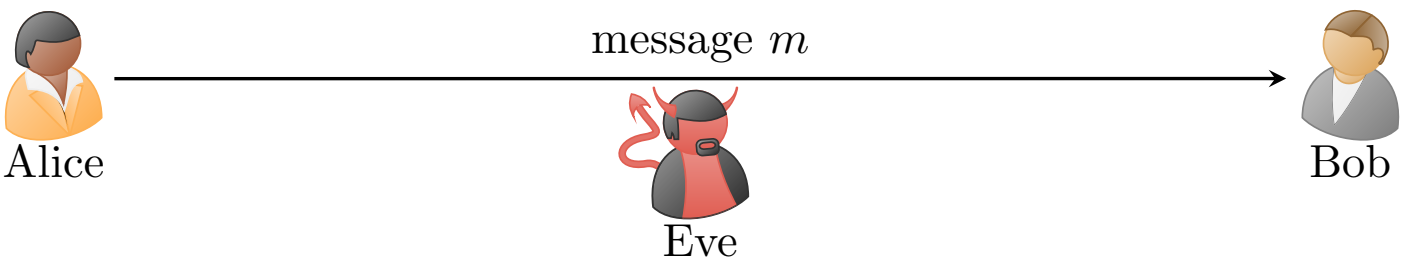
Non-malleable Codes against Lookahead Tampering

Divya Gupta[1], Hemanta K. Maji[2], Mingyuan Wang[2]

