A SHUFFLE ARGUMENT SECURE IN THE GENERIC MODEL

Prastudy Fauzi, Helger Lipmaa, Michal Zajac

University of Tartu, Estonia

ASIACRYPT 2016

Panoramix

OUR RESULTS

- A new efficient CRS-based NIZK shuffle argument

OUR RESULTS

- A new efficient CRS-based NIZK shuffle argument
- **Four+ times** more efficient **verification** than in prior work

OUR RESULTS

- A new efficient CRS-based NIZK shuffle argument
- **Four+ times** more efficient **verification** than in prior work
  - Verification time more critical

OUR RESULTS

- A new efficient CRS-based NIZK shuffle argument
- **Four+ times** more efficient **verification** than in prior work
  - Verification time more critical
- Soundness proof in the **Generic Bilinear Group Model**

OUR RESULTS

- A new efficient CRS-based NIZK shuffle argument
- **Four+ times** more efficient **verification** than in prior work
  - Verification time more critical
- Soundness proof in the **Generic Bilinear Group Model**
  - Very complicated **machine-assisted** proof

OUR RESULTS

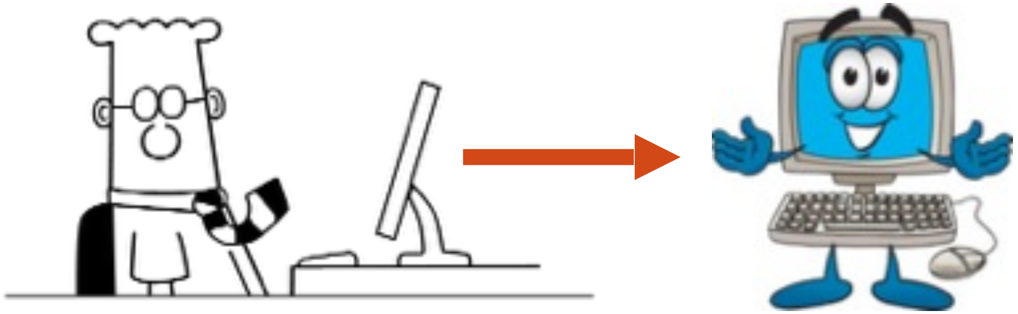
- A new efficient CRS-based NIZK shuffle argument
- **Four+ times** more efficient **verification** than in prior work
  - Verification time more critical
- Soundness proof in the **Generic Bilinear Group Model**
  - Very complicated **machine-assisted** proof
  - Use computer algebra to solve systems of polyn. eq.

OUR RESULTS

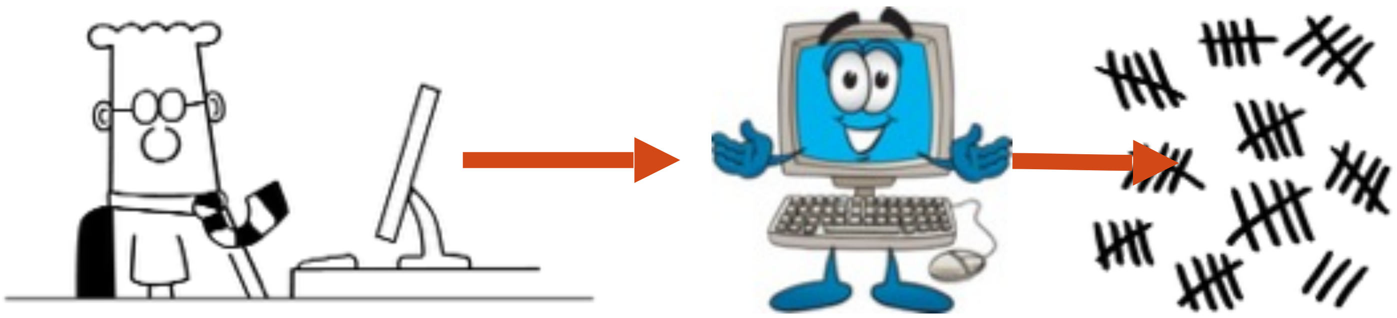
- A new efficient CRS-based NIZK shuffle argument
- **Four+ times** more efficient **verification** than in prior work
  - Verification time more critical
- Soundness proof in the **Generic Bilinear Group Model**
  - Very complicated **machine-assisted** proof
  - Use computer algebra to solve systems of polyn. eq.
    - Esp. to find **Gröbner bases**

A BIT OF MOTIVATION: E-VOTING

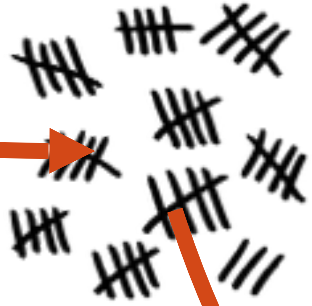
A BIT OF MOTIVATION: E-VOTING



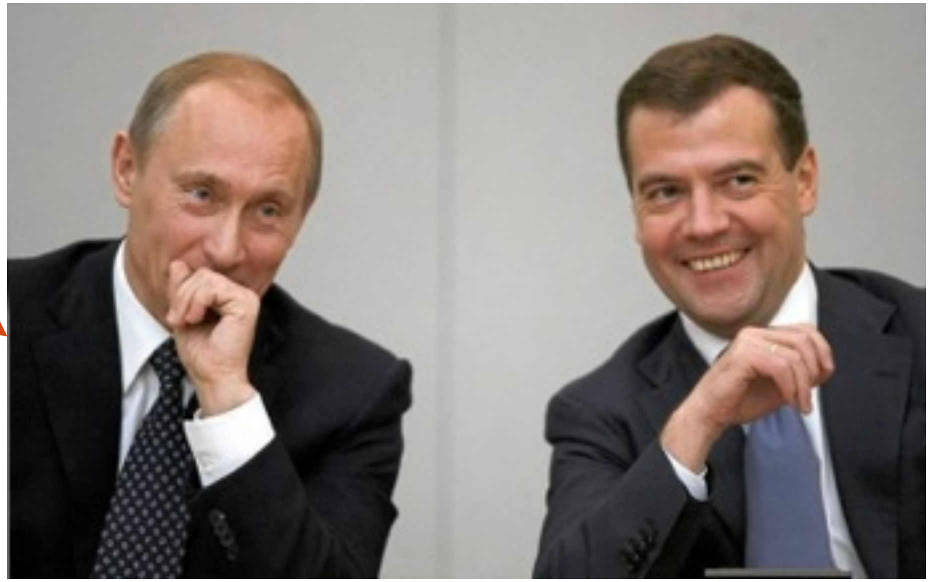
A BIT OF MOTIVATION: E-VOTING



A BIT OF MOTIVATION: E-VOTING

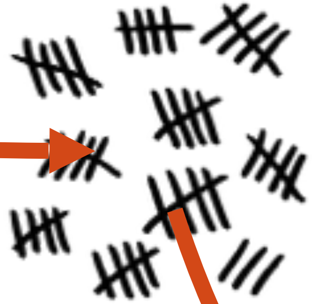


Lesson from the past:
It is not voters who counts,
but who counts the votes



- Can we get away with that?
- I'm 140% sure!

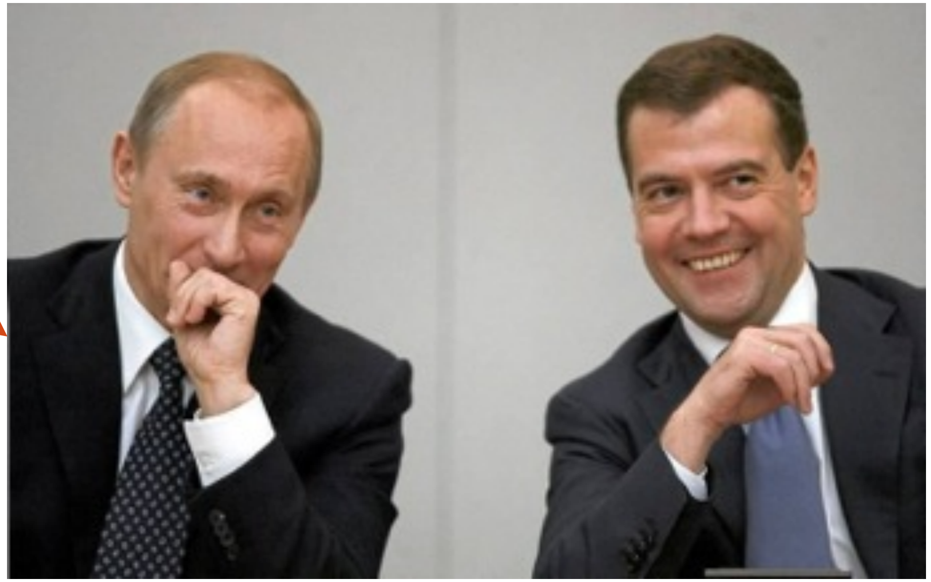
A BIT OF MOTIVATION: E-VOTING



Anonymity

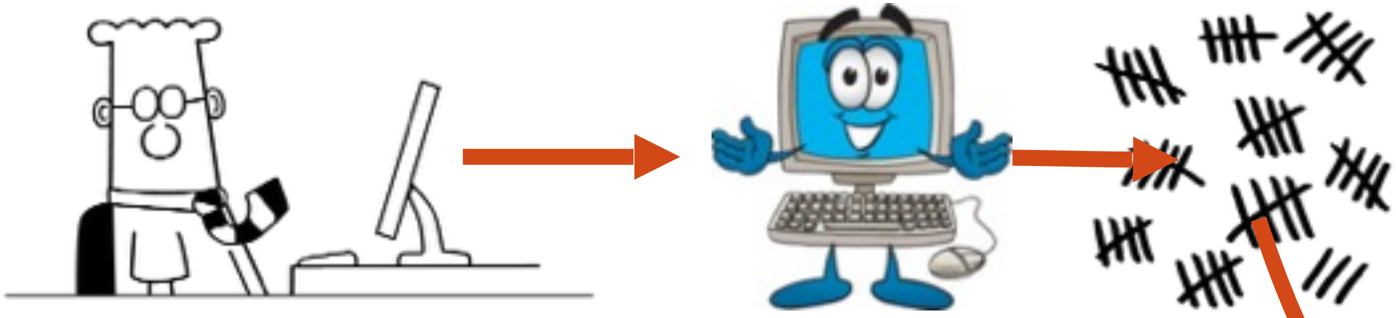
Correctness

Lesson from the past:
It is not voters who counts,
but who counts the votes

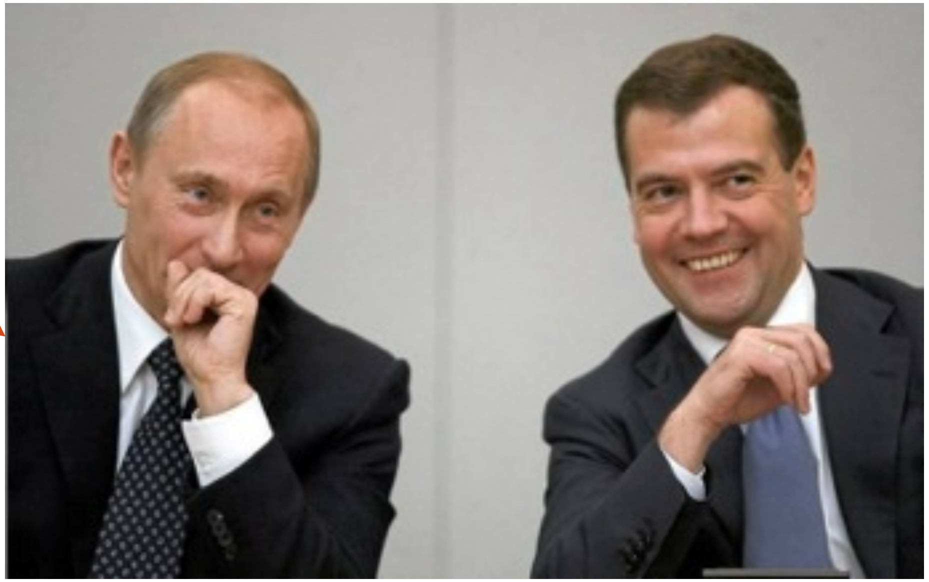


- Can we get away with that?
- I'm 140% sure!

A BIT OF MOTIVATION: E-VOTING



Lesson from the past:
It is not voters who counts,
but who counts the votes



- Can we get away with that?
- I’m 140% sure!

