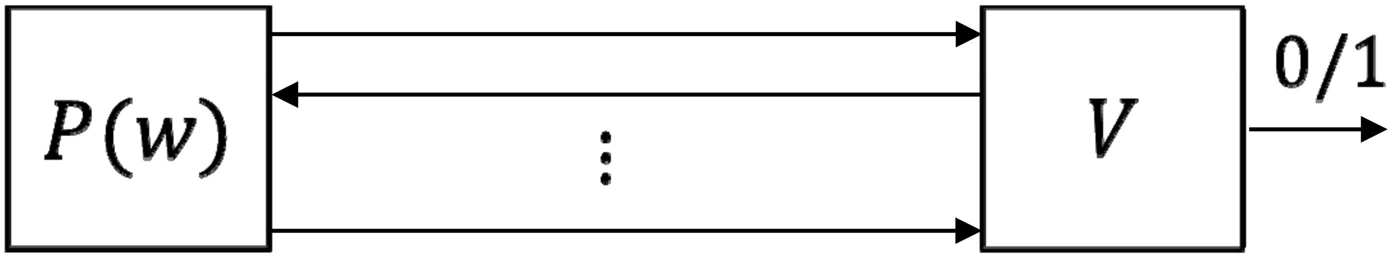
efficiency: $\mathrm{TIME}(P) = T^2$
$\mathrm{TIME}(\mathrm{V}) = (|M| + |x| + \log T)^2$

- completeness: $(M, x, T) \in L_U \rightarrow \Pr\left[\langle P(w), V_{(M,x,T)}\rangle = 1\right] = 1$
- soundness: $(M, x, T) \notin L_U \rightarrow \Pr\left[\langle P^*, V_{(M,x,T)}\rangle = 1\right] = \text{small}$
  for every P^*

Verifying NP Computations Fast

$\exists\, w$ s.t. $M(x,w)$ accepts in at most T steps?

statement
(M,x,T)

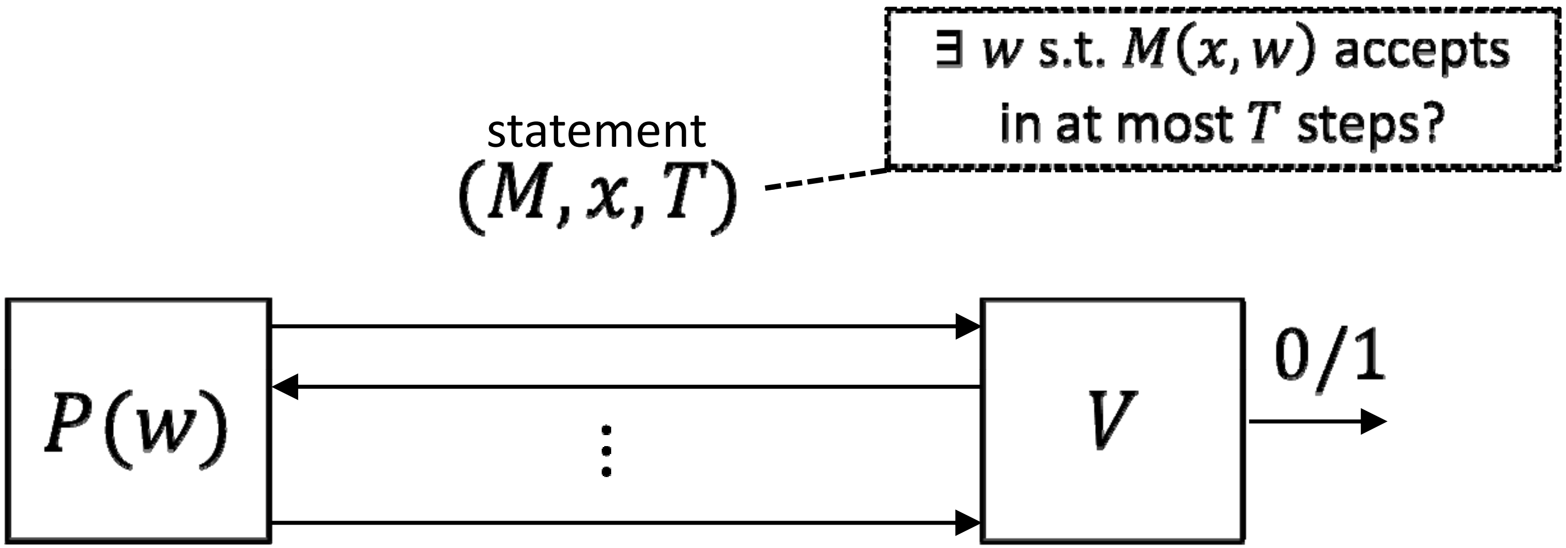


efficiency: $\mathrm{TIME}(P) = k^3 T^2$
$\mathrm{TIME}(V) = k^4(|M| + |x| + \log T)^2$

- completeness: $(M,x,T) \in L_U \rightarrow \Pr\left[\langle P(w), V_{(M,x,T)}\rangle = 1\right] = 1$
- soundness: $(M,x,T) \notin L_U \rightarrow \Pr\left[\langle P^*, V_{(M,x,T)}\rangle = 1\right] = \mathrm{negl}(k)$
  for every poly(k)-size P^*

[BHZ,GH,GVW,Wee]

Verifying NP Computations Fast



SUCCINCT ARGUMENT

efficiency: $\mathrm{TIME}(P) = k^3 T^2$

$\mathrm{TIME}(\mathrm{V}) = k^4(|M| + |x| + \log T)^2$

- completeness: $(M, x, T) \in L_U \rightarrow \Pr\left[\langle P(w), V_{(M,x,T)}\rangle = 1\right] = 1$
- soundness: $(M, x, T) \notin L_U \rightarrow \Pr\left[\langle P^*, V_{(M,x,T)}\rangle = 1\right] = \mathrm{negl}(k)$
  for every poly(k)-size P^*

[BHZ,GH,GVW,Wee]

WHAT KINDS OF SUCCINCT ARGUMENTS ARE THERE?

[Kilian]

tools: PCP system + collision-resistant hashing

1 offline message
3 online messages

public coins

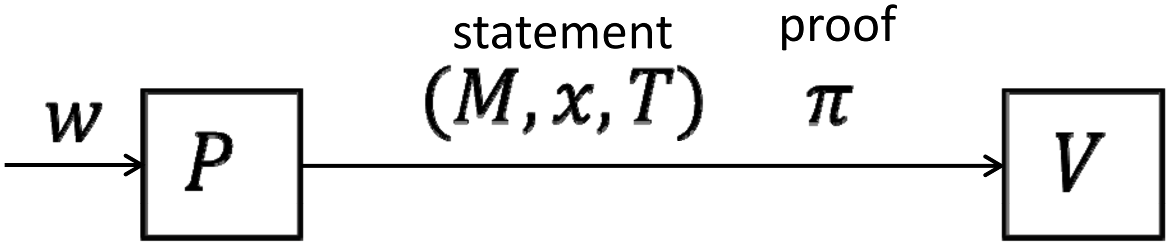
[Micali]

apply Fiat-Shamir paradigm in the Random Oracle model

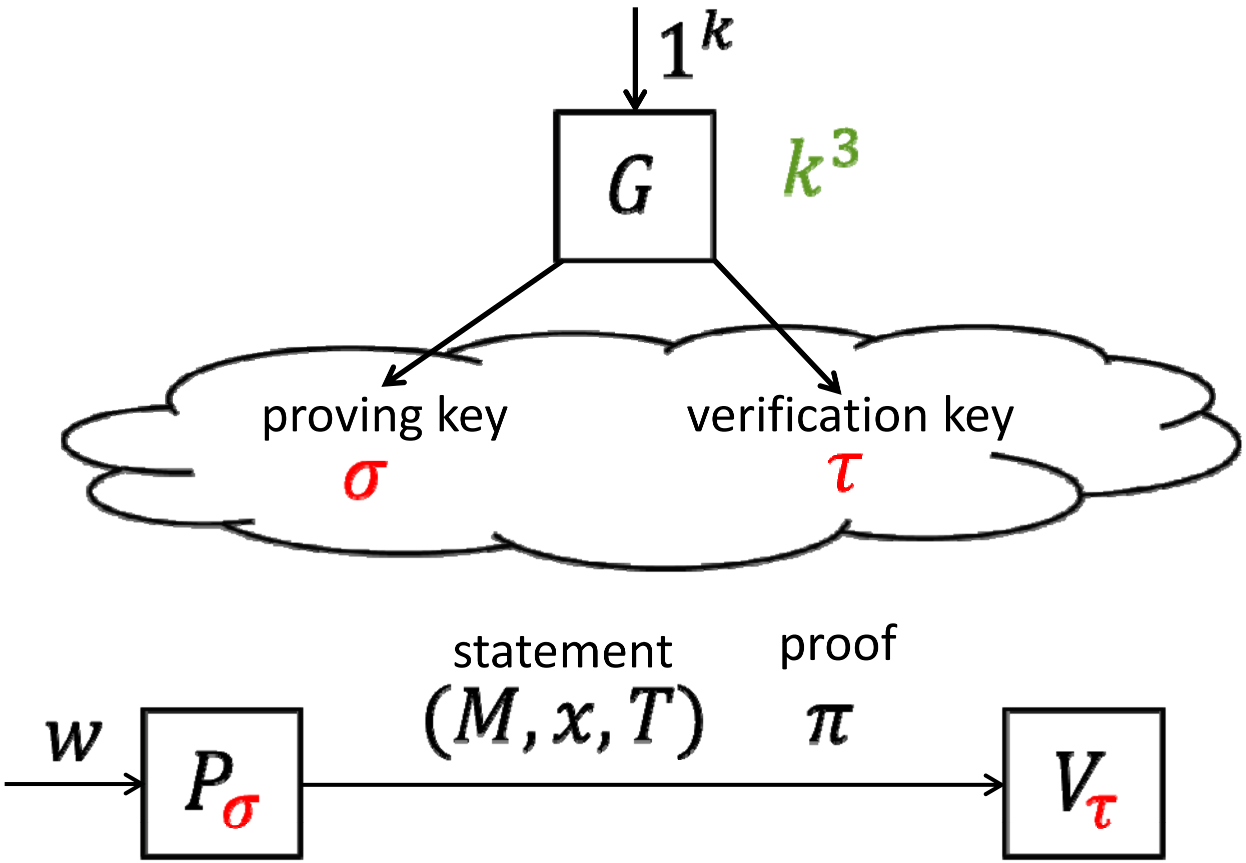
1 non-interactive & publicly-verifiable message

CAN WE REDUCE
ONLINE MESSAGES
W/O RANDOM ORACLES?

Succinct Non-Interactive Arguments (SNARGs)



Succinct Non-Interactive Arguments (SNARGs)



have privately-verifiable constructions
under relatively-clean (albeit non-falsifiable) assumptions
[DL] [Mie] [BCCTa] [DFH] [GLR] [BC]

Probabilistic Checking & Succinct Arguments

A VERY PRODUCTIVE PARADIGM

Step 1: information-theoretic probabilistic checking, in a model where the prover is restricted in some form