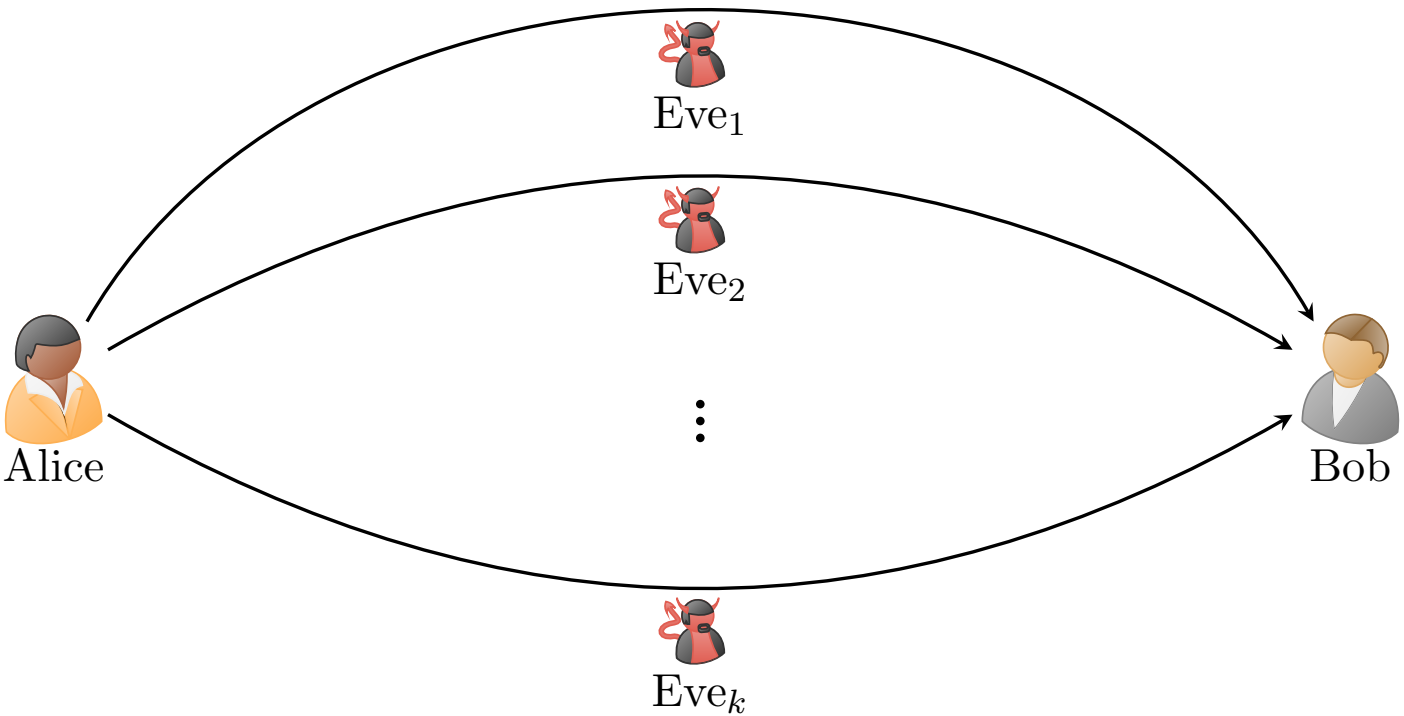
December 12, 2018

[1]Microsoft Research, Banaglore, India, divya.gupta@microsoft.com
[2]Purdue University, {hmaji,wang1929}@purdue.edu

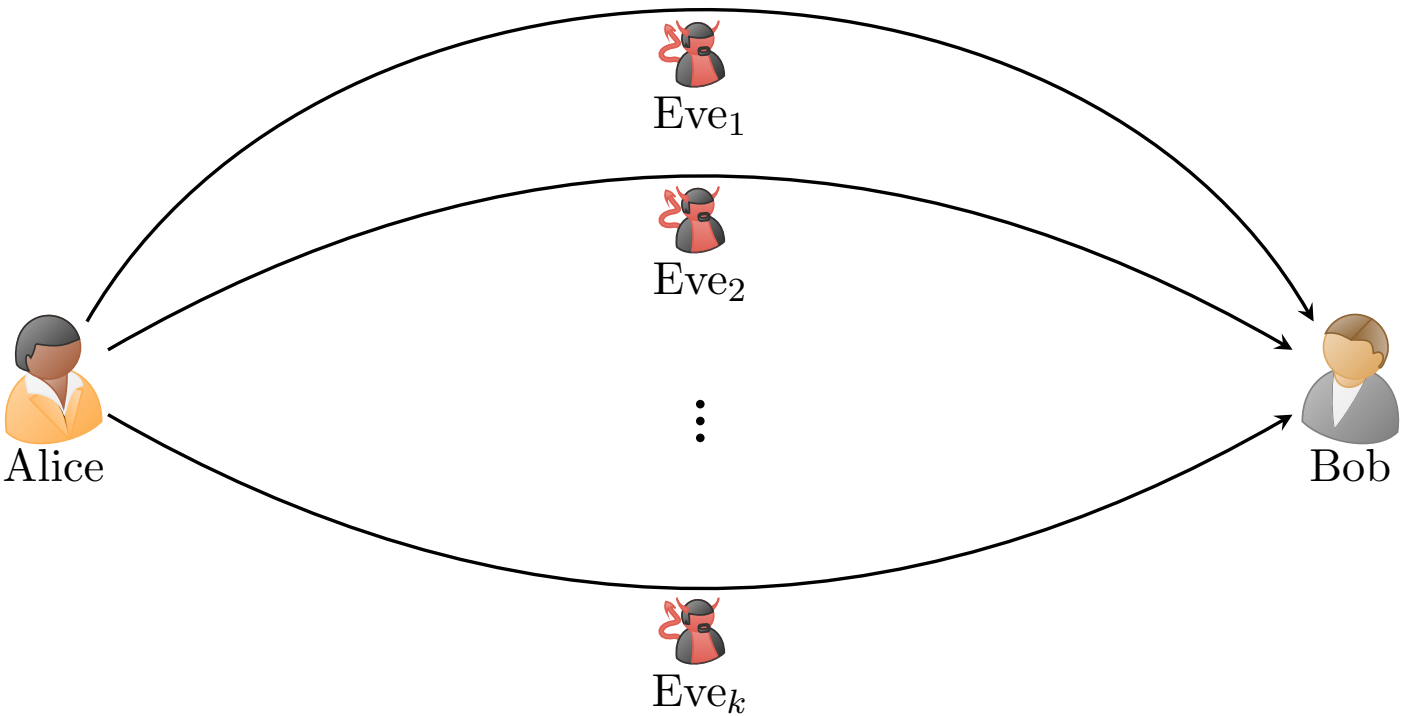
Non-malleable message transmission



Non-malleable message transmission



Non-malleable message transmission



Can we ensure non-malleability information theoretically?

