Security Bounds for Proof-Carrying Data from Straightline Extractors

Alessandro Chiesa, Ziyi Guan, Shahar Samocha, Eylon Yogev
What is proof-carrying data (PCD)?
- Recursive compositions of SNARKs.
- It's useful for efficiently verifying distributed computations.

**Problem:**
- PCD is deployed under the assumption "security of PCD" = "security of underlying SNARK".
- BUT existing security analyses show a huge gap in security ("PCD is far less secure than underlying SNARK").

**This work:**
- We propose an idealized PCD that models hash-based PCD in practice.
- We prove that this idealized PCD is as secure as its underlying SNARK.
What is proof-carrying data (PCD)? [1/2]

Proof-carrying data (PCD)
- Enables mutually distrustful parties to perform a distributed computation
- The correctness of each step can be verified efficiently

E.g. A simple distributed computation: summing six numbers

\[
x_1 + x_2 + x_3 + x_4 + x_5 + x_6
\]
What is proof-carrying data (PCD)? [2/2]

Proof-carrying data (PCD)
- Enables mutually distrustful parties to perform a distributed computation
- The correctness of each step can be verified efficiently

PCD transcript $T$ for a distributed computation with size $N = 8$ and depth $D = 3$

Correctness of transcript $T$ is determined by compliance predicate $\phi$
- Node $(2,3)$ is correct if $\phi(z_{2,3}, w_{2,3}, (z_{3,3}, z_{3,4})) = 1$.
- $T$ is $\phi$-compliant if all nodes are correct.

The proof string $\Pi_{2,3}$ attests that:
- node $(2,3)$ is correct, AND
- each child vertex of node $(2,3)$ has a valid proof string.

PCD prover $\mathbb{P}$ and PCD verifier $\mathbb{V}$

$(z_{2,3}, w_{2,3}) \xrightarrow{\mathbb{P}} z_{2,3} \xrightarrow{\Pi_{2,3}} b \in \{0,1\}$

$(z_{3,3}, \Pi_{3,3}), (z_{3,4}, \Pi_{3,4}) \xrightarrow{\mathbb{V}} \Pi_{2,3}$
Security guarantee of PCD

Perfect completeness: \( \mathbb{P} \) can convince \( \mathbb{V} \) of correct computations.

Knowledge soundness: \( \forall \) bounded \( \tilde{\mathbb{P}} \), \( \exists \) an efficient extractor \( \mathbb{E}_{\tilde{\mathbb{P}}} \) such that

\[
\Pr \left[ \begin{array}{c}
\vee (z_{\text{out}}, \Pi_{\text{out}}) = 1 \\
\wedge T \text{ is not } \phi\text{-compliant}
\end{array} \right] \leq \kappa(\lambda, D, N).
\]

\( \lambda \): security parameter  
\( T \): computation transcript  
\( D \): maximum transcript depth  
\( N \): maximum transcript size  

Output: \( z_{\text{out}} \)

\( b \in \{0, 1\} \)
Review: SNARK

PCD can be constructed from a SNARK (e.g., for CSAT).

CSAT := \{ ((C, x), w) : C(x, w) = 1 \}

ARG = (P_{ARG}, V_{ARG})

- Perfect completeness: $P_{ARG}$ convinces $V_{ARG}$ if $C(x, w) = 1$.
- Knowledge soundness: $\forall$ bounded $\tilde{P}_{ARG}$, $\exists$ an efficient extractor $E_{\tilde{P}_{ARG}}$ such that

$$
\Pr \left[ \begin{array}{c}
((C, x), w) \not\in \text{CSAT} \\
\land V_{ARG}(C, x, \pi) = 1
\end{array} \right] \leq \kappa_{ARG}(\lambda).
$$

$((C, x), w) \not\in \text{CSAT}$
Naive approach: concatenate SNARK proofs

SNARK prover for compliance predicate $\phi$

$(z_{2,3}, w_{2,3}, (z_{3,3}, z_{3,4}))$ → $P_{ARG}$ → $\pi_{2,3}$

$(z_{2,3}, w_{2,3})$ → $\Pi_{2,3} := \pi_{2,3} \parallel \Pi_{3,3} \parallel \Pi_{3,4}$

Issue: $\Pi_{2,3}$ is NOT succinct (linear in number of vertices)
Working idea: Recursively compose the SNARK proofs

PCD formalizes the recursive proof composition of a SNARK:
- PCD prover and verifier invoke SNARK prover and verifier (for CSAT) for the recursive circuit C.

