Practical proof systems: Implementations, applications, and next steps

Riad S. Wahby

Stanford

September 23rd, 2019
Verifier \( \mathcal{V} \)

\[ \text{computation } \Phi, \]  
\[ \text{input } x \]  
\[ \text{output } y \]  

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

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

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

[Bab85, GMR85, BCC86, BGHKMR90, 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, ...]
Verifier $\mathcal{V}$

\[ \text{computation } \Phi, \text{ input } x \]

output $y$

+ short proof $\pi$

Prover $\mathcal{P}$

\[ In \, general: \, \text{Proof } \pi \text{ convinces } \mathcal{V} \text{ that } y = \Phi(x). \]

\[ For \, zero \, knowledge: \, \pi \text{ convinces } \mathcal{V} \text{ that } \mathcal{P} \text{ knows witness } w \]

s.t. $y = \Phi(x, w)$, without revealing $w$. 

[Bab85, GMR85, BCC86, BGGHMR90, 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

Bottom line: this is a huge tradeoff space!
Costs and desiderata

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

Bottom line: this is a huge tradeoff space!
Costs and desiderata

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

- $P$ time
- $V$ time
- communication cost / proof size
- cryptographic assumptions
- trusted setup? per-$\Phi$ or universal?
- interactive or non-interactive?
Costs and desiderata

- $P$ time
- $V$ time
- communication cost / proof size
- cryptographic assumptions
- trusted setup? per-$\Phi$ or universal?
- interactive or non-interactive?
- zero knowledge?
Costs and desiderata

- $P$ time
- $V$ time
- communication cost / proof size
- cryptographic assumptions
- trusted setup? per-$\Phi$ 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-$\Phi$ 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 $x$, $P$ convinces $V$ that $y = \Phi(x, w)$ for a witness $w$ that $P$ knows.
Proof systems pipeline

On input $x$, $P$ convinces $V$ that $y = \Phi(x, w)$ for a witness $w$ that $P$ knows.
Proof systems pipeline

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

\begin{align*}
\Phi &: \text{witness checking computation} \\
\text{intermed. repr. } C \\
\text{proof machinery} \\
\mathcal{V} &: \text{computation} \\
\mathcal{P} &: \text{computation}
\end{align*}

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

\text{back-end}
Proof systems pipeline

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

- **Front-end**
  - $C$ is satisfied $\iff y = \Phi(x, w)$

- **Back-end**
  - Valid proof $\iff C$ 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 [WJBsTW17], Hyrax [WTsTW18], Spartan [Set19],
(ZK)\(v\)SQL [ZGKPP17\(\{a,b\}\)], \(v\)RAM [ZGKPP18], Libra [XZZPS19]

**MPC-in-the-head** [IKOS08]
ZKBoo [GMO16], ZKB++ [CDGORRSS17], Ligero [AHIV17]
Underlying machinery

Linear PCPs [IKO07, BCGI19] 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 [WJBsTW17], 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 [BBBPPWM18]
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 [WJBsTW17], 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 $\Sigma$-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 [WJBsTW17], 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 $\Sigma$-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 \cite{IKO07,BBCGI19} and QAPs \cite{GGPR13}
Pepper \cite{SBW11,SMBW12}, Ginger \cite{SVPBBW12}, Zaatar \cite{SBVBPW13},
Pinocchio \cite{PGHR13}, \cite{BCGTV13}, libSNARK \cite{BCTV14a}, \cite{BCTV14b},
ADSNARK \cite{BBFR15}, Geppetto \cite{CFHKKNPZ15}, \cite{Gro16}, ...

IPs \cite{GMR85,GKR08,Tha13}, MIPs \cite{BGKW88,BTVW14} \cite{CMT12}, Giraffe \cite{WJBsTWW17}, Hyrax \cite{WTsTW18}, Spartan \cite{Set19},
(ZK)\textit{v}SQL \cite{ZGKPP17\{a,b\}}, \textit{v}RAM \cite{ZGKPP18}, Libra \cite{XZZPS19}

MPC-in-the-head \cite{IKOS08}
ZKB\textit{oo} \cite{GMO16}, ZKB++ \cite{CDGORRSZ17}, Ligero \cite{AHIV17}

Generalized $\Sigma$-protocols \cite{Sch89,CP92,Oka92,Cra97} \cite{BCCGP16}, Bulletproofs \cite{BBBPWM18}

IOPs \cite{KR08,BCS16,RRR16} + Arguments \cite{Kil92,Mic94}
Aurora \cite{BCRSVW19}, STARK \cite{BBHR19}

Polynomial delegation \cite{KZG10,PST13}
Sonic \cite{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]
[CM12], 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 \(\Sigma\)-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]
    ZKBBoo [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 \([\text{BBBPWM18}]\), Hyrax \([\text{WTsTW18}]\), Spartan \([\text{Set19}]\)

ZKBoo \([\text{GMO16}]\), ZKB++ \([\text{CDGORRSZ17}]\), Ligero \([\text{AHIV17}]\),
Aurora \([\text{BCRSVVW19}]\), STARK \([\text{BBHR19}]\)