Non-malleable Codes

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

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

$$m \xrightarrow{\mathrm{Enc}} c \xrightarrow{\mathrm{Dec}} m$$

Non-malleable Codes

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

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

$$m \xrightarrow{\text{Enc}} c \xrightarrow{\text{Tampering } f} \widetilde{c} \xrightarrow{\text{Dec}} \widetilde{m}$$

Non-malleable Codes

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

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

Distribution of the Tampered Message: Tamper_{f}^{m}

Non-malleable Codes

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

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

Distribution of the Tampered Message: Tamper_f^m

- We want to ensure that Tamper_f^m is either the original message m or some unrelated message m^*

Non-malleable Codes

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

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

Distribution of the Tampered Message: Tamper_f^m

- We want to ensure that 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

$$f(c) = \mathrm{Enc}\Big(\mathrm{Dec}(c) + 1\Big)$$

Non-malleable Codes

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

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

Distribution of the Tampered Message: Tamper_f^m

- We want to ensure that 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

$$f(c) = \mathrm{Enc}\Big(\mathrm{Dec}(c) + 1\Big)$$

- Defined w.r.t. some fixed tampering family $\mathcal{F}$

Non-malleable Codes

Non-malleable Codes: Dziembowski, Pietrzak and Wichs [ICS-10]

Non-malleable Codes

Non-malleable Codes: Dziembowski, Pietrzak and Wichs [ICS-10]

For all tampering function f belonging to the tampering family $\mathcal{F}$,

Non-malleable Codes

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 Sim_f,

Non-malleable Codes

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 Sim_f, such that for all message m

Non-malleable Codes

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 Sim_f, such that for all message m

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

Non-malleable Codes

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 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 $\perp$

Non-malleable Codes

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 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 $\perp$
- We also allow Sim_f to output a special symbol same* to indicate the message remained unchanged

Non-malleable Codes

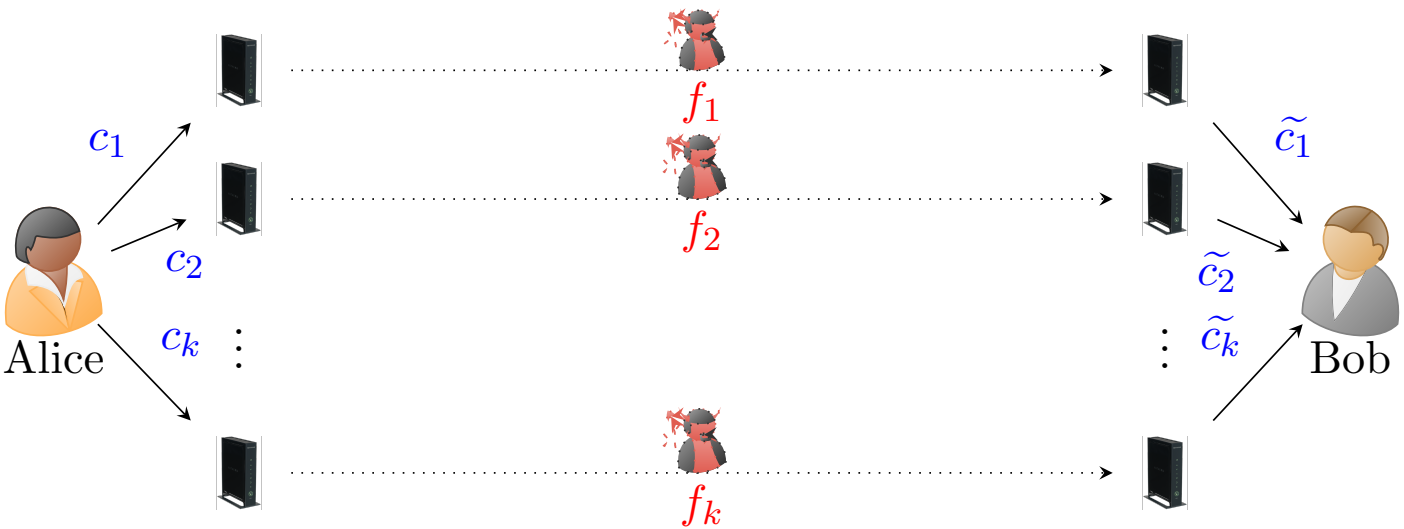
Non-malleable Codes: Dziembowski, Pietrzak and Wichs [ICS-10]

