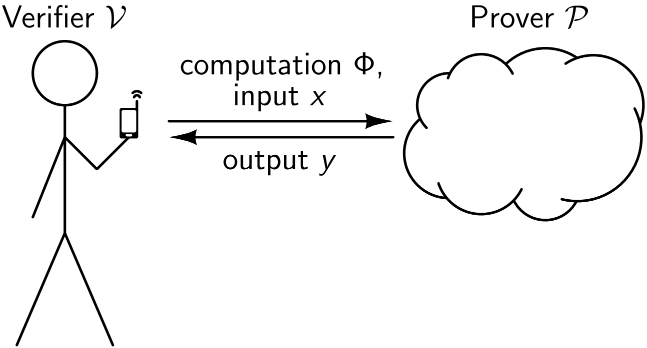
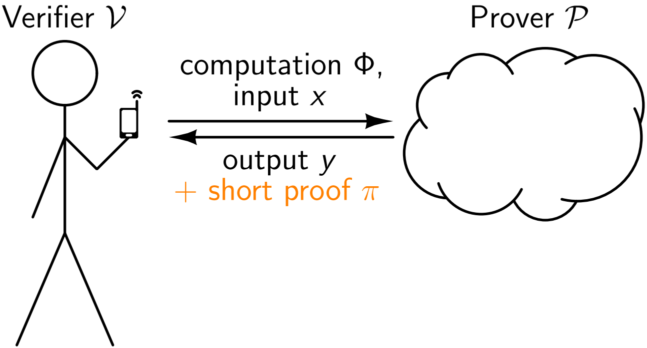
Practical proof systems: Implementations, applications, and next steps

Riad S. Wahby

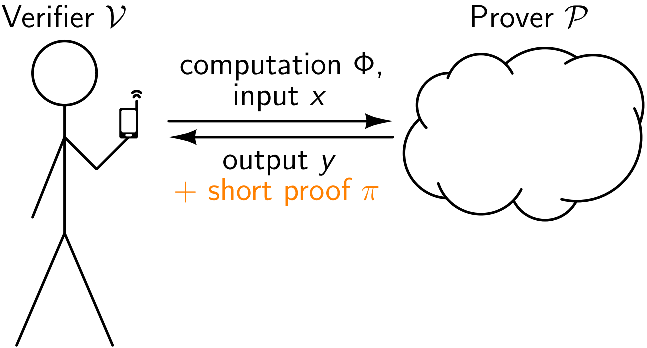
Stanford

September 23rd, 2019


Verifier $\mathcal{V}$
Prover $\mathcal{P}$
computation Φ,
input x
output y

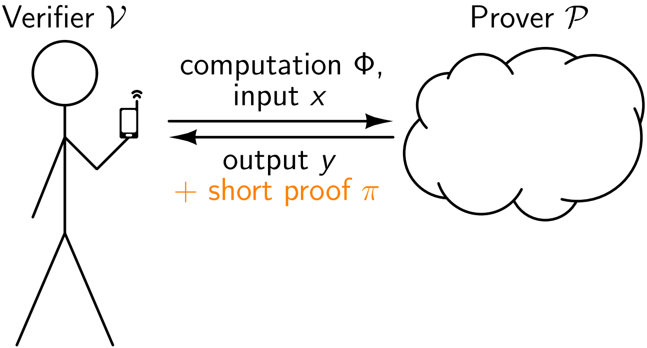


In general: Proof π convinces $\mathcal{V}$ that $y = \Phi(x)$.



In general: Proof π convinces $\mathcal{V}$ that $y = \Phi(x)$.

For zero knowledge: π convinces $\mathcal{V}$ that $\mathcal{P}$ knows *witness* w s.t. $y = \Phi(x, w)$, without revealing w.

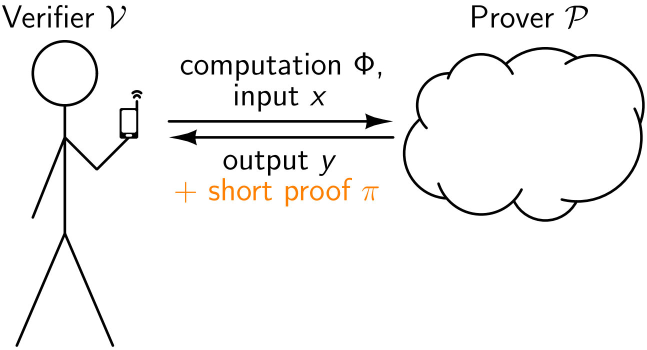


In general: Proof π convinces $\mathcal{V}$ that $y = \Phi(x)$.

For zero knowledge: π convinces $\mathcal{V}$ that $\mathcal{P}$ knows *witness* w s.t. $y = \Phi(x, w)$, without revealing w.

[Bab85, GMR85, BCC86, BGGHKMR90, BFLS91, FGLSS91, ALMSS92, AS92, Kil92, LFKN92, Sha92, Mic94, CD98, BG02, BS05, GOS06, BGHSV06, IKO07, GKR08, IKOS08, KR09, GGP10, Groth10, GLR11, Lip11, BCCT12, BCIOP13, GGPR13, PST13, Tha13, KRR14, BCCGP16, BCS16, Groth16, RRR16, BBC-GGI19, ...]

SBW11
CMT12
SMBW12
SVPBBW12
TRMP12
BCGTV13
BFRSBW13
BFR13
DFKP13
PGHR13
SBVBPW13
VSBW13
BCGGMTV14
BCTV14a
BCTV14b
FGP14
FL14
KPPSST14



BBFR15
CFHKKNPZ15
CTV15
KZMQCPPsS15
WSRHBW15
D-LFKP16
FFGKOP16
GMO16
NT16
WHGsW16
AHIV17
CDGORRSZ17
WJBsTWW17
ZGKPP17a
ZGKPP17b
BBBPWM18
KPS18
SAGL18
WTsTW18
WZCPS18
ZGKPP18
BAZB19
BBHR19
BCRSVW19
CFQ19
MBKM19
XZZPS19
...

In general: Proof π convinces $\mathcal{V}$ that $y = \Phi(x)$.

For zero knowledge: π convinces $\mathcal{V}$ that $\mathcal{P}$ knows *witness* w s.t. $y = \Phi(x, w)$, without revealing w.

[Bab85, GMR85, BCC86, BGGHKMR90, BFLS91, FGLSS91, ALMSS92, AS92, Kil92, LFKN92, Sha92, Mic94, CD98, BG02, BS05, GOS06, BGHSV06, IKO07, GKR08, IKOS08, KR09, GGP10, Groth10, GLR11, Lip11, BCCT12, BCIOP13, GGPR13, PST13, Tha13, KRR14, BCCGP16, BCS16, Groth16, RRR16, BBC-GGI19, ...]

Costs and desiderata

- $\mathcal{P}$ time
- $\mathcal{V}$ time
- communication cost / proof size

Costs and desiderata

- $\mathcal{P}$ time
- $\mathcal{V}$ time
- communication cost / proof size
- cryptographic assumptions

Costs and desiderata

- $\mathcal{P}$ time
- $\mathcal{V}$ time
- communication cost / proof size
- cryptographic assumptions
- trusted setup? per-Φ or universal?

Costs and desiderata

- $\mathcal{P}$ time
- $\mathcal{V}$ time
- communication cost / proof size
- cryptographic assumptions
- trusted setup? per-Φ or universal?
- interactive or non-interactive?

Costs and desiderata

- $\mathcal{P}$ time
- $\mathcal{V}$ time
- communication cost / proof size
- cryptographic assumptions
- trusted setup? per-Φ or universal?
- interactive or non-interactive?
- zero knowledge?

Costs and desiderata

- $\mathcal{P}$ time
- $\mathcal{V}$ time
- communication cost / proof size
- cryptographic assumptions
- trusted setup? per-Φ or universal?
- interactive or non-interactive?
- zero knowledge?
- model of computation / "expressiveness"

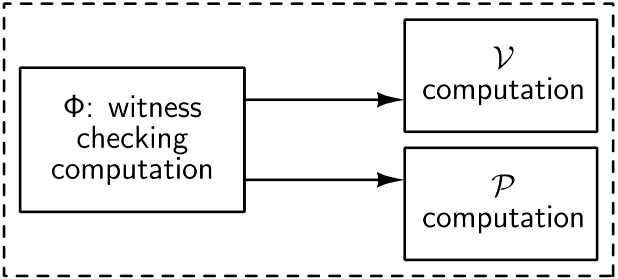
Costs and desiderata

- $\mathcal{P}$ time
- $\mathcal{V}$ time
- communication cost / proof size
- cryptographic assumptions
- trusted setup? per-Φ or universal?
- interactive or non-interactive?
- zero knowledge?
- model of computation / "expressiveness"

Bottom line: this is a huge tradeoff space!