\([\text{CMT12,Tha13}]\), Clover \([\text{BTVW14}]\), Giraffe \([\text{WJBsTWW17}]\) (CRHF/ROM)

Universal trusted setup

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

Sonic \([\text{MBKM19}]\)

Per-\(\Phi\) trusted setup

Pepper, Ginger, Zaatar \([\text{SBW11, SMBW12, SVPBBW12, SBVBWPW13}]\)

Pinocchio \([\text{PGHR13}]\), libSNARK \([\text{BCTV14a}]\),
ADSNARK \([\text{BBFR15}]\), Geppetto \([\text{CFHKKNPZ15}]\), \ldots
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]
- [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, SBVBWPW13] (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]
- [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]
ZKBoo [GMO16], ZKB++ [CDGORRSZ17], Ligero [AHIV17], Aurora [BCRSVW19], STARK [BBHR19] (CRHF/ROM)

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]

[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)
proof size, kiB (lower is better)

$log_2 M$, number of leaves in Merkle tree

<table>
<thead>
<tr>
<th></th>
<th>10^{-1}</th>
<th>10^0</th>
<th>10^1</th>
<th>10^2</th>
<th>10^3</th>
<th>10^4</th>
<th>10^5</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZKB++</td>
<td>100</td>
<td>2 × 100</td>
<td>3 × 100</td>
<td>4 × 100</td>
<td>6 × 100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Libra</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aurora</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bulletproofs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ligero</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>libSTARK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>libSNARK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Numbers are from Hyrax [WTsTW18], except Libra, Aurora, and libSNARK, which are from Libra [XZZPS19].
Numbers are from Hyrax [WTS18], 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 $x$, $P$ convinces $V$ that $y = \Phi(x, w)$ for a witness $w$ that $P$ knows

\[ \Phi: \text{witness checking computation} \rightarrow \text{intermed. repr. } C \]

\[ \text{front-end} \quad C \text{ is satisfied } \iff y = \Phi(x, w) \]

\[ \text{proof machinery} \rightarrow \begin{array}{c} \text{proof} \text{ computation} \\ \text{valid proof} \iff C \text{ is satisfied} \end{array} \]

\[ \text{back-end} \]
Representing $\Phi$ for execution on the back-end

"CPU": run $\Phi$ on unrolled FSM

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

[BCGTV13, BCTV14a, BCTV14b, CTV15, ZGKPP18, BBHR19]
Representing $\Phi$ for execution on the back-end

**“CPU”: run $\Phi$ on unrolled FSM**

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

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

- ... 

- **fetch-decode**
- **execute step $T$**
  - CPU state: pc, regs, ...

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

**“FPGA”: translate $\Phi$ directly to AC or constraints**

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

\[ \Rightarrow \]

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 $\Phi$ for execution on the back-end

**“CPU”**: run $\Phi$ on unrolled FSM

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

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

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

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

**“FPGA”**: translate $\Phi$ directly to AC or constraints

```
if (i >= 5)
  i1 = i0 + 1;
else
  i = i * 2;
```

```
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 $\Phi$ for execution on the back-end

“CPU”: run $\Phi$ on unrolled FSM

```c
// assume $i_0$ is $k+1$ bits
$i_4 = i_0 - 5$;
// prover supplies $i_4_0$, ..., $i_4_k$
assert ($i_4 - i_4_0 - 2 \times i_4_1 - \ldots - 2^k \times i_4_k \equiv 0$);
assert ($i_4_0 \times (1 - i_4_0) \equiv 0$);
...
assert ($i_4_{\{k-1\}} \times (1 - i_4_{\{k-1\}}) \equiv 0$);
assert ($i_4_k \equiv 0$);
```

