# Unconditionally Secure NIZK in the Fine-Grained Setting

**Yuyu Wang**[1] and Jiaxin pan[2]

1. University of Electronic Science and Technology of China
2. NTNU - Norwegian University of Science and Technology

# Standard cryptography

Honest party

polynomial-time

Adversary

polynomial-time

Assumption:

- Basic ones (e.g., one-way function)
- More advanced ones (e.g., factoring, discrete logarithm, DDH, LWE)
- Exotic ones (e.g, generic groups, algebraic groups)

# Standard cryptography

Honest party

polynomial-time

Adversary

polynomial-time

Unproven

Assumption:

- Basic ones (e.g., one-way function)
- More advanced ones (e.g., factoring, discrete logarithm, DDH, LWE)
- Exotic ones (e.g, generic groups, algebraic groups)

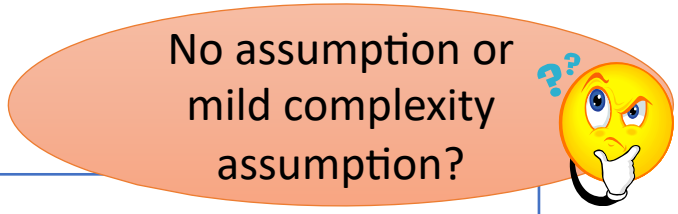
# Standard cryptography

Honest party

polynomial-time

Adversary

polynomial-time




Assumption:

- Basic ones (e.g., one-way function)
- More advanced ones (e.g., factoring, discrete logarithm, DDH, LWE)
- Exotic ones (e.g, generic groups, algebraic groups)

# Fine-grained cryptography

Honest party

Adversary

An honest party uses less resources than the adversary

# Fine-grained cryptography

Honest party

An honest party uses less resources than the adversary

Adversary

The resources of an adversary can be a-prior bounded

# Fine-grained cryptography

Honest party

An honest party uses less resources than the adversary

Adversary

The resources of an adversary can be a-prior bounded

- Based only on mild assumption or even no assumption

# Fine-grained cryptography

Honest party

An honest party uses less resources than the adversary

Adversary

The resources of an adversary can be a-prior

Existing fine-grained primitives:
NIKE [Mer78], OWF [BC20], PKE [DVV16], verifiable computation [CG18], HPS and trapdoor one-way function [EWT21], ABE [WPC21]

- Based only on mild assumption or even no asssumption

# Fine-grained cryptography

Honest party

An honest party uses less resources than the adversary

Adversary

The resources of an adversary can be a-prior

Existing fine-grained primitives:
NIZK [WP22] (Eurocrypt ’22)

- Bas… …nditionally secure

runs in $NC^1$

# Fine-grained cryptography

Honest party

An honest party uses less resources than the adversary

Adversary

The resources of an adversary can be a-prior

Existing fine-grained primitives:
NIZK [WP22] (Eurocrypt ’22)

- Bas… …nditionally secure

secure against adversaries in $NC^1$ if $NC^1 \neq \oplus L/poly$

# Fine-grained cryptography

Honest party

An honest party uses less resources than the adversary

Adversary

The resources of an adversary can be a-prior bounded.

Unconditionally secure fine-grained NIZK?

- Based only on mild a

# Our results

A fully fine-grained NIZK for AC0-circuit satisfiability (SAT)

- ❖the CRS generator, prover, verifier, simulator run in AC0
- ❖Perfect soundness and composable zero-knowledge against AC0
  (Dual mode: perfect zero-knowledge and computational soundness against AC0 )
- ❖No assumption

# Our results

A fully fine-grained NIZK for AC0-circuit satisfiability (SAT)

- ❖the CRS generator, prover, verifier, simulator run in AC0
- ❖Perfect soundness and composable zero-knowled[illegible]against AC0
  (Dual mode: perfect zero-knowledge and co[illegible]nal soundness against AC0 )
- ❖No assumption

Circuits with constant depth,
polynomial size, and unbounded fan-in
using AND, OR, and NOT gates.

# Our results

A fully fine-grained NIZK for AC0-circuit satisfiability (SAT)

- ❖the CRS generator, prove[illegible]verifier, simulator run in AC0
- ❖Perfect soundness and [illegible]posable zero-knowledge against AC0
  (Dual mode: perfect [illegible]owledge and computational soundness against AC0 )
- ❖No assumption

A statement circuit cannot go beyond AC0. Otherwise, even the honest prover in AC0 cannot decide with the witness whether the statement is true or not.

# Our results

A fully fine-grained NIZK for AC0-circuit satisfiability (SAT)

- ❖the CRS generator, prove[illegible] verifier, simulator run in AC0
- ❖Perfect soundness and [illegible]posable zero-knowledge against AC0
  (Dual mode: perfect ze[illegible]wledge and computational soundness against AC0 )
- ❖No assumption

If we allow the prover to run in polynomial time as in [BDK20] (Crypto '20), our NIZK supports all statements in NP.

# Our results

A fully fine-grained NIZK for AC0-circuit satisfiability (SAT)

- ❖the CRS generator, prover, verifier, simulator run in AC0
- ❖Perfect soundness and composable zero-knowledge against AC0
  (Dual mode: perfect zero-knowledge and computational soundness against AC0 )
- ❖No assumption

Extensions in the AC0-fine-grained setting:

1. Unconditionally secure non-interactive zaps
2. Unconditionally secure NIZKs in the URS model

# Applications

Applications:

1. Unconditional security against adversaries with limited parallel running-time.

2. Systems requiring “online security”, where attacks are valid only if they succeed immediately:

   - Our NIZK with composable zero-knowledge can be used to protect secrets only valuable in a short period of time.

   - The dual mode with computational soundness and perfect zero-knowledge perfectly protects secrets and guarantees security in a system requiring users to provide proofs in a short time.

# Applications

Applications:

1. Unconditional security against adversaries with limited parallel running-time.

2. Systems requiring “online security”, where attacks are valid only if they succeed immediately:

   - Our NIZK with composable zero-knowledge can be used to protect secrets only valuable in a short period of time.

   - The dual mode with computational soundness and perfect zero-knowledge perfectly protects secrets and guarantees security in a system requiring users to provide proofs in a short time.

# Applications

Applications:

1. Unconditional security against adversaries with limited parallel running-time.

2. Systems requiring “online security”, where attacks are valid only if they succeed immediately:

    - Our NIZK with composable zero-knowledge can be used to protect secrets only valuable in a short period of time.

    - The dual mode with computational soundness and perfect zero-knowledge perfectly protects secrets and guarantees security in a system requiring users to provide proofs in a short time.

# Applications

Applications:

1. Unconditional security against adversaries with limited parallel running-time.

2. Systems requiring “online security”, where attacks are valid only if they succeed immediately:

    - Our NIZK with composable zero-knowledge can be used to protect secrets only valuable in a short period of time.

    - The dual mode with computational soundness and perfect zero-knowledge perfectly protects secrets and guarantees security in a system requiring users to provide proofs in a short time.

# Impact of Our Work

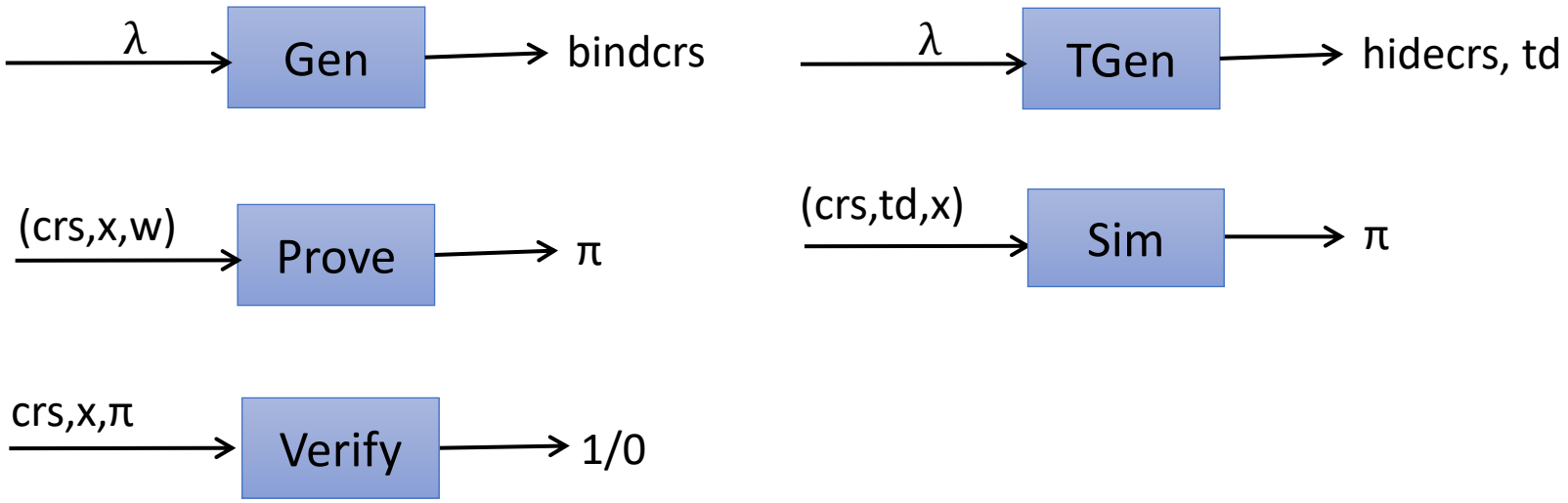
Before our work, it seemed that cryptographic assumptions implying PKE were necessary for NIZK in the standard model. Namely, NIZK seemed to be in Cryptomania.