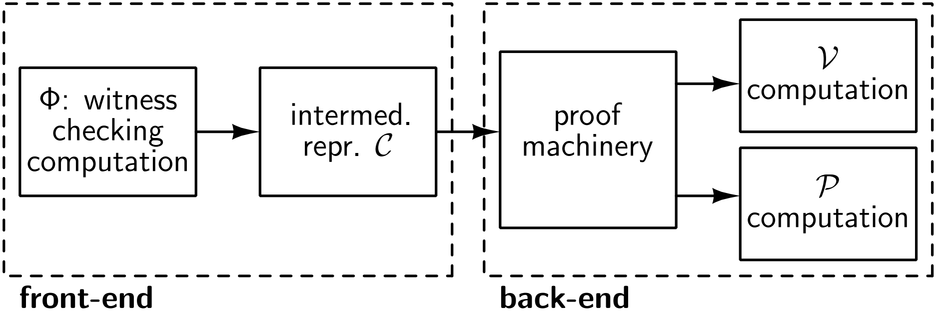
Proof systems pipeline

On input x, $\mathcal{P}$ convinces $\mathcal{V}$ that $y = \Phi(x, w)$
for a witness w that $\mathcal{P}$ knows



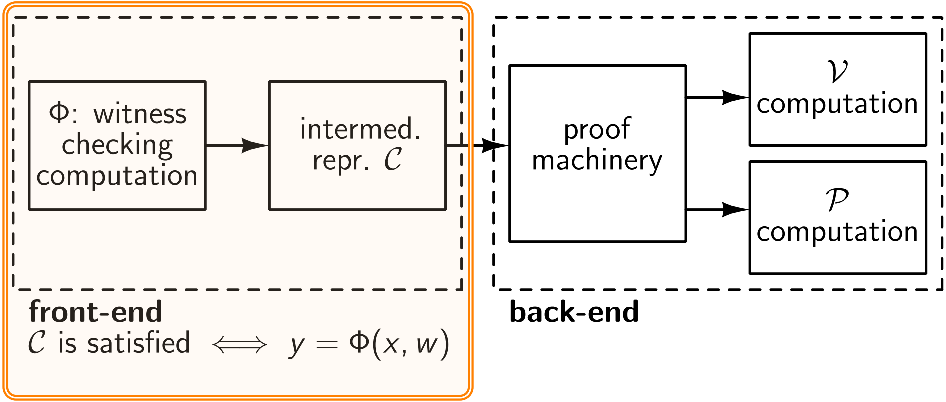
Proof systems pipeline

On input x, $\mathcal{P}$ convinces $\mathcal{V}$ that $y = \Phi(x, w)$
for a witness w that $\mathcal{P}$ knows



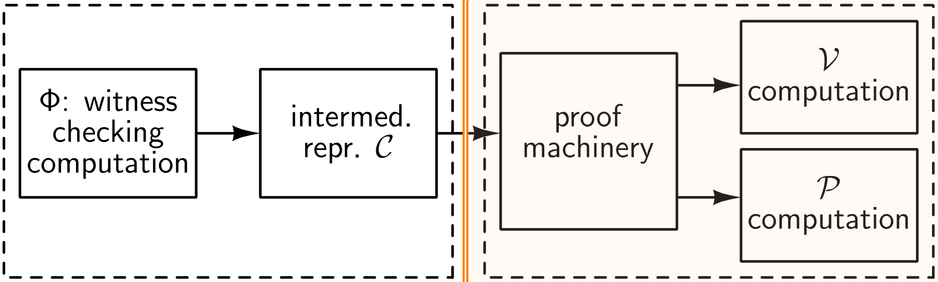
Proof systems pipeline

On input x, $\mathcal{P}$ convinces $\mathcal{V}$ that $y = \Phi(x, w)$
for a witness w that $\mathcal{P}$ knows



Proof systems pipeline

On input x, $\mathcal{P}$ convinces $\mathcal{V}$ that $y = \Phi(x, w)$
for a witness w that $\mathcal{P}$ knows



front-end
$\mathcal{C}$ is satisfied $\iff$ $y = \Phi(x, w)$

back-end
valid proof $\iff$ $\mathcal{C}$ is satisfied

Underlying machinery

Linear PCPs [IKO07,BBCGI19] and QAPs [GGPR13]

Pepper [SBW11,SMBW12], Ginger [SVPBBW12], Zaatar [SBVBPW13], Pinocchio [PGHR13], [BCGTV13], libSNARK [BCTV14a], [BCTV14b], ADSNARK [BBFR15], Geppetto [CFHKKNPZ15], [Gro16], . . .

Underlying machinery

Linear PCPs [IKO07,BBCGI19] and QAPs [GGPR13]

Pepper [SBW11,SMBW12], Ginger [SVPBBW12], Zaatar [SBVBPW13], Pinocchio [PGHR13], [BCGTV13], libSNARK [BCTV14a], [BCTV14b], ADSNARK [BBFR15], Geppetto [CFHKKNPZ15], [Gro16], . . .

IPs [GMR85,GKR08,Tha13], MIPs [BGKW88,BTVW14]

[CMT12], Giraffe [WJBsTWW17], Hyrax [WTsTW18], Spartan [Set19], (ZK)vSQL [ZGKPP17{a,b}], vRAM [ZGKPP18], Libra [XZZPS19]

Underlying machinery

Linear PCPs [IKO07,BBCGI19] and QAPs [GGPR13]

Pepper [SBW11,SMBW12], Ginger [SVPBBW12], Zaatar [SBVBPW13], Pinocchio [PGHR13], [BCGTV13], libSNARK [BCTV14a], [BCTV14b], ADSNARK [BBFR15], Geppetto [CFHKKNPZ15], [Gro16], ...

IPs [GMR85,GKR08,Tha13], MIPs [BGKW88,BTVW14]

[CMT12], Giraffe [WJBsTWW17], Hyrax [WTsTW18], Spartan [Set19], (ZK)vSQL [ZGKPP17{a,b}], vRAM [ZGKPP18], Libra [XZZPS19]

MPC-in-the-head [IKOS08]

ZKBoo [GMO16], ZKB++ [CDGORRSZ17], Ligero [AHIV17]

Underlying machinery

Linear PCPs [IKO07,BBCGI19] and QAPs [GGPR13]

Pepper [SBW11,SMBW12], Ginger [SVPBBW12], Zaatar [SBVBPW13], Pinocchio [PGHR13], [BCGTV13], libSNARK [BCTV14a], [BCTV14b], ADSNARK [BBFR15], Geppetto [CFHKKNPZ15], [Gro16], ...

IPs [GMR85,GKR08,Tha13], MIPs [BGKW88,BTVW14]

[CMT12], Giraffe [WJBsTWW17], Hyrax [WTsTW18], Spartan [Set19], (ZK)vSQL [ZGKPP17{a,b}], vRAM [ZGKPP18], Libra [XZZPS19]

MPC-in-the-head [IKOS08]

ZKBoo [GMO16], ZKB++ [CDGORRSZ17], Ligero [AHIV17]

Generalized Σ-protocols [Sch89,CP92,Oka92,Cra97]

[BCCGP16], Bulletproofs [BBBPWM18]

Underlying machinery

Linear PCPs [IKO07,BBCGI19] and QAPs [GGPR13]

Pepper [SBW11,SMBW12], Ginger [SVPBBW12], Zaatar [SBVBPW13], Pinocchio [PGHR13], [BCGTV13], libSNARK [BCTV14a], [BCTV14b], ADSNARK [BBFR15], Geppetto [CFHKKNPZ15], [Gro16], ...

IPs [GMR85,GKR08,Tha13], MIPs [BGKW88,BTVW14]

[CMT12], Giraffe [WJBsTWW17], Hyrax [WTsTW18], Spartan [Set19], (ZK)vSQL [ZGKPP17{a,b}], vRAM [ZGKPP18], Libra [XZZPS19]

MPC-in-the-head [IKOS08]

ZKBoo [GMO16], ZKB++ [CDGORRSZ17], Ligero [AHIV17]

Generalized Σ-protocols [Sch89,CP92,Oka92,Cra97]

[BCCGP16], Bulletproofs [BBBPWM18]

IOPs [KR08,BCS16,RRR16] + Arguments [Kil92,Mic94]

Aurora [BCRSVW19], STARK [BBHR19]

Underlying machinery

Linear PCPs [IKO07,BBCGI19] and QAPs [GGPR13]

Pepper [SBW11,SMBW12], Ginger [SVPBBW12], Zaatar [SBVBPW13], Pinocchio [PGHR13], [BCGTV13], libSNARK [BCTV14a], [BCTV14b], ADSNARK [BBFR15], Geppetto [CFHKKNPZ15], [Gro16], ...

IPs [GMR85,GKR08,Tha13], MIPs [BGKW88,BTVW14]

[CMT12], Giraffe [WJBsTWW17], Hyrax [WTsTW18], Spartan [Set19], (ZK)vSQL [ZGKPP17{a,b}], vRAM [ZGKPP18], Libra [XZZPS19]

MPC-in-the-head [IKOS08]

ZKBoo [GMO16], ZKB++ [CDGORRSZ17], Ligero [AHIV17]

Generalized Σ-protocols [Sch89,CP92,Oka92,Cra97]

[BCCGP16], Bulletproofs [BBBPWM18]

IOPs [KR08,BCS16,RRR16] + Arguments [Kil92,Mic94]

Aurora [BCRSVW19], STARK [BBHR19]

Polynomial delegation [KZG10,PST13]

Sonic [MBKM19]

Underlying machinery

Linear PCPs [IKO07,BBCGI19] and QAPs [GGPR13]

Pepper [SBW11,SMBW12], Ginger [SVPBBW12], Zaatar [SBVBPW13], Pinocchio [PGHR13], [BCGTV13], libSNARK [BCTV14a], [BCTV14b], ADSNARK [BBFR15], Geppetto [CFHKKNPZ15], [Gro16], . . .

IPs [GMR85,GKR08,Tha13], MIPs [BGKW88,BTVW14]

[CMT12], Giraffe [WJBsTWW17], Hyrax [WTsTW18], Spartan [Set19], (ZK)vSQL [ZGKPP17{a,b}], vRAM [ZGKPP18], Libra [XZZPS19]

MPC-in-the-head [IKOS08]

ZKBoo [GMO16], ZKB++ [CDGORRSZ17], Ligero [AHIV17]

Generalized Σ-protocols [Sch89,CP92,Oka92,Cra97]

[BCCGP16], Bulletproofs [BBBPWM18]

IOPs [KR08,BCS16,RRR16] + Arguments [Kil92,Mic94]

Aurora [BCRSVW19], STARK [BBHR19]

Polynomial delegation [KZG10,PST13]

Sonic [MBKM19]

Underlying machinery

Linear PCPs [IKO07,BBCGI19] and QAPs [GGPR13]

Pepper [SBW11,SMBW12], Ginger [SVPBBW12], Zaatar [SBVBPW13], Pinocchio [PGHR13], [BCGTV13], libSNARK [BCTV14a], [BCTV14b], ADSNARK [BBFR15], Geppetto [CFHKKNPZ15], [Gro16], ...