Recursive circuit

\[
\left( (\mathcal{C}, \mathcal{z}), (w, (\mathcal{z}_i, \Pi_i)_i) \right) \rightarrow \left( \mathcal{z}, w, (\mathcal{z}_i)_i \right) \rightarrow b_\phi \in \{0,1\} \\
\left( \mathcal{C}, \mathcal{z}_i, \Pi_i \right) \rightarrow b_{V_{ARG}} \in \{0,1\}
\]
Canonical security analysis of PCD

Size of extractor
- $|\tilde{P}_i| = |E_{i-1}| + O(m^i) \implies |E_{\tilde{P}}| = t_E(|\tilde{P}_i|)$
- $|E_i| \leq |E_{\tilde{P}}| + O(m^i)$
- $t_E : n \mapsto n^c \implies |E_{\tilde{P}}| = O\left( |\tilde{P}|^{c^0} \right)$

$\implies |E_{\tilde{P}}|$ is polynomial only when D is constant.

Non-black-box knowledge soundness is problematic: size of extractor grows too quickly.

Finding a better analysis remains a MAJOR open problem in this area.

Today: focus on PCD based on SNARKs with "strong" extraction.
Our result

**Theorem.** We prove a significantly improved security bound for PCD based on SNARKs with **straightline extraction**:

\[
\kappa(\lambda, q, D, N) \leq \kappa_{\text{ARG}}(\lambda, q, N)
\]

Prior works

\[
\kappa(\lambda, q, D, N) \leq \exp(D) \cdot \kappa_{\text{ARG}}(\lambda, q, N)
\]

No security when \(D\) is larger than constant.

In practice, SNARKs have non-black-box knowledge soundness. Straightline extraction only exists in idealized models. How can we apply our theorem in practice then?
Applications

Application 1 [main].
- We propose a new idealization of hash-based PCD used in practice as a “PCD” in the ROM.
- We apply our theorem: $\kappa(\lambda, q, D, N) \leq \kappa_{\text{ARG}}(\lambda, q, N) = \kappa_{\text{ARG}}(\lambda, q)$.
- First justification for current choice of parameters of hash-based PCD in practice! [Polygon, Sharp]

Application 2.
- [CT10]: SNARK with straightline extraction in the SROM (signed random oracle model).
- Their bound: $\kappa(\lambda, q, D, N) \leq N \cdot \kappa_{\text{ARG}}(\lambda, q, N)$.
- Our bound: $\kappa(\lambda, q, D, N) \leq \kappa_{\text{ARG}}(\lambda, q, N)$.

Application 3.
- [CCGOS23]: SNARK with straightline extraction in the AROM (arithmetized random oracle model).
- Their bound: $\kappa(\lambda, q, D, N) \leq N \cdot \kappa_{\text{ARG}}(\lambda, q, N)$.
- Our bound: $\kappa(\lambda, q, D, N) \leq \kappa_{\text{ARG}}(\lambda, q, N)$. 
Recursive proof composition with straightline extraction
**SNARKs with straightline extraction**

**SNARKs in an oracle model (e.g. ROM):**

\[
\begin{align*}
((C, x), w) & \xrightarrow{f} P \\
(C, x) & \xrightarrow{f} V \\
\pi & \xrightarrow{\text{very small}} b \in \{0,1\}
\end{align*}
\]

**Straightline knowledge soundness:** \(\exists\) a deterministic extractor \(E\) such that \(\forall\) bounded adversary \(\tilde{P}\),

\[
\Pr\left[ ((C, x), w) \notin \text{CSAT} \land V^f(C, x, \pi) = 1 \right] \leq \kappa_{\text{ARG}}(\lambda, q).
\]

\(\lambda\): security parameter

\(q\): adversary query bound

**Wonderful Fact:** in the ROM (and other interesting oracle models) there are SNARKs of interest with straightline extraction!

(E.g., the Micali SNARK and BCS SNARK and related constructions.)
Can’t we use the previous recursive composition?

