# Multi-Theorem Preprocessing NIZKs from Lattices

Sam Kim and David J. Wu
Stanford University

# Proof Systems and Argument Systems

[GMR85]

$x \in \{0,1\}^*$

NP language $\mathcal{L} \subseteq \{0,1\}^*$

accept if $x \in \mathcal{L}$

prover

verifier

**Completeness:** $\forall x \in \mathcal{L} : \Pr[\langle P, V \rangle(x) = \text{accept}] = 1$

*"Honest prover convinces honest verifier of true statements"*

**Soundness:** $\forall x \notin \mathcal{L},\ \forall P^* : \Pr[\langle P^*, V \rangle(x) = \text{accept}] \leq \varepsilon$

*"No prover can convince honest verifier of false statement"*

# Proof Systems and Argument Systems

[GMR85]

$x \in \{0,1\}^*$

NP language $\mathcal{L} \subseteq \{0,1\}^*$

accept if $x \in \mathcal{L}$

prover

verifier

In an argument system, we relax soundness to only consider computationally-bounded (i.e., polynomial-time) provers $P^*$

**Completeness:**

*"Honest p… …nvinces honest verifier of true statements"*

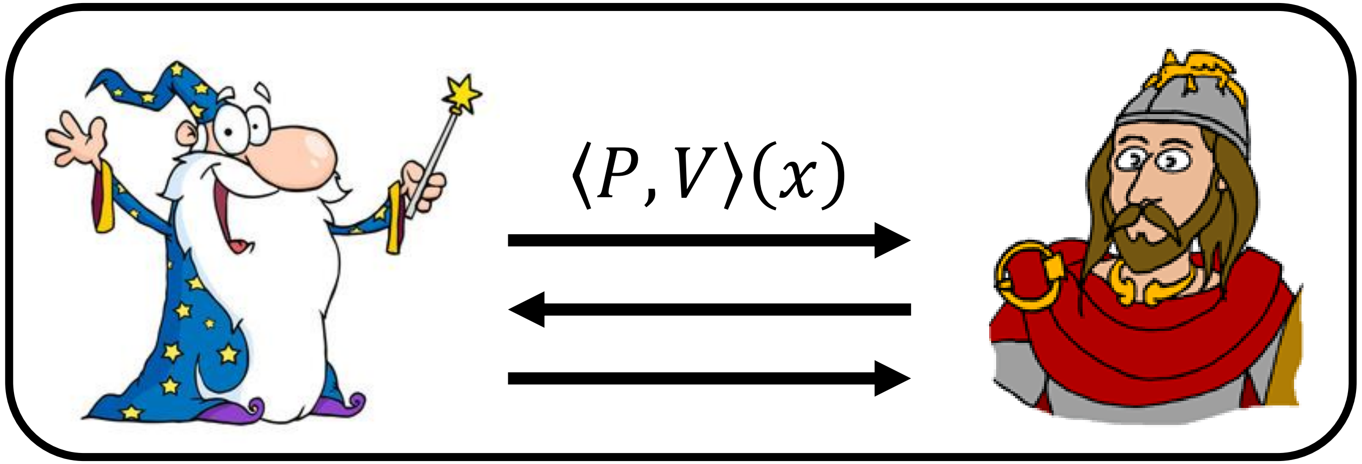
**Soundness:** $\forall x \notin \mathcal{L},\ \forall P^* : \Pr[\langle P^*, V\rangle(x) = \text{accept}] \leq \varepsilon$

*"No prover can convince honest verifier of false statement"*

# Zero-Knowledge Proofs for NP

[GMR85]

NP language $\mathcal{L} \subseteq \{0,1\}^*$




$\approx_c$


real distribution

ideal distribution

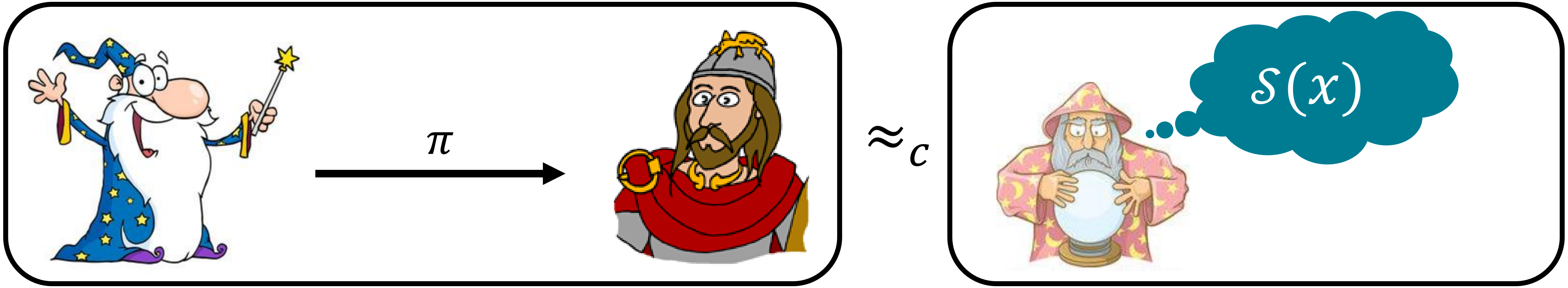
<u>**Zero-Knowledge**</u>**:** for all efficient verifiers $V^*$, there exists an efficient simulator $\mathcal{S}$ such that:

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

# Non-Interactive Zero-Knowledge (NIZK) Proofs

[BFM88]

NP language $\mathcal{L} \subseteq \{0,1\}^*$

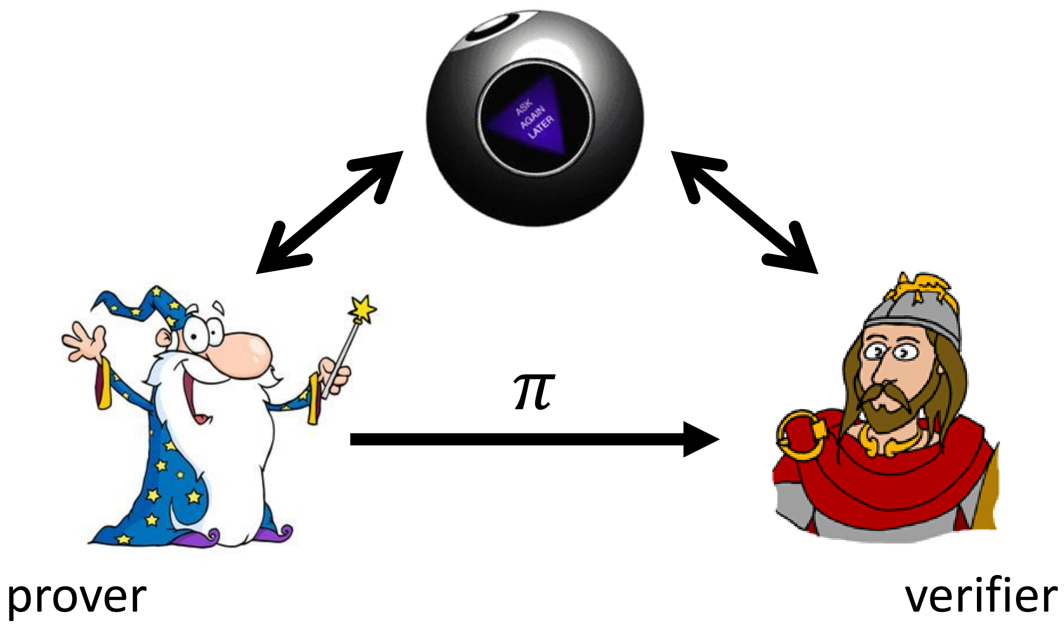



real distribution

ideal distribution

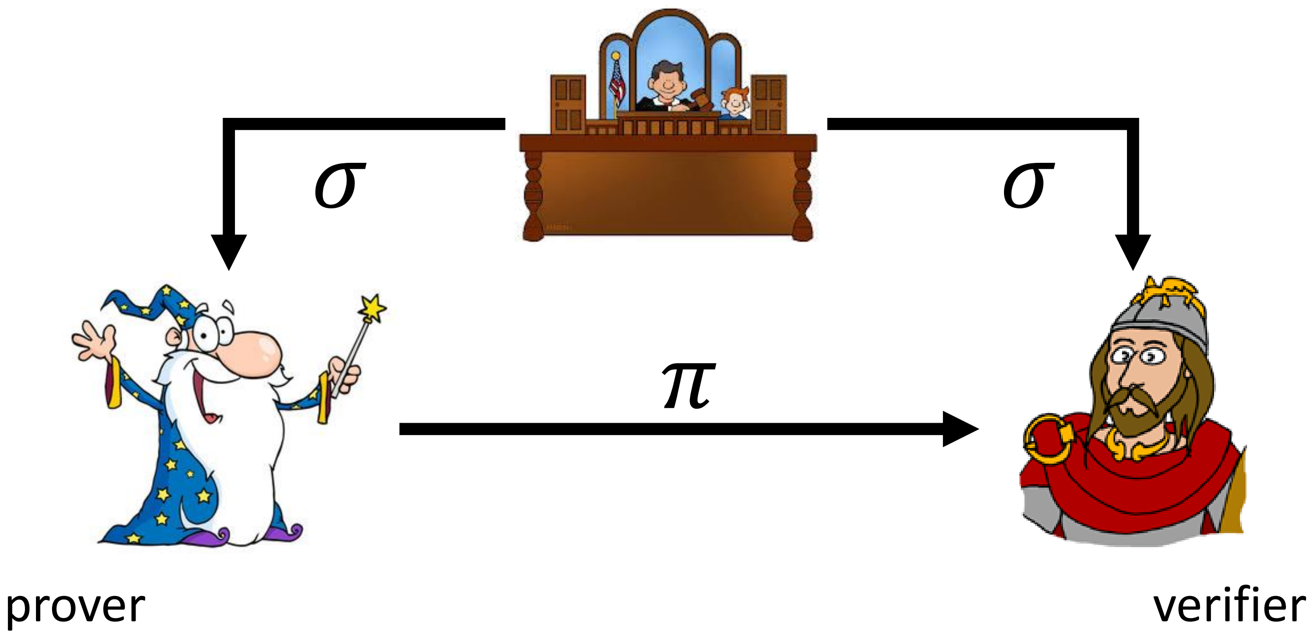
In the standard model, this is only achievable for languages $\mathcal{L} \in \mathrm{BPP}$