IPs [GMR85,GKR08,Tha13], MIPs [BGKW88,BTVW14]

[CMT12], Giraffe [WJBsTWW17], Hyrax [WTsTW18], Spartan [Set19], (ZK)vSQL [ZGKPP17{a,b}], vRAM [ZGKPP18], Libra [XZZPS19]

MPC-in-the-head [IKOS08]

ZKBoo [GMO16], ZKB++ [CDGORRSZ17], Ligero [AHIV17]

Generalized Σ-protocols [Sch89,CP92,Oka92,Cra97]

[BCCGP16], Bulletproofs [BBBPWM18]

IOPs [KR08,BCS16,RRR16] + Arguments [Kil92,Mic94]

Aurora [BCRSVW19], STARK [BBHR19]

Polynomial delegation [KZG10,PST13]

Sonic [MBKM19]

Underlying machinery

Linear PCPs [IKO07,BBCGI19] and QAPs [GGPR13]

Pepper [SBW11,SMBW12], Ginger [SVPBBW12], Zaatar [SBVBPW13], Pinocchio [PGHR13], [BCGTV13], libSNARK [BCTV14a], [BCTV14b], ADSNARK [BBFR15], Geppetto [CFHKKNPZ15], [Gro16], ...

IPs [GMR85,GKR08,Tha13], MIPs [BGKW88,BTVW14]

[CMT12], Giraffe [WJBsTWW17], Hyrax [WTsTW18], Spartan [Set19], (ZK)vSQL [ZGKPP17{a,b}], vRAM [ZGKPP18], Libra [XZZPS19]

MPC-in-the-head [IKOS08]

ZKBoo [GMO16], ZKB++ [CDGORRSZ17], Ligero [AHIV17]

Generalized Σ-protocols [Sch89,CP92,Oka92,Cra97]

[BCCGP16], Bulletproofs [BBBPWM18]

IOPs [KR08,BCS16,RRR16] + Arguments [Kil92,Mic94]

Aurora [BCRSVW19], STARK [BBHR19]

Polynomial delegation [KZG10,PST13]

Sonic [MBKM19]

Setup and cryptographic assumptions

No trusted setup

Bulletproofs [BBBPWM18], Hyrax [WTsTW18], Spartan [Set19]

ZKBoo [GMO16], ZKB++ [CDGORRSZ17], Ligero [AHIV17], Aurora [BCRSVW19], STARK [BBHR19]

[CMT12,Tha13], Clover [BTVW14], Giraffe [WJBsTWW17]

Setup and cryptographic assumptions

No trusted setup

Bulletproofs [BBBPWM18], Hyrax [WTsTW18], Spartan [Set19]

ZKBoo [GMO16], ZKB++ [CDGORRSZ17], Ligero [AHIV17], Aurora [BCRSVW19], STARK [BBHR19]

[CMT12,Tha13], Clover [BTVW14], Giraffe [WJBsTWW17]

Universal trusted setup

[BCGTV13], libSNARK/vnTinyRAM [BCTV14a], [BCTV14b], (ZK)vSQL [ZGKPP17{a,b}], vRAM [ZGKPP18], Libra [XZZPS19]

Sonic [MBKM19]

Setup and cryptographic assumptions

No trusted setup

Bulletproofs [BBBPWM18], Hyrax [WTsTW18], Spartan [Set19]

ZKBoo [GMO16], ZKB++ [CDGORRSZ17], Ligero [AHIV17], Aurora [BCRSVW19], STARK [BBHR19]

[CMT12,Tha13], Clover [BTVW14], Giraffe [WJBsTWW17]

Universal trusted setup

[BCGTV13], libSNARK/vnTinyRAM [BCTV14a], [BCTV14b], (ZK)vSQL [ZGKPP17{a,b}], vRAM [ZGKPP18], Libra [XZZPS19]

Sonic [MBKM19]

Per-Φ trusted setup

Pepper, Ginger, Zaatar [SBW11,SMBW12,SVPBBW12,SBVBPW13]

Pinocchio [PGHR13], libSNARK [BCTV14a], ADSNARK [BBFR15], Geppetto [CFHKKNPZ15], . . .

Setup and cryptographic assumptions

No trusted setup

Bulletproofs [BBBPWM18], Hyrax [WTsTW18], Spartan [Set19]

ZKBoo [GMO16], ZKB++ [CDGORRSZ17], Ligero [AHIV17], Aurora [BCRSVW19], STARK [BBHR19]

[CMT12,Tha13], Clover [BTVW14], Giraffe [WJBsTWW17] (unconditional)

Universal trusted setup

[BCGTV13], libSNARK/vnTinyRAM [BCTV14a], [BCTV14b], (ZK)vSQL [ZGKPP17{a,b}], vRAM [ZGKPP18], Libra [XZZPS19]

Sonic [MBKM19]

Per-Φ trusted setup

Pepper, Ginger, Zaatar [SBW11,SMBW12,SVPBBW12,SBVBPW13]

Pinocchio [PGHR13], libSNARK [BCTV14a], ADSNARK [BBFR15], Geppetto [CFHKKNPZ15], . . .

Setup and cryptographic assumptions

No trusted setup

Bulletproofs [BBBPWM18], Hyrax [WTsTW18], Spartan [Set19]

ZKBoo [GMO16], ZKB++ [CDGORRSZ17], Ligero [AHIV17], Aurora [BCRSVW19], STARK [BBHR19] (CRHF/ROM)

[CMT12,Tha13], Clover [BTVW14], Giraffe [WJBsTWW17] (unconditional)

Universal trusted setup

[BCGTV13], libSNARK/vnTinyRAM [BCTV14a], [BCTV14b], (ZK)vSQL [ZGKPP17{a,b}], vRAM [ZGKPP18], Libra [XZZPS19]

Sonic [MBKM19]

Per-Φ trusted setup

Pepper, Ginger, Zaatar [SBW11,SMBW12,SVPBBW12,SBVBPW13]

Pinocchio [PGHR13], libSNARK [BCTV14a], ADSNARK [BBFR15], Geppetto [CFHKKNPZ15], . . .

Setup and cryptographic assumptions

No trusted setup

Bulletproofs [BBBPWM18], Hyrax [WTsTW18], Spartan [Set19] (DH/ROM)

ZKBoo [GMO16], ZKB++ [CDGORRSZ17], Ligero [AHIV17], Aurora [BCRSVW19], STARK [BBHR19] (CRHF/ROM)

[CMT12,Tha13], Clover [BTVW14], Giraffe [WJBsTWW17] (unconditional)

Universal trusted setup

[BCGTV13], libSNARK/vnTinyRAM [BCTV14a], [BCTV14b], (ZK)vSQL [ZGKPP17{a,b}], vRAM [ZGKPP18], Libra [XZZPS19]

Sonic [MBKM19]

Per-Φ trusted setup

Pepper, Ginger, Zaatar [SBW11,SMBW12,SVPBBW12,SBVBPW13] (DH)

Pinocchio [PGHR13], libSNARK [BCTV14a], ADSNARK [BBFR15], Geppetto [CFHKKNPZ15], . . .

Setup and cryptographic assumptions

No trusted setup

Bulletproofs [BBBPWM18], Hyrax [WTsTW18], Spartan [Set19] (DH/ROM)

ZKBoo [GMO16], ZKB++ [CDGORRSZ17], Ligero [AHIV17], Aurora [BCRSVW19], STARK [BBHR19] (CRHF/ROM)

[CMT12,Tha13], Clover [BTVW14], Giraffe [WJBsTWW17] (unconditional)

Universal trusted setup

[BCGTV13], libSNARK/vnTinyRAM [BCTV14a], [BCTV14b], (ZK)vSQL [ZGKPP17{a,b}], vRAM [ZGKPP18], Libra [XZZPS19] (KoE)

Sonic [MBKM19]

Per-Φ trusted setup

Pepper, Ginger, Zaatar [SBW11,SMBW12,SVPBBW12,SBVBPW13] (DH)

Pinocchio [PGHR13], libSNARK [BCTV14a], ADSNARK [BBFR15], Geppetto [CFHKKNPZ15], . . . (KoE)

Setup and cryptographic assumptions

No trusted setup

Bulletproofs [BBBPWM18], Hyrax [WTsTW18], Spartan [Set19] (DH/ROM)

ZKBoo [GMO16], ZKB++ [CDGORRSZ17], Ligero [AHIV17], Aurora [BCRSVW19], STARK [BBHR19] (CRHF/ROM)

[CMT12,Tha13], Clover [BTVW14], Giraffe [WJBsTWW17] (unconditional)

Universal trusted setup

[BCGTV13], libSNARK/vnTinyRAM [BCTV14a], [BCTV14b], (ZK)vSQL [ZGKPP17{a,b}], vRAM [ZGKPP18], Libra [XZZPS19] (KoE)

Sonic [MBKM19] (algebraic group model)

Per-Φ trusted setup

Pepper, Ginger, Zaatar [SBW11,SMBW12,SVPBBW12,SBVBPW13] (DH)

Pinocchio [PGHR13], libSNARK [BCTV14a], ADSNARK [BBFR15], Geppetto [CFHKKNPZ15], ... (KoE)

Setup and cryptographic assumptions

No trusted setup

Bulletproofs [BBBPWM18], Hyrax [WTsTW18], Spartan [Set19] (DH/ROM)

ZKBoo [GMO16], ZKB++ [CDGORRSZ17], Ligero [AHIV17], Aurora [BCRSVW19], STARK [BBHR19] (CRHF/ROM)

[CMT12,Tha13], Clover [BTVW14], Giraffe [WJBsTWW17] (unconditional)

Universal trusted setup

[BCGTV13], libSNARK/vnTinyRAM [BCTV14a], [BCTV14b], (ZK)vSQL [ZGKPP17{a,b}], vRAM [ZGKPP18], Libra [XZZPS19] (KoE)

Sonic [MBKM19] (algebraic group model)

Per-Φ trusted setup

Pepper, Ginger, Zaatar [SBW11,SMBW12,SVPBBW12,SBVBPW13] (DH)
(setup amortizes over a batch)

