(Nondeterministic) Hardness vs. Non-Malleability

Marshall Ball (NYU), Dana Dachman-Soled (UMD), Julian Loss (CISPA)
Error Correcting Codes

- Goal: send message $m$
Error Correcting Codes

- Goal: send message $m$

$$m = 01010011010001$$
Error Correcting Codes

- Goal: send message $m$
- Problem: what if $m$ contains errors?

$$m = 01010011010001$$
Error Correcting Codes

- Goal: send message $m$
- Problem: what if $m$ contains errors?

$m = 01010111001001$
Error Correcting Codes

- Goal: send message $m$
- Problem: what if $m$ contains errors?
- Solution: error correcting codes

$$m = 01010111001001$$
Error Correcting Codes

- Goal: send message $m$
- Problem: what if $m$ contains errors?
- Solution: error correcting codes
Error Correcting Codes

- Goal: send message $m$
- Problem: what if $m$ contains errors?
- Solution: error correcting codes

$$ E(m) = 000101111001001 = c $$
Error Correcting Codes

- Goal: send message $m$
- Problem: what if $m$ contains errors?
- Solution: error correcting codes

$E(m) = 00010111001001 = c$

$D(c) = m$
Error Correcting Codes

- Goal: send message $m$
- Problem: what if $m$ contains errors?
- Solution: error correcting codes

$$E(m) = 00010011010001 = \hat{c}$$
Error Correcting Codes

- Goal: send message $m$
- Problem: what if $m$ contains errors?
- Solution: error correcting codes

$$E(m) = 00010011010001 = \hat{c}$$

$$D(\hat{c}) \overset{?}{=} m$$
Error Correcting Codes

- Goal: send message $m$
- Problem: what if $m$ contains errors?
- Solution: error correcting codes
- What if $\hat{c}$ doesn’t decode to $m$?

$$E(m) = 00010011010001 = \hat{c}$$

$$D(\hat{c}) \overset{?}{=} m$$
Tampering Attack

c = E(m)
Tampering Attack

$$c = E(m)$$
Tampering Attack

- Adversary may tamper $c$ into $\hat{c}$ s.t. $D(\hat{c}) = \hat{m} \neq m$

$$c = E(m)$$
Tampering Attack

- Adversary may tamper $c$ into $\hat{c}$ s.t. $D(\hat{c}) = \hat{m} \neq m$
Tampering Attack

- Adversary may tamper $c$ into $\hat{c}$ s.t. $D(\hat{c}) = \hat{m} \neq m$

$$D(\hat{c}) = \hat{m}$$
Tampering Attack

- Adversary may tamper $c$ into $\hat{c}$ s.t. $D(\hat{c}) = \hat{m} \neq m$
- Potentially devastating consequences!

$$D(\hat{c}) = \hat{m}$$
Tampering Attack

- Adversary may tamper \(c\) into \(\hat{c}\) s.t. \(D(\hat{c}) = \hat{m} \neq m\)
- Potentially devastating consequences!

\[
m = \text{Order } \text{pizza} \text{ for dinner} \quad D(\hat{c}) = \hat{m}
\]
Tampering Attack

- Adversary may tamper $c$ into $\hat{c}$ s.t. $D(\hat{c}) = \hat{m} \neq m$
- Potentially devastating consequences!

$\hat{m} = \text{Order for dinner}$

$D(\hat{c}) = \hat{m}$
Non-Malleable Codes (Dziembowski, Pietrzak, Wichs `10)
Non-Malleable Codes (Dziembowski, Pietrzak, Wichs ‘10)

- Non-Malleable Code: code $(E, D)$ that prevents tampering

\[
\begin{array}{c}
E \\
\end{array} \quad \rightarrow \quad \begin{array}{c}
D \\
\end{array}
\]
Non-Malleable Codes (Dziembowski, Pietrzak, Wichs `10)

- Non-Malleable Code: code $(E, D)$ that prevents tampering
Non-Malleable Codes (Dziembowski, Pietrzak, Wichs `10)

- Non-Malleable Code: code $(E, D)$ that prevents tampering
Non-Malleable Codes (Dziembowski, Pietrzak, Wichs `10)

- Non-Malleable Code: code \((E, D)\) that prevents tampering
Non-Malleable Codes (Dziembowski, Pietrzak, Wichs `10)

- Non-Malleable Code: code \((E,D)\) that prevents tampering
Non-Malleable Codes (Dziembowski, Pietrzak, Wichs `10)

- Non-Malleable Code: code \((E, D)\) that prevents tampering
- \(D\) either:
Non-Malleable Codes (Dziembowski, Pietrzak, Wichs `10)

- Non-Malleable Code: code $(E, D)$ that prevents tampering
- $D$ either:
  - Decodes correctly

\[
m \xrightarrow{\text{E}} \hat{c} \xrightarrow{\text{D}} m
\]
Non-Malleable Codes (Dziembowski, Pietrzak, Wichs `10)