Step 2: use cryptography to force the restriction

EXAMPLES

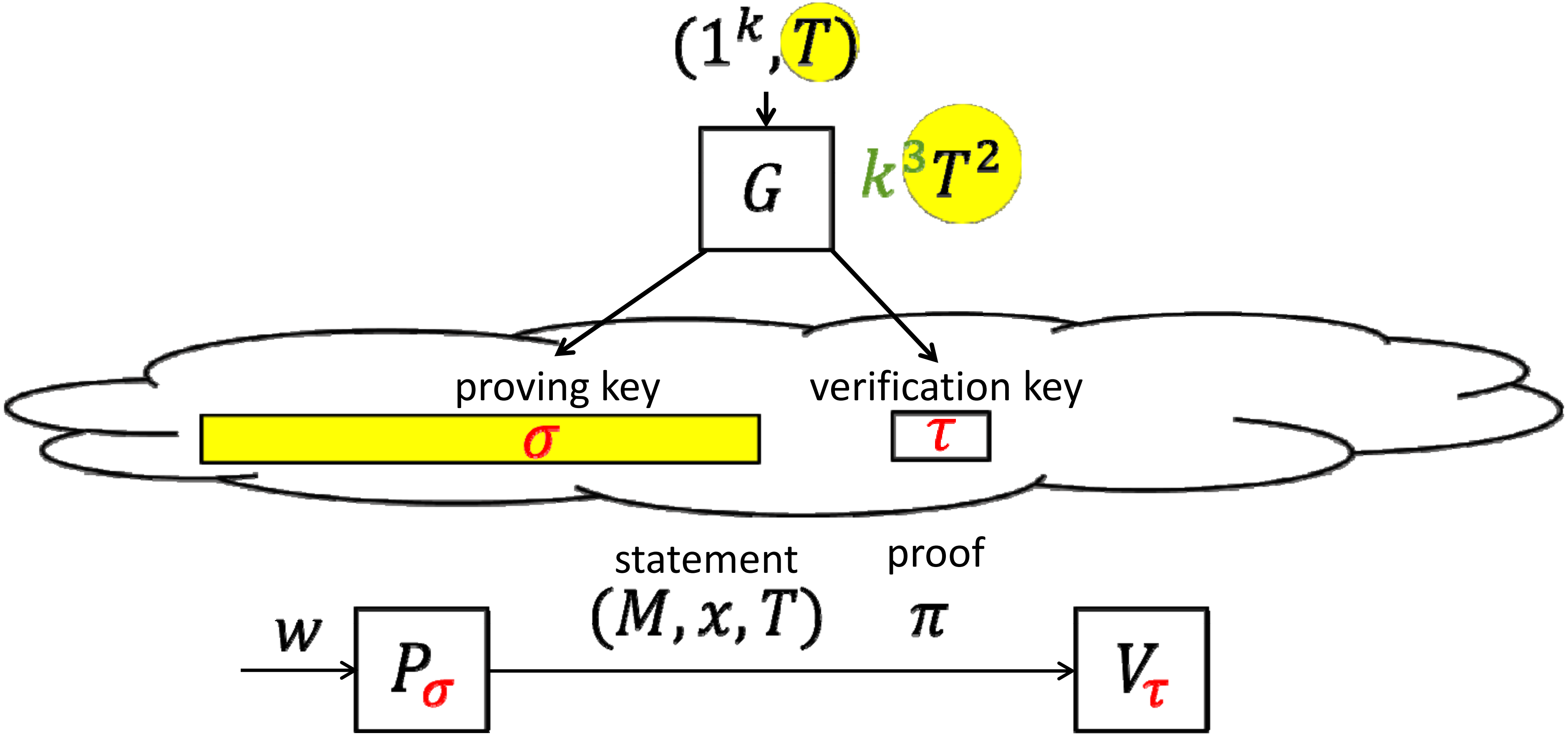
Step 1 = design a PCP
Step 2 = force prover to commit to a PCP

Step 1 = design a no-signaling MIP
Step 2 = force prover to act as no-signaling provers

Step 1 = design an MIP
Step 2 = force prover to act as non-communicating provers

...

TODAY: Preprocessing SNARGs



Preprocessing SNARGs

- setup work is amortized over MANY proofs
- can obtain public verifiability [Groth,Lipmaa,GGPR]
- lead to constructions w/o expensive preprocessing [BCCTb]
  (provided the SNARG has a natural POK)

step 1: reduce CircuitSAT to algebraic satisfaction problem
step 2: use crypto to succinctly verify the latter

surprising: do not seem to rely on probabilistic checking!

OUR CONTRIBUTIONS

THIS WORK

give a general recipe to construct preprocessing SNARGs;
the recipe is a new instantiation of the paradigm

Specifically:

Step 1: (information-theoretic)
design a 2-message linear interactive proof (LIP)

Step 2: (cryptographic)
force prover to act as a linear function

results for Step 1: constructions of succinct LIPs
results for Step 2: compilers for private and public cases

- simpler and more efficient preprocessing SNARGs
- re-interpret previous constructions from new perspective

Linear PCPs

A PCP where the proof oracle is a linear function.
Previously used in another instantiation of paradigm:

[IKO]

linear PCP

$\downarrow$ + linearity testing

strong
linear PCP

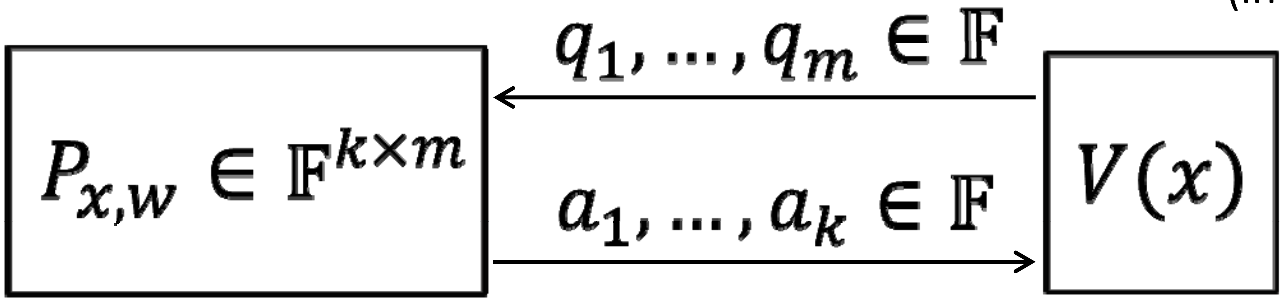
$\downarrow$ + function commitment

4-msg NP argument
with small communication

Linear Interactive Proofs (LIPs)

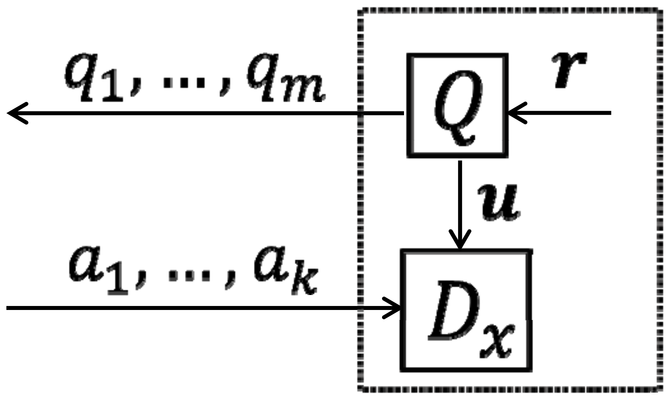
The prover is **algebraically bounded**: specifically, linear.

(in both completeness and soundness!)

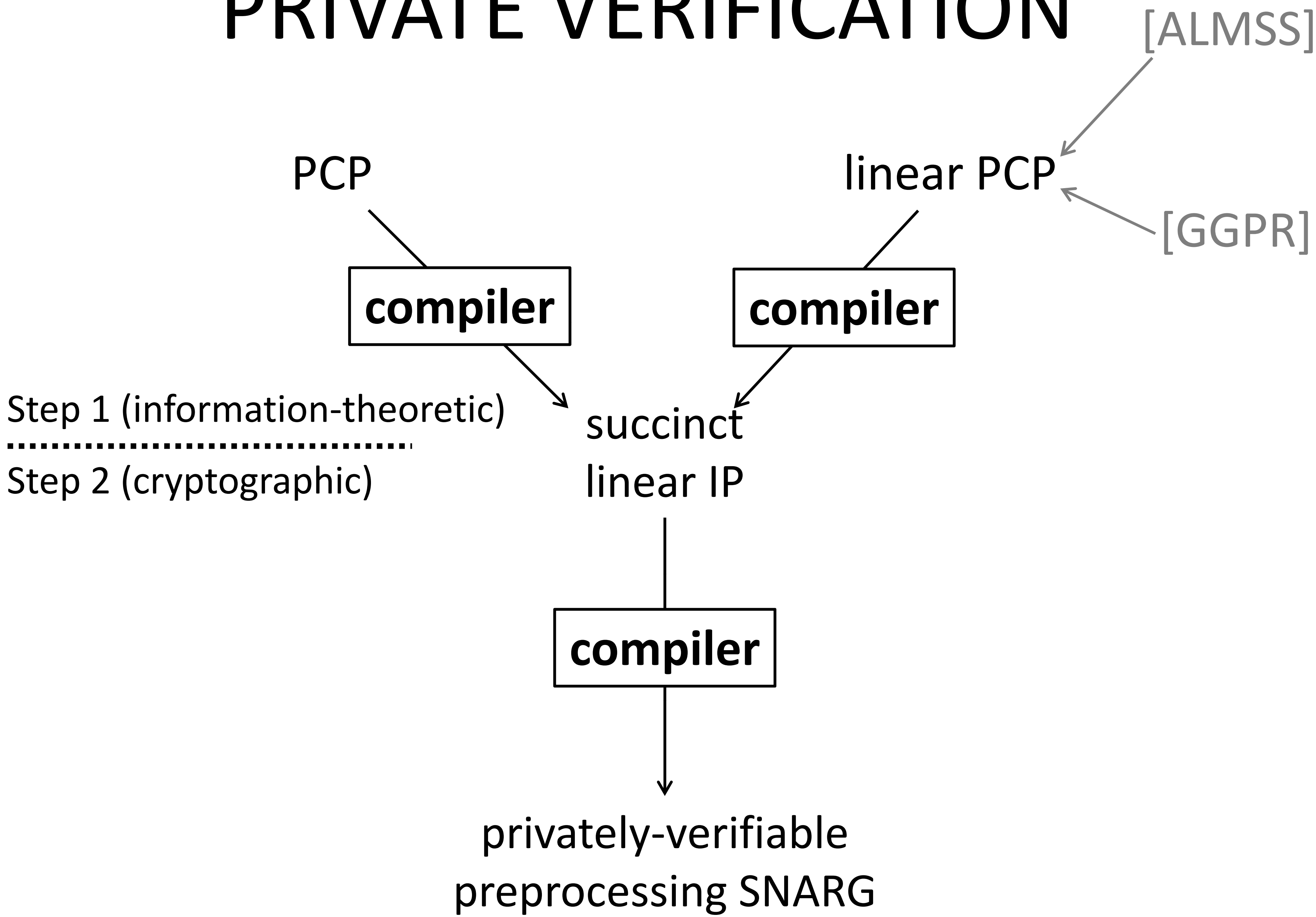


We are interested in LIPs that are *input oblivious*:

$V = (Q, D)$ s.t.

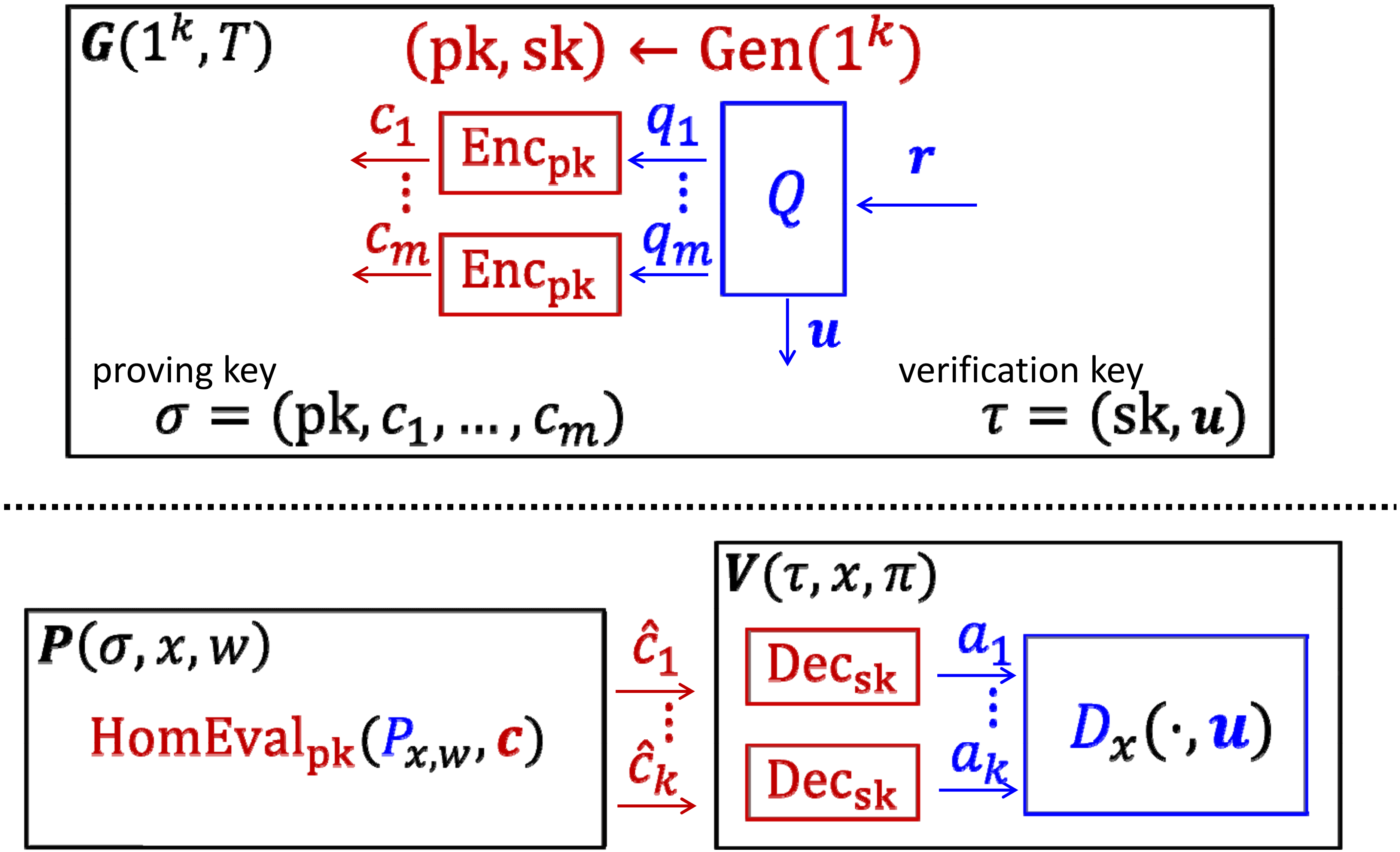


PRIVATE VERIFICATION



Step 2: From LIP To pp SNARG

(private verification)