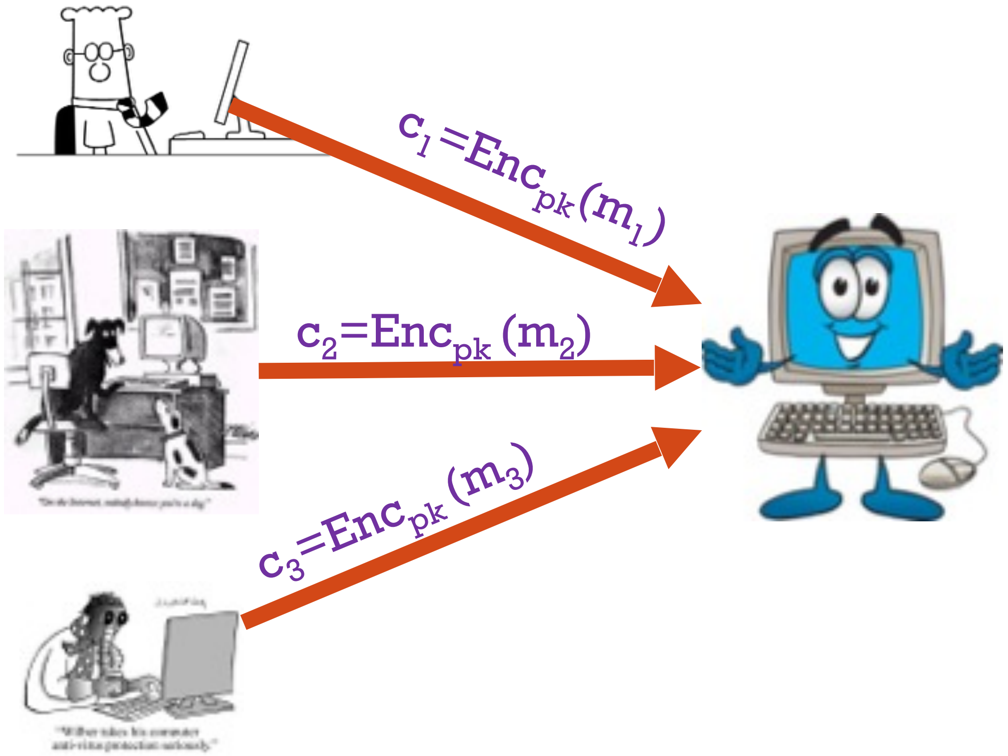
Anonymity

Correctness

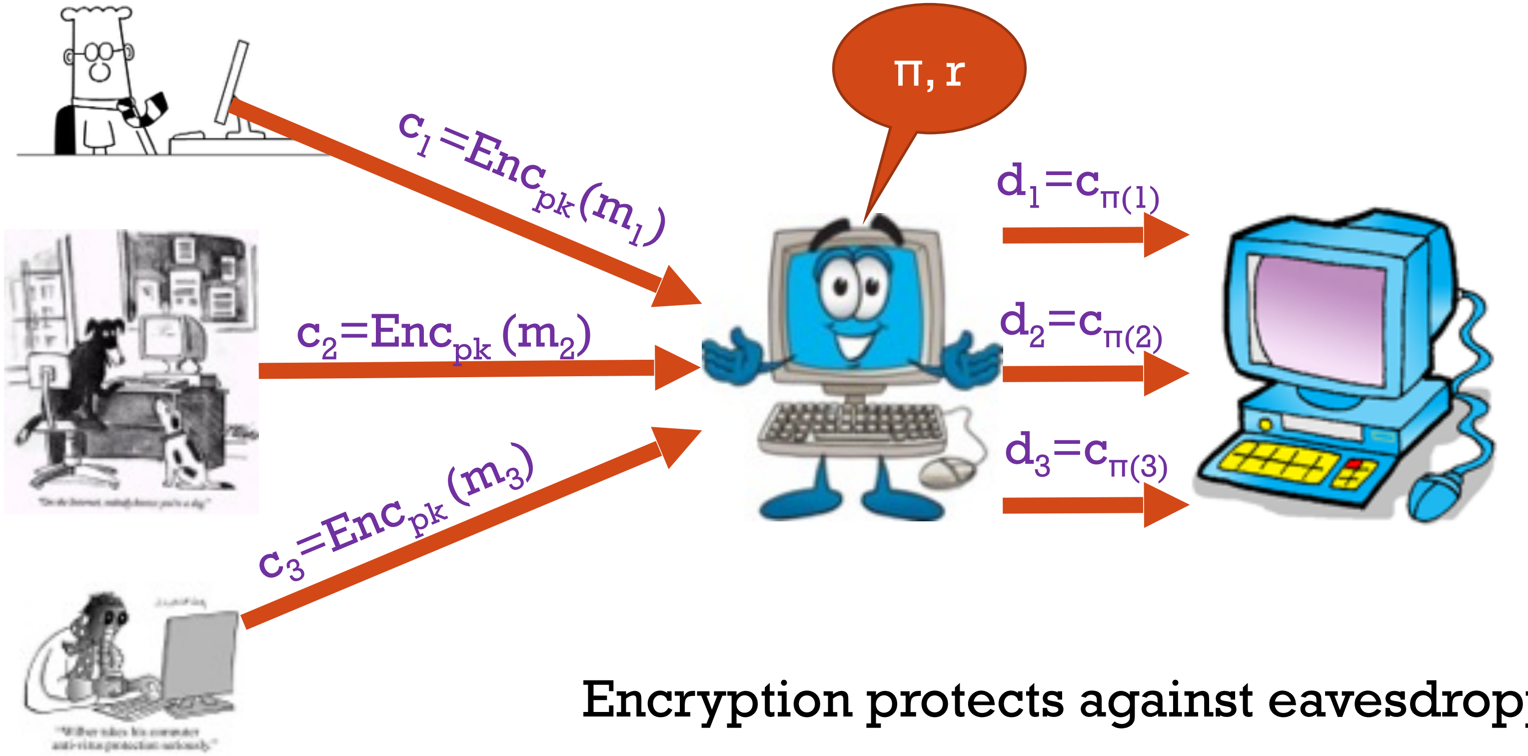
Data is public
(Data, source) is private

SIMPLE MIX-NETS

SIMPLE MIX-NETS

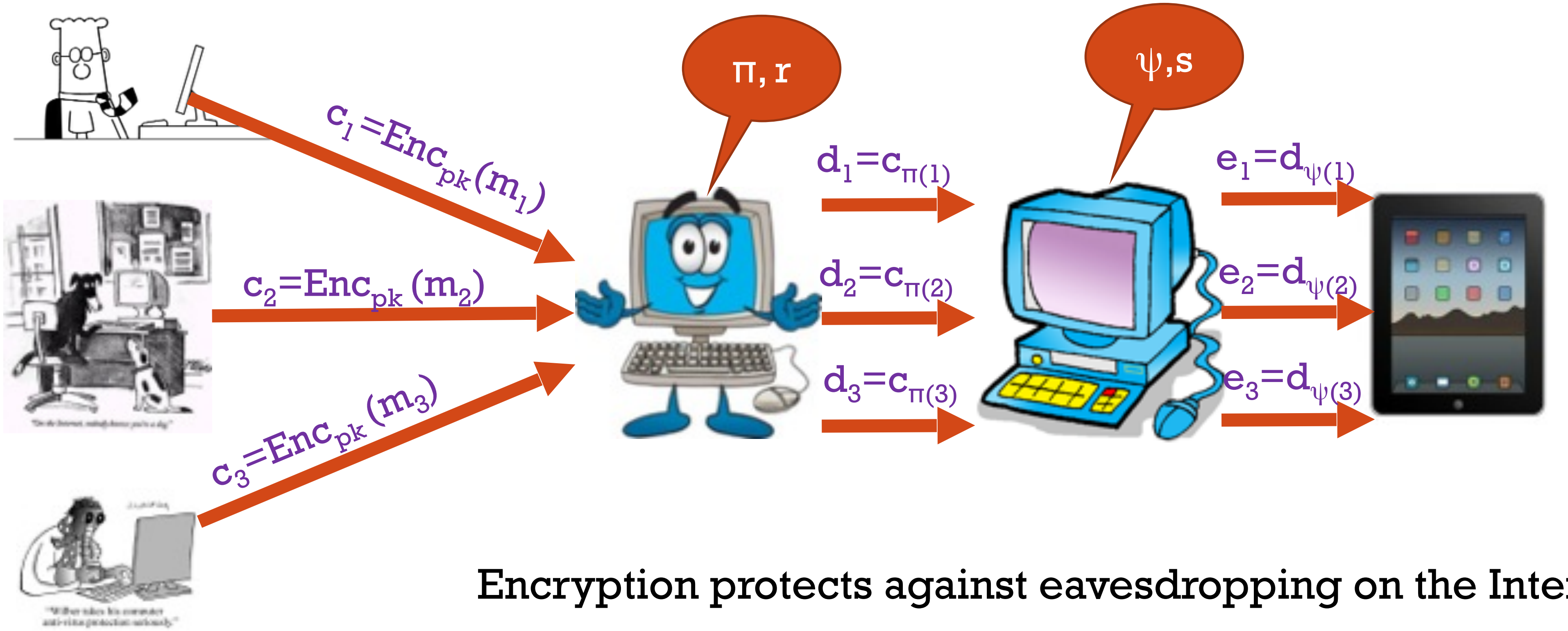


SIMPLE MIX-NETS



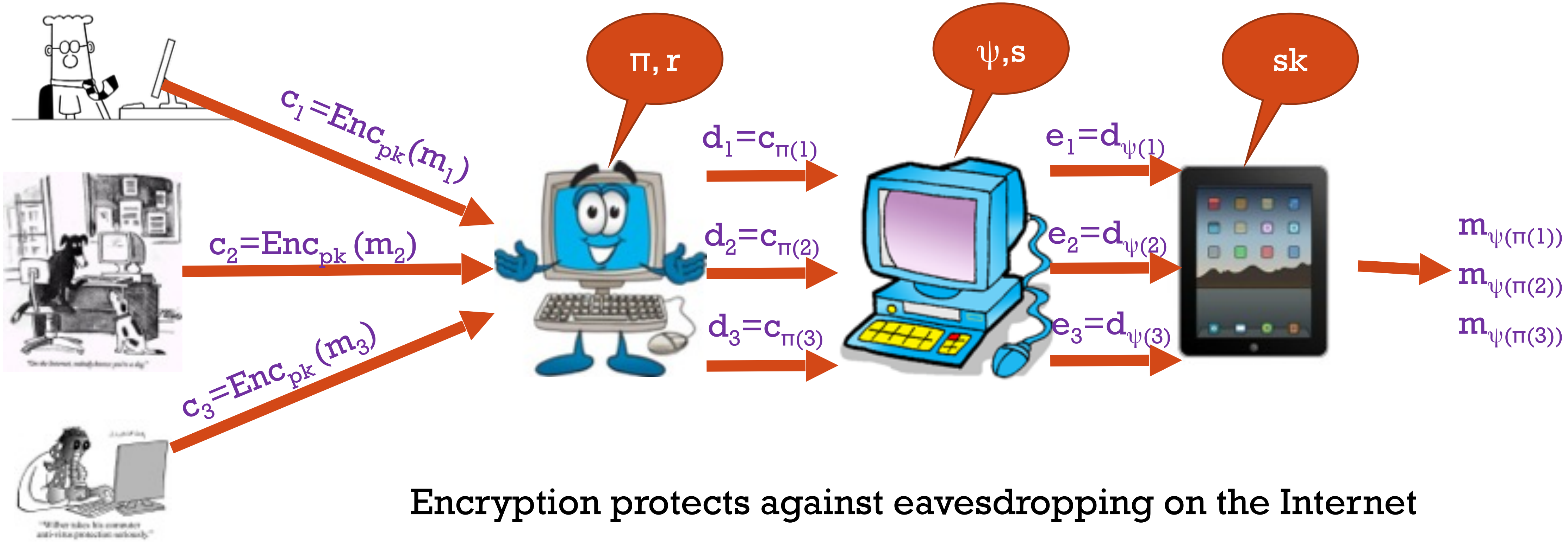
Encryption protects against eavesdropping on the Internet

SIMPLE MIX-NETS



Encryption protects against eavesdropping on the Internet

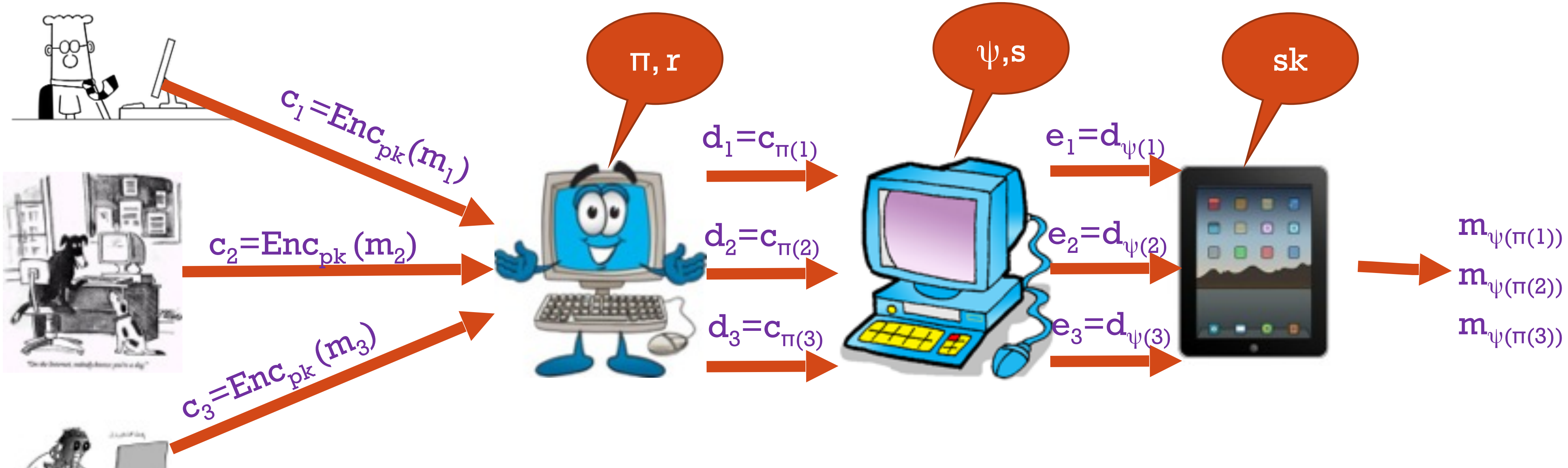
SIMPLE MIX-NETS



Encryption protects against eavesdropping on the Internet

SIMPLE MIX-NETS

Anonymity ✓

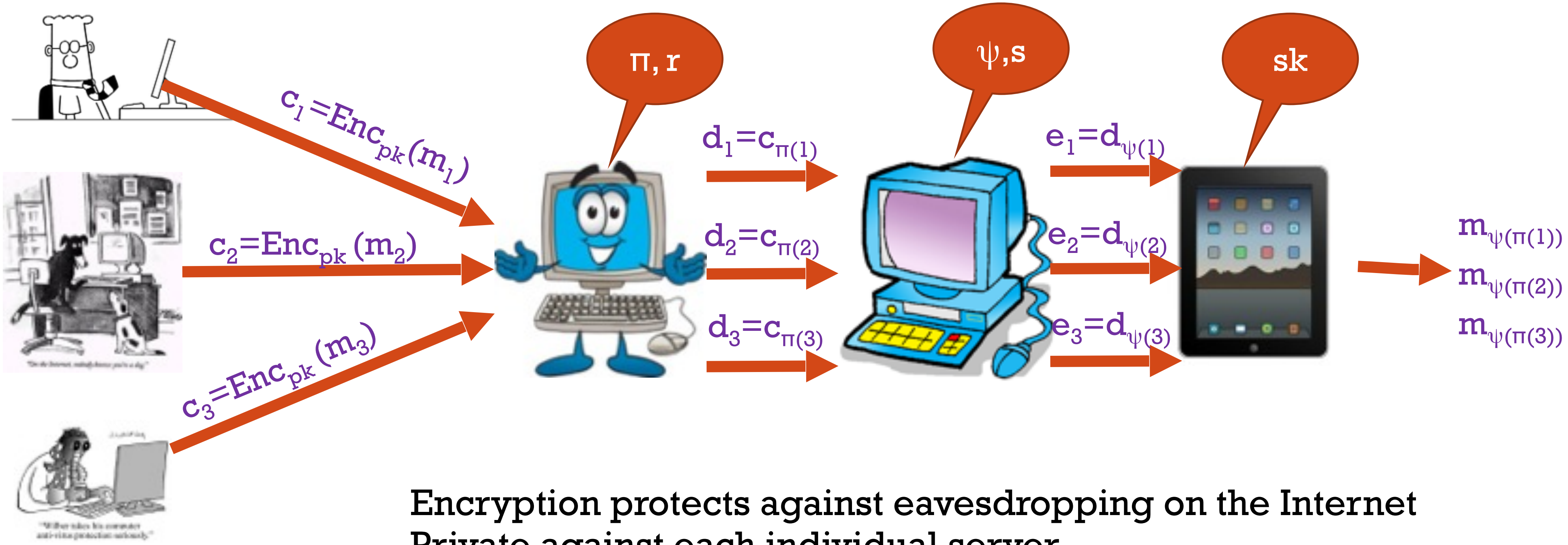


Encryption protects against eavesdropping on the Internet
Private against each individual server