```c
if ($i \geq 5$)
    $i = i + 1$;
else
    $i = i \times 2$;
```

```c
i1 = i0 + 1;
i2 = i0 \times 2;
i3 = (i0 \geq 5) ?
    i1 : i2;
```

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

\[\textbf{[GKR08]-derived systems need a \textit{low depth} circuit:}
[\text{CMT12, WHGsW16, WJBsTWW17, WT}\text{sTW18, XZZPS19}]\]
Performance vs. expressiveness costs

lower

special purpose

vSQL [ZGKPP17]
Bellman gadgetlib [BCTV14a]
LegoSNARK [CFQ19]
c0c0 [KZMQCPPsS15]

vRAM [ZGKPP18]
Buffet [WSRHBW15]
STARK [BBHR19]
(vn)TinyRAM [BCTV14a] [BCGTV13]

pure

Giraffe [WJBsTWW17]
Allspice [VSBW13]

Geppetto 

xJsnark [KPS18]

Pinocchio [PGHR13]
Circom
Ginger [SVPBBW12]
Pepper [SMBW12]

Pantry [BFRSBW13]

stateful

ADSNARK [BBFR15]

(vn)TinyRAM

higher

control flow

[Tha13]
[ZGKPP17]

[BBHR19]
[CTV15]

better
Front-end comparison

Extrapolated \( P \) execution time, normalized to Buffet

- 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
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:

<table>
<thead>
<tr>
<th></th>
<th>Pantry</th>
<th>BCTV14a</th>
<th>Buffet</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m \times m$</td>
<td>215</td>
<td>7</td>
<td>215</td>
</tr>
<tr>
<td>merge sort $k$ elements</td>
<td>8</td>
<td>32</td>
<td>512</td>
</tr>
<tr>
<td>Knuth-Morris-Pratt search</td>
<td>$n = 4, \ell = 8$</td>
<td>$n = 16, \ell = 160$</td>
<td>$n = 256, \ell = 2900$</td>
</tr>
</tbody>
</table>

$vRAM$ [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: $100 \times$ larger instances

[WZCPS18, Fig. 4]
DIZK: $100 \times$ faster execution

[WZCPS18, Fig. 5]
Cryptocurrencies!

ZCash (following [BCGGMTV14]) uses ZK for anonymity: no one knows who you are.
Privacy: transaction values are hidden.

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

https://github.com/barryWhiteHat/roll_up
Let’s build a bank out of a smart contract!

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

[GitHub Link]
Let’s build a bank out of a smart contract!

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

[GitHub link: https://github.com/barryWhiteHat/roll_up]
Spice [SAGL18]: verifiable *concurrent* services (in ZK)

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

[SAGL18, Fig. 1]

**Issue:** need verifiable storage with concurrency
Spice [SA18]: verifiable concurrent services (in ZK)

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

**Idea:** adapt primitives from memory checking literature [BEGKN91,CDDG03,AEKKMPR17]

(source: Srinath's talk)
Spice [SAGL18]: verifiable concurrent services (in ZK)

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

Performance results:

<table>
<thead>
<tr>
<th></th>
<th>get</th>
<th>put</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pantry</td>
<td>0.078</td>
<td>0.039</td>
</tr>
<tr>
<td>Pantry+Jubjub</td>
<td>0.153</td>
<td>0.076</td>
</tr>
<tr>
<td>Geppetto</td>
<td>0.002</td>
<td>0.002</td>
</tr>
<tr>
<td>Spice (1-thread)</td>
<td>3.6</td>
<td>3.6</td>
</tr>
<tr>
<td>Spice (512-threads)</td>
<td>1366</td>
<td>1370</td>
</tr>
</tbody>
</table>

[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

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

What if the chip’s manufacturer inserts a back door?

US DoD controls supply chain with trusted foundries.
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.

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)

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 $\ldots$, 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]

Principal

\[
\Phi \rightarrow \text{designs for } \mathcal{P}, \mathcal{V}
\]
Verifiable ASICs [WHGsW16,WJBsTWW17]

Principal

\( \Phi \rightarrow \text{designs for } \mathcal{P}, \mathcal{V} \)

Trusted fab (slow) builds \( \mathcal{V} \)

Untrusted fab (fast) builds \( \mathcal{P} \)
Verifiable ASICs [WHGsW16, WJBsTWW17]

- Principal
  \( \Phi \rightarrow \text{designs for } P, V \)

- Trusted fab (slow) builds \( V \)

- Integrator

- Untrusted fab (fast) builds \( P \)
Verifiable ASICs [WHGsW16,WJBsTWW17]

Principal

\[ \Phi \rightarrow \text{designs for } \mathcal{P}, \mathcal{V} \]

Untrusted fab (fast) builds \( \mathcal{V} \)

Trusted fab (slow) builds \( \mathcal{V} \)

Integrator

Untrusted fab (fast) builds \( \mathcal{P} \)

Input \( \mathcal{V} \)

Output \( \mathcal{P} \)
Verifiable ASICs [WHGsW16, WJBsTWW17]

---

**Principal**

\[ \Phi \rightarrow \text{designs for } \mathcal{P}, \mathcal{V} \]

---

**Integrator**

---

**Trusted fab (slow) builds \( \mathcal{V} \)**

---

**Untrusted fab (fast) builds \( \mathcal{P} \)**

---

**\( \mathcal{V} \)**

\[ x \rightarrow y \]

\[ \text{proof that } y = \Phi(x) \]

---

**\( \mathcal{P} \)**

---

**Verifiable ASICs [WHGsW16, WJBsTWW17]**
Verifiable ASICs [WHGsW16, WJBsTWW17]: Curve25519

Energy consumption, Joules

Total energy cost, Joules
(lower is better)

Native
Giraffe
Zebra

$\log_2 N$, number of copies of subcircuit
0.01
0.1
1
10
100
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- and 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

updateable SRS with updateable proofs

some steps in this direction: [Lip19](https://ia.cr/2019/333)
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 \( \forall \) or trusted setup

MPC-in-the-head beyond the sqrt barrier

- ZKB++ and Ligero are \textit{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:
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 [BCGT\textsuperscript{13},BCTV\textsuperscript{14a}], vRAM [ZGKPP\textsuperscript{18}], STARK [BBHR\textsuperscript{19}] 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