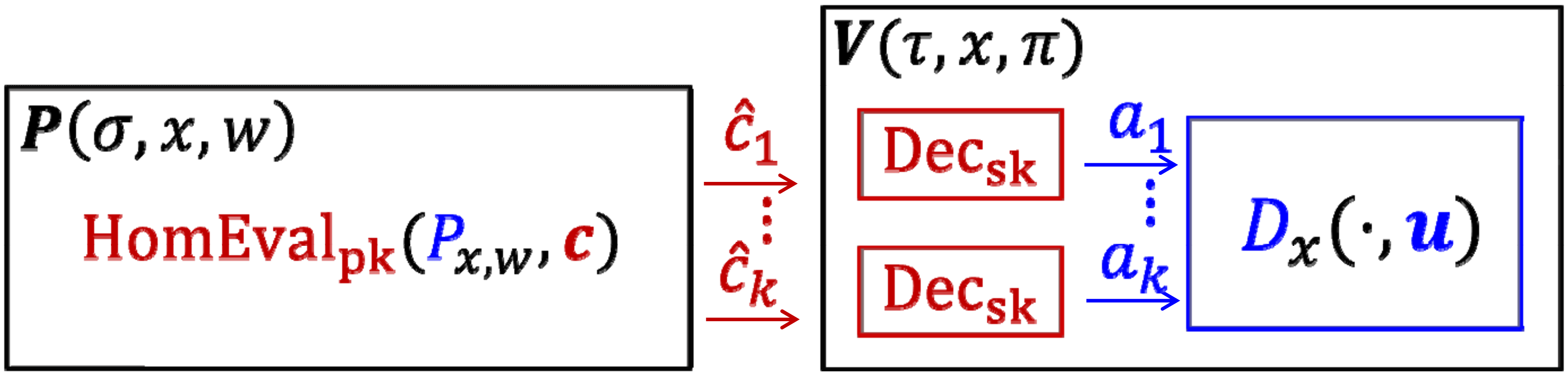
Step 2: From LIP To pp SNARG
(private verification)

Linear Targeted Malleability (~[BSW])
encryption scheme that ONLY allows
$\mathbb{F}$-linear homomorphic operations
(e.g., knowledge variant of Paillier)

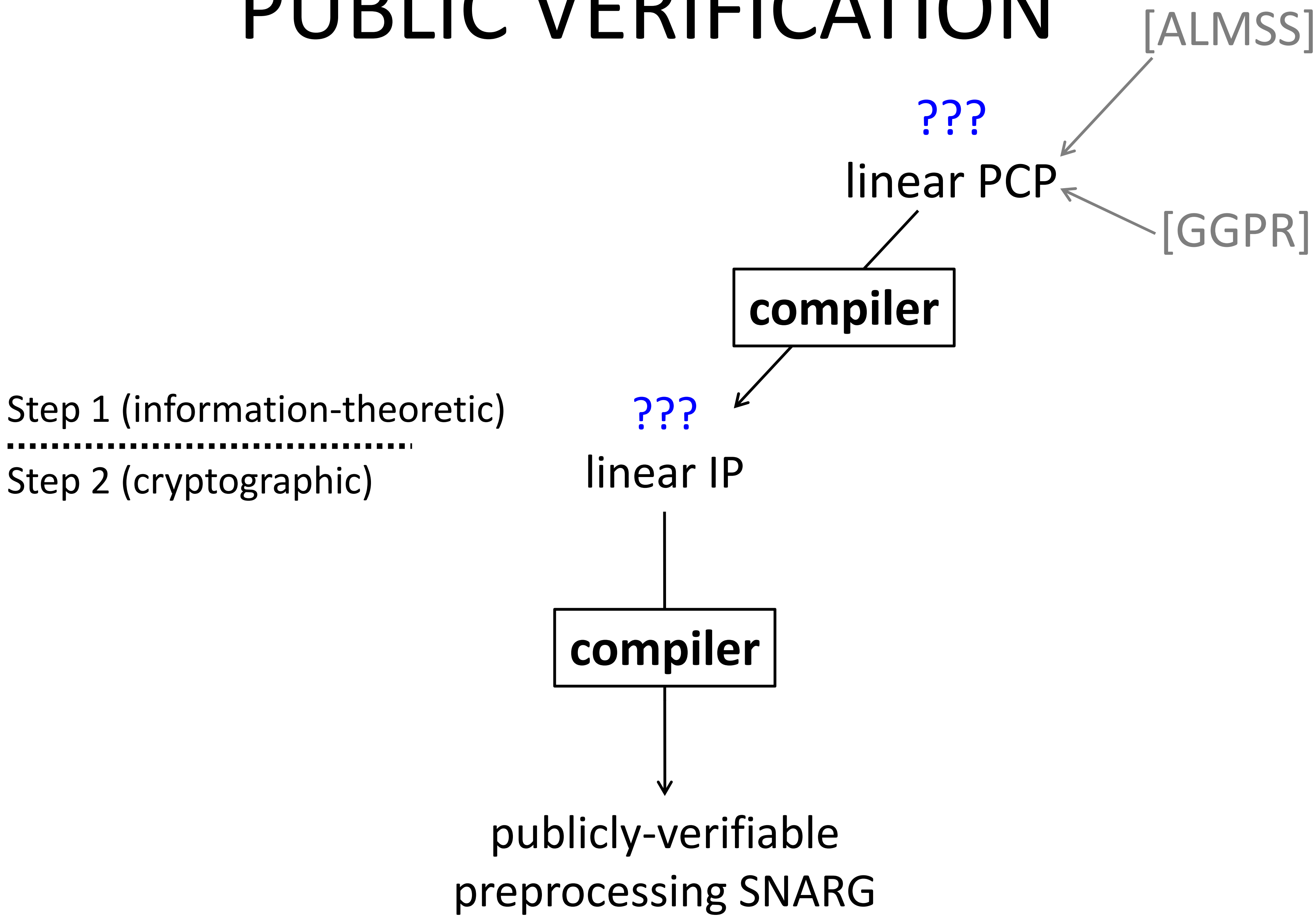
non-falsifiable assumption
(somewhat justified by [GW])

proving key
$$\sigma = (\mathrm{pk}, c_1, \ldots, c_m)$$

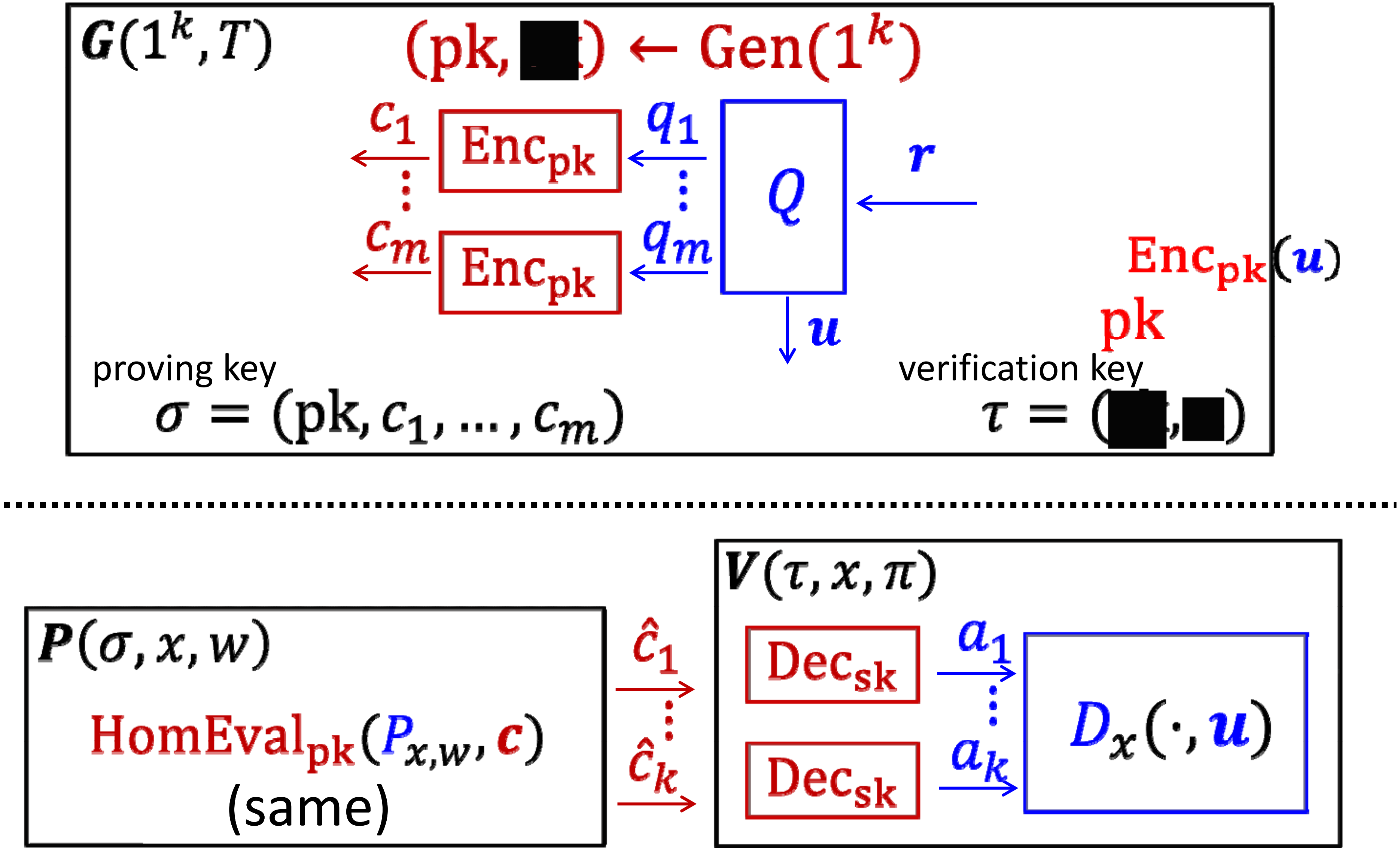
verification key
$$\tau = (\mathrm{sk}, \boldsymbol{u})$$



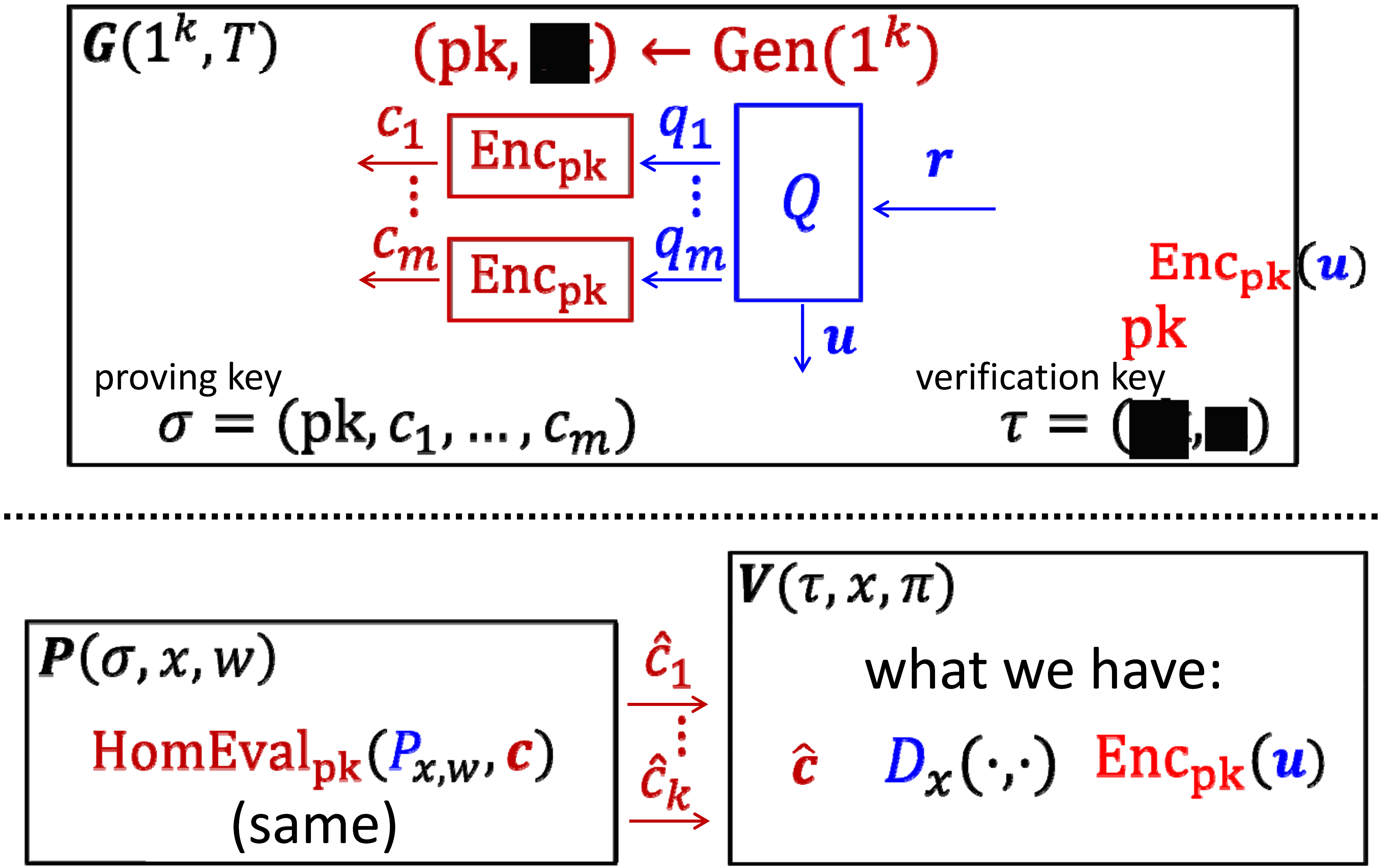
PUBLIC VERIFICATION



Step 2: From LIP To pp SNARG (sketch)