SIMPLE MIX-NETS

Anonymity ✓

Correctness ✗

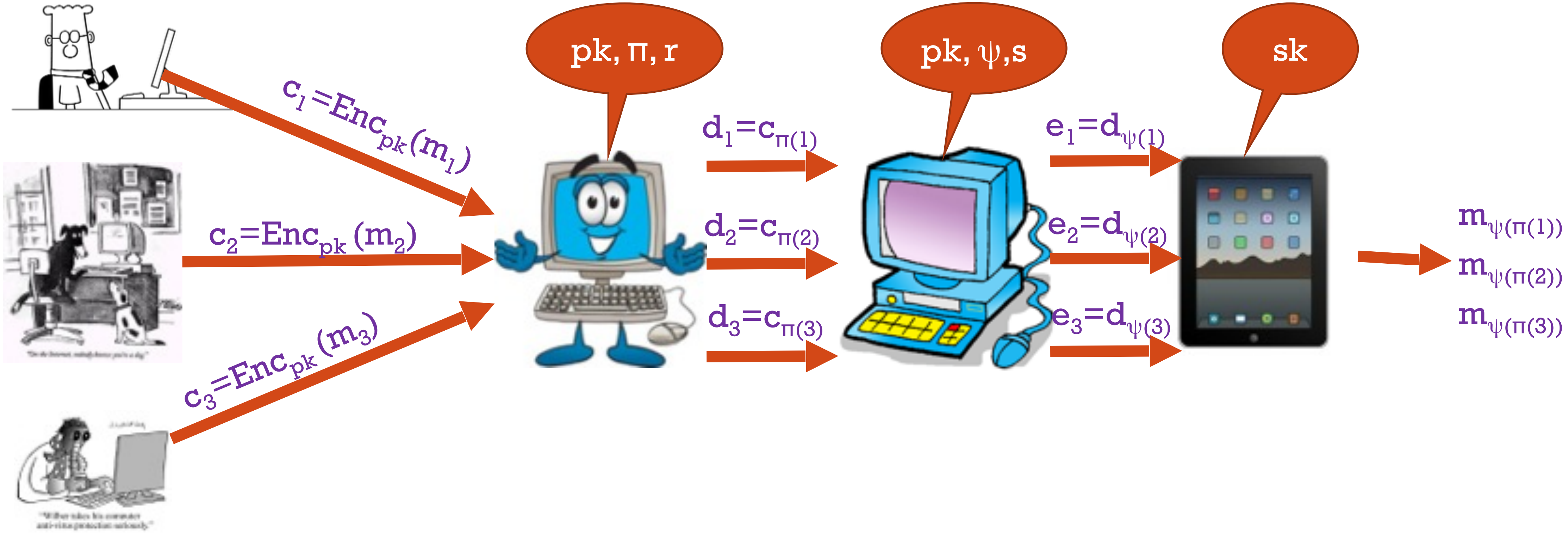


Encryption protects against eavesdropping on the Internet

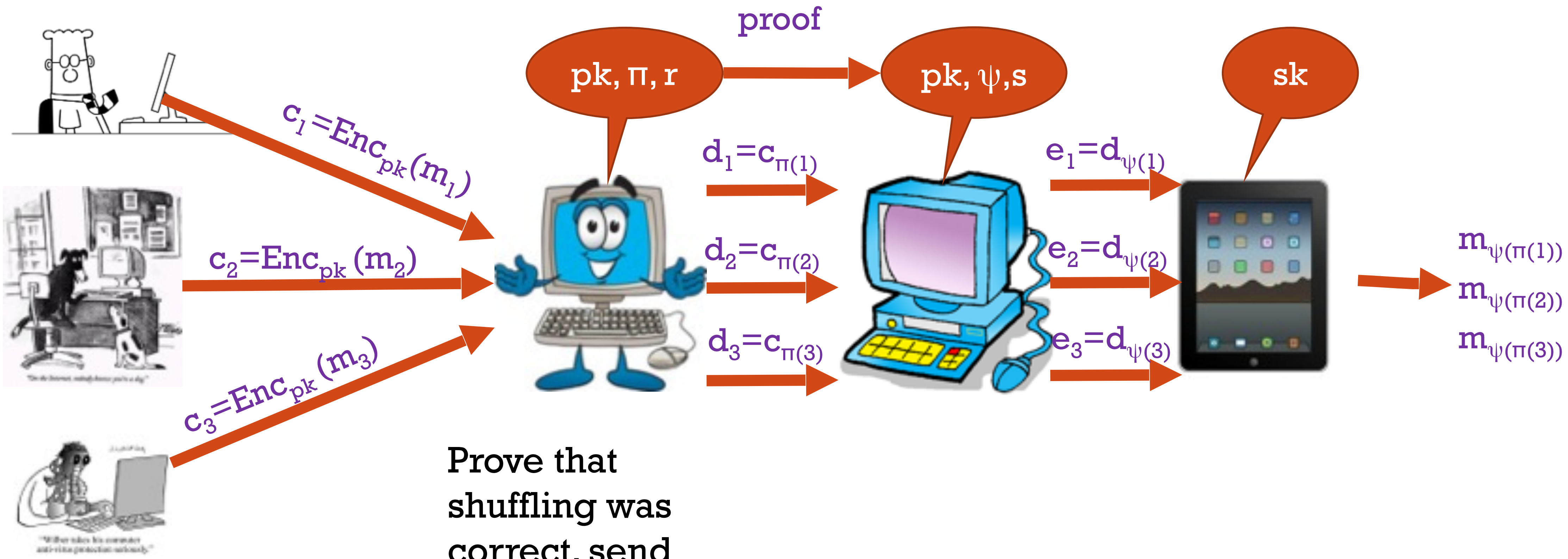
Private against each individual server

Not enough: what if a server cheats?

ACCOUNTABLE MIX-NETS

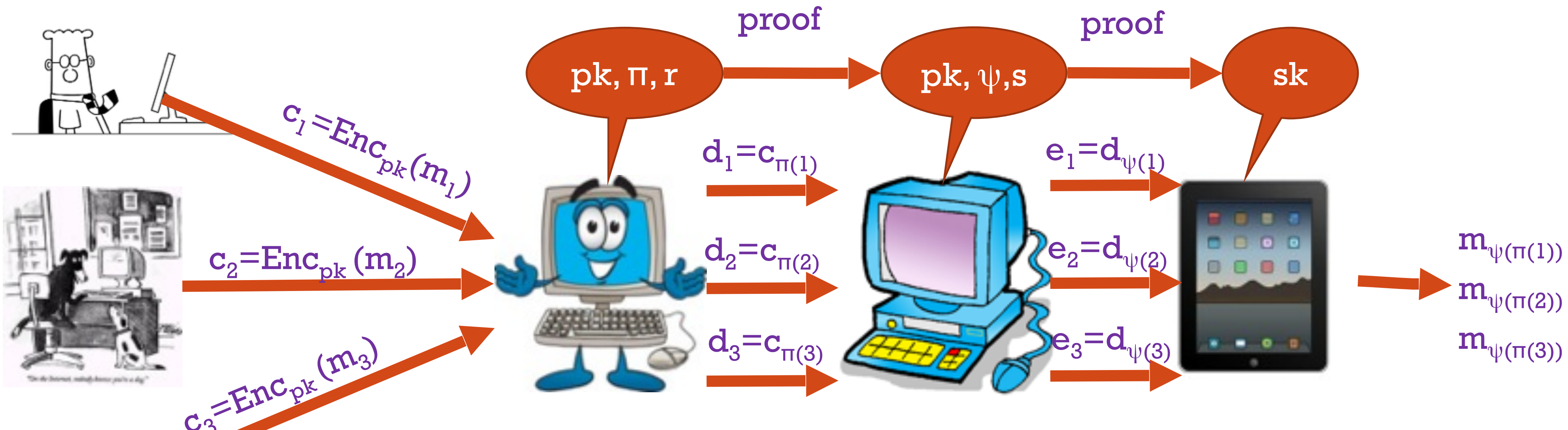


ACCOUNTABLE MIX-NETS



Prove that shuffling was correct, send proof to the next server

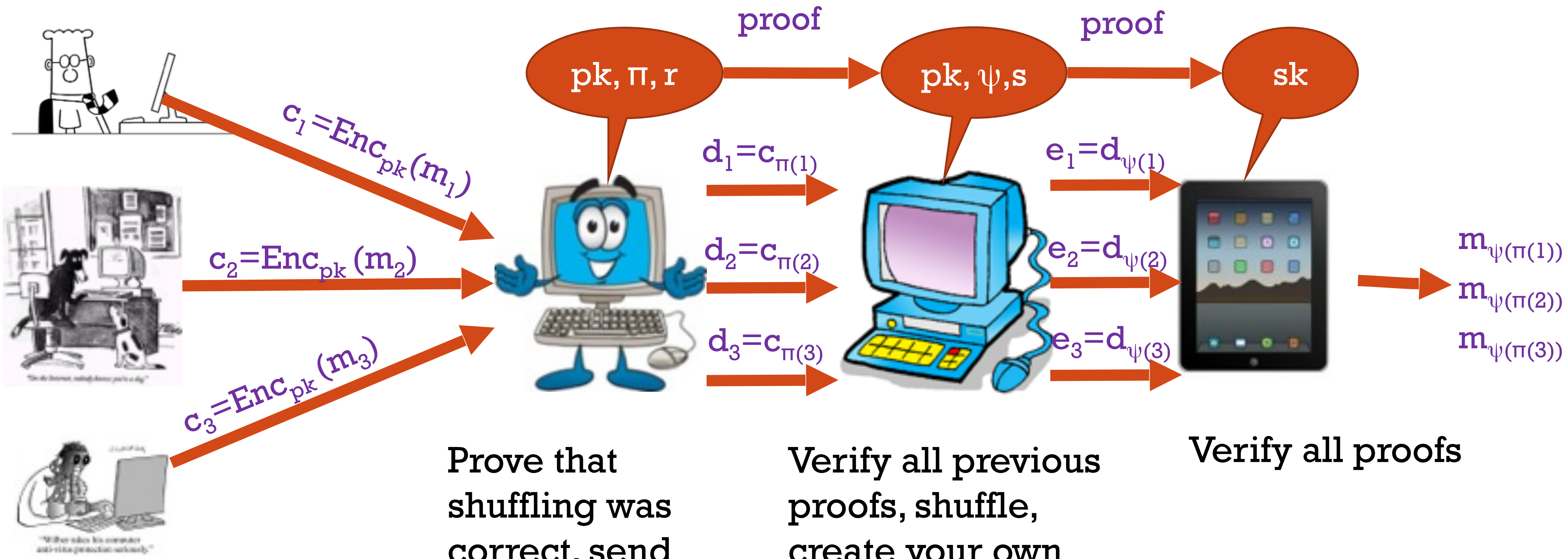
ACCOUNTABLE MIX-NETS



Prove that shuffling was correct, send proof to the next server

Verify all previous proofs, shuffle, create your own proof

ACCOUNTABLE MIX-NETS



Prove that shuffling was correct, send proof to the next server

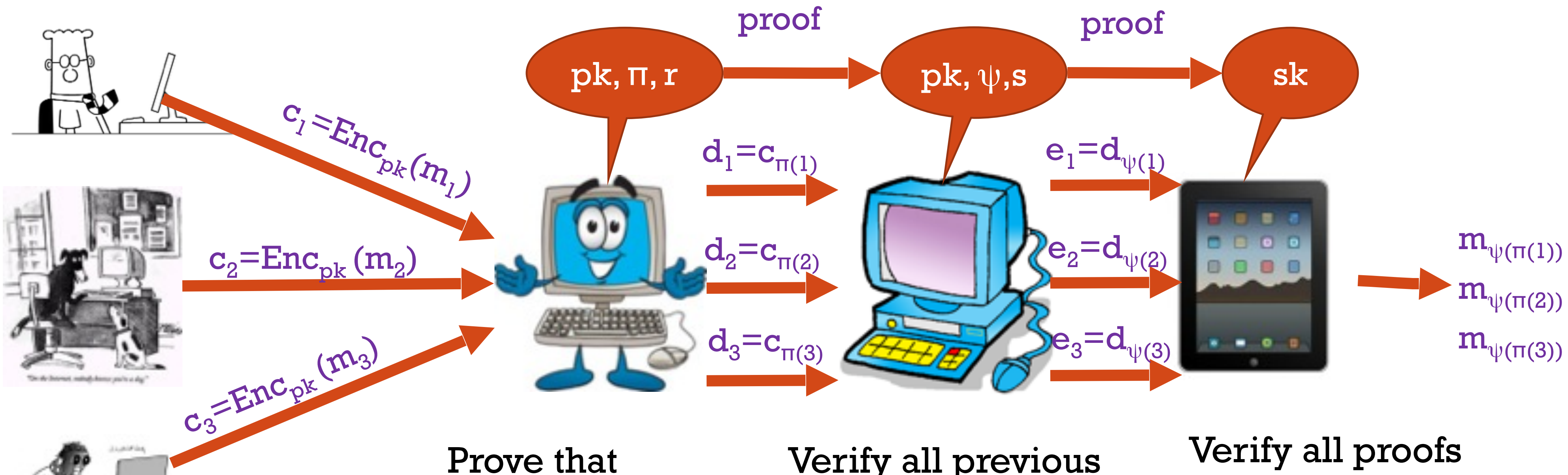
Verify all previous proofs, shuffle, create your own proof

Verify all proofs

ACCOUNTABLE MIX-NETS

Anonymity ✓

Correctness ✓



Prove that shuffling was correct, send proof to the next server

Verify all previous proofs, shuffle, create your own proof

Verify all proofs

SHUFFLE ARGUMENT

- **Shuffle argument:**
  - efficient zero knowledge argument of correctness of shuffling
    Mix-server permutes ciphertexts, re-encrypt them and provides a proof that he has done it correctly.

SHUFFLE ARGUMENT

- **Shuffle argument:**
  - efficient zero knowledge argument of correctness of shuffling
    Mix-server permutes ciphertexts, re-encrypt them and provides a proof that he has done it correctly.
- Existing CRS model arguments not very efficient

CRS-BASED SHUFFLE ARGUMENTS

Assumption proposed in that paper, proof in GBGM

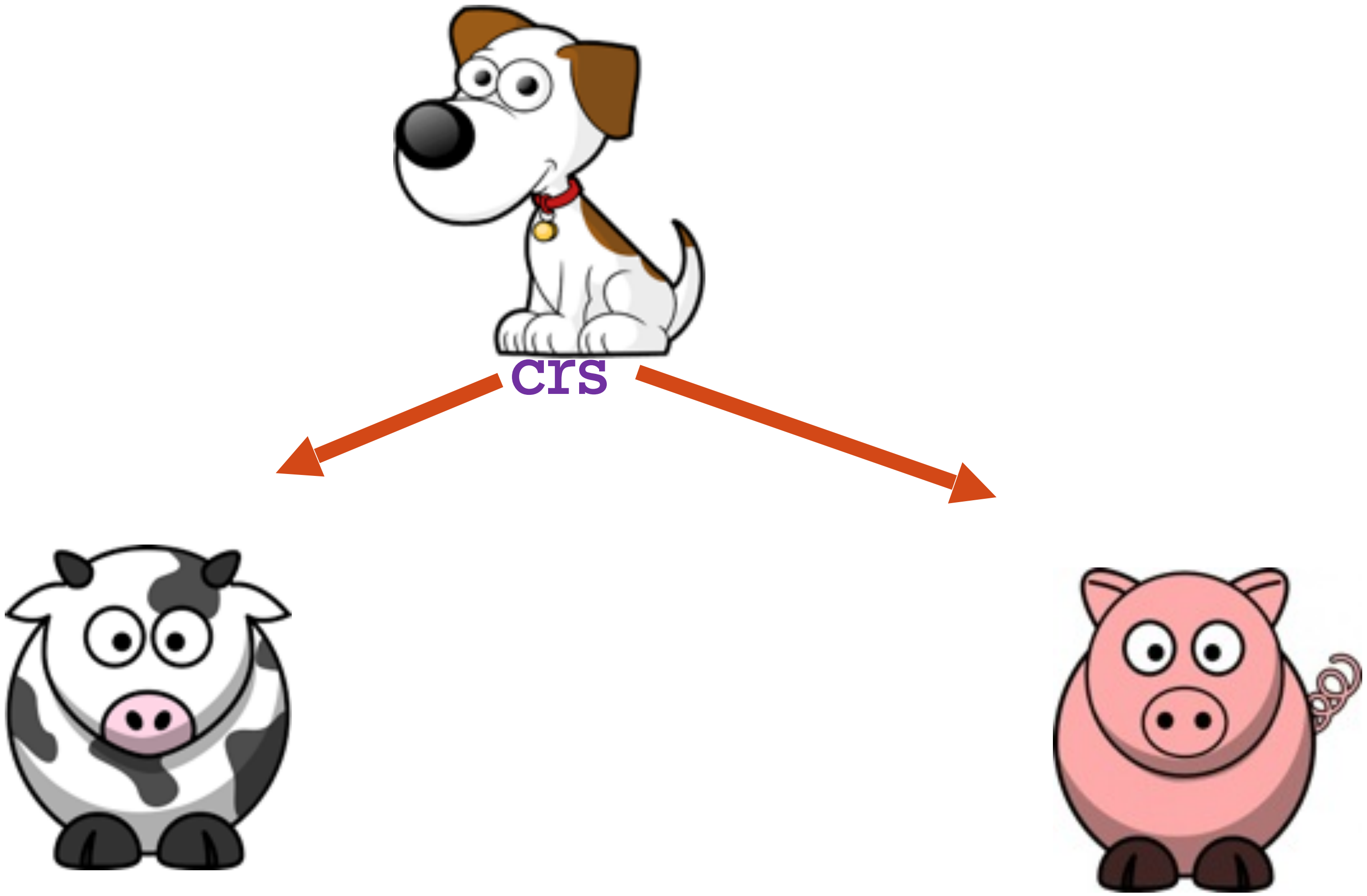
Assmpt. proposed 2010+, but not in that paper, proof in GBGM

