Multi-Theorem Preprocessing NIZKs from Lattices

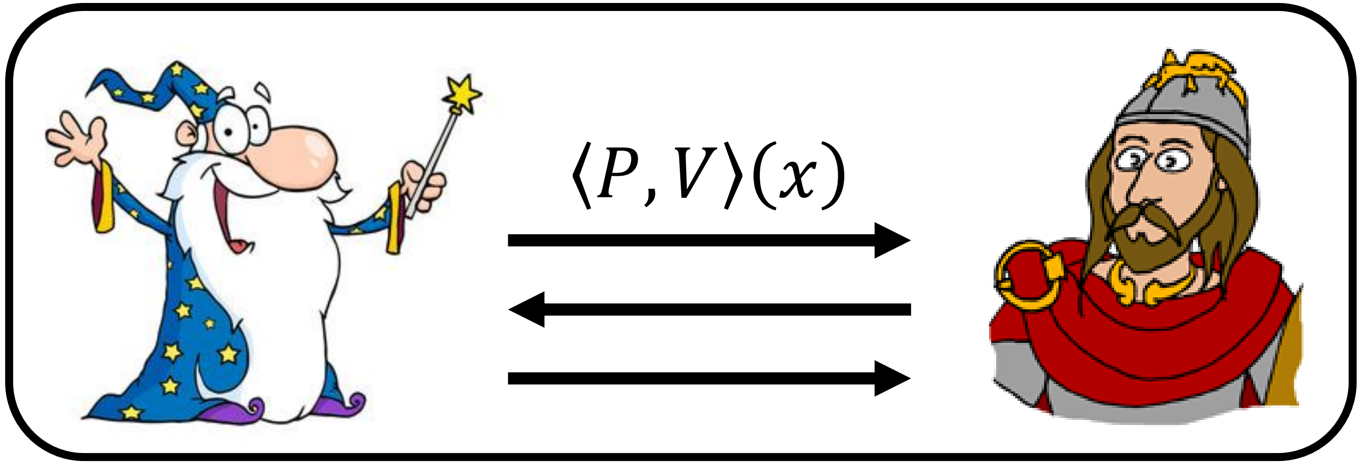
Sam Kim and David J. Wu

Stanford University

Zero-Knowledge Proofs for NP

[GMR85]

NP language $\mathcal{L}$



$\approx_c$

real distribution

ideal distribution

<u>Zero-Knowledge</u>: for all efficient verifiers V^*, there exists an efficient simulator $\mathcal{S}$ such that:

$$\forall x \in \mathcal{L} : \langle P, V^* \rangle(x) \approx_c \mathcal{S}(x)$$

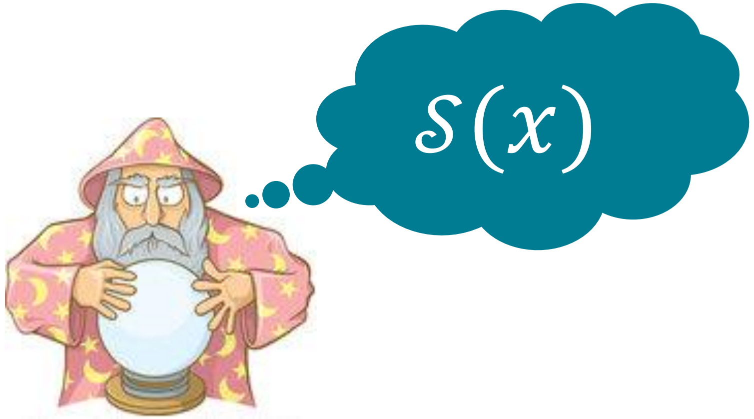
Non-Interactive Zero-Knowledge (NIZK) Proofs

[BFM88]

