Boosting Verifiable Computation on Encrypted Data

Dario Fiore, **Anca Nitulescu**,
David Pointcheval

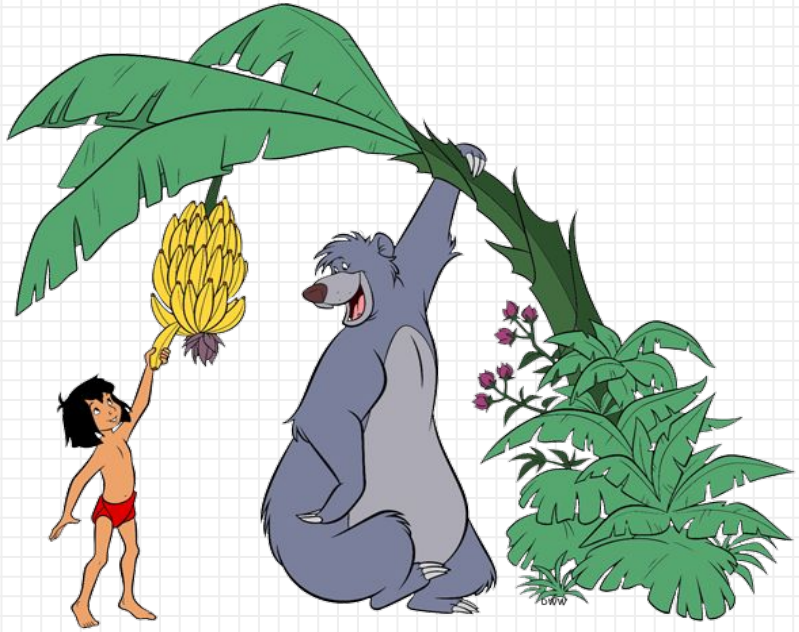
PKC 2020

The Bare Necessities of a Cloud User

(In times of Pandemics)

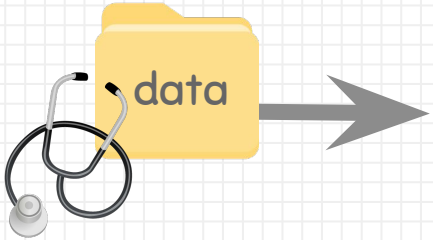
Motivational Tale:
The Bare Necessities of a Cloud User

(In times of Pandemics)

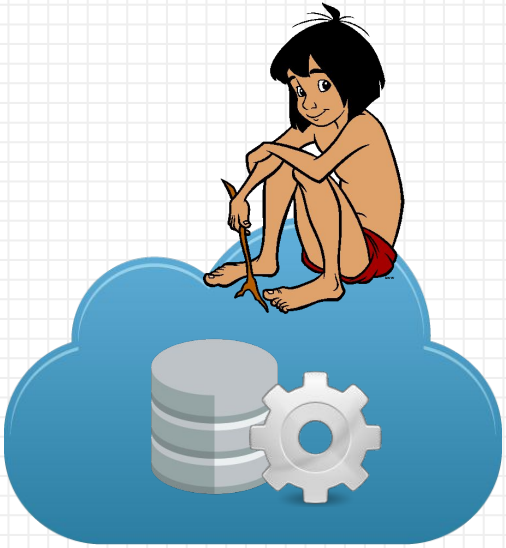


Pandemics biometric surveillance systems

Client



User delegates its personal data
to a symptom tracking app

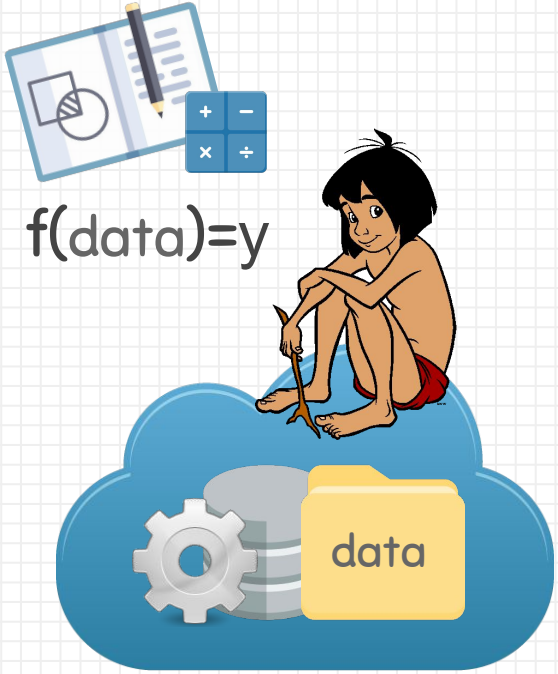


Server

Pandemics biometric surveillance systems

Client

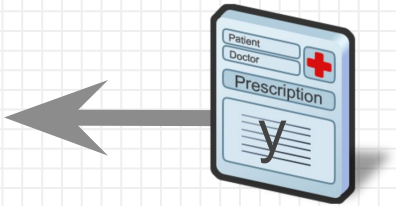
User delegates its symptoms
Server computes diagnosis



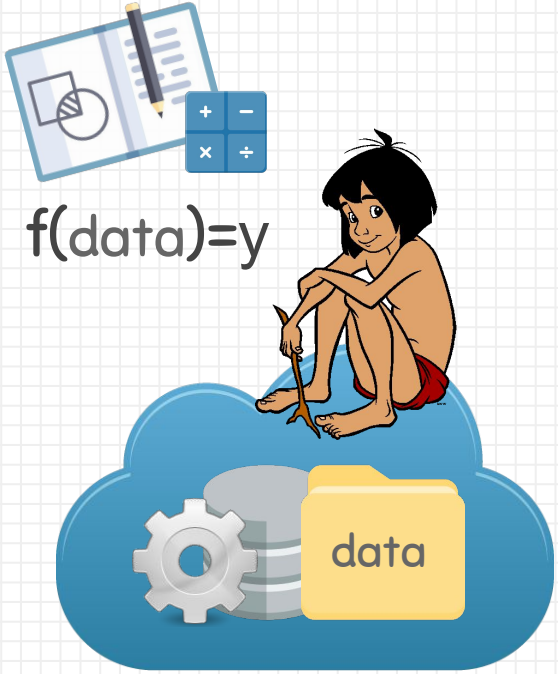
Server

Pandemics biometric surveillance systems

Client



Server sends back diagnosis



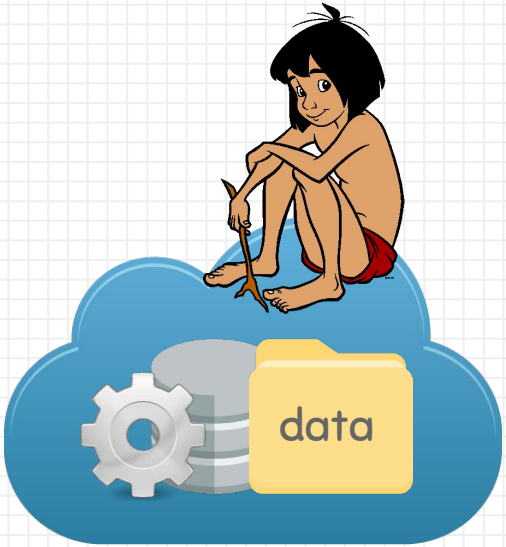
Server

So many benefits!

Client

User receives diagnosis

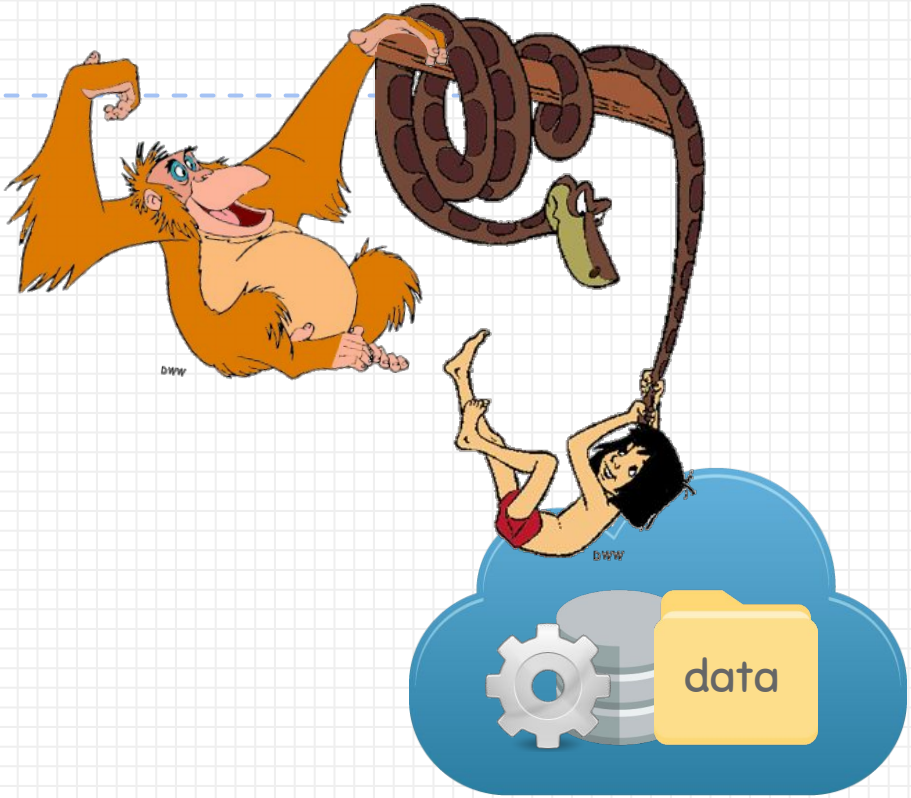
Happy to hear he is healthy



Server

Untrusted Server

Client



Server

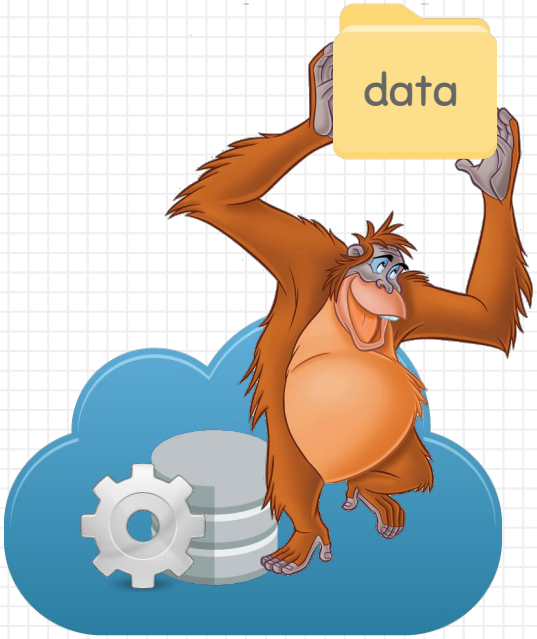
User runs the risk of a corrupted server