	Lipmaa-Zhang (2012)	Fauzi-Lipmaa (2016)	This paper
CRS length	*7n + 6*	*8n + 17*	*3n + 14*
Communic.	*12n + 11*	*9n + 2*	*7n + 3*
P comp. (units)	*36*	*19.8*	*24.3*
V comp. (units)	*196*	*126*	*36.3*
GBGM?	*PSDL, DLIN (comp.)*	*TSDH, PCDH, PSP (comp.)*	*Pure GBGM*
Soundness	*Full*	*Culpable*	*Full*

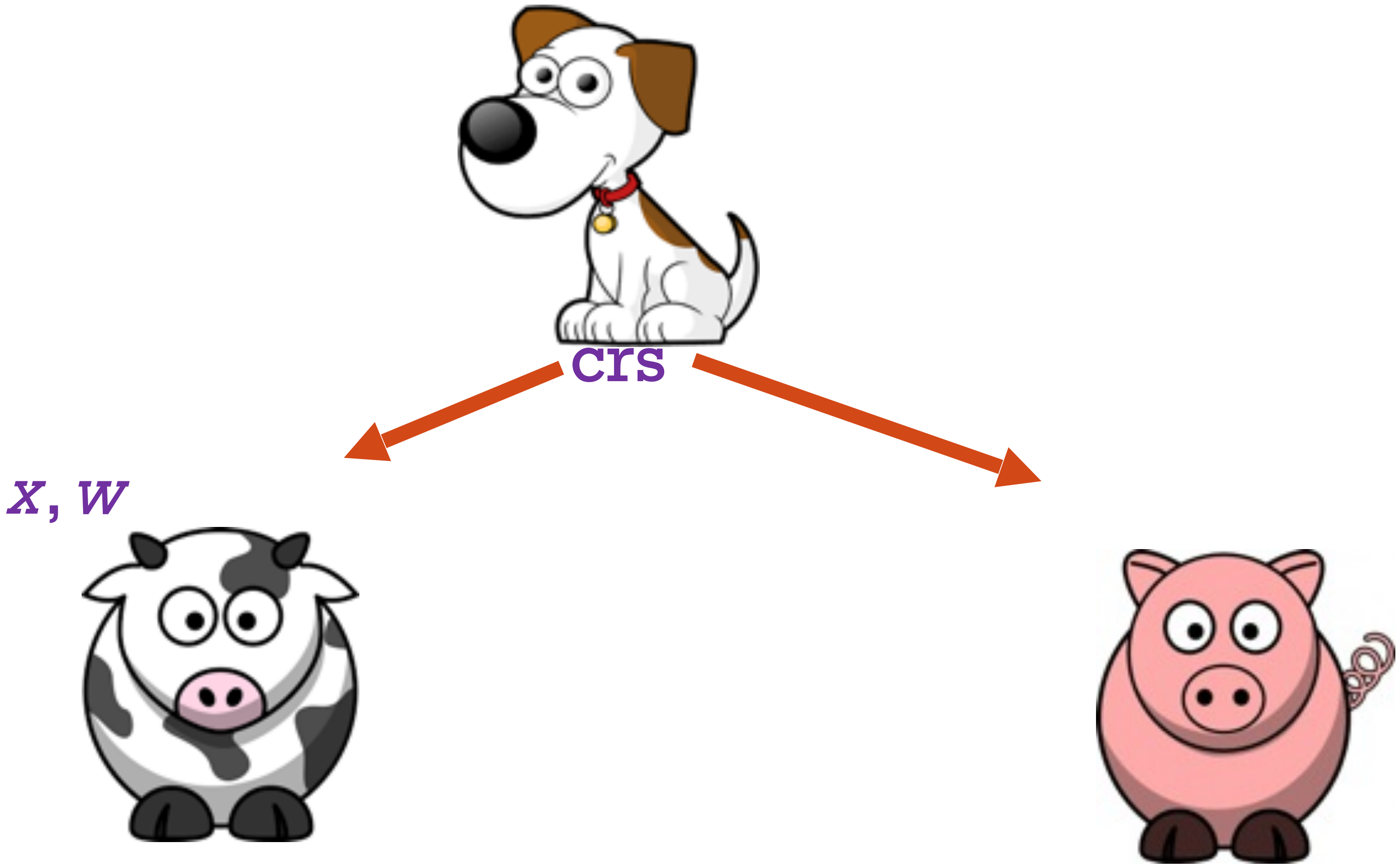
n: number of ciphertexts (say 100,000)