In the AC0 fine-grained setting, only “minicrypt primitives” (e.g., OWF, SKE, wPRF) are known to exist, while we achieved NIZK.

NIZK is in minicrypt?

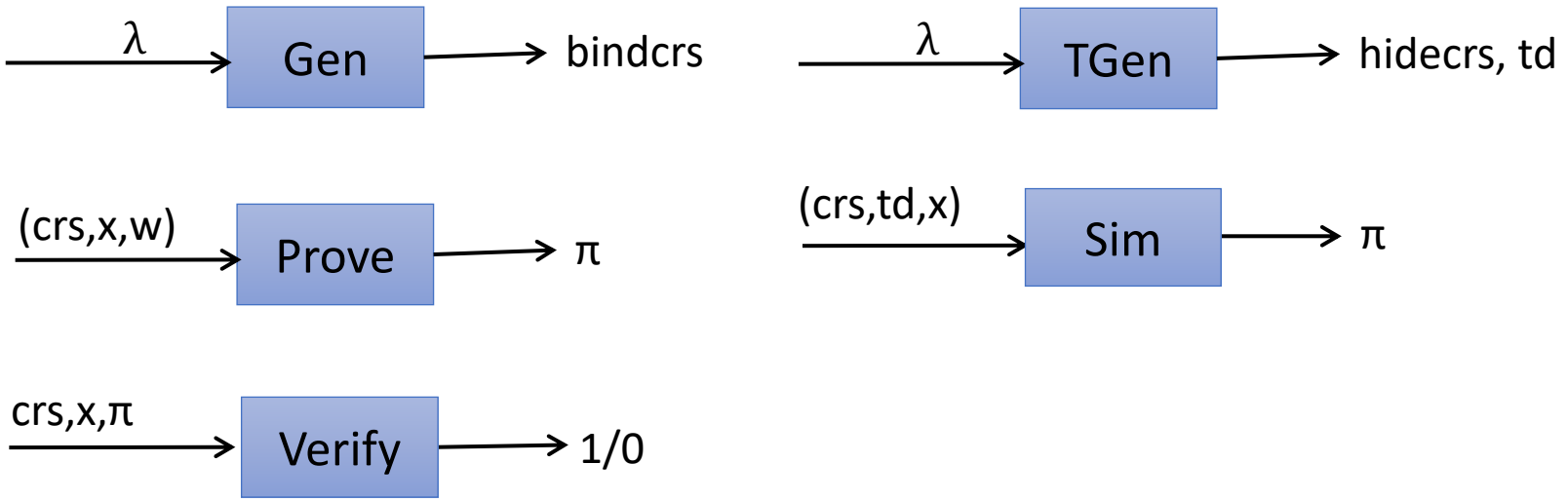
# Definition of NIZK

$x \in L$ iff $\exists w$ s. t. $R(x, w) = 1$



# Definition of NIZK

$x \in L$ iff $\exists w$ s. t. $R(x, w) = 1$

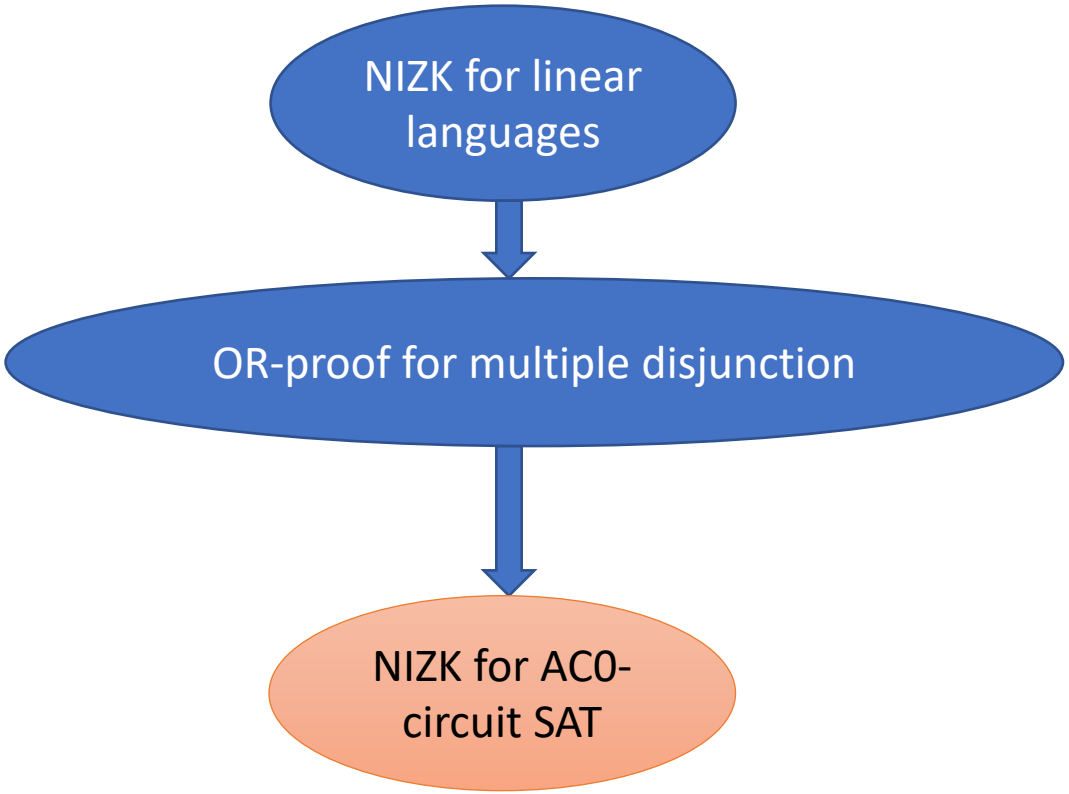



Completeness: honest proofs must pass the verification.

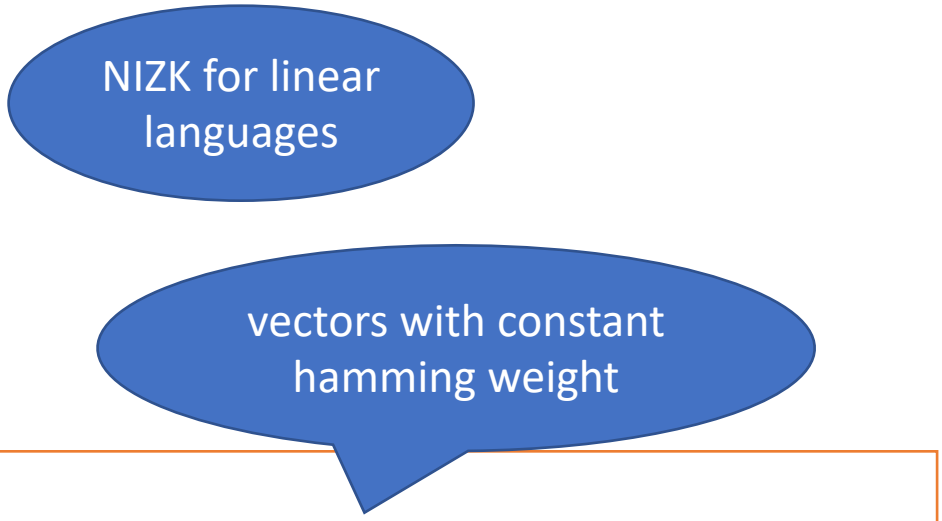
Perfect soundness: when crs is binding, there exists no valid proof for x if $x \notin L$.

(Composable) zero knowledge: bindcrs and hidecrs are indistinguishable, and when crs is hiding, Sim perfectly simultates honest proofs.

# NIZK for AC0-circuit SAT



# NIZK for linear languages




Two facts:

1. Inner product of two sparse vectors is computable in AC0.

# NIZK for linear languages


Two facts:

1. Inner product of two sparse vectors is computable in AC0.
2. The parity of a random vector is not computable in AC0 [Hås14,IM11].