- Non-Malleable Code: code \((E, D)\) that prevents tampering
- \(D\) either:
  - Decodes correctly
  - Outputs unrelated \(\hat{m}\)

\[ m \rightarrow E \rightarrow \hat{c} \rightarrow D \rightarrow m, \hat{m} \]
Defining Security
Defining Security

$m$
Defining Security

\[ m \xrightarrow{E} c \]
Defining Security

- Tampering modelled as function $f$
Defining Security

- Tampering modelled as function $f$
Defining Security

- Tampering modelled as function $f$
Defining Security

- Tampering modelled as function $f$
Tampering modelled as function $f$
Defining Security

- Tampering modelled as function $f$
- Sim samples same or $\hat{m}$ independently from $m$
Defining Security

- Tampering modelled as function $f$
- Sim samples same or $\hat{m}$ independently from $m$
Tampering modelled as function $f$
Sim samples same or $\hat{m}$ independently from $m$
Defining Security

- Tampering modelled as function $f$
- Sim samples same or $\hat{m}$ independently from $m$
Defining Security

- Tampering modelled as function $f$
- Sim samples same or $\hat{m}$ independently from $m$
Defining Security

- Tampering modelled as function $f$
- Sim samples same or $\hat{m}$ independently from $m$
Tampering modelled as function $f$
Sim samples same or $\hat{m}$ independently from $m$
Defining Security

- Tampering modelled as function $f$
- Sim samples same or $\hat{m}$ independently from $m$
Defining Security

- Tampering modelled as function $f$
- Sim samples same or $\hat{m}$ independently from $m$
- Induces indistinguishable distribution $D_f$ over identity/constant functions
Defining Security

- Tampering modelled as function $f$
- Sim samples same or $\hat{m}$ independently from $m$
- Induces indistinguishable distribution $D_f$ over identity/constant functions
Goals of this Work
Goals of this Work

- Efficient and explicit NMCs
Goals of this Work

- Efficient and explicit NMCs
- Plain model, plausible assumptions
Goals of this Work

- Efficient and explicit NMCs
- Plain model, plausible assumptions
- Interesting tampering classes: arbitrary polynomial size circuits
Goals of this Work

- Efficient and explicit NMCs
- Plain model, plausible assumptions
- Interesting tampering classes: arbitrary polynomial size circuits
- **Problem: goals are inherently conflicting!**
Goals of this Work

- Efficient and explicit NMCs
- Plain model, plausible assumptions
- Interesting tampering classes: arbitrary polynomial size circuits
- **Problem**: goals are inherently conflicting!

\[ m \]
Goals of this Work

- Efficient and explicit NMCs
- Plain model, plausible assumptions
- Interesting tampering classes: arbitrary polynomial size circuits
- **Problem**: goals are inherently conflicting!
Goals of this Work

- Efficient and explicit NMCs
- Plain model, plausible assumptions
- Interesting tampering classes: arbitrary polynomial size circuits
- **Problem:** goals are inherently conflicting!
Goals of this Work

- Efficient and explicit NMCs
- Plain model, plausible assumptions
- Interesting tampering classes: arbitrary polynomial size circuits
- **Problem: goals are inherently conflicting!**
Goals of this Work

- Efficient and explicit NMCs
- Plain model, plausible assumptions
- Interesting tampering classes: arbitrary polynomial size circuits
- **Problem: goals are inherently conflicting!**
Goals of this Work

- Efficient and explicit NMCs
- Plain model, plausible assumptions
- Interesting tampering classes: arbitrary polynomial size circuits
- **Problem: goals are inherently conflicting!**
Goals of this Work

- Efficient and explicit NMCs
- Plain model, plausible assumptions
- Interesting tampering classes: arbitrary polynomial size circuits
- **Problem:** goals are inherently conflicting!
Goals of this Work

- Efficient and explicit NMCs
- Plain model, plausible assumptions
- Interesting tampering classes: arbitrary polynomial size circuits
- **Problem:** goals are inherently conflicting!
Goals of this Work

- Efficient and explicit NMCs
- Plain model, plausible assumptions
- Interesting tampering classes: arbitrary polynomial size circuits
- **Problem:** goals are inherently conflicting!
Goals of this Work

- Efficient and explicit NMCs
- Plain model, plausible assumptions
- Interesting tampering classes: arbitrary polynomial size circuits
- **Problem**: goals are inherently conflicting!
Next Best Thing?
Next Best Thing?