1 unit = *n* million machine cycles
According to speed records on BN curves

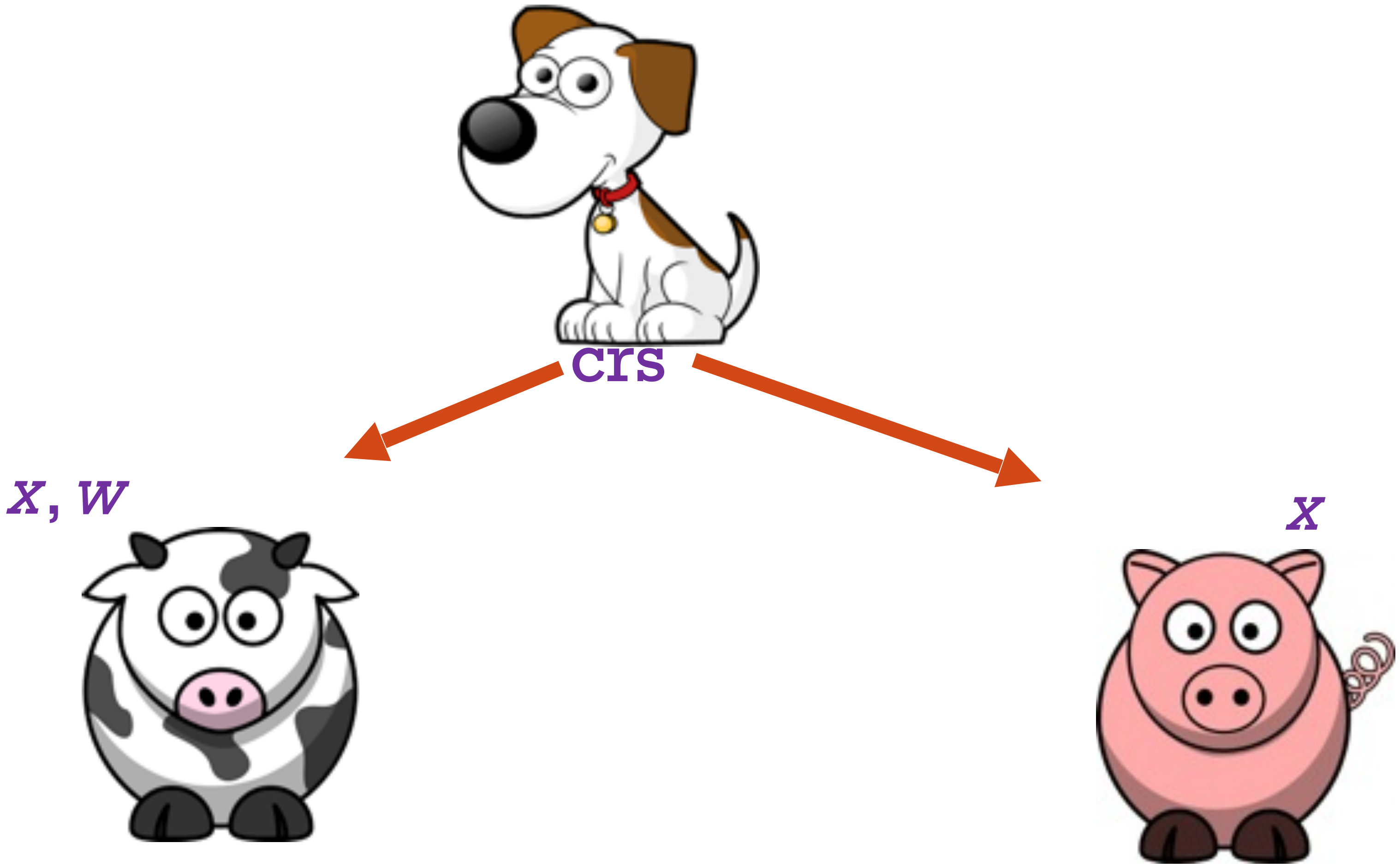
ZERO KNOWLEDGE: CRS MODEL



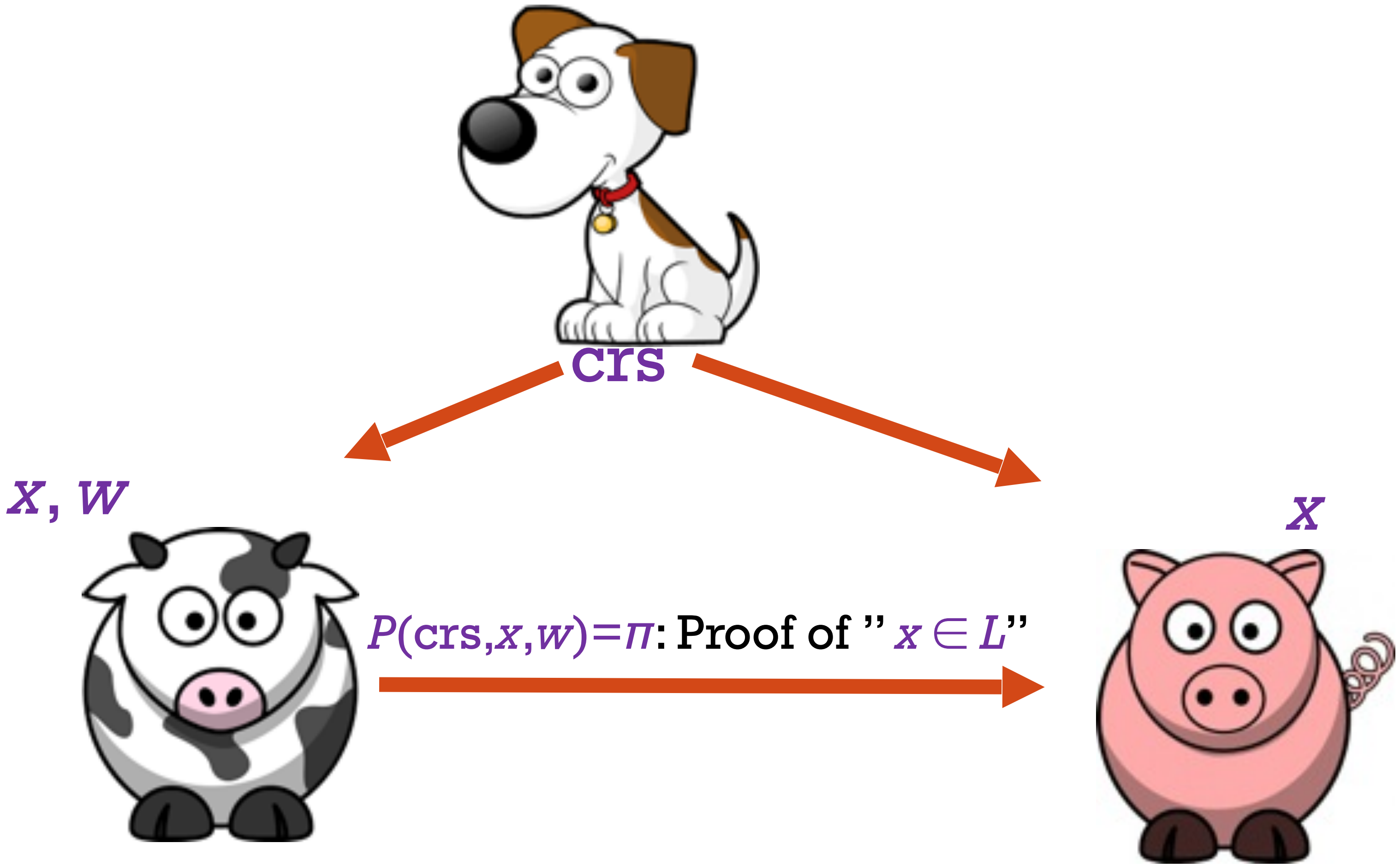
ZERO KNOWLEDGE: CRS MODEL



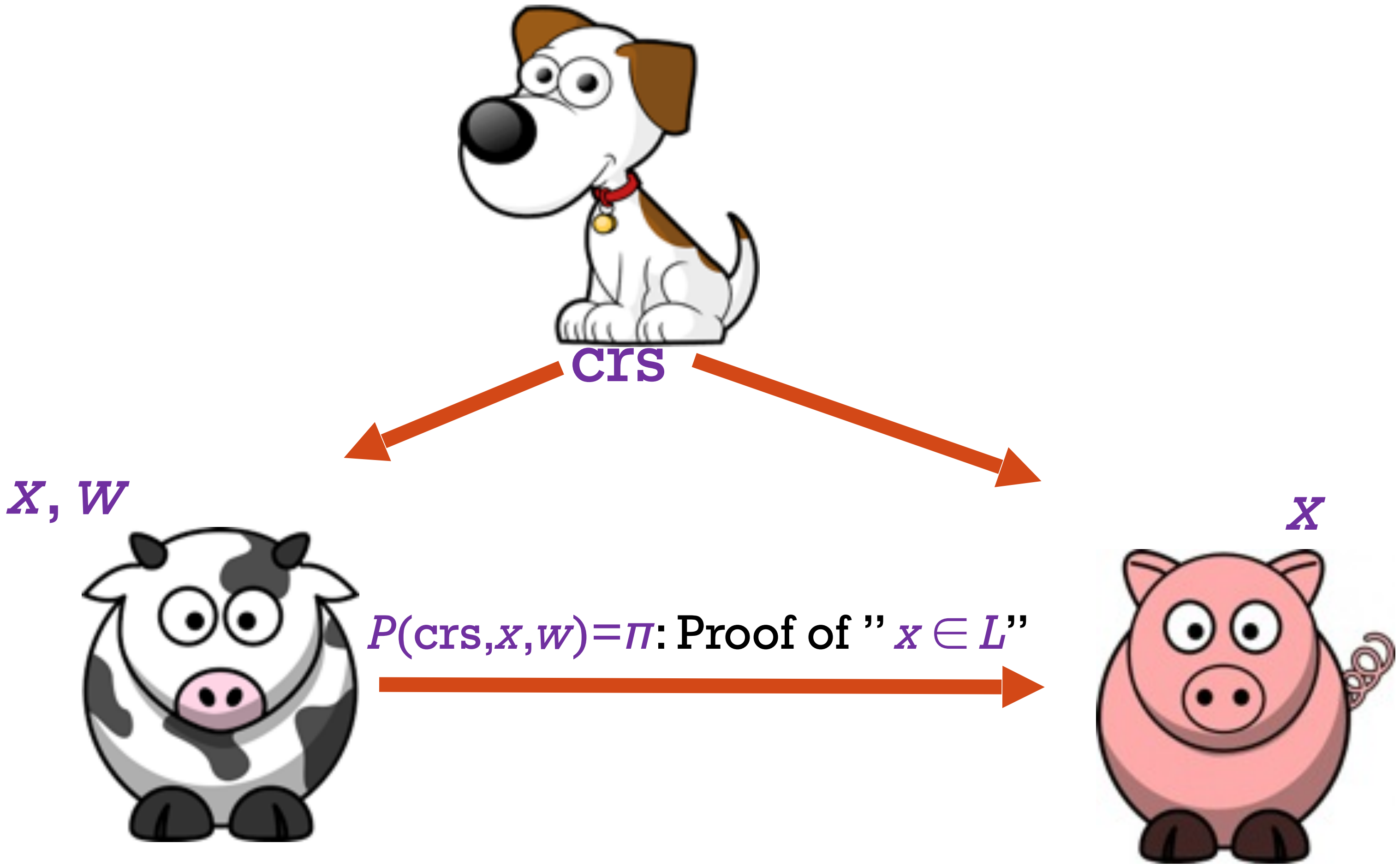
ZERO KNOWLEDGE: CRS MODEL



ZERO KNOWLEDGE: CRS MODEL

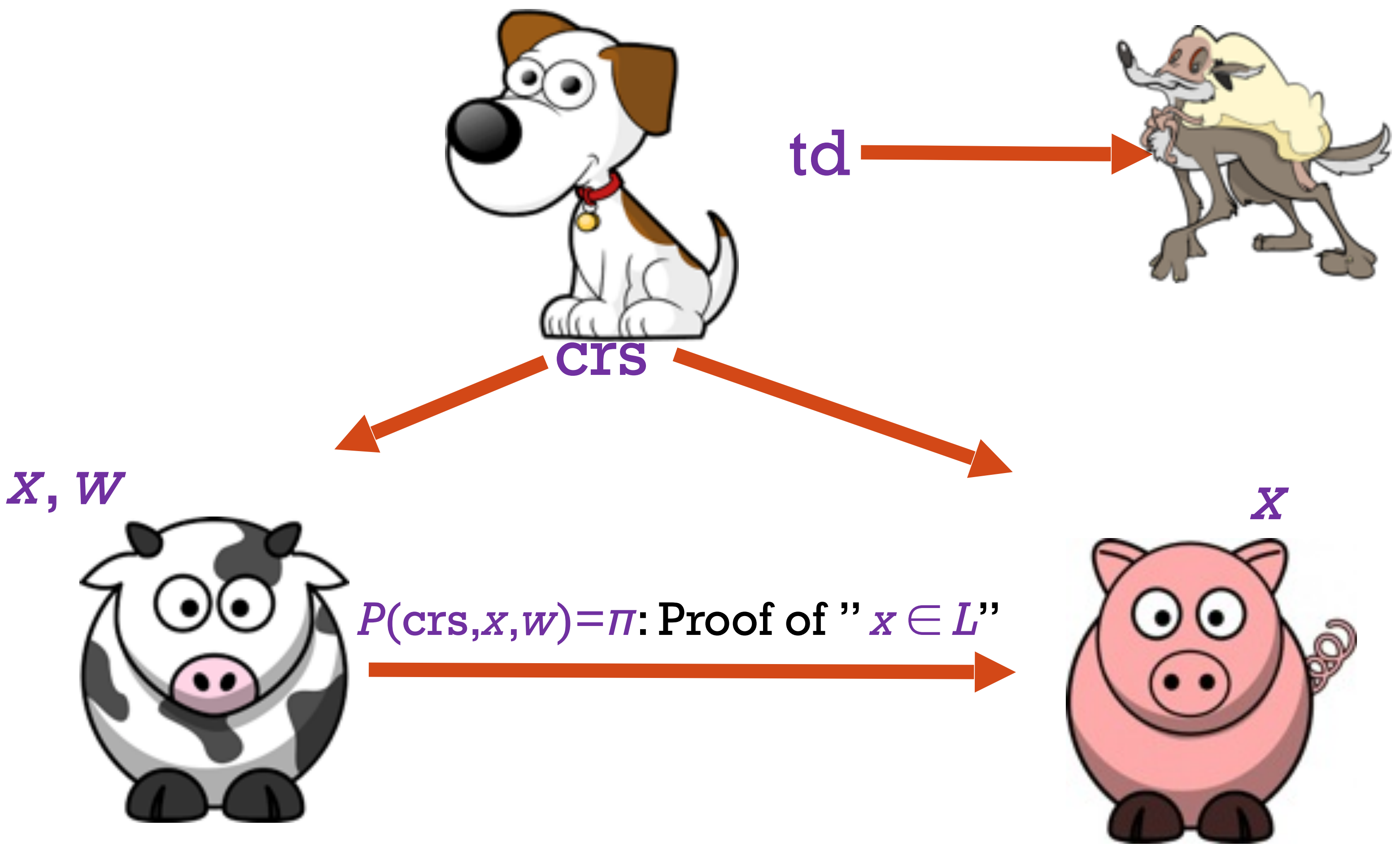


ZERO KNOWLEDGE: CRS MODEL



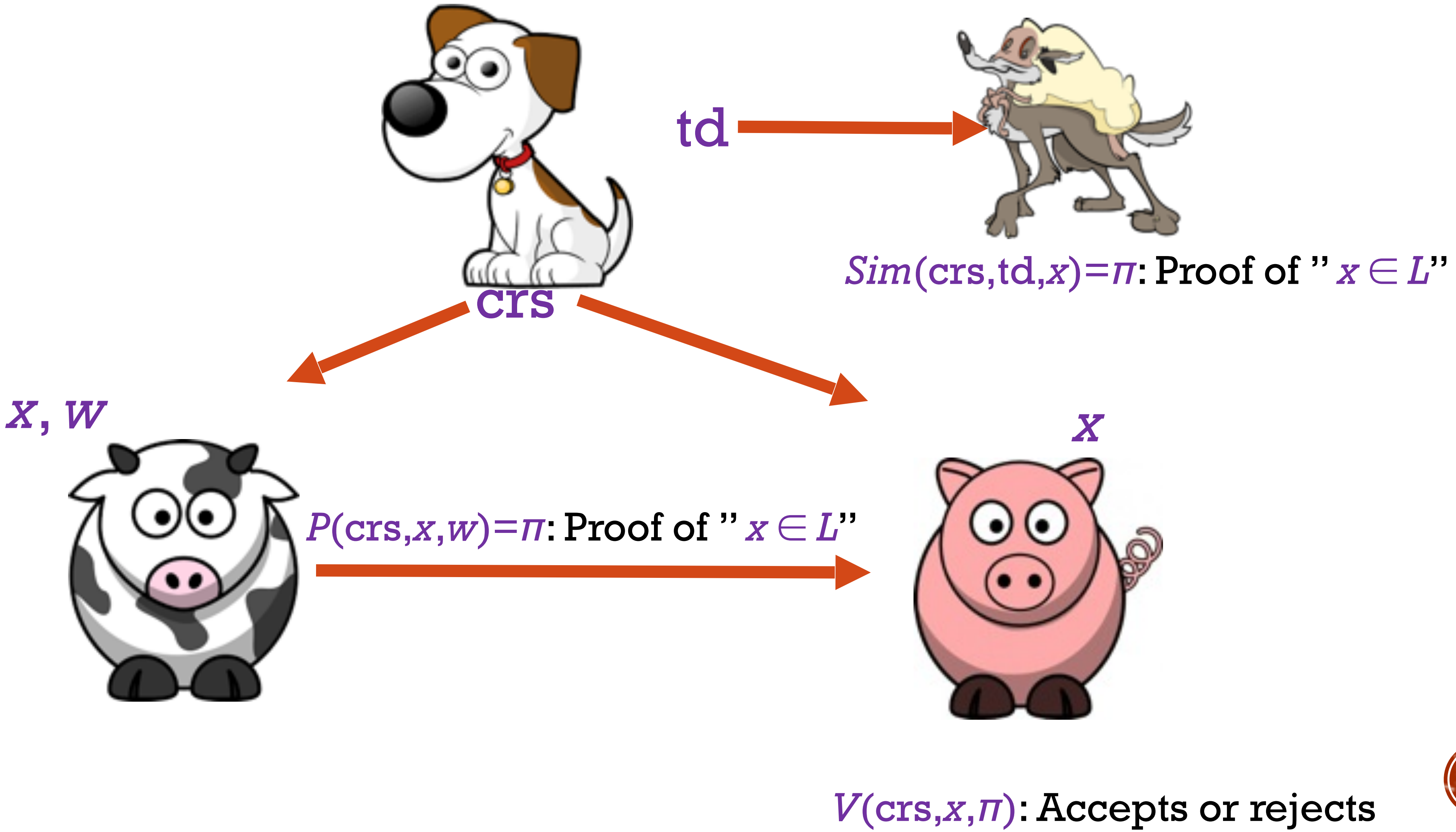
$V(\mathrm{crs}, x, \pi)$: Accepts or rejects

ZERO KNOWLEDGE: CRS MODEL



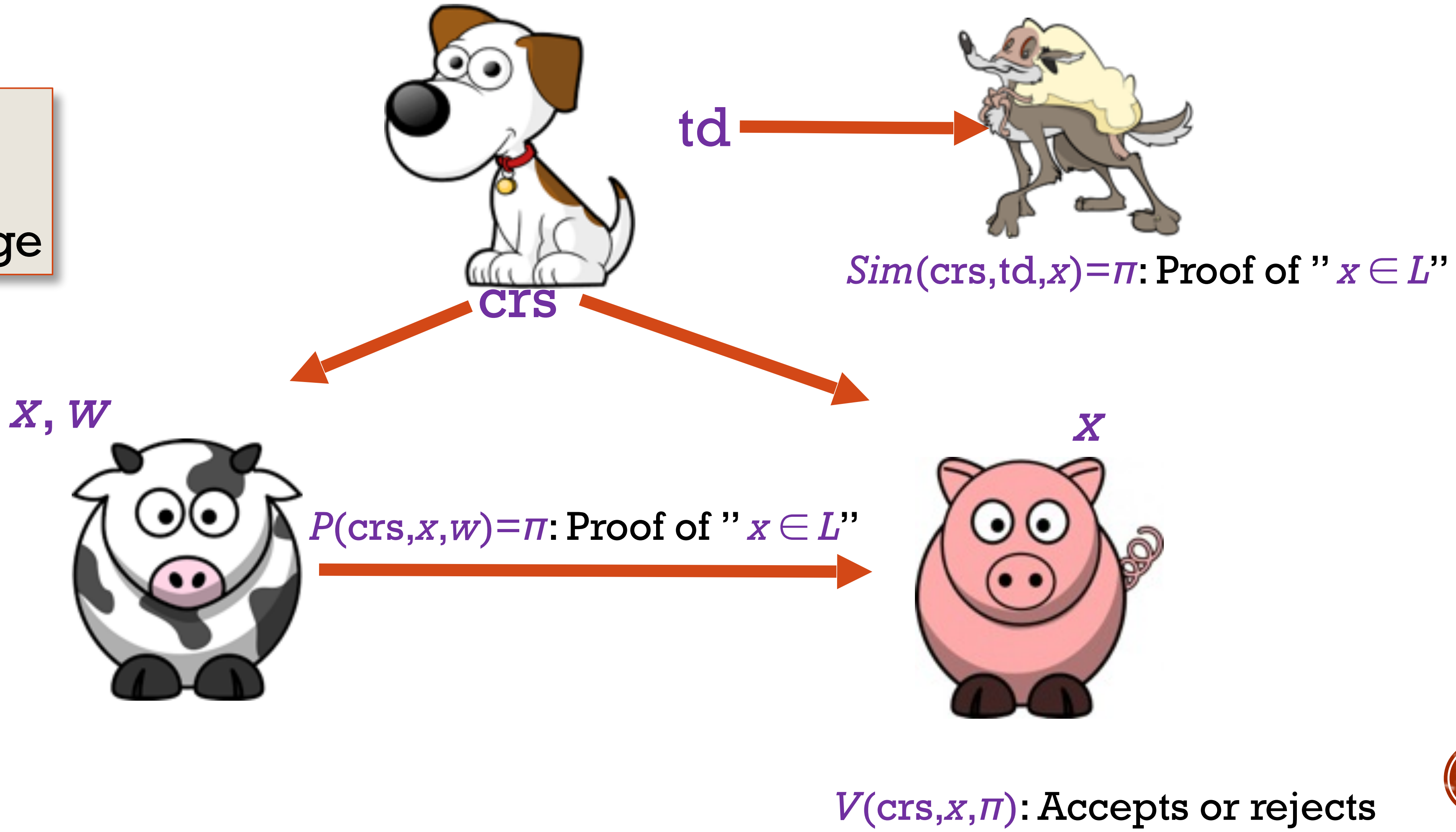
$V(\mathrm{crs},x,\pi)$: Accepts or rejects

ZERO KNOWLEDGE: CRS MODEL



ZERO KNOWLEDGE: CRS MODEL

Correctness
Soundness
Zero knowledge



BILINEAR PAIRINGS

BILINEAR PAIRINGS

- Three cyclic groups of the same order q: $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$

BILINEAR PAIRINGS

- Three cyclic groups of the same order q: $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$
- Generators g_1 of $\mathbb{G}_1$, g_2 of $\mathbb{G}_2$, g_T of $\mathbb{G}_T$

BILINEAR PAIRINGS

- Three cyclic groups of the same order q: $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$
- Generators g_1 of $\mathbb{G}_1$, g_2 of $\mathbb{G}_2$, g_T of $\mathbb{G}_T$
- **Bilinear map:** e: $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$

BILINEAR PAIRINGS

- Three cyclic groups of the same order q: $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$
- Generators g_1 of $\mathbb{G}_1$, g_2 of $\mathbb{G}_2$, g_T of $\mathbb{G}_T$
- **Bilinear map:** $e: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$
- Requirements:
  - Efficiently computable
  - Non-degeneracy: $e\,(g_1, g_2) \neq 1$
  - Bilinearity: $e\,(g_1{}^a, g_2{}^b) = e\,(g_1, g_2)^{ab}$

ASSUMPTIONS & PAIRINGS

- Inverting pairings should be hard

ASSUMPTIONS & PAIRINGS

- Inverting pairings should be hard
  - Given $e(A, B)$, compute either A or B

ASSUMPTIONS & PAIRINGS

- Inverting pairings should be hard
  - Given $e(A, B)$, compute either A or B
  - Analogous to DL: given g^a, compute a

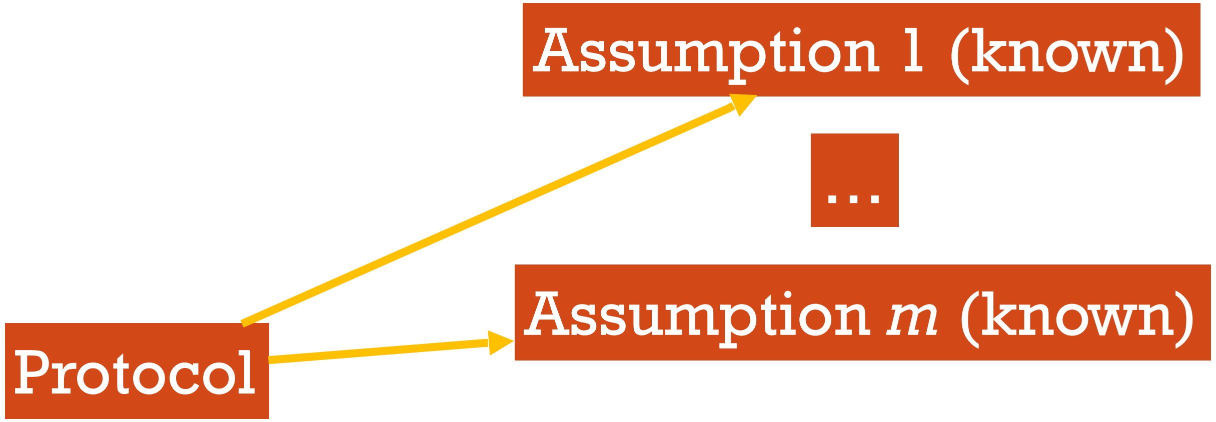
ASSUMPTIONS & PAIRINGS

- Inverting pairings should be hard
  - Given $e\,(A, B)$, compute either A or B
  - Analogous to DL: given g^a, compute a
- What else should be hard?