What can go wrong? Data can be stolen

Client

Confidential data is exposed

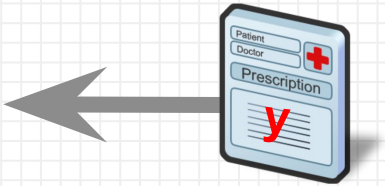
symptoms



Server

What can go wrong? Results can be modified

Client



f(data)≠y

Server

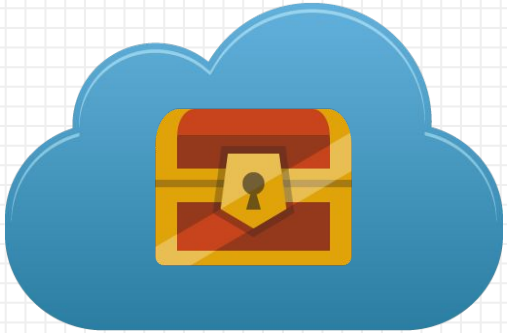
Results are not guaranteed to be correct

diagnosis

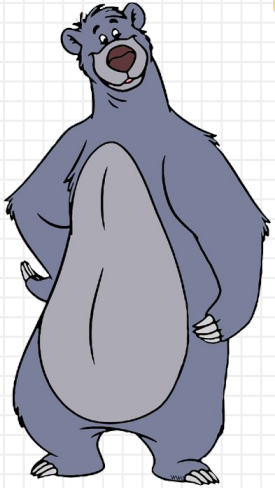
Solution for Privacy of Inputs

Data Privacy

Encryption

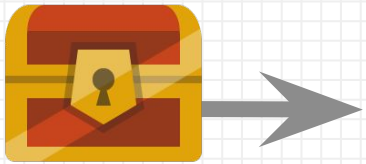


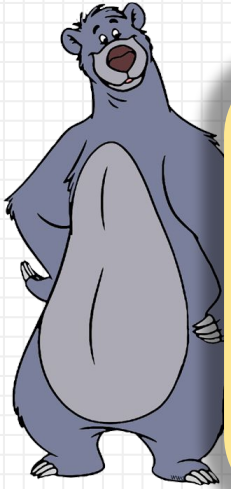
Server



(Fully) Homomorphic Encryption

Data Privacy





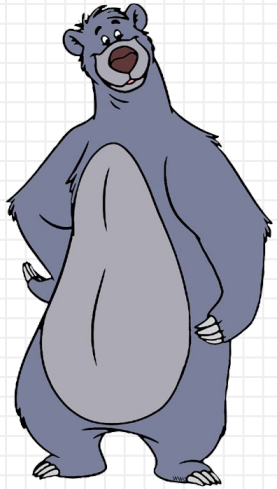
Homomorphic Encryption

- ✘ Privacy of inputs
- ✘ Malleability of data
- ✘ Privacy of output

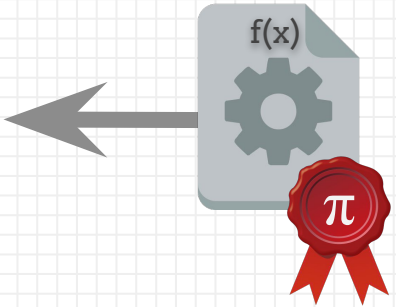
[Gen09, BV11, BGV12, GSW13, CGGI16, CKKS17...]

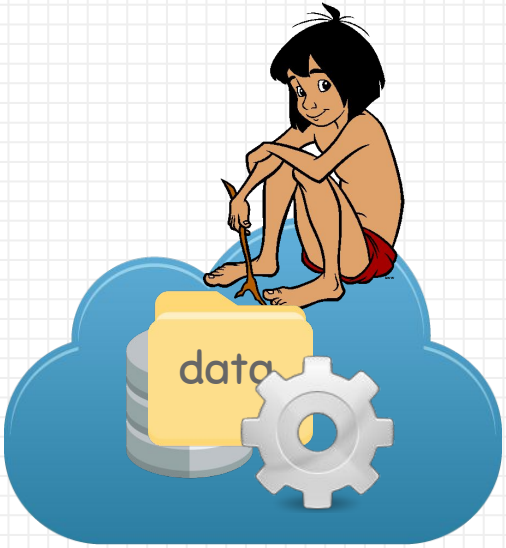
Server

Solution for Integrity of the Computation



Verifiable Computation





Server

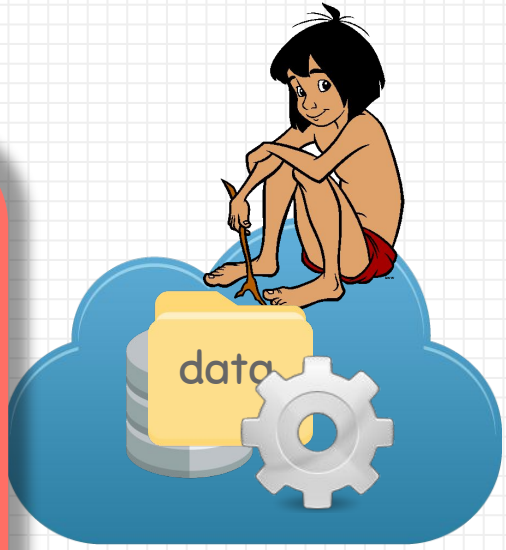
SNARKs = Proof Systems for lazy clients

Verifiable Computation

zk-SNARKs

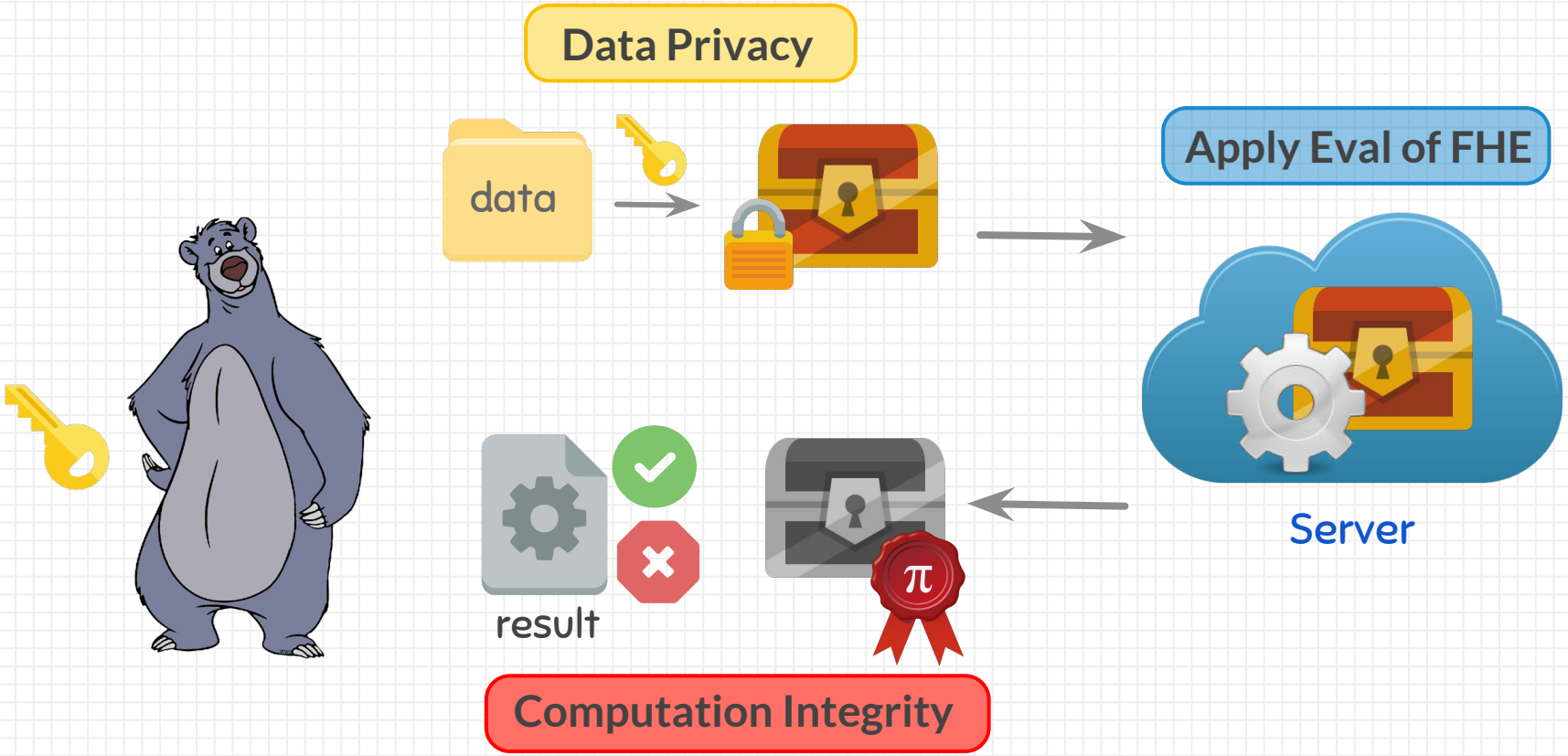
- ✘ Proof is succinct
- ✘ Minimal interaction
- ✘ Client verifies efficiently
- ✘ Server algo remains secret

[GGP10, GGPR13, PHGR13, Gro16, BBC+18...]



Server

Full Solution: Verifiable Computation on Encrypted Data



Full Solution: Verifiable Computation on Encrypted Data

[GGP10] Non-interactive VC: Outsourcing computation to untrusted workers.
Rosario Gennaro, Craig Gentry, Bryan Parno

- ✘ Combines garbled circuits and FHE
- ✘ Non-interactive VC scheme for arbitrary functions
- ✘ Privacy for inputs and outputs (from Server)

[GKP+13] How to run turing machines on encrypted data.
Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, Nickolai Zeldovich

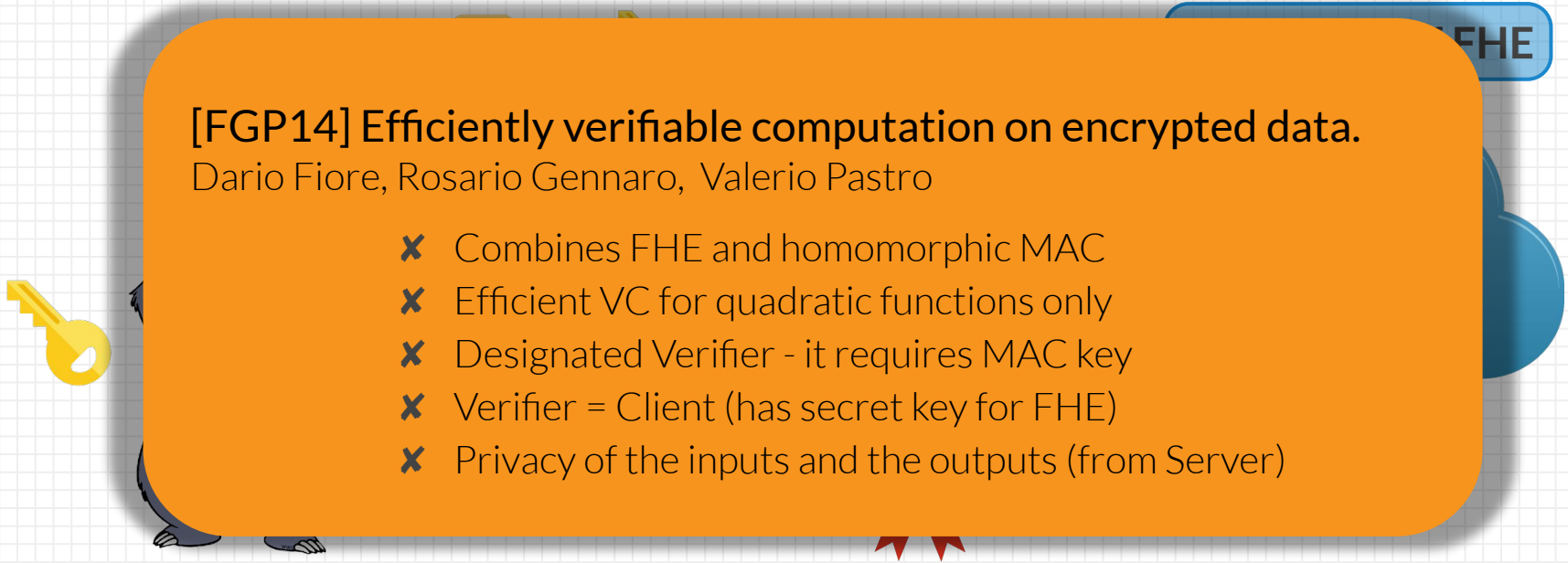
- ✘ Uses succinct single-key functional encryption scheme
- ✘ VC for functions with a single bit of output
- ✘ privacy of the inputs, but not of the outputs

