AN EVOLUTION OF MODELS FOR ZERO-KNOWLEDGE PROOFS

SARAH MEIKLEJOHN (GOOGLE & UCL)

INTRODUCTION

This talk will cover the rapid evolution of zero-knowledge proofs according to their **models** and **applications**

For an introduction to other aspects, check out:

- https://zkproof.org/
- Jens Groth's excellent invited talk at Crypto 2021

INTRODUCTION TO ME 👋

I’m a professor at UCL and (recently) a researcher at Google

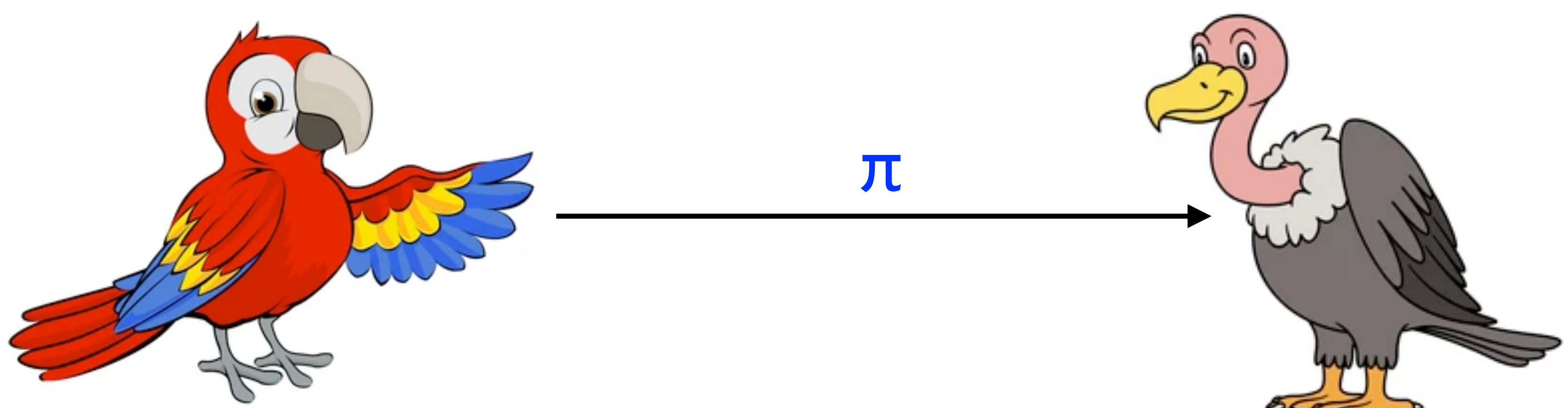
I try to both construct privacy-enhancing technologies and empirically measure their success (e.g. I have done a lot of research on de-anonymizing cryptocurrencies)

At Google I work on the Certificate Transparency team, looking at verifiable data structures (like Merkle trees)

INTRODUCTION TO ZERO KNOWLEDGE

In a zero-knowledge proof [GMR89], a prover wants to convince a verifier that there exists a **witness** w corresponding to some **instance** x of a language L_R (witness w for the **statement** $(x,w) \in R$)

In a non-interactive zero-knowledge proof (NIZK) [BFM88], this is done without any interaction

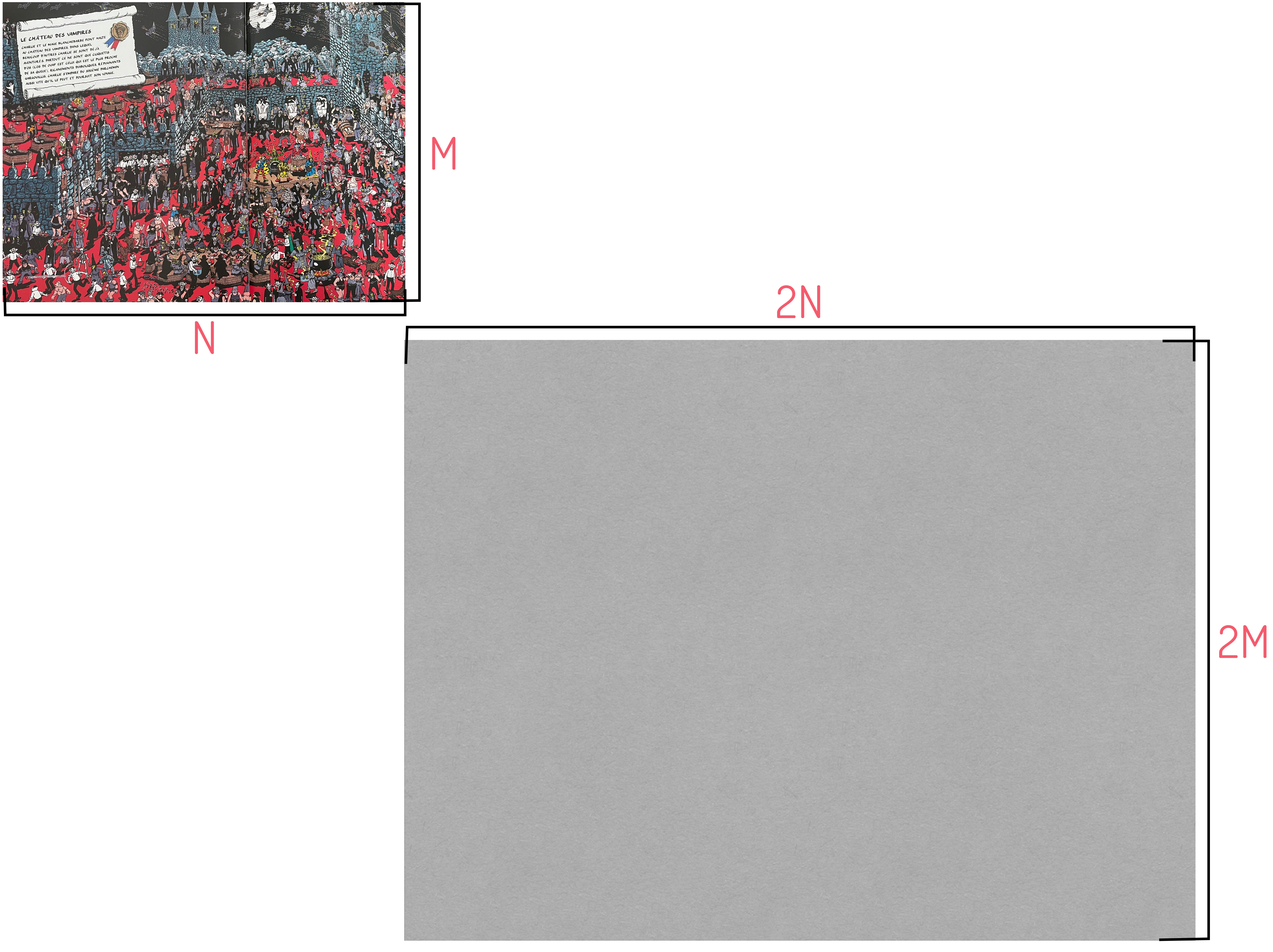


Prove(x, w) → π   Verify(x, π) → accept/reject

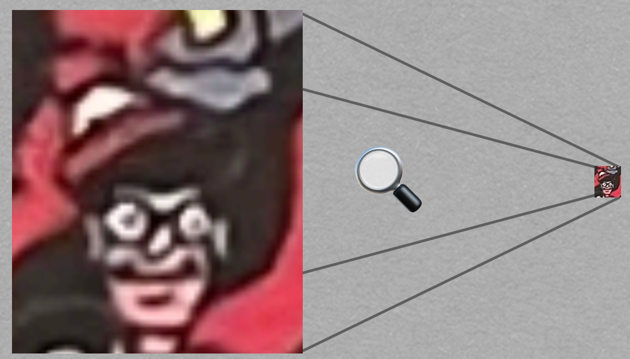
Soundness: hard for the prover to convince the verifier if $x \notin L_R$

Zero knowledge: the verifier learns nothing except that $x \in L_R$

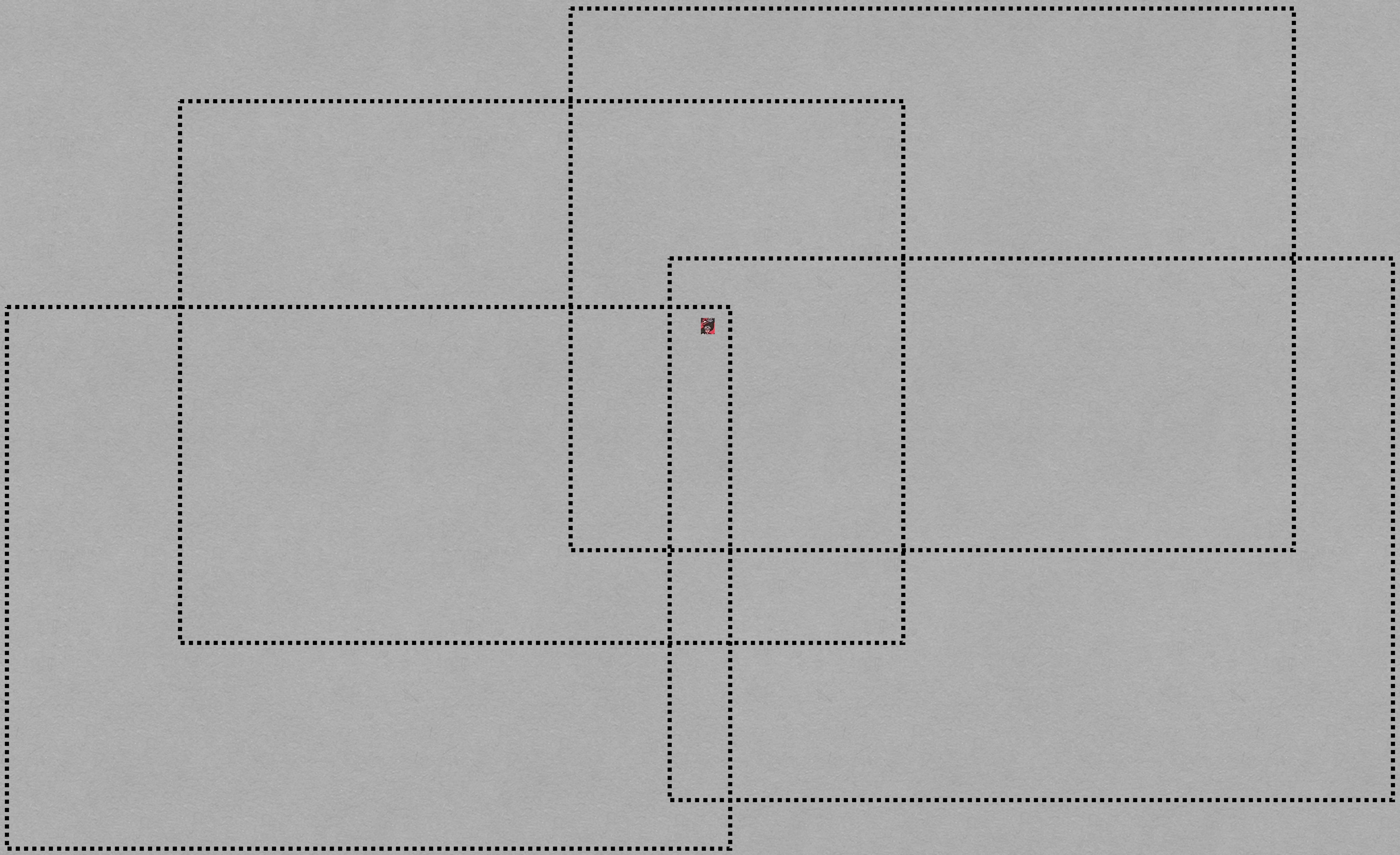

LE CHÂTEAU DES VAMPIRES
CHARLIE ET LE MAGE BLANCHEBARBE FONT HALTE
AU CHÂTEAU DES VAMPIRES, DANS LEQUEL
BEAUCOUP D'AUTRES CHARLIE SE SONT DÉJÀ
AVENTURÉS. PARTOUT CE NE SONT QUE CLIQUETIS
D'OS (L'OS DE OUAF EST CELUI QUI EST LE PLUS PROCHE
DE SA QUEUE), RICANEMENTS DIABOLIQUES, RÉPUGNANTS
GARGOUILLIS. CHARLIE S'EMPARE DU SIXIÈME PARCHEMIN
AUSSI VITE QU'IL LE PEUT ET POURSUIT SON VOYAGE.


LE CHÂTEAU DES VAMPIRES
CHARLIE ET LE MAGE BLANCHEBARBE FONT HALTE AU CHÂTEAU DES VAMPIRES, DANS LEQUEL BEAUCOUP D'AUTRES CHARLIE SE SONT DÉJÀ AVENTURÉS. PARTOUT CE NE SONT QUE CLIQUETIS D'OS (L'OS DE OUAF EST CELUI QUI EST LE PLUS PROCHE DE SA QUEUE), RICANEMENTS DIABOLIQUES, RÉPUGNANTS GARGOUILLIS. CHARLIE S'EMPARE DU SIXIÈME PARCHEMIN AUSSI VITE QU'IL LE PEUT ET POURSUIT SON VOYAGE.
M
N
2N
2M

SOUNDNESS: THE VERIFIER CAN SEE WALDO FOR THEMSELVES!



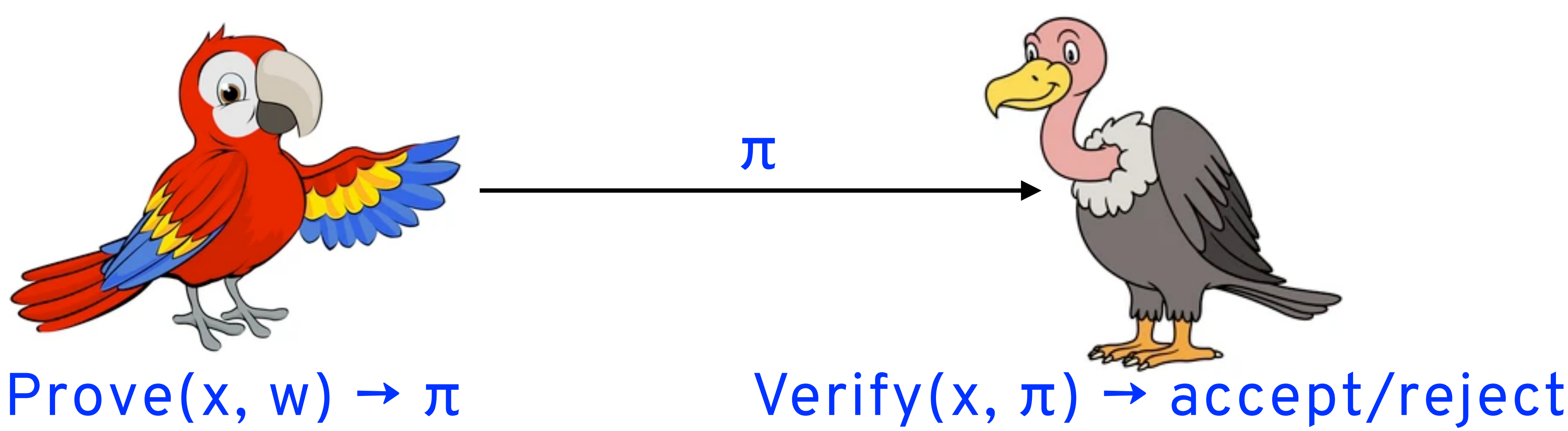
ZERO KNOWLEDGE: THE BOOK COULD BE ANYWHERE



DEFINING ZERO KNOWLEDGE

In a zero-knowledge proof [GMR89], a prover wants to convince a verifier that there exists a **witness** w corresponding to some **instance** x of a language L_R (witness w for the **statement** $(x,w) \in R$)

In a non-interactive zero-knowledge proof (NIZK) [BFM88], this is done without any interaction



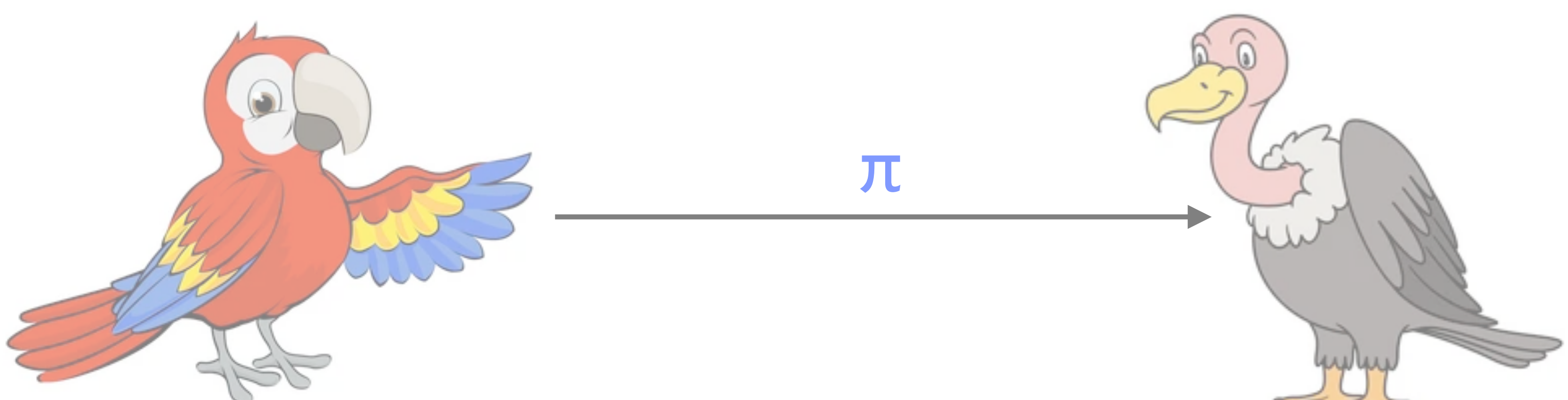
Soundness: hard for the prover to convince the verifier if $x \notin L_R$