# NIZK for linear languages

$$\mathbf{E} = \begin{pmatrix} 1 & & & & \\ 1 & 1 & & & \\ & & \ddots & & \\ & & & 1 & 1 \\ & & & & 1 \end{pmatrix} \in \{0,1\}^{\lambda \times (\lambda - 1)} \qquad \mathbf{e} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \in \{0,1\}^{\lambda}$$

# NIZK for linear languages

The parity of a random string is not computatble in AC0 [Hås14,IMP12].

$$D_0=\{Et \mid t \leftarrow \{0,1\}^{\lambda-1}\} \approx_{AC0} D_1=\{Et+e \mid t \leftarrow \{0,1\}^{\lambda-1}\}$$

$$\mathbf{E}=\begin{pmatrix} 1 & & & & \\ 1 & 1 & & & \\ & & \ddots & & \\ & & & 1 & 1 \\ & & & & 1 \end{pmatrix} \in \{0,1\}^{\lambda\times(\lambda-1)} \qquad \mathbf{e}=\begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \in \{0,1\}^{\lambda}$$

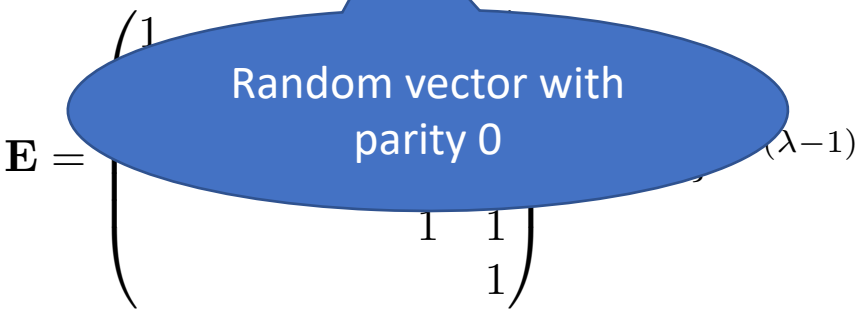
# NIZK for linear languages

The parity of a random string is not computatble in AC0 [Hås14,IMP12].

$D_0=\{Et | t \leftarrow \{0,1\}^{\lambda-1}\} \approx_{AC0} D_1=\{Et+e | t \leftarrow \{0,1\}^{\lambda-1}\}$

Random vector with parity 0

Random vector with parity 1



# NIZK for linear languages

The parity of a random string is not computatble in AC0 [Hås14,IMP12].

$$D_0=\{Et|t\leftarrow\{0,1\}^{\lambda-1}\}\approx_{AC0} D_1=\{Et+e|t\leftarrow\{0,1\}^{\lambda-1}\}$$

$$\mathbf{E}=\begin{pmatrix}1 & & & & \\ 1 & 1 & & & \\ & & \ddots & & \\ & & & 1 & 1 \\ & & & & 1\end{pmatrix}\in\{0,1\}^{\lambda\times(\lambda-}$$

Sampleable in AC0 since the row vectors in E are sparse

# NIZK for linear languages

Column vectors are sparse

$M \in \{0,1\}^{n \times t}$

$\exists\ w\ s.t.\ x=Mw$

$r = Et + e \quad (t \leftarrow \{0,1\}^{\lambda-1})$

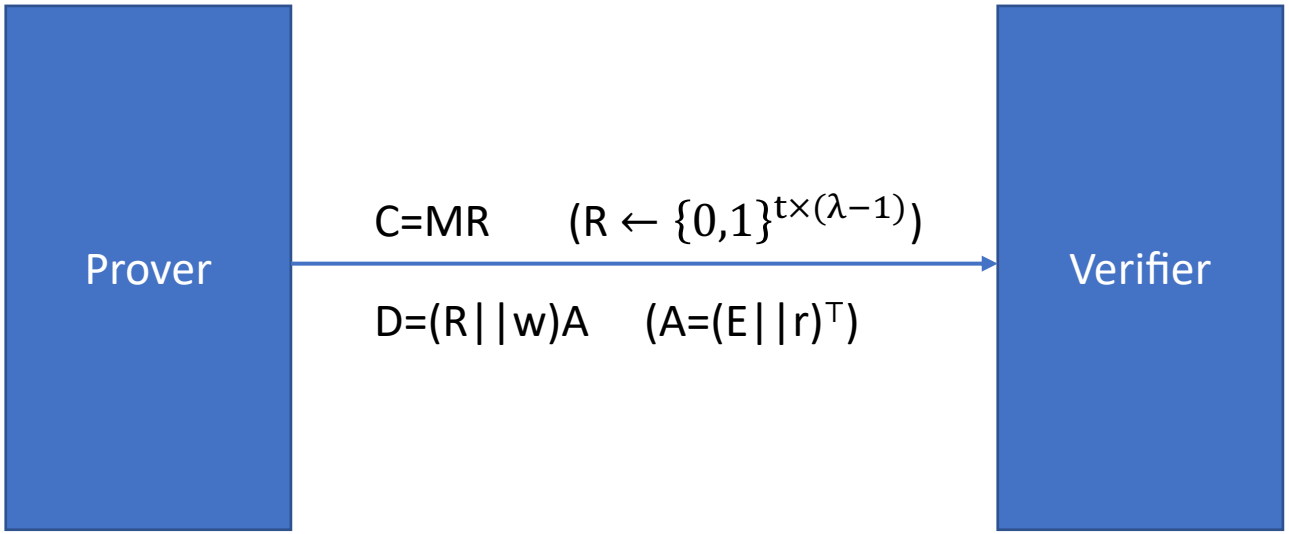
bindCRS

# NIZK for linear languages

$M \in \{0,1\}^{n \times t}$

$r = Et + e \quad (t \leftarrow \{0,1\}^{\lambda-1})$

$\exists$ w s. t. x=Mw

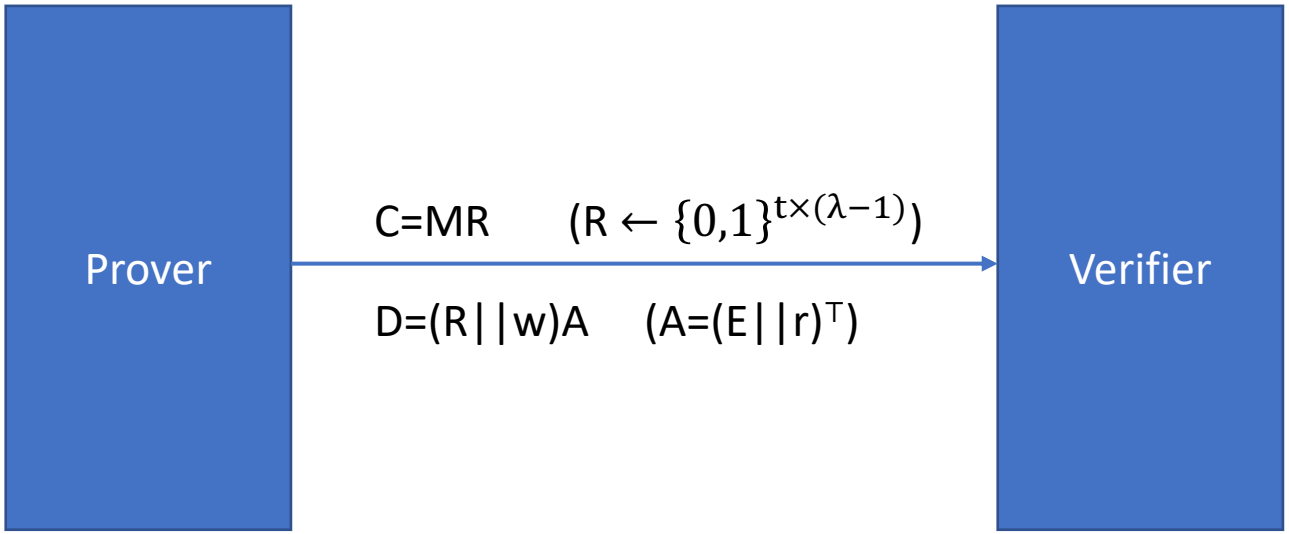



Verify if (C||x)A=MD

# NIZK for linear languages

$M \in \{0,1\}^{n \times t}$ $\quad r = Et + e \ \ (t \leftarrow \{0,1\}^{\lambda-1})$

