Non-malleable Codes against Lookahead Tampering

Divya Gupta\textsuperscript{1}, Hemanta K. Maji\textsuperscript{2}, Mingyuan Wang\textsuperscript{2}

December 12, 2018

\textsuperscript{1}Microsoft Research, Banaglore, India, divya.gupta@microsoft.com
\textsuperscript{2}Purdue University, \{hmaji,wang1929\}@purdue.edu
Non-malleable message transmission

Can we ensure non-malleability information theoretically?
Non-malleable message transmission

Can we ensure non-malleability information theoretically?
Non-malleable message transmission

Can we ensure non-malleability information theoretically?
Suppose we have a coding scheme \((\text{Enc}, \text{Dec})\):

- \(\text{Enc} : \{0, 1\}^\ell \rightarrow \{0, 1\}^n\) (possibly probabilistic)
- \(\text{Dec} : \{0, 1\}^n \rightarrow \{0, 1\}^\ell \cup \{\perp\}\) (deterministic)

\[
\begin{align*}
  m & \xrightarrow{\text{Enc}} c & \xrightarrow{\text{Dec}} m
\end{align*}
\]
Suppose we have a coding scheme (Enc, Dec):

- Enc : \( \{0, 1\}^\ell \rightarrow \{0, 1\}^n \) (possibly probabilistic)
- Dec : \( \{0, 1\}^n \rightarrow \{0, 1\}^\ell \cup \{\perp\} \) (deterministic)

We want to ensure that \( \text{Tampering } f \) is either the original message \( m \) or some unrelated message \( m^* \). We cannot allow \( f \) to be an arbitrary function. For example, consider the following tampering function:

\[
 f(c) = \text{Enc}(\text{Dec}(c) + 1)
\]

Defined w.r.t. some fixed tampering family \( F \).
Non-malleable Codes

Suppose we have a coding scheme \((\text{Enc}, \text{Dec})\):

- \(\text{Enc} : \{0, 1\}^\ell \rightarrow \{0, 1\}^n\) (possibly probabilistic)
- \(\text{Dec} : \{0, 1\}^n \rightarrow \{0, 1\}^\ell \cup \{\perp\}\) (deterministic)

Distribution of the Tampered Message: \(\text{Tamper}_{f}^{m}\)
Non-malleable Codes

Suppose we have a coding scheme \((\text{Enc}, \text{Dec})\):

- **Enc**: \(\{0, 1\}^\ell \rightarrow \{0, 1\}^n\) (possibly probabilistic)
- **Dec**: \(\{0, 1\}^n \rightarrow \{0, 1\}^{\ell} \cup \{\perp\}\) (deterministic)

\[
\begin{array}{cccc}
m & \xrightarrow{\text{Enc}} & c & \xrightarrow{\text{Tampering } f} \tilde{c} & \xrightarrow{\text{Dec}} \tilde{m} \\
\end{array}
\]

Distribution of the Tampered Message: \(\text{Tamper}_f^m\)

- We want to ensure that \(\text{Tamper}_f^m\) is either the original message \(m\) or some unrelated message \(m^*\)
Non-malleable Codes

Suppose we have a coding scheme \((\text{Enc}, \text{Dec})\):

- \(\text{Enc} : \{0, 1\}^\ell \rightarrow \{0, 1\}^n\) (possibly probabilistic)
- \(\text{Dec} : \{0, 1\}^n \rightarrow \{0, 1\}^\ell \cup \{\perp\}\) (deterministic)

Distribution of the Tampered Message: \(\text{Tamper}_{f}^{m}\)

- We want to ensure that \(\text{Tamper}_{f}^{m}\) is either the original message \(m\) or some unrelated message \(m^*\)
- We cannot allow \(f\) to be an arbitrary function
  For example, consider the following tampering function

\[
f(c) = \text{Enc} \left( \text{Dec}(c) + 1 \right)
\]
Non-malleable Codes

Suppose we have a coding scheme \((\text{Enc, Dec})\):
- \(\text{Enc} : \{0, 1\}^\ell \rightarrow \{0, 1\}^n\) (possibly probabilistic)
- \(\text{Dec} : \{0, 1\}^n \rightarrow \{0, 1\}^\ell \cup \{\perp\}\) (deterministic)