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

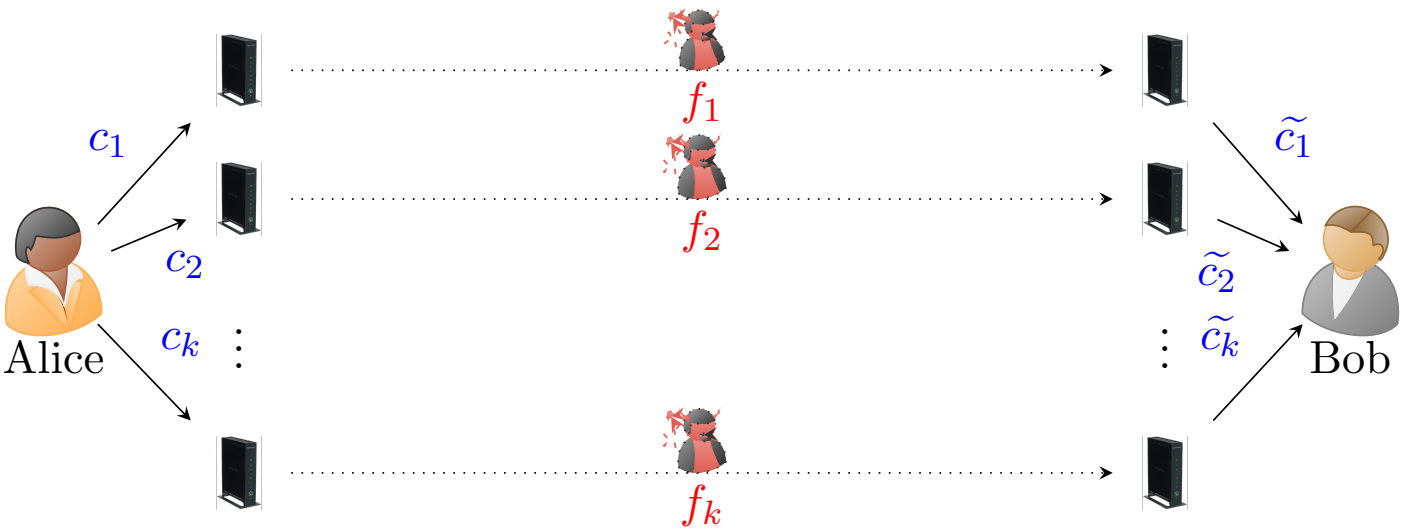
$$\mathrm{Tamper}_f^m \approx \mathrm{copy}(\mathrm{Sim}_f, m)$$

- Simulator can either output a fixed message m^* or indicate that tampering occurred by outputting $\perp$
- We also allow Sim_f to output a special symbol same* to indicate the message remained unchanged