Recursive circuit

\[ \left( (C, z), (w, (z_i, \Pi_i)_i) \right) \]

\[ (z, w, (z_i)_i) \]

\[ (C, z_i, \Pi_i) \]

\[ C^f \]

\[ \phi^f \]

\[ b_{\phi} \in \{0,1\} \]

\[ b := b_{\phi} \land b_{V_{ARG}} \]

\[ V_{ARG}^f \]

\[ b_{V_{ARG}} \in \{0,1\} \]

\[ P^f \]

\[ (C, z, w) \]

\[ (z_i, \Pi_i)_i \]

\[ (C, z_i, \Pi_i) \]

\[ \Pi \]

\[ C^f \]

\[ V_{ARG}^f \]

\[ b \]

\[ P_{ARG}^f \]

\[ V_{ARG}^f \]

\[ b \]

\[ \text{ISSUE! C has oracle access to } f. \]

\[ P_{ARG} \text{ and } V_{ARG} \text{ need to prove computations involving oracle } f. \]
Relativized SNARKs in an oracle model

We need SNARK in the oracle model that can prove/verify for oracle relations
- Relativized SNARK!

$$\text{CSAT}^f := \{(C, x, w) : C^f(x, w) = 1\}$$

Relativized SNARK for $\text{CSAT}^f$ →Recursive proof composition →PCD with straightline extraction

$\text{PCD straightline knowledge soundness: } \exists$ a deterministic extractor $E$ such that $\forall$ bounded adversary $\tilde{P}$,

$$\Pr \left[ \begin{array}{c} \forall(z_{out}, \Pi) = 1 \\ \land T \text{ is not } \phi\text{-compatible} \end{array} \right] \left[ \begin{array}{c} f \leftarrow U(\lambda) \\ (\phi, z_{out}, \Pi_{out}) \leftarrow \tilde{P}^f \\ T \leftarrow E(\phi, z_{out}, \Pi_{out}, \text{tr}) \end{array} \right] \leq \kappa(\lambda, q, N).$$

$\lambda$: security parameter
$N$: maximum transcript size
$q$: adversary query bound

Output: $z_{out}$

$E \leftarrow (\phi, z_{out}, \Pi_{out}, \text{tr})$
$\tilde{P} \leftarrow \phi$
$\Pi_{out} \rightarrow \tilde{P}$
$V \rightarrow b = 1$
Concrete security of PCD with straightline extraction
Construction of the PCD extractor

In general, PCD extractor is constructed by repeatedly invoking SNARK extractor.

\[(\phi, z, \Pi, \text{tr}) \rightarrow E \rightarrow T\]

Extraction queue \(Q\)

Extraction queue \(Q\)

Parse \(w_{v_1}\) as \((w_1, (z_{2,i}, \Pi_{(v_2,i,v_1)})_{i \in [3]}))\)

In general, PCD extractor is constructed by repeatedly invoking SNARK extractor.
Security analysis in previous works

A natural analysis gives us this bound: $\kappa(\lambda, q, N) \leq N \cdot \kappa_{\text{ARG}}(\lambda, q, N)$

- Each recursion pays the knowledge soundness error of the argument.
- The $i$-th extraction: invoking $E_{\text{ARG}}$ for a corresponding argument prover $\tilde{P}_i$.

Warning: the actual construction of $\tilde{P}_i$ is more complicated. This is for intuitive explanation only.
Our security analysis [1/2]

There is one vertex in $T$ that is not $\phi$-compliant

$\implies$ Find such vertex in one pass and output it

$\implies$ $\kappa(\lambda, q, N) \leq \kappa_{\text{ARG}}(\lambda, q, N)$. 
Our security analysis [2/2]

Our theorem: \( \kappa(\lambda, q, D, N) \leq \kappa_{\text{ARG}}(\lambda, q, N) \)
Application:
Set security for hash-based PCD
Warm-up: analyzing hash-based SNARKs

Three-step recipe:

Step 1. Model the hash function as "ideal": a random function.
- the hash-based SNARK is idealized as a **SNARK in the random oracle model (ROM-SNARK)**.

Step 2. Establish **concrete** security bounds for the ROM-SNARK.

Step 3. Set security parameters of the hash-based SNARK accordingly.

Careful!! Idealization is applicable only for black-box use of the hash function. Fortunately, applicable for the hash-based SNARKs we care about (e.g. Micali SNARK).
First attempt for idealization of hash-based PCD

PCDs are deployed based on various approaches. A popular approach is hash-based PCD.

Nevertheless, practitioners use hash-based PCD as if it's as secure as the hash-based SNARK. (!!)