Zero knowledge: the verifier learns nothing except that $x \in L_R$

DEFINING ZERO KNOWLEDGE

In a zero-knowledge proof [GMR89], a prover wants to convince a verifier that there exists a **witness** w corresponding to some **instance** x of a language L_R (witness w for the **statement** $(x,w) \in R$)

In a non-interactive zero-knowledge proof (NIZK) [BFM88], this is done without any interaction



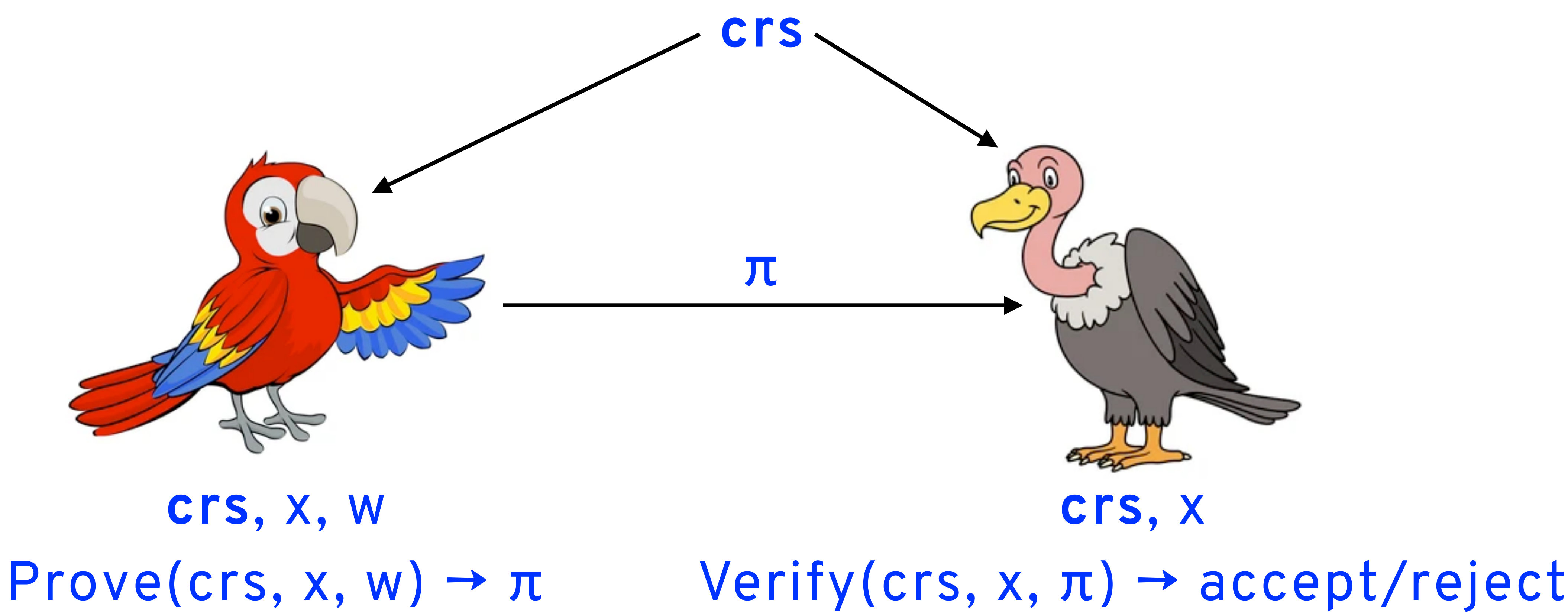
Prove(x, w) → π

Verify(x, π) → accept/reject

Soundness: hard for the prover to convince the verifier if $x \notin L_R$

Zero knowledge: the verifier learns nothing except that $x \in L_R$

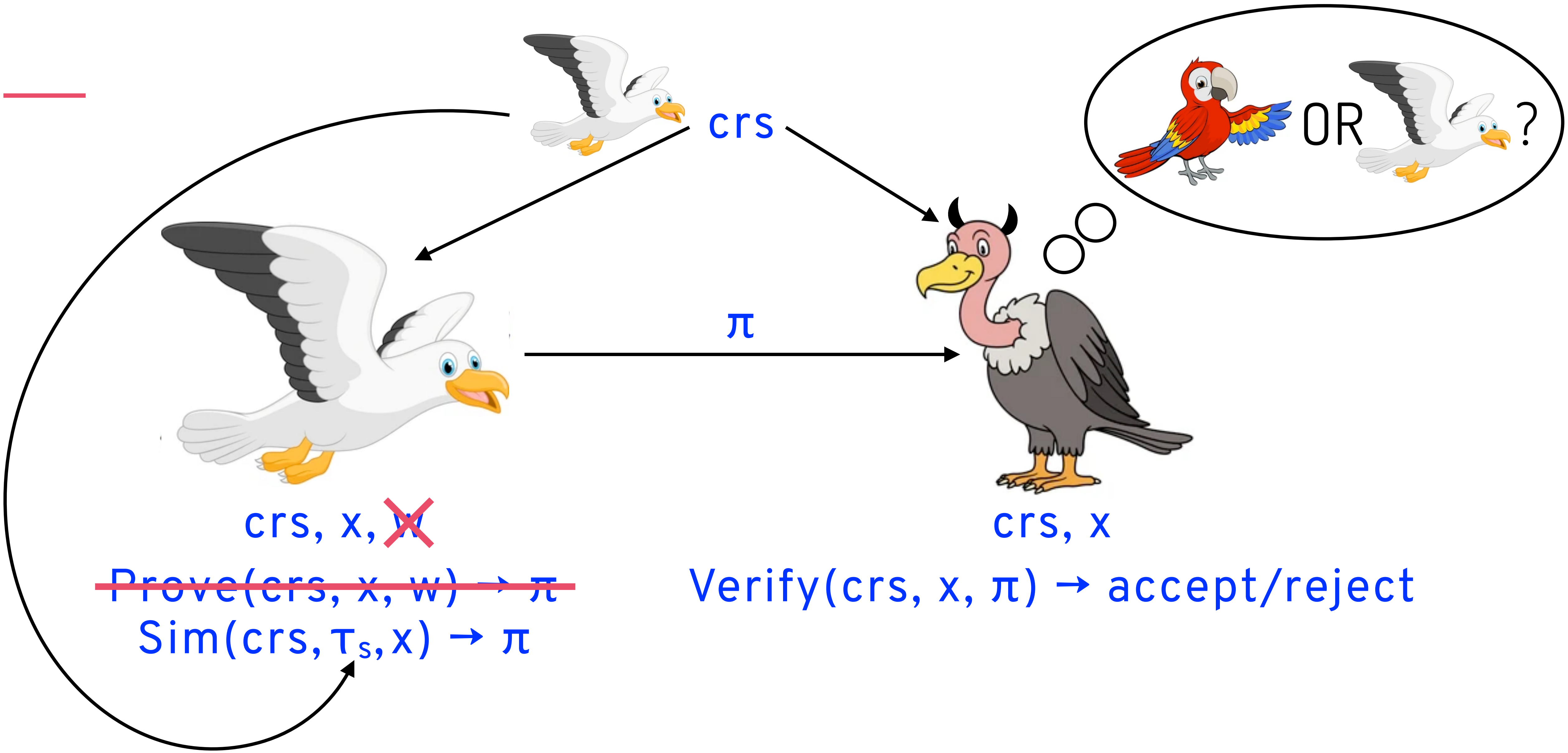
DEFINING ZERO KNOWLEDGE



This **common reference string** needs to exist [GO94]; can be

- random (trustless setup) or structured (trusted setup)
- specific to a given relation or universal

A ZERO-KNOWLEDGE SIMULATOR



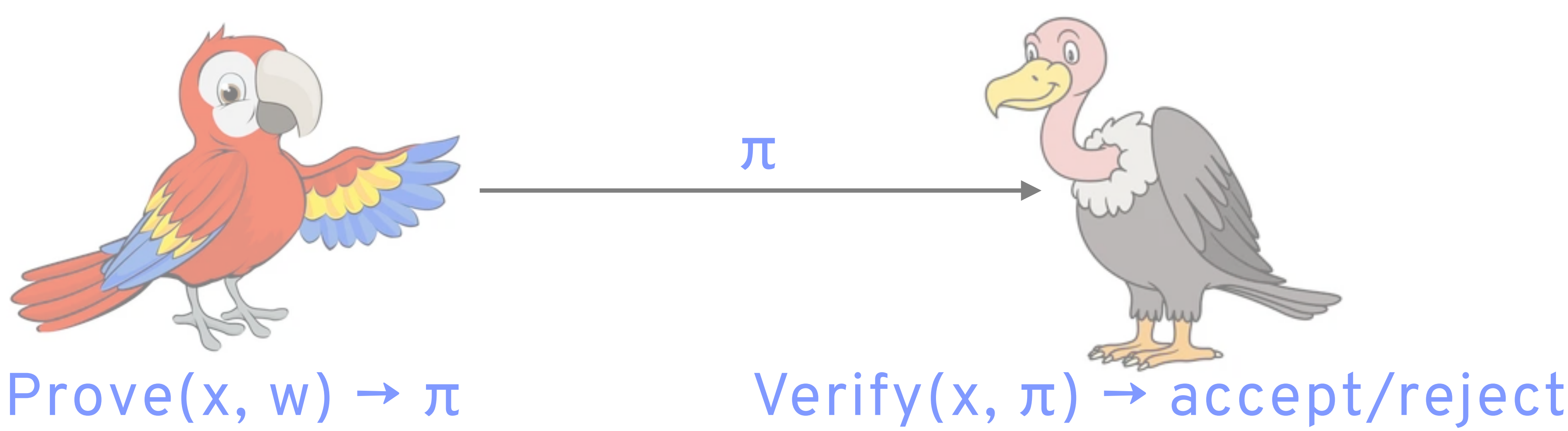
Zero knowledge: the verifier can’t tell if it’s interacting with the prover or with a simulator (who doesn’t know a witness)

- Perfect zero knowledge if the distributions are identical (not just indistinguishable)

DEFINING ZERO KNOWLEDGE

In a zero-knowledge proof [GMR89], a prover wants to convince a verifier that there exists a **witness** w corresponding to some **instance** x of a language L_R (witness w for the **statement** $(x,w)\in R$)

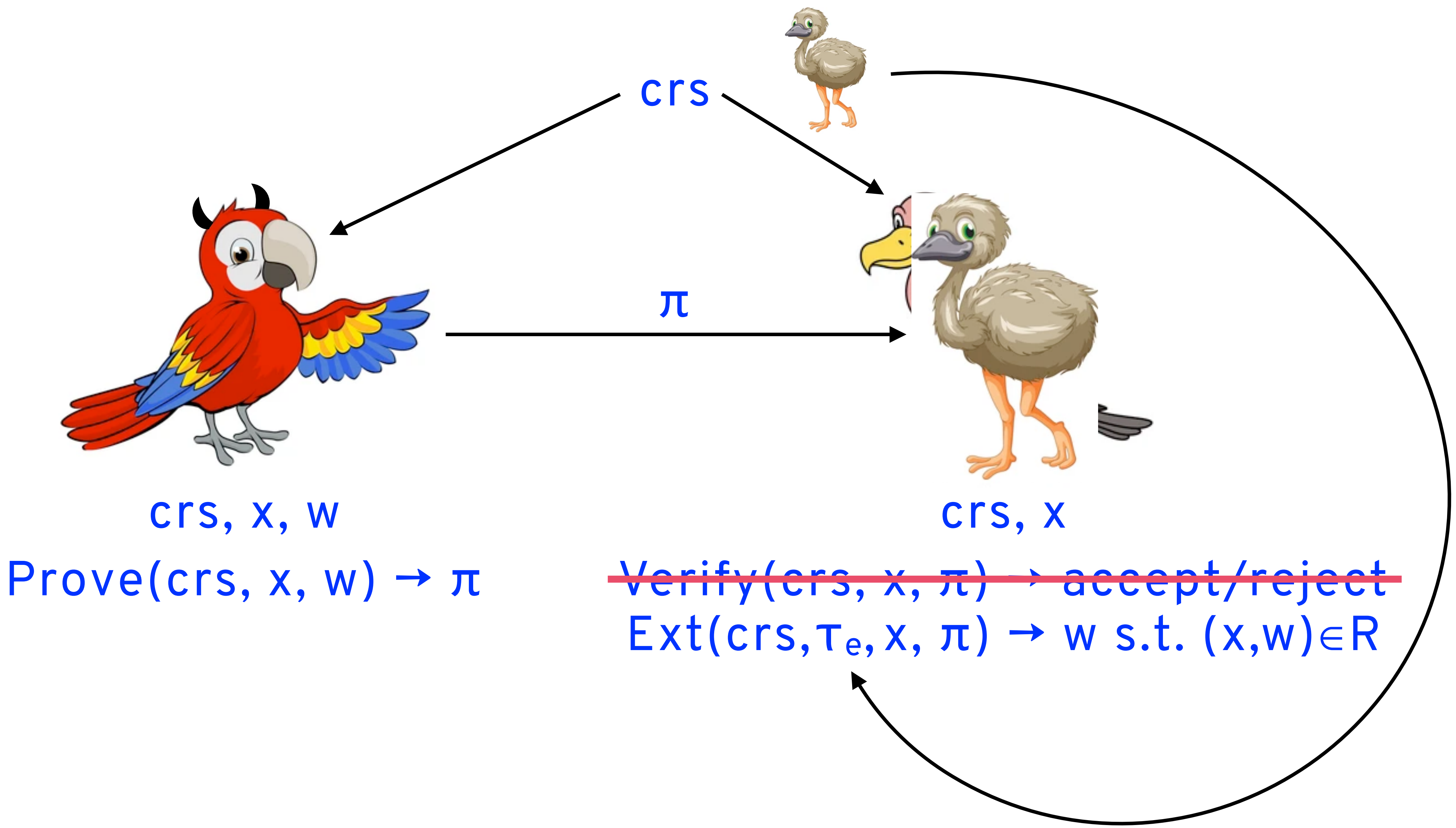
In a non-interactive zero-knowledge proof (NIZK) [BFM88], this is done without any interaction



Soundness: hard for the prover to convince the verifier if $x\notin L_R$

Zero knowledge: the verifier learns nothing except that $x\in L_R$

EXTRACTABILITY



Extractability: there exists a PT extractor that can do this...

- ...for all provers (proof of knowledge)
- ...for all PPT provers (argument of knowledge)

PROTOCOLS AND PROOF SIZES

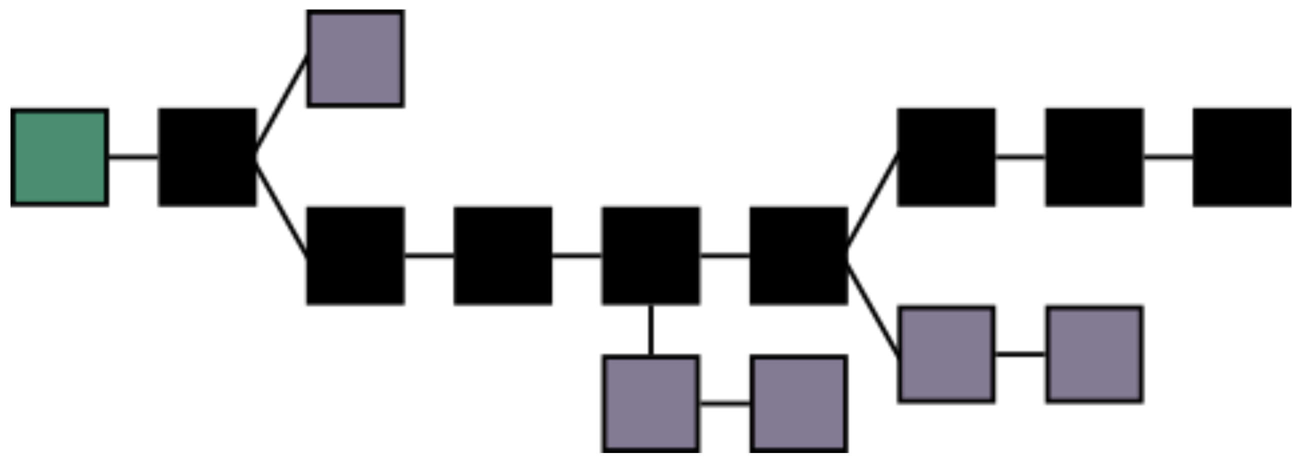
	Linear	Polylog
DLOG	CD97	
Pairings	GS08	
CRHFs		Kilian92

PROTOCOLS AND PROOF SIZES

	Linear	Sublinear	Polylog
DLOG	CD97	Groth09	SUCCINCT NON-INTERACTIVE ARGUMENTS (**SNARGS**) OF KNOWLEDGE (**SNARKS**)
Pairings	GS08	Groth10	
CRHFs			Kilian92