$\exists$ w s. t. x=Mw

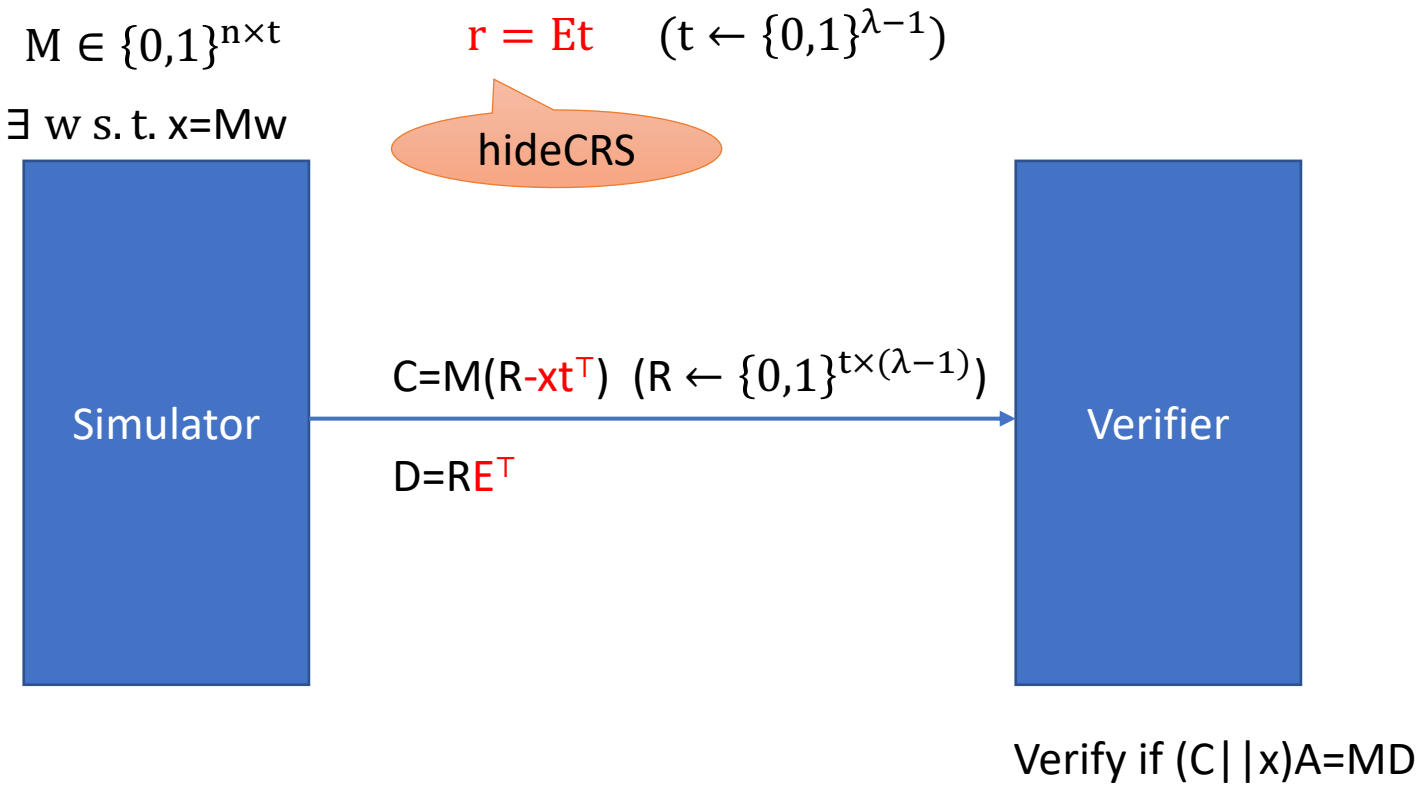



Verify if (C||x)A=MD

Complexity: row vectors of M and column vectors of A are sparse => prover and verifier run in AC0.

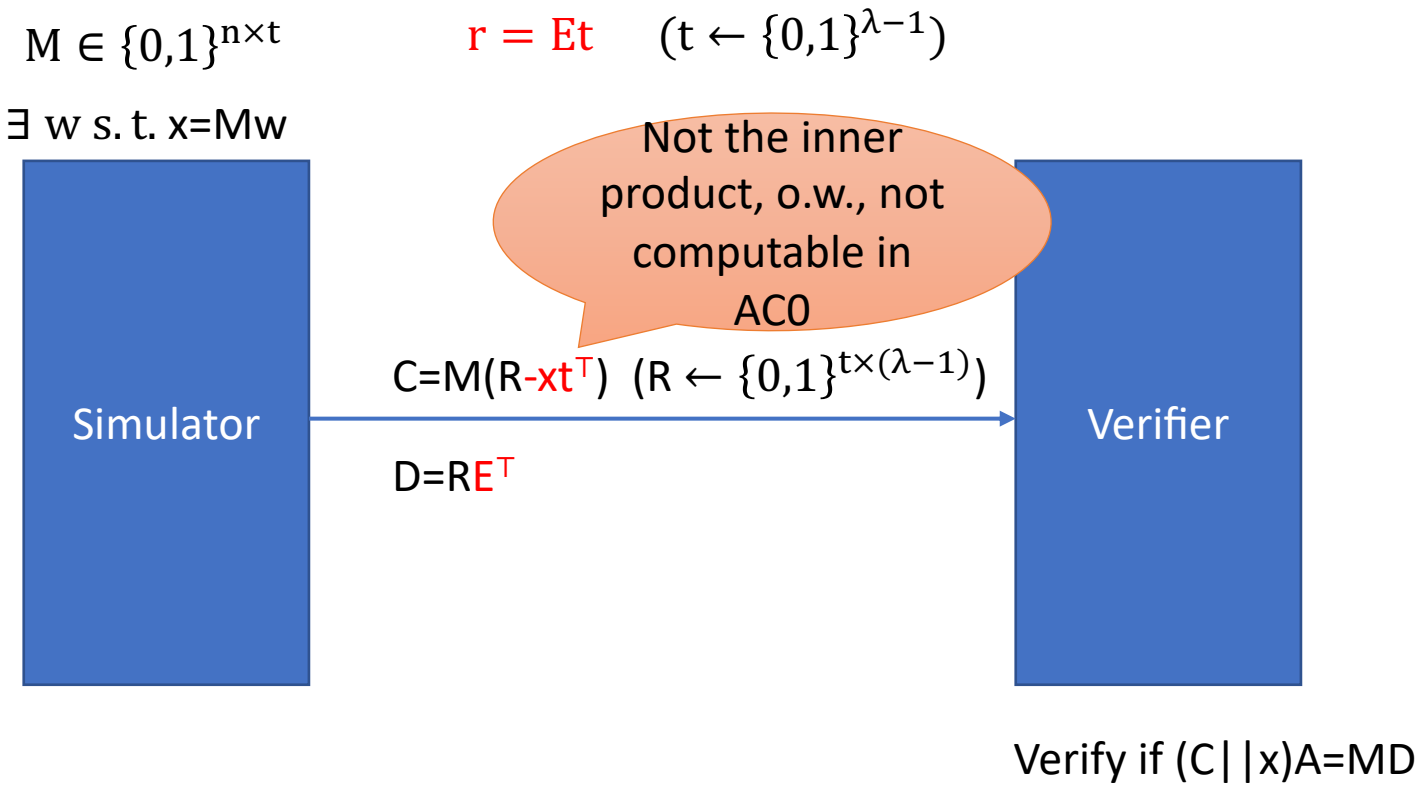
Perfect soundness: A is of full rank => C||x=MD$A^{-1}$ => x is in Span(M)

# NIZK for linear languages



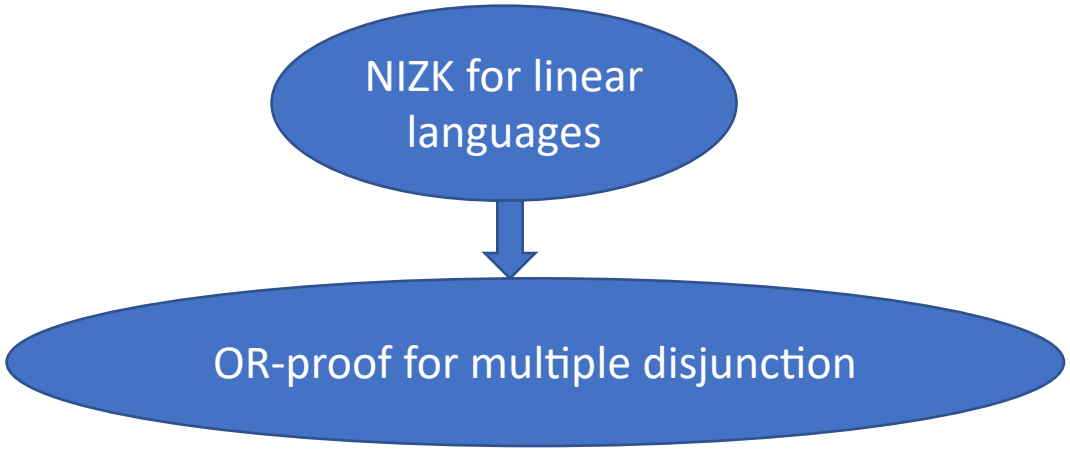

Composable zero-knowledge: when switching r to a random vector in Span(E), there exists a simulator in AC0 that perfectly simulates the prover.