NP language $\mathcal{L}$

$\xrightarrow{\pi}$

$\approx_c$

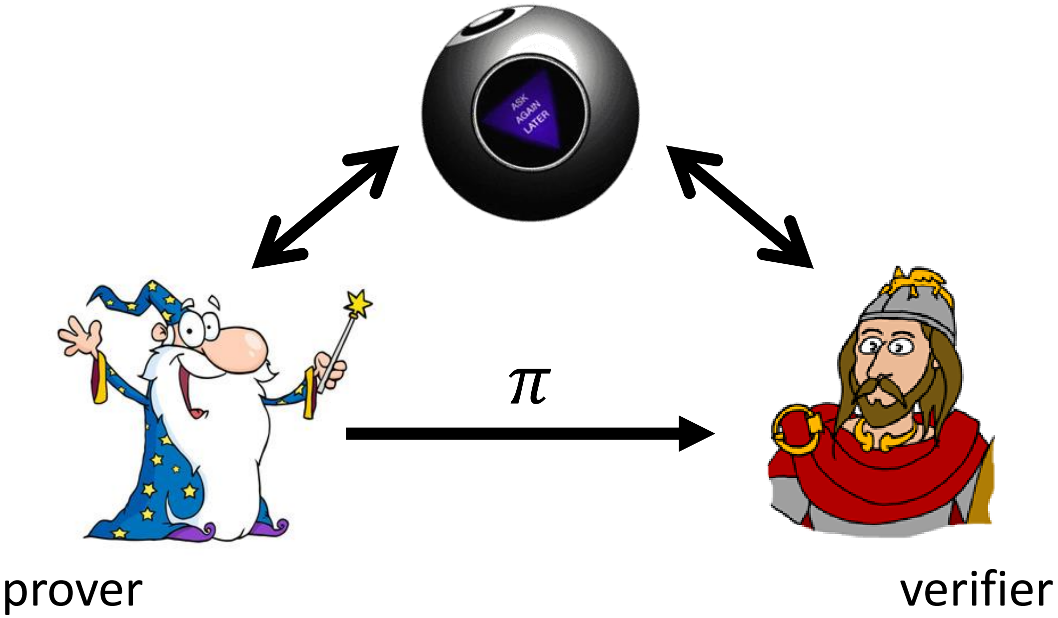


real distribution

ideal distribution

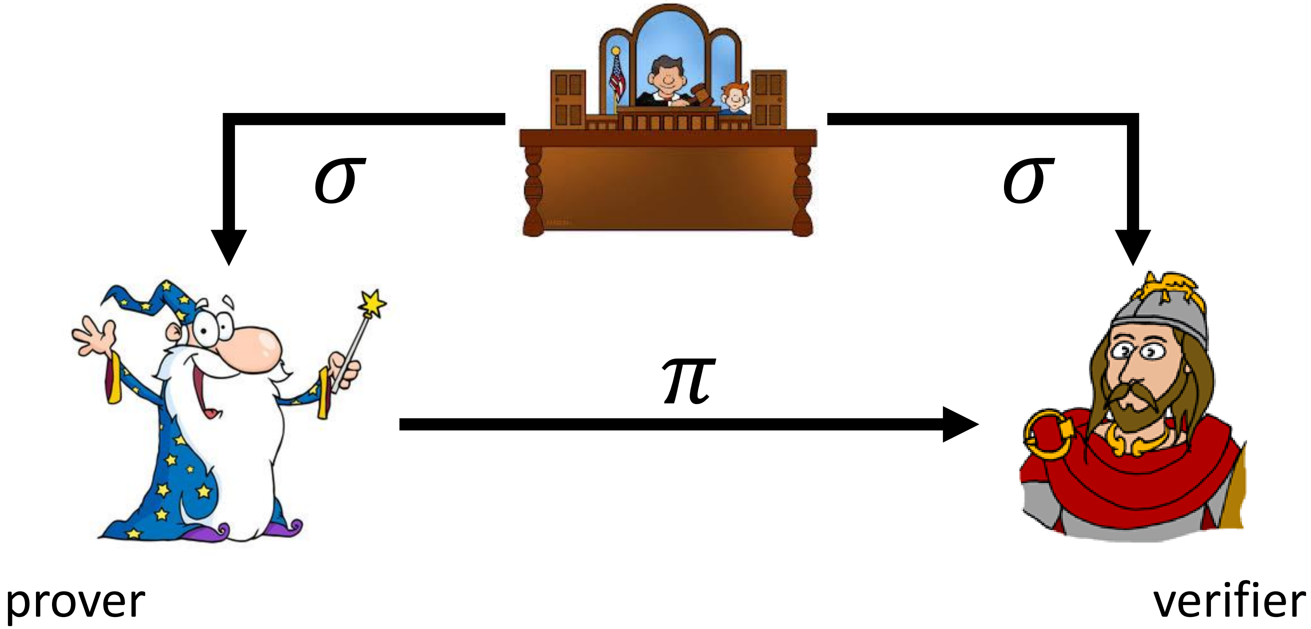
In the standard model, this is only achievable for languages $\mathcal{L} \in \mathrm{BPP}$

Which Assumptions give NIZKs for NP?



Random Oracle Model

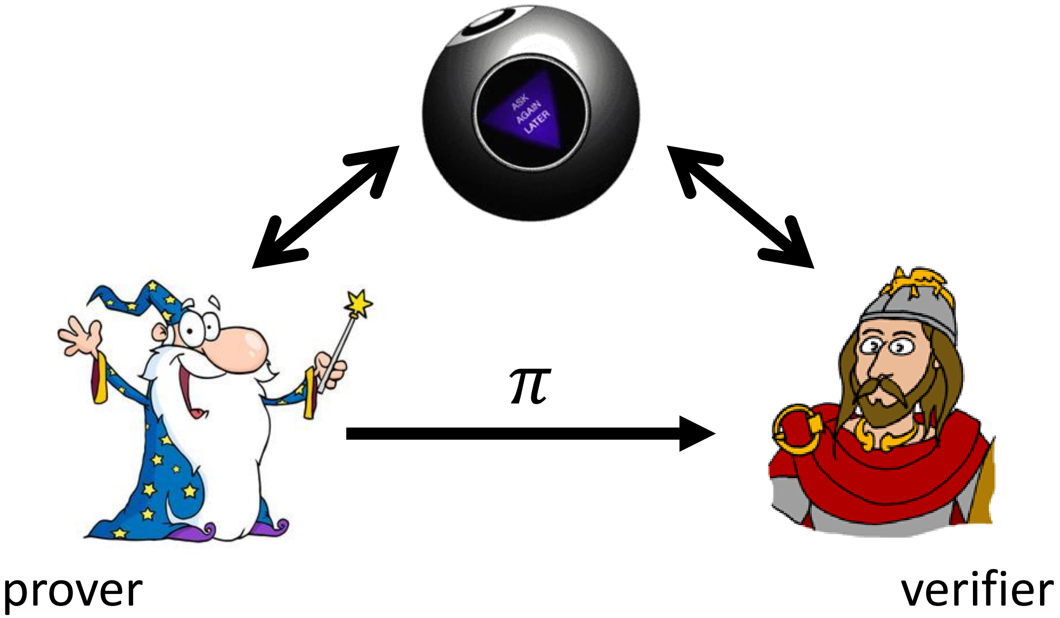
[FS86, PS96]



Common Reference String (CRS) Model

- Quadratic Residuosity [BFM88, DMP87, BDMP91]
- Trapdoor Permutations [FLS90, DDO+01, Gro10]
- Pairings [GOS06]
- Indistinguishability Obfuscation + OWFs [SW14]

Which Assumptions give NIZKs for NP?