Step 2: From LIP To pp SNARG (sketch)



Step 2: From LIP To pp SNARG (sketch)

$G(1^k, T)$

$(\text{pk}, $ [redacted] $) \leftarrow \text{Gen}(1^k)$

$c_1 \leftarrow \boxed{\text{Enc}_{\text{pk}}} \leftarrow q_1$ ⋮ $c_m \leftarrow \boxed{\text{Enc}_{\text{pk}}} \leftarrow q_m$ $\boxed{Q} \xleftarrow{} r$

$\boxed{Q} \rightarrow u$

$\text{Enc}_{\text{pk}}(u)$

pk

proving key

$\sigma = (\text{pk}, c_1, \ldots, c_m)$

verification key

$\tau = ($ [redacted] $)$

$P(\sigma, x, w)$

$\text{HomEval}_{\text{pk}}(P_{x,w}, c)$

(same)

$\hat{c}_1 \rightarrow$ ⋮ $\hat{c}_k \rightarrow$

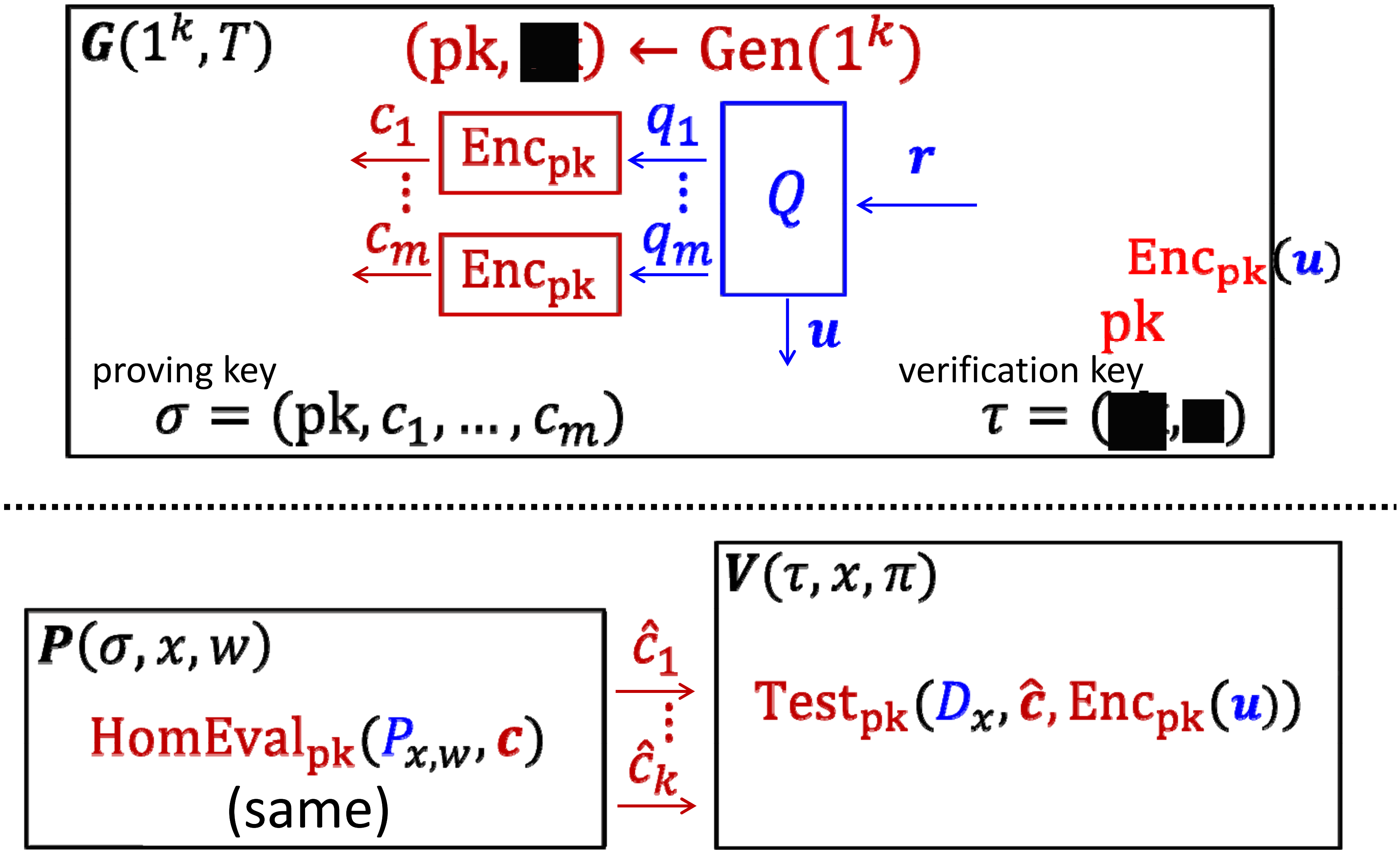
$V(\tau, x, \pi)$

need to test roots:

$\text{HomEval}_{\text{pk}}(D_x, \hat{c}, \text{Enc}_{\text{pk}}(u))$

$\in \text{Enc}_{\text{pk}}(0)$?

Step 2: From LIP To pp SNARG (sketch)



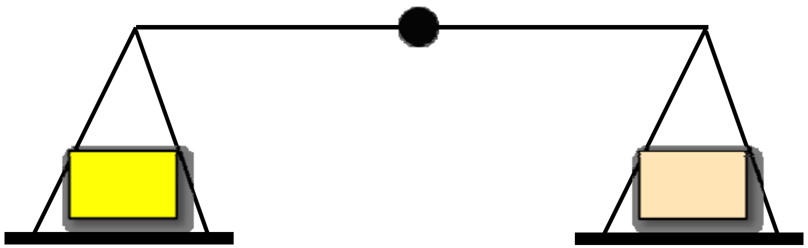
Publicly-Verifiable pp SNARGs

???
linear
interactive
proof → **compiler** → publicly-verifiable preprocessing SNARG

uses bilinear maps + KEA
gives us Encoding with:
- test quadratic predicates
- certain one-way hardness
- *almost-linear* homomorph.

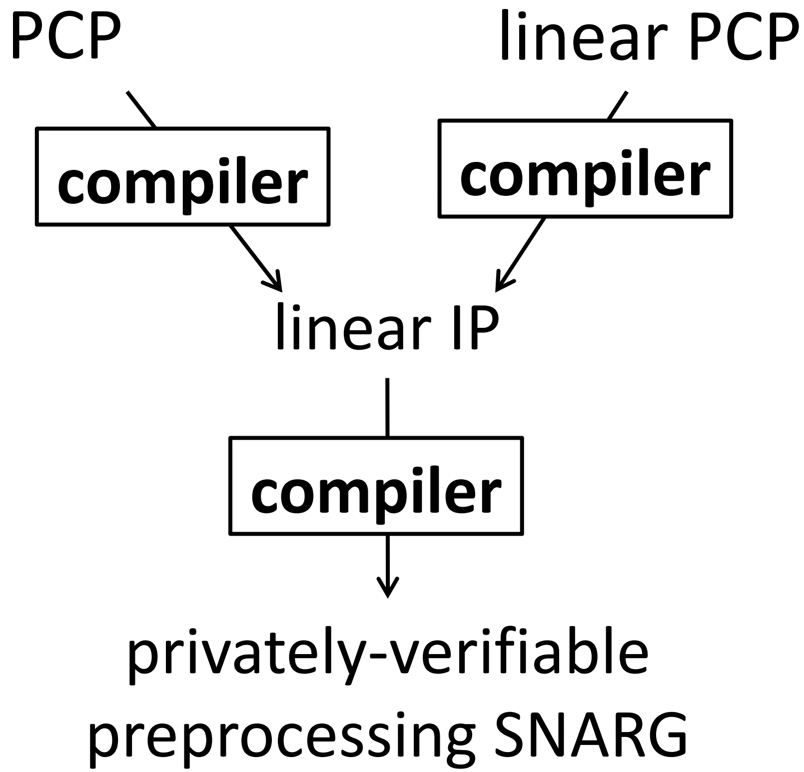
???
=
query/decision algorithms
are <u>low-degree polynomials</u>

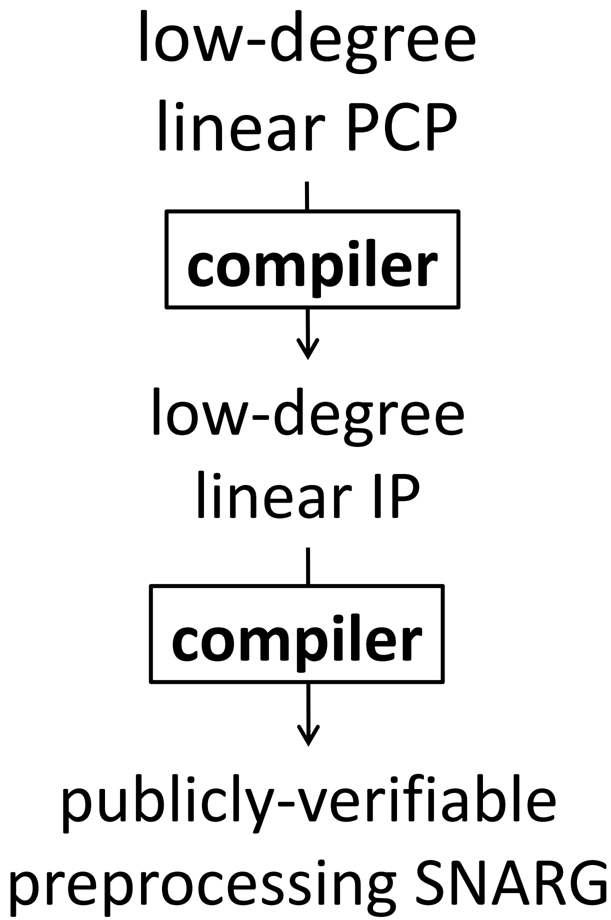
a balancing act:



Summary

- A simple and motivated recipe





- See paper for more (including ZK generic transformation)

THANKS!

http://eprint.iacr.org/2012/718

THANKS!

http://eprint.iacr.org/2012/718

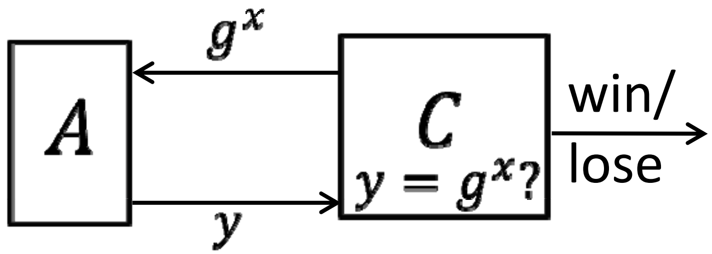
FOLLOWING ARE OLD SLIDES

[Naor 03] [GW11]

Falsifiable Assumptions

An assumption is *falsifiable* if a challenger can efficiently test, via an interactive protocol, whether an efficient adversary breaks it.