[FGP14] Efficiently verifiable computation on encrypted data.
Dario Fiore, Rosario Gennaro, Valerio Pastro

- ✘ Combines FHE and VC
- ✘ VC for quadratic functions only
- ✘ privacy of the inputs, but not of the outputs

Full Solution: Verifiable Computation on Encrypted Data

Data Privacy



[FGP14] Efficiently verifiable computation on encrypted data.
Dario Fiore, Rosario Gennaro, Valerio Pastro

- ✘ Combines FHE and homomorphic MAC
- ✘ Efficient VC for quadratic functions only
- ✘ Designated Verifier - it requires MAC key
- ✘ Verifier = Client (has secret key for FHE)
- ✘ Privacy of the inputs and the outputs (from Server)

Computation Integrity

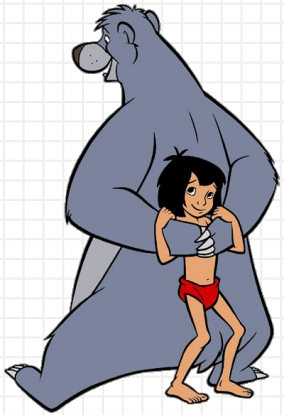
Outline

Private VC
Goals
Strategy

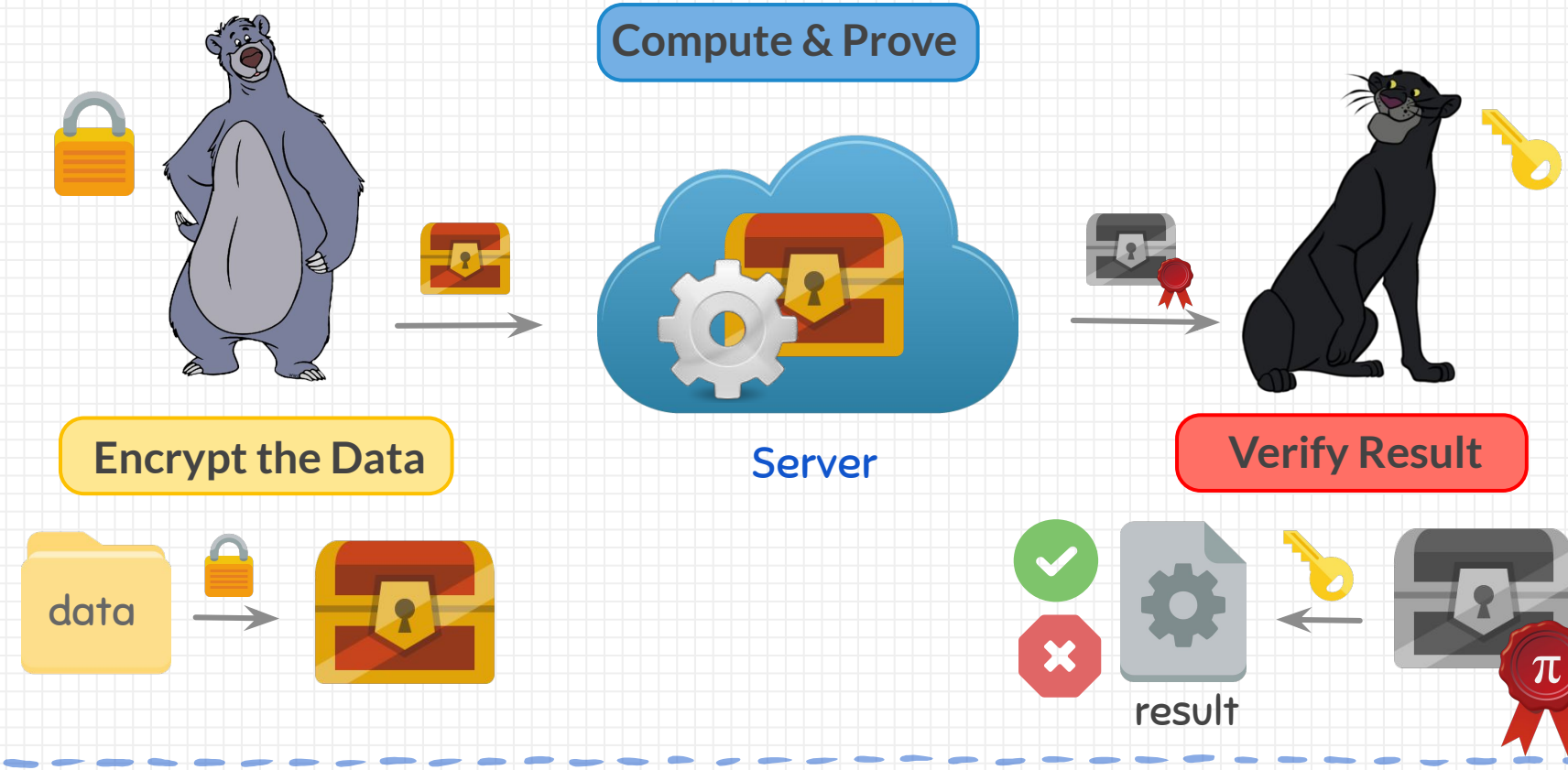
Building Blocks
Polynomial Commitments
CaP zk-SNARKs

Technical Challenges

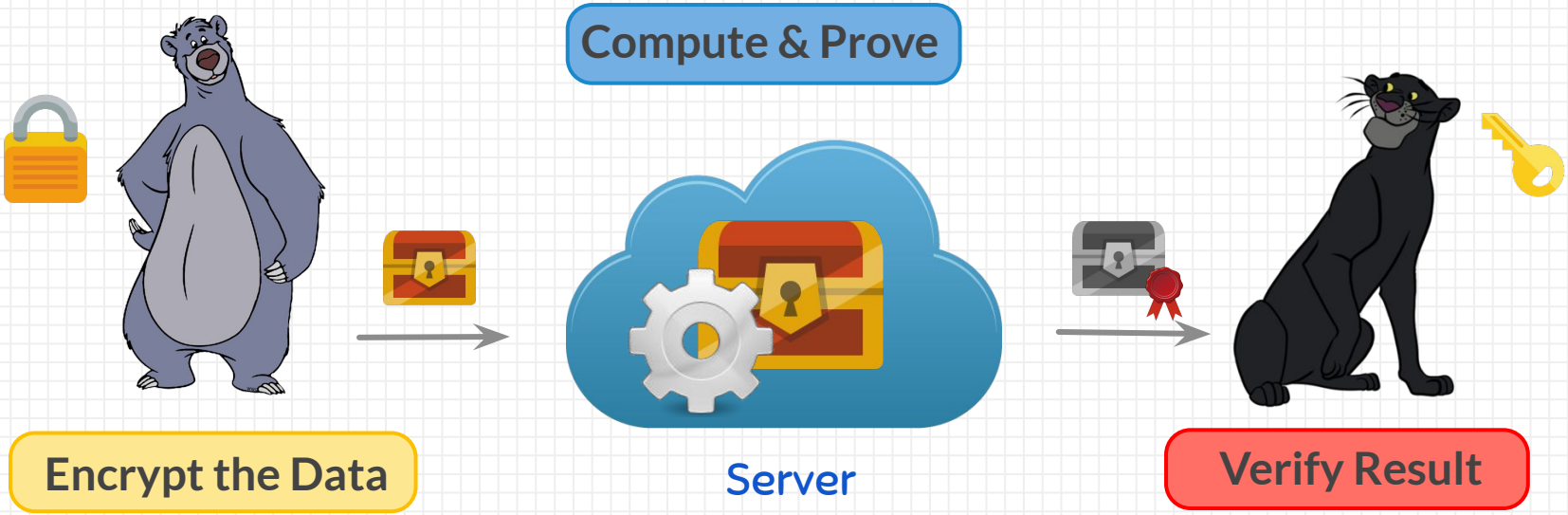
The END



Publicly Verifiable Computation with Privacy



Publicly Verifiable Computation with Privacy



Solution that improves on **[FGP14]** :

- ✘ Public verifiable: Client & Verifier do not share keys
- ✘ Efficiency for higher degree computations (arithmetic circuits)

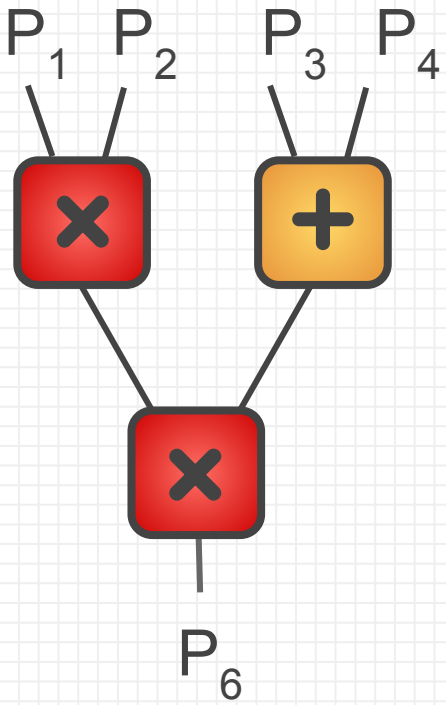
Idea: Exploit the specificity of FHE ciphertexts

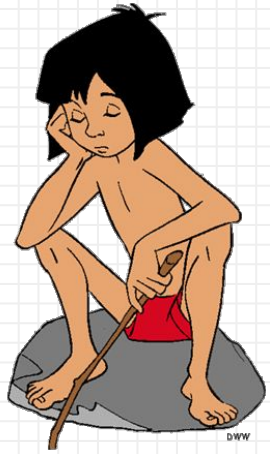
Compactly Commit to ciphertexts

Prove efficiently evaluation of circuit on ciphertexts

zk-SNARK for verifiable and private delegation of computation

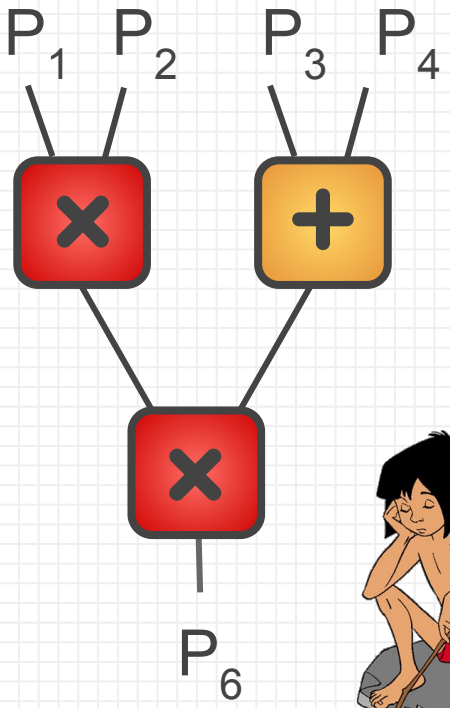
FHE: Ciphertexts = Polynomials (ring-LWE, [BV11])