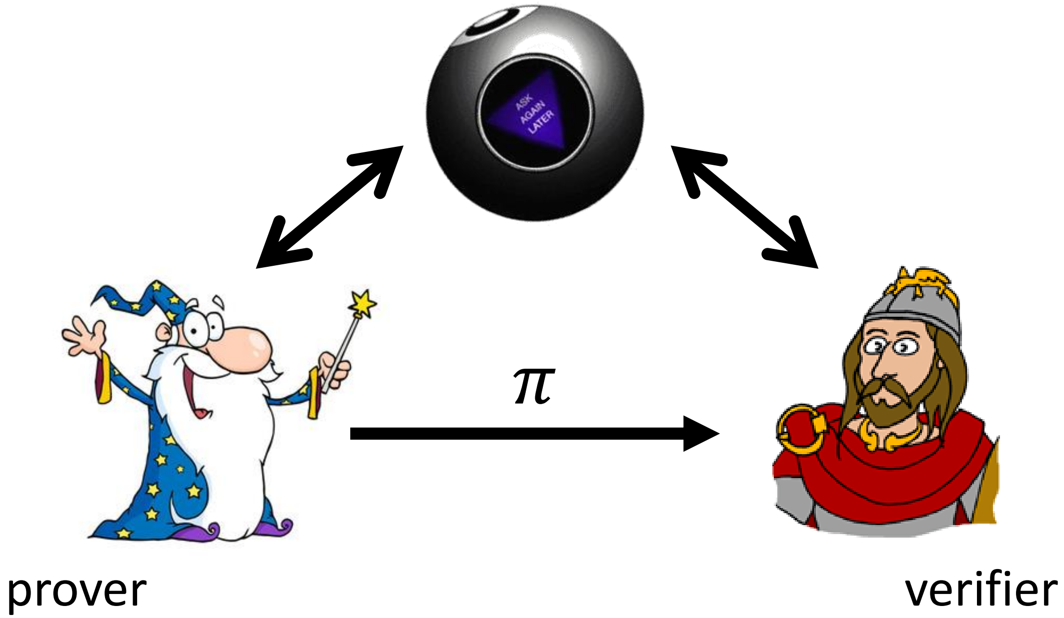


Random Oracle Model
[FS86, PS96]

Several major classes of assumptions missing:
- Discrete-log based assumptions (e.g., CDH, DDH)
- Lattice-based assumptions (e.g., SIS, LWE)

Common Reference String (CRS) Model
- Quadratic Residuosity [BFM88, DMP87, BDMP91]
- Trapdoor Permutations [FLS90, DDO+01, Gro10]
- Pairings [GOS06]
- Indistinguishability Obfuscation + OWFs [SW14]

Which Assumptions give NIZKs for NP?



Random Oracle Model
[FS86, PS96]

Several major classes of assumptions missing:
- Discrete-log based assumptions (e.g., CDH, DDH)
- Lattice-based assumptions (e.g., SIS, LWE)

Common Reference String (CRS) Model
- Quadratic Residuosity [BFM88, DMP87, BDMP91]
- Trapdoor Permutations [FLS90, DDO+01, Gro10]
- Pairings [GOS06]
- Indistinguishability Obfuscation + OWFs [SW14]

NIZKs in the Preprocessing Model

[DMP88]

(Trusted) setup algorithm generates both proving key k_P and a verification key k_V

k_P

k_V

prover

$\pi = \mathrm{Prove}(k_P, x, w)$

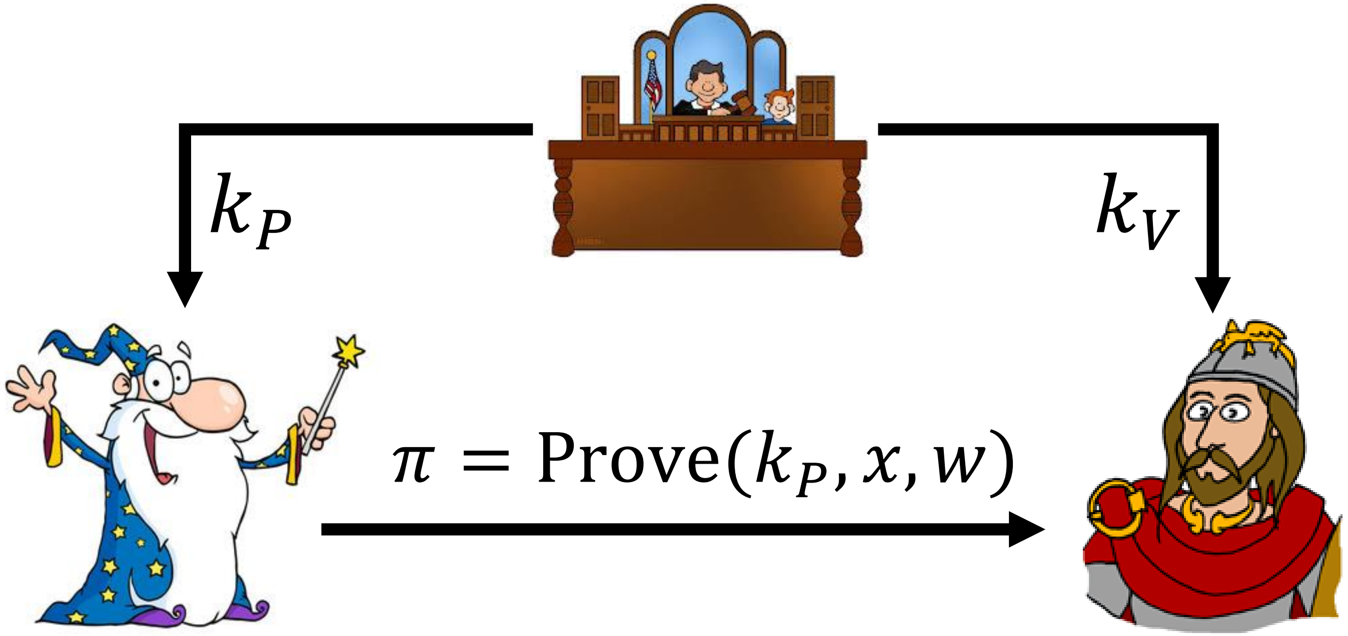
Prover algorithm takes proving key k_P, NP statement x, and NP witness w

verifier

$\mathrm{Verify}(k_V, x, \pi)$

NIZKs in the Preprocessing Model

[DMP88]