k-Lookahead Tampering model

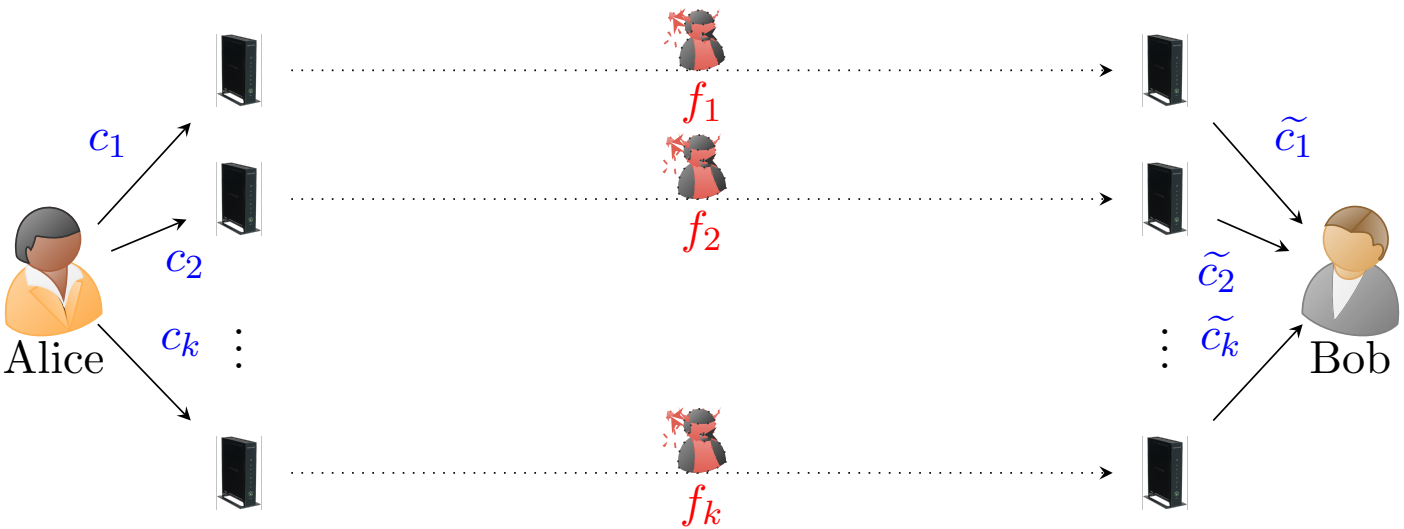


k-Lookahead Tampering model



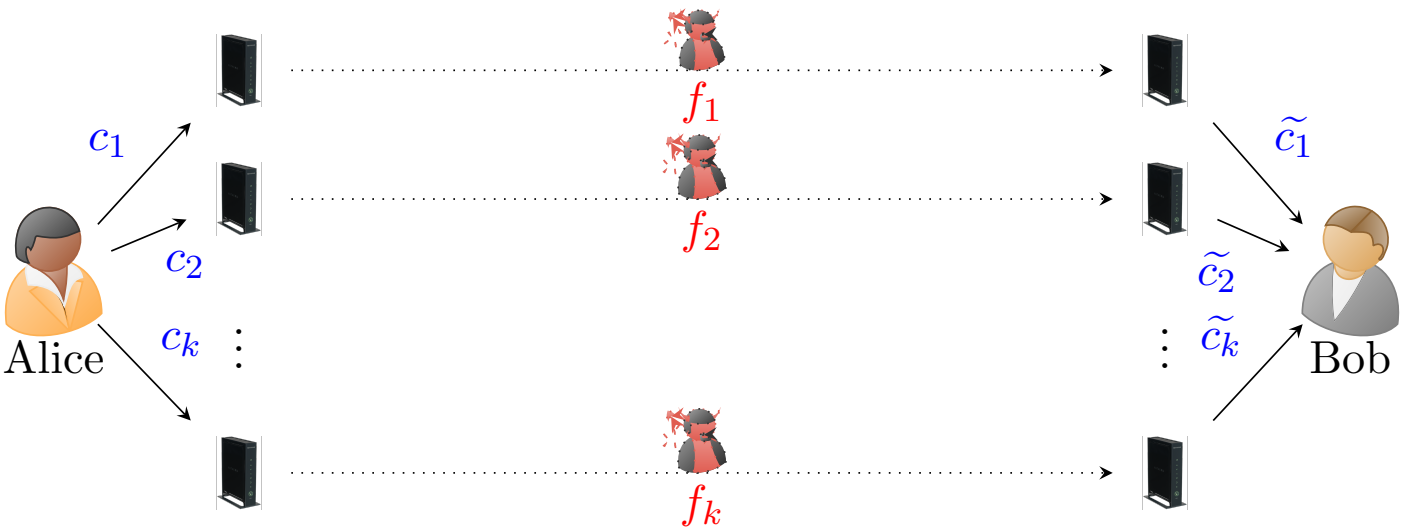
- In k-lookahead tampering model, codeword c is divided into k shares $(c_1, c_2, \ldots, c_k)$