Distribution of the Tampered Message: \(\text{Tamper}^m_f\)

- We want to ensure that \(\text{Tamper}^m_f\) is either the original message \(m\) or some unrelated message \(m^*\)
- We cannot allow \(f\) to be an arbitrary function
  For example, consider the following tampering function
  \[ f(c) = \text{Enc}\left(\text{Dec}(c) + 1\right) \]
- Defined w.r.t. some fixed tampering family \(\mathcal{F}\)
Non-malleable Codes: Dziembowski, Pietrzak and Wichs [ICS-10]
Non-malleable Codes: Dziembowski, Pietrzak and Wichs [ICS-10]

For all tampering function $f$ belonging to the tampering family $\mathcal{F}$,
Non-malleable Codes: Dziembowski, Pietrzak and Wichs [ICS-10]

For all tampering function $f$ belonging to the tampering family $\mathcal{F}$, there exists a simulator $\text{Sim}_f$,

- Simulator can either output a fixed message $m^*$ or indicate that tampering occurred by outputting ⊥.
  - We also allow $\text{Sim}_f$ to output a special symbol same* to indicate the message remained unchanged.
Non-malleable Codes: Dziembowski, Pietrzak and Wichs [ICS-10]

For all tampering function $f$ belonging to the tampering family $\mathcal{F}$, there exists a simulator $\text{Sim}_f$, such that for all message $m$
Non-malleable Codes: Dziembowski, Pietrzak and Wichs [ICS-10]

For all tampering function $f$ belonging to the tampering family $F$, there exists a simulator $Sim_f$, such that for all message $m$

$$\text{Tamper}^m_f \approx Sim_f$$
Non-malleable Codes: Dziembowski, Pietrzak and Wichs [ICS-10]

For all tampering function $f$ belonging to the tampering family $\mathcal{F}$, there exists a simulator $\text{Sim}_f$, such that for all message $m$

$$\text{Tamper}^m_f \approx \text{Sim}_f$$

- Simulator can either output a fixed message $m^*$ or indicate that tampering occurred by outputting $\perp$
Non-malleable Codes: Dziembowski, Pietrzak and Wichs [ICS-10]

For all tampering function $f$ belonging to the tampering family $\mathcal{F}$, there exists a simulator $\text{Sim}_f$, such that for all message $m$

$$\text{Tamper}_f^m \approx \text{Sim}_f$$

- Simulator can either output a fixed message $m^*$ or indicate that tampering occurred by outputting $\bot$
- We also allow $\text{Sim}_f$ to output a special symbol same* to indicate the message remained unchanged
Non-malleable Codes: Dziembowski, Pietrzak and Wichs [ICS-10]

For all tampering function $f$ belonging to the tampering family $\mathcal{F}$, there exists a simulator $\text{Sim}_f$, such that for all message $m$

$$\text{Tamper}_f^m \approx \text{copy}(\text{Sim}_f, m)$$

- Simulator can either output a fixed message $m^*$ or indicate that tampering occurred by outputting $\bot$
- We also allow $\text{Sim}_f$ to output a special symbol same* to indicate the message remained unchanged
In the $k$-Lookahead Tampering model, codeword $c$ is divided into $k$ shares $(c_1, c_2, \ldots, c_k)$. Each share $c_i$ is tampered by a tampering function $f_i$ independently. Each tampering function $f_i$ tampers the corresponding codeword $c_i$ in a streaming manner. If the adversary blocks or slows the information stream, it would outrightly signal his intrusion.
In $k$-lookahead tampering model, codeword $c$ is divided into $k$ shares $(c_1, c_2, \ldots, c_k)$
In the $k$-lookahead tampering model, the codeword $c$ is divided into $k$ shares $(c_1, c_2, \ldots, c_k)$. Each share $c_i$ is tampered by a tampering function $f_i$ independently.
In \(k\)-lookahead tampering model, codeword \(c\) is divided into \(k\) shares \((c_1, c_2, \ldots, c_k)\). Each share \(c_i\) is tampered by a tampering function \(f_i\) independently. Each tampering function \(f_i\) tampers the corresponding codeword \(c_i\) in a streaming manner.
In $k$-lookahead tampering model, codeword $c$ is divided into $k$ shares $(c_1, c_2, \ldots, c_k)$.