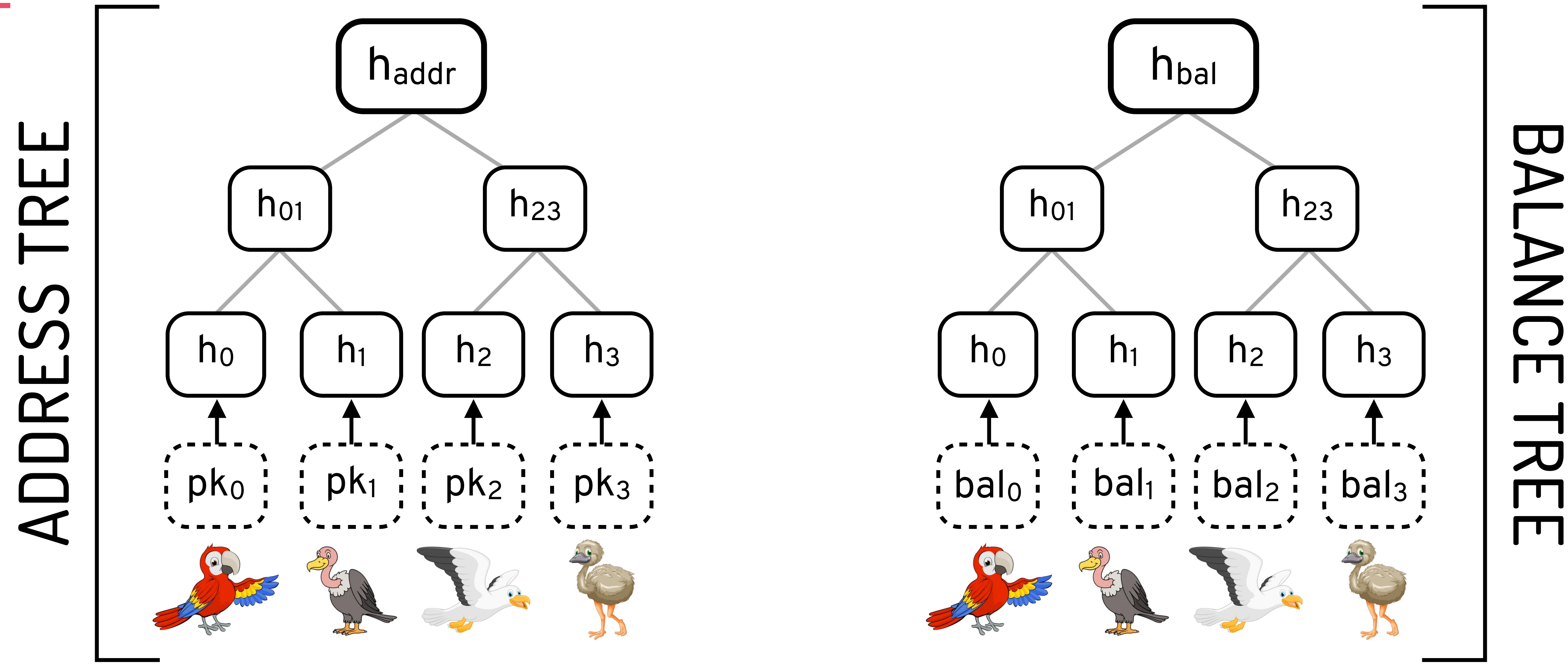
BLOCKCHAIN BASICS

A **blockchain** is an ordered collection of **transactions**



All transactions in the chain are replayed by all peers (**full nodes**) in a network to ensure they agree on its current **state**

MAINTAINING STATE

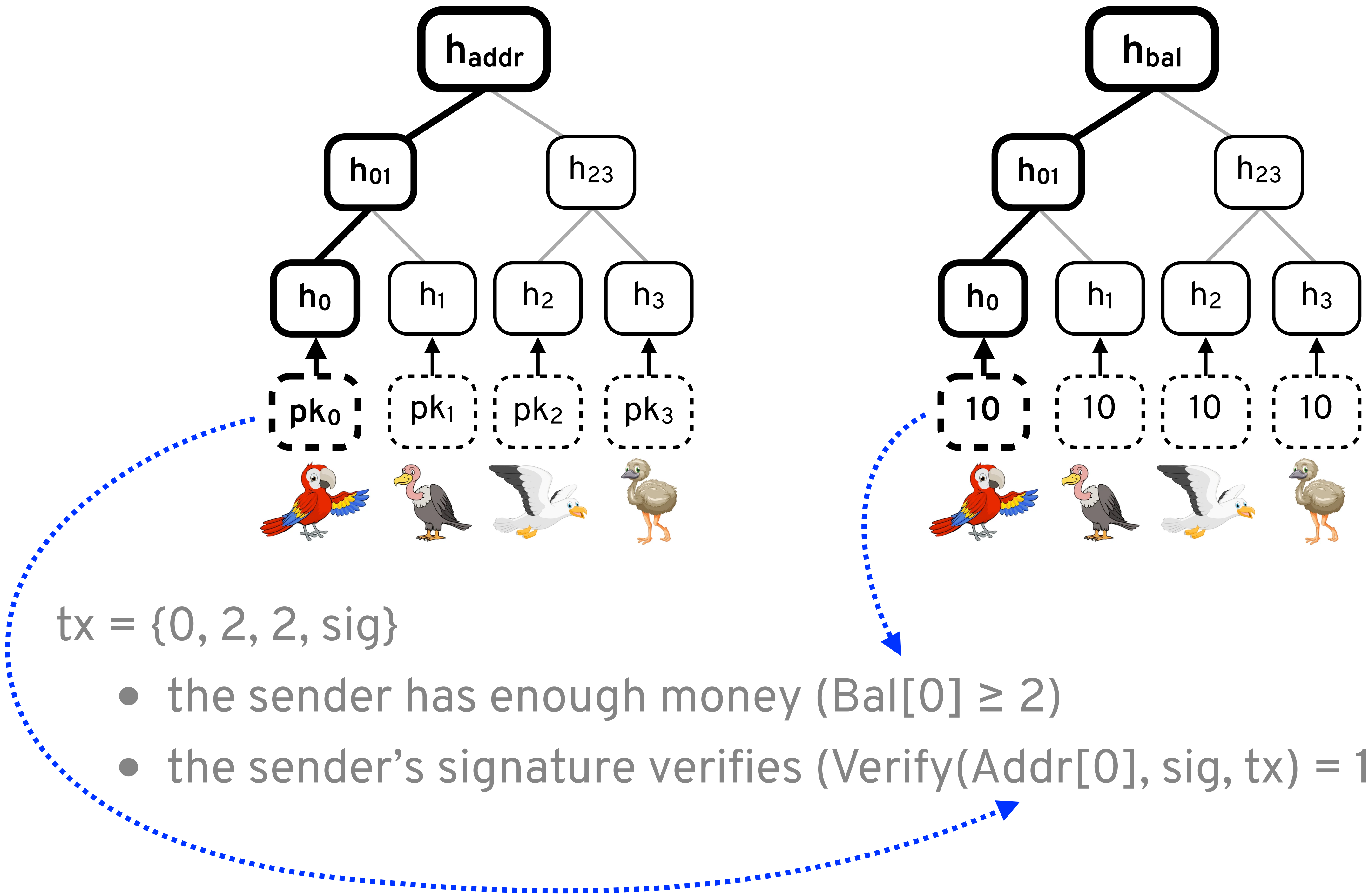


tx = {i_{from}, i_{to}, amt, sig} is **valid** if

- the sender has enough money (Bal[i_{from}] ≥ amt)
- the sender's signature verifies (Verify(Addr[i_{from}], sig, tx) = 1)

Can **process** tx(Bal): Bal[i_{from}] -= amt and Bal[i_{to}] += amt

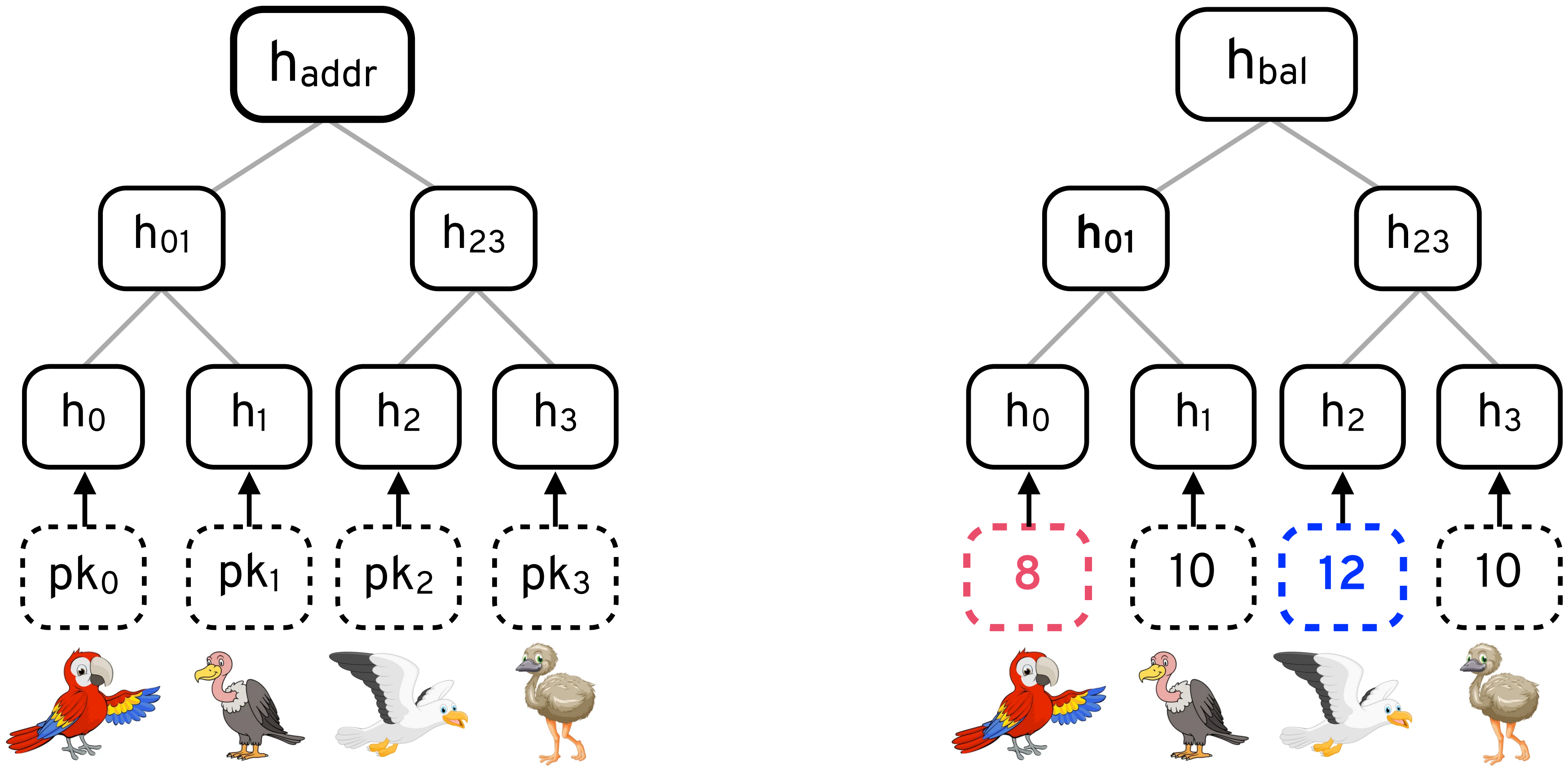
CHECKING TRANSACTION VALIDITY



tx = {0, 2, 2, sig}

- the sender has enough money (Bal[0] ≥ 2)
- the sender's signature verifies (Verify(Addr[0], sig, tx) = 1)

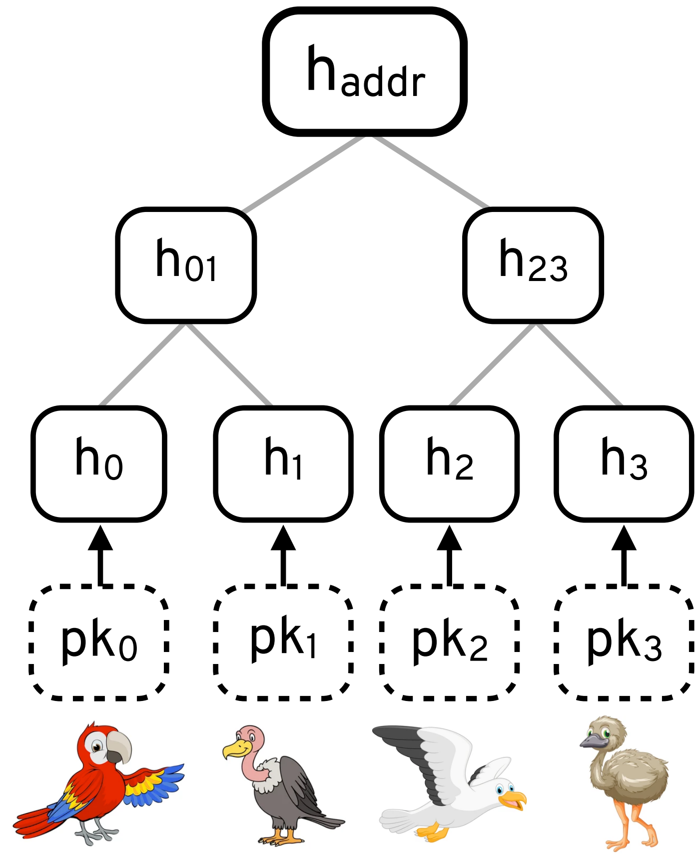
PROCESSING TRANSACTIONS

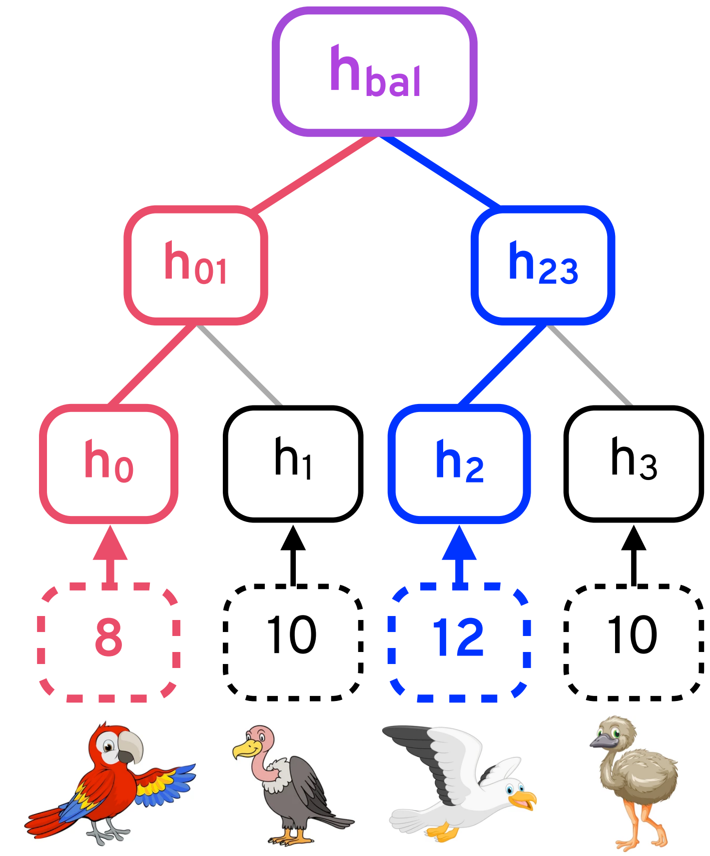


tx = $\{i_P, i_S, 2, sig\}$

- $Bal[i_{from}]$ -= amt and $Bal[i_{to}]$ += amt

PROCESSING TRANSACTIONS

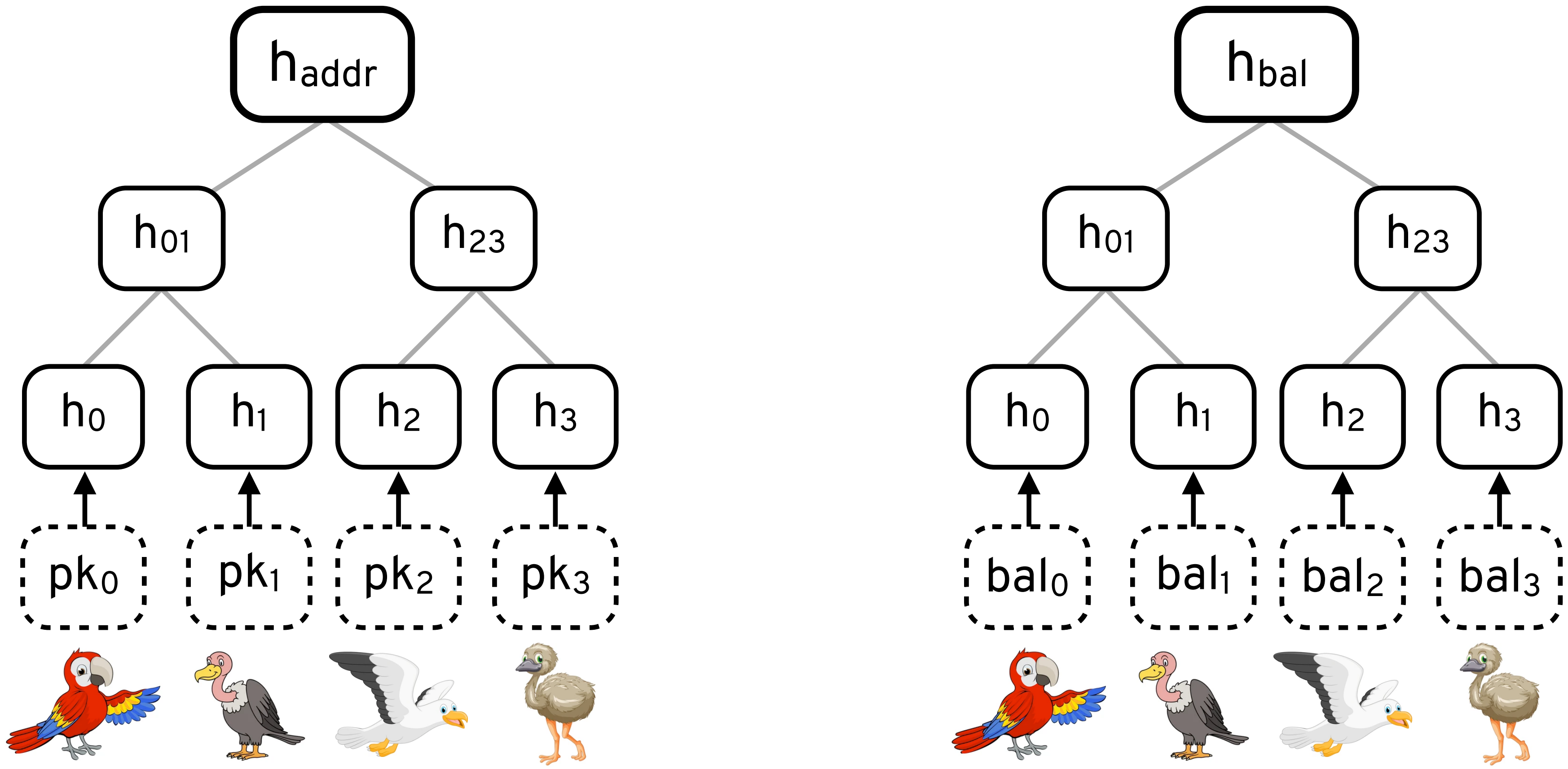




tx = {i_P, i_S, 2, sig}

- Bal[i_{from}] -= amt and Bal[i_{to}] += amt
- Bal changes, so its root changes from h_{bal} to h_{bal}