# NIZK for linear languages




Composable zero-knowledge: when switching r to a random vector in Span(E), there exists a simulator in AC0 that perfectly simulates the prover.

# NIZK for AC0 circuits



# OR-proof for 1-out-of-2 disjunction

$M_0 \in \{0,1\}^{n \times t}$
$M_1 \in \{0,1\}^{n \times t}$
$x_j = M_j w$ for some $j \in \{0,1\}$

$r = Et + e \ \ (t \leftarrow \{0,1\}^{\lambda - 1})$

Prover

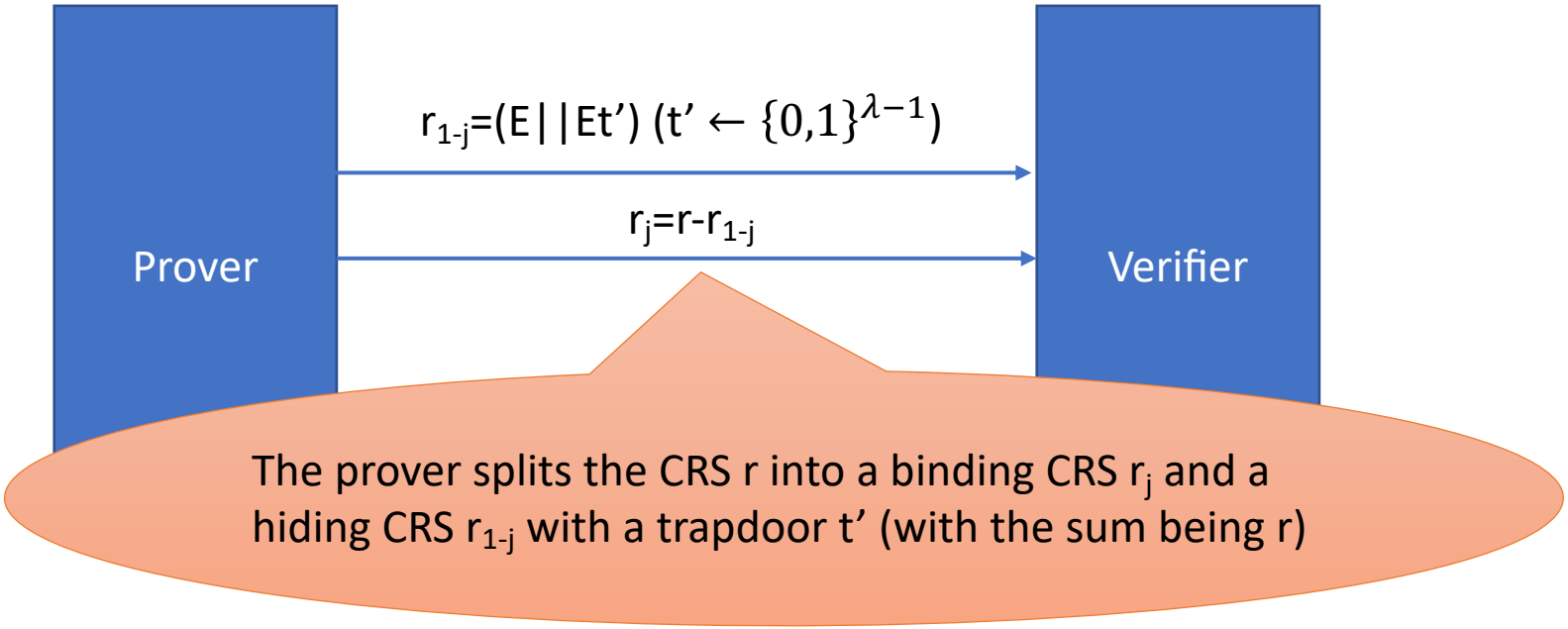
Verifier

# OR-proof for 1-out-of-2 disjunction

$M_0 \in \{0,1\}^{n \times t}$
$M_1 \in \{0,1\}^{n \times t}$

$r = Et + e \ \ (t \leftarrow \{0,1\}^{\lambda-1})$

$x_j = M_j w$ for some $j \in \{0,1\}$

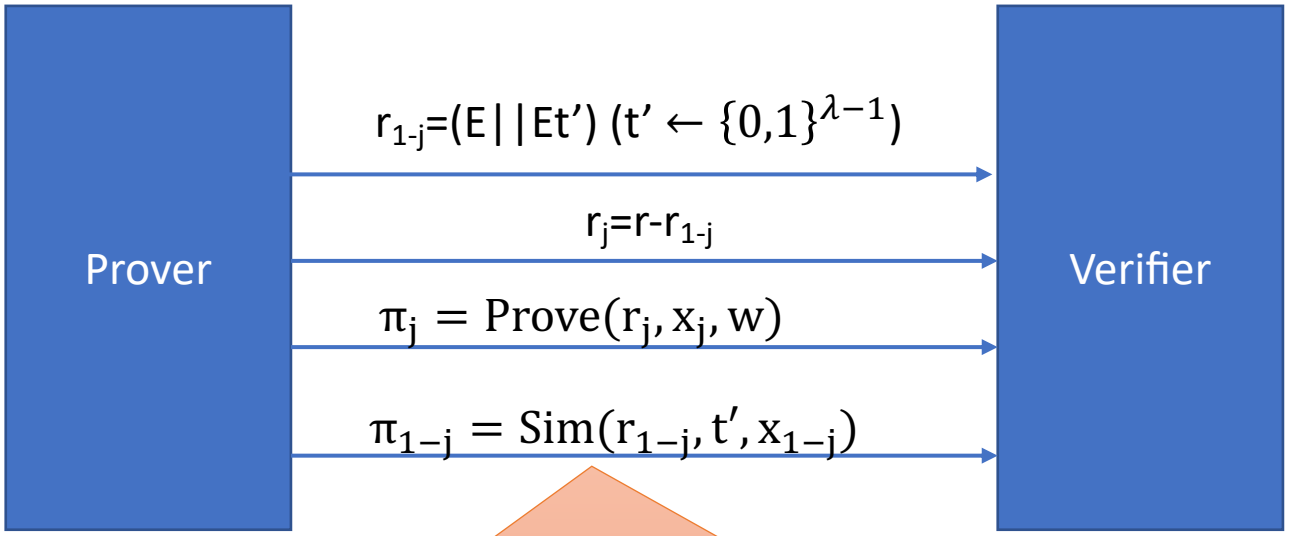


# OR-proof for 1-out-of-2 disjunction

$M_0 \in \{0,1\}^{n \times t}$
$M_1 \in \{0,1\}^{n \times t}$

$x_j = M_j w$ for some $j \in \{0,1\}$

$r = Et + e \ \ (t \leftarrow \{0,1\}^{\lambda-1})$



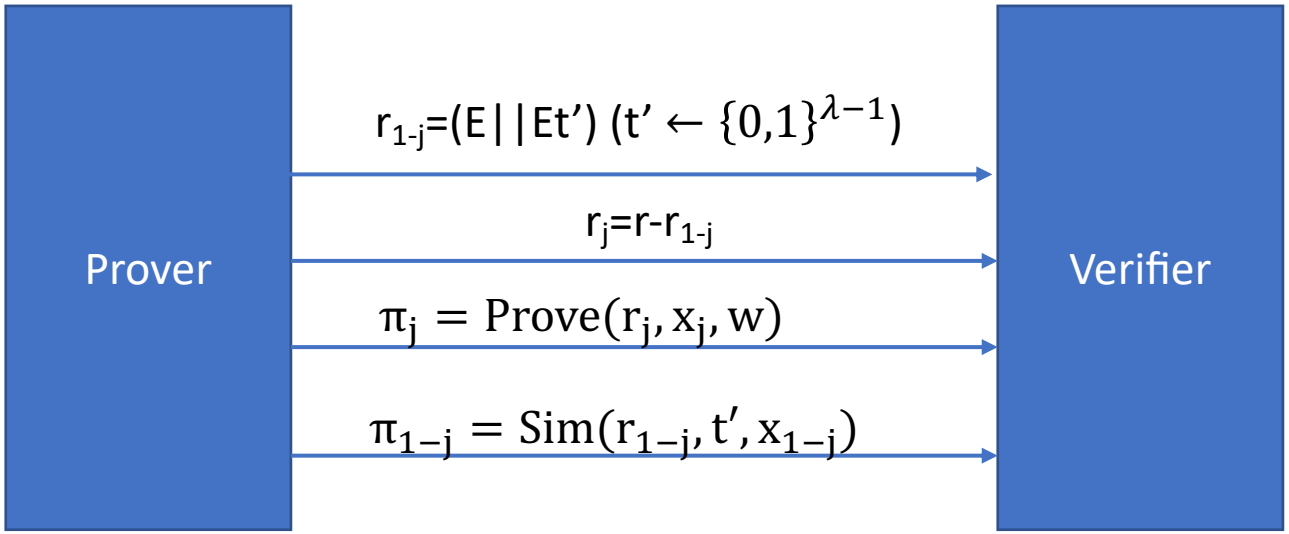

Then it generates proofs for $x_j$ and $x_{1-j}$ with w and t' respectively by making use of the prover and simulator of NIZK for linear languages.