- Each share $c_i$ is tampered by a tampering function $f_i$ independently.
- Each tampering function $f_i$ tampers the corresponding codeword $c_i$ in a streaming manner.
- If the adversary blocks or slows the information stream, it would outrightly signal his intrusion.
$k$-Lookahead Tampering Model

Strongest: when the number of blocks $m$ equals to 1. This is the widely studied $k$-split-state tampering. 

Weakest: when the number of blocks $m$ equals to the length of $c_i$, i.e. each bit is a block.
### $k$-Lookahead Tampering Model

**$c_i$:**

\[
c_i^{(1)} \quad c_i^{(2)} \quad c_i^{(3)} \quad \ldots \quad c_i^{(m)}
\]

**$\tilde{c}_i$:**

\[
\tilde{c}_i^{(1)} \quad \tilde{c}_i^{(2)} \quad \tilde{c}_i^{(3)} \quad \ldots \quad \tilde{c}_i^{(m)}
\]

- **Strongest:** when the number of blocks $m$ equals to 1. This is the widely studied $k$-split-state tampering.
- **Weakest:** when the number of blocks $m$ equals to the length of $c_i$, i.e. each bit is a block.
$k$-Lookahead Tampering Model

\[ c_i: \quad \tilde{c}_i: \]

\[ c_i^{(1)} \quad c_i^{(2)} \quad c_i^{(3)} \quad \ldots \quad c_i^{(m)} \]

\[ f_i^{(1)} \]

Strongest: when the number of blocks $m$ equals to 1. This is the widely studied $k$-split-state tampering.

Weakest: when the number of blocks $m$ equals to the length of $c_i$, i.e. each bit is a block.
$k$-Lookahead Tampering Model

$\tilde{c}_i$: 

\[
\begin{array}{cccccc}
\tilde{c}_i^{(1)} & \tilde{c}_i^{(2)} & \tilde{c}_i^{(3)} & \cdots & \tilde{c}_i^{(m)} \\
\end{array}
\]

$c_i$: 

\[
\begin{array}{cccccc}
c_i^{(1)} & c_i^{(2)} & c_i^{(3)} & \cdots & c_i^{(m)} \\
\end{array}
\]

Strongest: when the number of blocks $m$ equals to 1. This is the widely studied $k$-split-state tampering.

Weakest: when the number of blocks $m$ equals to the length of $c_i$, i.e. each bit is a block.
$k$-Lookahead Tampering Model

Strongest: when the number of blocks $m$ equals to 1. This is the widely studied $k$-split-state tampering.

Weakest: when the number of blocks $m$ equals to the length of $c_i$, i.e. each bit is a block.
$k$-Lookahead Tampering Model

$\tilde{c}_i$: 

$\tilde{c}_i^{(1)} \quad \tilde{c}_i^{(2)} \quad \tilde{c}_i^{(3)} \quad \ldots \quad \tilde{c}_i^{(m)}$

$c_i$: 

$c_i^{(1)} \quad c_i^{(2)} \quad c_i^{(3)} \quad \ldots \quad c_i^{(m)}$

$f_i^{(m)}$ 

Strongest: when the number of blocks $m$ equals to 1. This is the widely studied $k$-split-state tampering.

Weakest: when the number of blocks $m$ equals to the length of $c_i$, i.e. each bit is a block.
Strongest: when the number of blocks $m$ equals to 1. This is the widely studied $k$-split-state tampering

Weakest: when the number of blocks $m$ equals to the length of $c_i$, i.e. each bit is a block
Our Contributions

For $k$-split state tampering, Cheraghchi and Guruswami [ITCS-14] showed that the best achievable rate $\ell/n \leq 1 - 1/k$.

Upper Bound

For $k$-lookahead, the best achievable rate is $1 - 1/k$.

NMC against 2-lookahead

There exists an efficient non-malleable code, with negligible simulation error, against the 2-lookahead tampering with rate $1/3$.

NMC against 3-split-state

There exists an efficient non-malleable code, with negligible simulation error, against the 3-split-state tampering with rate $1/3$.