# Which Assumptions give NIZKs for NP?




Random Oracle Model

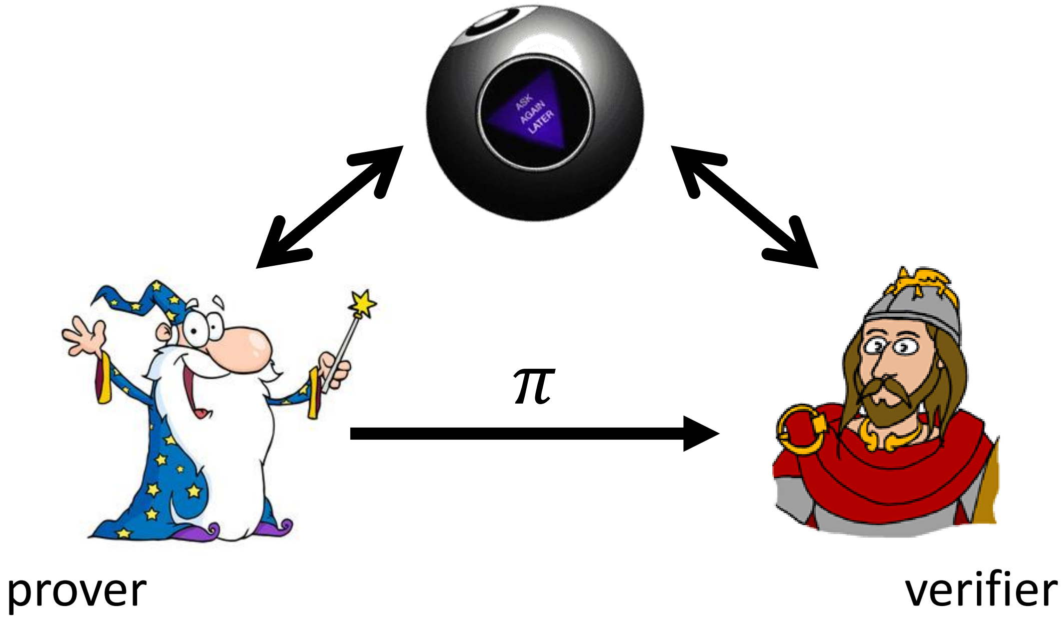
[FS86, PS96]




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]

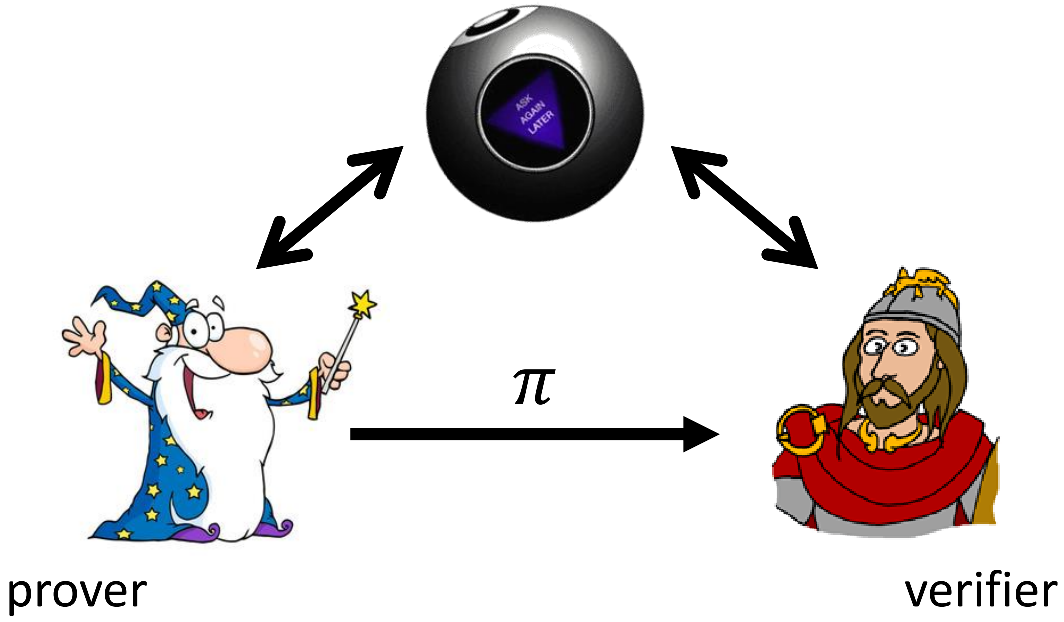
Several major classes of assumptions missing:

- 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

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

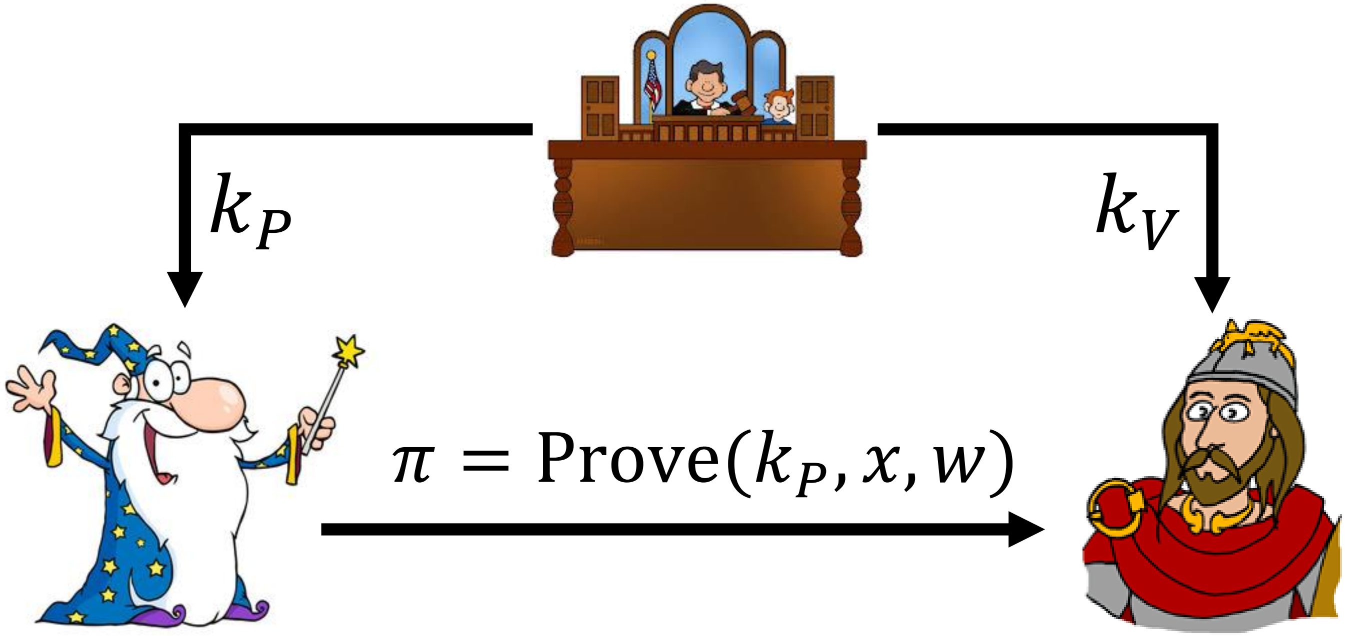
Prover algorithm takes proving key $k_P$, NP statement $x$, and NP witness $w$

verifier

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

# NIZKs in the Preprocessing Model

[DMP88]



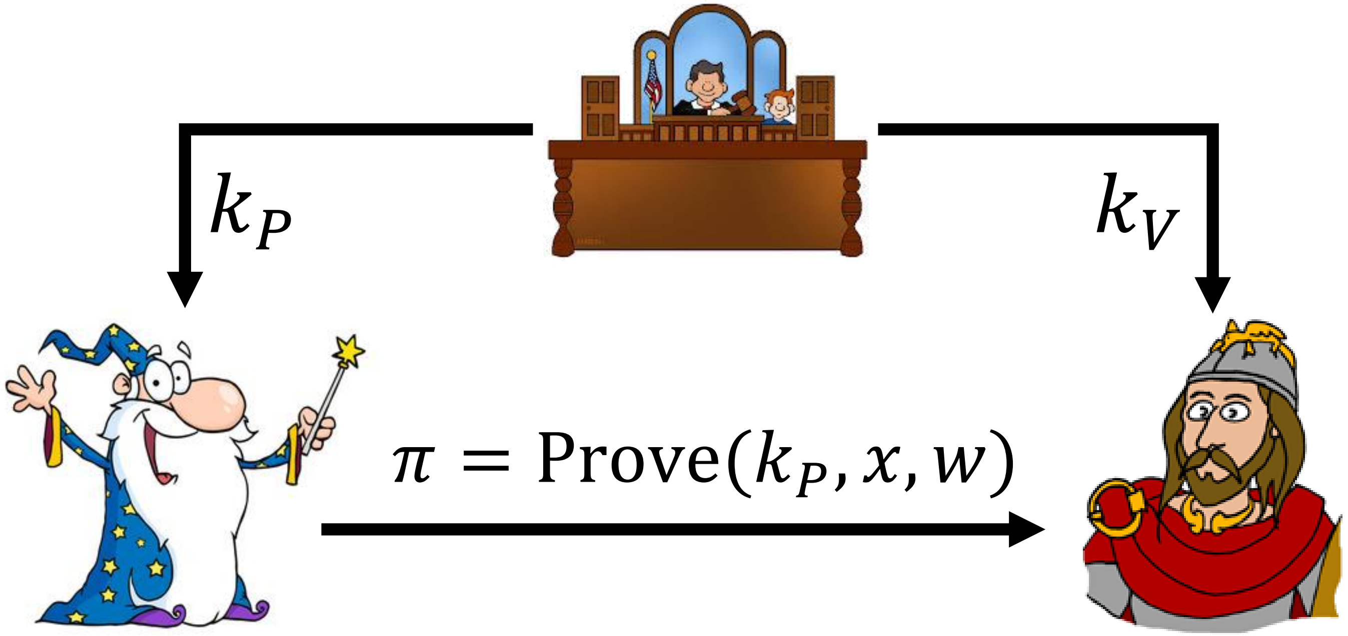

Simpler model than CRS model:

- 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>

## Preprocessing NIZKs

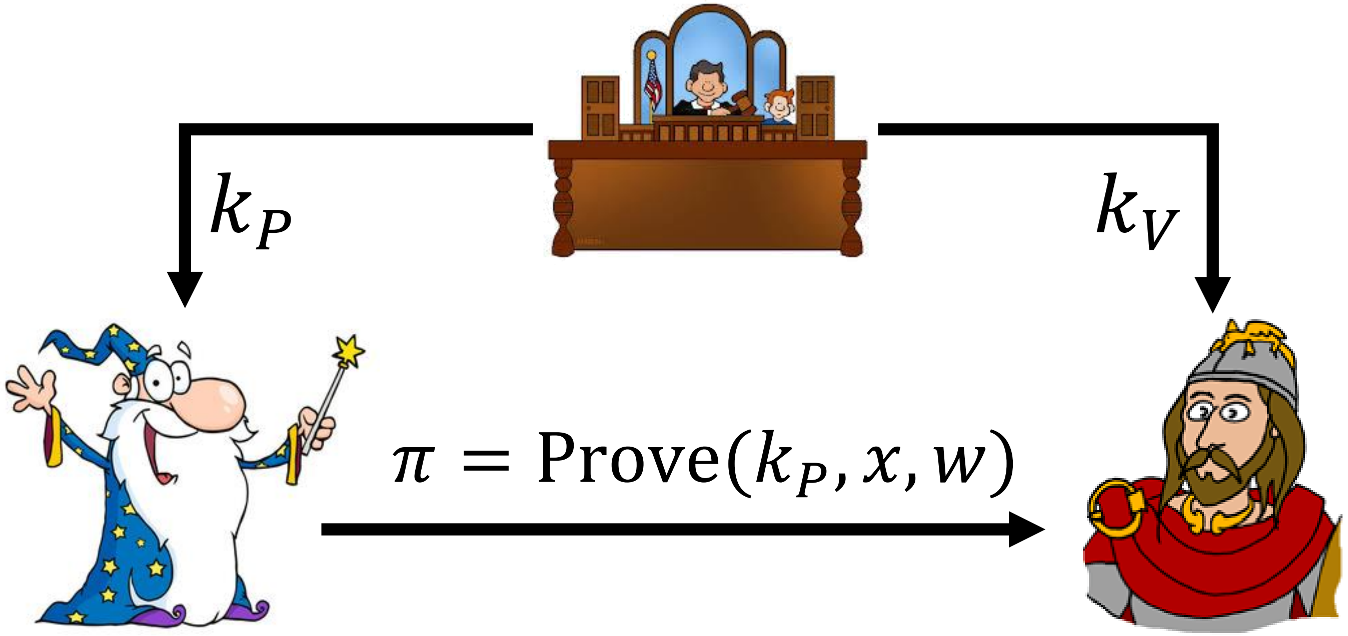
- One-Way Functions [DMP88, LS90, Dam92, IKOS09]
- Oblivious Transfer [KMO89]

## Designated-Verifier NIZKs

- 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

## Preprocessing NIZKs

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

## Designated-Verifier NIZKs

- 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

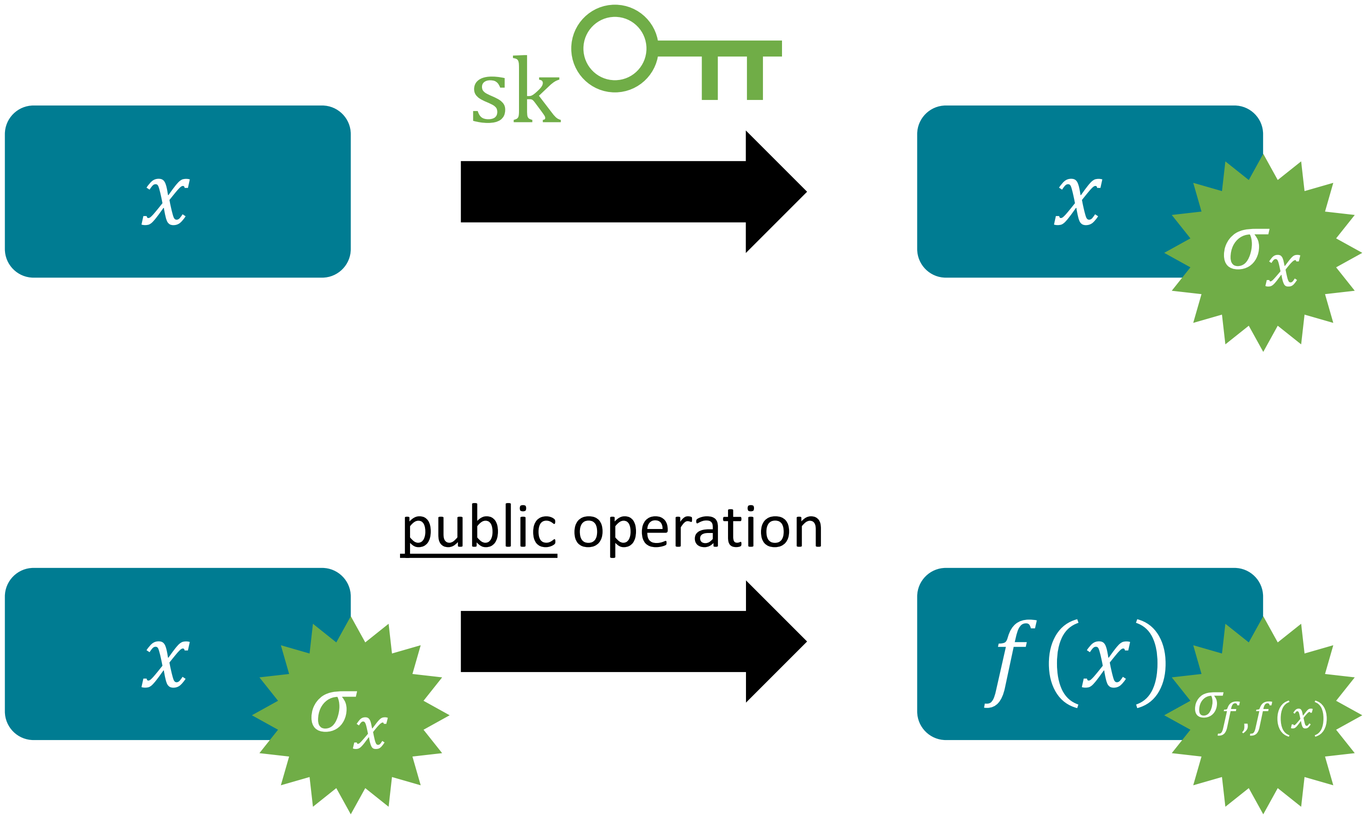
# Our Results

*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
  - Two-round, succinct MPC from lattices in a "reusable preprocessing" model

# Starting Point: Homomorphic Signatures

[BF11, GVW15, ABC+15]




$\sigma_x$ 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]

$x$ $\sigma_x$ → <u>public</u> 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:

vk ←

$x$ →

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

$y$ $\sigma_{f,y}$ →

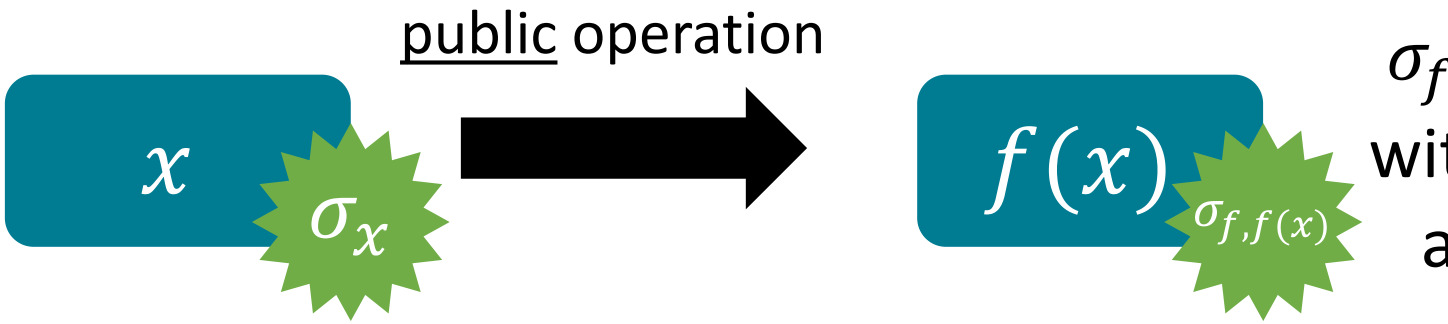
$\mathrm{sk}$

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

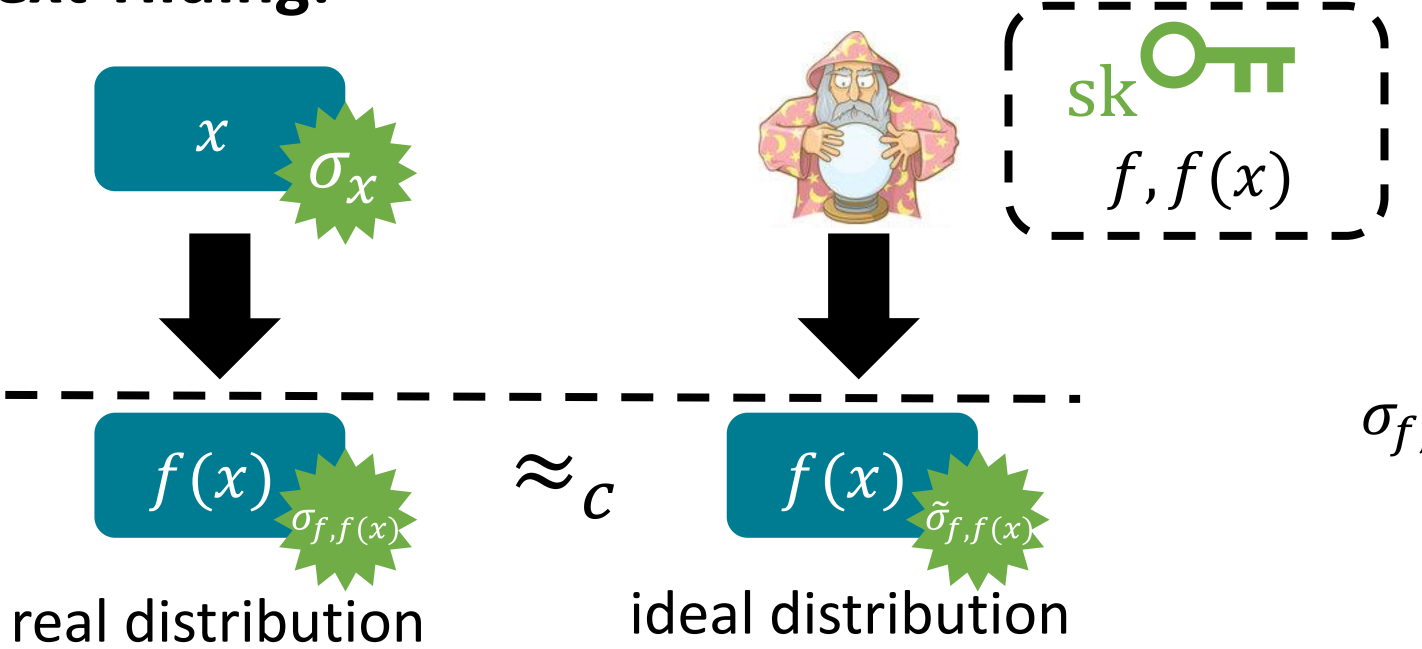
# Starting Point: Homomorphic Signatures

[BF11, GVW15, ABC+15]




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

**Context-Hiding:**




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

$w$ $\sigma_w$

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

$w$ $\sigma_w$

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 $(x, w)$

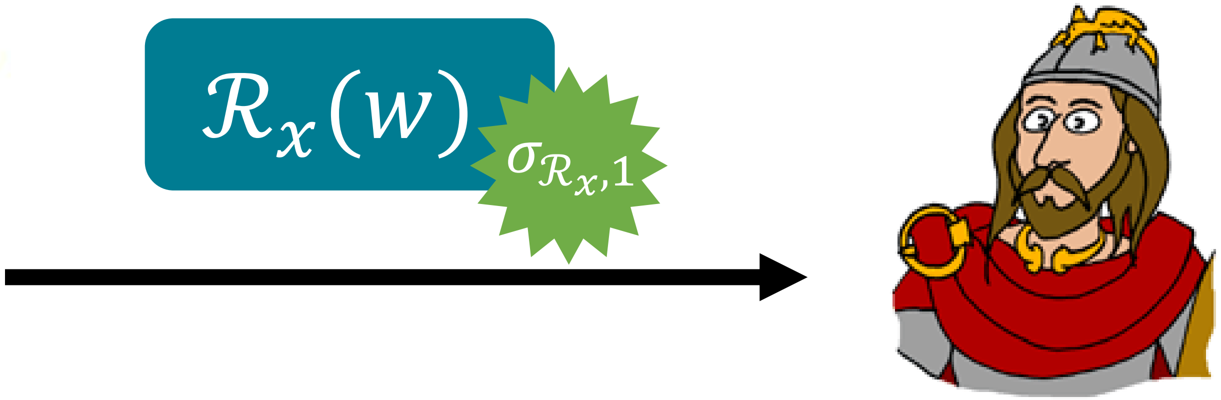
verifier $(x)$

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

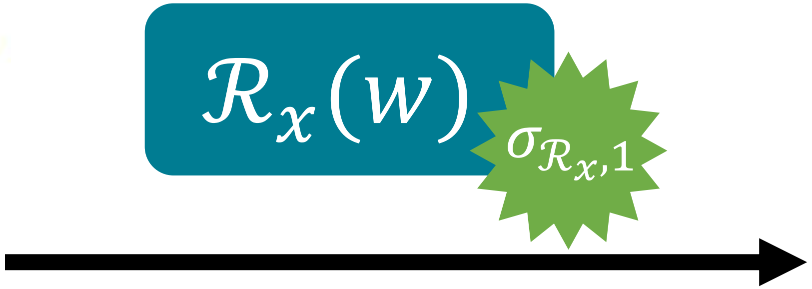



**Soundness:** Follows from unforgeability; if verifier accepts, then $\sigma_{\mathcal{R}_x,1}$ is a signature on 1 with respect to function $\mathcal{R}_x$, but $\mathcal{R}_x(w) = 0$

# Homomorphic Signatures to Preprocessing NIZKs


Suppose prover has a signature on $w$

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




**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

$w$ $\sigma_w$

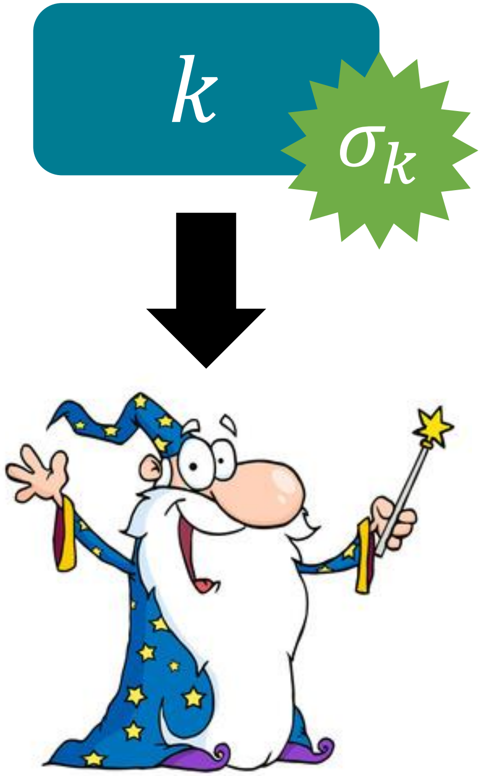
Suppose prover has a signature on $w$

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

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

**Problem:** Prover needs signature on $w$, which depends on the <u>statement</u> being proven (cannot be generated in preprocessing phase)

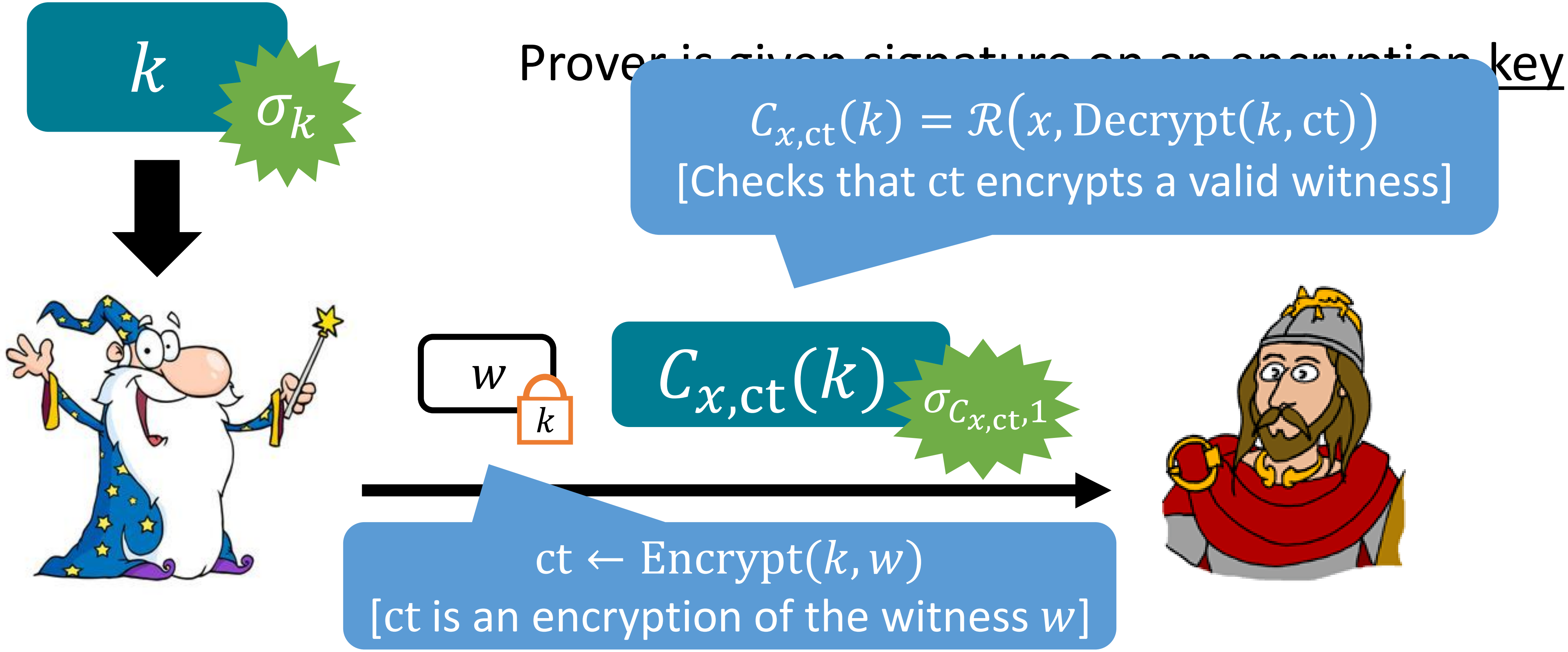
# Homomorphic Signatures to Preprocessing NIZKs