# OR-proof for 1-out-of-2 disjunction

$M_0 \in \{0,1\}^{n \times t}$
$M_1 \in \{0,1\}^{n \times t}$

$r = Et + e \ \ (t \leftarrow \{0,1\}^{\lambda-1})$

$x_j = M_j w$ for some $j \in \{0,1\}$




Soundness: when the parity of r is 1, either $r_0$ or $r_1$ has parity 1 (binding).

Zero-knowledge: when the parity of r is 0, both $r_0$ and $r_1$ have parity 0 (hiding).

# OR-proof for 1-out-of-n disjunction

A natural idea:

1. Assuming the jth statement is true, let the prover split the original bindCRS r into $r_1, \cdots, r_n$ s.t. $r = r_1 + r_2 + \cdots + r_n$ and only $r_j$ has parity 1.
2. The verifier checks $r = r_1 + r_2 + \cdots + r_n$.

Neither can be performed in AC0

# OR-proof for 1-out-of-n disjunction

Solution: a double layers sampler to sample n CRSs with the sum being r in AC0.

First layer:
Sample n trapdoors $(t_1, \cdots t_n)$ with the sum being a 0-vector.
- by sampling n-bit strings with parity being 0

# OR-proof for 1-out-of-n disjunction

Solution: a double layers sampler to sample n CRSs with the sum being r in AC0.

First layer:
Sample n trapdoors $(t_1, \cdots t_n)$ with the sum being a 0-vector.

Second layer:
1. Sample n hiding CRSs as $(r_1, \cdots, r_n) = (Et_1, \cdots, Et_n)$.

We now obtain n hiding CRSs with the sum being 0.

# OR-proof for 1-out-of-n disjunction

Solution: a double layers sampler to sample n CRSs with the sum being r in AC0.

First layer:
Sample n trapdoors $(t_1, \cdots t_n)$ with the sum being a 0-vector.

Second layer:
1. Sample n hiding CRSs as $(r_1, \cdots, r_n) = (Et_1, \cdots, Et_n)$.
2. Add $r_j$ with r.

We now obtain n random CRSs with the sum being r and the jth CRS being binding.

# OR-proof for 1-out-of-n disjunction

Solution: a double layers sampler to sample n CRSs with the sum being r in AC0.

First layer:
Sample n trapdoors $(t_1, \cdots t_n)$ with the sum being a 0-vector.

Second layer:
1. Sample n hiding CRSs as $(r_1, \cdots, r_n) = (Et_1, \cdots, Et_n)$.
2. Add $r_j$ with r.

This sampler allows a prover to split r into n CRSs in AC0

# OR-proof for 1-out-of-n disjunction

Solution: a double layers sampler to sample n CRSs with the sum being r in AC0.

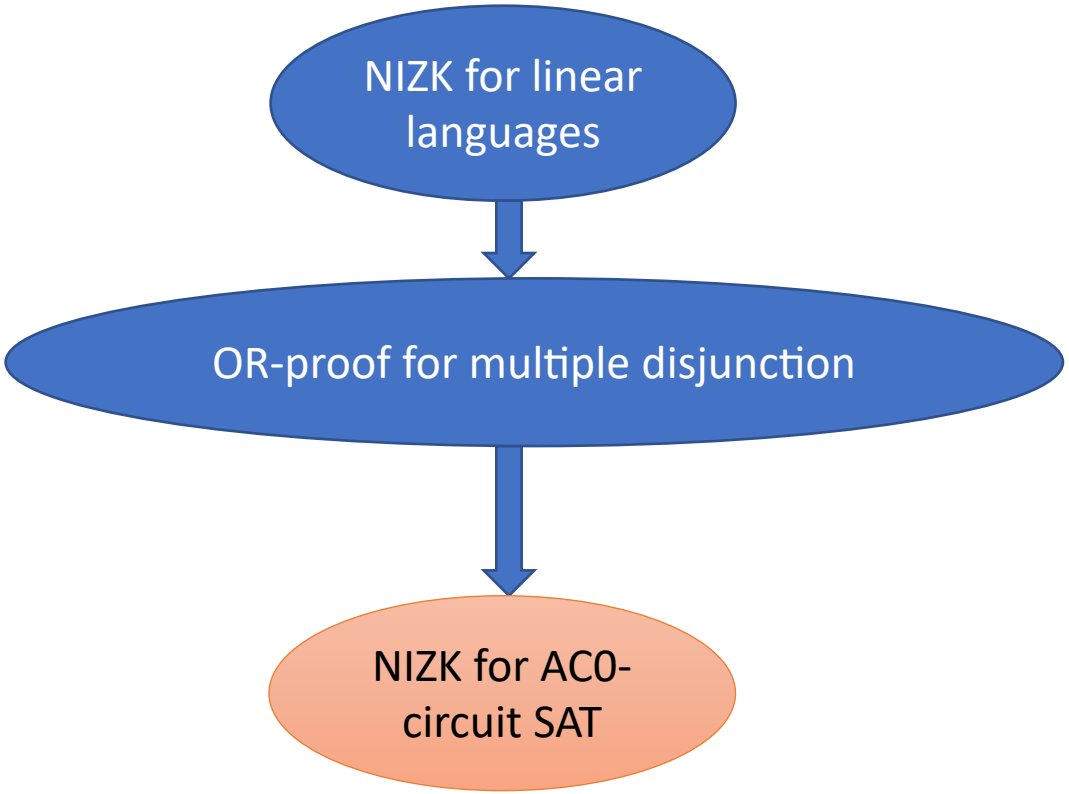
With the proof, a verifier can check the validity of split CRSs in AC0.

First layer:
Sample n trapdoors $(t_1, \cdots t_n)$ with the sum being a

Second layer:
1. Sample n hiding CRSs as $(r_1, \cdots, r_n) = (Et_1, \cdots, E$
2. Add $r_j$ with r.

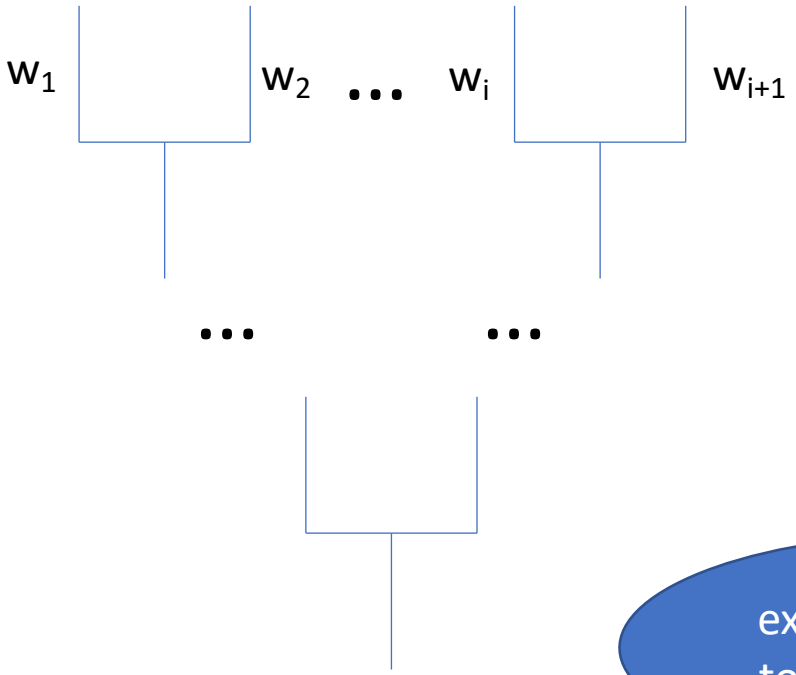
Additional step for verification:
Use the NIZK for linear languages to prove that $r_1 + \cdots + r_n = r$
- an AC0 extractor extracting the witness from $(t_1, \cdots t_n)$