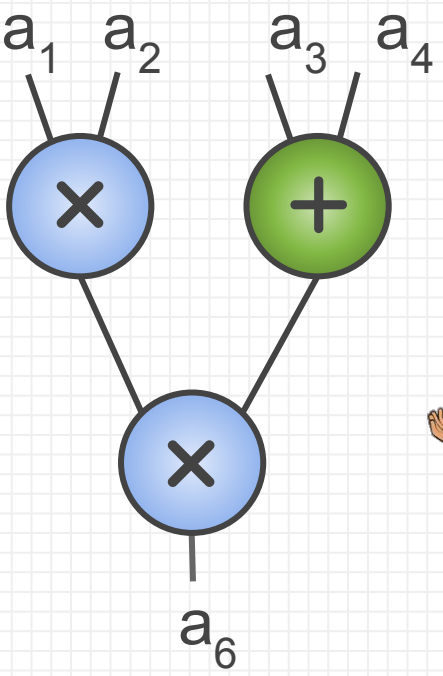


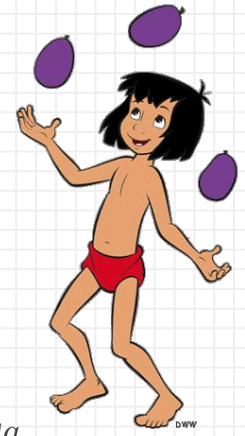
$$\mathbb{R}_q = \mathbb{Z}_q[x]/R(x)$$

Circuit over ciphertexts / over plaintexts



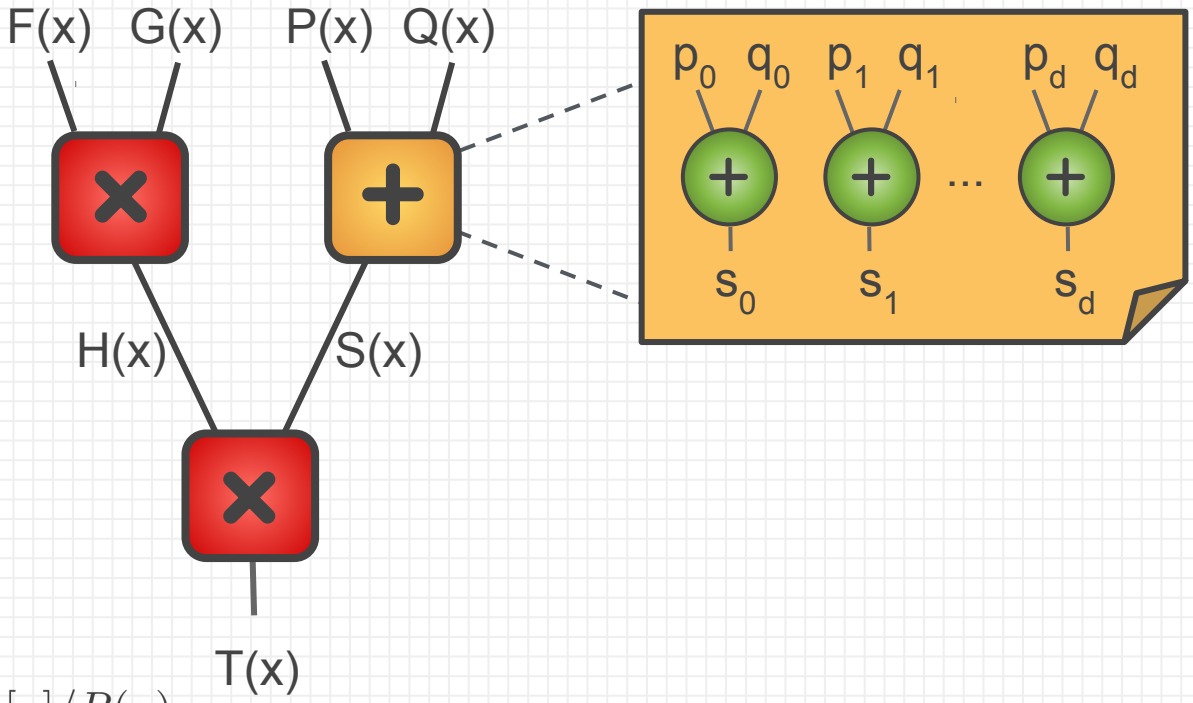
$\mathbb{R}_q = \mathbb{Z}_q[x]/R(x)$





$\mathbb{Z}_q$

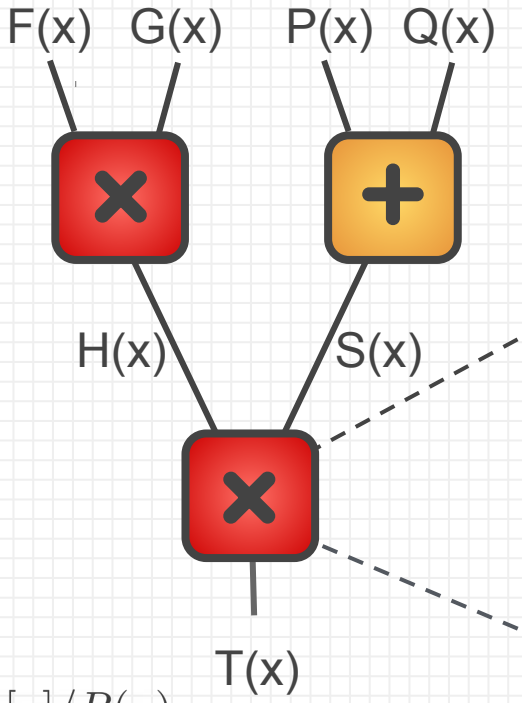
Arithmetic Circuit over Polynomials



O(d) scalar additions

$\mathbb{R}_q = \mathbb{Z}_q[x]/R(x)$

Arithmetic Circuit over Polynomials

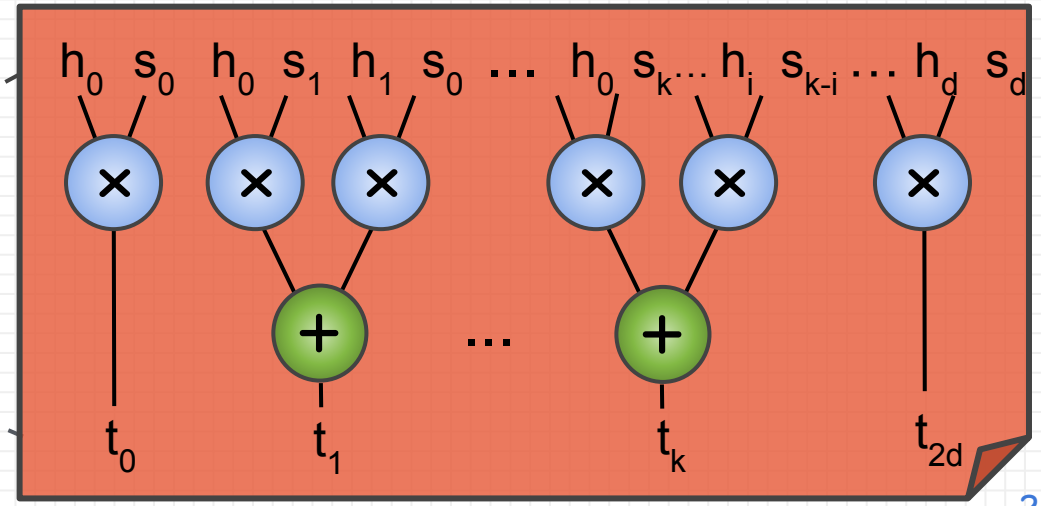


$\mathbb{R}_q = \mathbb{Z}_q[x]/R(x)$

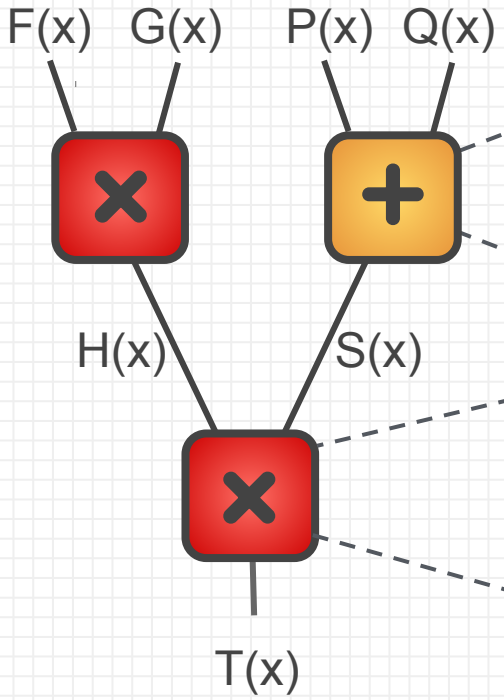
~ d^2 scalar multiplications

& reductions modulo $R(x)$ of deg d

$\mathbb{R}_q = \mathbb{Z}_q[x]/R(x)$



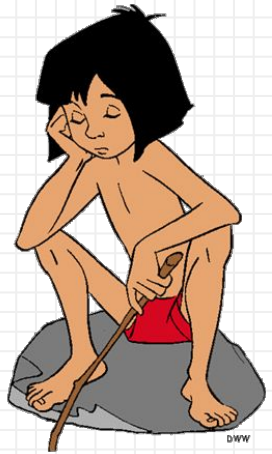
Arithmetic Circuit over Polynomials



O(*d*) scalar additions

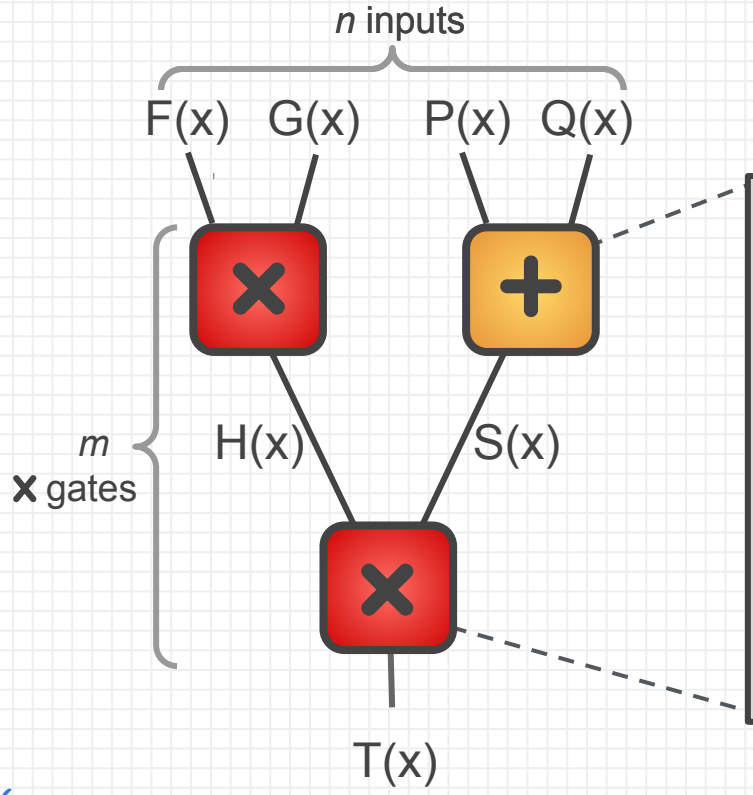
$O(d^2)$ scalar multiplications

~ $O(d \log d)$ for large d



$$\mathbb{R}_q = \mathbb{Z}_q[x]/R(x)$$

Challenge: Circuit over Polynomials

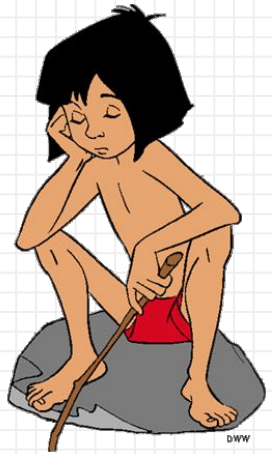


O(*n* · *d*)
scalar additions

&

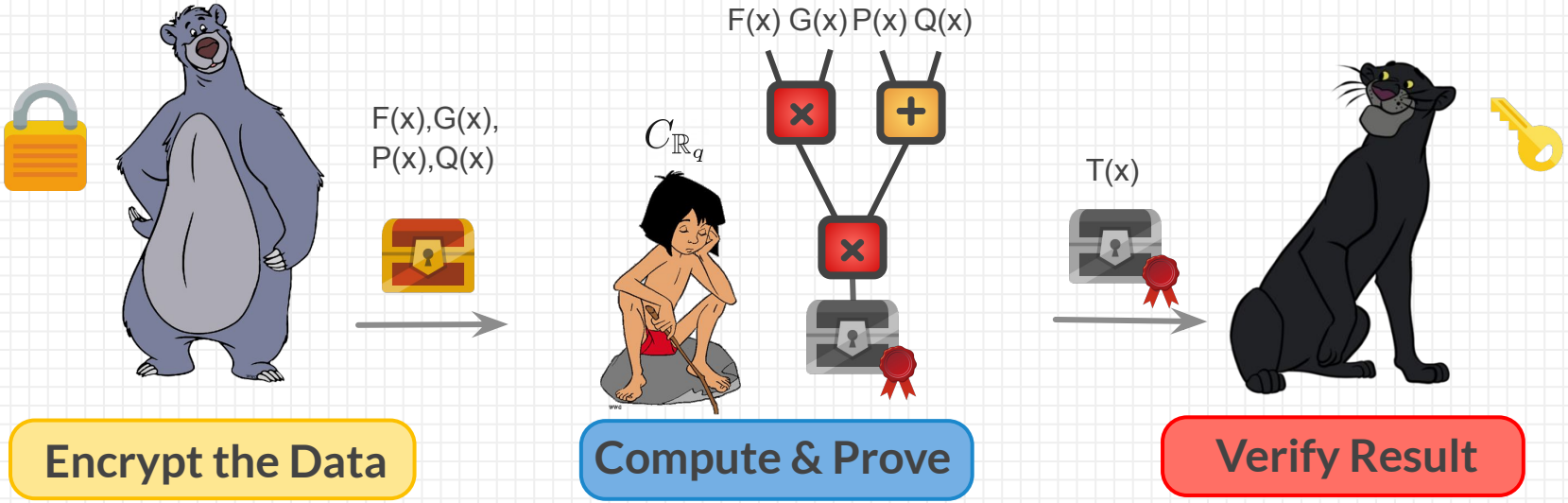
O(*m* · *d* · *log d*)
scalar multiplications