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

**Solution:** Add one layer of indirection!

# Homomorphic Signatures to Preprocessing NIZKs




**Solution:** Add one layer of indirection!

# Homomorphic Signatures to Preprocessing NIZKs

$k$ $\sigma_k$

Prove … key

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

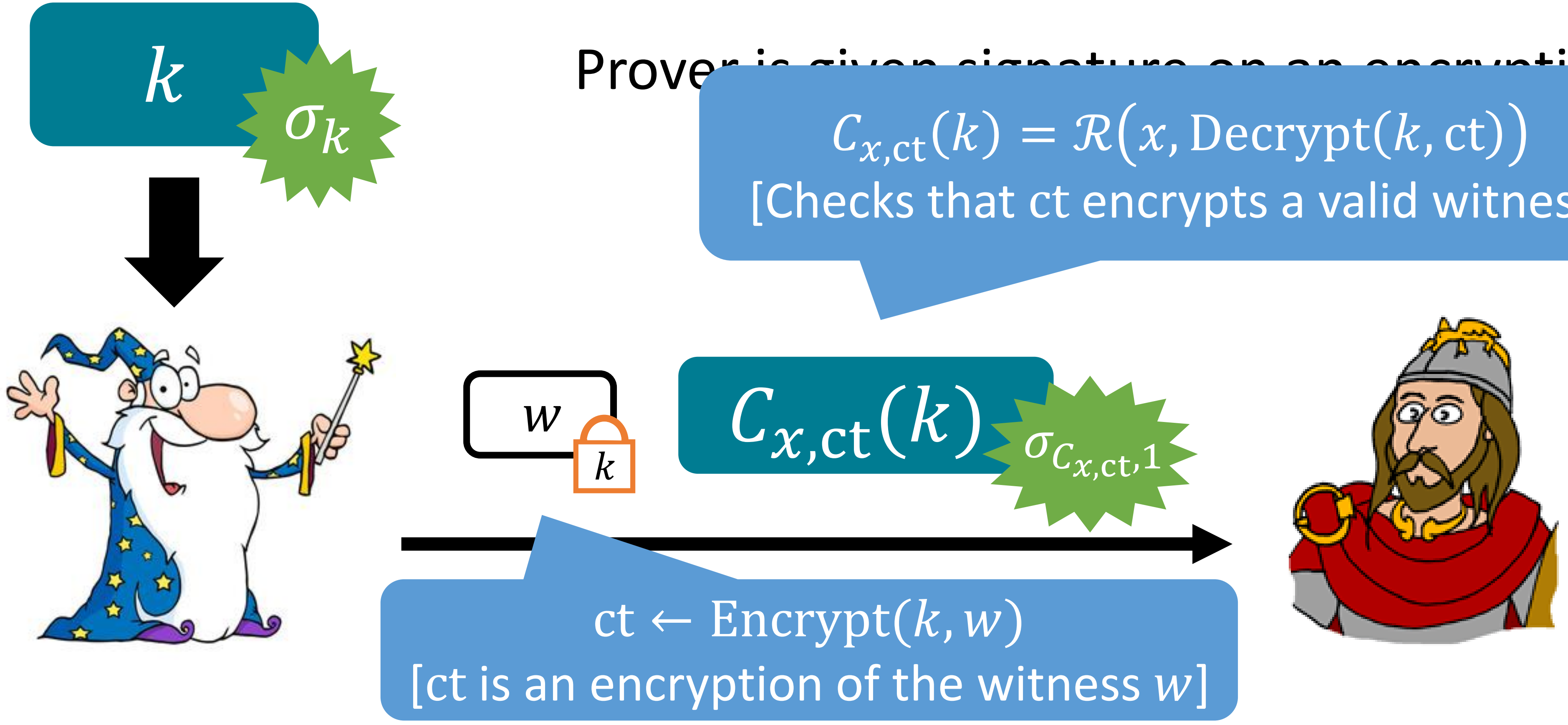
$w$ $k$

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

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

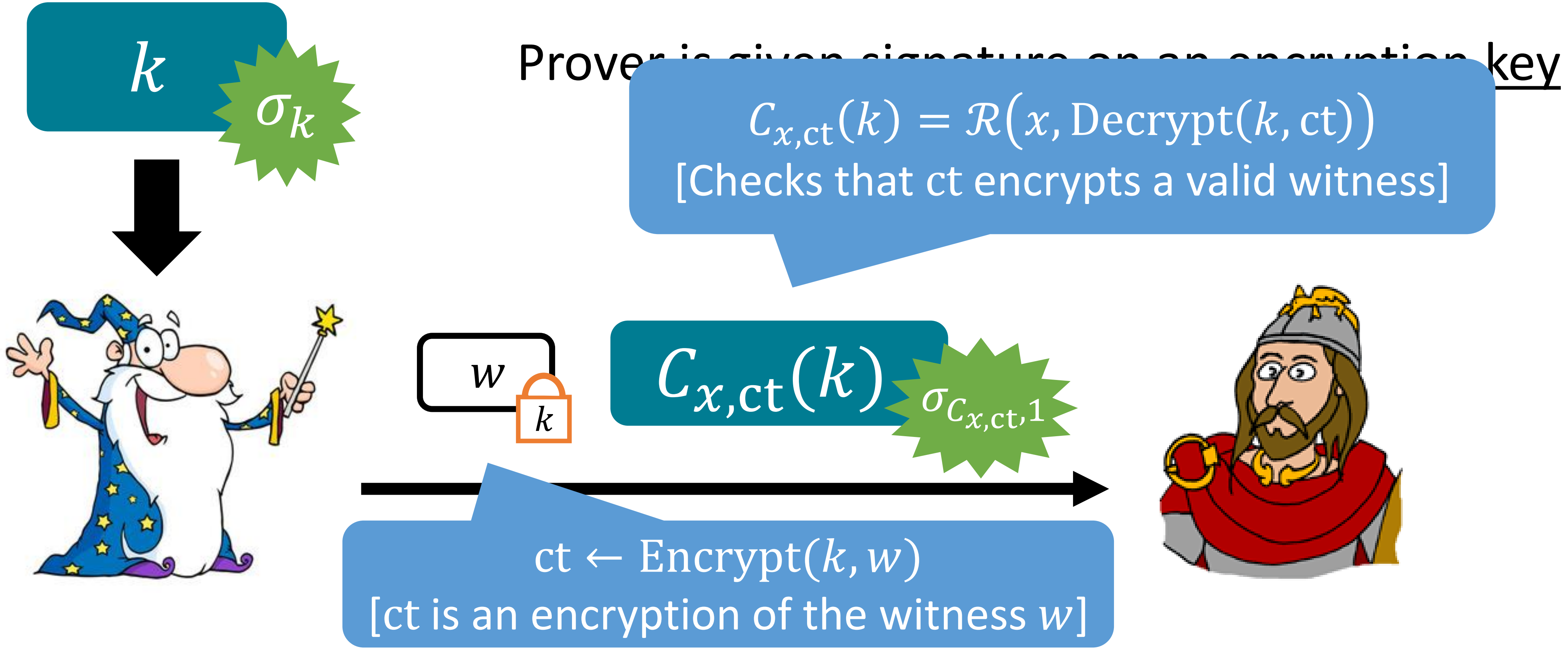
Verifier checks that $\sigma_{C_{x,\text{ct}},1}$ is a signature on 1 with respect to function $C_{x,\text{ct}}$

# Homomorphic Signatures to Preprocessing NIZKs




**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




**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

(also publishes vk)

$k$ $\sigma_k$

$k_P$

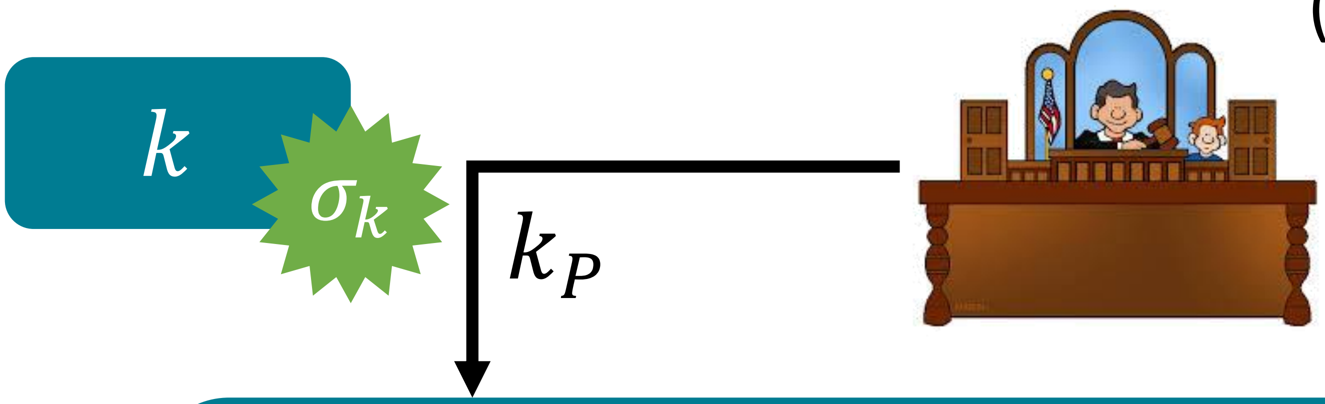
$\pi = \text{Prove}(k_P, x, w)$

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

Designated-prover NIZK from context-hiding homomorphic signatures

# Homomorphic Signatures to Preprocessing NIZKs

(also publishes vk)