In an independent and concurrent work, Kanukurthi, Obbattu and Sekar [EUROCRYPT-18] also obtained a rate $1/3$ construction in 3-split-state.
Our Contributions

For $k$-split-state tampering, Cheraghchi and Guruswami [ITCS-14] showed that the best achievable rate $\frac{\ell}{n} \leq 1 - \frac{1}{k}$

**Upper Bound**

For $k$-lookahead, the best achievable rate is $1 - \frac{1}{k}$
Our Contributions

For $k$-split-state tampering, Cheraghchi and Guruswami [ITCS-14] showed that the best achievable rate $\ell/n \leq 1 - 1/k$

**Upper Bound**

For $k$-lookahead, the best achievable rate is $1 - 1/k$

**NMC against 2-lookahead**

There exists an efficient non-malleable code, with negligible simulation error, against the 2-lookahead tampering with rate $1/3$
Our Contributions

For $k$-split-state tampering, Cheraghchi and Guruswami [ITCS-14] showed that the best achievable rate $\ell/n \leq 1 - 1/k$

**Upper Bound**

For $k$-lookahead, the best achievable rate is $1 - 1/k$

**NMC against 2-lookahead**

There exists an efficient non-malleable code, with negligible simulation error, against the 2-lookahead tampering with rate $1/3$

**NMC against 3-split-state**

There exists an efficient non-malleable code, with negligible simulation error, against the 3-split-state tampering with rate $1/3$

In an independent and concurrent work, Kanukurthi, Obbattu and Sekar [EUROCRYPT-18] also obtained a rate $1/3$ construction in 3-split-state
Related Works: Split-State

In prior works, the construction of $k$-lookahead NMC coincided with $k$-split-state NMC.
In prior works, the construction of $k$-lookahead NMC coincided with $k$-split-state NMC

<table>
<thead>
<tr>
<th>$k$</th>
<th>Work</th>
<th>Best Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Chattopadhyay and Zuckerman [FOCS-14]</td>
<td>(small) const.</td>
</tr>
<tr>
<td>4</td>
<td>Kanukurthi, Obbattu and Sekar [TCC-17]</td>
<td>1/3</td>
</tr>
<tr>
<td>3</td>
<td>Kanukurthi, Obbattu and Sekar [EUROCRYPT-18]</td>
<td>1/3</td>
</tr>
</tbody>
</table>
| 2   | Dziembowski, Kazana and Obremski [CRYPTO-13]  
Aggarwal, Dodis and Lovett [STOC-14]  
Aggarwal, Dodis, Kazana and Obremski [STOC-15]  
Li [STOC-17] | 1/\log n |
Theorem 1

For $k$-lookahead, the best achievable rate is $1 - 1/k$
Theorem 1

For $k$-lookahead, the best achievable rate is $1 - 1/k$

- We prove it for the weakest model, where tampering function tampers one-bit at a time
Theorem 1

For $k$-lookahead, the best achievable rate is $1 - 1/k$

- We prove it for the weakest model, where tampering function tampers one-bit at a time
- Cheraghchi and Guruswami [ITCS-14] showed that if the rate is higher than $1 - 1/k$, there will exist a distinguisher $D$ and two messages $m_0, m_1$ such that $D$ can use the longest state to distinguish the encoding of $m_0$ and $m_1$
Theorem 1

For $k$-lookahead, the best achievable rate is $1 - 1/k$

- We prove it for the weakest model, where tampering function tampers one-bit at a time
- Cheraghchi and Guruswami [ITCS-14] showed that if the rate is higher than $1 - 1/k$, there will exist a distinguisher $D$ and two messages $m_0, m_1$ such that $D$ can use the longest state to distinguish the encoding of $m_0$ and $m_1$

State-1  \[\rightarrow\]  State-2  \[\cdots\]  State-$k$

Reveal information about the message
Except for the last block of the longest state, we will rewrite the codeword to with a fixed codeword $c^*$, which encodes a fixed message $\text{Dec}(c^*) = m^*$.
Except for the last block of the longest state, we will rewrite the codeword to with a fixed codeword $c^*$, which encodes a fixed message $\text{Dec}(c^*) = m^*$.