Pinocchio [PGHR13], libSNARK [BCTV14a], ADSNARK [BBFR15], Geppetto [CFHKKNPZ15], ... (KoE)
(setup amortizes forever)

Setup and cryptographic assumptions

No trusted setup

Bulletproofs [BBBPWM18], Hyrax [WTsTW18], Spartan [Set19] (DH/ROM)

ZKBoo [GMO16], ZKB++ [CDGORRSZ17], Ligero [AHIV17], Aurora [BCRSVW19], STARK [BBHR19] (CRHF/ROM)

[CMT12,Tha13], Clover [BTVW14], Giraffe [WJBsTWW17] (unconditional)

Universal trusted setup

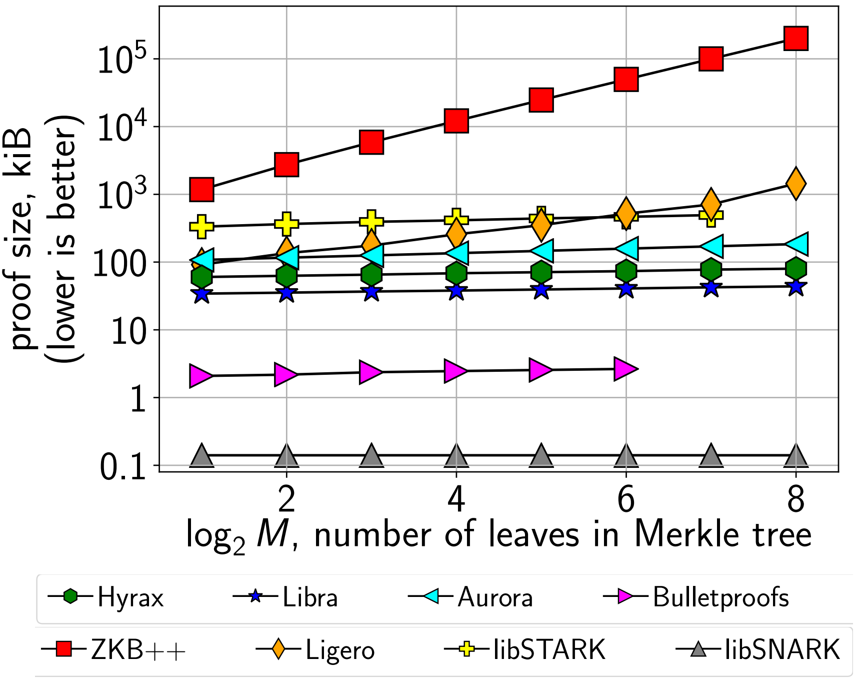
[BCGTV13], libSNARK/vnTinyRAM [BCTV14a], [BCTV14b], (ZK)vSQL [ZGKPP17{a,b}], vRAM [ZGKPP18], Libra [XZZPS19] (KoE)

Sonic [MBKM19] (algebraic group model)

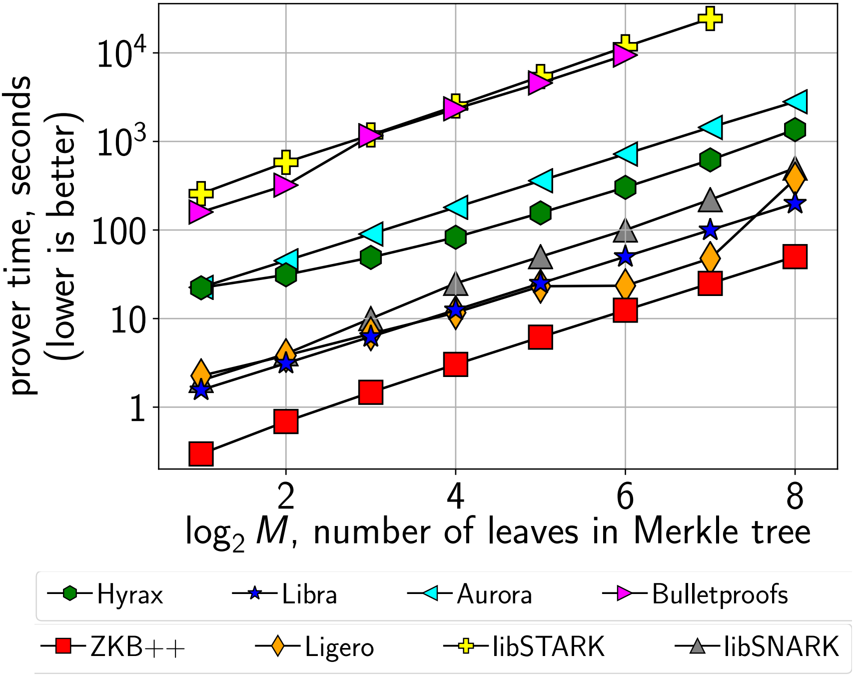
Per-Φ trusted setup

Pepper, Ginger, Zaatar [SBW11,SMBW12,SVPBBW12,SBVBPW13] (DH)
(setup amortizes over a batch)

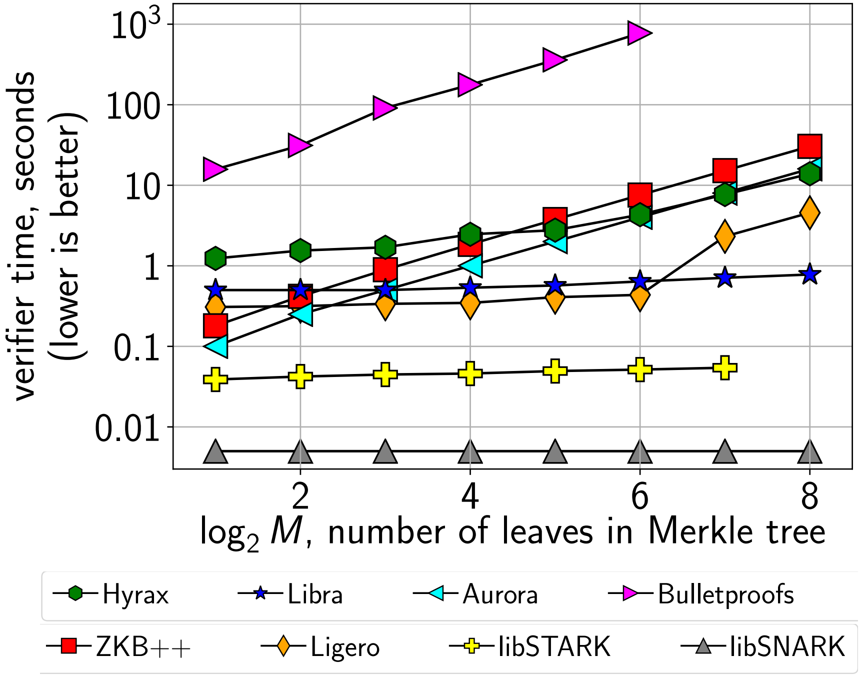
Pinocchio [PGHR13], libSNARK [BCTV14a], ADSNARK [BBFR15], Geppetto [CFHKKNPZ15], ... (KoE)
(setup amortizes forever)



Numbers are from Hyrax [WTsTW18], except Libra, Aurora, and libSNARK, which are from Libra [XZZPS19]



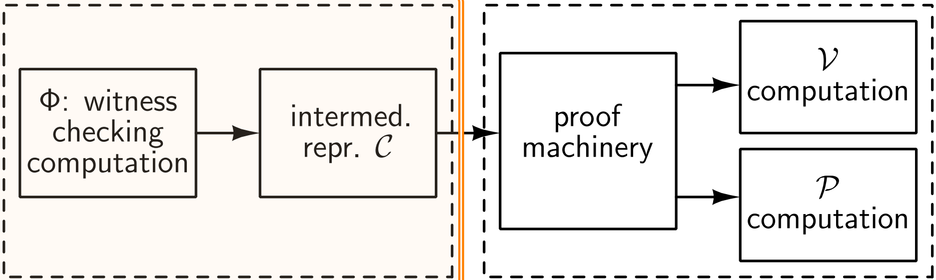
Numbers are from Hyrax [WTsTW18], except Libra, Aurora, and libSNARK, which are from Libra [XZZPS19]



Numbers are from Hyrax [WTsTW18], except Libra, Aurora, and libSNARK, which are from Libra [XZZPS19]

Proof system construction

On input x, $\mathcal{P}$ convinces $\mathcal{V}$ that $y = \Phi(x, w)$
for a witness w that $\mathcal{P}$ knows



front-end
$\mathcal{C}$ is satisfied $\iff y = \Phi(x, w)$

back-end
valid proof $\iff \mathcal{C}$ is satisfied

Representing Φ for execution on the back-end

"CPU": run Φ on unrolled FSM

fetch-decode-execute step 1	fetch-decode-execute step 2	. . .	fetch-decode-execute step T
CPU state: pc, regs, . . .	CPU state: pc, regs, . . .		CPU state: pc, regs, . . .

[BCGTV13,BCTV14a,BCTV14b,CTV15,ZGKPP18,BBHR19]

Representing Φ for execution on the back-end

“CPU”: run Φ on unrolled FSM

fetch-decode-execute step 1	fetch-decode-execute step 2	. . .	fetch-decode-execute step T
CPU state: pc, regs, . . .	CPU state: pc, regs, . . .		CPU state: pc, regs, . . .

[BCGTV13,BCTV14a,BCTV14b,CTV15,ZGKPP18,BBHR19]

“FPGA”: translate Φ directly to AC or constraints

```
if (i >= 5)
    i = i + 1;
else
    i = i * 2;
```

$\Longrightarrow$

```
i1 = i0 + 1;
i2 = i0 * 2;
i3 = (i0 >= 5) ?
        i1 : i2;
```

[. . . , SVPBBW12, BFRSBW13, SBVBPW13, PGHR13, VSBW13, BBFR15, CFHKKNPZ15, KZMQCPPsS15, WSRHBW15, BCCGP16, BBBPWM18, KPS18, BCRSVW19, MBKM19, Circom, Bellman, . . .]