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

# Homomorphic Signatures to Preprocessing NIZKs

$k$ $\sigma_k$

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

**Homomorphic signatures:** unforgeability against computationally-bounded adversaries; yields NIZK argument

**Homomorphic commitments:** unforgeability holds against unbounded adversaries; yields NIZK proof

- Unclear how to implement preprocessing efficiently, so focus will be on homomorphic signature construction

**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 $k$

# Constructing Homomorphic Signatures

[GVW15]

$x$ → $x_1$ | $x_2$ | $\cdots$ | $x_\ell$

**Message space:** will sign message bit-by-bit

---

**Verification key:**

"target matrix" for each bit of message:

gadget matrix

$\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ | $\boldsymbol{B}_1 \in \mathbb{Z}_q^{n \times m}$ | $\cdots$ | $\boldsymbol{B}_\ell \in \mathbb{Z}_q^{n \times m}$ | $\boldsymbol{G} \in \mathbb{Z}_q^{n \times m}$

**Signing key:**

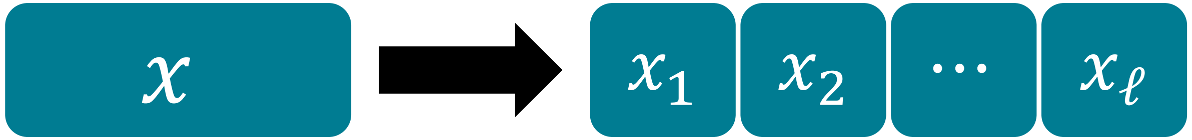
$\boldsymbol{T}_{\boldsymbol{A}} \in \mathbb{Z}_q^{m \times m}$

Trapdoor $\boldsymbol{T}_{\boldsymbol{A}}$ allows sampling short $\boldsymbol{R} \in \mathbb{Z}_q^{m \times m}$ such that $\boldsymbol{AR} = \boldsymbol{B}$ for any $\boldsymbol{B} \in \mathbb{Z}_q^{n \times m}$

[$\boldsymbol{T}_{\boldsymbol{A}}$ is an SIS trapdoor for $\boldsymbol{A}$]

# Constructing Homomorphic Signatures

[GVW15]

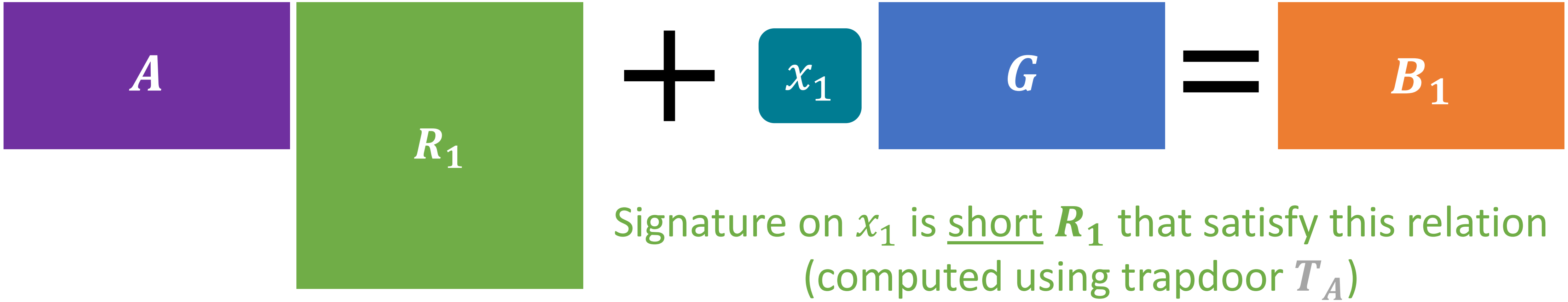



**Message space:** will sign message bit-by-bit

**Verification key:** $\boldsymbol{A}, \boldsymbol{B}_1, \ldots, \boldsymbol{B}_\ell, \boldsymbol{G} \in \mathbb{Z}_q^{n \times m}$

**Signing key:** $\boldsymbol{T}_{\boldsymbol{A}} \in \mathbb{Z}_q^{m \times m}$

**Sign message $x$ bit-by-bit:**




Signature on $x_1$ is short $\boldsymbol{R}_1$ that satisfy this relation (computed using trapdoor $\boldsymbol{T}_{\boldsymbol{A}}$)

# Constructing Homomorphic Signatures

[GVW15]

$x \rightarrow x_1 \; x_2 \; \cdots \; x_\ell$

**Message space:** will sign message bit-by-bit

---

**Verification key:** $\boldsymbol{A}, \boldsymbol{B}_1, \ldots, \boldsymbol{B}_\ell, \boldsymbol{G} \in \mathbb{Z}_q^{n \times m}$

**Signing key:** $\boldsymbol{T}_{\boldsymbol{A}} \in \mathbb{Z}_q^{m \times m}$

**Sign message $x$ bit-by-bit:**

$$\boldsymbol{A}\boldsymbol{R}_1 + x_1 \cdot \boldsymbol{G} = \boldsymbol{B}_1$$
$$\boldsymbol{A}\boldsymbol{R}_2 + x_2 \cdot \boldsymbol{G} = \boldsymbol{B}_2$$
$$\vdots \qquad\qquad \vdots$$
$$\boldsymbol{A}\boldsymbol{R}_\ell + x_\ell \cdot \boldsymbol{G} = \boldsymbol{B}_\ell$$

$$\sigma_x = (\boldsymbol{R}_1, \ldots, \boldsymbol{R}_\ell)$$

Verification consists of checking that $\boldsymbol{R}_1, \ldots, \boldsymbol{R}_\ell$ satisfy these relations

# Constructing Homomorphic Signatures

[GVW15]

$x$ → $x_1$ $x_2$ $\cdots$ $x_\ell$

**Message space:** will sign message bit-by-bit

---

**Verification key:** $A, B_1, \dots, B_\ell, G \in \mathbb{Z}_q^{n \times m}$

**Signing key:** $T_A \in \mathbb{Z}_q^{m \times m}$

**Sign message $x$ bit-by-bit:**

$$AR_1 + x_1 \cdot G = B_1$$
$$AR_2 + x_2 \cdot G = B_2$$
$$\vdots \qquad\qquad \vdots$$
$$AR_\ell + x_\ell \cdot G = B_\ell$$

GSW homomorphic operations →

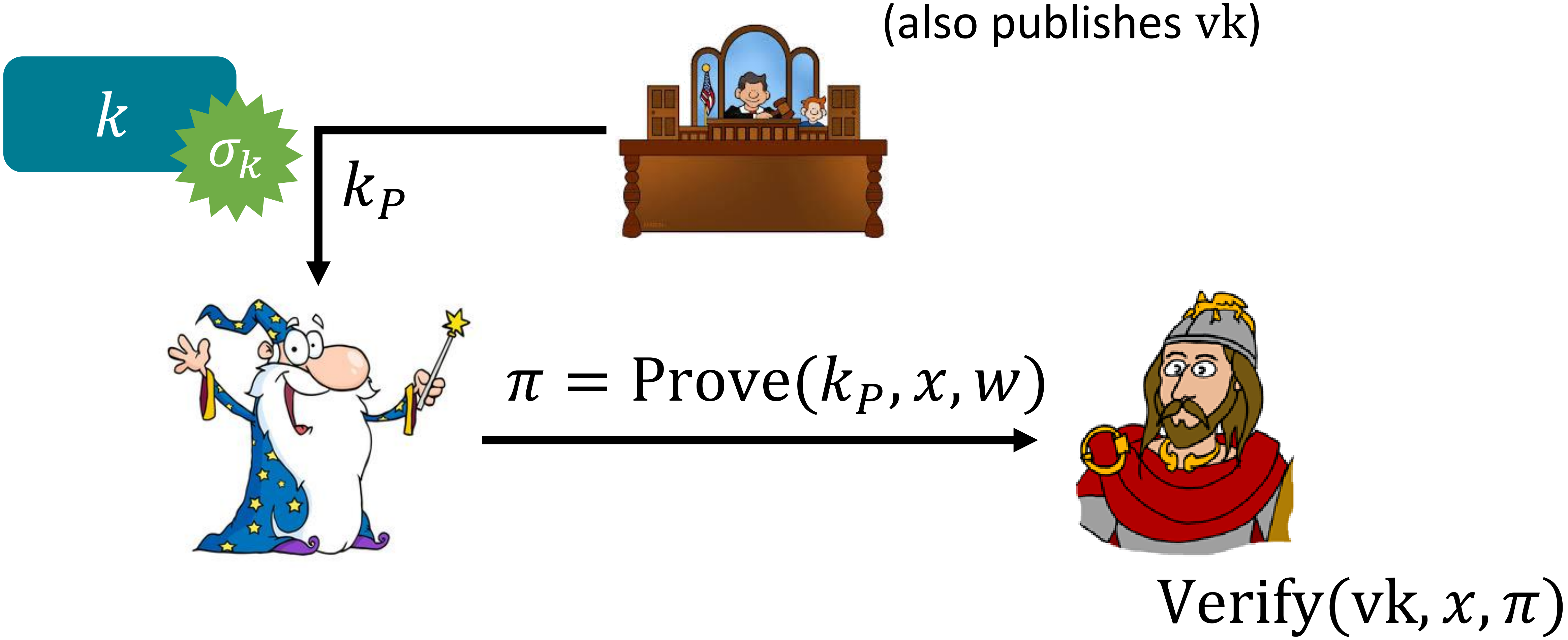
Function of $f, R_1, \dots, R_\ell$ and $x_1, \dots, x_\ell$