At the last block, we invoke the distinguisher $D$. Depending on the distinguisher output, we either fill in the last block of $c^*$ or make the codeword invalid.
Except for the last block of the longest state, we will rewrite the codeword to with a fixed codeword $c^*$, which encodes a fixed message $\text{Dec}(c^*) = m^*$

At the last block, we invoke the distinguisher $\mathcal{D}$. Depending on the distinguisher output, we either fill in the last block of $c^*$ or make the codeword invalid

The probability of

$$\Pr \left[ \text{Tamper}^0_f = m^* \right] \quad \text{and} \quad \Pr \left[ \text{Tamper}^1_f = m^* \right]$$

will be significantly different
There exists an efficient non-malleable code, with negligible simulation error, against the 2-lookahead tampering with rate 1/3
NMC against 2-lookahead

There exists an efficient non-malleable code, with negligible simulation error, against the 2-lookahead tampering with rate $1/3$.

- Kanukurthi, Obbattu and Sekar [TCC-17]’s construction of four-state NMC is the starting point of our construction.
Construction of Kanukurthi, Obbattu and Sekar
[TCC-17]
Construction of Kanukurthi, Obbattu and Sekar
[TCC-17]
Construction of Kanukurthi, Obbattu and Sekar
[TCC-17]
Construction of Kanukurthi, Obbattu and Sekar
[TCC-17]
Construction of Kanukurthi, Obbattu and Sekar
[TCC-17]
Construction of Kanukurthi, Obbattu and Sekar
[TCC-17]
Now, instead of storing $c$, $w$, $L$, $R$ individually, we are going to merge some states and store it as $(w, R) \rightarrow$ and $(L, c) \rightarrow$

This results in two issues:
- The tampering on $R$ now depends on $w$
- The tampering on $c$ now depends on $L$
Now, instead of storing $c, w, L, R$ individually, we are going to merge some states and store it as $(w, R)$ and $(L, c)$.
Now, instead of storing \( c, w, L, R \) individually, we are going to merge some states and store it as \((w, R)\) and \((L, c)\).

This results in two issues:

- The tampering on \( R \) now depends on \( w \)
- The tampering on \( c \) now depends on \( L \)
Our Modification

- Our codeword: \((w, R)\) and \((L, c)\)

The tampering on \(R\) influences only Tag and seed. We view this as additional leakage on \(w\).

We use an additional property of Aggarwal, Dodis and Lovett [STOC-14]'s construction, called augmented non-malleability (identified by Aggarwal et al. [AAG+16]). At an intuitive level, this property allows us the freedom to simulate the left state \(L\) and, hence, simulate the tampering on \(c\).
Our Modification

- Our codeword: \((w, R)\) and \((L, c)\)
- The tampering on \(R\) influence only Tag and seed \(s\).
Our Modification

- Our codeword: \((w, R)\) and \((L, c)\)
- The tampering on \(R\) influence only Tag and seed \(s\). We view this as additional leakage on \(w\)
Our Modification

Our codeword: \((w, R)\) and \((L, c)\)

The tampering on \(R\) influence only Tag and seed \(s\). We view this as additional leakage on \(w\)

We use an additional property of Aggarwal, Dodis and Lovett [STOC-14]'s construction, called \textit{augmented non-malleability} (identified by Aggarwal et al. [AAG+16]). At an intuitive level, this property allows us the freedom to simulate the left state \(L\) and, hence, simulate the tampering on \(c\)
There exists an efficient non-malleable code, with negligible simulation error, against the 3-split-state tampering with rate $1/3$. 

Our 3-state codeword: $c$, $(L, w) \leftarrow \text{XOR} \implies (R, s) \leftarrow \text{2-state NMC}$. 

Tampering on $L$ and $w$ depends on each other, resolved similarly as 2-lookahead proof.
3-state NMC

NMC against 3-split-state

There exists an efficient non-malleable code, with negligible simulation error, against the 3-split-state tampering with rate $1/3$

Our 3-state codeword: $c$, $(L, w)$ and $R$
There exists an efficient non-malleable code, with negligible simulation error, against the 3-split-state tampering with rate $1/3$.

Our 3-state codeword: $c, (L, w)$ and $R$

Tampering on $L$ and $w$ depends on each other

Resolved similarly as 2-lookahead proof
Summary of Our Results

- For \( k \)-lookahead, the best achievable rate is \( 1 - 1/k \)
- There exists an efficient non-malleable code, with negligible simulation error, against the 2-lookahead tampering with rate 1/3
- There exists an efficient non-malleable code, with negligible simulation error, against the 3-split-state tampering with rate 1/3

Thanks!