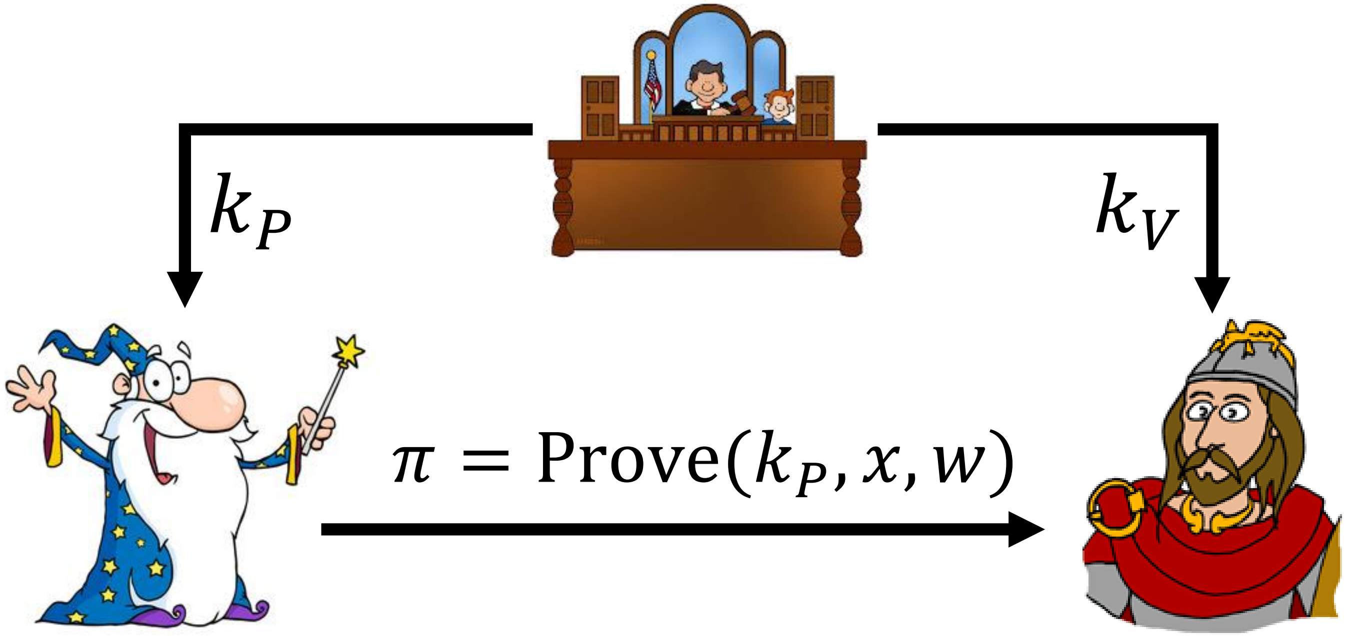
Simpler model than CRS model:

- Soundness holds assuming k_V is <u>hidden</u>
- Zero-knowledge holds assuming k_P is <u>hidden</u>

If only k_V is private (i.e., k_P is public), then the NIZK is <u>designated-verifier</u>

NIZKs in the Preprocessing Model

[DMP88]



Simpler model than CRS model:

- Soundness holds assuming k_V is <u>hidden</u>
- Zero-knowledge holds assuming k_P is <u>hidden</u>

<u>Preprocessing NIZKs</u>

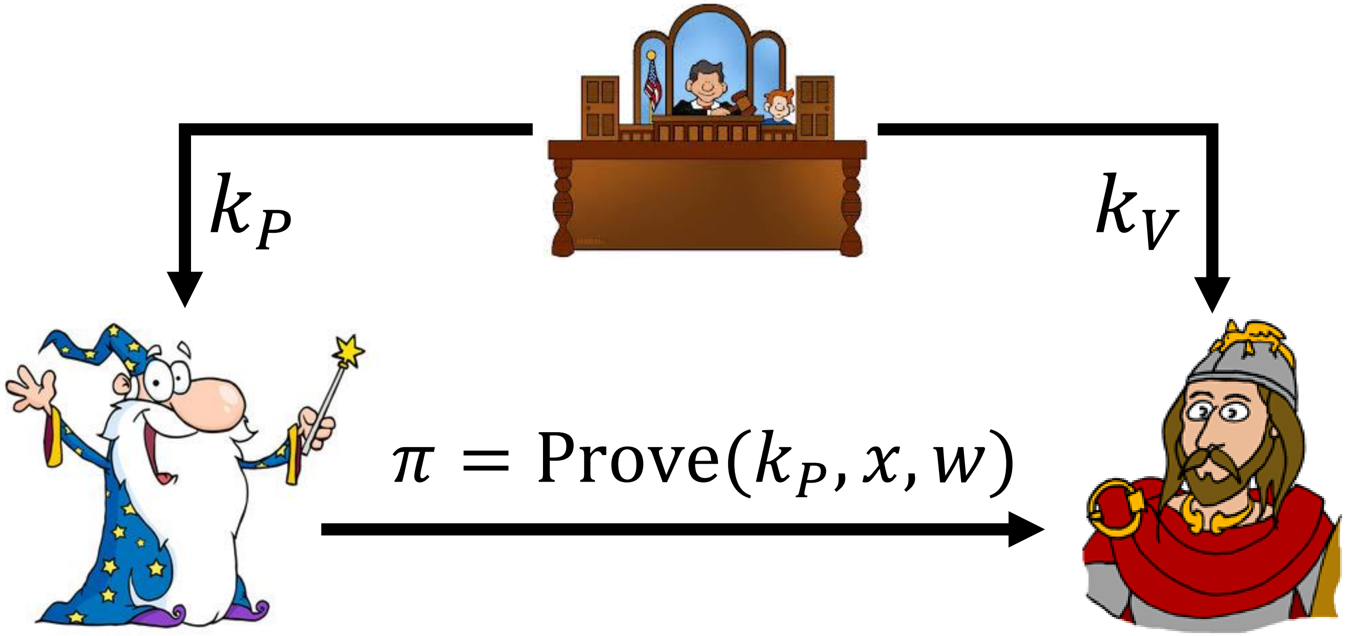
- One-Way Functions [DMP88, LS90, Dam92, IKOS09]
- Oblivious Transfer [KMO89]