Representing Φ for execution on the back-end

"CPU": run Φ on unrolled FSM

fetch-decode-execute step 1	fetch-decode-execute step 2	. . .	fetch-decode-execute step T
CPU state: pc, regs, . . .	CPU state: pc, regs, . . .		CPU state: pc, regs, . . .

[BCGTV13,BCTV14a,BCTV14b,CTV15,ZGKPP18,BBHR19]

"FPGA": translate Φ directly to AC or constraints

```
if (i >= 5)
    i = i + 1;
else
    i = i * 2;
```

$\Longrightarrow$

```
i1 = i0 + 1;
i2 = i0 * 2;
i3 = (i0 >= 5) ?
        i1 : i2;
```

[. . . , SVPBBW12, BFRSBW13, SBVBPW13, PGHR13, VSBW13, BBFR15, CFHKKNPZ15, KZMQCPPsS15, WSRHBW15, BCCGP16, BBBPWM18, KPS18, BCRSVW19, MBKM19, Circom, Bellman, . . .]

☞[GKR08]-derived systems need a *low depth* circuit:
[CMT12, WHGsW16, WJBsTWW17, WTsTW18, XZZPS19]

Representing Φ for execution on the back-end

"CPU": run Φ on unrolled FSM

```
// assume i0 is k+1 bits
i4 = i0 - 5;
// prover supplies i4_0, ..., i4_k
assert (i4 - i4_0 - 2 * i4_1 - ... - 2^k * i4_k == 0);
assert (i4_0 * (1 - i4_0) == 0);
...
assert (i4_{k-1} * (1 - i4_{k-1}) == 0);
assert (i4_k == 0);
```

nts

```
if (i >= 5)
    i = i + 1;
else
    i = i * 2;
```

$\Longrightarrow$

```
i1 = i0 + 1;
i2 = i0 * 2;
i3 = (i0 >= 5) ?
          i1 : i2;
```

[..., SVPBBW12, BFRSBW13, SBVBPW13, PGHR13, VSBW13, BBFR15, CFHKKNPZ15, KZMQCPPsS15, WSRHBW15, BCCGP16, BBBPWM18, KPS18, BCRSVW19, MBKM19, Circom, Bellman, ...]

☞[GKR08]-derived systems need a *low depth* circuit:
[CMT12, WHGsW16, WJBsTWW17, WTsTW18, XZZPS19]

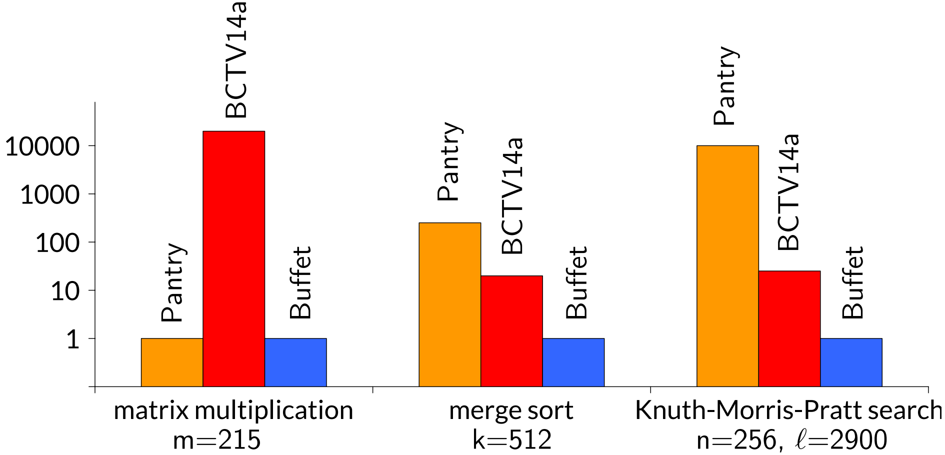
Performance vs. expressiveness

costs	special purpose	pure	stateful	general control flow
lower	[Tha13] vSQL [ZGKPP17] Bellman gadgetlib [BCTV14a] LegoSNARK [CFQ19] c0c0 [KZMQCPPsS15]	Giraffe [WJBsTWW17] Allspice [VSBW13] Zaatar [SBVBPW13] Pinocchio [PGHR13] Circom Ginger [SVPBBW12] Pepper [SMBW12]	xJsnark [KPS18] Geppetto [CFHKKNPZ15] ADSNARK [BBFR15] Pantry [BFRSBW13]	vRAM [ZGKPP18] Buffet [WSRHBW15] STARK [BBHR19]
				(vn)TinyRAM [BCTV14a] [BCGTV13]
higher				[BCTV14b] [CTV15]

better

Front-end comparison

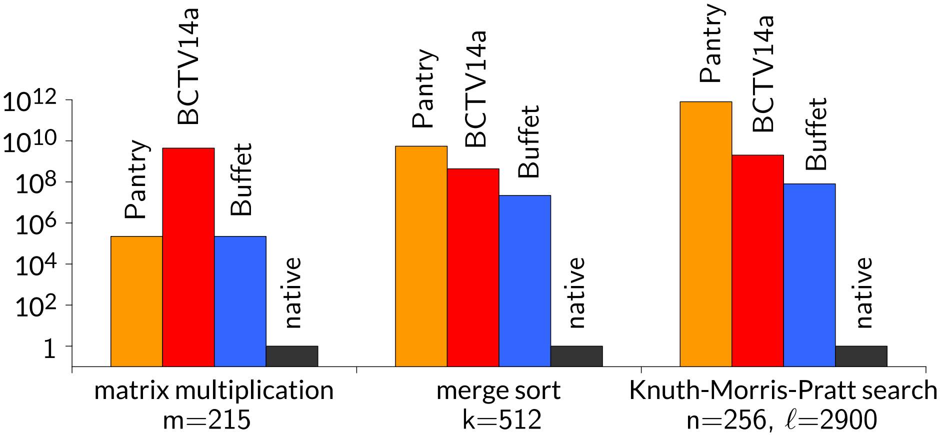
Extrapolated $\mathcal{P}$ execution time, normalized to Buffet



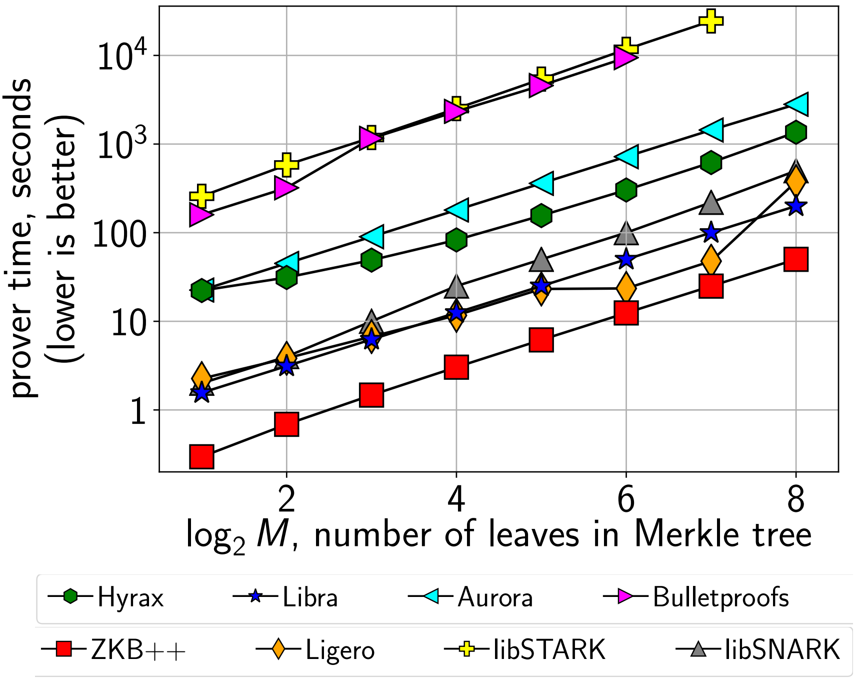
- xJsnark [KPS18] improves upon Buffet by up to ≈3×
- vRAM [ZGKPP18] (builds on and refines [Tha13] back-end) is ≈22× faster than Buffet for matmult, comparable otherwise

Reality check

Extrapolated $\mathcal{P}$ execution time, normalized to native execution



- xJsnark [KPS18] improves upon Buffet by up to $\approx 3\times$
- vRAM [ZGKPP18] (builds on and refines [Tha13] back-end) is $\approx 22\times$ faster than Buffet for matmult, comparable otherwise



Numbers are from Hyrax [WTsTW18], except Libra, Aurora, and libSNARK, which are from Libra [XZZPS19]

Reality check 2: reachable problem sizes

For $\approx 10^7$ gates, $\mathcal{P}$ needs $\approx$16–32 GiB of RAM.