*for polynomials of degree *d*



$$\mathbb{R}_q = \mathbb{Z}_q[x]/R(x)$$

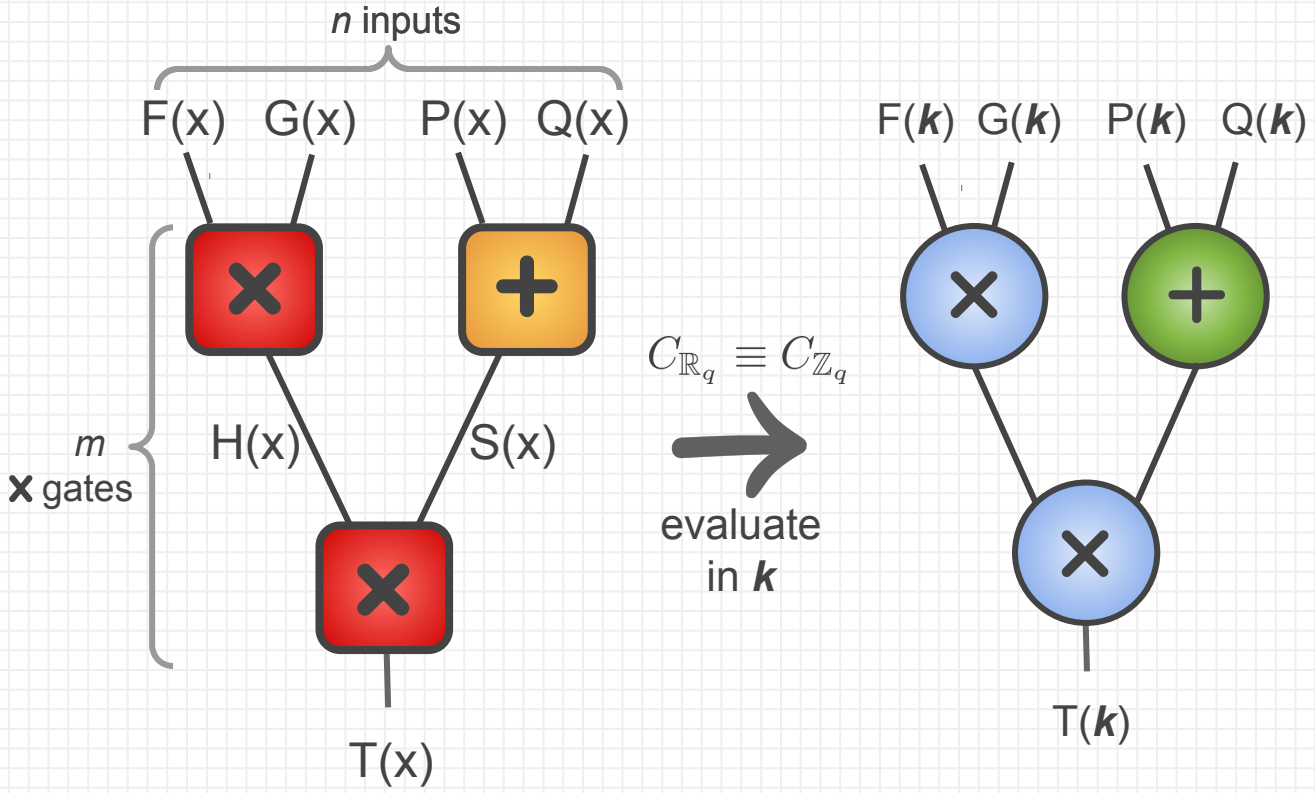
Goals: Efficient VC with Privacy

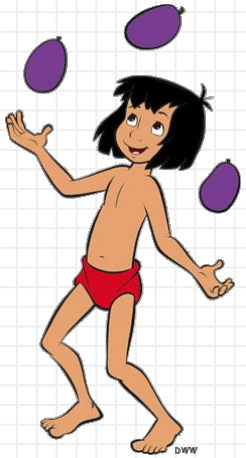


Solution that:

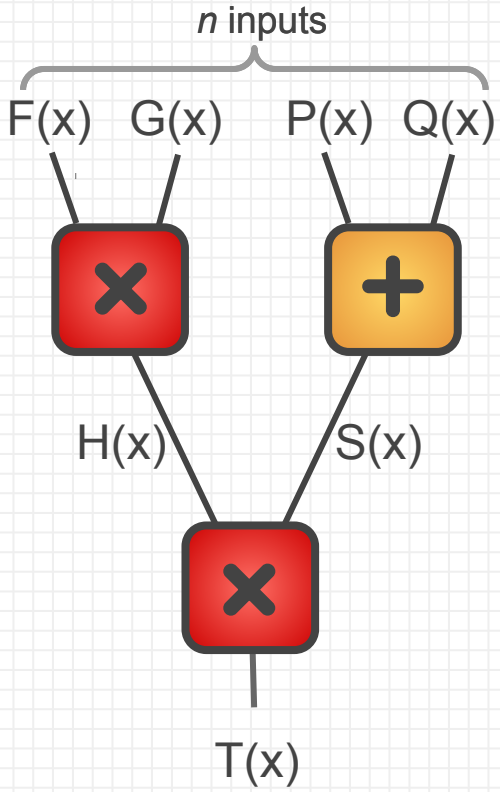
- ✘ Compactly commits to the input ciphertexts → hiding from Verifier
- ✘ Reduces the proof for $C_{\mathbb{R}_q}$ → efficiency close to cleartext proof for $C_{\mathbb{Z}_q}$

Compress Circuit over Polynomials



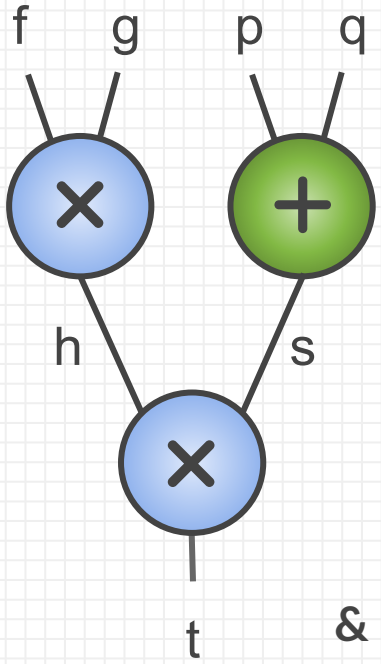


Prove Circuit over Scalars & Evaluation in k

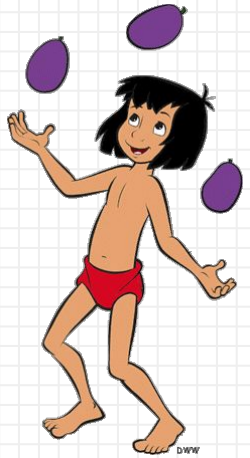


$C_{\mathbb{R}_q} \equiv C_{\mathbb{Z}_q}$

evaluate in ***k***



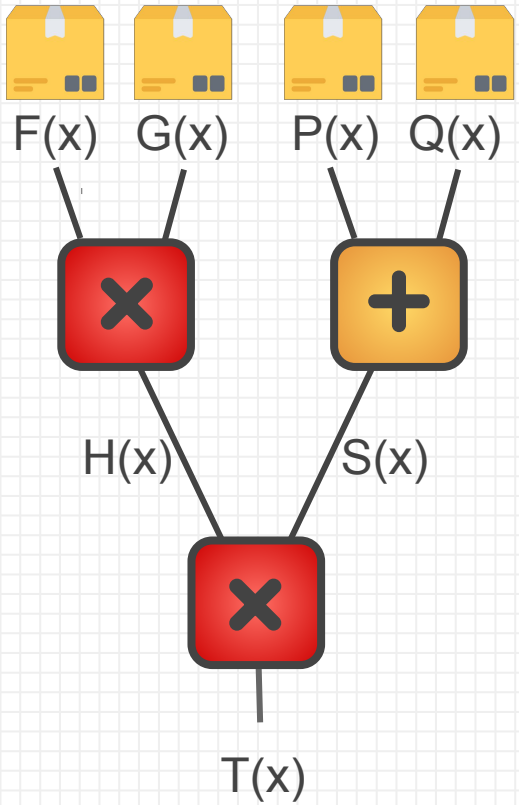
&

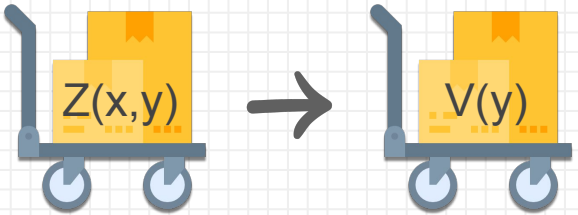


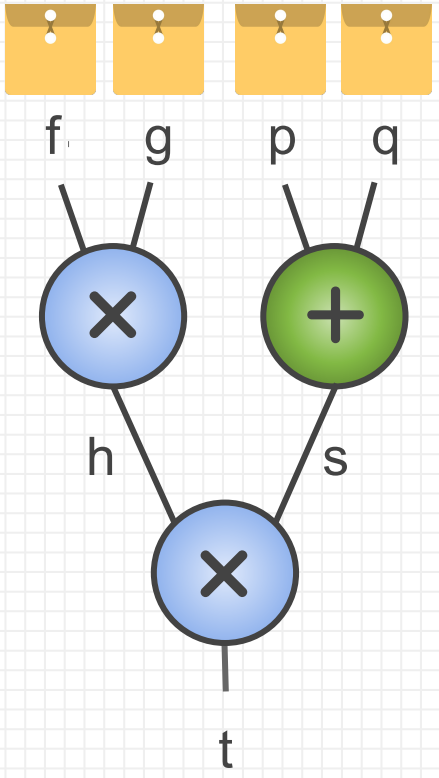
$|C_{\mathbb{Z}_q}| + d \cdot n$

f = F(***k***) p = P(***k***)
g = G(***k***) q = Q(***k***)