<u>Designated-Verifier NIZKs</u>

- Additively-homomorphic encryption [CD04, DFN06, CG15]

NIZKs in the Preprocessing Model

[DMP88]



Preprocessing NIZKs

- One-Way Functions [DMP88, LS90, Dam92, IKOS09]
- Oblivious Transfer [KMO89]

Designated-Verifier NIZKs

- Additively-homomorphic encryption [CD04, DFN06, CG15]

Existing constructions only provide bounded-theorem soundness or bounded-theorem zero-knowledge

NIZKs in the Preprocessing Model

[DMP88]

Bounded-theorem soundness: Soundness holds in a setting where prover can see verifier's response on an *a priori* bounded number of queries – "verifier rejection problem"

Bounded-theorem zero-knowledge: Zero-knowledge holds in a setting where verifier can see proofs on an *a priori* bounded number of statements

Existing constructions only provide bounded-theorem soundness or bounded-theorem zero-knowledge

<u>Preprocessing NIZKs</u>

- One-Way Functions [DMP88, LS90, Dam92, IKOS09]
- Oblivious Transfer [KMO89]

<u>Designated-Verifier NIZKs</u>

- Additively-homomorphic encryption [CD04, DFN06, CG15]

NIZKs in the Preprocessing Model

[DMP88]

Only known constructions of multi-theorem NIZKs in the preprocessing model are those in the CRS model

Can we realize multi-theorem NIZKs in the preprocessing model from standard lattice assumptions?

Hope: Preprocessing NIZKs is a stepping stone towards NIZKs from standard lattice assumptions

Our Results

Can we realize multi-theorem NIZKs in the preprocessing model from standard lattice assumptions?

- First <u>multi-theorem</u> preprocessing NIZK from LWE
  (in fact, a "designated-prover" NIZK)
- Preprocessing step can be efficiently implemented using OT
- Several new MPC protocols from lattices:
  - Succinct version of GMW compiler from lattices

Our Results

Can we realize multi-theorem NIZKs in the preprocessing model from standard lattice assumptions?

Communication overhead is proportional to <u>depth</u> of the computation rather than the <u>size</u> of the computation

processing NIZK from LWE

rover" NIZK)

be efficiently implemented using OT

cols from lattices:

- Succinct version of GMW compiler from lattices

Our Results

Can we realize multi-theorem NIZKs in the preprocessing model from standard lattice assumptions?

- First multi-theorem preprocessing NIZK from LWE

  (in fact, a "designated-prover" NIZK)
- Preprocessing step can be efficiently im
- Several new MPC protocols from lattice
  - Succinct version of GMW compiler from la
  - Two-round, succinct MPC from lattices in a "reusable preprocessing" model

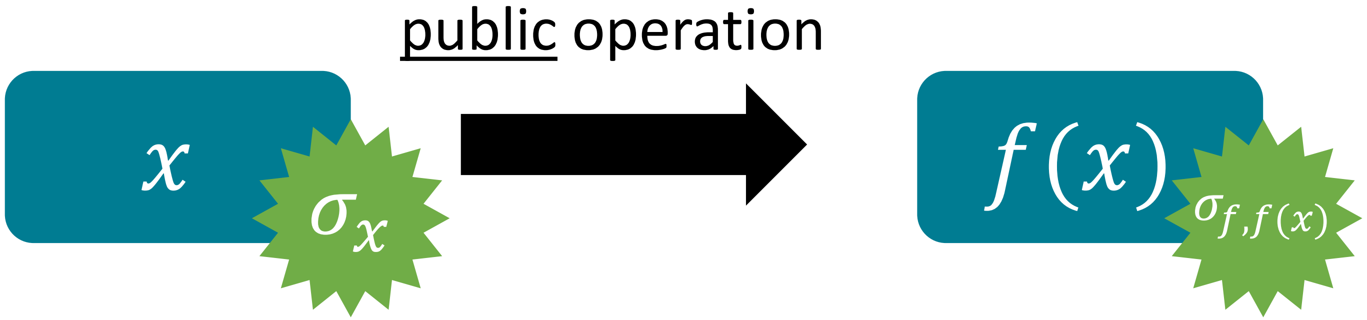
Preprocessing can be done once and then reused for *arbitrarily* many computations

Total communication proportional to depth of computation

Starting Point: Homomorphic Signatures

[BF11, GVW15, ABC+15]

σ_x is a signature on x
with respect to a
verification key vk



$\sigma_{f,f(x)}$ is a signature on $f(x)$
with respect to the function f
and the verification key vk

Homomorphic signatures enable computations on <u>signed</u> data

Starting Point: Homomorphic Signatures

[BF11, GVW15, ABC+15]

x σ_x —public operation→ $f(x)$ $\sigma_{f,f(x)}$

$\sigma_{f,f(x)}$ is a signature on $f(x)$ with respect to the function f and the verification key vk