- For every $c \in O(1)$, we give efficient NMC for $n^c$-size circuit tampering
For every $c \in O(1)$, we give efficient NMC for $n^c$-size circuit tampering

**Problem:** implies polynomial circuit lower bounds
For every $c \in O(1)$, we give efficient NMC for $n^c$-size circuit tampering

- **Problem:** implies polynomial circuit lower bounds
- **Solution:** assume such lower bounds!
Limitations of Prior Works
Limitations of Prior Works

- **Non-explicit** monte-carlo constructions:
Limitations of Prior Works

- **Non-explicit** monte-carlo constructions:
  - Cheraghchi Guruswami `14
Limitations of Prior Works

- **Non-explicit** monte-carlo constructions:
  - Cheraghchi Guruswami `14
  - Faust Mukherjee Venturi Wichs `14
Limitations of Prior Works

- **Non-explicit** monte-carlo constructions:
  - Cheraghchi Guruswami `14
  - Faust Mukherjee Venturi Wichs `14
- Computationally-secure constructions from strong crypto (currently requires ROM):
Limitations of Prior Works

- **Non-explicit** monte-carlo constructions:
  - Cheraghchi Guruswami `14
  - Faust Mukherjee Venturi Wichs `14

- Computationally-secure constructions from strong crypto (currently requires ROM):
  - Ball Dachman-Soled Kulkarni Lin Malkin `19
Limitations of Prior Works

- **Non-explicit** monte-carlo constructions:
  - Cheraghchi Guruswami `14
  - Faust Mukherjee Venturi Wichs `14

- Computationally-secure constructions from strong crypto (currently requires ROM):
  - Ball Dachman-Soled Kulkarni Lin Malkin `19
  - Dachman-Soled Komargodski Pass `20
Main Hardness Assumption and Theorem
Main Hardness Assumption and Theorem

Define $E = \text{DTIME}[2^{O(n)}]$
Main Hardness Assumption and Theorem

- Define $E = \text{DTIME}[2^{O(n)}]$

- **Conjecture 1:** $\exists \gamma \in (0,1), L \in E$ s.t. for almost all $n$, $L$ is undecidable for non-deterministic circuits of size $2^{\gamma \cdot n}$
Main Hardness Assumption and Theorem

- Define $E = \text{DTIME} \left[ 2^{O(n)} \right]$

- **Conjecture 1:** \( \exists \gamma \in (0,1), L \in E \) s.t. for almost all \( n \), \( L \) is undecidable for non-deterministic circuits of size \( 2^{\gamma \cdot n} \)

- Properties:
Main Hardness Assumption and Theorem

- Define $E = \text{DTIME}[2^{O(n)}]$

- **Conjecture 1:** $\exists \gamma \in (0, 1), L \in E$ s.t. for almost all $n$, $L$ is undecidable for non-deterministic circuits of size $2^{\gamma \cdot n}$

- Properties:
  - Worst-Case Assumption
Main Hardness Assumption and Theorem

- Define \( E = \text{DTIME} \left[ 2^{O(n)} \right] \)

- **Conjecture 1:** \( \exists \gamma \in (0,1), L \in E \) s.t. for almost all \( n \), \( L \) is undecidable for non-deterministic circuits of size \( 2^{\gamma \cdot n} \)

- Properties:
  - Worst-Case Assumption
  - \( E \) has complete problems
Main Hardness Assumption and Theorem

- Define $E = \text{DTIME}[2^{O(n)}]$

- **Conjecture 1:** $\exists \gamma \in (0,1), L \in E$ s.t. for almost all $n$, $L$ is undecidable for non-deterministic circuits of size $2^{\gamma \cdot n}$

- **Properties:**
  - Worst-Case Assumption
  - $E$ has complete problems
  - Orthogonal to crypto (to the best of our knowledge)
Define $E = \text{DTIME} \left[ 2^{O(n)} \right]$

**Conjecture 1:** $\exists \gamma \in (0,1), L \in E$ s.t. for almost all $n$, $L$ is undecidable for non-deterministic circuits of size $2^{\gamma \cdot n}$

**Properties:**
- Worst-Case Assumption
- $E$ has complete problems
- Orthogonal to crypto (to the best of our knowledge)

**Theorem:** Suppose that Conjecture 1 is true. Then, for all constants $c$, there exists an (explicit) $n^{-c}$-NMC for $n^c$-sized circuits.
Key Obstacle

- Code $(E, D)$ must be hard for $n^c$-sized circuits
Key Obstacle

- Code $(E, D)$ must be hard for $n^c$-sized circuits
- Reduction from Conjecture 1 must simulate tampering experiment
Key Obstacle

- Code \((E, D)\) must be hard for \(n^c\)-sized circuits
- Reduction from Conjecture 1 must simulate tampering experiment
- **Solution**: Non-deterministic reduction + strong statistical tool
Bounded Communication Tampering
Bounded Communication Tampering

- Main ingredient: split state tampering with bounded communication
Bounded Communication Tampering