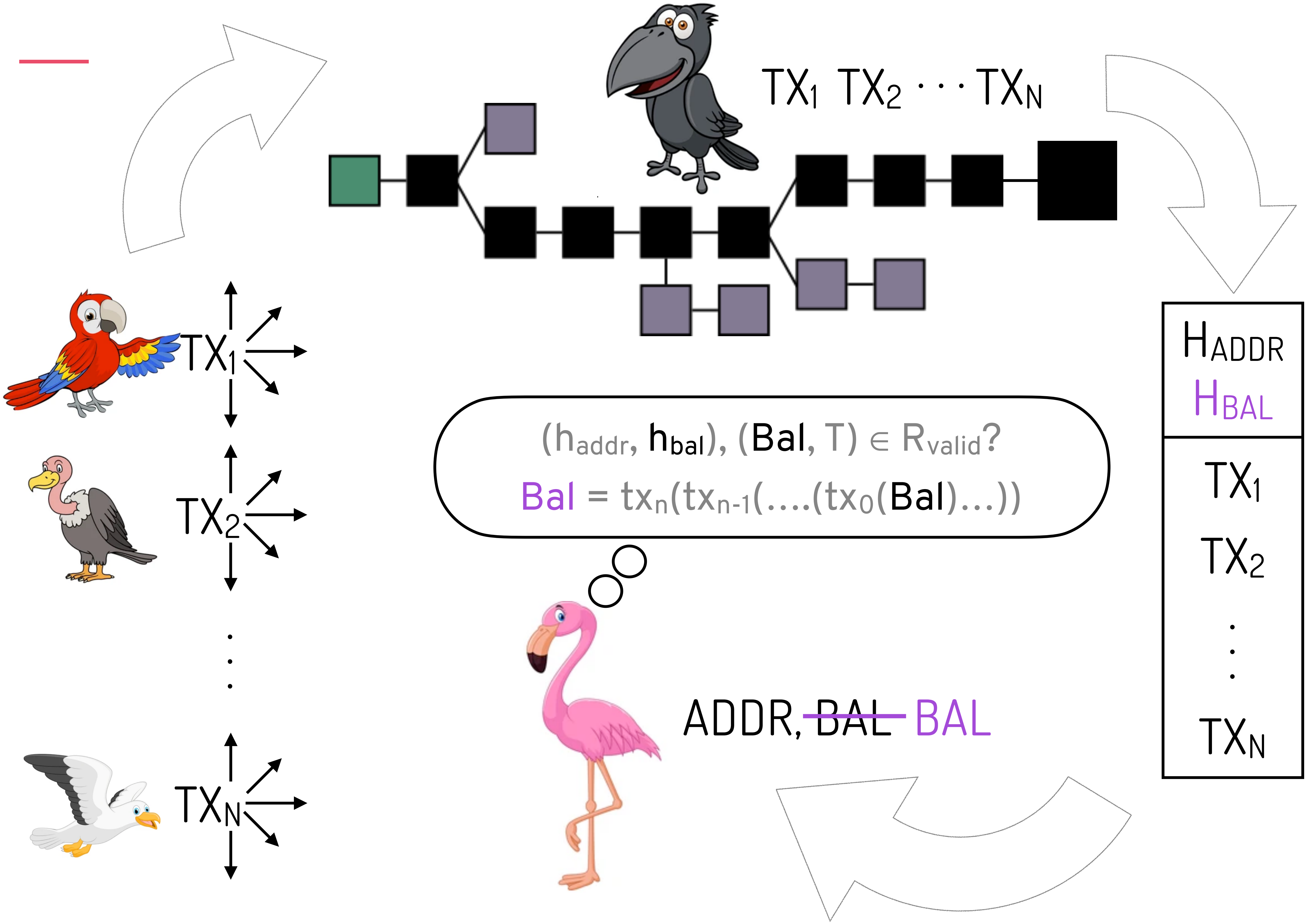
MAINTAINING STATE



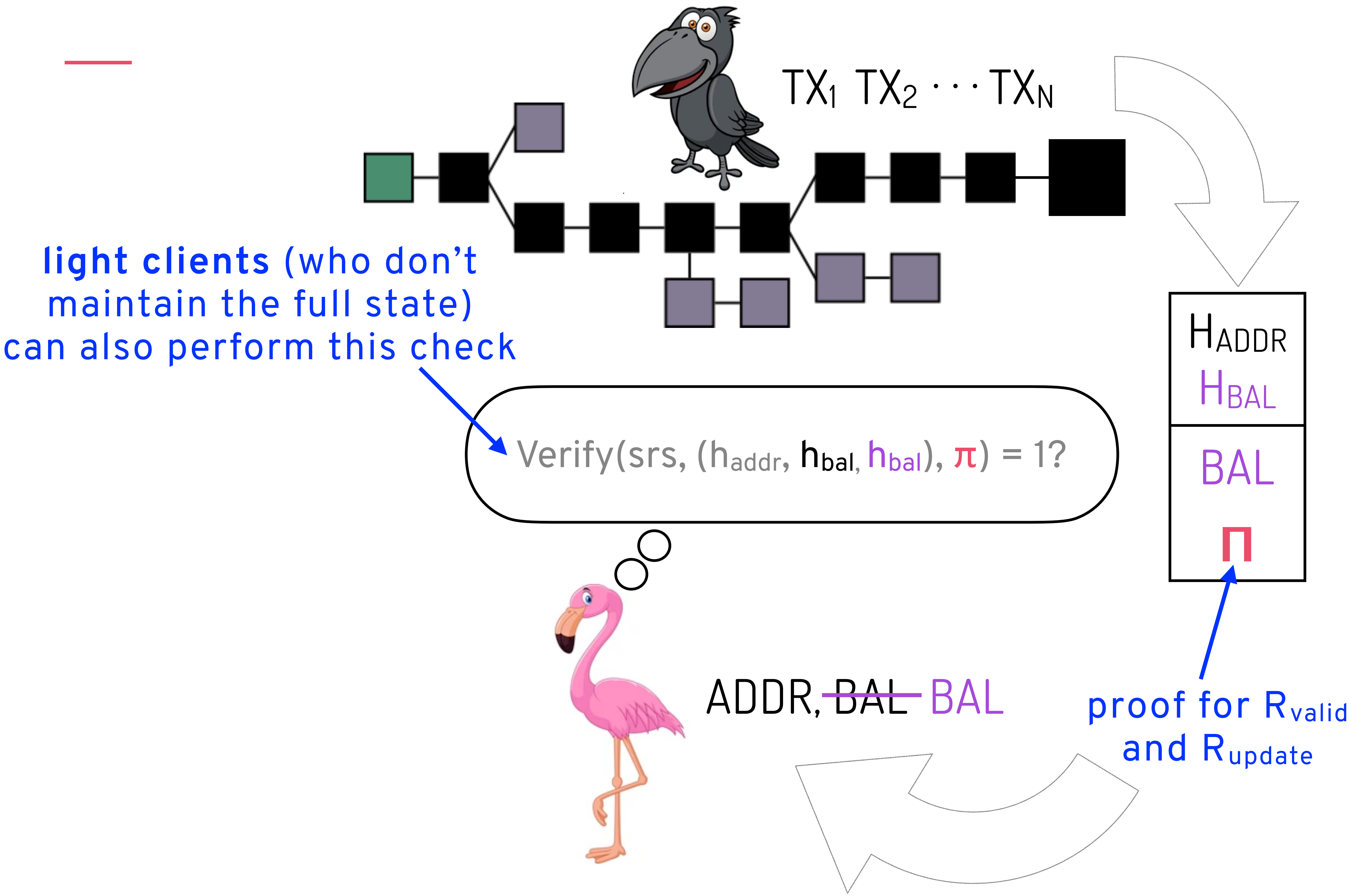
$x = (h_{addr}, h_{bal})$, $w = (Bal, T) \in R_{valid} \Leftrightarrow$ (1) $h_{bal} = root(Bal)$ (correct root) and (2) all txs in T are valid (according to h_{addr}) (valid transactions)

$x = (h_{addr}, h_{bal}, h_{bal})$, $w = (Bal, Bal, T) \in R_{update} \Leftrightarrow$ (1) $h_{bal} = root(Bal)$ and $h_{bal} = root(Bal)$ (correct roots) and (2) $Bal = tx_n(tx_{n-1}(....(tx_0(Bal)...))$ (correct state update)

UPDATING GLOBAL STATE

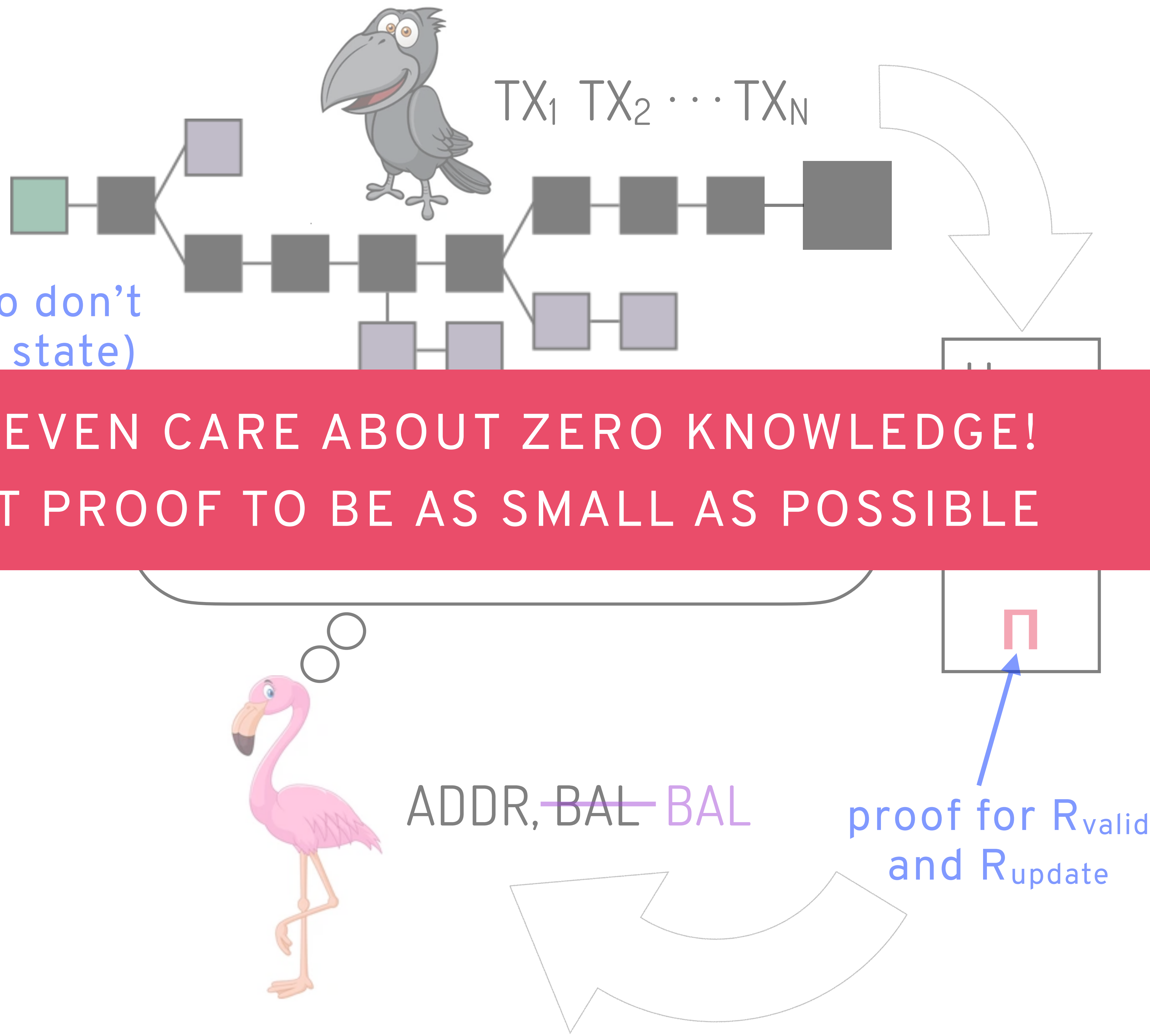


ZK-ROLLUPS


TX_1 TX_2 ··· TX_N
light clients (who don't maintain the full state) can also perform this check
Verify(srs, (h_{addr}, h_{bal}, h_{bal}), π) = 1?
H_{ADDR}
H_{BAL}
BAL
π
ADDR, ~~BAL~~ BAL
proof for R_{valid} and R_{update}

ZK-ROLLUPS

light clients (who don't maintain the full state)



WE DON'T EVEN CARE ABOUT ZERO KNOWLEDGE!

JUST WANT PROOF TO BE AS SMALL AS POSSIBLE

PROTOCOLS AND PROOF SIZES

	Linear	Sublinear	Polylog	Constant
DLOG	CD97	Groth09, Hyrax	BCC+16, Bulletproofs	Halo
Pairings	GS08	Groth10	LMR19 Libra	*SNARKs* [GGPR13], Groth16
Lattices				*DV SNARKs* BISW17,18 GMNO18
CRHFs	ZKBoo	Ligero	Kilian92, *STARKs* Aurora Fractal	

STATE OF THE ART (GROTH'16) HAS 3 GROUP ELEMENTS AND REQUIRES 3 PAIRINGS TO VERIFY

SNARKS + BLOCKCHAINS

Having small proofs that can be verified quickly is really useful for agreeing on a shared state in a scalable way

But, these proofs have their costs

- Substantial prover runtime [BCL20, BCG20, GKR+21]
- Known constant-sized SNARKs require a structured reference string (SRS), which means relying on trusted third parties

PROVER RUNTIME

The number of constraints for a proof system involving hashes depends hugely on the hash function

PROVING KNOWLEDGE OF X SUCH THAT H(X) = Y

SHA256

```
Compiling sha.pok

Compiled code written to 'out'
Number of constraints: 48946
```

PEDERSEN

```
Compiling pedersen.pok

Compiled code written to 'out'
Number of constraints: 3940
```

POSEIDON [GKR+21]

```
Compiling poseidon.pok

Compiled code written to 'out'
Number of constraints: 298
```

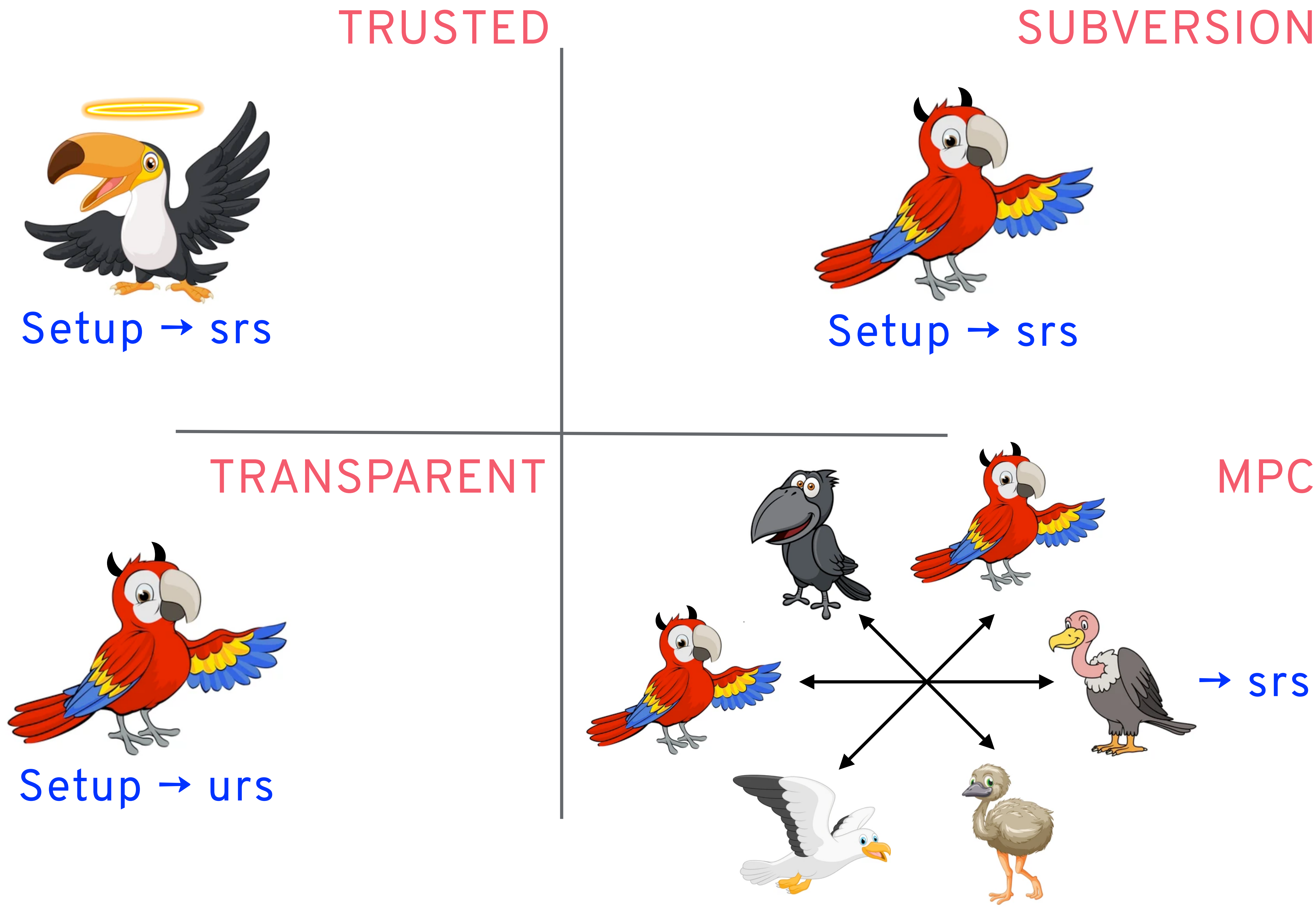
SNARKS + BLOCKCHAINS

Having small proofs that can be verified quickly is really useful for agreeing on a shared state in a scalable way