(One-Time) Unforgeability:

$\xleftarrow{\text{vk}}$

$\xrightarrow{x}$

σ_x $\sigma_x \leftarrow \mathrm{Sign}(\mathrm{sk}, x)$ $\longleftarrow$

$\xrightarrow{y \quad \sigma_{f,y}}$

sk

Adversary wins if $\sigma_{f,y}$ is a valid signature on y with respect to function f, but $y \neq f(x)$

Unforgeable if no efficient adversary can win

Starting Point: Homomorphic Signatures

[BF11, GVW15, ABC+15]

x σ_x — public operation → $f(x)$ $\sigma_{f,f(x)}$

$\sigma_{f,f(x)}$ is a signature on $f(x)$ with respect to the function f and the verification key vk

Context-Hiding:

x σ_x

sk

$f, f(x)$

$f(x)$ $\sigma_{f,f(x)}$ $\approx_c$ $f(x)$ $\tilde{\sigma}_{f,f(x)}$

real distribution

ideal distribution

Looks like a zero-knowledge property!

$\sigma_{f,f(x)}$ hides the original input x (up to what is revealed by $f, f(x)$)

[Generalizes to multiple signatures]

Homomorphic Signatures to Preprocessing NIZKs

w σ_w

Suppose prover has a signature on w

Prover evaluates function $\mathcal{R}_x(w) = \mathcal{R}(x, w)$

$\mathcal{R}_x(w)$ $\sigma_{\mathcal{R}_x,1}$

prover (x, w)

verifier (x)

Goal: Convince verifier that there exists w such that $\mathcal{R}(x, w) = 1$

Homomorphic Signatures to Preprocessing NIZKs

w σ_w

Suppose prover has a signature on w

Prover evaluates function $\mathcal{R}_x(w) = \mathcal{R}(x, w)$

$\mathcal{R}_x(w)$ $\sigma_{\mathcal{R}_x,1}$

prover (x, w)

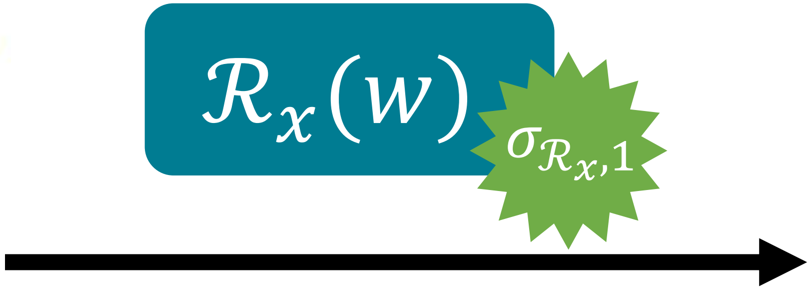
verifier (x)

Verifier checks that $\sigma_{\mathcal{R}_x,1}$ is a signature on 1 with respect to function $\mathcal{R}_x$

Homomorphic Signatures to Preprocessing NIZKs

Suppose prover has a signature on w

Prover evaluates function $\mathcal{R}_x(w) = \mathcal{R}(x, w)$



Soundness: Follows from unforgeability; if verifier accepts, then $\sigma_{\mathcal{R}_x,1}$ is a signature on 1 with respect to function $\mathcal{R}_x$, but $\mathcal{R}_x(w) = 0$