k-Lookahead Tampering model



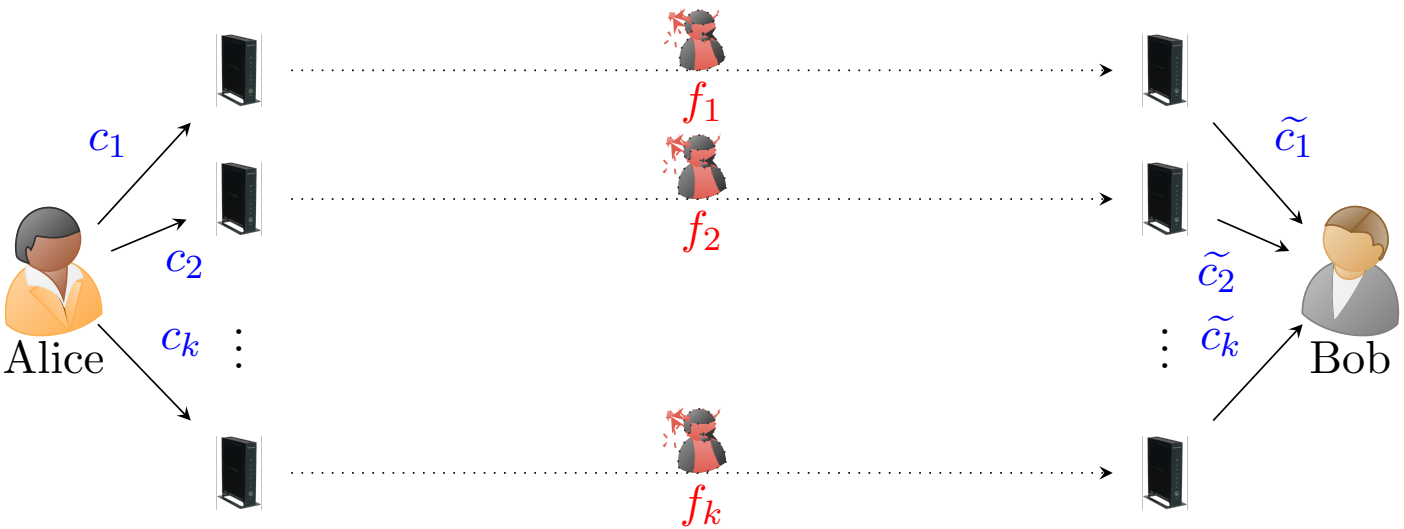
- In k-lookahead tampering model, codeword c is divided into k shares $(c_1, c_2, \ldots, c_k)$
- Each share c_i is tampered by a tampering function f_i independently