Limiting computations to these sizes yields:

	Pantry	**BCTV14a**	**Buffet**
matrix multiplication $m \times m$	215	7	215
merge sort k elements	8	32	512
Knuth-Morris-Pratt search needle length $= n$ haystack length $= \ell$	$n = 4$, $\ell = 8$	$n = 16$, $\ell = 160$	$n = 256$, $\ell = 2900$

☞ vRAM [ZGKPP18] increases reachable sizes by $\approx 10\times$

DIZK [WZCPS18]: distributing $\mathcal{P}$'s workload

Idea: run $\mathcal{P}$ as a distributed computation

DIZK [WZCPS18]: distributing $\mathcal{P}$'s workload

Idea: run $\mathcal{P}$ as a distributed computation

Challenge: need to compute gigantic FFT!
(among others)

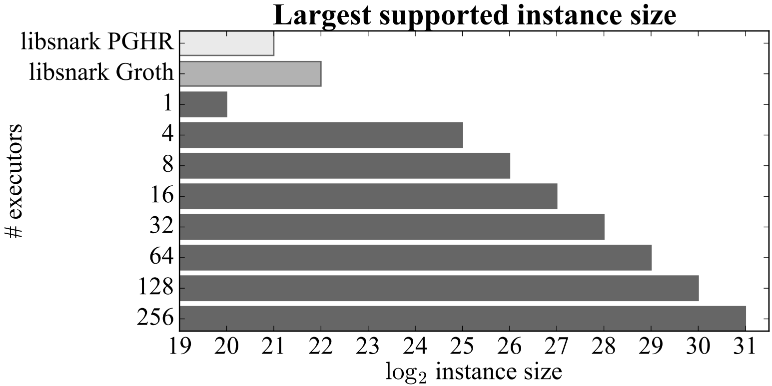
DIZK [WZCPS18]: distributing $\mathcal{P}$'s workload

Idea: run $\mathcal{P}$ as a distributed computation

Challenge: need to compute gigantic FFT!
(among others)

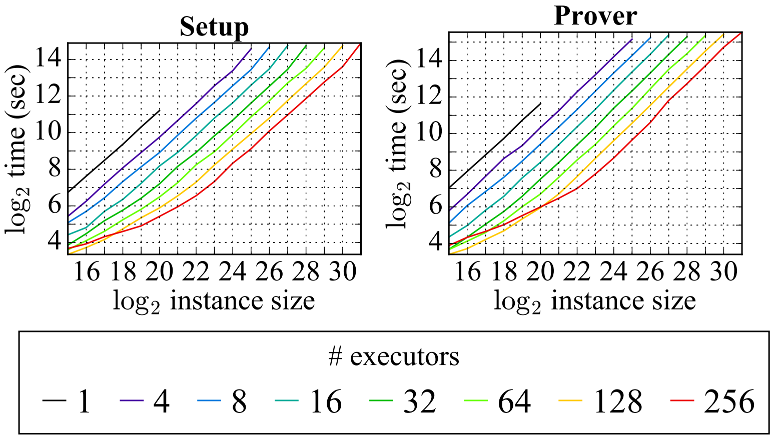
☞[Sze11]: converts size-n FFT to two $\sqrt{n}$-sized batches of $\sqrt{n}$-sized tasks

DIZK: 100× larger instances



[WZCPS18, Fig. 4]

DIZK: 100× faster execution



[WZCPS18, Fig. 5]

Cryptocurrencies!

ZCash (following [BCGGMTV14]) uses ZK for

- anonymity: no one knows who you are
- privacy: transaction values are hidden

Cryptocurrencies!

ZCash (following [BCGGMTV14]) uses ZK for

anonymity: no one knows who you are

privacy: transaction values are hidden

Confidential Transactions (via [BBBPWM18])

Cryptocurrencies!

ZCash (following [BCGGMTV14]) uses ZK for

anonymity: no one knows who you are

privacy: transaction values are hidden

Confidential Transactions (via [BBBPWM18])

CODA (via [CT12,BCTV14b])

☞ constant-sized blockchain via recursive proof composition

Cryptocurrencies!

ZCash (following [BCGGMTV14]) uses ZK for

anonymity: no one knows who you are

privacy: transaction values are hidden

Confidential Transactions (via [BBBPWM18])

CODA (via [CT12,BCTV14b])

☞ constant-sized blockchain via recursive proof composition

Private airdrops [BJPW19] (ePrint soon)

free money from the internet using existing credentials (e.g., GitHub) without revealing your identity

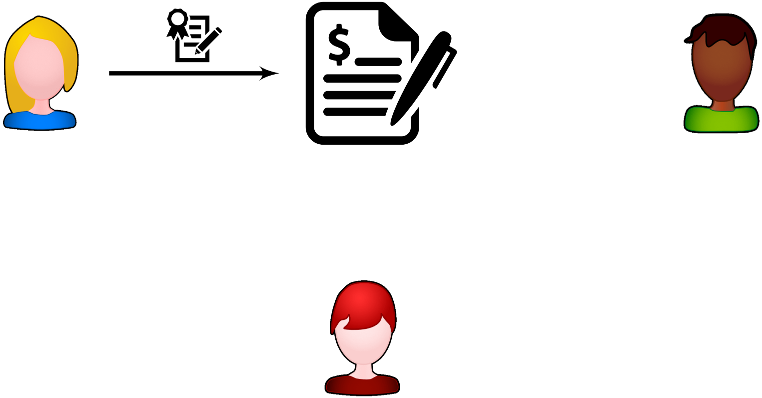
☞ *not* a general-purpose proof system!

Let's build a bank out of a smart contract!

Roll_up

https://github.com/barryWhiteHat/roll_up

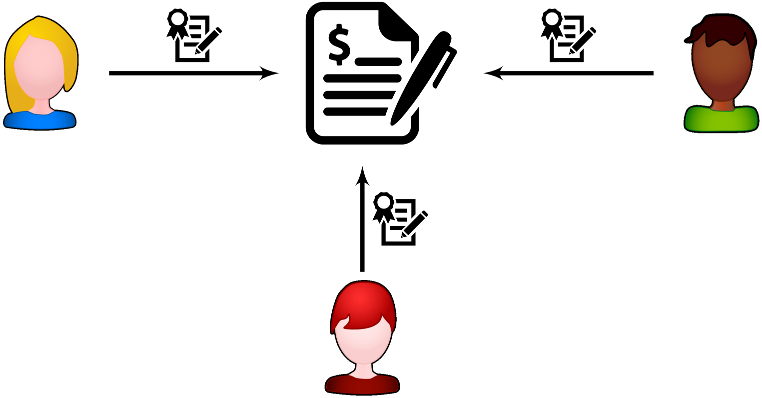
Let's build a bank out of a smart contract!



Roll_up

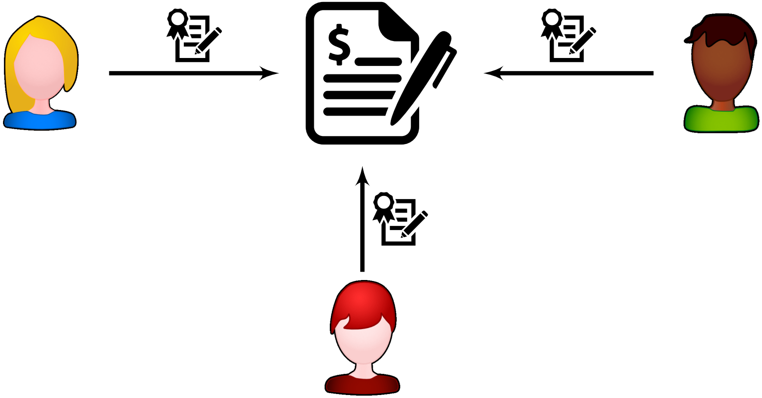
https://github.com/barryWhiteHat/roll_up

Let's build a bank out of a smart contract!



Roll_up https://github.com/barryWhiteHat/roll_up

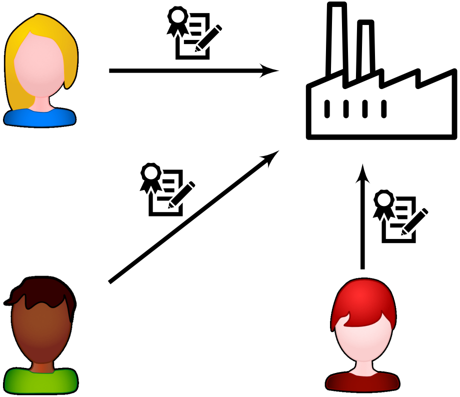
Let's build a bank out of a smart contract!



Issues: on-chain work and data cost $$$!; slow!

Roll_up https://github.com/barryWhiteHat/roll_up

Let's build a bank out of a smart contract!

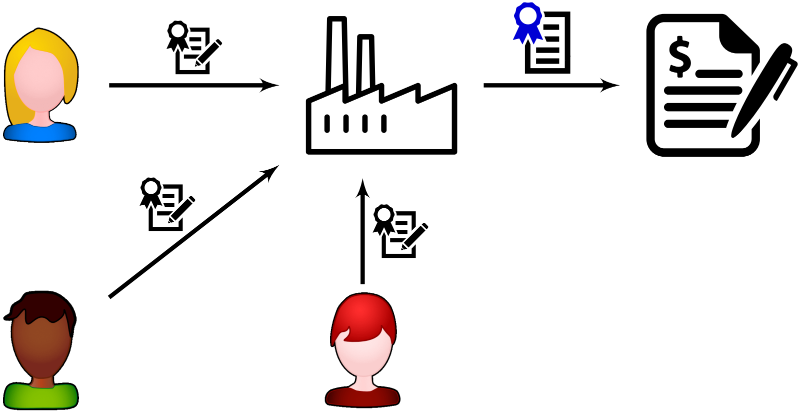


Idea: use an off-chain, *untrusted* aggregator

Roll_up

https://github.com/barryWhiteHat/roll_up

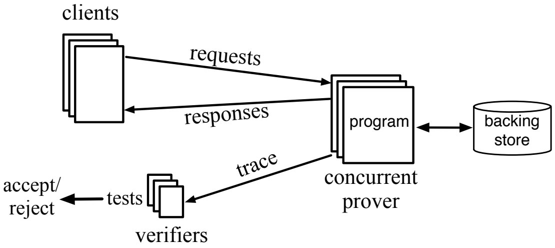
Let's build a bank out of a smart contract!



Idea: use an off-chain, *untrusted* aggregator to prove validity of a batch of transactions

Spice [SAGL18]: verifiable *concurrent* services (in ZK)

(e.g., a cloud-hosted wallet service.)