Homomorphic Signatures to Preprocessing NIZKs

w σ_w

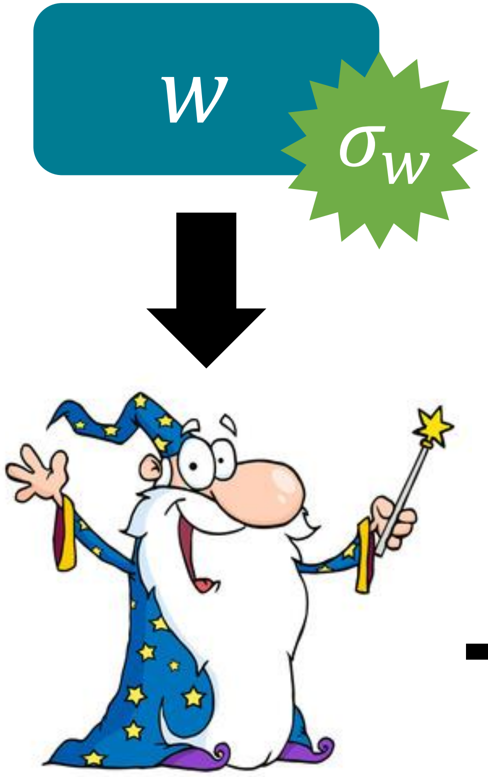
Suppose prover has a signature on w

Prover evaluates function $\mathcal{R}_x(w) = \mathcal{R}(x, w)$

$\mathcal{R}_x(w)$ $\sigma_{\mathcal{R}_x,1}$

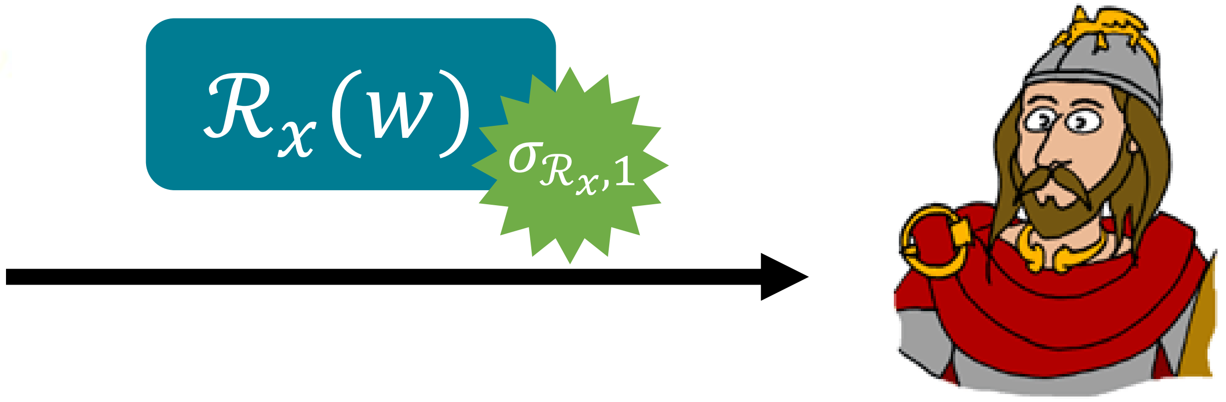
Zero-Knowledge: Follows from context-hiding; signature $\sigma_{\mathcal{R}_x,1}$ can be simulated given sk, $\mathcal{R}_x$ and $\mathcal{R}_x(w) = 1$

Homomorphic Signatures to Preprocessing NIZKs



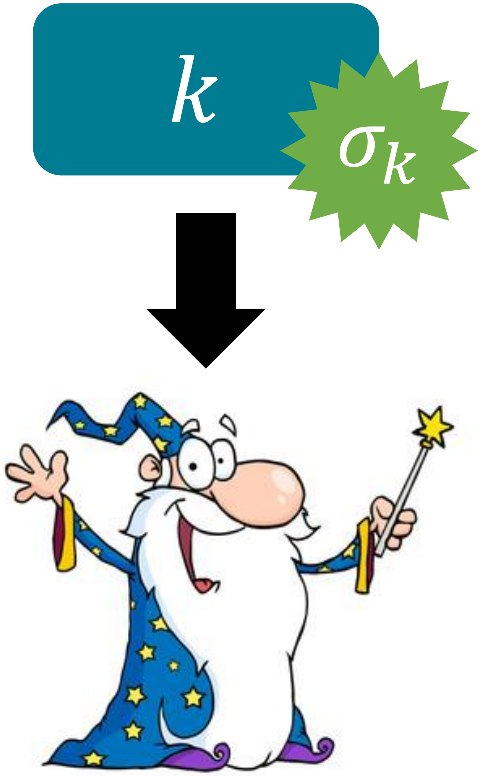
Suppose prover has a signature on w

Prover evaluates function $\mathcal{R}_x(w) = \mathcal{R}(x, w)$



Problem: Prover needs signature on w, which depends on the statement being proven (cannot be generated in preprocessing phase)

Homomorphic Signatures to Preprocessing NIZKs



Prover is given signature on an encryption key (unknown to the verifier)

Solution: Add one layer of indirection!

Homomorphic Signatures to Preprocessing NIZKs

k σ_k

Prover is given signature on an encryption key

$C_{x,\text{ct}}(k) = \mathcal{R}(x, \text{Decrypt}(k, \text{ct}))$
[Checks that ct encrypts a valid witness]

w k

$C_{x,\text{ct}}(k)$ $\sigma_{C_{x,\text{ct}},1}$

$\text{ct} \leftarrow \text{Encrypt}(k, w)$
[ct is an encryption of the witness w]

Solution: Add one layer of indirection!

Homomorphic Signatures to Preprocessing NIZKs

k σ_k

Prover is given signature on an encryption key

$C_{x,\mathrm{ct}}(k) = \mathcal{R}\big(x, \mathrm{Decrypt}(k, \mathrm{ct})\big)$
[Checks that ct encrypts a valid witness]

w k

$C_{x,\mathrm{ct}}(k)$ $\sigma_{C_{x,\mathrm{ct}},1}$

$\mathrm{ct} \leftarrow \mathrm{Encrypt}(k, w)$
[ct is an encryption of the witness w]

Verifier checks that $\sigma_{C_{x,\mathrm{ct}},1}$ is a signature on 1 with respect to function $C_{x,\mathrm{ct}}$

Homomorphic Signatures to Preprocessing NIZKs

k σ_k

Prover is given signature on an encryption key

$C_{x,\mathrm{ct}}(k) = \mathcal{R}\big(x, \mathrm{Decrypt}(k, \mathrm{ct})\big)$
[Checks that ct encrypts a valid witness]

w k

$C_{x,\mathrm{ct}}(k)$ $\sigma_{C_{x,\mathrm{ct}},1}$

$\mathrm{ct} \leftarrow \mathrm{Encrypt}(k, w)$
[ct is an encryption of the witness w]

Soundness: Follows from unforgeability; if verifier accepts, then $\sigma_{C_{x,\mathrm{ct}},1}$ is a signature on 1 with respect to function $C_{x,\mathrm{ct}}$, but $C_{x,\mathrm{ct}}(k) = 0$ for all k

Homomorphic Signatures to Preprocessing NIZKs

k σ_k

Prover is given signature on an encryption key

$C_{x,\mathrm{ct}}(k) = \mathcal{R}\big(x, \mathrm{Decrypt}(k, \mathrm{ct})\big)$
[Checks that ct encrypts a valid witness]

w k

$C_{x,\mathrm{ct}}(k)$ $\sigma_{C_{x,\mathrm{ct}},1}$

$\mathrm{ct} \leftarrow \mathrm{Encrypt}(k, w)$
[ct is an encryption of the witness w]

Zero-Knowledge: Follows from context-hiding and semantic security; signature $\sigma_{C_{x,\mathrm{ct}},1}$ can be simulated given sk, $C_{x,\mathrm{ct}}$ and $C_{x,\mathrm{ct}}(k) = 1$ and so, ct hides w