- Main ingredient: split state tampering with bounded communication

\[ E(m) \]
Main ingredient: split state tampering with bounded communication

\[ E(m) \]
Main ingredient: split state tampering with bounded communication
Main ingredient: split state tampering with bounded communication

\[ E(m) \]

\[ T \ll |c_A|, |c_B| \]
Main ingredient: split state tampering with bounded communication
Bounded Communication Tampering

- Main ingredient: split state tampering with bounded communication
- Known NMCs for this tampering class in the standard model

\[ E(m) \]

\[ C_A \]

\[ A \]

\[ T \ll |c_A|, |c_B| \]

\[ C_B \]

\[ B \]

\[ \tilde{C}_A \]

\[ \tilde{C}_B \]
Our Construction for $n^c$-Size Circuits
Our Construction for $n^c$-Size Circuits

- Start from:
Our Construction for $n^c$-Size Circuits

- Start from:
  - NMC $(\tilde{E}, \tilde{D})$ for split-state bounded communication tampering
Our Construction for $n^c$-Size Circuits

- Start from:
  - NMC $(\tilde{E}, \tilde{D})$ for split-state bounded communication tampering
  - $\text{PRG}(s) \approx$ uniform for non-deterministic circuits of size $n^c$
Our Construction for $n^c$-Size Circuits

- Start from:
  - NMC ($\tilde{E}, \tilde{D}$) for split-state bounded communication tampering
  - PRG($s$) $\approx$ uniform for non-deterministic circuits of size $n^c$
  - $E(x) = (s, c_B)$ s.t. $(\text{PRG}(s), c_B) \in \tilde{E}(x)$
Our Construction for $n^c$-Size Circuits

- Start from:
  - NMC ($\tilde{E}, \tilde{D}$) for split-state bounded communication tampering
  - PRG($s$) $\approx$ uniform for non-deterministic circuits of size $n^c$
- $E(x) = (s, c_B)$ s.t. $(\text{PRG}(s), c_B) \in \tilde{E}(x)$
- $D(s', c'_B) = \tilde{D}(\text{PRG}(s'), c'_B)$
Proof Idea

A \rightarrow B

A \leftarrow B
Proof Idea

\[ c_A \leftarrow \{0,1\}^n \]

\[ A \leftrightarrow C_B \]

\[ B \]
Code is secure if $c_A$ is random
Proof Idea

- Code is secure if $c_A$ is random
- Assume that code is broken if $c_A = \text{PRG}(s)$
Proof Idea

- Code is secure if \( c_A \) is random
- Assume that code is broken if \( c_A = \text{PRG}(s) \)
- Then there exists efficient tampering \( f \)
Proof Idea

- Code is secure if $c_A$ is random
- Assume that code is broken if $c_A = \text{PRG}(s)$
- Then there exists efficient tampering $f$
Proof Idea

- Code is secure if $c_A$ is random
- Assume that code is broken if $c_A = \text{PRG}(s)$
- Then there exists efficient tampering $f$
Proof Idea

- Code is secure if $c_A$ is random
- Assume that code is broken if $c_A = \text{PRG}(s)$
- Then there exists efficient tampering $f$
- Code is secure if $c_A$ is random
- Assume that code is broken if $c_A = \text{PRG}(s)$
- Then there exists efficient tampering $f$
Proof Idea

- Code is secure if $c_A$ is random
- Assume that code is broken if $c_A = \text{PRG}(s)$
- Then there exists efficient tampering $f$
- Leads to a distinguisher on PRG
Proof Idea

- Code is secure if $c_A$ is random
- Assume that code is broken if $c_A = \text{PRG}(s)$
- Then there exists efficient tampering $f$
- Leads to a distinguisher on PRG

\[
\begin{align*}
    c_A &= \text{PRG}(s) \\
    \tilde{c}_A &= \text{PRG}(\tilde{s}) \\
    (\tilde{s}, \tilde{c}_B) &= f(s, c_B)
\end{align*}
\]
Merlin-Arthur Protocol

\[ y = \text{PRG}(s) \]
Merlin-Arthur Protocol

- Protocol accepts \((s, \text{PRG}(s))\) and rejects \((s, U)\)
Merlin-Arthur Protocol

- Protocol accepts \((s, \text{PRG}(s))\) and rejects \((s, U)\)
- Merlin is unbounded, can evaluate PRG

\[ y = \text{PRG}(s) \]
Merlin-Arthur Protocol

- Protocol accepts \((s, \text{PRG}(s))\) and rejects \((s, U)\)
- Merlin is unbounded, can evaluate PRG
- Arthur is efficient as tampering \(f\) is efficient
Merlin-Arthur Protocol

- Protocol accepts \( (s, \text{PRG}(s)) \) and rejects \( (s, U) \)
- Merlin is unbounded, can evaluate PRG
- Arthur is efficient as tampering \( f \) is efficient
- Turn into a non-deterministic distinguisher for PRG via known techniques
Thanks!