<table>
<thead>
<tr>
<th>Standard Model</th>
<th>Random Oracle Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hash-based PCD</td>
<td>ROM-SNARK for CSAT</td>
</tr>
<tr>
<td>Expensive security analysis (extractor blows up)</td>
<td>Recursive proof composition</td>
</tr>
</tbody>
</table>

Can our new analysis justify the above practice?

<table>
<thead>
<tr>
<th>PCD with straightline extraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recursively proof composition</td>
</tr>
</tbody>
</table>

Our theorem: $\kappa(\lambda, q, D, N) \leq \kappa_{\text{ARG}}(\lambda, q, N)$

Nevertheless, practitioners use hash-based PCD as if it's as secure as the hash-based SNARK. (!!)
Second attempt for idealization of hash-based PCD

What we hope to do

Idealization is applicable only for black-box use of the hash function - not true in general.

Our theorem: $\kappa(\lambda, q, D, N) \leq \kappa_{\text{ARG}}(\lambda, q, N)$

Hash-based PCD $\xrightarrow{\text{Idealize}}$ PCD in the ROM $\xrightarrow{\text{Recursive proof composition}}$ Relativized SNARK in the ROM with straightline extraction

Reality

Not believed to exist! [Val08, HN23]

Our theorem: $\kappa(\lambda, q, D, N) \leq \kappa_{\text{ARG}}(\lambda, q, N)$

Can’t apply
Our idealization for hash-based PCD

Issue: Hash-based PCD uses hash function in a non-black-box way.
Observation 1: PCD looks at hash function to check the correctness, it doesn’t “destroy” the hash function.
Observation 2: C is an oracle circuit because $V_{ARG}$ make oracle queries.
Solution: Forward all the queries of C by asking $P_{ARG}$ to attach C’s “query-answer trace” in the proof.

Forwarding the queries makes the proof non-succinct

Hash-based PCD  \[\xrightarrow{\text{Idealize}}\]  NON-SUCCINCT PCD in the ROM with straightline extraction

Recursive proof composition  \[\xrightarrow{\text{Forward queries}}\]  NON-SUCCINCT relativized NARK in the ROM with straightline extraction

Our theorem: $\kappa(\lambda, q, D, N) \leq \kappa_{ARG}(\lambda, q, N) = \kappa_{ARG}(\lambda, q)$
Last step: relativized ROM-NARK

Idea: Given an oracle circuit, remove its oracle gate by attaching its “query-answer trace” to instance.
What is proof-carrying data (PCD)?
- Recursive compositions of SNARKs.
- It’s useful for efficiently verifying distributed computations.

Problem:
- PCD is deployed under the assumption "security of PCD" = "security of underlying SNARK".
- BUT existing security analyses show a huge gap in security ("PCD is far less secure than underlying SNARK").

This work:
- We propose an idealized PCD that models hash-based PCD in practice.
- We prove that this idealized PCD is as secure as its underlying SNARK.
Thank you!

https://eprint.iacr.org/2023/1646
Technical extension: Probabilistic straightline extraction
Probabilistic straightline extraction

**Probabilistic straightline knowledge soundness for SNARKs:**

\[ \exists \text{ a probabilistic extractor } E \text{ such that } \forall \text{ bounded adversary } \tilde{P}, \]
\[ \Pr \left[ \left( (C, x), w \right) \notin \text{CSAT}^f \wedge V^f(C, x, \pi) = 1 \right] \leq \kappa_{\text{ARG}}(\lambda, q). \]

\[ f \leftarrow U(\lambda) \]
\[ (C, x, \pi) \leftarrow \tilde{P}^f \]
\[ w \leftarrow E(C, x, \pi, \text{tr}) \]

**Relativized SNARK for CSAT^f with probabilistic straightline extraction**

Recursive proof composition

**PCD with probabilistic straightline extraction**

**PCD probabilistic straightline knowledge soundness:**

\[ \Pr \left[ \forall(z_{\text{out}}, \Pi) = 1 \wedge T \text{ is not } \phi\text{-compatible} \right] \leq \kappa(\lambda, q, N). \]

\[ f \leftarrow U(\lambda) \]
\[ (\phi, z_{\text{out}}, \Pi_{\text{out}}) \leftarrow \tilde{\phi}^f \]
\[ T \leftarrow E(\phi, z_{\text{out}}, \Pi_{\text{out}}, \text{tr}) \]

\[ \lambda: \text{ security parameter} \]
\[ q: \text{ adversary query bound} \]

\[ \lambda: \text{ security parameter} \]
\[ N: \text{ maximum transcript size} \]
\[ q: \text{ adversary query bound} \]
Our security analysis