Commit & Prove Evaluation







Our Techniques

Private VC

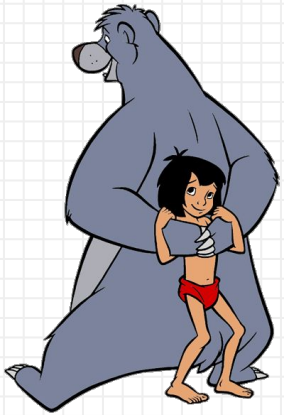
Goals
Strategy

Building Blocks

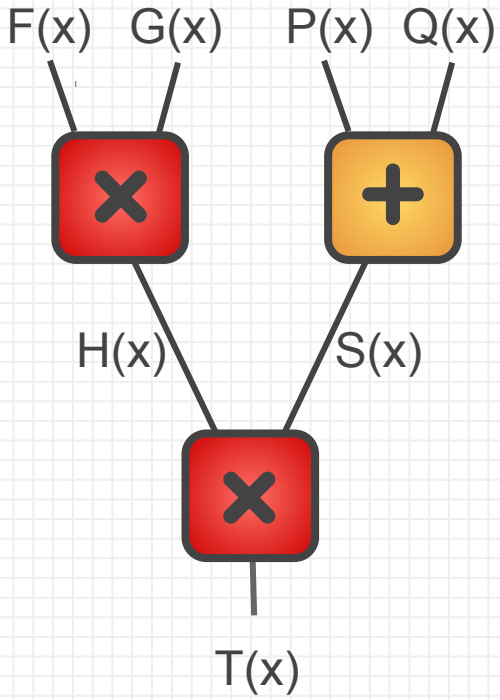
Polynomial Commitments
CaP zk-SNARKs

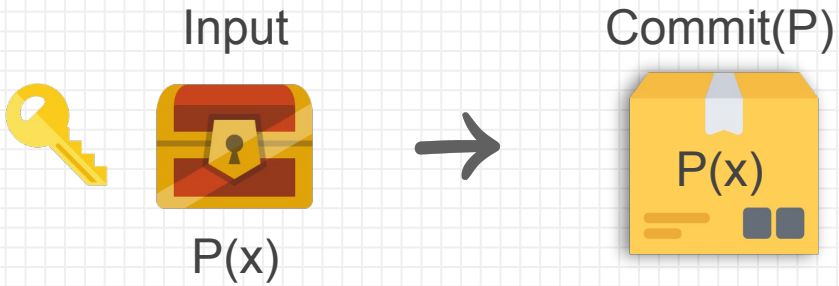
Technical Challenges

The END

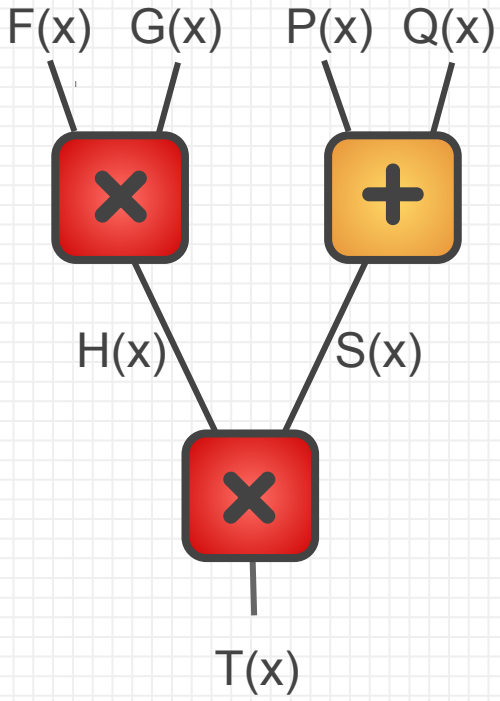


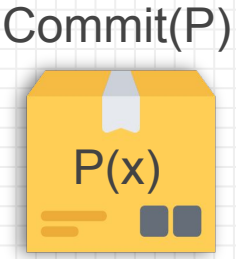
Polynomial Commitments



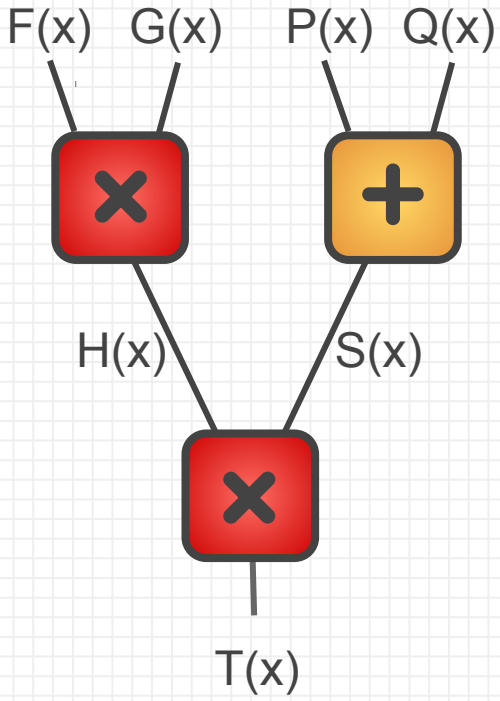


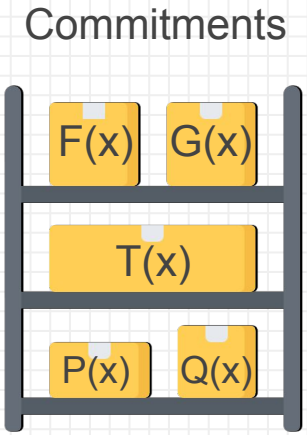
Polynomial Commitments – hiding inputs

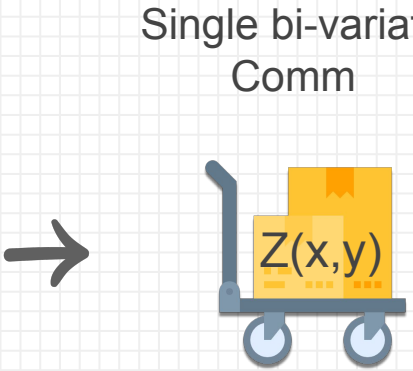




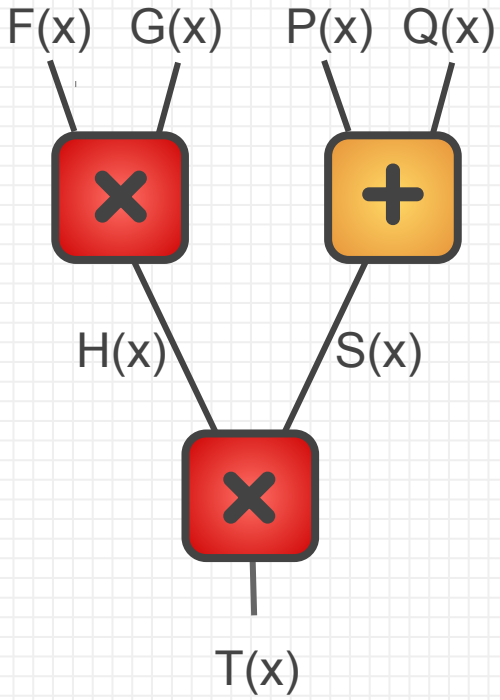
Multi-Polynomial Commitments

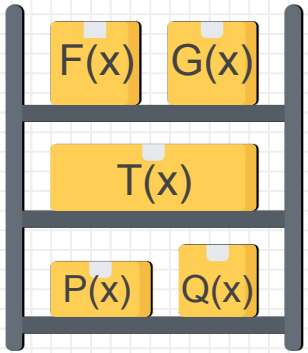






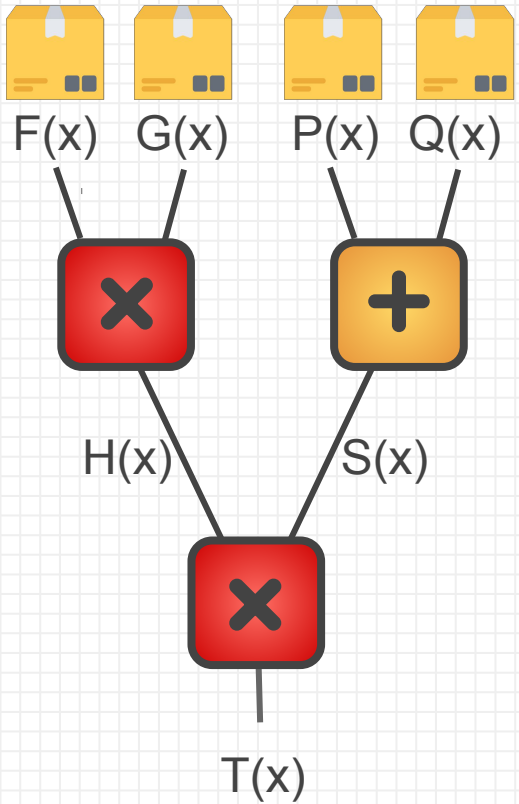
Multi-Polynomial Commitments

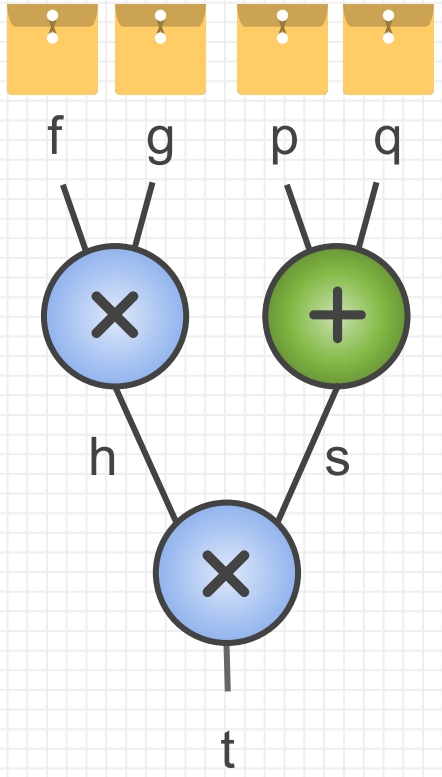




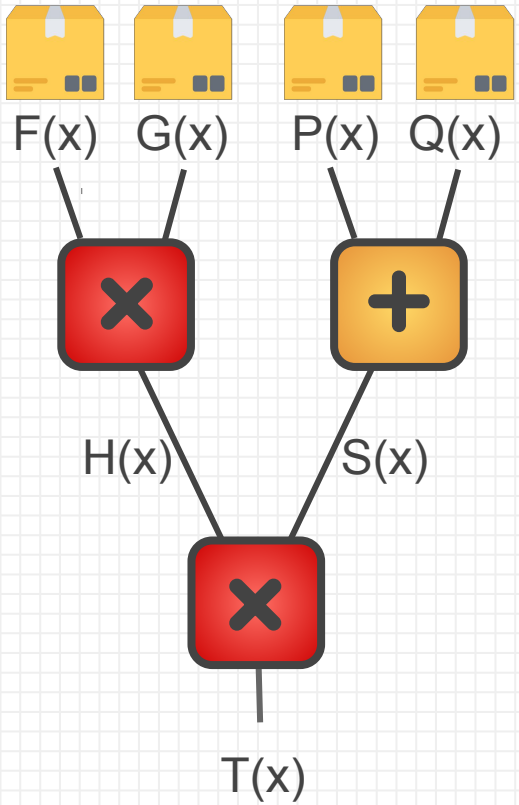
$$Z(x,\mathbf{y}) = F(x) + G(x)\mathbf{y} + T(x)\mathbf{y}^2 + P(x)\mathbf{y}^3 + Q(x)\mathbf{y}^4$$