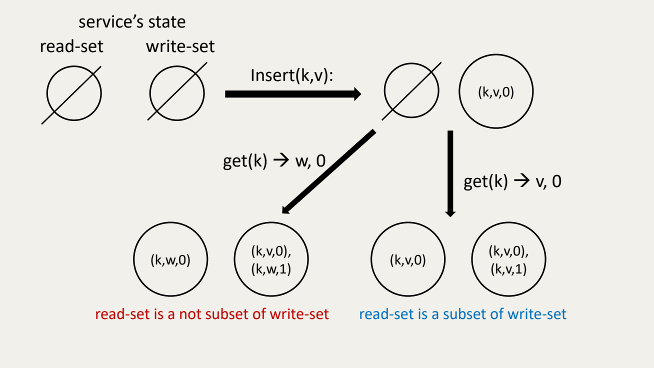
[SAGL18, Fig. 1]

Issue: need verifiable storage with concurrency

Spice [SAGL18]: verifiable *concurrent* services (in ZK)

(e.g., a cloud-hosted wallet service.)

Idea: adapt primitives from memory checking literature [BEGKN91,CDDGE03,AEKKMPR17]



(source: Srinath's talk)

Spice [SAGL18]: verifiable *concurrent* services (in ZK)

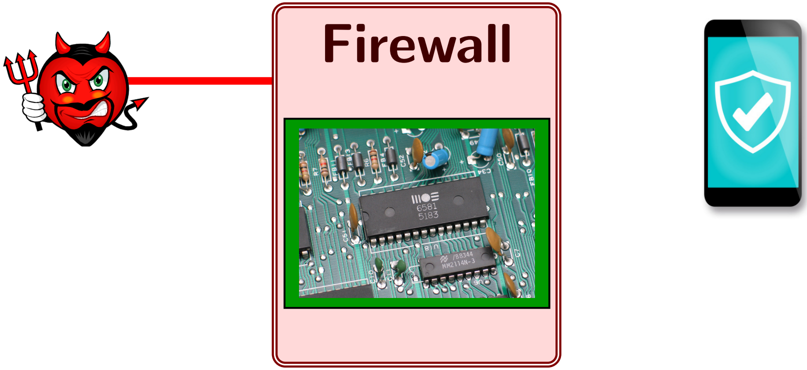
(e.g., a cloud-hosted wallet service.)

Performance results:

	`get`	`put`
Pantry	0.078	0.039
Pantry+Jubjub	0.153	0.076
Geppetto	0.002	0.002
Spice (1-thread)	3.6	3.6
Spice (512-threads)	1366	1370

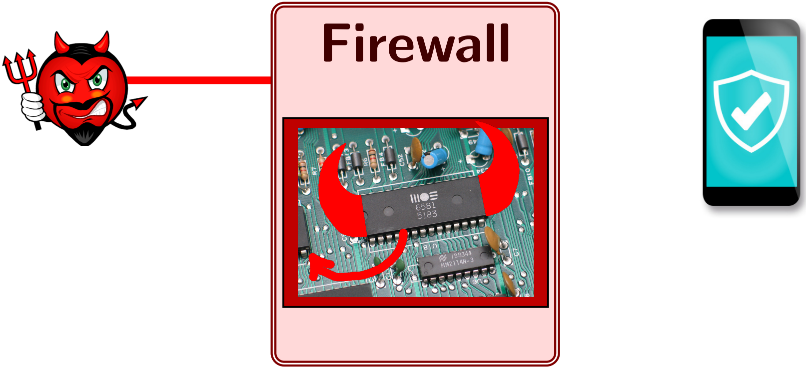
[SAGL18, Fig. 9]

How can we build trustworthy hardware?



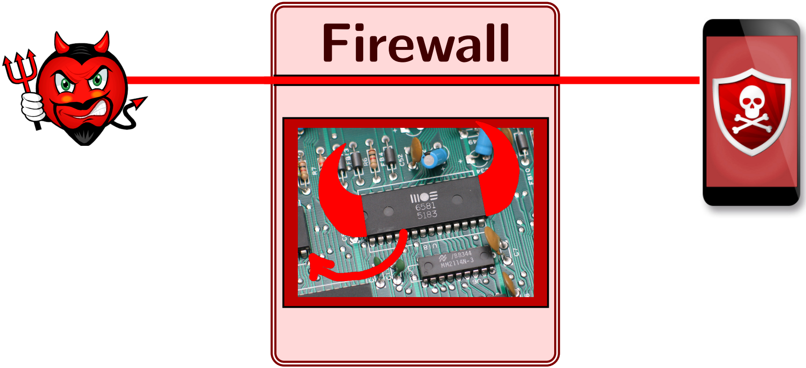
e.g., a custom chip for network packet processing whose manufacture we outsource to a third party

Untrusted manufacturers can craft hardware Trojans



What if the chip's manufacturer inserts a **back door**?

Untrusted manufacturers can craft hardware Trojans

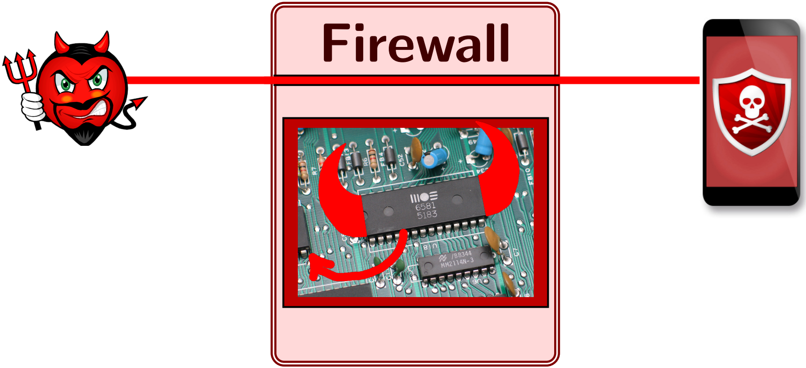


What if the chip's manufacturer inserts a **back door**?

Threat: incorrect execution of the packet filter

(Other concerns, e.g., secret state, are important but orthogonal)

Untrusted manufacturers can craft hardware Trojans



What if the chip's manufacturer inserts a **back door**?

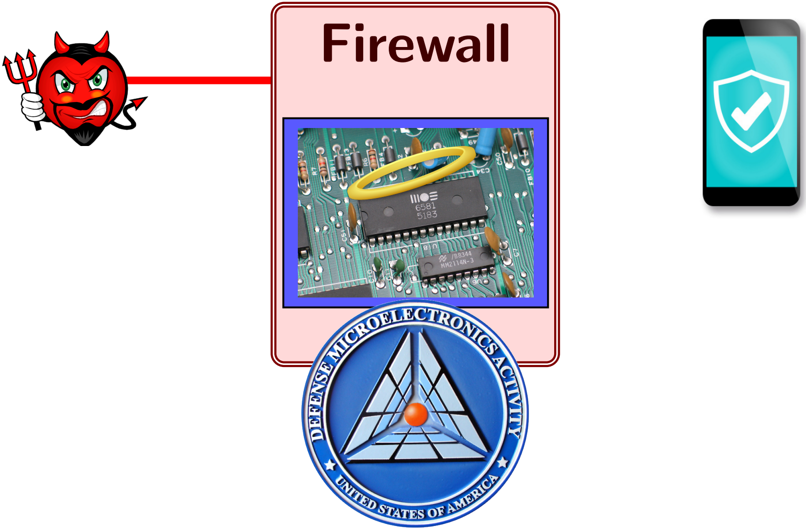
The Cybercrime Economy

Fake tech gear has infiltrated the U.S. government

by David Goldman @DavidGoldmanCNN

November 8, 2012: 3:10 PM ET

Untrusted manufacturers can craft hardware Trojans



US DoD controls supply chain with **trusted foundries**.

Trusted fabs are the only way to get strong guarantees

For example, stealthy trojans can thwart post-fab detection
[A2: Analog Malicious Hardware, Yang et al., Oakland16; Stealthy Dopant-Level Trojans, Becker et al., CHES13]

Trusted fabs are the only way to get strong guarantees

For example, stealthy trojans can thwart post-fab detection [A2: Analog Malicious Hardware, Yang et al., Oakland16; Stealthy Dopant-Level Trojans, Becker et al., CHES13]

But trusted fabrication is not a panacea:

- ✗ Only 5 countries have cutting-edge fabs on-shore
- ✗ Building a new fab takes $$$$$$, years of R&D

Trusted fabs are the only way to get strong guarantees

For example, stealthy trojans can thwart post-fab detection
[A2: Analog Malicious Hardware, Yang et al., Oakland16; Stealthy Dopant-Level Trojans, Becker et al., CHES13]

But trusted fabrication is not a panacea:

- ✗ Only 5 countries have cutting-edge fabs on-shore
- ✗ Building a new fab takes $$$$$$, years of R&D
- ✗ Semiconductor scaling: chip area and energy go with square and cube of transistor length ("critical dimension")
- ✗ So using an old fab means an enormous performance hit
  e.g., India's best on-shore fab is $10^8\times$ behind state of the art

Trusted fabs are the only way to get strong guarantees

For example, stealthy trojans can thwart post-fab detection
[A2: Analog Malicious Hardware, Yang et al., Oakland16; Stealthy Dopant-Level Trojans, Becker et al., CHES13]

But trusted fabrication is not a panacea:

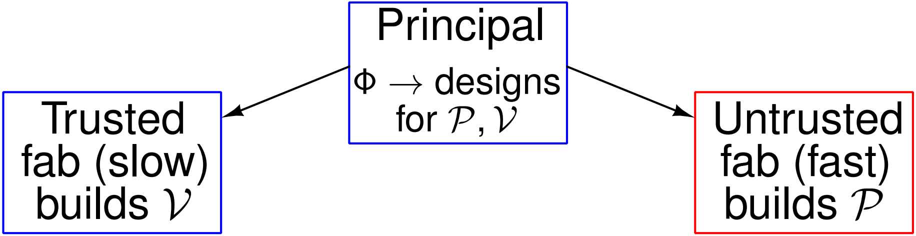
- ✗ Only 5 countries have cutting-edge fabs on-shore
- ✗ Building a new fab takes $$$$$$, years of R&D
- ✗ Semiconductor scaling: chip area and energy go with square and cube of transistor length ("critical dimension")
- ✗ So using an old fab means an enormous performance hit
  e.g., India's best on-shore fab is $10^8\times$ behind state of the art