Example: DL $\equiv$ " $\max_A \Pr[\langle A, C\rangle = \mathrm{win}] \leq \mathrm{negl}$ "



Other examples:

DDH, RSA, LWE, QR, …

Unexamples (i.e., *non-falsifiable*):

- (P, V) is ZK (not a game: requires a simulator)
- knowledge of exponent: given random $(g, h) \in G \times G$ can't efficiently generate (g^β, h^β) without "knowing" β

(not a game: requires an extractor)

[Dam91] [HT98]

non-falsifiable assumptions are not all equally strong/complex

By investigating such assumptions and their power, we may:

- identify “nice” NF assumptions

- discover entirely new constructions

Bilinear Techniques & Preprocessing

coming from a line of work on NIZKs [Groth,GOS,AF]
seeking to minimize #group elements in a NI proof

very different construction approach

	supported functions	# messages		offline work is cheap?	secure w. verifier oracle?	publicly verifiable?	main assumption
		offline	online				
[Groth] [Lipmaa] [GGPR]	NP	1	1	NO	YES	YES	“KEA”

the verifier must preprocess
the circuit to make a CRS

1)

Bootstrapping SNARKs:

A New Path To the Holy Grail

[Bitansky, Canetti, C, Tromer]

Bootstrapping SNAR<u>K</u>s

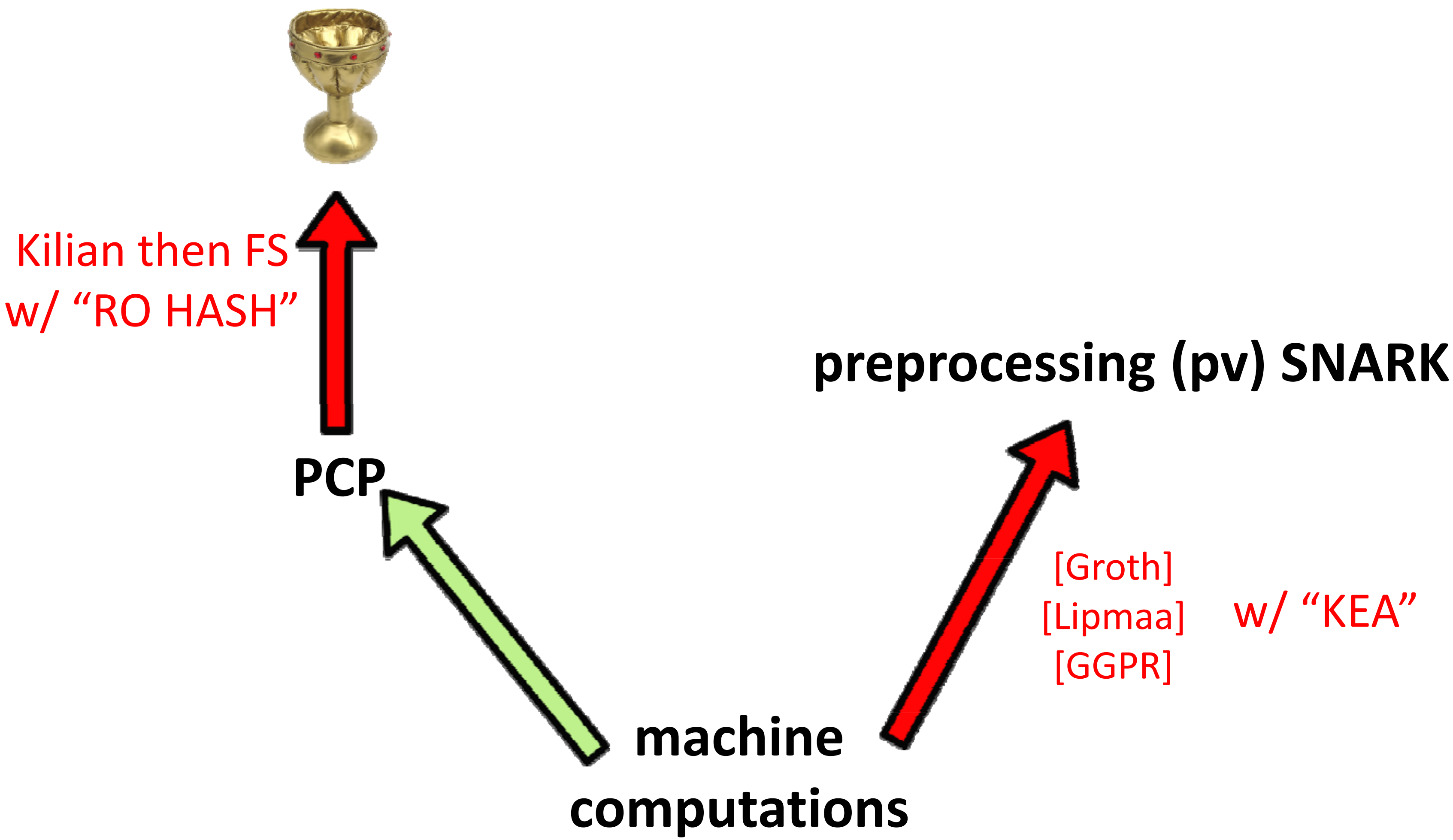
provided a SNARG has a natural *proof of knowledge*
(all known ones do)
its efficiency properties can be improved to “optimal”

	supported functions	# messages offline	# messages online	offline work is cheap?	secure w. verifier oracle?	publicly verifiable?	main assumption
[BCCTb]	NP	1	1	X	X	X	X
	NP	**1**	**1**	**YES**	**X**	**X**	**X + CRH**

ADDITIONAL EFFICIENCY BONUSES

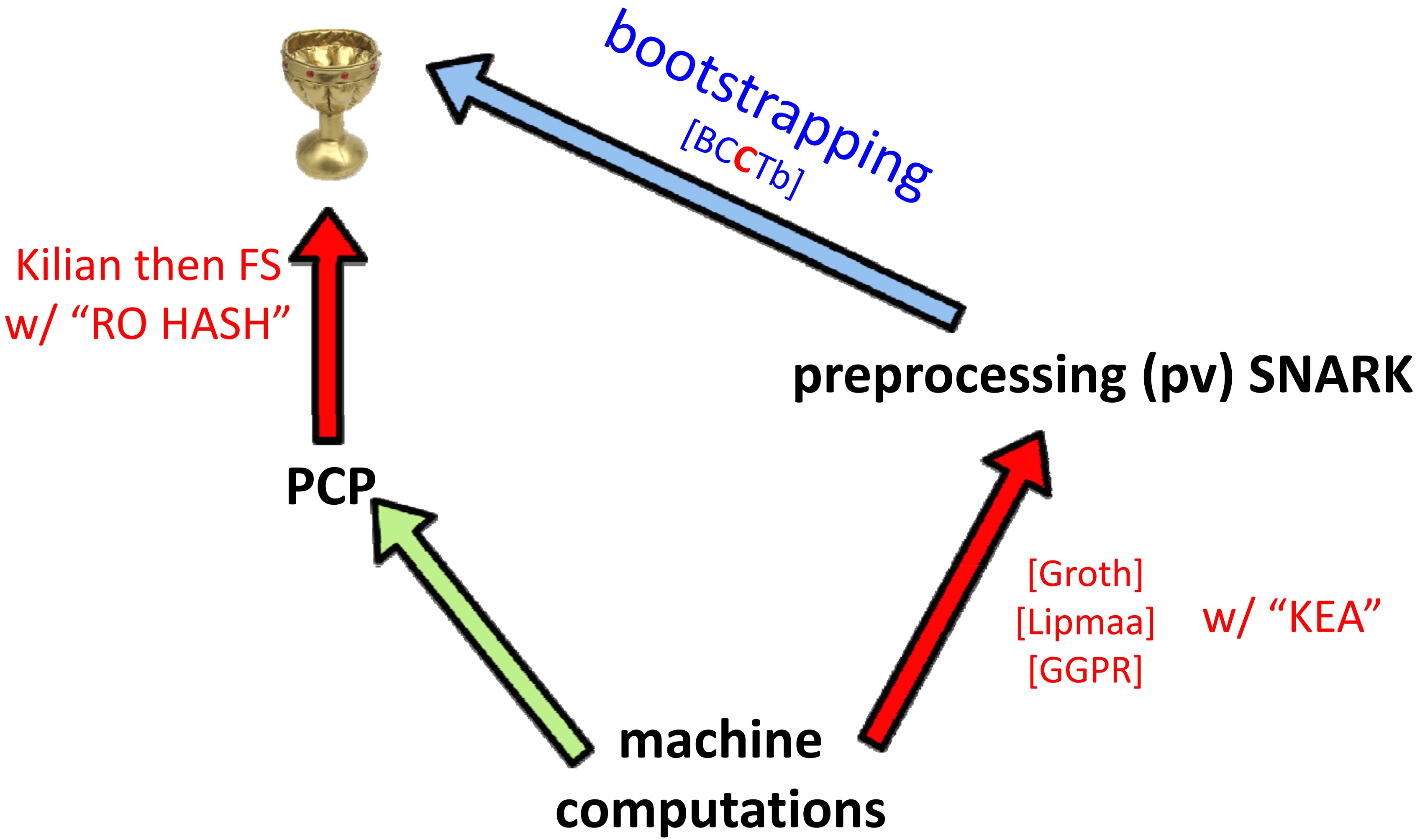
- <u>*complexity preservation*</u>:
  prover in $T \cdot \mathrm{poly}(k)$ time for T time computations
  $S \cdot \mathrm{poly}(k)$ space S space
- transformation does not invoke the PCP theorem

The Old Path

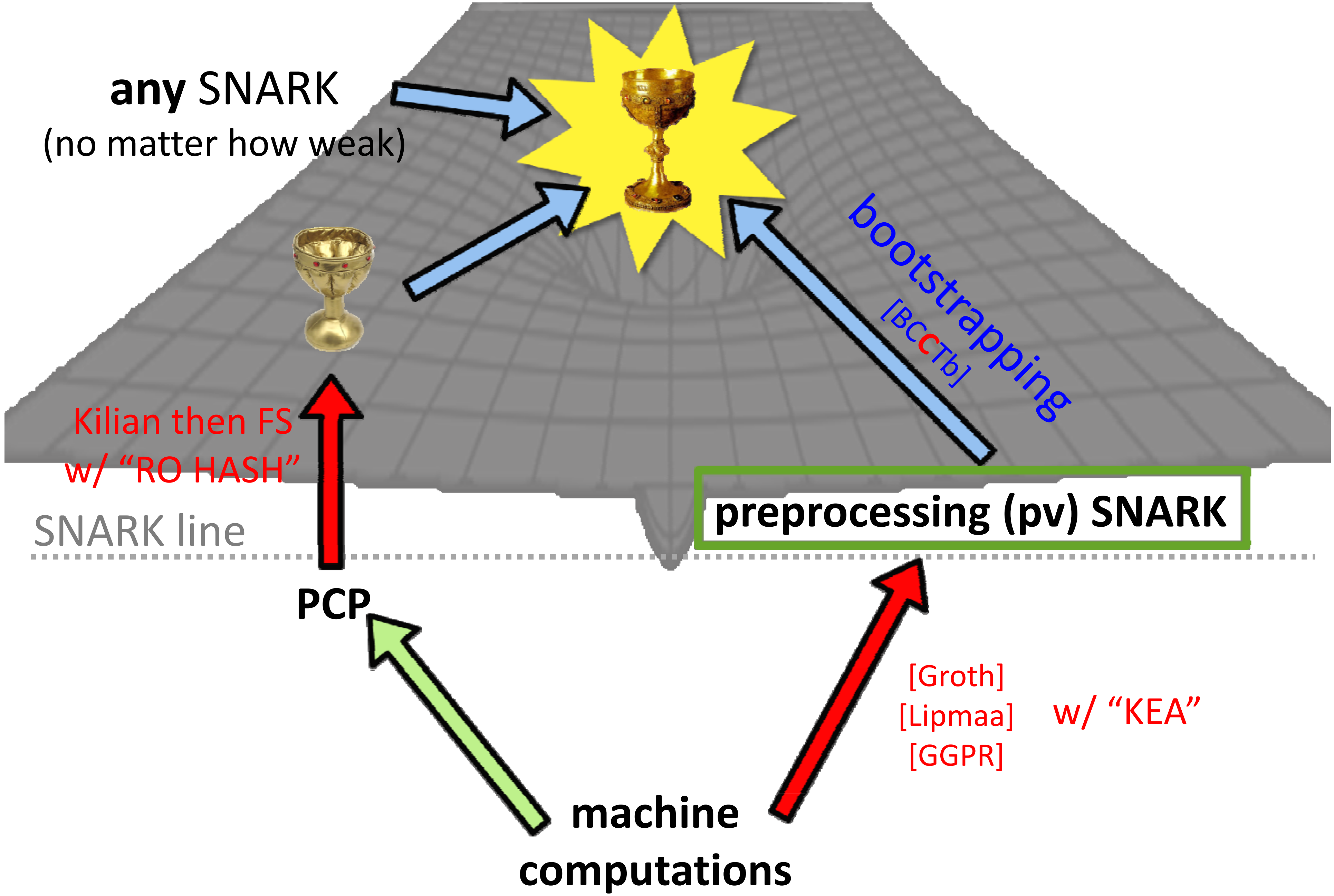


Kilian then FS
w/ "RO HASH"
PCP
preprocessing (pv) SNARK
[Groth]
[Lipmaa]
[GGPR]
w/ "KEA"
machine
computations

A New Path


bootstrapping
[BCCTb]
Kilian then FS
w/ “RO HASH”
preprocessing (pv) SNARK
PCP
[Groth]
[Lipmaa]
[GGPR]
w/ “KEA”
machine computations

A New & Better Path



2)

A General Technique For Making Preprocessing SNARKs

[Bitansky, **C**, Ishai, Ostrovsky, Paneth]

How Make Preprocessing SNARKs?