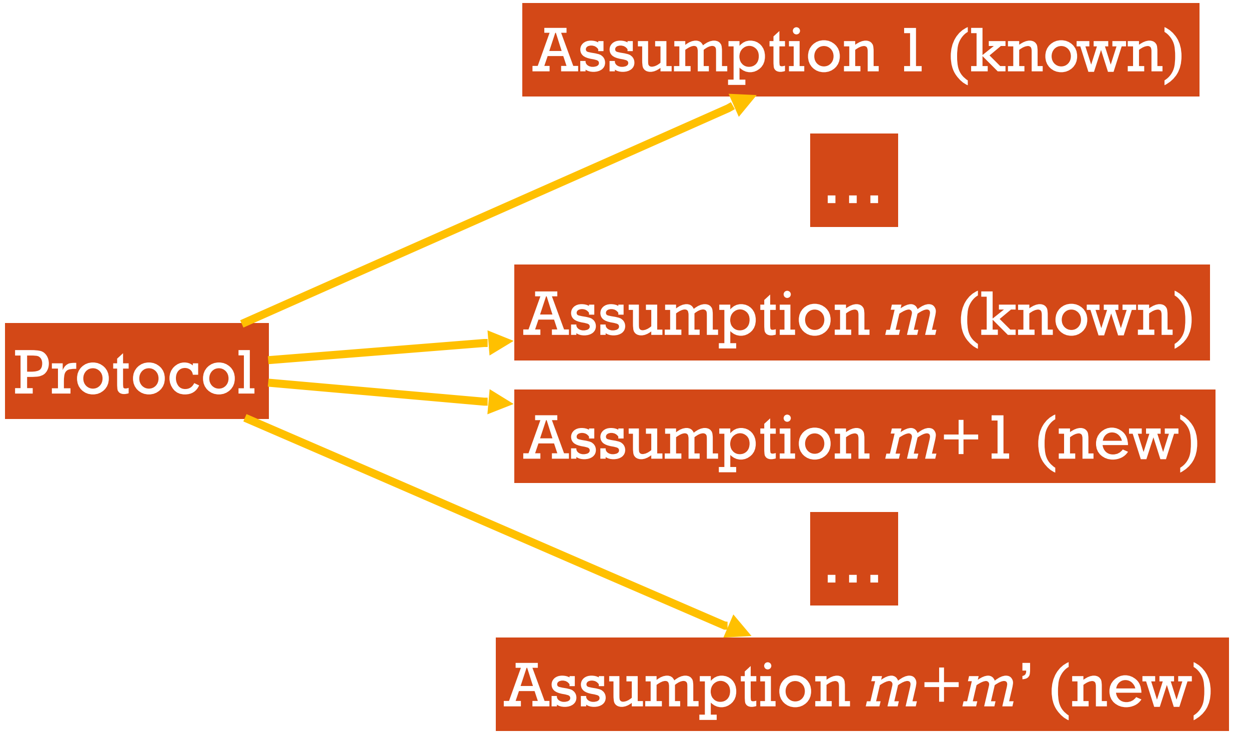
NON-GENERIC APPROACH

Protocol

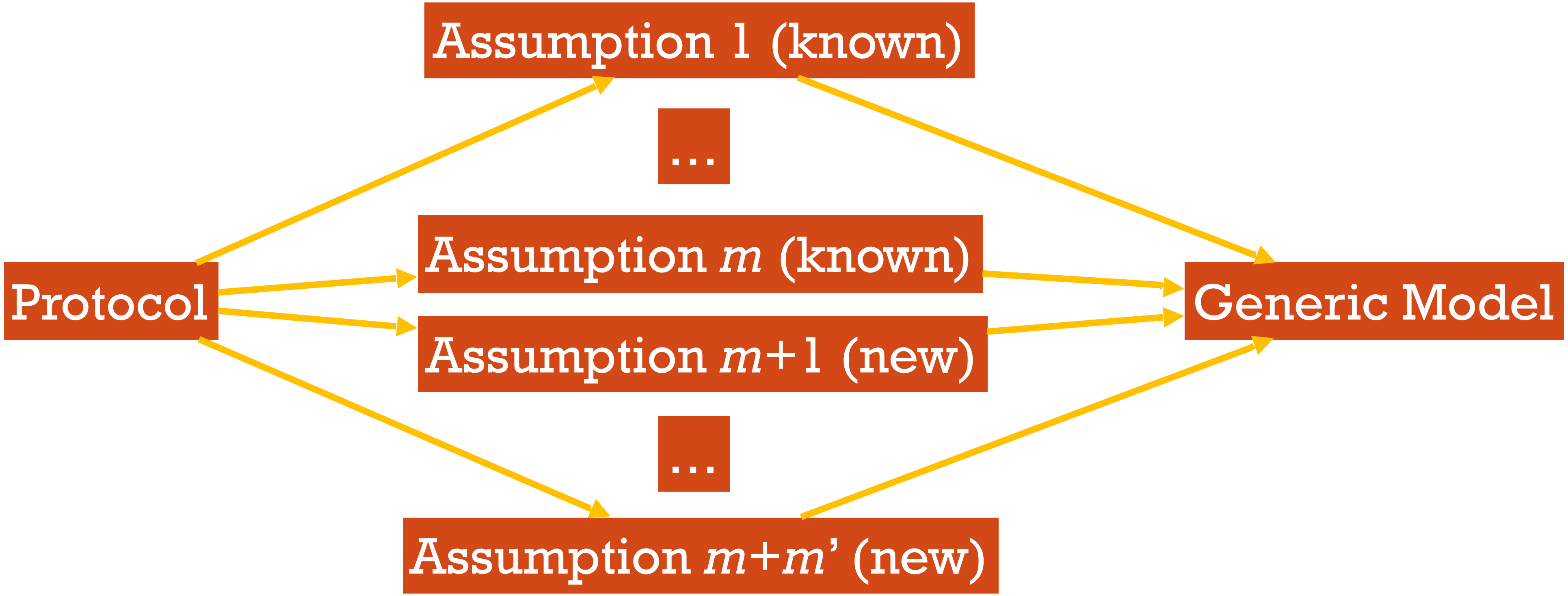
NON-GENERIC APPROACH



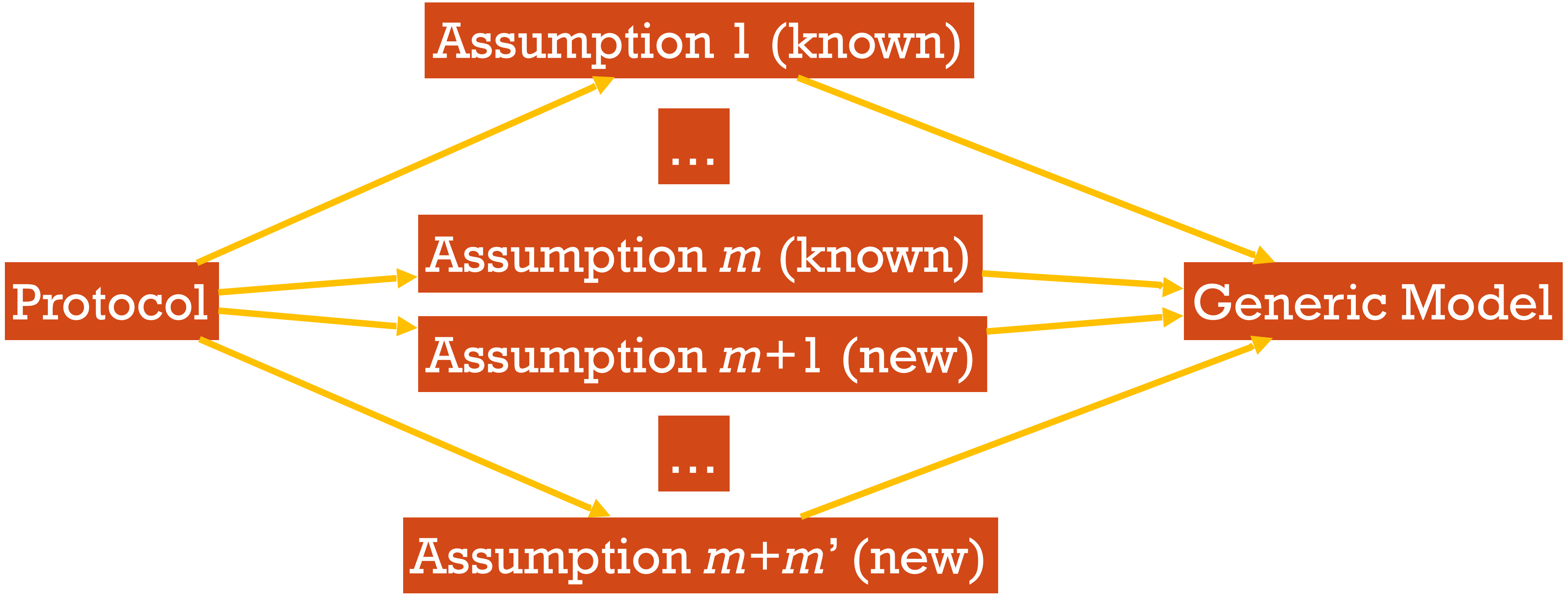
NON-GENERIC APPROACH



NON-GENERIC APPROACH

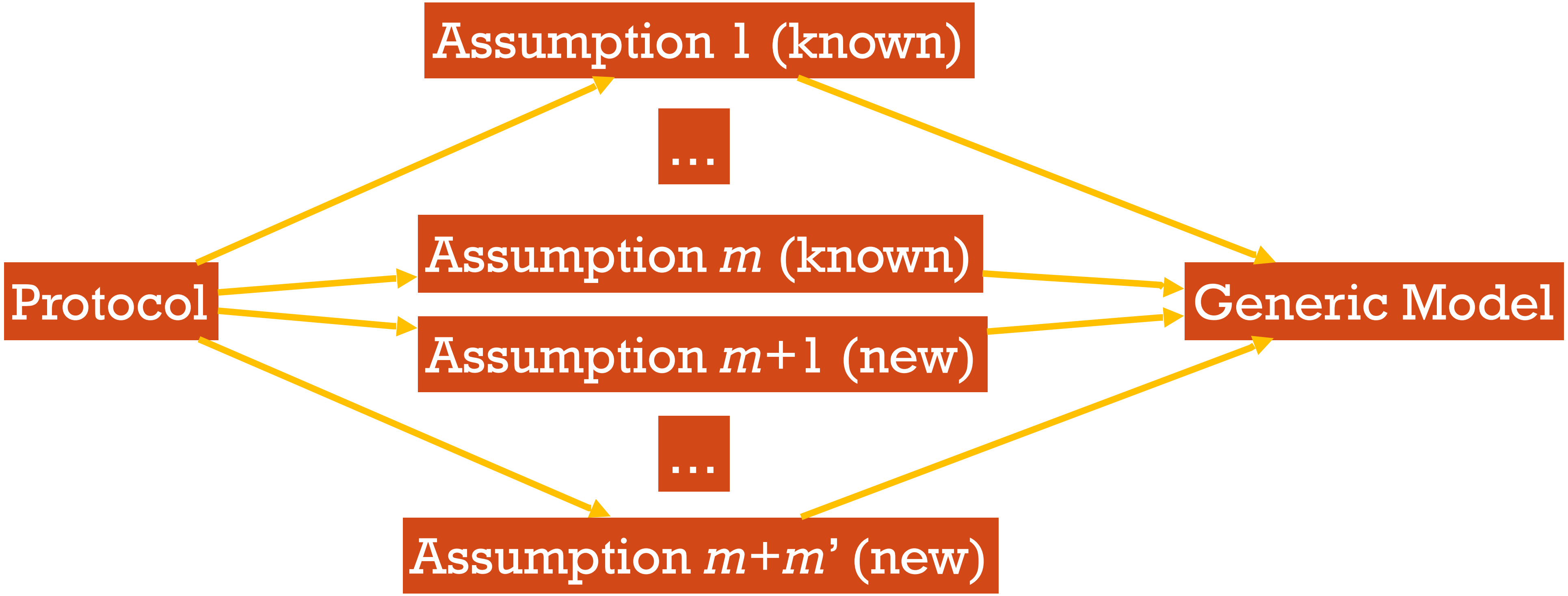


NON-GENERIC APPROACH



Pro: nice if m' is not big, or most assumptions are well-known, or…

NON-GENERIC APPROACH



Pro: nice if m’ is not big, or most assumptions are well-known, or…

Con: each arrow might mean a loss in efficiency

GENERIC MODEL APPROACH

Protocol $\longrightarrow$ Generic Model

Con: proof in GGM is only for restricted adversaries

Pro: only one arrow, thus smaller loss in efficiency

GENERIC BILINEAR GROUP MODEL

- **Meta-Assumption:** adversary only has access to

GENERIC BILINEAR GROUP MODEL

- **Meta-Assumption:** adversary only has access to
  - group operations, bilinear map, equality tests

GENERIC BILINEAR GROUP MODEL

- **Meta-Assumption:** adversary only has access to
  - group operations, bilinear map, equality tests
- Each computed element in G_i (i=1, 2) is given by group operation of two already known elements

GENERIC BILINEAR GROUP MODEL

- **Meta-Assumption:** adversary only has access to
  - group operations, bilinear map, equality tests
- Each computed element in G_i (i=1, 2) is given by group operation of two already known elements
- Recursively, **DL** of each computed element is a known polynomial of some indeterminates

GENERIC BILINEAR GROUP MODEL

- **Meta-Assumption:** adversary only has access to
  - group operations, bilinear map, equality tests
- Each computed element in G_i $(i=1, 2)$ is given by group operation of two already known elements
- Recursively, **DL** of each computed element is a known polynomial of some indeterminates
- Note: we do not handle G_T as a generic group