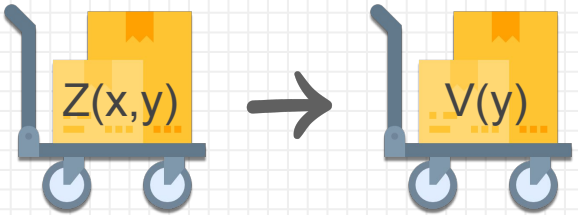
Commit & Prove Evaluation

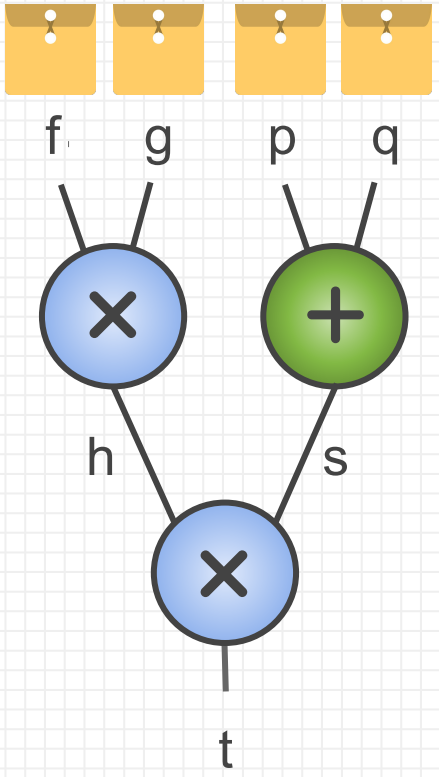




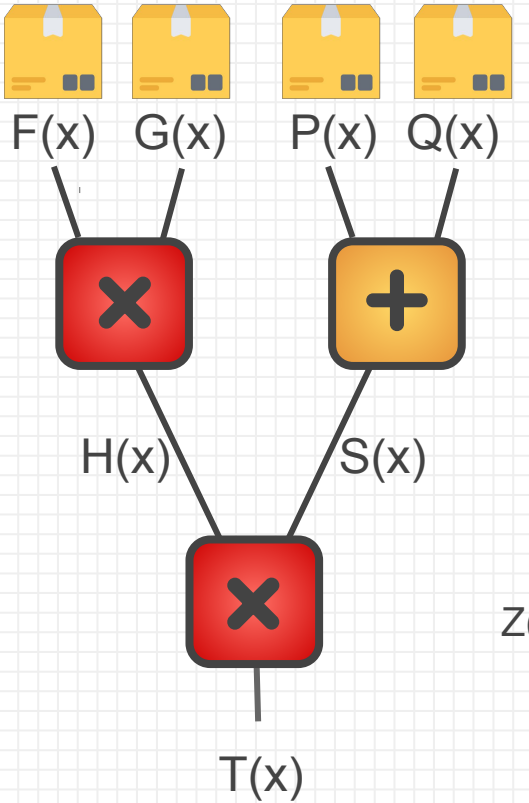
Commit & Prove Evaluation

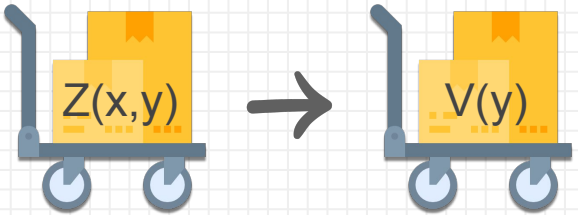






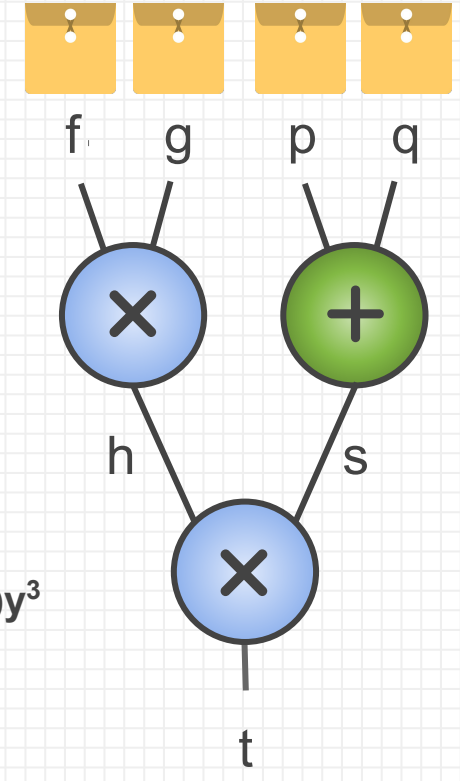
Many Evaluations = Partial Evaluation



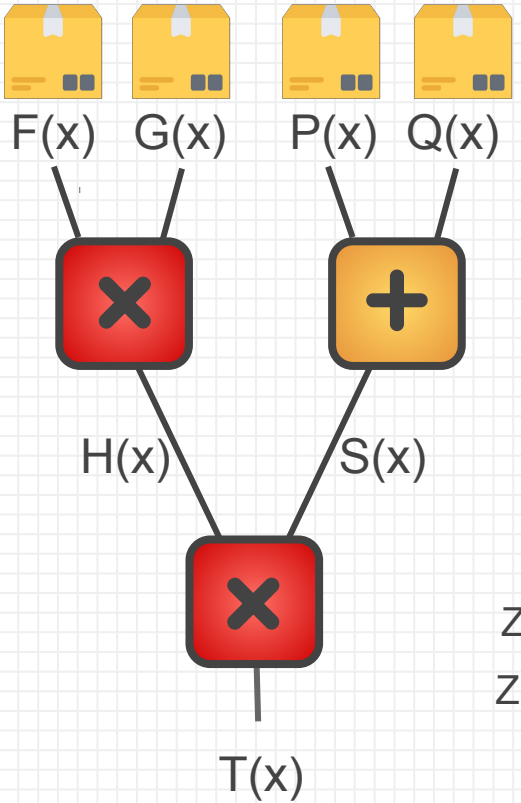


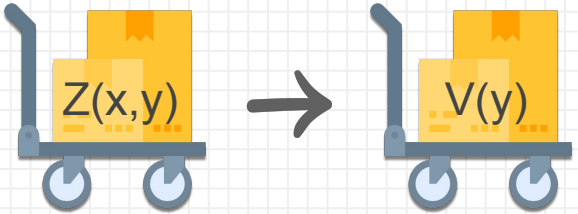
$$Z(x,y) = F(x) + G(x)\mathbf{y} + P(x)\mathbf{y}^2 + Q(x)\mathbf{y}^3$$

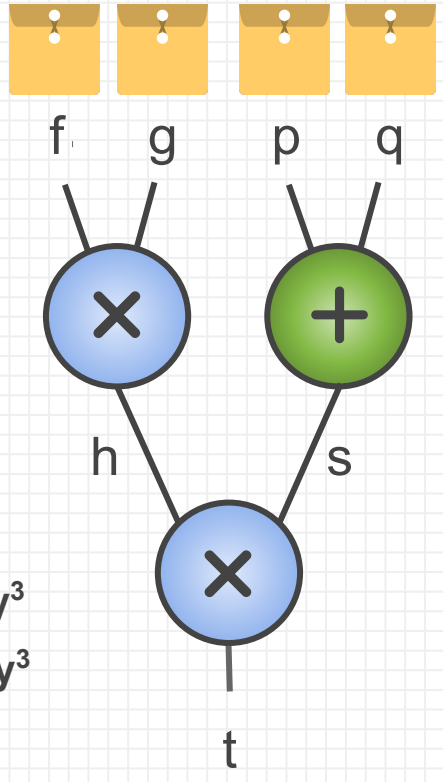
$$V(y) = f + g\,\mathbf{y} + p\,\mathbf{y}^2 + q\,\mathbf{y}^3$$



Many Evaluations = Partial Evaluation





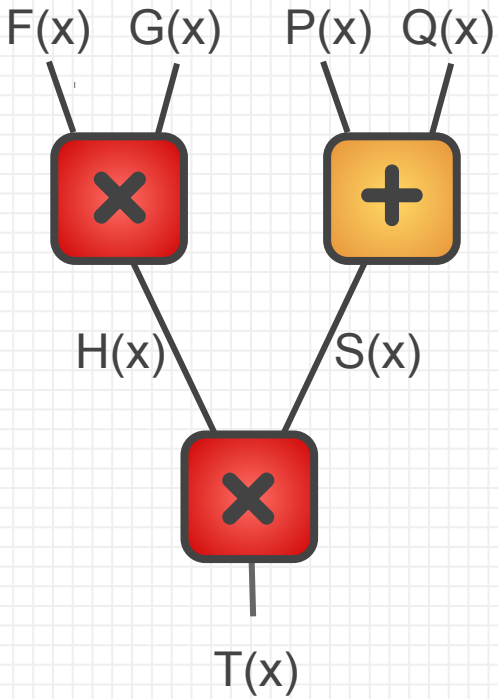


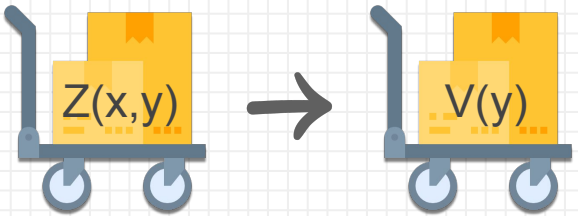
$$Z(x,y) = F(x) + G(x)\mathbf{y} + P(x)\mathbf{y}^2 + Q(x)\mathbf{y}^3$$

$$Z(\mathbf{k},y) = F(\mathbf{k}) + G(\mathbf{k})\mathbf{y} + P(\mathbf{k})\mathbf{y}^2 + Q(\mathbf{k})\mathbf{y}^3$$

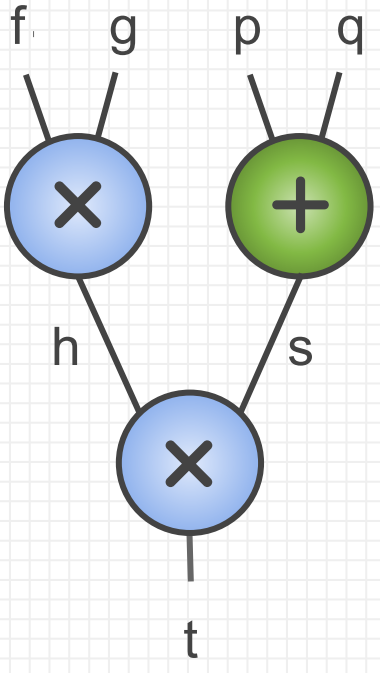
$$= V(y) = f + g\,\mathbf{y} + p\,\mathbf{y}^2 + q\,\mathbf{y}^3$$