# NIZK for AC0-circuit SAT



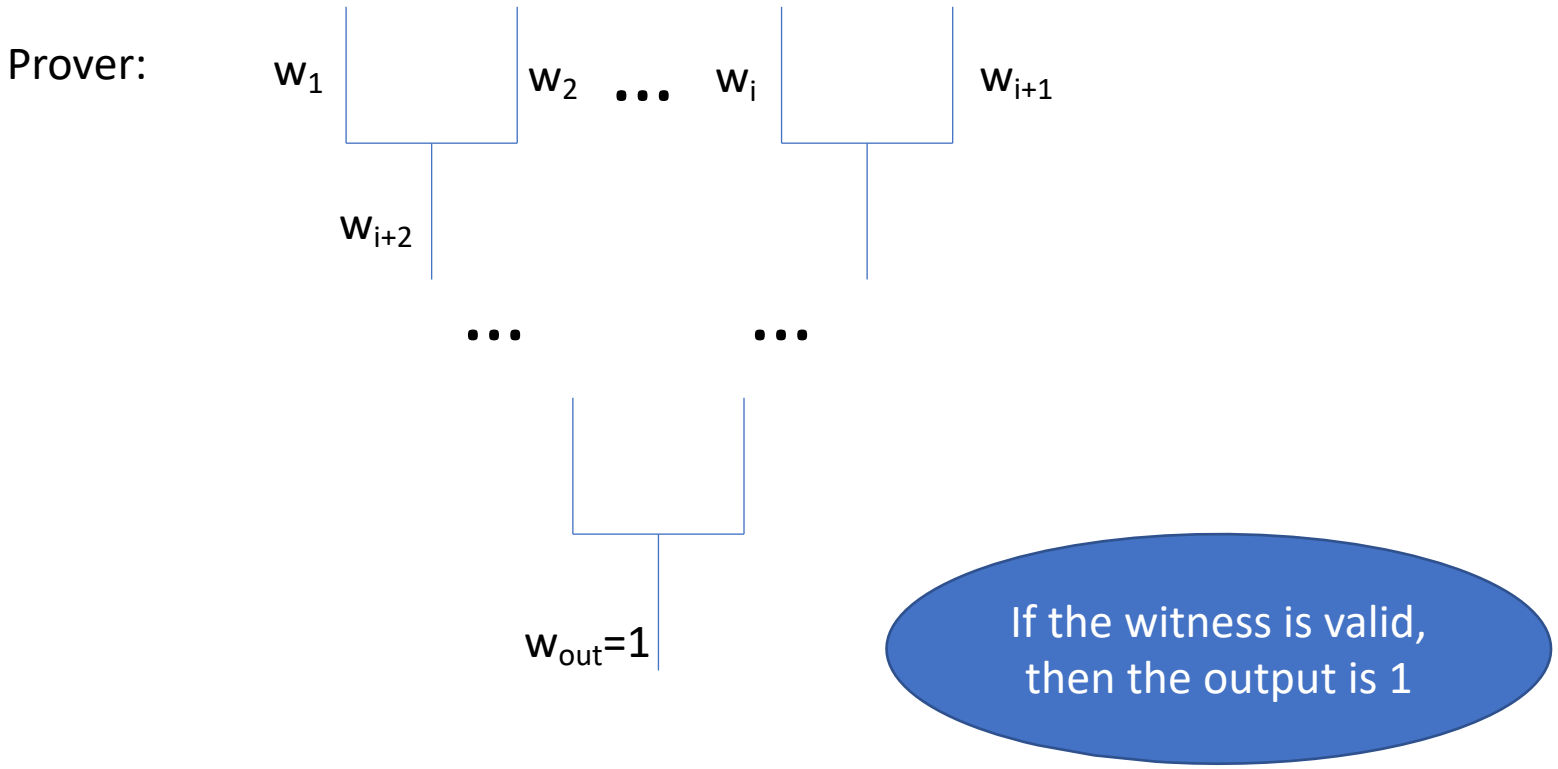
# NIZK for AC0-circuit SAT

Prover:



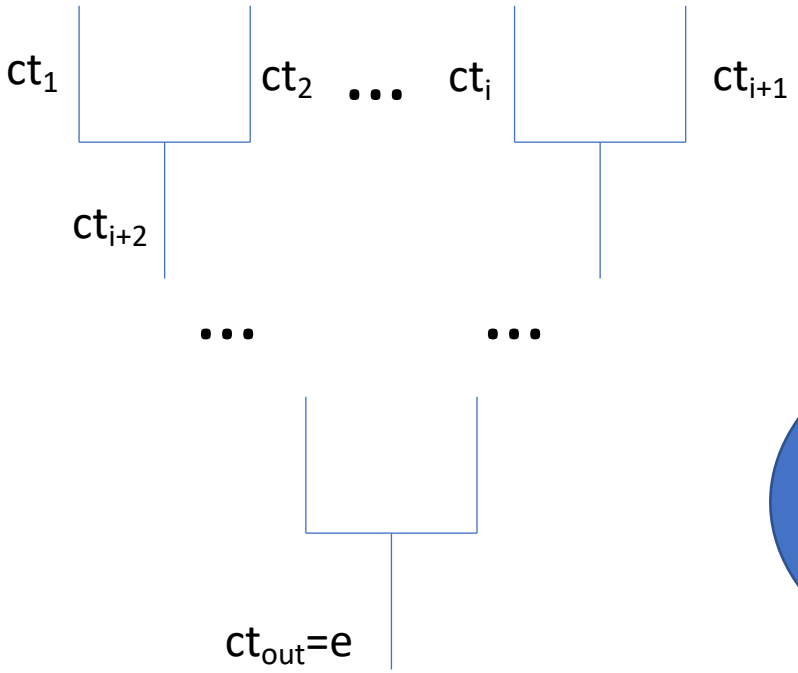

The prover first extends the witness to contain bits of all wires

# NIZK for AC0-circuit SAT



# NIZK for AC0-circuit SAT

Prover:

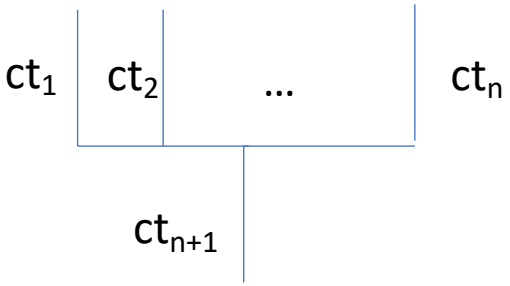



$$ct_i = Er_i + ew_i$$

The prover encodes $w_i$ as a random vector $ct_i$ with the parity being $w_i$

# NIZK for AC0-circuit SAT

Prover:

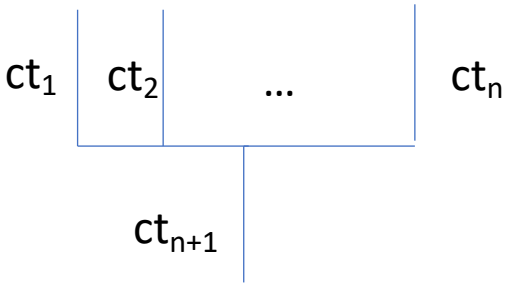



Unbounded fan-in AND gate

For AND-gate, use the OR-proof for multiple disjunction to prove that either $ct_i$-e$\in \mathrm{Span}(\mathrm{E})$ for all i or $ct_j$ ,$ct_{n+1} \in \mathrm{Span}(\mathrm{E})$ for some j.

# NIZK for AC0-circuit SAT

Prover:



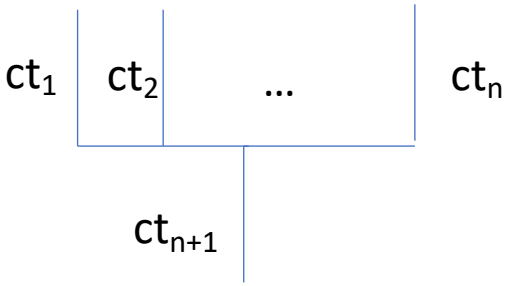

Unbounded fan-in AND gate

For AND-gate, use the OR-proof for multiple disjunction to prove that either $ct_i$-e$\in$ $\mathrm{Span}(\mathrm{E})$ for all i or $ct_j$ ,$ct_{n+1} \in \mathrm{Span}(\mathrm{E})$ for some j.