SOUNDNESS IN GBGM

SOUNDNESS IN GBGM

X_1

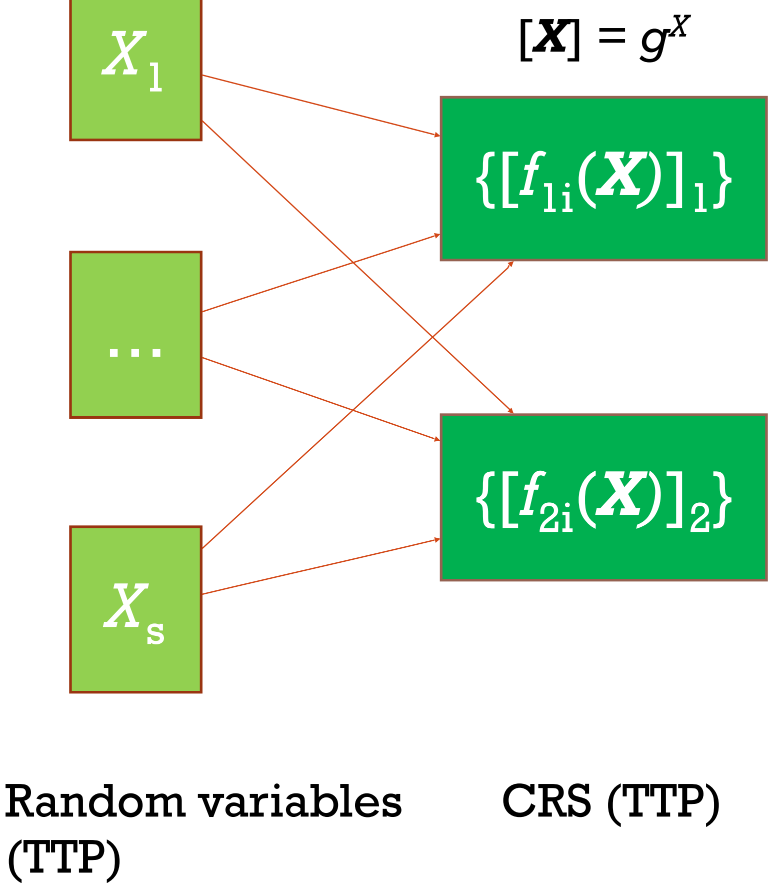
…

X_s

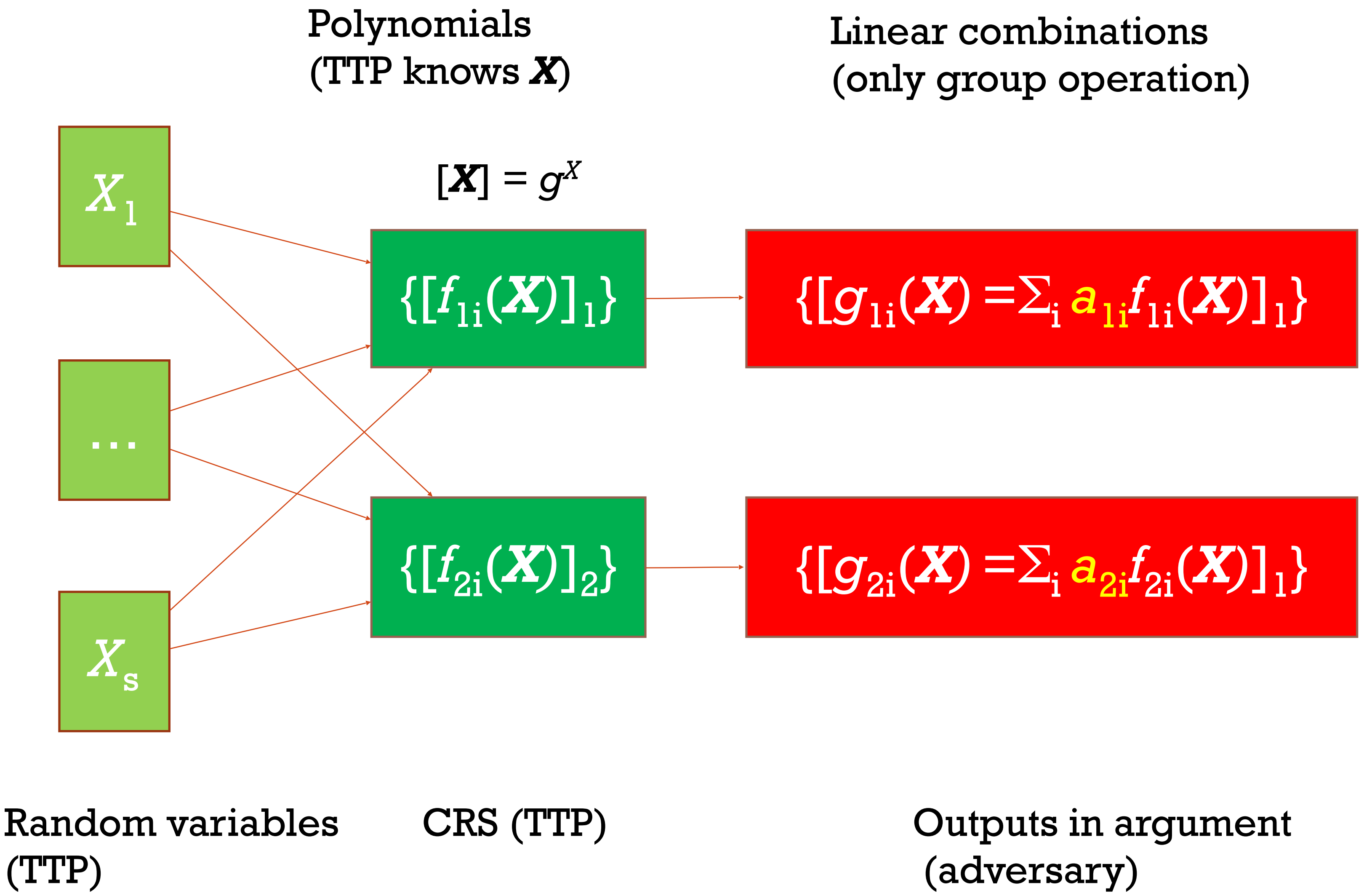
Random variables
(TTP)

SOUNDNESS IN GBGM

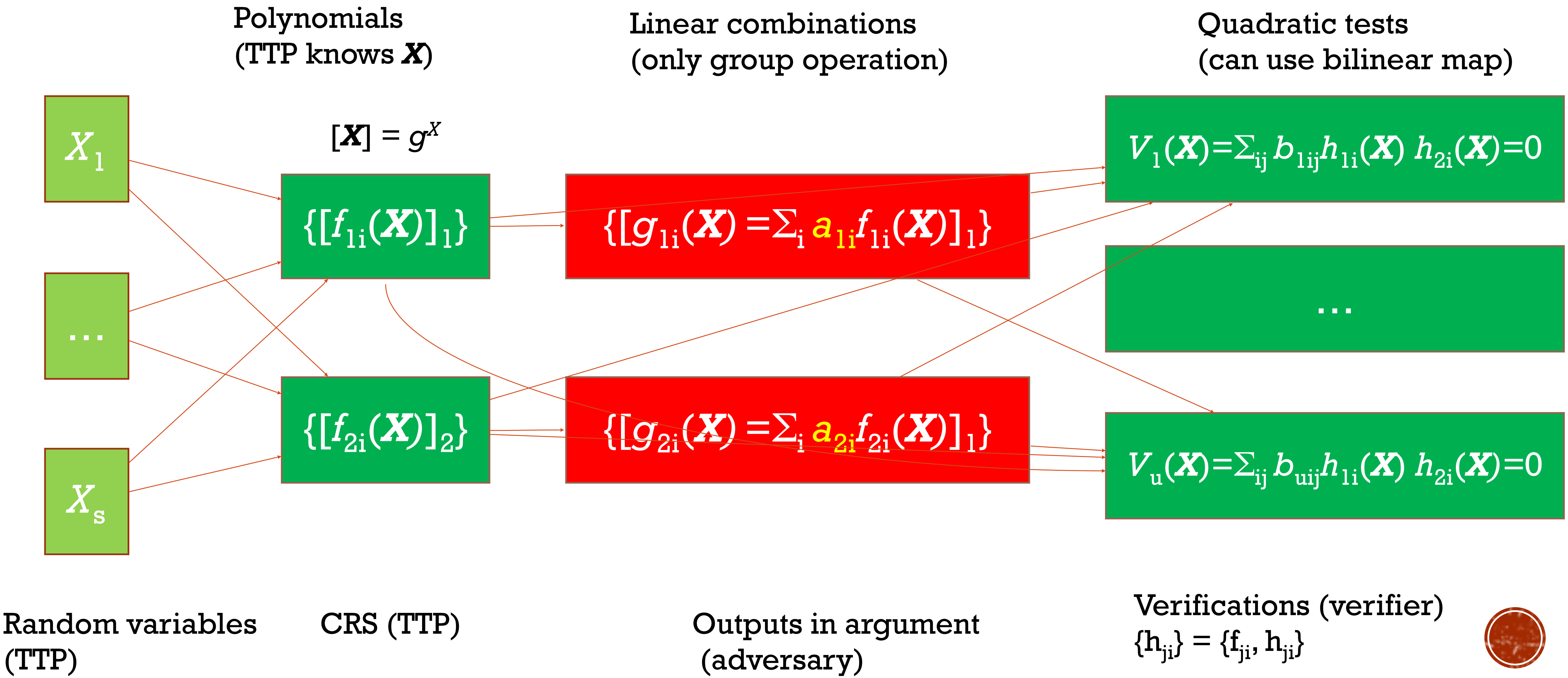
Polynomials
(TTP knows $\boldsymbol{X}$)



SOUNDNESS IN GBGM



SOUNDNESS IN GBGM



SOUNDNESS IN GBGM

- jth verification equation ascertains $V_j(\boldsymbol{X}) = 0$

SOUNDNESS IN GBGM

- jth verification equation ascertains $V_j(\boldsymbol{X}) = 0$
- Solve system of polynomial equations $\{V_j(\boldsymbol{X}) = 0\}$ in coefficients a_{ji} chosen by the adversary

SOUNDNESS IN GBGM

- jth verification equation ascertains $V_j(\boldsymbol{X}) = 0$
- Solve system of polynomial equations $\{V_j(\boldsymbol{X}) = 0\}$ in coefficients a_{ji} chosen by the adversary
- Show that solution's coefficients are "nice"

SOUNDNESS IN GBGM

- jth verification equation ascertains $V_j(\boldsymbol{X}) = 0$
- Solve system of polynomial equations $\{V_j(\boldsymbol{X}) = 0\}$ in coefficients a_{ji} chosen by the adversary
- Show that solution's coefficients are "nice"
  - = restricted to be as in the honest case

INTUITION: CONSTRUCTING ARGUMENT

- **Decomposing:**

INTUITION: CONSTRUCTING ARGUMENT

- **Decomposing:**
  - Write down main building blocks you need to prove in argument

INTUITION: CONSTRUCTING ARGUMENT

- **Decomposing:**
  - Write down main building blocks you need to prove in argument
  - Each "subargument" should be efficiently verifiable (by a single pairing)

INTUITION: CONSTRUCTING ARGUMENT

- **Decomposing:**
  - Write down main building blocks you need to prove in argument
  - Each ”subargument” should be efficiently verifiable (by a single pairing)
  - Ascertain each subargument is sound independently

INTUITION: CONSTRUCTING ARGUMENT

- **Decomposing:**
  - Write down main building blocks you need to prove in argument
  - Each ”subargument” should be efficiently verifiable (by a single pairing)
  - Ascertain each subargument is sound independently
- **CRS composition:**

INTUITION: CONSTRUCTING ARGUMENT

- **Decomposing:**
  - Write down main building blocks you need to prove in argument
  - Each ”subargument” should be efficiently verifiable (by a single pairing)
  - Ascertain each subargument is sound independently

- **CRS composition:**
  - Compose CRS-s of individual subarguments together, getting one big CRS

INTUITION: CONSTRUCTING ARGUMENT

INTUITION: CONSTRUCTING ARGUMENT

INTUITION: CONSTRUCTING ARGUMENT

- **Soundness check:**
  - Is the composed protocol sound?
    - Subarguments get extra inputs in CRS
  - If not: introduce new random variables that guarantee CRS elements are used in only correct subarguments, reiterate

SUBARGUMENTS

- **”Permutation matrix argument”:**

SUBARGUMENTS

- **”Permutation matrix argument”:**
  - Prover commits to permutation; proves this is done correctly

SUBARGUMENTS

- **”Permutation matrix argument”:**
  - Prover commits to permutation; proves this is done correctly
- **”Consistency argument”:**

SUBARGUMENTS

- **”Permutation matrix argument”:**
  - Prover commits to permutation; proves this is done correctly
- **”Consistency argument”:**
  - Prover proves she used the committed permutation to shuffle ciphertexts

SUBARGUMENTS

- **”Permutation matrix argument”:**
  - Prover commits to permutation; proves this is done correctly
- **”Consistency argument”:**
  - Prover proves she used the committed permutation to shuffle ciphertexts
- **”Validity argument”:**

SUBARGUMENTS

- **”Permutation matrix argument”:**
  - Prover commits to permutation; proves this is done correctly
- **”Consistency argument”:**
  - Prover proves she used the committed permutation to shuffle ciphertexts
- **”Validity argument”:**
  - Prover proves each ciphertext has been formed ”correctly”

SUBARGUMENTS

- **"Permutation matrix argument":**
  - Prover commits to permutation; proves this is done correctly
- **"Consistency argument":**
  - Prover proves she used the committed permutation to shuffle ciphertexts
- **"Validity argument":**
  - Prover proves each ciphertext has been formed "correctly"
  - **Correctly:** so that the soundness proof goes through

SUBARGUMENTS

- **”Permutation matrix argument”:**
  - Prover commits to permutation; proves this is done correctly
- **”Consistency argument”:**
  - Prover proves she used the committed permutation to shuffle ciphertexts
- **”Validity argument”:**
  - Prover proves each ciphertext has been formed ”correctly”
  - **Correctly:** so that the soundness proof goes through

PERMUTATION MATRIX ARGUMENT

- **Lemma.** A matrix is permutation matrix iff
  1. It is **stochastic** // rows sum to (1, …, 1)
  2. Each row is **1-sparse**

At most one coefficient is non-zero

PERMUTATION MATRIX ARGUMENT

- **Lemma.** A matrix is permutation matrix iff
  1. It is **stochastic** // rows sum to (1, …, 1)
  2. Each row is **1-sparse**

At most one coefficient is non-zero

1-SPARSITY ARGUMENT

- **Commitment:**

1-SPARSITY ARGUMENT

- **Commitment:**

$P_i\ (X)$ are **linearly independent, well-chosen** polynomials

$$[A_i(\boldsymbol{X})]_i = [a_I P_I\ (X) + rX_\varrho]_i \quad // \text{ i} = 1, 2$$

1-SPARSITY ARGUMENT

- **Commitment:** $P_i\ (X)$ are **linearly independent**, **well-chosen** polynomials

$$[A_i(\boldsymbol{X})]_i = [a_I P_I\ (X) + rX_\varrho]_i \qquad // \ \mathrm{i} = 1, 2$$

- **Argument:** // "square span programs"

1-SPARSITY ARGUMENT

- **Commitment:** $P_i\,(X)$ are **linearly independent**, **well-chosen** polynomials

$$[A_i(\boldsymbol{X})]_i = [a_I P_I\,(X) + rX_\varrho]_i \qquad // \; i = 1, 2$$

- **Argument:** // "square span programs"

$$[\pi(\boldsymbol{X})]_1 = [((a_I P_I\,(X) + P_0\,(X) + rX_\varrho)^2 - 1) \,/\, X_\varrho]_1$$

1-SPARSITY ARGUMENT

- **Commitment:** $P_i\,(X)$ are **linearly independent**, **well-chosen** polynomials

$$[A_i(\boldsymbol{X})]_i = [a_I P_I\,(X) + rX_\varrho]_i \qquad // \ \mathrm{i} = 1, 2$$

- **Argument:** // ”square span programs”

$$[\pi(\boldsymbol{X})]_1 = [((a_I P_I\,(X) + P_0\,(X) + rX_\varrho)^2 - 1) \,/\, X_\varrho]_1$$

- Verification equation:

1-SPARSITY ARGUMENT

- **Commitment:** $P_i\ (X)$ are **linearly independent**, **well-chosen** polynomials

$$[A_i(\mathbf{X})]_i = [a_I P_I\ (X) + rX_\varrho]_i \qquad // \ i = 1, 2$$

- **Argument:** // "square span programs"

$$[\pi(\mathbf{X})]_1 = [((a_I P_I\ (X) + P_0\ (X) + rX_\varrho)^2 - 1)\ /\ X_\varrho]_1$$

- Verification equation:

$$V\ (\mathbf{X}) := (A_1(\mathbf{X}) + X_\alpha + P_0\ (X))\ (A_2(\mathbf{X}) - X_\alpha + P_0\ (X)) - \pi(\mathbf{X})\ X_\varrho - (1 - X_\alpha)^2$$

1-SPARSITY ARGUMENT

- **Commitment:** $P_i\,(X)$ are **linearly independent**, **well-chosen** polynomials

$$[A_i(\mathbf{X})]_i = [a_I P_I\,(X) + rX_\varrho]_i \qquad // \; i = 1, 2$$

- **Argument:** // "square span programs"

$$[\pi(\mathbf{X})]_1 = [((a_I P_I\,(X) + P_0\,(X) + rX_\varrho)^2 - 1)\,/\,X_\varrho]_1$$