But, these proofs have their costs

- Substantial prover runtime [BCL20, BCG20, GKR+21]
- Known constant-sized SNARKs require a structured reference string (SRS), which means relying on trusted third parties

GENERATING A REFERENCE STRING



REFERENCE STRING GENERATION

~~Setup → srs~~

Setup → (srs, τ)

In many known systems, Setup also outputs a simulation trapdoor

REFERENCE STRING GENERATION

~~Setup → srs~~

Setup → (srs, τ)

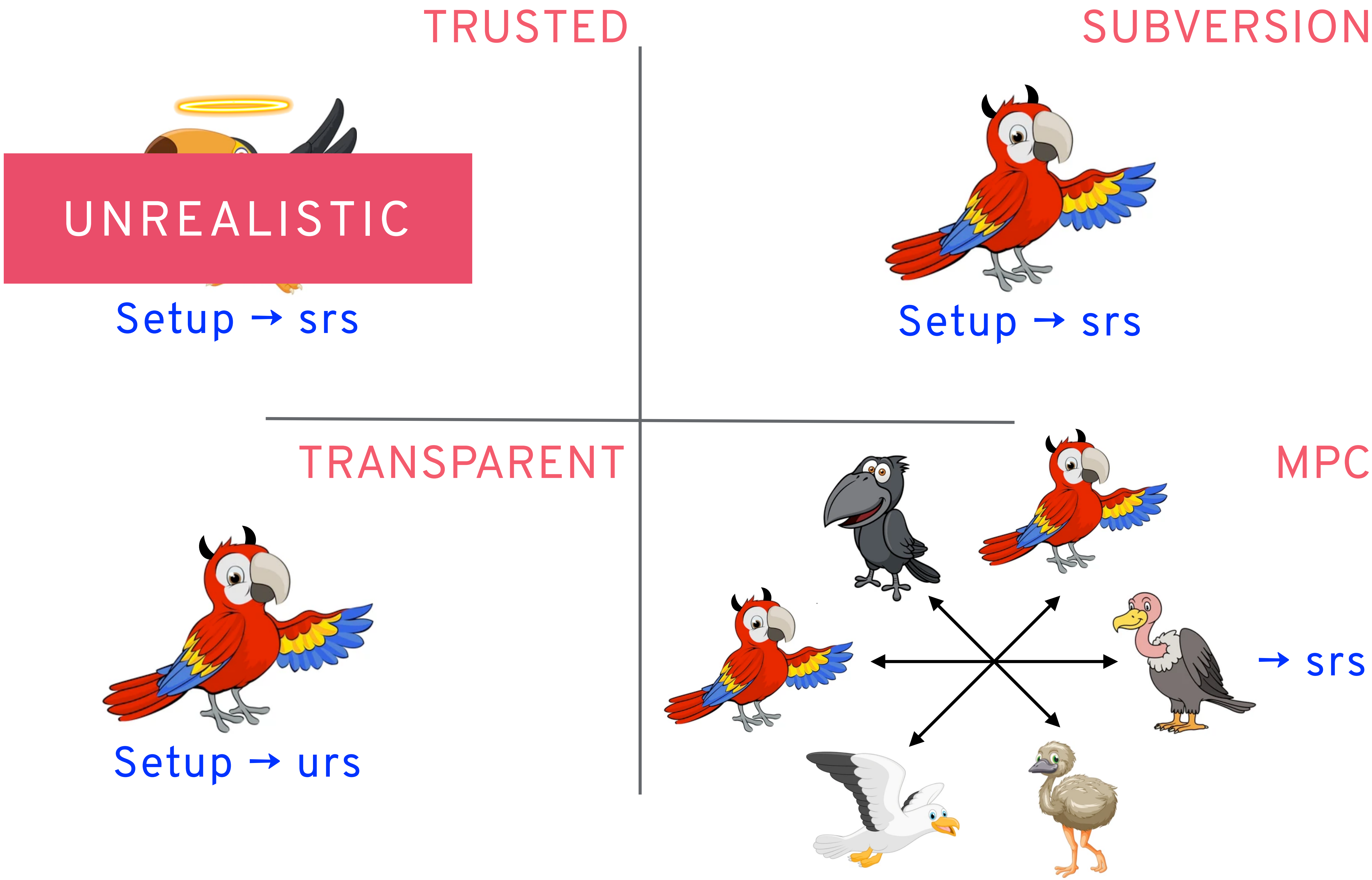
In many known systems, Setup also outputs a simulation trapdoor

Example: for $srs = (g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^q})$, $\tau = \alpha$

If a party knows τ, they can provide proofs of false statements

In a cryptocurrency setting (like Zcash), this would allow this party to spend coins they don't have

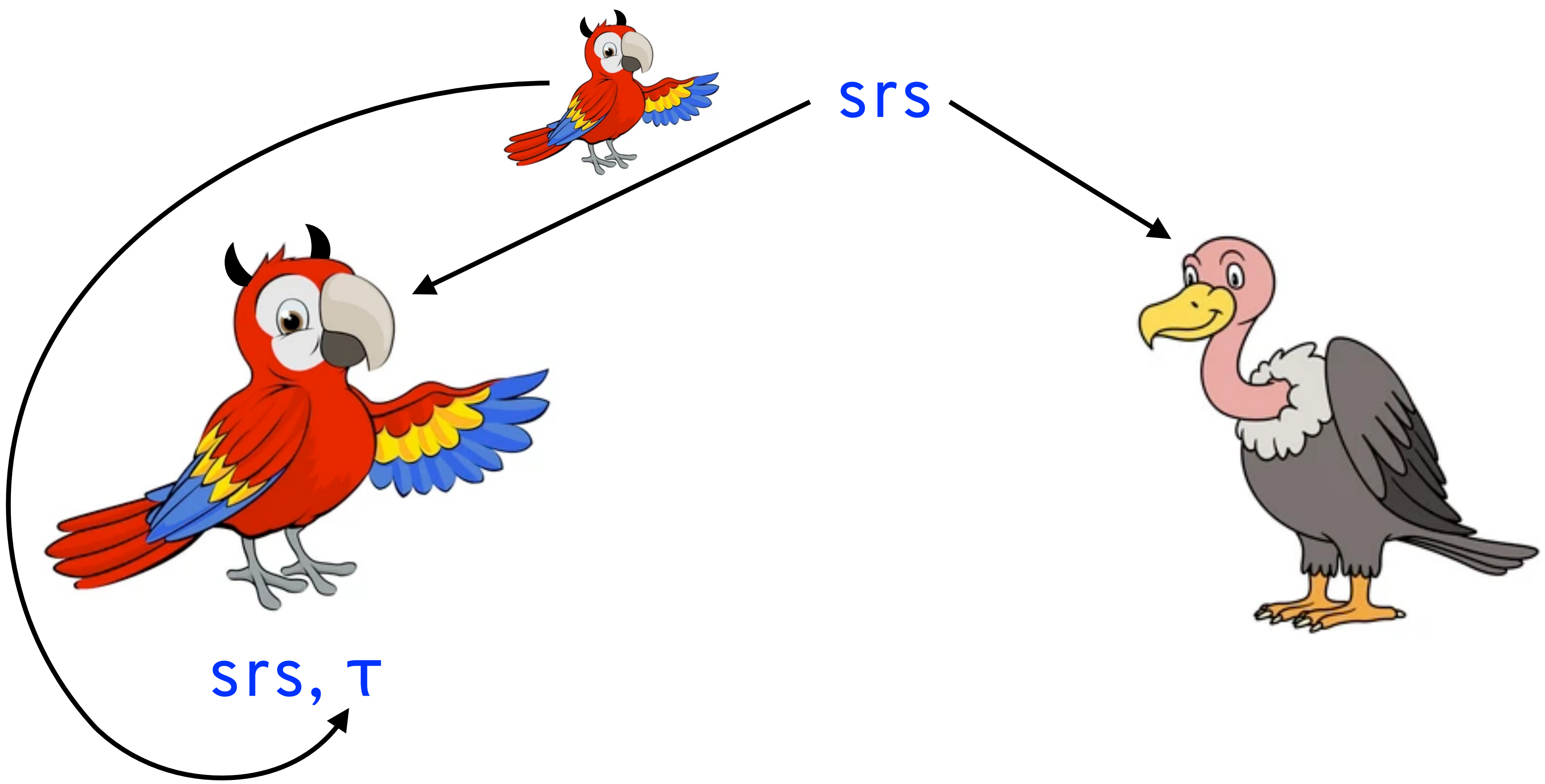
GENERATING A REFERENCE STRING



SUBVERSION [BFS16]

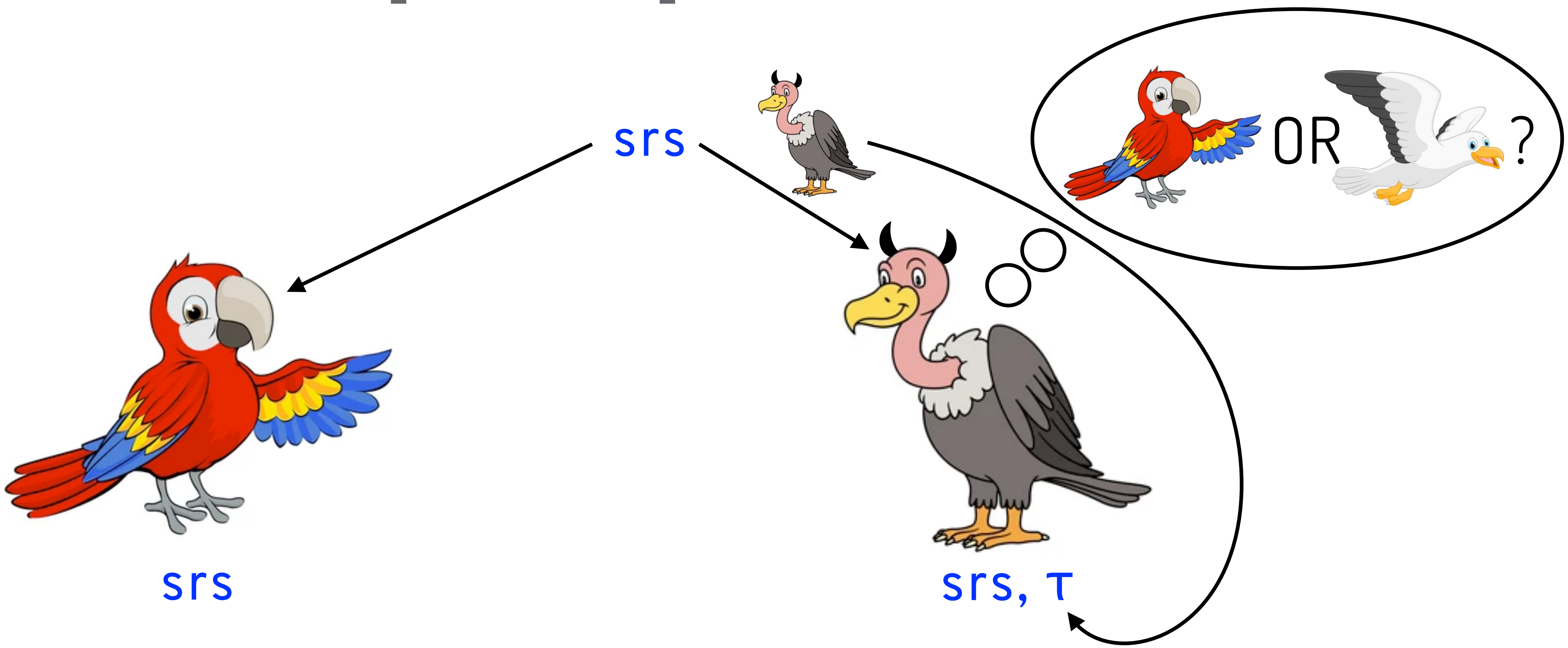
Subverting the reference string was considered by Bellare, Fuchsbauer, and Scafuro in 2016

SUBVERSION [BFS16]



Subversion soundness (S-SND): the prover can't prove false statements *even if it generated the SRS*

SUBVERSION [BFS16]



Subversion soundness (S-SND): the prover can't prove false statements *even if it generated the SRS*

Subversion zero knowledge (S-ZK): the verifier can't tell if it's interacting with the prover or with a simulator, *even if it generated the SRS*

SUBVERSION [BFS16]

Subverting the reference string was first considered by Bellare, Fuchsbauer, and Scafuro in 2016

They showed that:

- S-SND and (normal) ZK cannot be achieved (following [GO94])
- S-SND and S-WI can be achieved
- S-ZK and (normal) SND can be achieved

GENERATING A REFERENCE STRING

TRUSTED

UNREALISTIC

Setup → srs

SUBVERSION

IMPOSSIBLE :(

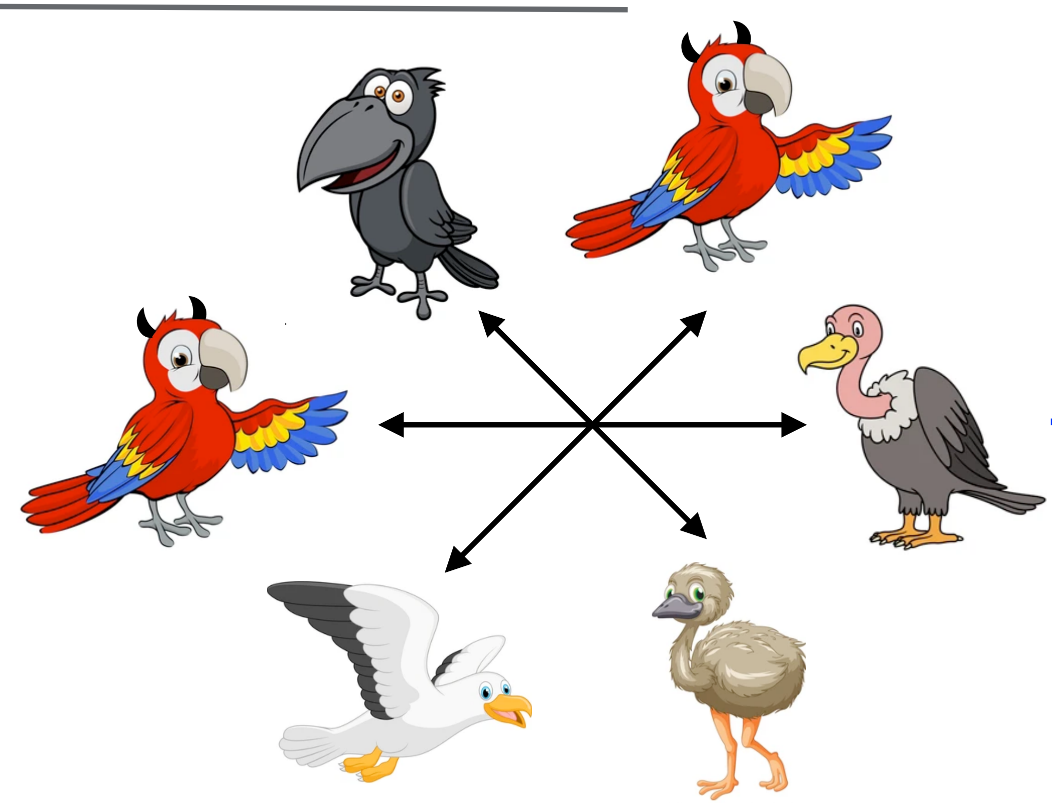
Setup → srs

TRANSPARENT

STARKS

Setup → urs

MPC



→ srs

SETUP VIA MULTI-PARTY COMPUTATION

Researchers have developed optimized MPC protocols for generating structured reference strings for various SNARKs:

- Pinocchio [BGG17]
- Groth16 [BCG+15, BGM17, ABL+19, KMSV21]

These protocols have been run in practice in complex **ceremonies** (for Zcash, Aztec, Filecoin, etc.)