Enough to ensure consistence of encoded wires

# NIZK for AC0-circuit SAT

Prover:



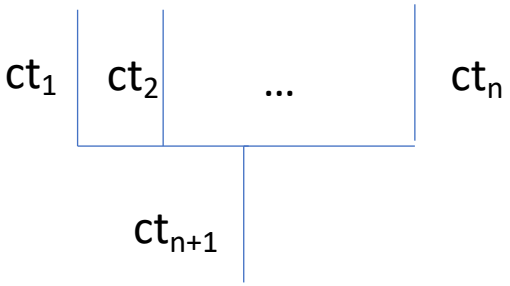

Unbounded fan-in AND gate

For AND-gate, use the OR-proof for multiple disjunction to prove that either $ct_i$-e$\in \mathrm{Span}(\mathrm{E})$ for all i or $ct_j$ ,$ct_{n+1} \in \mathrm{Span}(\mathrm{E})$ for some j.

A challenge: to get the witness, the prover has to decide which wire corresponds to 0 when the output is 0. The prover cannot find such wire by searching each wire in order.

# NIZK for AC0-circuit SAT

Prover:



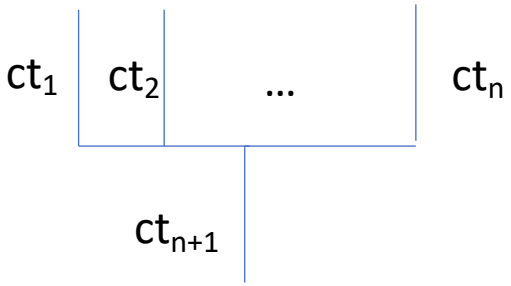

Unbounded fan-in AND gate

For AND-gate, use the OR-proof for multiple disjunction to prove that either $ct_i$-e$\in \mathrm{Span}(\mathrm{E})$ for all i or $ct_j$ ,$ct_{n+1} \in \mathrm{Span}(\mathrm{E})$ for some j.

We prove the existence of an AC0-algorithm that can output the index of the first 0-bit or 1-bit amongst n bits.

# NIZK for AC0-circuit SAT

Prover:




Unbounded fan-in AND gate

For AND-gate, use the OR-proof for multiple disjunction to prove that either $ct_i - e \in \mathrm{Span}(E)$ for all i or $ct_j, ct_{n+1} \in \mathrm{Span}(E)$ for some j.

For OR-gates, we prove the consistence in a similar way.
For NOT-gates, we prove that the sum of e and the input and output encodings are in Span(E).

# Extensions

- Conversion to non-interactive zaps (NIWI in the plain model)

# Extensions

- Conversion to non-interactive zaps (NIWI in the plain model)

# Extensions

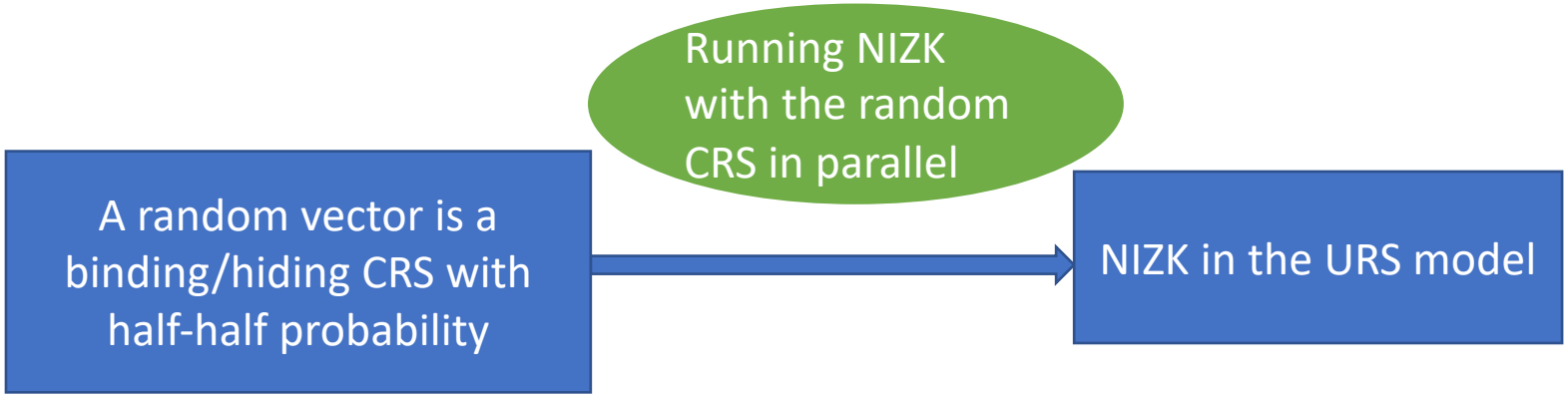
- Conversion to non-interactive zaps (NIWI in the plain model)


- Conversion to NIZKs in the URS model

# Extensions

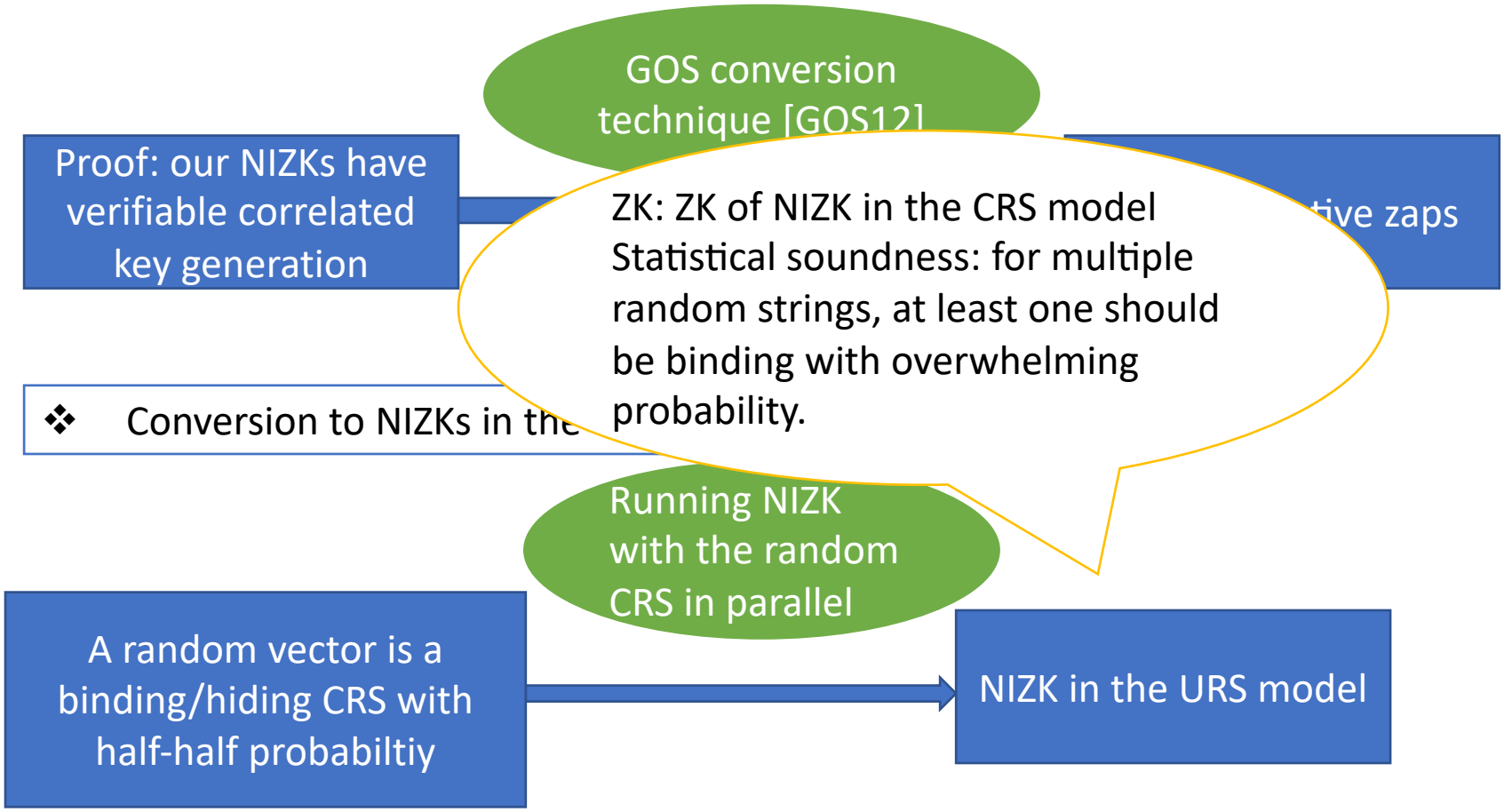
- Conversion to non-interactive zaps (NIWI in the plain model)


- Conversion to NIZKs in the URS model



# Extensions

- Conversion to non-interactive zaps (NIWI in the plain model)




- Conversion to NIZKs in the

# Conclusion

Fine-grained proof systems unconditionally secure against AC0 adversaries

1. NIZK for AC0-circuit SAT
2. Non-interactive zaps
3. NIZKs in the URS model