Function of $f, B_1, \dots, B_\ell$

$$AR_f + f(x) \cdot G = B_f$$

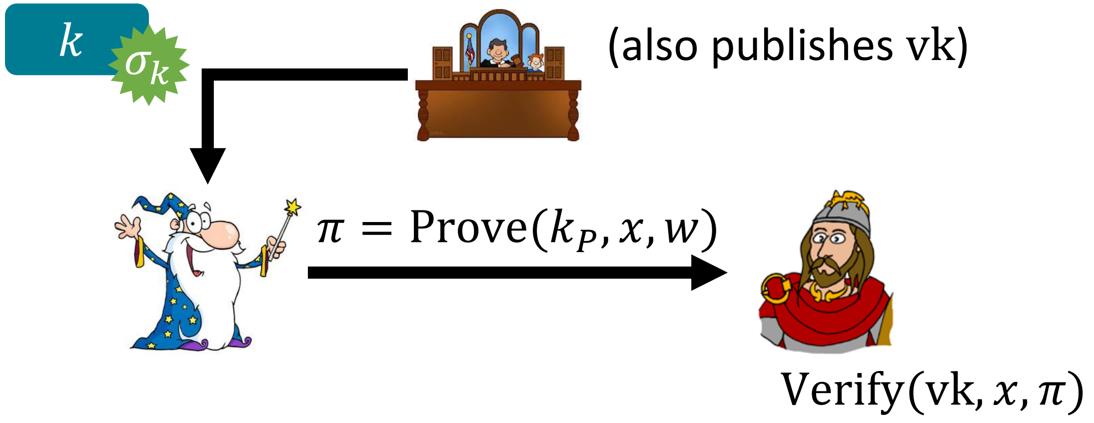
Additional techniques needed for context-hiding

# Homomorphic Signatures to Preprocessing NIZKs

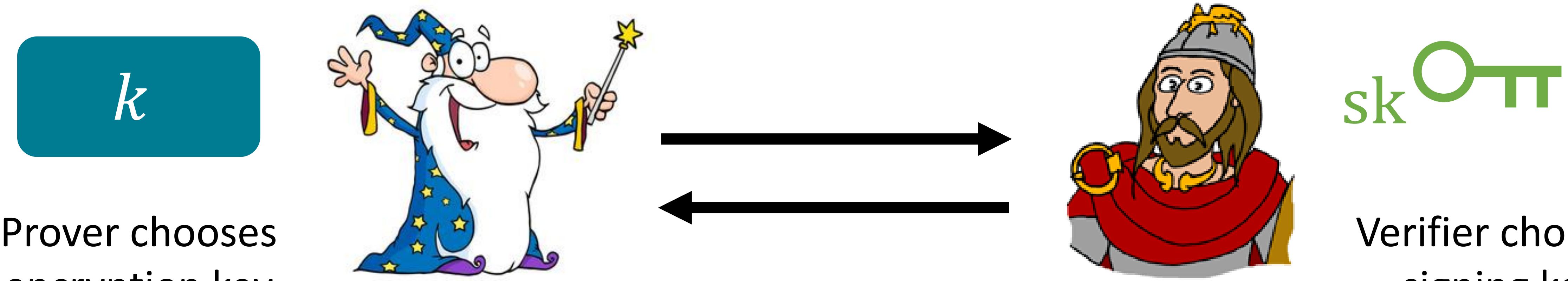



<u>Designated-prover</u> NIZK from context-hiding homomorphic signatures

# 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

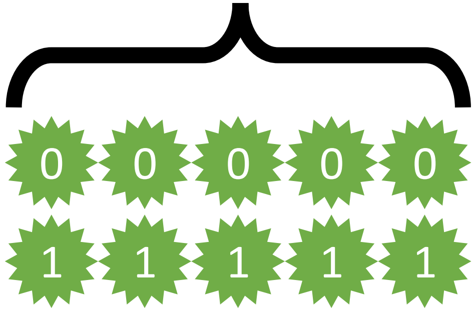
Verifier chooses signing key

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

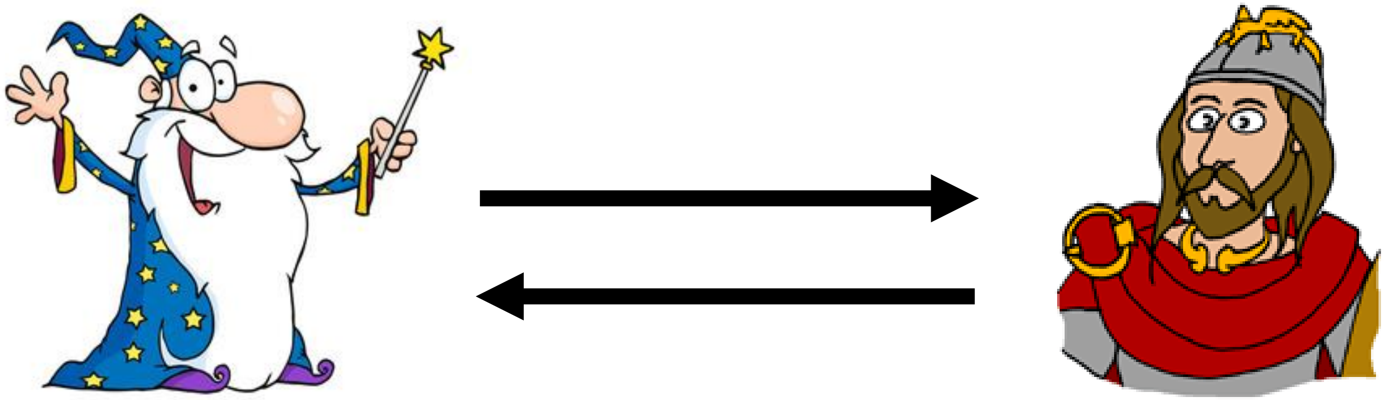
# Blind Homomorphic Signatures

- Recall that signature on the encryption key $k$ consists of $|k|$ signatures on the bits of $k$
- Prover can use oblivious transfer (OT) to obtain signatures on each bit of $k$

signatures on bits of $k$




$k$



$\mathsf{sk}$

Prover chooses encryption key

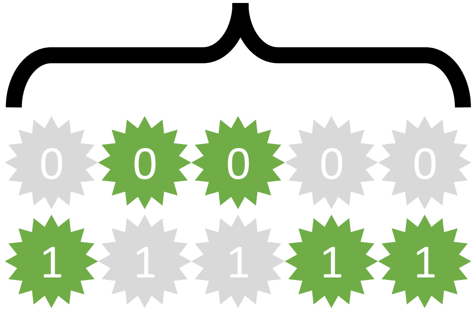
Verifier chooses signing key

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

# Blind Homomorphic Signatures

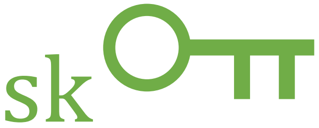
- Recall that signature on the encryption key $k$ consists of $|k|$ signatures on the bits of $k$
- Prover can use oblivious transfer (OT) to obtain signatures on each bit of $k$
- Some additional work needed for *malicious* security
  [See paper for details]