If the SRS is not **universal** though, the ceremony needs to be re-run every time the protocol changes

- Zcash Sprout ceremony in 2016 (6 participants)
- Zcash Sapling ceremony in 2018 (87 participants)

UPDATABILITY [GKM**M**M18, MBK**M**19]

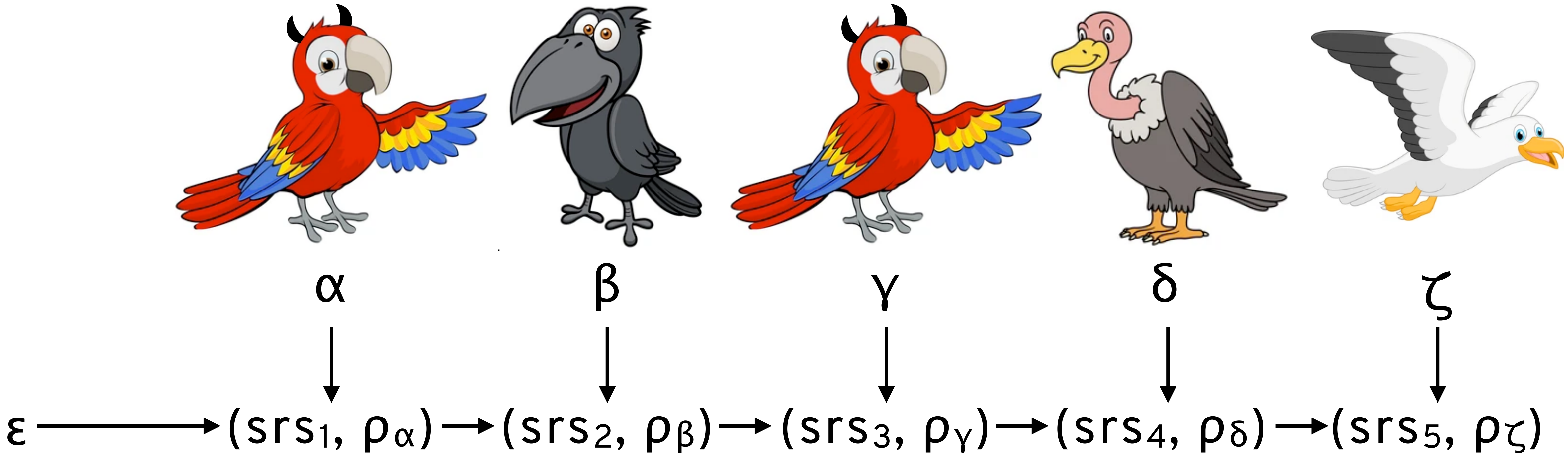
α

$\varepsilon \rightarrow (srs_1, \rho_\alpha)$

update proof: proof that srs_1 uses additional randomness α (can be publicly verified)

universal SRS from which circuit-specific SRSs can be derived

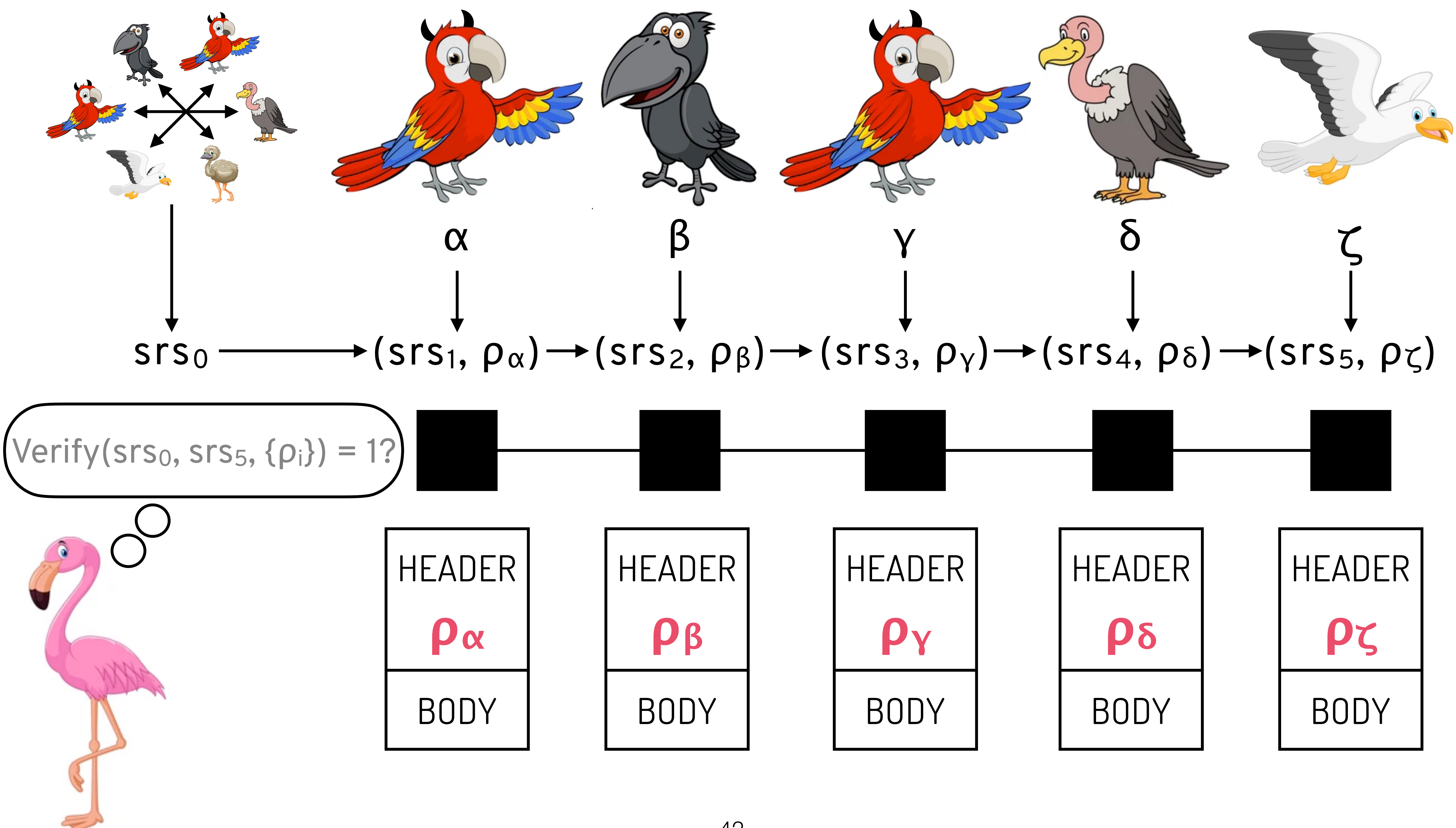
UPDATABILITY [GKM**M**M18, MBK**M**19]



No one know the trapdoor $\alpha \oplus \beta \oplus \gamma \oplus \delta \oplus \zeta$ of srs_5 if at least one party is honest

The set of parties is not fixed and the process doesn’t have an end: a new party can come contribute randomness any time they want

UPDATABILITY [GKM**M**M18, MBK**M**19]

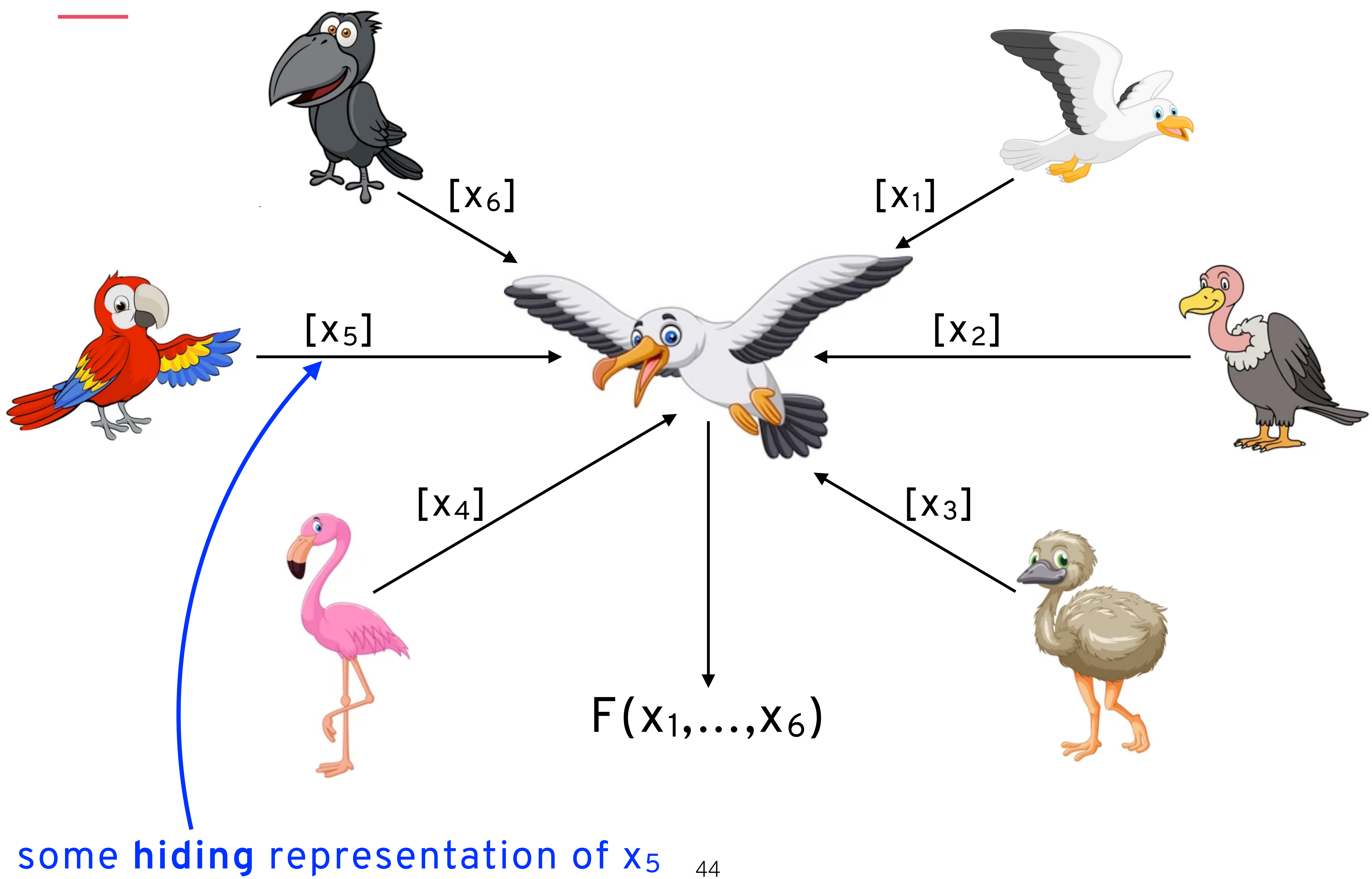


PROTOCOLS AND PROOF SIZES

	Linear	Sublinear	Polylog	Constant
DLOG	CD97	Groth09	BCC+16 Hyrax AC20	Spartan Halo Bulletproofs
Pairings	GS08	Groth10	LMR19 Libra	*SNARKs* [GGPR13] Sonic Plonk Groth16 Marlin
Lattices	BKLP15 BCS21	BBC+18	BLNS20	*DV SNARKs* BISW17,18 GMNO18
CRHFs	ZKBoo BCG+17	Ligero BCG20	Kilian92	*STARKs* Aurora Fractal

Thanks to Jonathan Bootle for the original version of this slide!

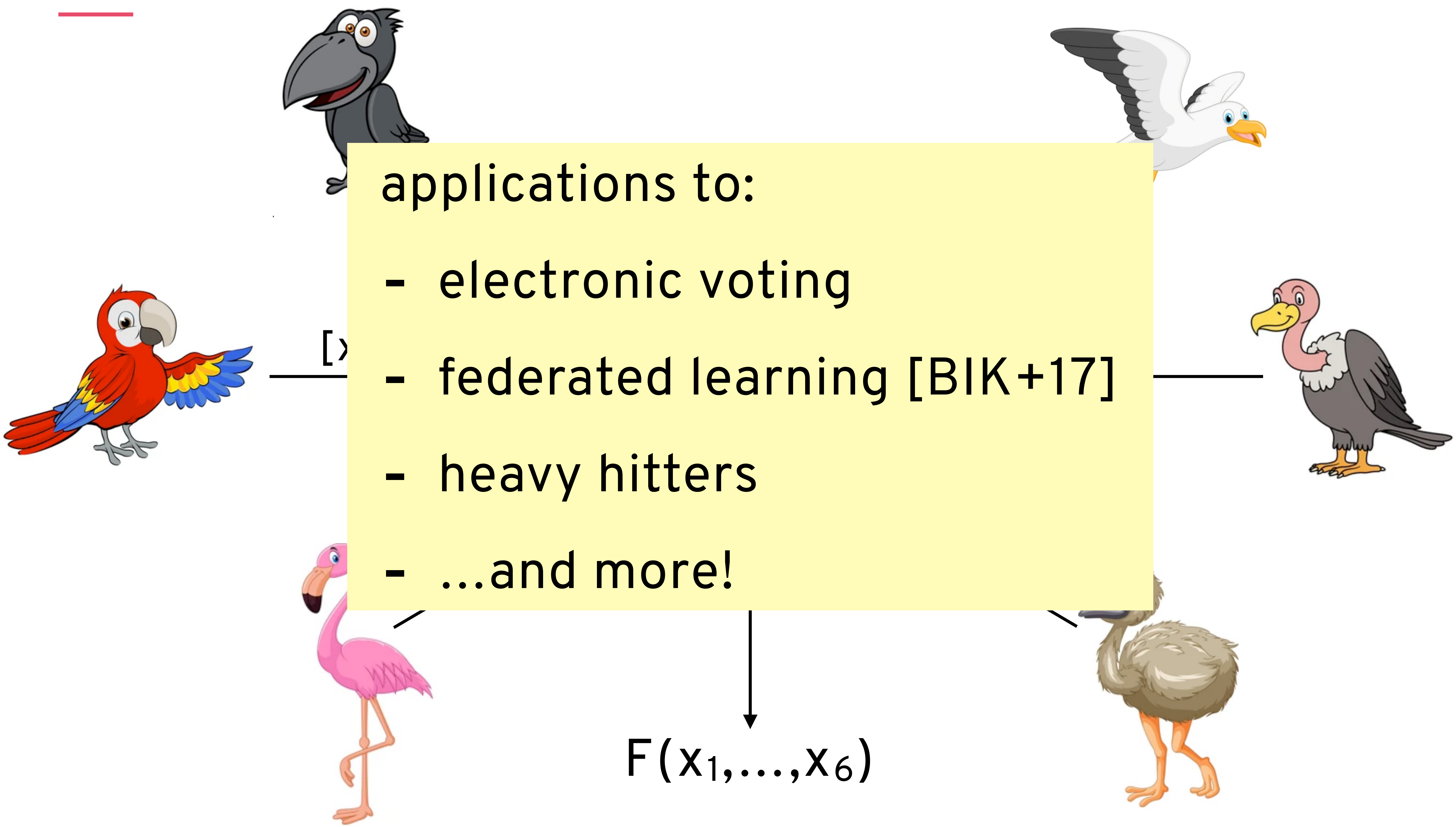
PRIVATE AGGREGATE COMPUTATIONS



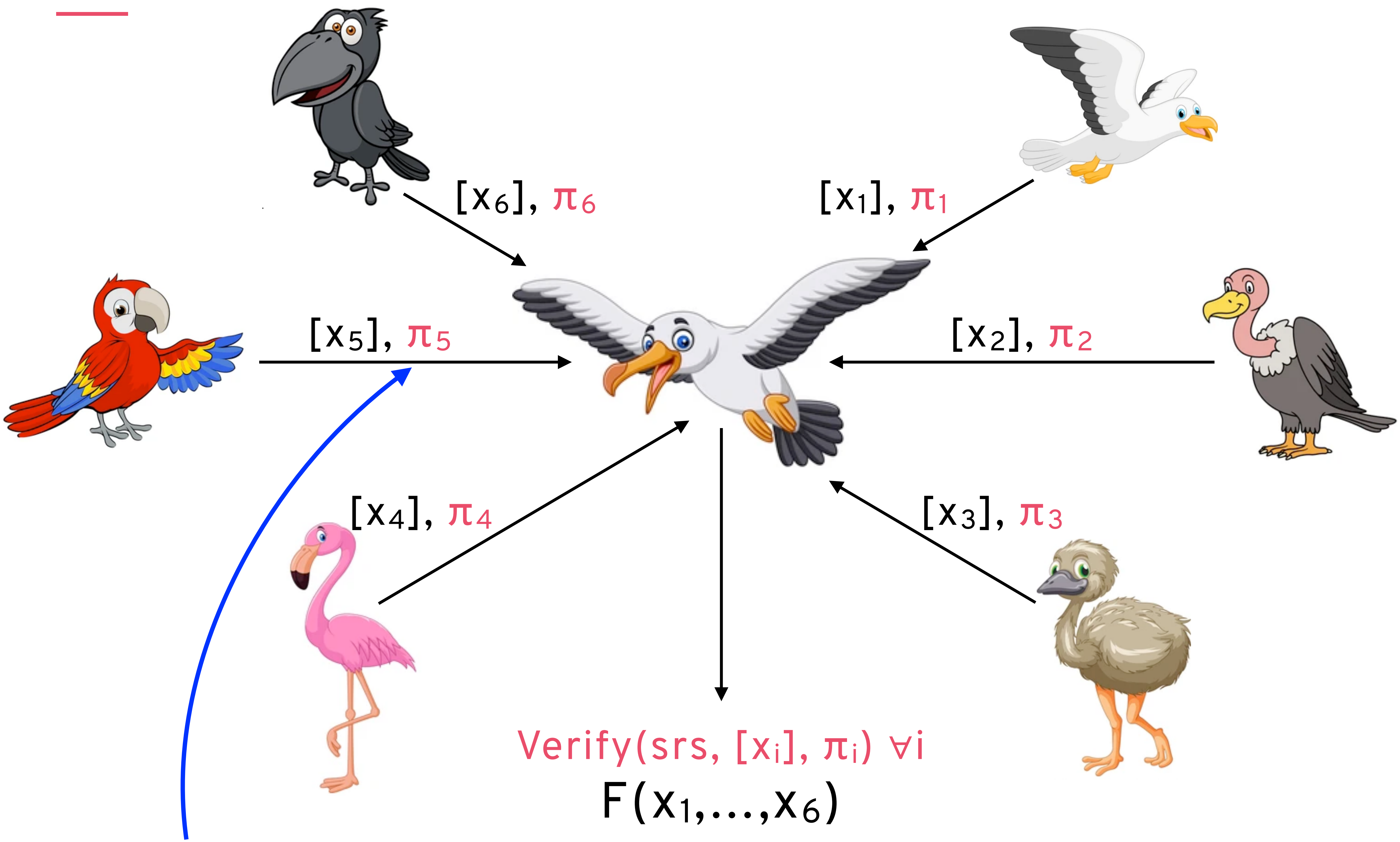
PRIVATE AGGREGATE COMPUTATIONS

applications to:

- electronic voting
- federated learning [BIK+17]
- heavy hitters
- ...and more!