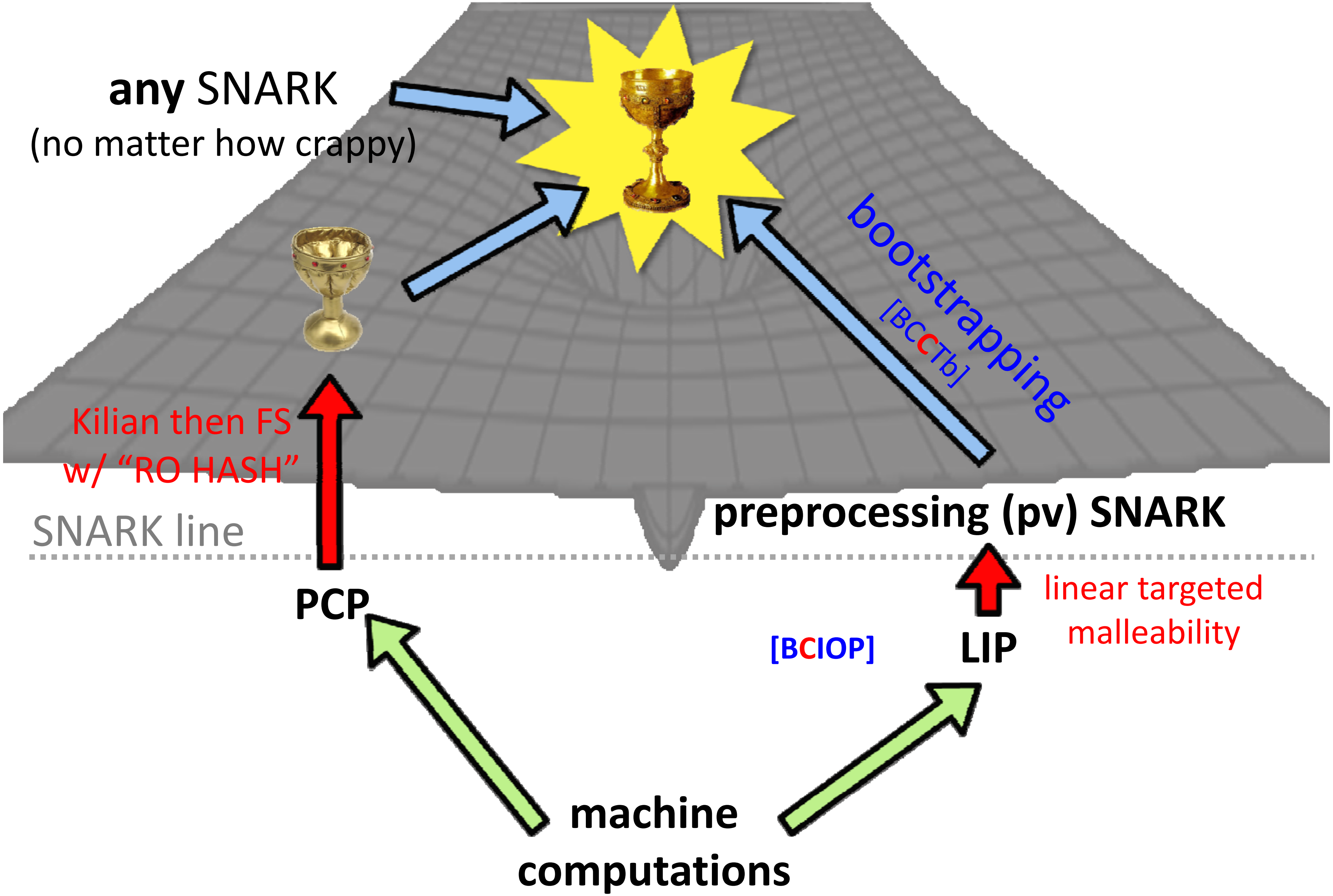
	supported functions	# messages		offline work is cheap?	secure w. verifier oracle?	publicly verifiable?	main assumption
		offline	online				
[Groth] [Lipmaa] [GGPR] **[BCIOP]**	NP	1	1	NO	YES	YES	"KEA"

General technique to make preprocessing SNARKs:

Step 1 = design a 2-message linear interactive proof (LIP)

Step 2 = force prover to act as a linear function

A New & Better Path



REST OF TALK

1) on bootstrapping SNARKs

[Bitansky, Canetti, **C**, Tromer]

2) on making preprocessing SNARKs

[Bitansky, **C**, Ishai, Ostrovsky, Paneth]

1

ON BOOTSTRAPPING SNARKs

Theorem

Suppose CRHs exist.

Then there are efficient T_1 and T_2 such that:

$$\text{any SNARK}$$
$$\Big\Downarrow T_1$$
$$\text{complexity-preserving SNARK}$$
$$\Big\Downarrow T_2$$
$$\text{complexity-preserving PCD}$$

complexity-preserving =
- no preprocessing
- prover has quasi-optimal time & space complexity

high-level intuition
with no abstraction layers for

Theorem' (removing preprocessing)

Suppose CRHs exist.

Then there is an efficient T such that:

preprocessing publicly-verifiable SNARK

$$\Downarrow T$$

publicly-verifiable SNARK

WHAT DO WE DO??

The Core Idea:
bootstrap the SNARK

Main Observation

only need to budget for <u>small</u> computations…

as small as SNARK verification
(plus a bit more)

⇒ inefficiencies are ``localized''
and thus become inexpensive!

2

ON MAKING PREPROCESSING SNARKs

Designing Efficient Arguments

Step 1: information-theoretic probabilistic checking,
in a model where the prover is restricted in some form

Step 2: use cryptography to "implement" the model

EXAMPLES

Step 1 = design a PCP
Step 2 = force prover to commit to a PCP

Step 1 = design a nsMIP
Step 2 = force prover to act as no-signaling provers

Step 1 = design an MIP
Step 2 = force prover to act as non-communicating provers

Designing Efficient Arguments

Step 1: information-theoretic probabilistic checking,
in a model where the prover is restricted in some form

Step 2: use cryptography to “implement” the model

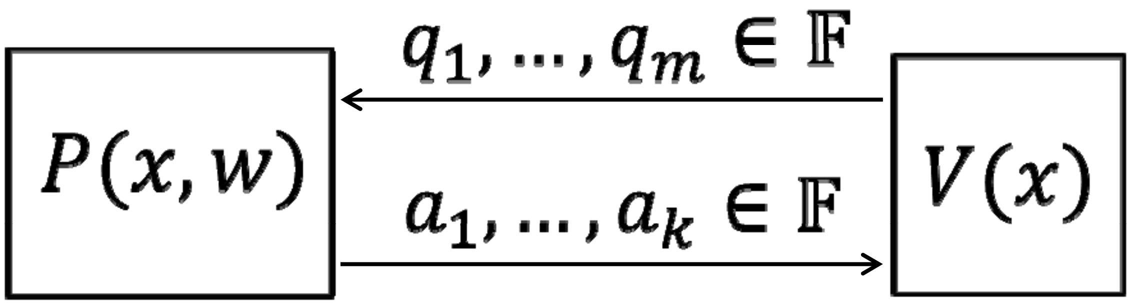
A New Example

Step 1 = design a 2-message linear interactive proof
Step 2 = force prover to act as a linear function

Q1: how to design LIPs with suitable efficiency?
Q2: how to use crypto to make a prover linear?

Linear Interactive Proofs (LIPs)

The prover is **algebraically bounded**: specifically, linear.

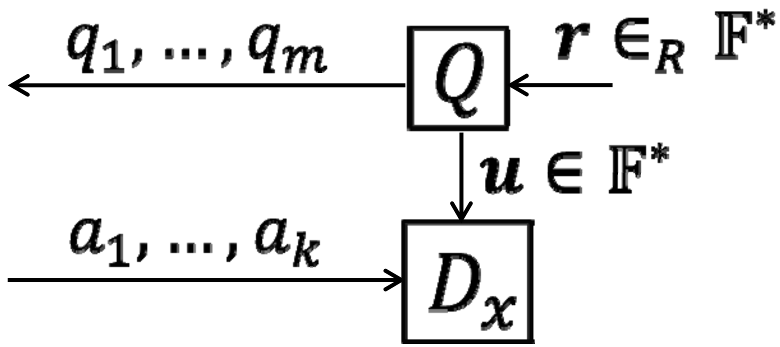


$$\exists\, \Pi \in \mathbb{F}^{k \times m}, \boldsymbol{b} \in \mathbb{F}^{k} \text{ s.t. } \boldsymbol{a} = \Pi \mathbf{q} + \mathbf{b}$$

($\Pi, \boldsymbol{b}$ depend on x, w)

We are interested in LIPs that are:

- *succinct*: $k = O(1)$
- *input oblivious*: $V = (Q, D)$ s.t.

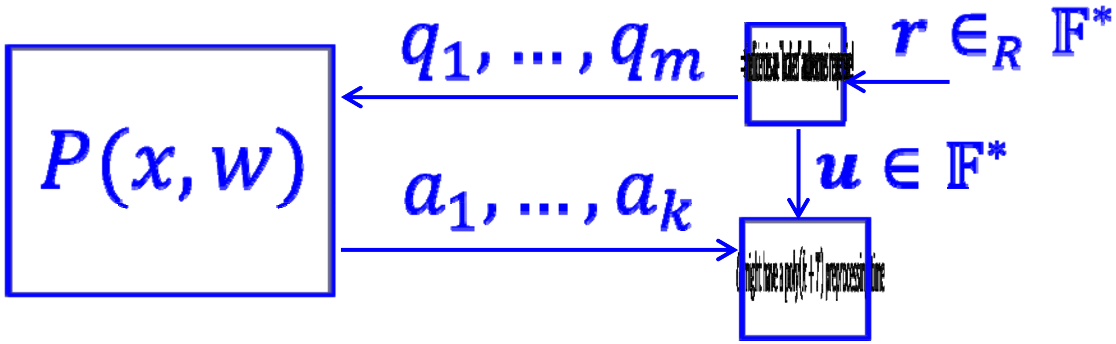


Step 2: Making Provers Linear

WARM UP: from LIP to <u>privately-verifiable</u> pp SNARK

TOOLS:

LIP $(P, (Q, D))$

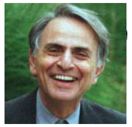


&

"crypto"

Step 2: Making Provers Linear

WARM UP: from LIP to privately-verifiable pp SNARK


$(1^k) \equiv$

$(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathrm{Gen}(1^k)$

$r \in_R \mathbb{F}^*$ → Q → $q_1, \dots, q_m$ → $\mathrm{Enc}_{\mathrm{pk}}$ → $c_1, \dots, c_m$

Q → $\boldsymbol{u} \in \mathbb{F}^*$

proving key

$\sigma = (\mathrm{pk}, c_1, \dots, c_m)$

verification key

$\tau = (\mathrm{sk}, \boldsymbol{u})$

$$\boldsymbol{P}(\sigma, x, w) = \mathrm{HomEval}_{\mathrm{pk}}(P(x, w), \boldsymbol{c})$$

$\hat{c}_1, \dots, \hat{c}_k$ →

$\boldsymbol{V}(\tau, x, \pi)$: $\mathrm{Dec}_{\mathrm{sk}}$ → $a_1, \dots, a_k$ → $D_x(\cdot, \boldsymbol{u})$

Step 2: Making Provers Linear

WARM UP: from LIP to privately-verifiable pp SNARK

Linear Targeted Malleability (~[BSW])

encryption scheme that ONLY allows $\mathbb{F}$-additive homomorphic operations

non-falsifiable assumption

(e.g., Paillier)

proving key

$$\sigma = (\mathrm{pk}, c_1, \dots, c_m)$$

verification key

$$\tau = (\mathrm{sk}, \boldsymbol{u})$$

$$\boldsymbol{P}(\sigma, x, w) = \mathrm{HomEval}_{\mathrm{pk}}(P(x, w), \boldsymbol{c})$$

$\hat{c}_1, \dots, \hat{c}_k$

$\boldsymbol{V}(\tau, x, \pi)$

$\mathrm{Dec}_{\mathrm{sk}}$ $\xrightarrow{a_1}$ … $\mathrm{Dec}_{\mathrm{sk}}$ $\xrightarrow{a_k}$ $D_x(\cdot, \boldsymbol{u})$

Step 2: Making Provers Linear

What happens if we want public verifiability?