Homomorphic Signatures to Preprocessing NIZKs

k σ_k

(also publishes vk)

k_P

$\pi = \mathrm{Prove}(k_P, x, w)$

$\mathrm{Verify}(\mathrm{vk}, x, \pi)$

<u>Designated-prover</u> NIZK from context-hiding homomorphic signatures

Homomorphic Signatures to Preprocessing NIZKs

(also publishes vk)

k

σ_k

k_P

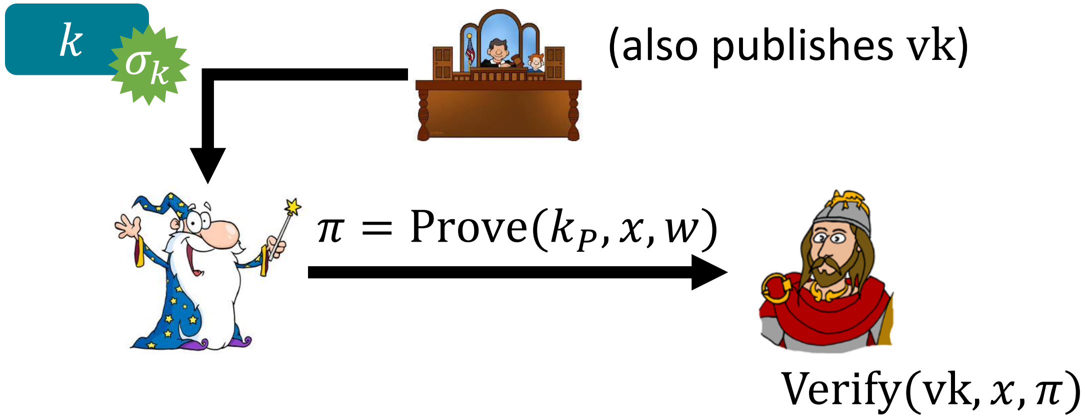
Can instantiate context-hiding homomorphic signatures with <u>lattice-based</u> scheme from [GVW15]

[Need some additional properties, but [GVW15] satisfies all properties with some modification]

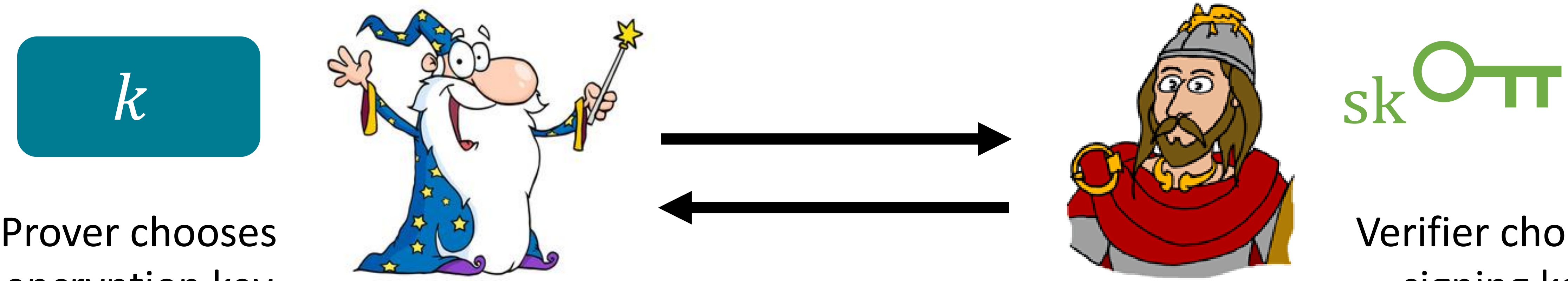
$\mathsf{Verify}(\mathsf{vk}, x, \pi)$

<u>Designated-prover</u> NIZK from context-hiding homomorphic signatures

Implementing the Preprocessing Phase



Can use generic MPC protocols, but can do this more efficiently using a specialized protocol



Prover chooses encryption key

Verifier chooses signing key

Goal: prover obtains signature on k without revealing k to verifier

Implementing the Preprocessing Phase

Desired notion is a
blind homomorphic signature

k

Prover chooses encryption key

sk

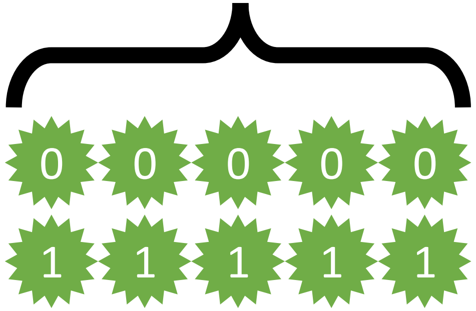
Verifier chooses signing key

Goal: prover obtains signature on k without revealing k to verifier

Blind Homomorphic Signatures

- Assume that homomorphic signatures is bitwise (can sign each bit of a message *independently*)
- Prover can then OT for the signatures on each bit of k
- Some additional work needed for *malicious* security
  [See paper for details]

signatures on bits of k



k

OT for signatures on bits of k

sk

Prover chooses encryption key

Verifier chooses signing key

Goal: prover obtains signature on k without revealing k to verifier

Summary

Can we realize multi-theorem NIZKs in the preprocessing model from standard lattice assumptions?

- New multi-theorem designated-prover (public-verifier) NIZKs from homomorphic signatures (based on LWE)
- New notion of blind homomorphic signatures (formalized in the UC model) for efficient implementation of preprocessing (from OT)
- New UC-secure NIZK in the preprocessing model from lattices
  - Succinct MPC protocol and succinct GMW compiler

[See paper for details]

Open Problems

NIZKs from lattices in the CRS model

- Publishing prover state in our preprocessing NIZK compromises zero-knowledge (reveals secret key prover uses to encrypt witnesses)

Multi-theorem preprocessing NIZKs from discrete log assumptions (e.g., CDH, DDH)

Thank you!

`https://eprint.iacr.org/2018/272`