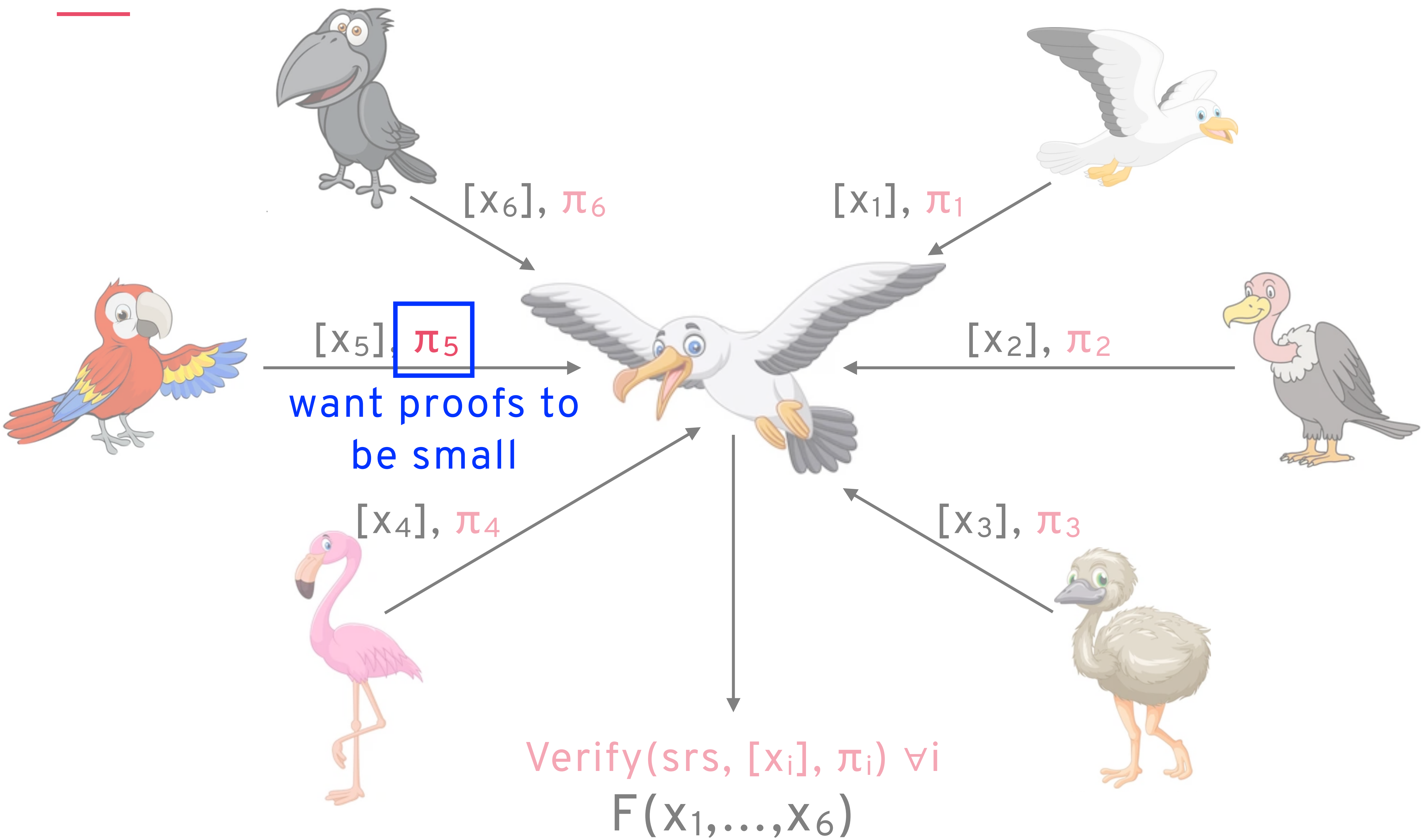


PRIVATE AGGREGATE COMPUTATIONS

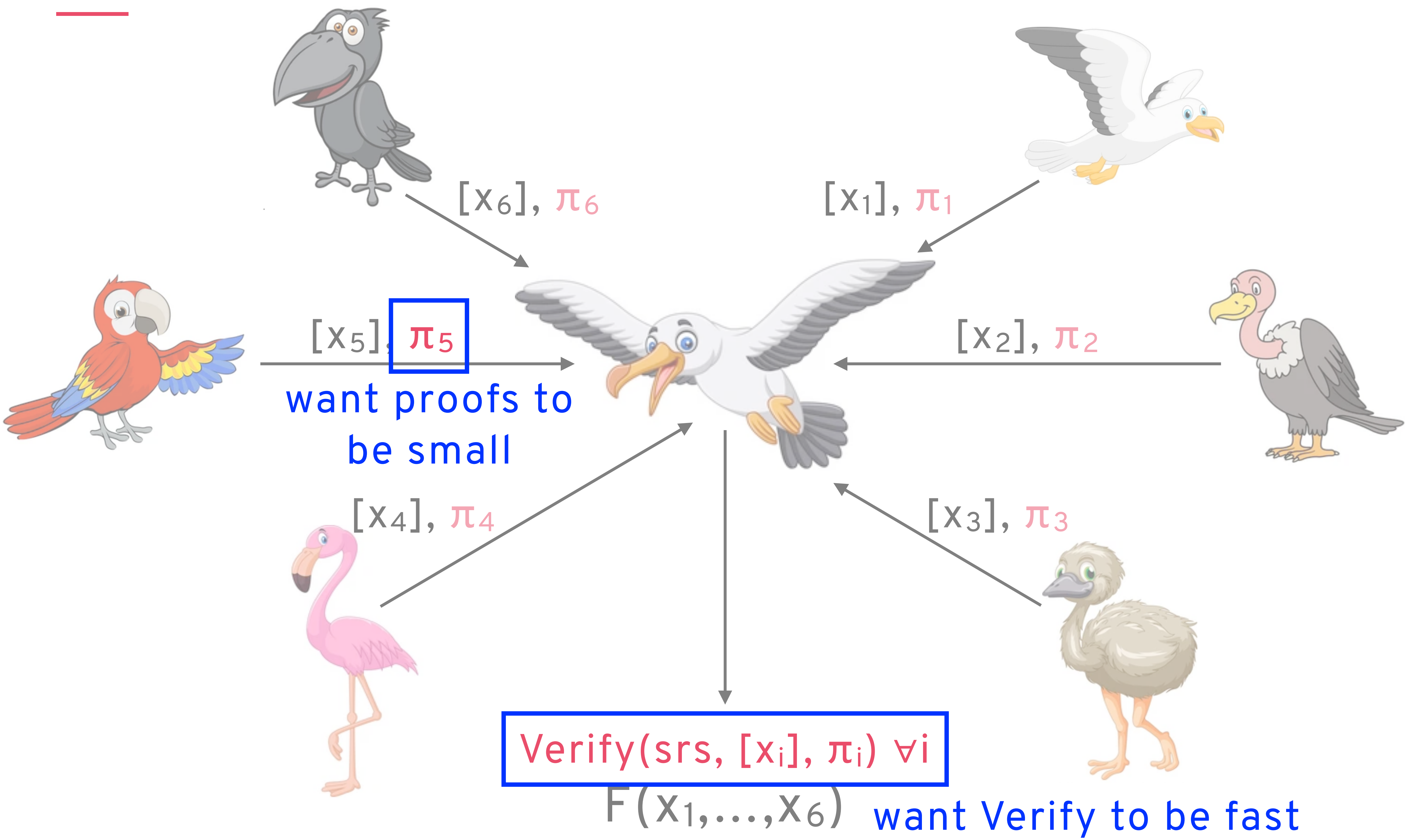


a proof that x_5 has the right form

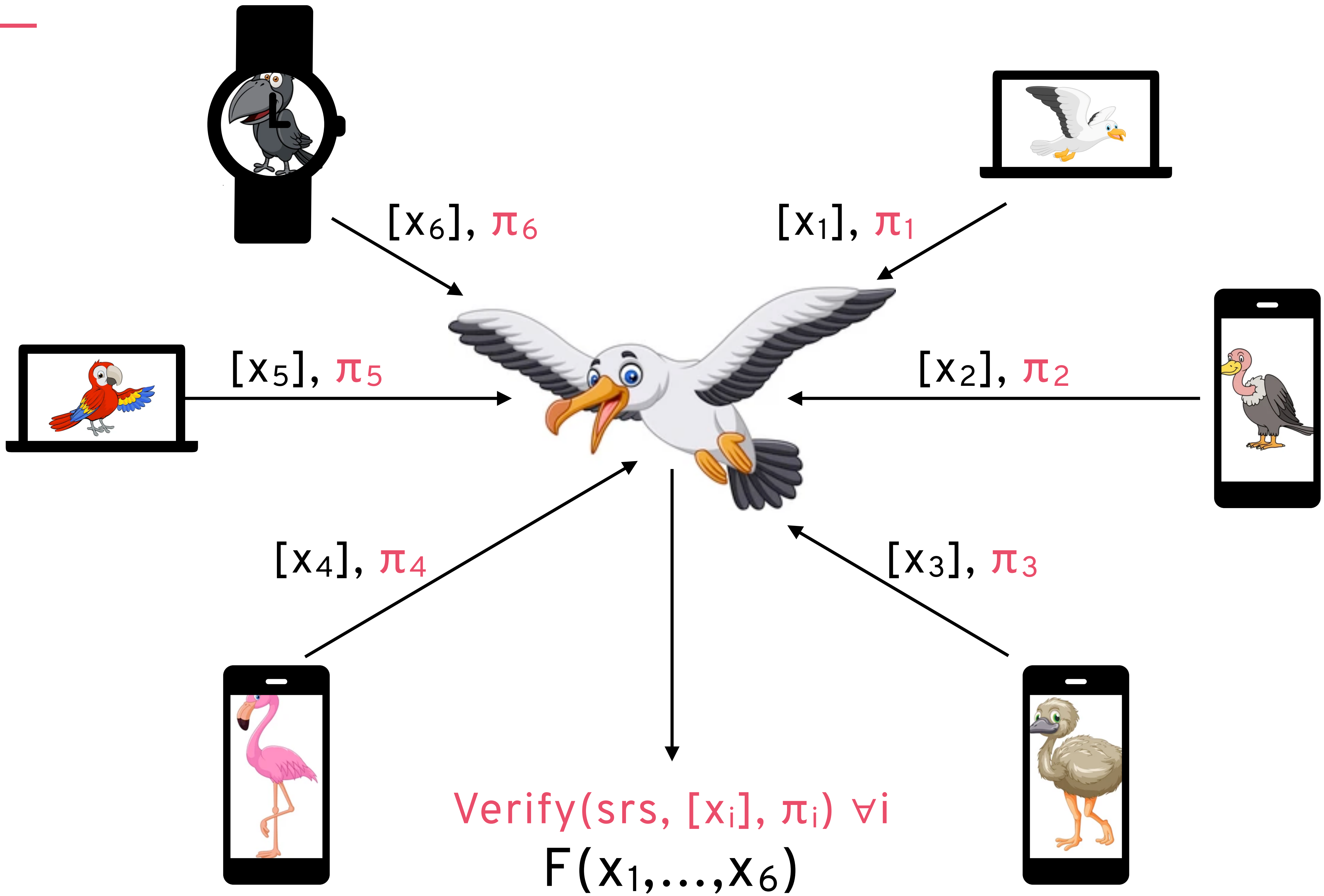
PRIVATE AGGREGATE COMPUTATIONS



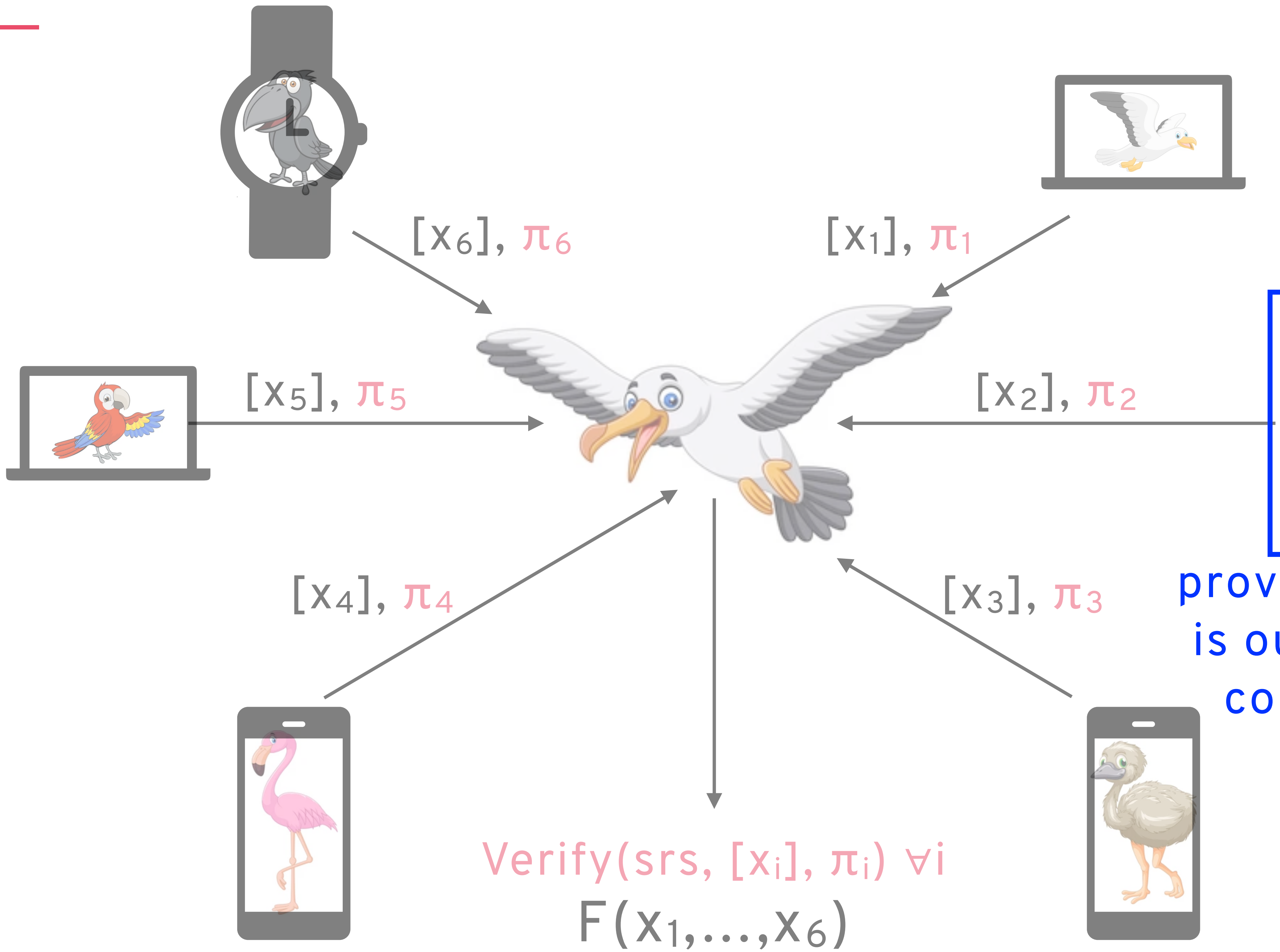
PRIVATE AGGREGATE COMPUTATIONS



PRIVATE AGGREGATE COMPUTATIONS

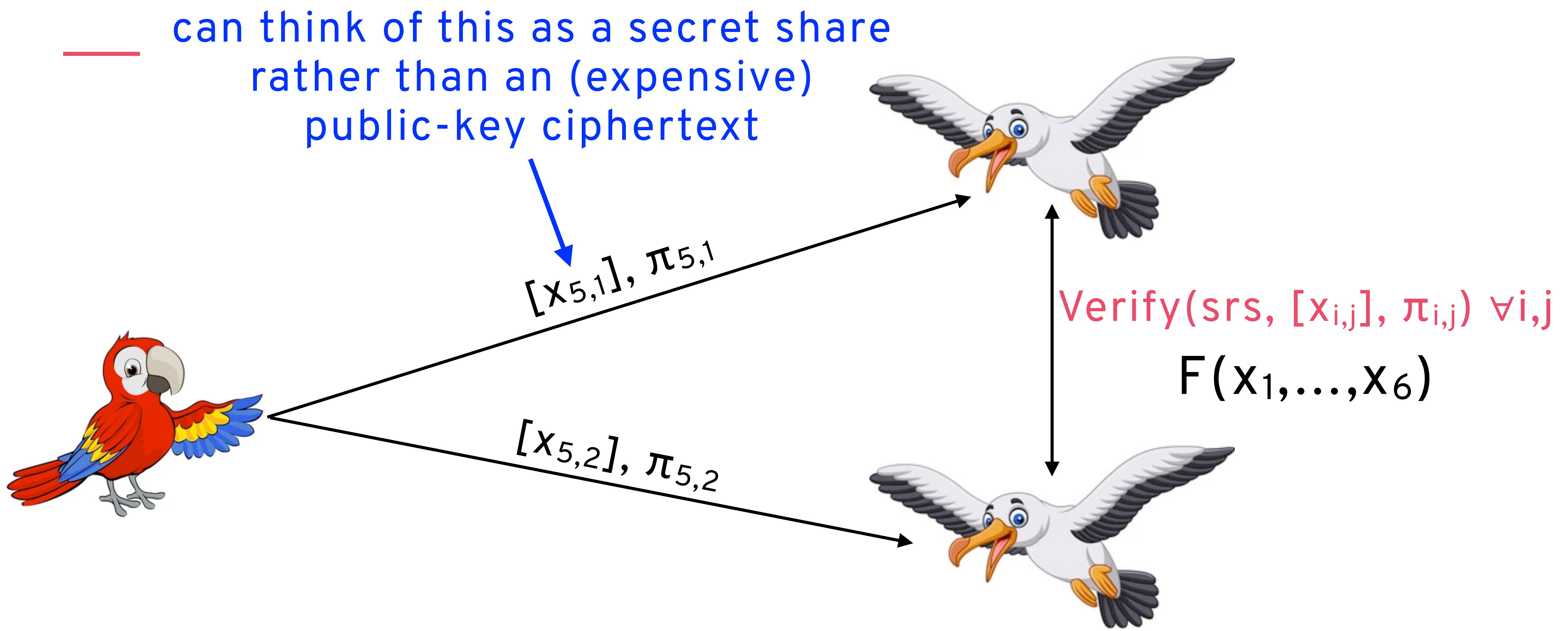


PRIVATE AGGREGATE COMPUTATIONS



prover runtime is our biggest constraint!