signatures on bits of $k$




$k$

OT for signatures on bits of $k$




Prover chooses encryption key

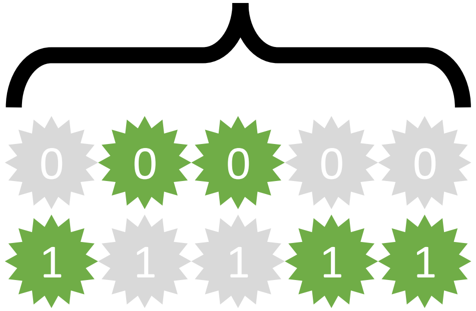
Verifier chooses signing key

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

# Blind Homomorphic Signatures

**Takeaway:** Preprocessing can be implemented using $\text{poly}(\lambda)$ parallel OT invocations

signatures on bits of $k$




$k$

OT for signatures on bits of $k$

$\text{sk}$ 

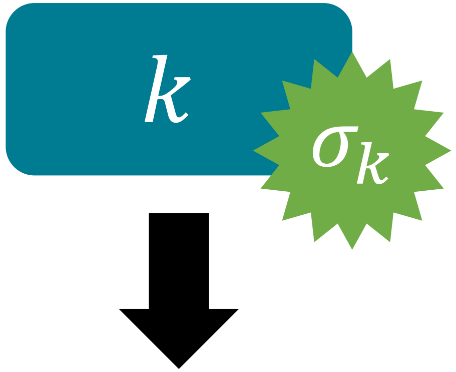
Prover chooses encryption key

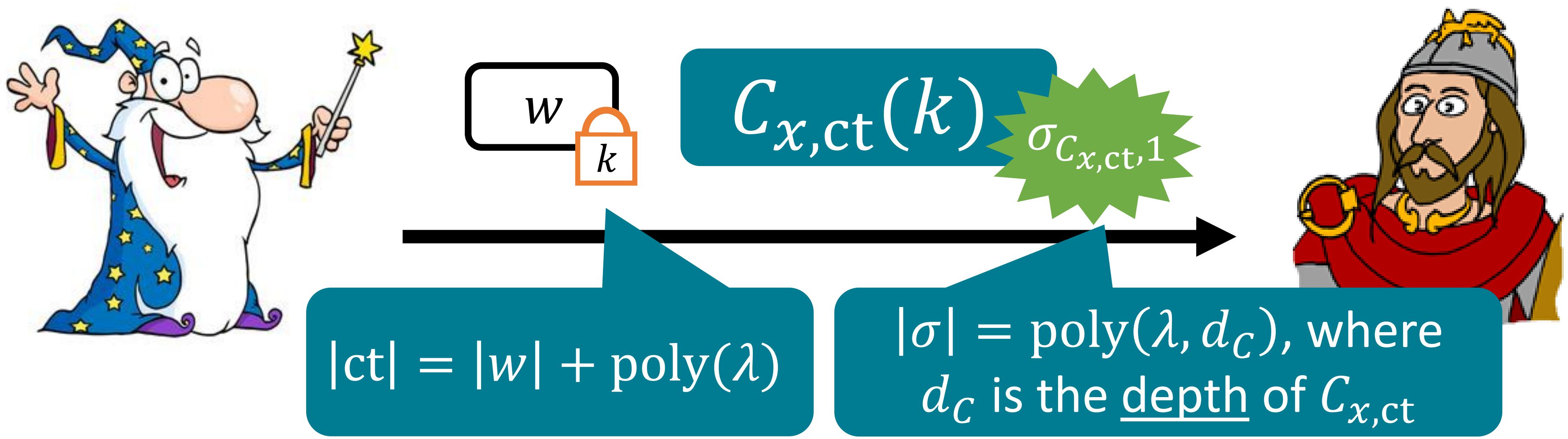
Verifier chooses signing key

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

# Proof Size and Amortization

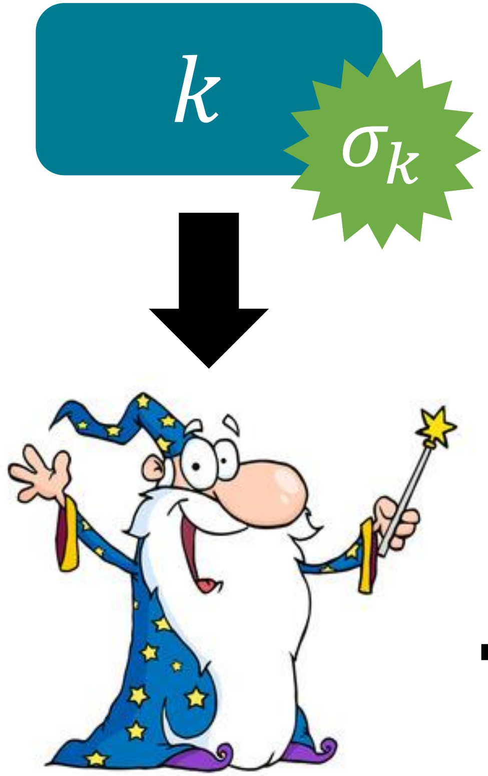
**Recall:** proof consists of an encryption of the witness and a signature



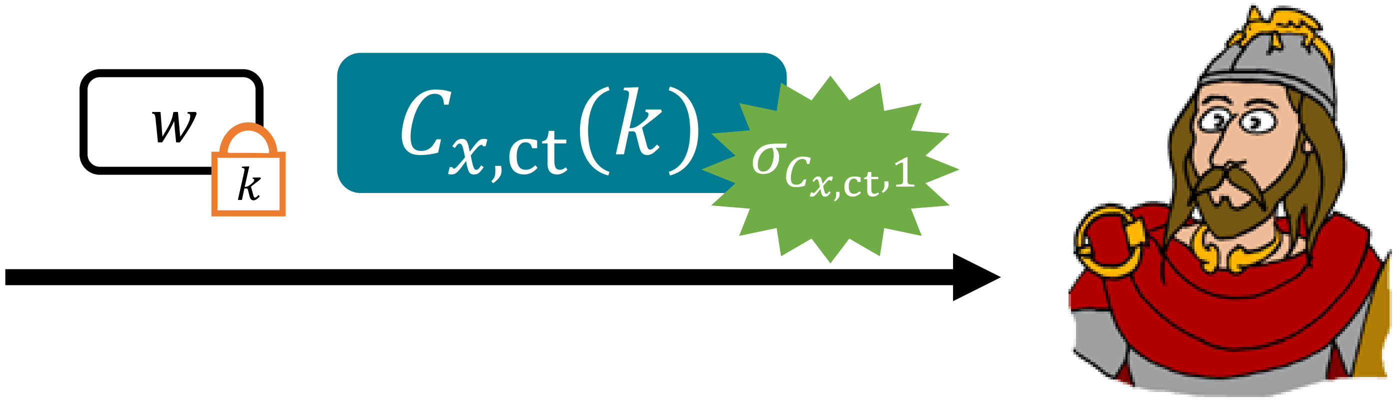




Length of NIZK is typically proportional to the size of the NP relation (rather than the depth), and moreover, the overhead is multiplicative in $\lambda$ (rather than additive)

# Proof Size and Amortization




**Observation:** If same witness used for multiple proofs, ciphertext only needs to be included once

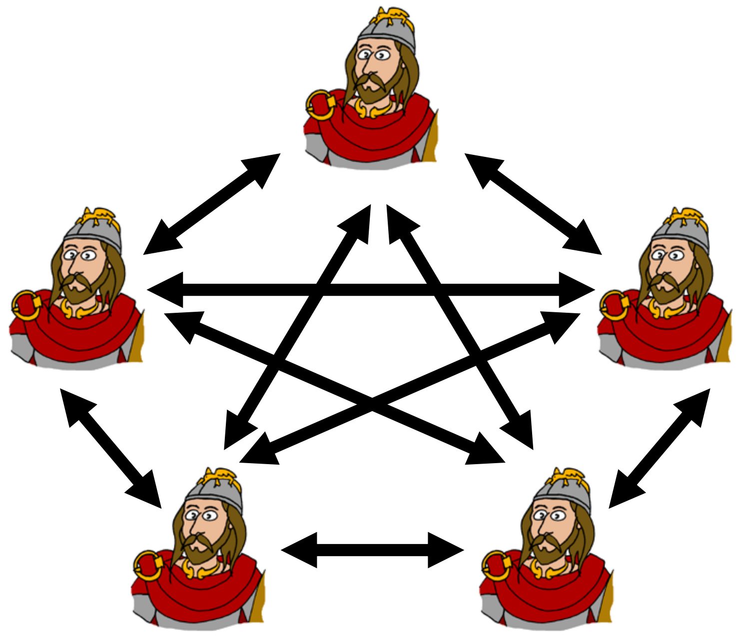



Suppose same witness $w$ used to prove statements $x_1, \dots, x_n$ (with respect to $C_1, \dots, C_n$):

$$\sum_{i \in [n]} |\pi_i| = |w| + \sum_{i \in [n]} \text{poly}(\lambda, d_i)$$

Depth of $C_1, \dots, C_n$

# A Succinct GMW Compiler



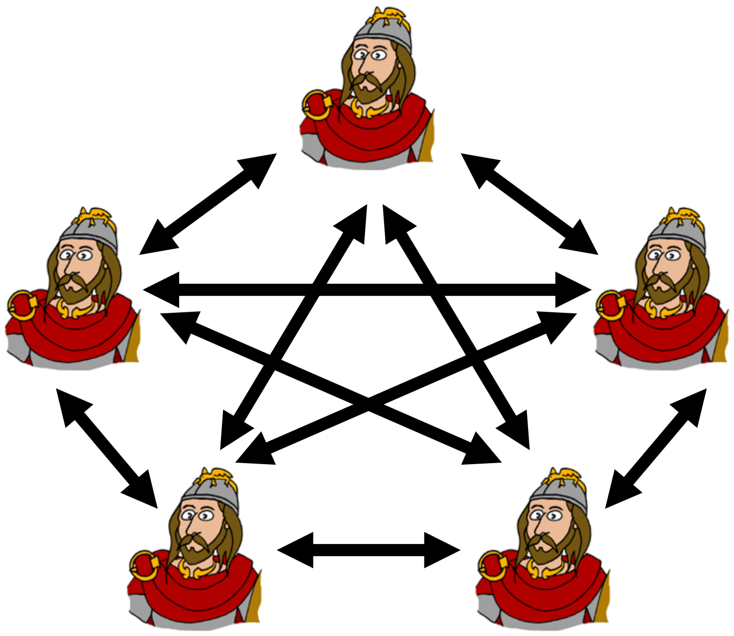
**MPC:** multiple parties seek to compute a joint function of their private inputs

Classic GMW compiler (semi-honest to malicious compiler):

1. Each party broadcasts commitment to their local input and randomness
2. Parties run a coin-flipping protocol to determine parties' randomness used for computation
3. Parties run semi-honest MPC protocol and attach a NIZK proof that each message is consistent with committed values and randomness

**Key observation:** NIZK proofs share <u>common</u> witness (the committed inputs and randomness)

# A Succinct GMW Compiler



**MPC:** multiple parties seek to compute a joint function of their private inputs

Communication overhead is

$$n \cdot |x| + \mathrm{poly}(n, \lambda, d)$$

where $|x|$ is length of parties' input and $d$ is depth (rather than *size*) of the computation

**Key observation:** NIZK proofs share common witness (the committed inputs and randomness)

# 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)

- Weaker primitive of homomorphic MAC suffices (will also require secret key to verify proofs)

**Thank you!**

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