Being able to test properties of the prover's answers implies that we must **give up** semantic security.

(1) What notion of security should $\mathrm{Enc}_{\mathbf{pk}}(q_i)$ satisfy?

In particular, security must be preserved even given certain **leakage** on the queries.

(2) What kinds of LIPs then suffice?

Step 2: Making Provers Linear

(1) What notion of security should $\mathrm{Enc}_{\mathrm{pk}}(q_i)$ satisfy?

(1) Δ-power OW

$$s \not\leftarrow A(\mathrm{pk}, \mathrm{Enc}_{\mathrm{pk}}(s), \mathrm{Enc}_{\mathrm{pk}}(s^2), \dots, \mathrm{Enc}_{\mathrm{pk}}(s^\Delta))$$

$$p^* \not\leftarrow A(\mathrm{pk}, \mathrm{Enc}_{\mathrm{pk}}(p_1(s)), \dots, \mathrm{Enc}_{\mathrm{pk}}(p_\ell(s)))$$

$p^*(s) = 0, p^* \neq 0$ $\qquad$ $p_1, \dots, p_\ell$ of degree Δ

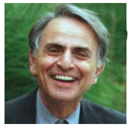
(2) What kinds of LIPs then suffice?

(2) LIPs with low-degree verifiers

Def: an LIP $(P, (Q, D))$ has degree (d_Q, d_D) if
i) $Q(\boldsymbol{r})$ has total degree at most d_Q
ii) $D_x(\boldsymbol{u}, \boldsymbol{a})$ has total degree at most d_D
(Q and D_x are multivalued multivariate polynomials over $\mathbb{F}$)

Step 2: Making Provers Linear

SKETCH

 $(1^k) \equiv$

$(\text{pk}, \quad) \leftarrow \text{Gen}(1^k)$

$r \in_R \mathbb{F}^*$ → Q → $q_1, \ldots, q_m$ → Enc_{pk} → $c_1, \ldots, c_m$

$u \in \mathbb{F}^*$

$\text{AEnc}_{\text{pk}}(u)$

pk

proving key

$\sigma = (\text{pk}, c_1, \ldots, c_m)$

verification key

$\tau = (\quad , \quad)$

$P(\sigma, x, w)$
$=$
$\text{HomEval}_{\text{pk}}(P(x, w), \boldsymbol{c})$

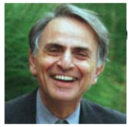
$\hat{c}_1, \ldots, \hat{c}_k$ →

$V(\tau, x, \pi)$

Dec_{sk} → $a_1, \ldots, a_k$ → $D_x(\cdot, \boldsymbol{u})$

Step 2: Making Provers Linear

SKETCH

$(1^k) \equiv$

$(\text{pk}, \;) \leftarrow \text{Gen}(1^k)$

$\leftarrow \text{Enc}_{\text{pk}} \leftarrow q_1 \leftarrow Q \leftarrow r \in_R \mathbb{F}^*$

$\vdots$

$c_m \leftarrow \text{Enc}_{\text{pk}} \leftarrow q_m$

$\downarrow u \in \mathbb{F}^*$

$\text{AEnc}_{\text{pk}}(u)$

pk

proving key

$\sigma = (\text{pk}, c_1, \ldots, c_m)$

verification key

$\tau = (\;, \;)$

$P(\sigma, x, w)$
$=$
$\text{HomEval}_{\text{pk}}(P(x, w), c)$

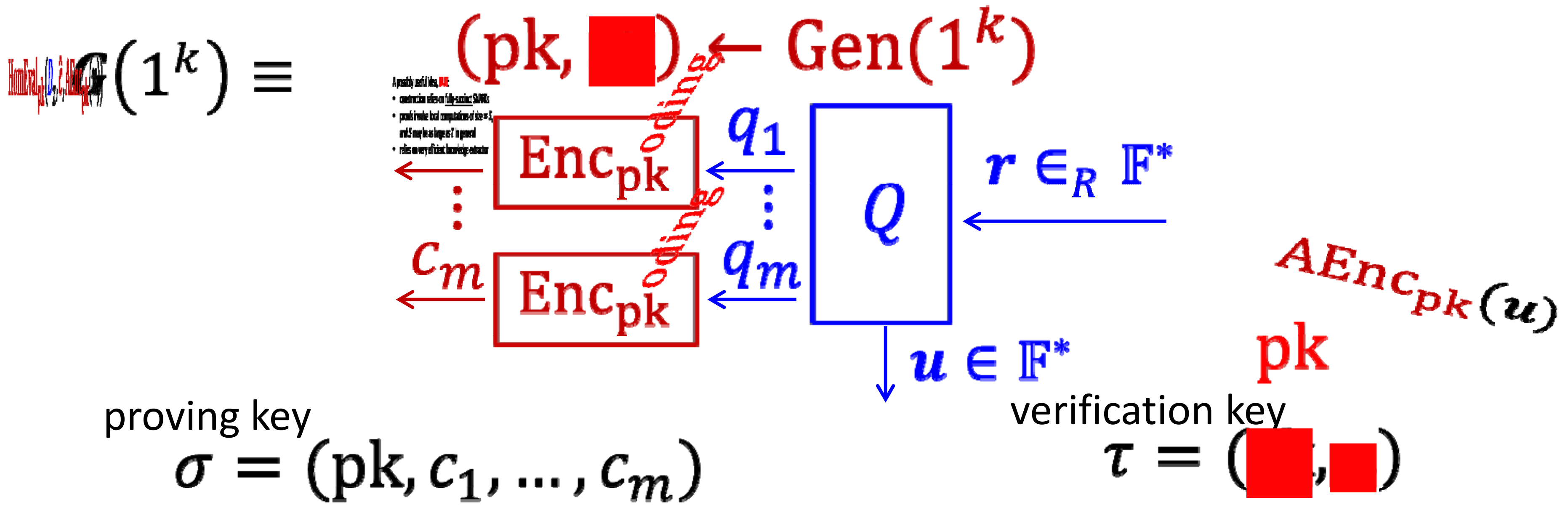
$\xrightarrow{\hat{c}_1} \vdots \xrightarrow{\hat{c}_k}$

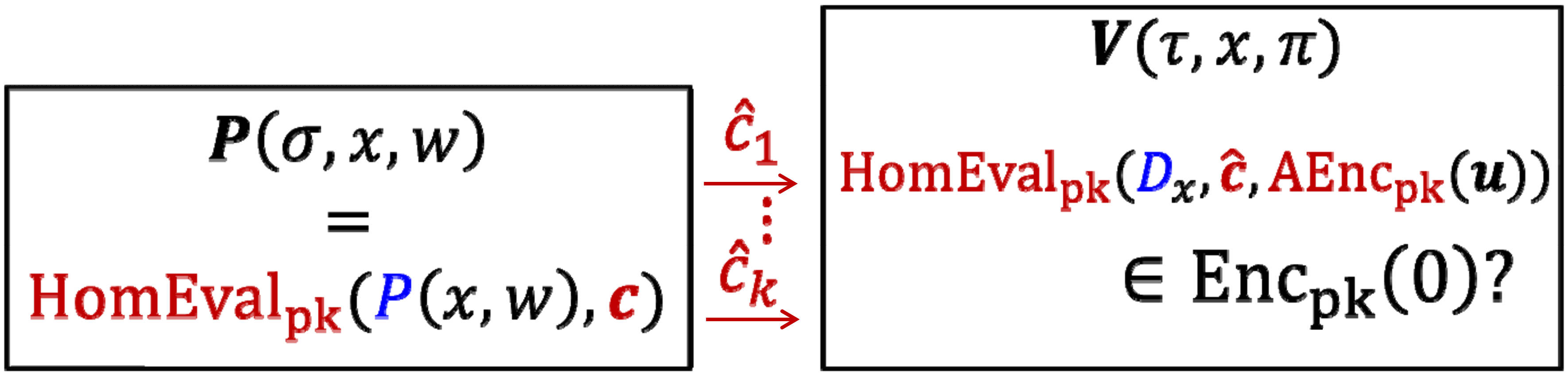
$V(\tau, x, \pi)$

$\hat{c} \quad D_x(\cdot,\cdot) \quad \text{AEnc}_{\text{pk}}(u)$

Step 2: Making Provers Linear

SKETCH

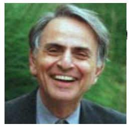




Step 2: Making Provers Linear

SKETCH

- test root with bilinear map so need $d_D = 2$
- similar linear TM assumption

$(1^k) \equiv$

$(\mathrm{pk}, \;) \leftarrow \mathrm{Gen}(1^k)$

$r \in_R \mathbb{F}^*$

Q

$q_1 \rightarrow \mathrm{Enc}_{\mathrm{pk}} \rightarrow$ $\vdots$

$q_m \rightarrow \mathrm{Enc}_{\mathrm{pk}} \rightarrow c_m$

$u \in \mathbb{F}^*$

$\mathrm{AEnc}_{\mathrm{pk}}(u)$

pk

proving key

$\sigma = (\mathrm{pk}, c_1, \ldots, c_m)$

verification key

$\tau = (\;, \;)$

$P(\sigma, x, w)$
$=$
$\mathrm{HomEval}_{\mathrm{pk}}(P(x, w), c)$

$\hat{c}_1 \rightarrow \; \vdots \; \hat{c}_k \rightarrow$

$V(\tau, x, \pi)$

$\mathrm{Test}_{\mathrm{pk}}(D_x, \hat{c}, \mathrm{AEnc}_{\mathrm{pk}}(u))$

"tests quadratic roots"

publicly-verifiable pp SNARK

Step 2 ✓

LIP with degree (poly(k),2)

Step 1

???

publicly-verifiable pp SNARK

Step 2

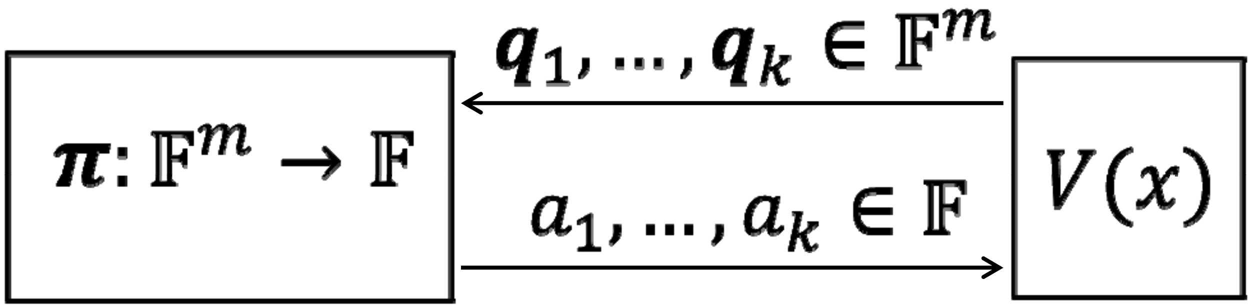
LIP with degree (poly(k),2)

Step 1

Linear PCP with degree (poly(k),2)

Linear PCPs (LPCPs)

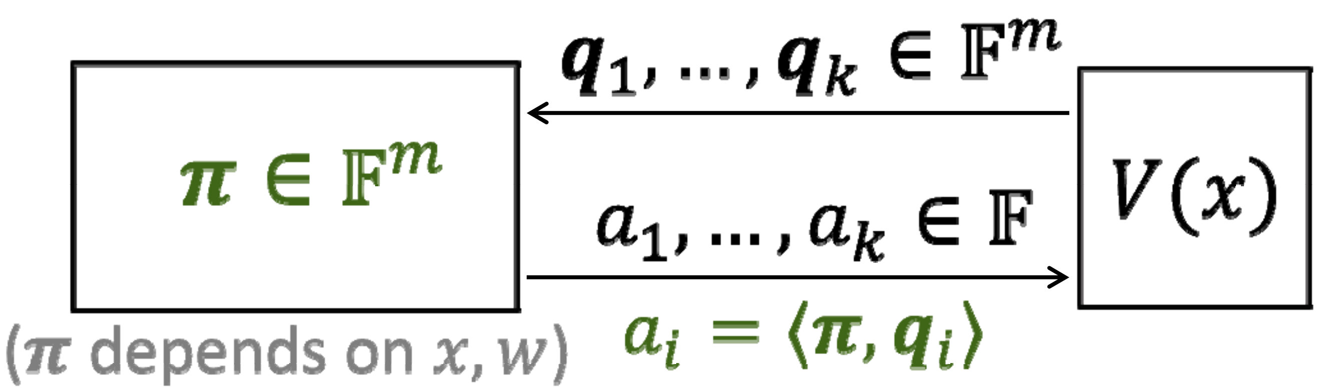
A PCP in which (honest and dishonest) proofs are **$\mathbb{F}$-linear**.