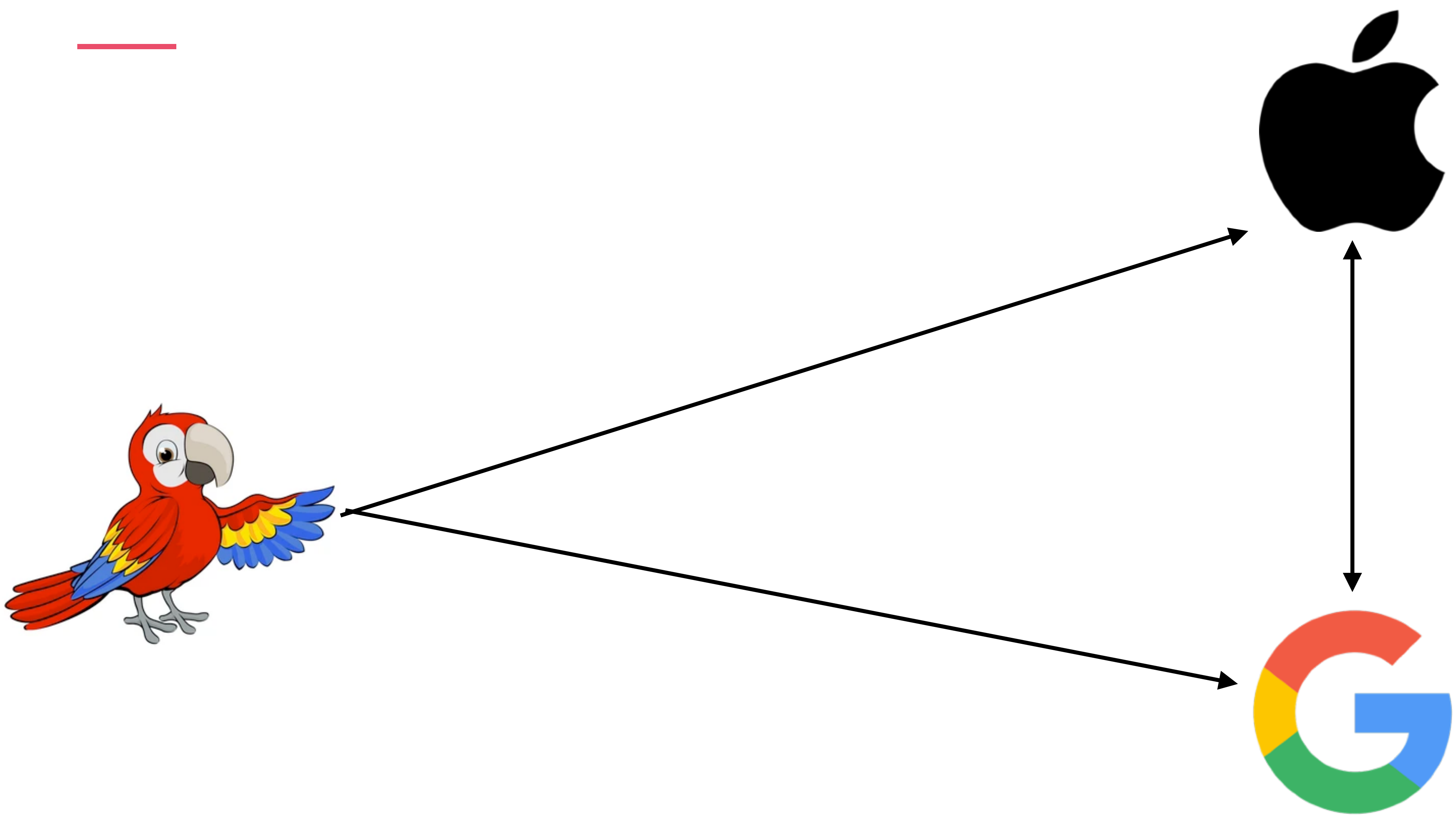
DISTRIBUTED ZERO KNOWLEDGE



Distributed zero-knowledge proofs [ACF02, C-GB17, BBC-G+19] consider one prover and multiple verifiers who do not all collude

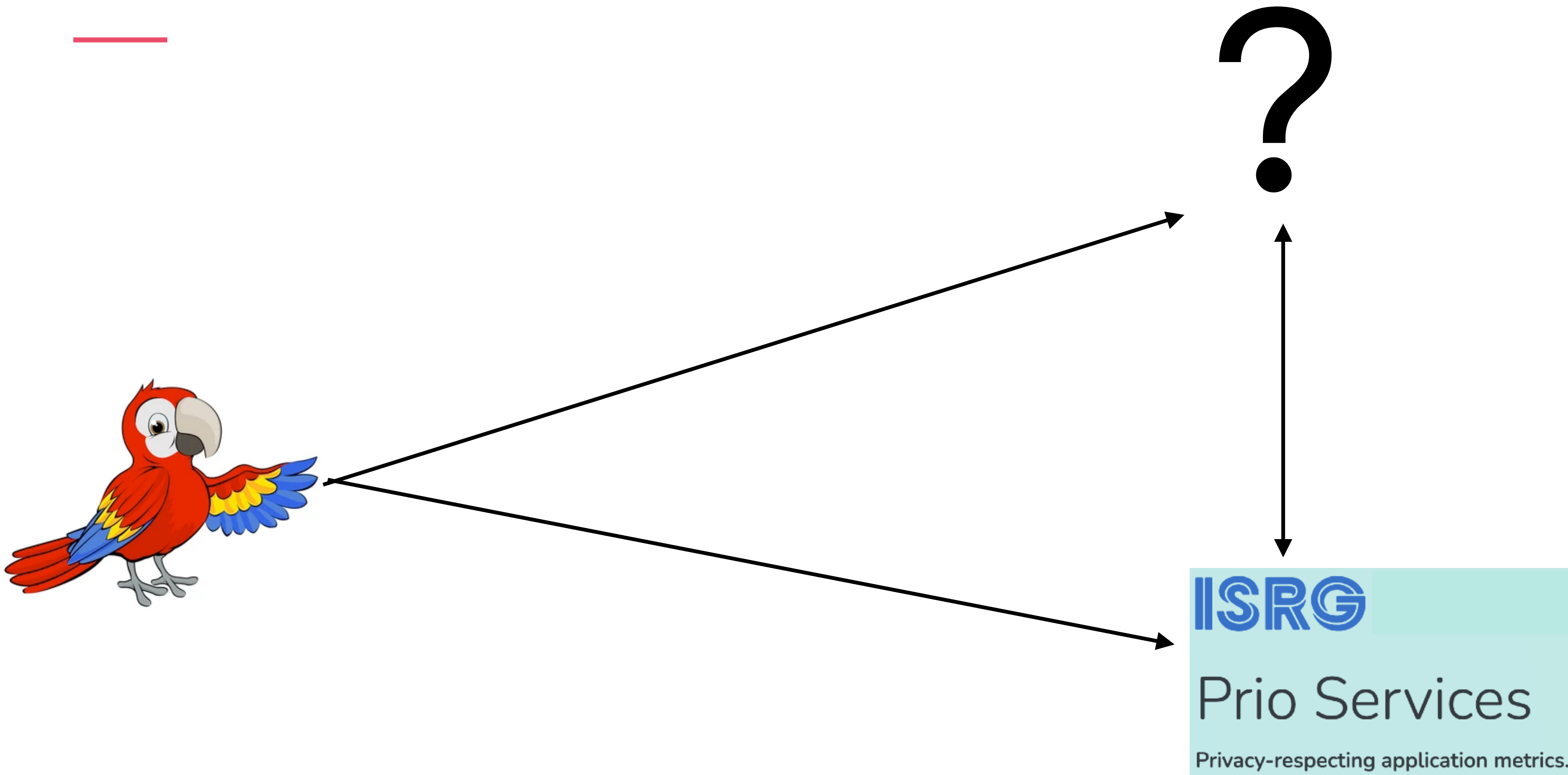
Can do this with constant communication between the two verifiers and lower computation for both the prover and verifier

DISTRIBUTED ZERO KNOWLEDGE



Used by Apple and Google in their exposure notification system

DISTRIBUTED ZERO KNOWLEDGE

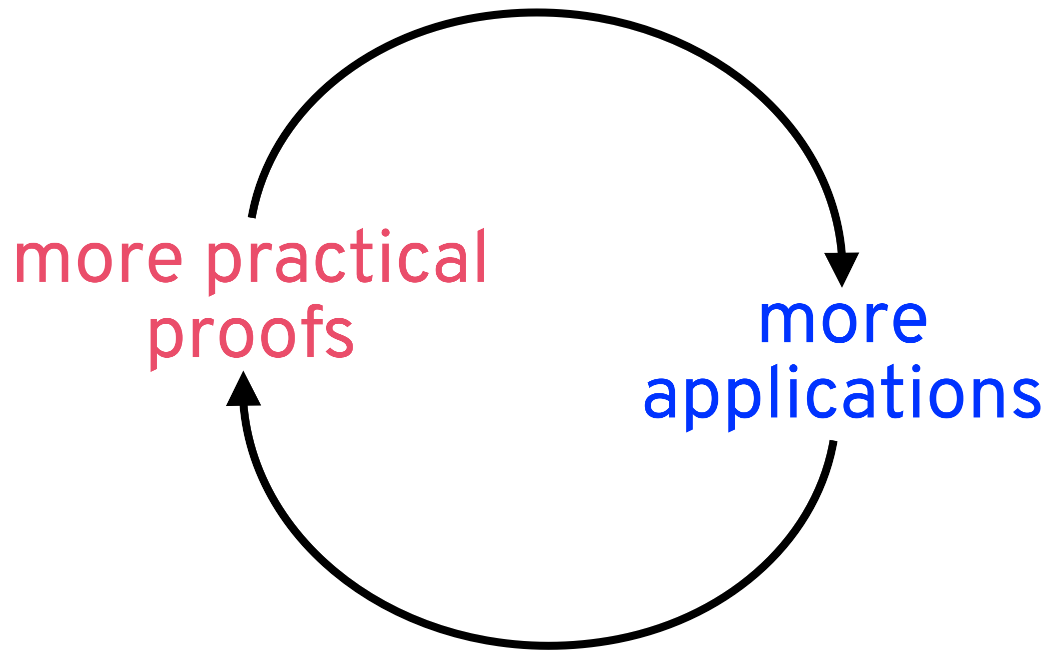


Mozilla has experimented with Prio for telemetry data

ISRG offers running “the other server” as a service

CONCLUSIONS

It's a fun and exciting time to be working on zero-knowledge proofs!



There is a ton of work in terms of:

- S{N,T}ARK-friendly hash functions, data structures, etc.
- Models for new applications that enable new constructions
- Improved techniques and optimizations
- Post-quantum friendliness

THANKS!

ANY QUESTIONS?

REFERENCES (IN ORDER OF APPEARANCE)

[GMR89]: Goldwasser, Micali, and Rackoff, The knowledge complexity of interactive proof systems
[BFM88]: Blum, Feldman, and Micali, Non-interactive zero-knowledge and its applications
[GO94]: Goldreich and Oren, Definitions and properties of zero-knowledge proof systems
[CD97]: Cramer and Damgård, Zero-knowledge proofs for finite field arithmetic; or: Can zero-knowledge be for free?
[GS08]: Groth and Sahai, Efficient non-interactive proof systems for bilinear groups
[Kilian92]: Kilian, A note on efficient zero-knowledge proofs and arguments
[Groth09]: Groth, Linear Algebra with Sub-linear Zero-Knowledge Arguments
[Groth10]: Groth, Short Pairing-Based Non-interactive Zero-Knowledge Arguments
[BCC+16]: Bootle et al., Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting
Halo: Bowe, Grigg, and Hopwood, Halo: Recursive Proof Composition without a Trusted Setup
Hyrax: Wahby et al., Doubly-efficient zkSNARKs without trusted setup
Bulletproofs: Bünz et al., Bulletproofs: Short Proofs for Confidential Transactions and More
[GGPR13]: Gennaro et al., Quadratic Span Programs and Succinct NIZKs without PCPs
[LMR19]: Lai, Malavolta, and Ronge, Succinct Arguments for Bilinear Group Arithmetic: Practical Structure-Preserving Cryptography
Libra: Xie et al., Libra: Succinct Zero-Knowledge Proofs with Optimal Prover Computation
[Groth16]: Groth, On the Size of Pairing-Based Non-interactive Arguments
[BISW17]: Boneh et al., Lattice-based SNARGs and their application to more efficient obfuscation
[BISW18]: Boneh et al., Quasi-optimal snargs via linear multi-prover interactive proofs
[GMNO18]: Gennaro et al., Lattice-Based zk-SNARKs from Square Span Programs
ZKBoo: Giacomelli, Madsen, and Orlandi, ZKBoo: Faster Zero-Knowledge for Boolean Circuits
Ligero: Ames et al., Ligero: Lightweight Sublinear Arguments Without a Trusted Setup
Aurora: Ben-Sasson et al., Aurora: Transparent Succinct Arguments for R1CS
Fractal: Chiesa, Ojha, and Spooner, Fractal: Post-Quantum and Transparent Recursive Proofs from Holography
[BCL20]: Bootle, Chiesa, and Liu, Zero-Knowledge IOPs with Linear-Time Prover and Polylogarithmic-Time Verifier
[BCG20]: Bootle, Chiesa, and Groth, Linear-Time Arguments with Sublinear Verification from Tensor Codes

REFERENCES (IN ORDER OF APPEARANCE)

[GKR+21]: Grassi et al., Poseidon: A New Hash Function for Zero-Knowledge Proof Systems
[BFS16]: Bellare, Fuchsbauer, and Scafuro, NIZKs with an Untrusted CRS: Security in the Face of Parameter Subversion
[BGG17]: Bowe et al., A multi-party protocol for constructing the public parameters of the Pinocchio zk-SNARK
[BCG+15]: Ben-Sasson et al., Secure Sampling of Public Parameters for Succinct Zero Knowledge Proofs
[BGM17]: Bowe et al., Scalable Multi-party Computation for zk-SNARK Parameters in the Random Beacon Model
[ABL+19]: Abdolmaleki et al., UC-secure CRS generation for SNARKs
[KMSV21]: Kohlweiss et al., Snarky Ceremonies
[GKMMM18]: Groth et al., Updatable and Universal Common Reference String with Applications to zk-SNARKs
Sonic / [MBKM19]: Maller et al., Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updatable Structured Reference Strings
Marlin: Chiesa et al., Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS
Plonk: Gabizon, Williamson, and Ciobotaru, PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge
Spartan: Setty, Efficient and general-purpose zkSNARKs without trusted setup
[AC20]: Attema and Cramer, Compressed Σ-Protocol Theory and Practical Application to Plug & Play Secure Algorithmics
[BBC+18]: Baum et al., Sub-Linear Lattice-Based Zero-Knowledge Arguments for Arithmetic Circuits
[BKLP15]: Benhamouda et al., Efficient zero-knowledge proofs for commitments from learning with errors over rings
[BCS21]: Bootle, Chiesa, and Sotiraki, Sumcheck Arguments and their Applications
[BLNS20]: Bootle et al., A non-PCP Approach to Succinct Quantum-Safe Zero-Knowledge
[BCG+17]: Ben-Sasson et al., Interactive Oracle Proofs with Constant Rate and Query Complexity
[BIK+17]: Bonawitz et al., Practical secure aggregation for privacy-preserving machine learning
[ACF02]: Abe et al., Non-interactive Distributed-Verifier Proofs and Proving Relations among Commitments
[C-GB17]: Corrigan-Gibbs and Boneh, Prio: private, robust, and scalable computation of aggregate statistics
[BBC-G+19]: Boneh et al., Zero-Knowledge Proofs on Secret-Shared Data via Fully Linear PCPs