k-Lookahead Tampering model



- 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

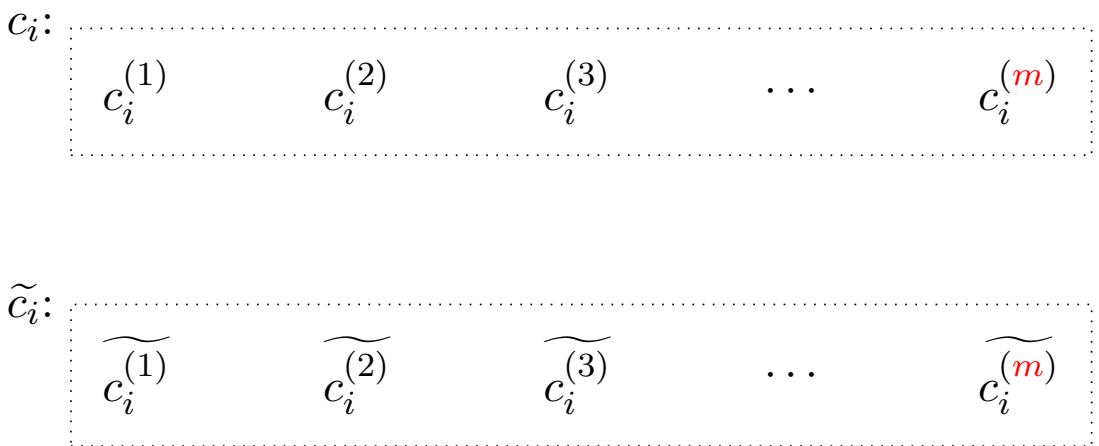
k-Lookahead Tampering model



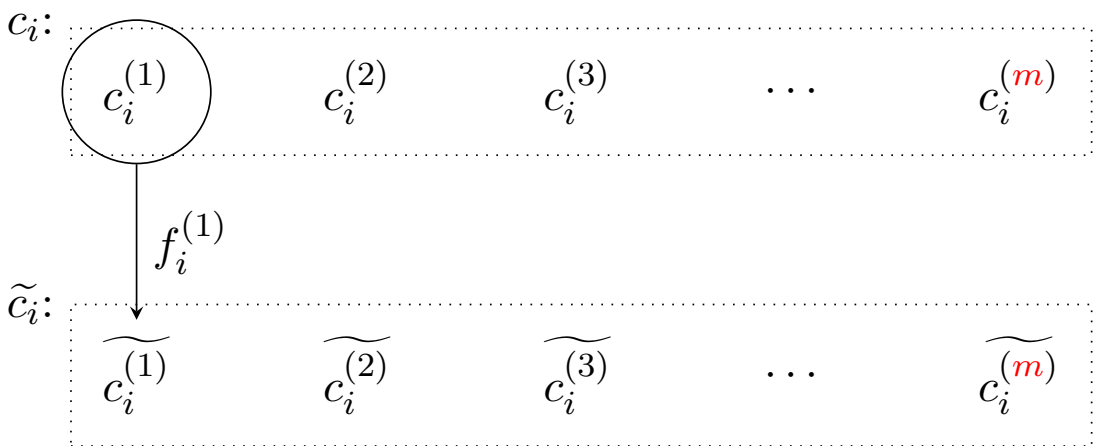
- In k-lookahead tampering model, codeword c is divided into k shares $(c_1, c_2, \ldots, c_k)$
- Each share c_i is tampered by a tampering function f_i independently
- Each tampering function f_i tampers the corresponding codeword c_i in a streaming manner
- If the adversary blocks or slows the information stream, it would outrightly signal his intrusion