- Verification equation:

$$V\,(\mathbf{X}) := (A_1(\mathbf{X}) + X_\alpha + P_0\,(X))\,(A_2(\mathbf{X}) - X_\alpha + P_0\,(X)) - \pi(\mathbf{X})\,X_\varrho - (1 - X_\alpha)^2$$

$$= 0$$

honest prover: $[A_i(\boldsymbol{X})]_i = [a_I P_I(X) + rX_\varrho]_i$

SOUNDNESS PROOF: IDEA

honest prover: $[A_i(\boldsymbol{X})]_i = [a_I P_I(X) + rX_\varrho]_i$

SOUNDNESS PROOF: IDEA

- In GBGM we know constants $a_{1i}, A_{1\varrho}, \ldots$, s.t. for $\boldsymbol{X} = (X, X_\varrho, X_\alpha, X_\beta, X_\gamma, X_{sk})$

honest prover: $[A_i(\boldsymbol{X})]_i = [a_I P_I(X) + rX_\varrho]_i$

SOUNDNESS PROOF: IDEA

- In GBGM we know constants $a_{1i}, A_{1\varrho}, \ldots$, s.t. for $\boldsymbol{X} = (X, X_\varrho, X_\alpha, X_\beta, X_\gamma, X_{sk})$

$$A_1(\boldsymbol{X}) = \sum a_{1i} P_i(X) + A_{1\varrho} X_\varrho + A_{1\alpha}(X_\alpha + P_0(X)) + A_{11} P_0(X) + \ldots$$

CRS: $(\{[P_i(\boldsymbol{X})]_1\}_i,\ [\boldsymbol{X}_\varrho]_1,\ [\boldsymbol{X}_\alpha + P_0(\boldsymbol{X})]_1,\ [P_0(\boldsymbol{X})]_1, \ldots,$
$(\{[P_i(\boldsymbol{X})]_2\}_i,\ [\boldsymbol{X}_\varrho]_2, [-\boldsymbol{X}_\alpha + P_0(\boldsymbol{X})]_2, [1]_2, \ldots)$

honest prover: $[A_i(\boldsymbol{X})]_i = [a_I P_I(X) + rX_\varrho]_i$

SOUNDNESS PROOF: IDEA

- In GBGM we know constants $a_{1i}, A_{1\varrho}, \ldots$, s.t. for $\boldsymbol{X} = (X, X_\varrho, X_\alpha, X_\beta, X_\gamma, X_{sk})$

$$A_1(\boldsymbol{X}) = \sum a_{1i} P_i(X) + A_{1\varrho} X_\varrho + A_{1\alpha}(X_\alpha + P_0(X)) + A_{11} P_0(X) + \ldots$$

$$A_2(\boldsymbol{X}) = \sum a_{2i} P_i(X) + A_{2\varrho} X_\varrho + A_{2\alpha}(-X_\alpha + P_0(X)) + A_{21} + \ldots$$

CRS: $(\{[P_i(\boldsymbol{X})]_1\}_i,\ [\boldsymbol{X}_\varrho]_1,\ [\boldsymbol{X}_\alpha + P_0(\boldsymbol{X})]_1,\ [P_0(\boldsymbol{X})]_1, \ldots,$
$(\{[P_i(\boldsymbol{X})]_2\}_i,\ [\boldsymbol{X}_\varrho]_2, [-\boldsymbol{X}_\alpha + P_0(\boldsymbol{X})]_2, [1]_2, \ldots)$

honest prover: $[A_i(\boldsymbol{X})]_i = [a_I P_I(X) + rX_\varrho]_i$

SOUNDNESS PROOF: IDEA

- In GBGM we know constants $a_{1i}, A_{1\varrho}, \ldots$, s.t. for $\boldsymbol{X} = (X, X_\varrho, X_\alpha, X_\beta, X_\gamma, X_{sk})$

$$A_1(\boldsymbol{X}) = \Sigma\, a_{1i} P_i(X) + A_{1\varrho} X_\varrho + A_{1\alpha}(X_\alpha + P_0(X)) + A_{11} P_0(X) + \ldots$$

$$A_2(\boldsymbol{X}) = \Sigma\, a_{2i} P_i(X) + A_{2\varrho} X_\varrho + A_{2\alpha}(-X_\alpha + P_0(X)) + A_{21} + \ldots$$

$$\pi(\boldsymbol{X}) = \Sigma\, \pi_i P_i(X) + \pi_\varrho X_\varrho + \pi_\alpha(X_\alpha + P_0(X)) + \pi_1 P_0(X) + \ldots$$

CRS: $(\{[P_i(\boldsymbol{X})]_1\}_i,\ [\boldsymbol{X}_\varrho]_1,\ [\boldsymbol{X}_\alpha + P_0(\boldsymbol{X})]_1,\ [P_0(\boldsymbol{X})]_1, \ldots,$
$(\{[P_i(\boldsymbol{X})]_2\}_i,\ [\boldsymbol{X}_\varrho]_2,\ [-\boldsymbol{X}_\alpha + P_0(\boldsymbol{X})]_2,\ [1]_2, \ldots)$

honest prover: $[A_i(\boldsymbol{X})]_i = [a_I P_I(X) + rX_\varrho]_i$

SOUNDNESS PROOF: IDEA

- In GBGM we know constants $a_{1i}, A_{1\varrho}, \ldots$, s.t. for $\boldsymbol{X} = (X, X_\varrho, X_\alpha, X_\beta, X_\gamma, X_{sk})$

$$A_1(\boldsymbol{X}) = \sum a_{1i} P_i(X) + A_{1\varrho} X_\varrho + A_{1\alpha}(X_\alpha + P_0(X)) + A_{11} P_0(X) + \ldots$$

$$A_2(\boldsymbol{X}) = \sum a_{2i} P_i(X) + A_{2\varrho} X_\varrho + A_{2\alpha}(-X_\alpha + P_0(X)) + A_{21} + \ldots$$

$$\pi(\boldsymbol{X}) = \sum \pi_i P_i(X) + \pi_\varrho X_\varrho + \pi_\alpha(X_\alpha + P_0(X)) + \pi_1 P_0(X) + \ldots$$

- Verification equation states

CRS: $(\{[P_i(\boldsymbol{X})]_1\}_i,\ [\boldsymbol{X}_\varrho]_1,\ [\boldsymbol{X}_\alpha + P_0(\boldsymbol{X})]_1,\ [P_0(\boldsymbol{X})]_1, \ldots,$
$(\{[P_i(\boldsymbol{X})]_2\}_i,\ [\boldsymbol{X}_\varrho]_2, [-\boldsymbol{X}_\alpha + P_0(\boldsymbol{X})]_2, [1]_2, \ldots)$

honest prover: $[A_i(\mathbf{X})]_i = [a_I P_I(X) + rX_\varrho]_i$

SOUNDNESS PROOF: IDEA

- In GBGM we know constants $a_{1i}, A_{1\varrho}, \ldots$, s.t. for $\mathbf{X} = (X, X_\varrho, X_\alpha, X_\beta, X_\gamma, X_{sk})$

$$A_1(\mathbf{X}) = \Sigma\, a_{1i} P_i(X) + A_{1\varrho} X_\varrho + A_{1\alpha}(X_\alpha + P_0(X)) + A_{11} P_0(X) + \ldots$$

$$A_2(\mathbf{X}) = \Sigma\, a_{2i} P_i(X) + A_{2\varrho} X_\varrho + A_{2\alpha}(-X_\alpha + P_0(X)) + A_{21} + \ldots$$

$$\pi(\mathbf{X}) = \Sigma\, \pi_i P_i(X) + \pi_\varrho X_\varrho + \pi_\alpha(X_\alpha + P_0(X)) + \pi_1 P_0(X) + \ldots$$

- Verification equation states

$$V(\mathbf{X}) = (A_1(\mathbf{X}) + X_\alpha + P_0(X))(A_2(\mathbf{X}) - X_\alpha + P_0(X)) - \pi(\mathbf{X}) X_\varrho - (1 - X_\alpha)^2 = 0$$

CRS: $(\{[P_i(\mathbf{X})]_1\}_i,\ [\mathbf{X}_\varrho]_1,\ [\mathbf{X}_\alpha + P_0(\mathbf{X})]_1,\ [P_0(\mathbf{X})]_1, \ldots,$
$(\{[P_i(\mathbf{X})]_2\}_i,\ [\mathbf{X}_\varrho]_2,\ [-\mathbf{X}_\alpha + P_0(\mathbf{X})]_2,\ [1]_2, \ldots)$

honest prover: $[A_i(\boldsymbol{X})]_i = [a_I P_I(X) + rX_\varrho]_i$

SOUNDNESS PROOF: IDEA

- In GBGM we know constants $a_{1i}, A_{1\varrho}, \ldots$, s.t. for $\boldsymbol{X} = (X, X_\varrho, X_\alpha, X_\beta, X_\gamma, X_{sk})$

$$A_1(\boldsymbol{X}) = \Sigma\, a_{1i} P_i(X) + A_{1\varrho} X_\varrho + A_{1\alpha}(X_\alpha + P_0(X)) + A_{11} P_0(X) + \ldots$$

$$A_2(\boldsymbol{X}) = \Sigma\, a_{2i} P_i(X) + A_{2\varrho} X_\varrho + A_{2\alpha}(-X_\alpha + P_0(X)) + A_{21} + \ldots$$

$$\pi(\boldsymbol{X}) = \Sigma\, \pi_i P_i(X) + \pi_\varrho X_\varrho + \pi_\alpha(X_\alpha + P_0(X)) + \pi_1 P_0(X) + \ldots$$

- Verification equation states

$$V(\boldsymbol{X}) = (A_1(\boldsymbol{X}) + X_\alpha + P_0(X))(A_2(\boldsymbol{X}) - X_\alpha + P_0(X)) - \pi(\boldsymbol{X}) X_\varrho - (1 - X_\alpha)^2 = 0$$

- Goal: find coefficients s.t. verification equation is satisfied

CRS: $(\{[P_i(\boldsymbol{X})]_1\}_i,\ [X_\varrho]_1,\ [X_\alpha + P_0(\boldsymbol{X})]_1,\ [P_0(\boldsymbol{X})]_1, \ldots,$
$(\{[P_i(\boldsymbol{X})]_2\}_i,\ [X_\varrho]_2,\ [-X_\alpha + P_0(\boldsymbol{X})]_2,\ [1]_2, \ldots)$

SOLVING SYSTEM OF POL. EQUATIONS

SOLVING SYSTEM OF POL. EQUATIONS

- **Goal:**
  - find coefficients s.t. $V(\mathbf{X}) = 0$

SOLVING SYSTEM OF POL. EQUATIONS

- **Goal:**
  - find coefficients s.t. $V(\mathbf{X}) = 0$
- **Step 1:**
  - $V(\mathbf{X}) = 0$ iff each coefficient $[X_\alpha{}^j X_\varrho{}^k \ldots]\, V(\mathbf{X}) = 0$

SOLVING SYSTEM OF POL. EQUATIONS

- **Goal:**
  - find coefficients s.t. $V(\mathbf{X}) = 0$
- **Step 1:**
  - $V(\mathbf{X}) = 0$ iff each coefficient $[X_\alpha{}^j X_\varrho{}^k \ldots]\, V(\mathbf{X}) = 0$
- This is a system of polynomial equations
  - … and a nasty one
  - of more than 20 polynomial equations

$\{i_1,\ldots,i_4\}$	$\mathrm{coeff}_{\mu(\boldsymbol{i})}(\mathcal{V}_{1sp}(\boldsymbol{X})\cdot\mathcal{V}^*_{1sp}(\boldsymbol{X}))$
$\{1,2,1,0\}$	$-A_\varrho(B_\alpha+1)+(A_\alpha+1)B_\varrho-C_\alpha$
$\{1,2,0,1\}$	$-A_\gamma(B_\alpha+1)$
$\{1,2,0,0\}$	$-A_{\varrho\beta}(B_\alpha+1)$
$\{1,1,2,0\}$	$(A_\alpha+1)B_\beta$
$\{1,1,1,1\}$	$(A_\alpha+1)B_\gamma$
$\{1,1,1,0\}$	$-a(X)(B_\alpha+1)+(A_\alpha+1)\,(b(X)+B_1)-A_0(B_\alpha+1)P_0(X)$
$\{1,0,1,0\}$	$-(B_\alpha+1)Z(X)a^\dagger(X)$
$\{0,3,1,0\}$	$A_\varrho B_\varrho-C_\varrho$
$\{0,3,0,1\}$	$A_\gamma B_\varrho-C_\gamma$
$\{0,3,0,0\}$	$A_{\varrho\beta}B_\varrho-C_{\varrho\beta}$
$\{0,2,2,0\}$	$A_\varrho B_\beta$
$\{0,2,1,1\}$	$A_\gamma B_\beta+A_\varrho B_\gamma$
$\{0,2,1,0\}$	$a(X)B_\varrho+A_\varrho\,(b(X)+B_1)+A_{\varrho\beta}B_\beta+$ $P_0(X)\,(A_\varrho(B_\alpha+1)+(A_\alpha+A_0+1)\,B_\varrho-C_\alpha-C_0)-c(X)$
$\{0,2,0,2\}$	$A_\gamma B_\gamma$
$\{0,2,0,1\}$	$A_\gamma\,(b(X)+B_1)+A_{\varrho\beta}B_\gamma+A_\gamma(B_\alpha+1)P_0(X)$
$\{0,2,0,0\}$	$A_{\varrho\beta}\,(b(X)+B_1)+A_{\varrho\beta}(B_\alpha+1)P_0(X)$
$\{0,1,2,0\}$	$a(X)B_\beta+(A_\alpha+A_0+1)\,B_\beta P_0(X)$
$\{0,1,1,1\}$	$a(X)B_\gamma+(A_\alpha+A_0+1)\,B_\gamma P_0(X)$
$\{0,1,1,0\}$	$-Z(X)c^\dagger(X)+P_0(X)\,(a(X)(B_\alpha+1)+(A_\alpha+A_0+1)\,(b(X)+B_1))+$ $a(X)\,(b(X)+B_1)+(A_\alpha+A_0+1)\,(B_\alpha+1)P_0(X)^2-1+$ $B_\varrho Z(X)a^\dagger(X)$
$\{0,0,2,0\}$	$B_\beta Z(X)a^\dagger(X)$
$\{0,0,1,1\}$	$B_\gamma Z(X)a^\dagger(X)$
$\{0,0,1,0\}$	$Z(X)\,(b(X)+B_1)\,a^\dagger(X)+(B_\alpha+1)P_0(X)Z(X)a^\dagger(X)$

SOLVING…

SOLVING…

- Used a mixture of computer algebra system and manual labor

SOLVING…

- Used a mixture of computer algebra system and manual labor

1. Use linear independence of $P_i\,(X)$ to split some coefficients

SOLVING…

- Used a mixture of computer algebra system and manual labor

1. Use linear independence of $P_i(X)$ to split some coefficients
2. Construct Gröbner basis of system of polynomial equations
   - Needs(?) a CAS…

SOLVING…

- Used a mixture of computer algebra system and manual labor

1. Use linear independence of $P_i\,(X)$ to split some coefficients
2. Construct Gröbner basis of system of polynomial equations
   - Needs(?) a CAS…
3. Solve the Gröbner basis
   - Can be done manually or by using CAS

SOLVING…

- Used a mixture of computer algebra system and manual labor

1. Use linear independence of $P_i(X)$ to split some coefficients
2. Construct Gröbner basis of system of polynomial equations
   - Needs(?) a CAS…
3. Solve the Gröbner basis
   - Can be done manually or by using CAS

- Obtain that $A_i(X) = a_I P_I(X)$ => Sound

THANK YOU!

Panoramix