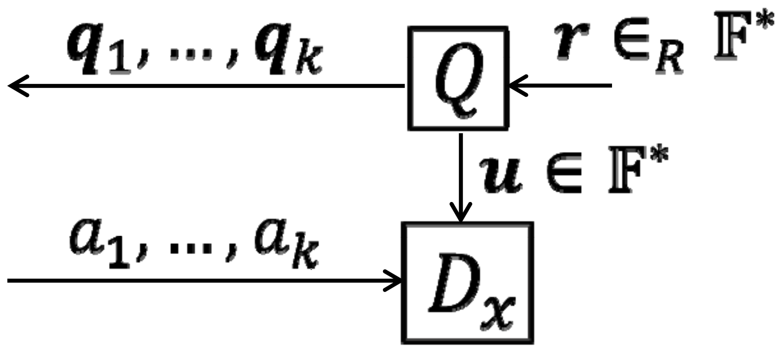
($\boldsymbol{\pi}$ depends on x, w)

Linear PCPs (LPCPs)

A PCP in which (honest and dishonest) proofs are **$\mathbb{F}$-linear**.



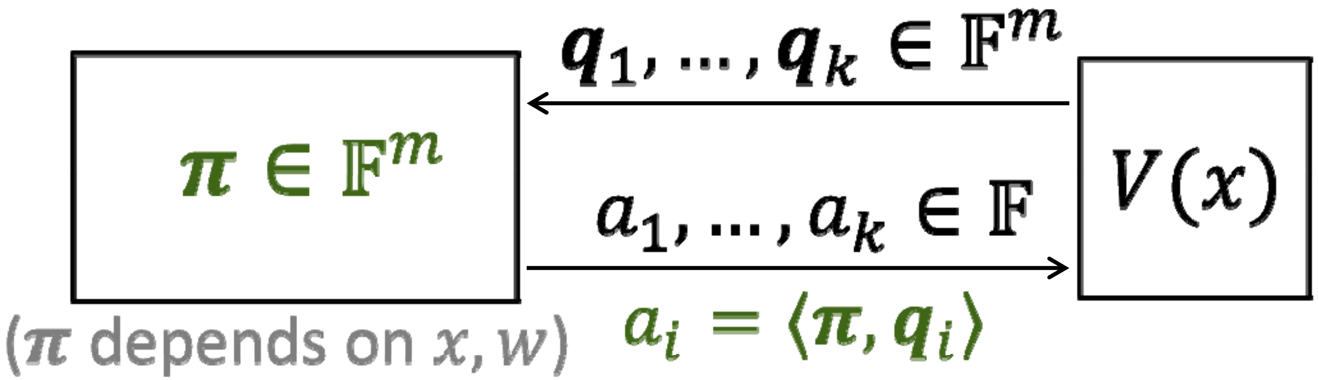
Similarly, *input oblivious*: $V = (Q, D)$ s.t.



and *degrees* (d_Q, d_D).

Linear PCPs (LPCPs)

A PCP in which (honest and dishonest) proofs are $\mathbb{F}$**-linear**.



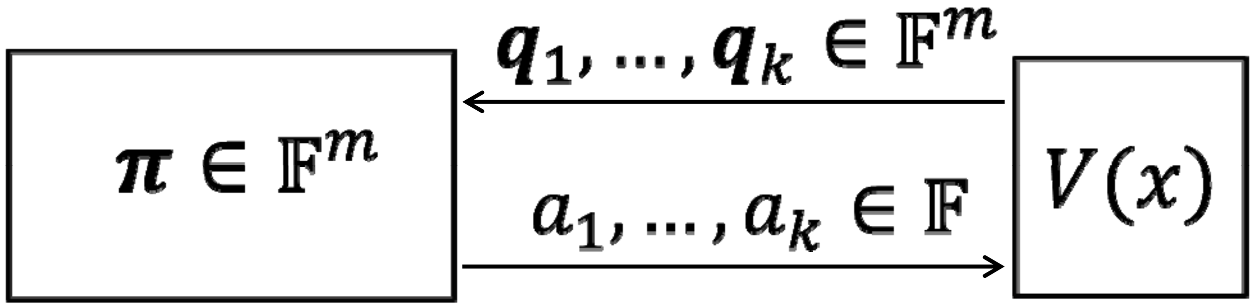
Two technical notes:

(1) *linear PCP* in [IKO,SMBW,SVP+,SBV+] does not restrict oracle to be linear in dishonest case

(2) not the same as *linear PCPP* in [BSHLM09,Mei12]; there it is a proximity tester for the kernels of linear circuits

From LPCPs To LIPs

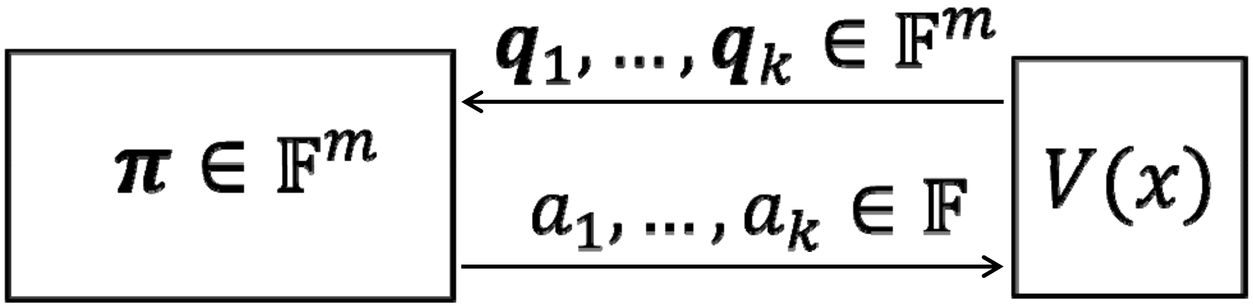
Given a k-query m-length LPCP, how to construct an LIP of similar efficiency?



Why isn't an LPCP already an LIP? **Consistency.**

From LPCPs To LIPs

Given a k-query m-length LPCP, how to construct an LIP of similar efficiency?

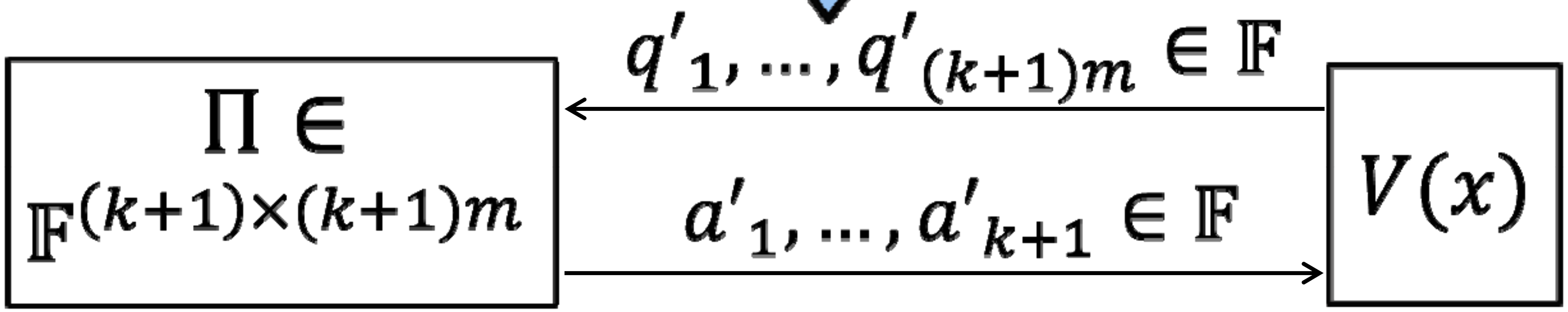


+

consistency check

$$\boldsymbol{q}_{k+1} = \textstyle\sum_{i=1}^{k} \alpha_i \boldsymbol{q}_i \text{ where } \alpha_1, \ldots, \alpha_k \in_R \mathbb{F}$$

preserves algebraic properties!



publicly-verifiable pp SNARK

Step 2

LIP with degree (poly(k),2)

Step 1

Linear PCP with degree (poly(k),2)

Step 0

???

publicly-verifiable pp SNARK

Step 2 ✓

LIP with degree (poly(k),2)

Step 1 ✓

Linear PCP with degree (poly(k),2)

Step 0

designing LPCPs for NP with O(1) queries is easy!

publicly-verifiable pp SNARK

Step 2 ✓

LIP with degree (poly(k),2)

Step 1 ✓

Linear PCP with degree (poly(k),2)

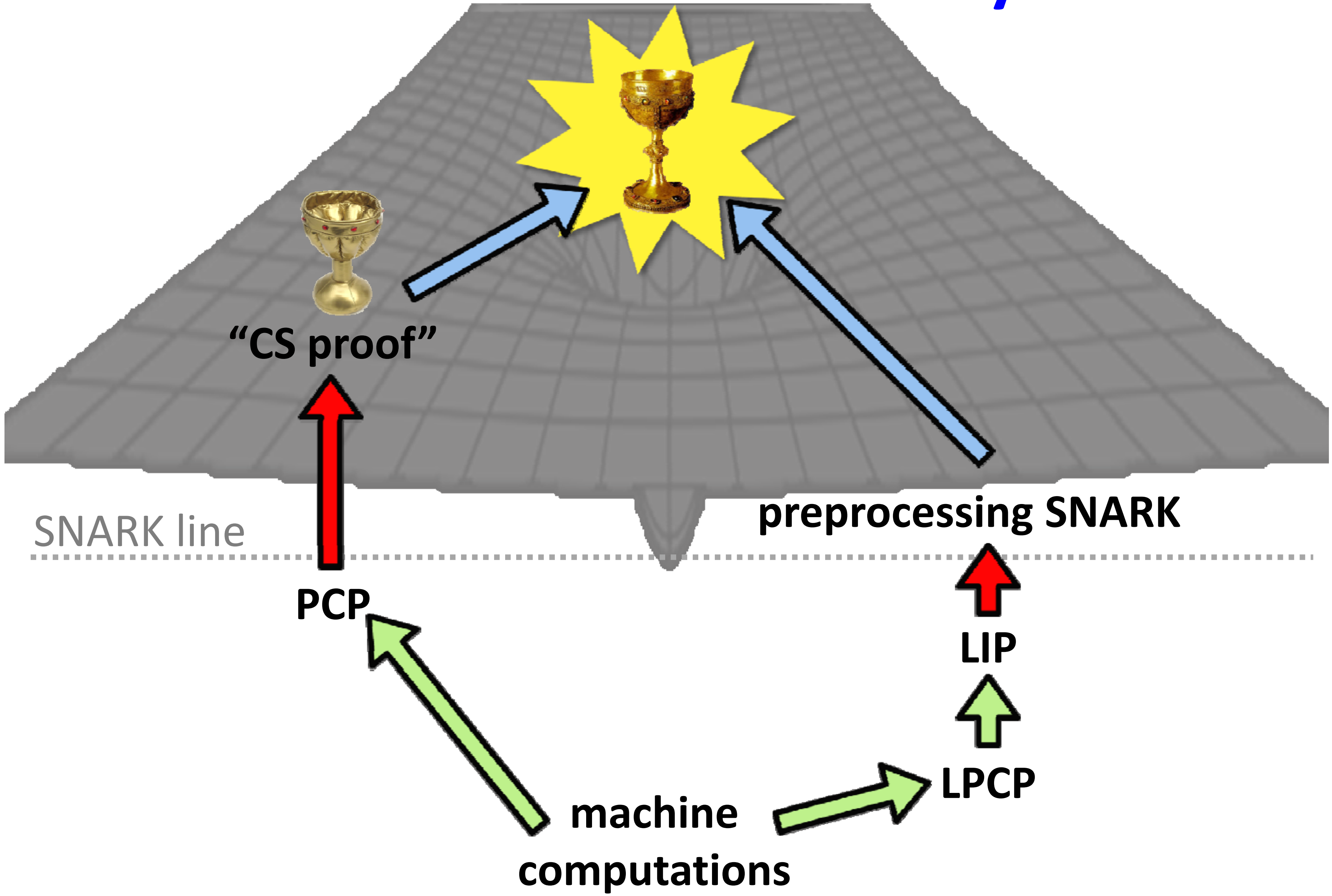
Step 0

[ALMSS]
$m = O(|C|^2)$
$k = 3$

[GGPR]
$m = O(|C|)$
$k = 3$ (or 4)

system of O(|C|) quadratic equations over $\mathbb{F}$

Two Known Paths To The Holy Grail



THANKS!

THANKS!