**Theorem.** We prove an improved security bound even for PCD based on SNARKs with **probabilistic straightline extraction:**

\[
\kappa(\lambda, q, D, N) \leq N \cdot \kappa_{\text{ARG}}(\lambda, q, N)
\]

The multiplicative factor $N$ is tight:
- With probabilistic straightline extraction, at each node, $E$ pays for both the extraction error and the randomness error of $E_{\text{ARG}}$.
- If let $\epsilon$ be the randomness error of $E_{\text{ARG}}$, it's possible to show:
  \[
  \kappa(\lambda, q, D, N) \leq \kappa_{\text{ARG}}(\lambda, q, N) + N \cdot \epsilon.
  \]
Application:
Improved concrete security for black-box PCD constructions
PCD in the SROM

- Signed random oracle model (SROM):
  - On input $x$, samples a random answer $y$, generates a signature $\sigma$ on $(x, y)$, and outputs $(y, \sigma)$.
  - Repeated inputs have the same answer.

- [CT10]: SNARK in the ROM $\rightarrow$ SNARK in the SROM (preserves straightline extraction)
  - The argument verifier doesn’t need to query the oracle: verify $\sigma$ is enough.
  - [CT10] gives a bound $\kappa(\lambda, q, N) \leq N \cdot \kappa_{\text{ARG}}(\lambda, q, N)$.
  - Our analysis improves it to $\kappa(\lambda, q, N) \leq \kappa_{\text{ARG}}(\lambda, q, N)$. 
PCD in the AROM

• Arithmetized random oracle model (AROM):
  - A random oracle: idealization of a concrete hash function $h$;
  - An arithmetization oracle: idealization of a low degree polynomial that encodes the circuit of $h$.

• [CCGOS22]: SNARK in the ROM $\rightarrow$ SNARK in the AROM (preserves straightline extraction)
  - Queries in the AROM can be accumulated.
  - [CCGOS22] gives a bound $\kappa(\lambda, q, N) \leq N \cdot \kappa_{\text{ARG}}(\lambda, q, N)$.
  - Our analysis improves it to $\kappa(\lambda, q, N) \leq \kappa_{\text{ARG}}(\lambda, q, N)$.
Example:
Real-world compliance predicate with unbounded transcript size
A real-world compliance predicate

- \( h : \{0,1\}^* \rightarrow \{0,1\}^\lambda \), a collision resistant hash function.
- \( M \): a universal Turing machine. On input a program \( P \) and an input \( x \), \( M(P, x) \) outputs \( P(x) \).
- \( T \in \mathbb{N} \) a maximum time bound.

No restriction on the size of the transcript!

- \( \mathbb{N} \) can be arbitrarily large \( \implies \) prior works can not guarantee security.
- Our result shows that security of the underlying SNARK is inherited by the PCD without loss.

\[
(z, w, \perp) \xrightarrow{} \phi_{h,M,T} \xrightarrow{} b \in \{0,1\}
\]

Base case.

Parse \( z \) as \( (y, t) \)
Parse \( w \) as \( (P, x) \)
\[
b := \begin{cases} 
  t \leq T \land M(P, x) = y \\
  \land M(P, x) \text{ runs in } t \text{ steps} 
\end{cases} 
\]

Recursive case.

Parse \( z \) as \( (y, t) \)
Parse \( z_i \) as \( (y_i, t_i) \) for each \( i \)
\[
b := \begin{cases} 
  t = 0 \land w = \perp \\
  \land \forall i, t_i \leq T \land h((y_i)_i) = y 
\end{cases} 
\]
Recursive STARKs

- Computation in Ethereum smart contract is expensive:
  - Each computation step is re-executed by every node.
- Layer 2 proof-based rollups: move computation off-chain.
  - User sends computation requests to an aggregator.
  - Aggregator produces a SNARK proof about batch of computations.
  - Ethereum smart contract verifiers the SNARK proof and update states.
- Aggregator: PCD prover.
- Ethereum smart contract: PCD verifier.