Proof of Many Evaluations

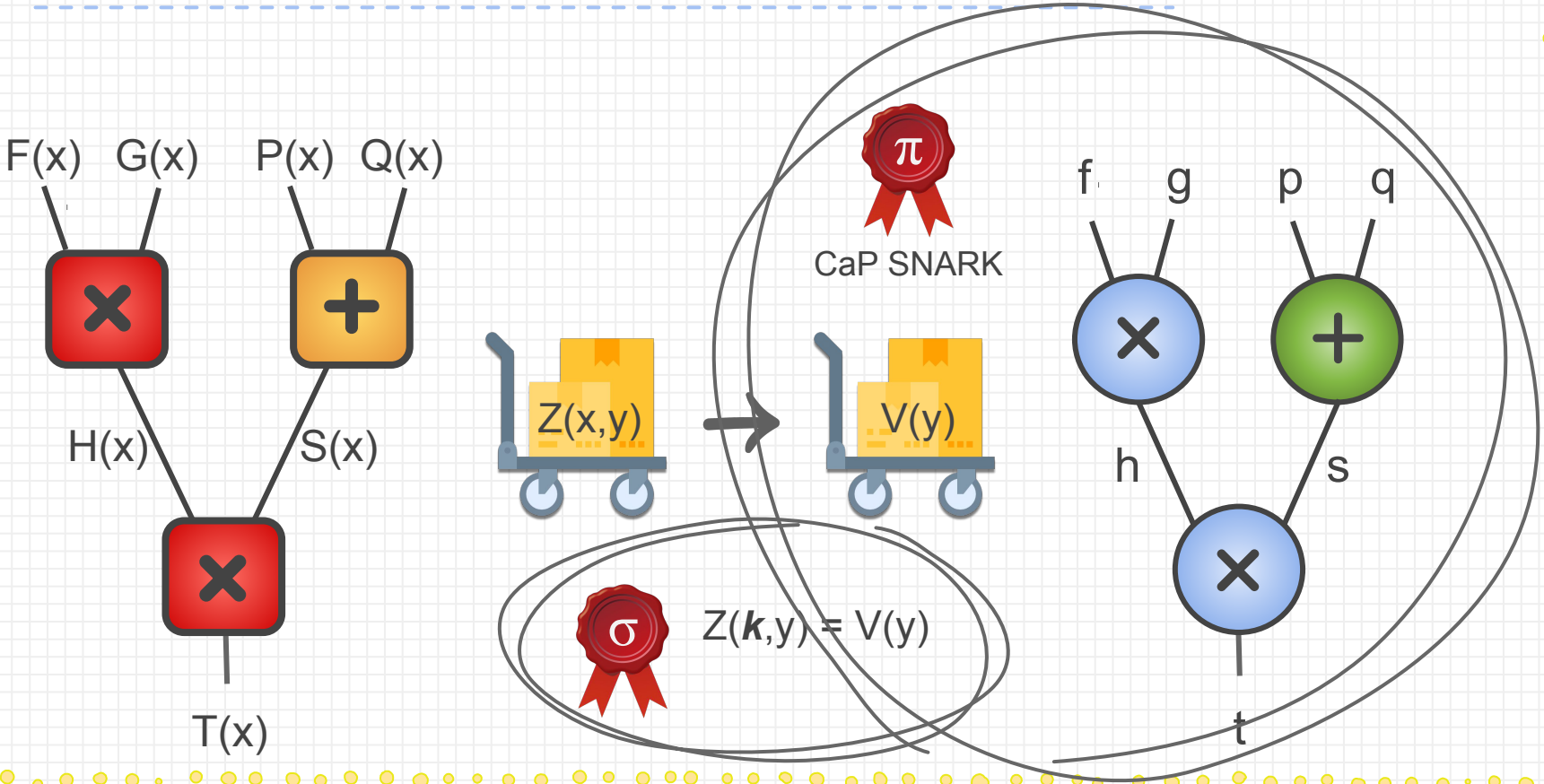




$Z(\boldsymbol{k},y) = V(y)$

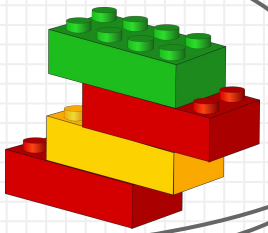


Proof of Many Evaluations

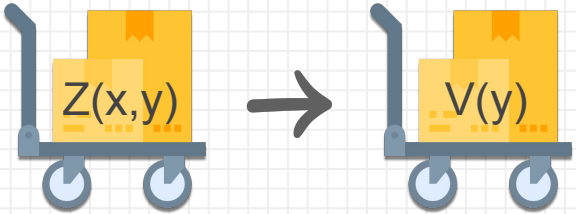


Reuse the same commitment

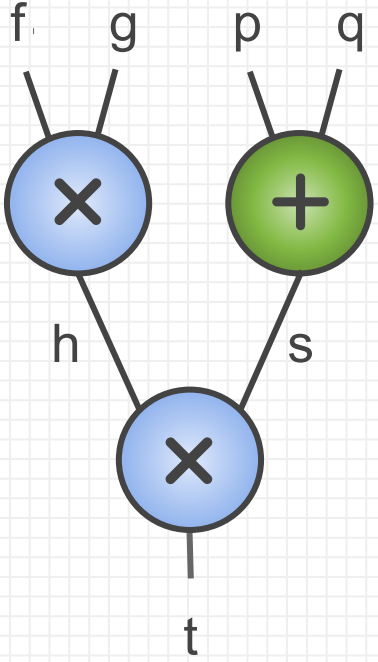
[CFQ19] Modular Commit-and-Prove (LegoSNARK)



CaP SNARK

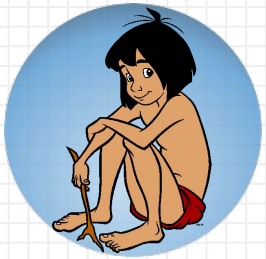


Z(***k***,y) = V(y)



Σ – Protocols & Fiat-Shamir Heuristic

$Z(\boldsymbol{k}, y) = V(y)$

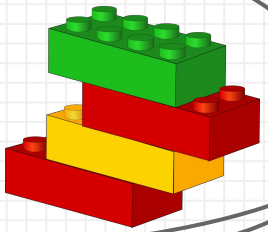


Interactive Proof → **Random Oracle Model** → **CaP zk-SNARK for Multi-Polynomial Evaluation**

P: Commits to polynomials	**P:** Commits to polynomials	✘ based on the SDH and PKE assumptions
V: Sends random point	**P: Queries point to RO**	✘ non-interactive and zero-knowledge
P: Prove the evaluation	**P:** Prove the evaluation	✘ evaluations are committed (never opened)

Reuse the same commitment

[CFQ19] Modular Commit-and-Prove (LegoSNARK)



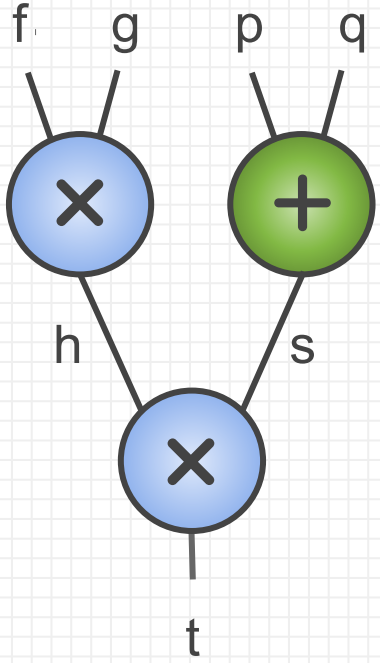
π

CaP SNARK

Z(x,y) → V(y)

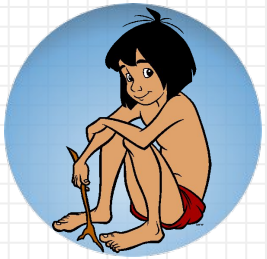
σ

Z(***k***,y) = V(y)



CaP zk-SNARK for Arithmetic Circuits

CaP SNARK
[CFQ19]



zk-SNARK	Pre-Processing	Lego-SNARK “lifting” tool
Groth 16	**CRS for QAP** Quadratic Arithmetic Programs	**LegoGro16**
UAC - GKMMM 18	Universal, circuit-independent, updatable CRS	**LegoUAC**

Review of Contributions

Private VC

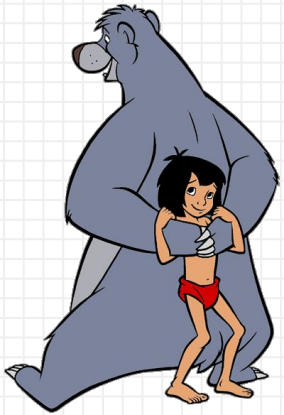
Goals
Strategy

Building Blocks

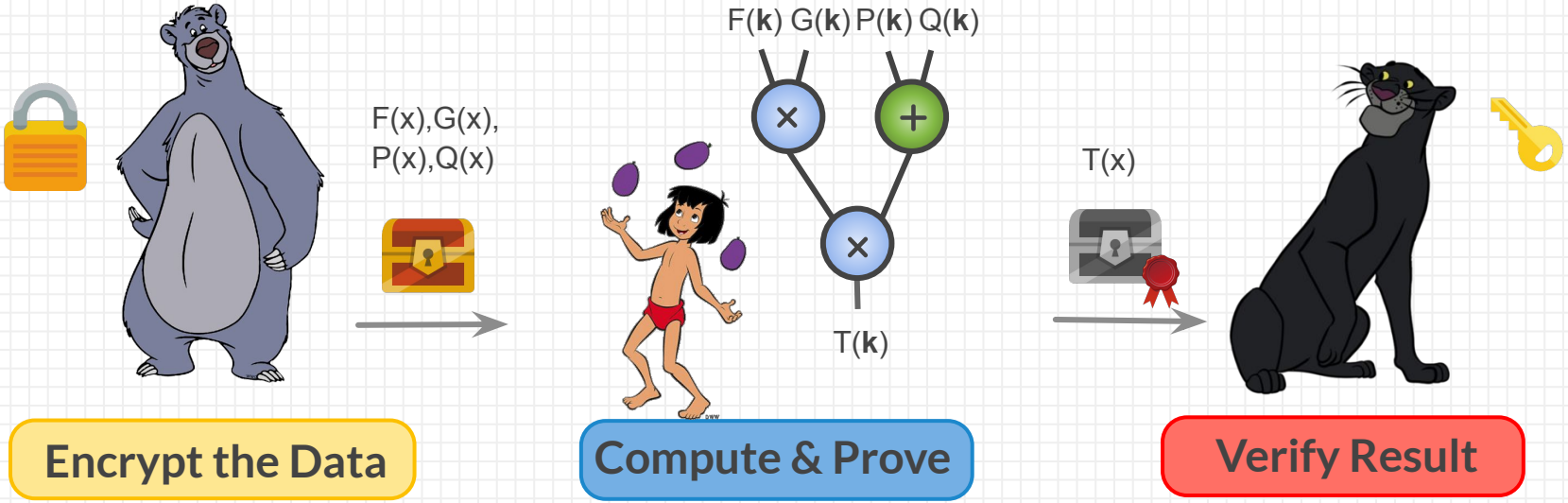
Polynomial Commitments
CaP zk-SNARKs

Technical Challenges

The END



zk-SNARK for polynomial ring computations



- ✘ CaP-SNARK for simultaneous evaluation of many committed polynomials
  (based on the SDH and PKE assumptions in the RO Model)
- ✘ ZK: randomisation of ciphertexts & commited results of evaluation

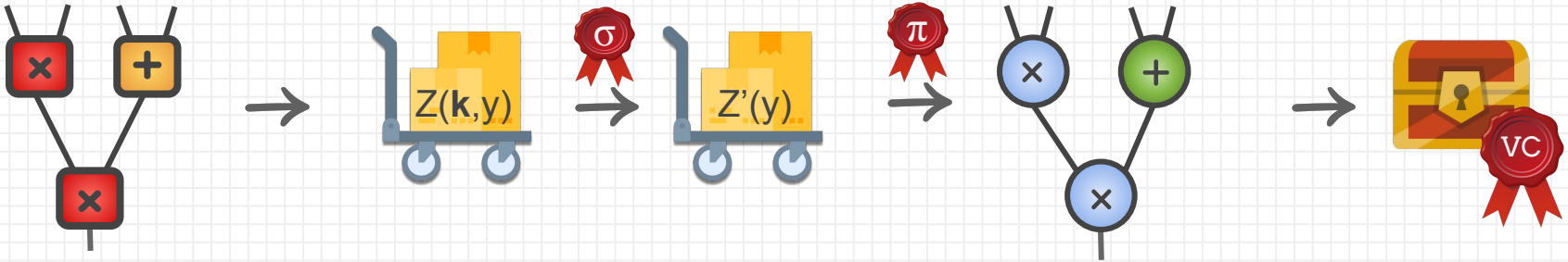
Review of contributions

Compactly Commit to Polynomials

ZK Proof for evaluation in random point **k**

CaP zk-SNARK for arithmetic circuit over scalars

Verifiable Computation with **Privacy**



Thank you!

eprint.iacr.org/2020/132

Credits

Special thanks to all those who made and released these resources for free:

- ✘ Presentation template by SlidesCarnival
- ✘ Illustrations by Disneyclips, Iconfinder and Flaticon