Idea: outsource computations to untrusted chips

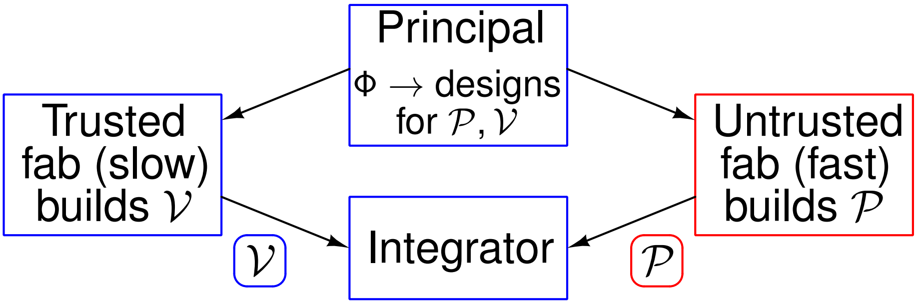
Verifiable ASICs [WHGsW16,WJBsTWW17]

$\boxed{\begin{array}{c}\text{Principal}\\ \Phi \to \text{designs}\\ \text{for } \mathcal{P}, \mathcal{V}\end{array}}$

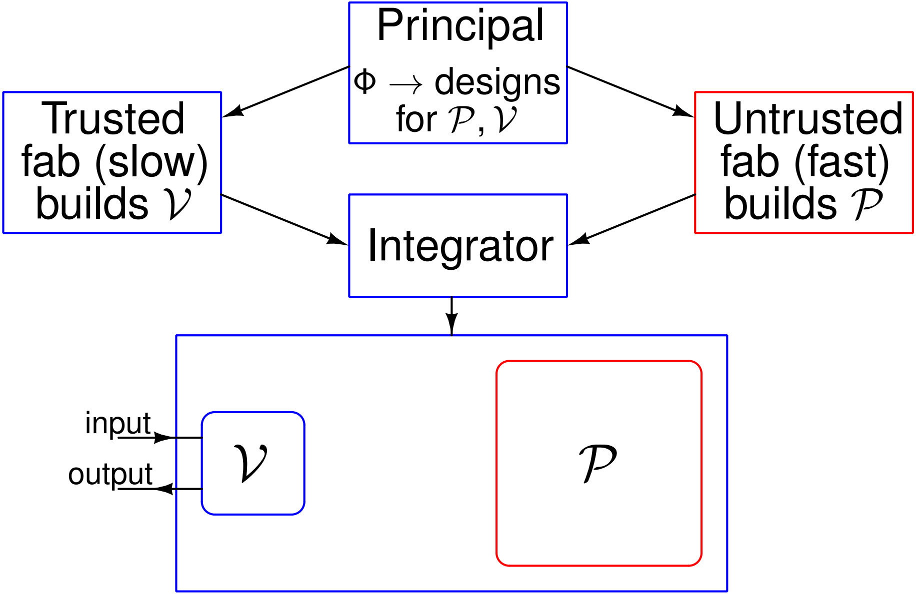
Verifiable ASICs [WHGsW16,WJBsTWW17]



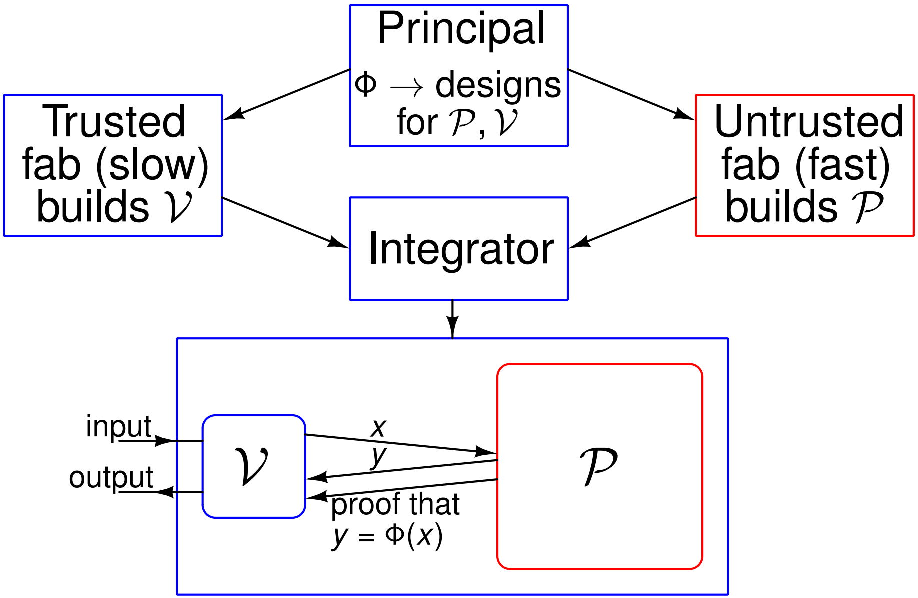
Verifiable ASICs [WHGsW16,WJBsTWW17]



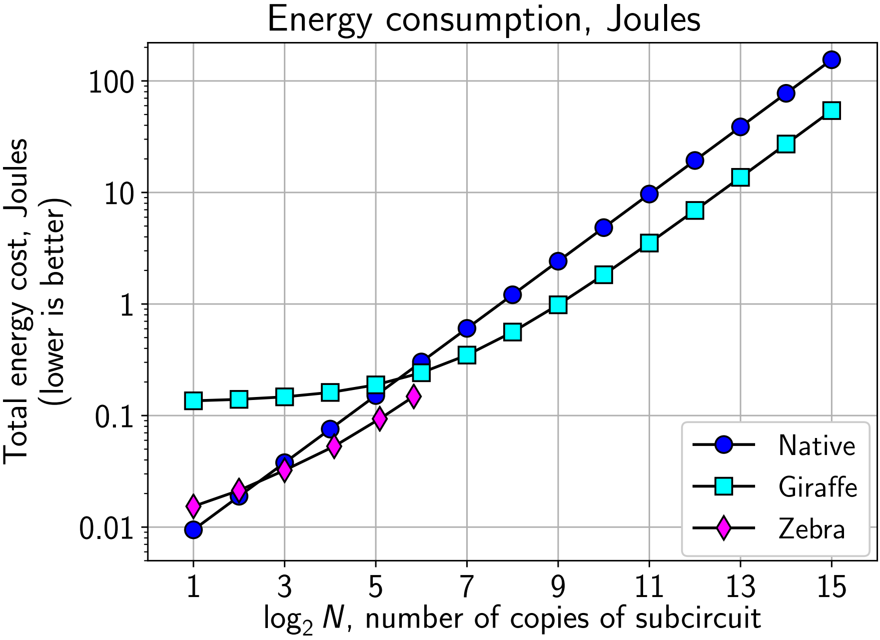
Verifiable ASICs [WHGsW16,WJBsTWW17]



Verifiable ASICs [WHGsW16,WJBsTWW17]



Verifiable ASICs [WHGsW16,WJBsTWW17]: Curve25519



Wishlist: back-ends

avoiding FFTs

☞ major bottleneck in systems based on QAPs and IOPs;
the “quasilinear barrier”
memory-, communication-intensive, costly to distribute

Wishlist: back-ends

avoiding FFTs

- ☞ major bottleneck in systems based on QAPs and IOPs;
  the "quasilinear barrier"
  memory-, communication-intensive, costly to distribute

better multilinear polynomial commitments

- ☞ major bottleneck in systems based on IPs and MIPs;
  sqrt-sized *or* expensive for $\mathcal{V}$ *or* trusted setup

Wishlist: back-ends

avoiding FFTs

☞ major bottleneck in systems based on QAPs and IOPs;
the "quasilinear barrier"
memory-, communication-intensive, costly to distribute

better multilinear polynomial commitments

☞ major bottleneck in systems based on IPs and MIPs;
sqrt-sized *or* expensive for $\mathcal{V}$ *or* trusted setup

MPC-in-the-head beyond the sqrt barrier

☞ ZKB++ and Ligero are *super fast* with minimal assumptions; can we get smaller proofs?

Wishlist: back-ends

avoiding FFTs

☞ major bottleneck in systems based on QAPs and IOPs;
the "quasilinear barrier"
memory-, communication-intensive, costly to distribute

better multilinear polynomial commitments

☞ major bottleneck in systems based on IPs and MIPs;
sqrt-sized *or* expensive for $\mathcal{V}$ *or* trusted setup

MPC-in-the-head beyond the sqrt barrier

☞ ZKB++ and Ligero are *super fast* with minimal assumptions; can we get smaller proofs?

updateable SRS with updateable proofs

☞ some steps in this direction:
[Lip19] `https://ia.cr/2019/333`

Wishlist: front-ends

beyond the AC model

- ☞ "natural" computations are ugly as ACs: bitwise ops, comparisons; this is a *major* cost, e.g., in SHA-256 TinyRAM [BCGTV13,BCTV14a], vRAM [ZGKPP18], STARK [BBHR19] point the way; can we go further?

Wishlist: front-ends

beyond the AC model

- ☞ “natural” computations are ugly as ACs: bitwise ops, comparisons; this is a *major* cost, e.g., in SHA-256
  TinyRAM [BCGTV13,BCTV14a], vRAM [ZGKPP18], STARK [BBHR19] point the way; can we go further?

compilers for everyone!

- ☞ recent work *hand tunes* statements, relies on authors' intuition and implicit knowledge
  let's *systematize* this knowledge, *automate* tuning
- ✓ improved accessibility and real-world deployability
- ✓ highly leveraged work for the research community:
  simpler, higher quality evaluations, easier-to-interpret results

Recap

- ☞ huge design space!

Recap

☞ huge design space!

✗ costs are still high

Recap

- ☞ huge design space!
- ✗ costs are still high
- ✓ nevertheless, lots of cool applications. . .

Recap

- ☞ huge design space!
- ✗ costs are still high
- ✓ nevertheless, lots of cool applications…

…and plenty of research questions to explore!

`rsw@cs.stanford.edu`