k-Lookahead Tampering Model

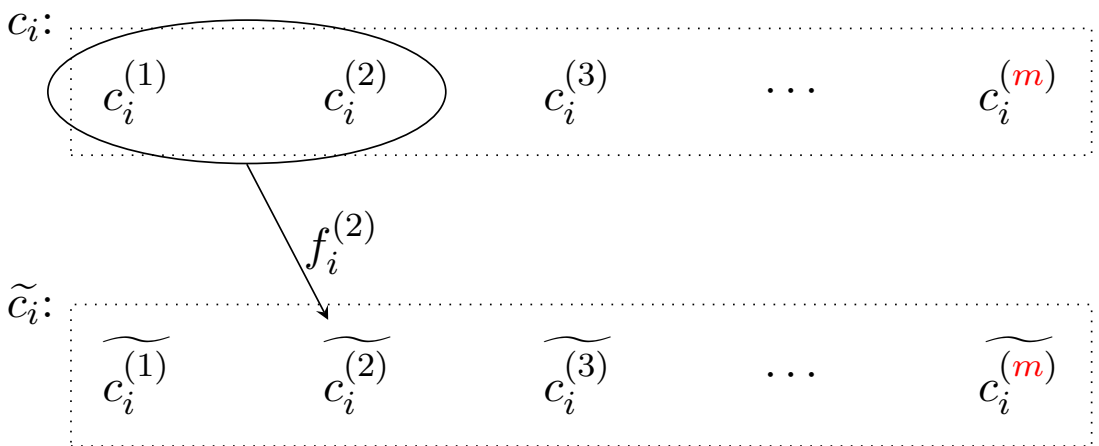
k-Lookahead Tampering Model



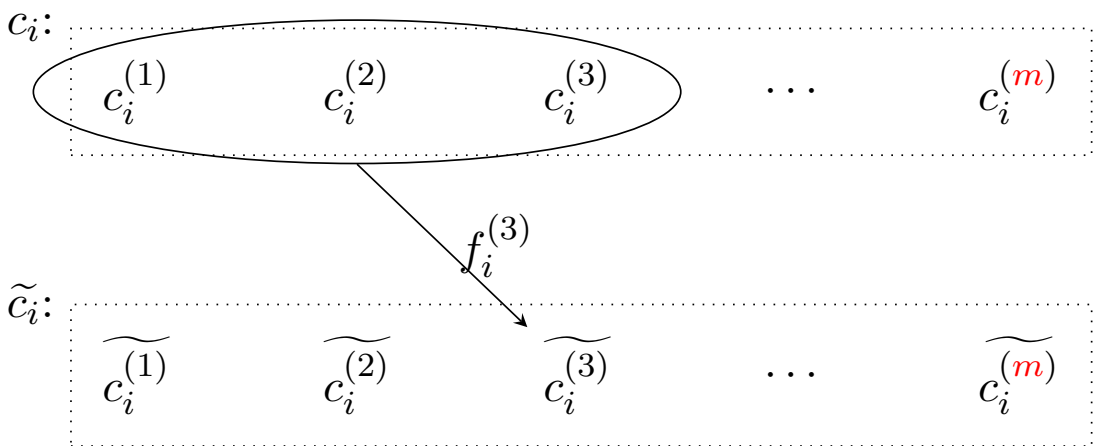
k-Lookahead Tampering Model



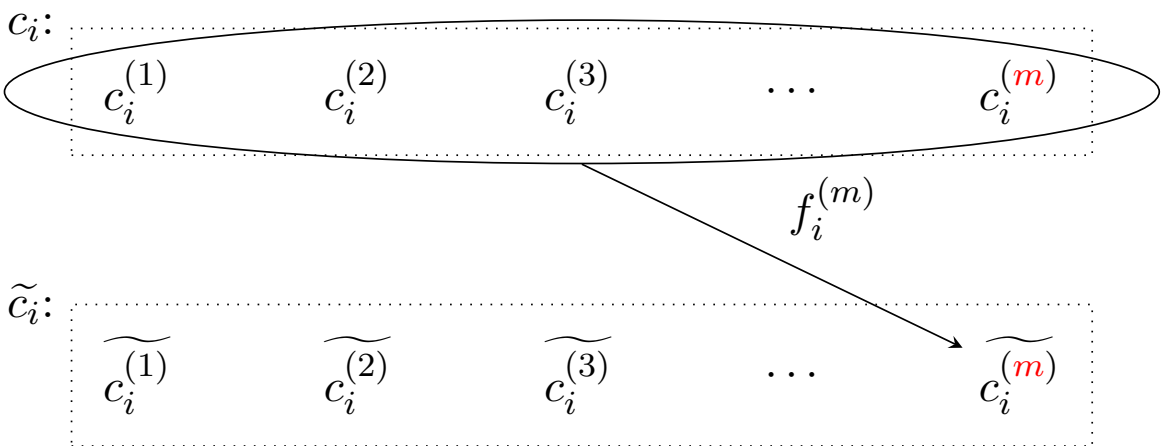
k-Lookahead Tampering Model



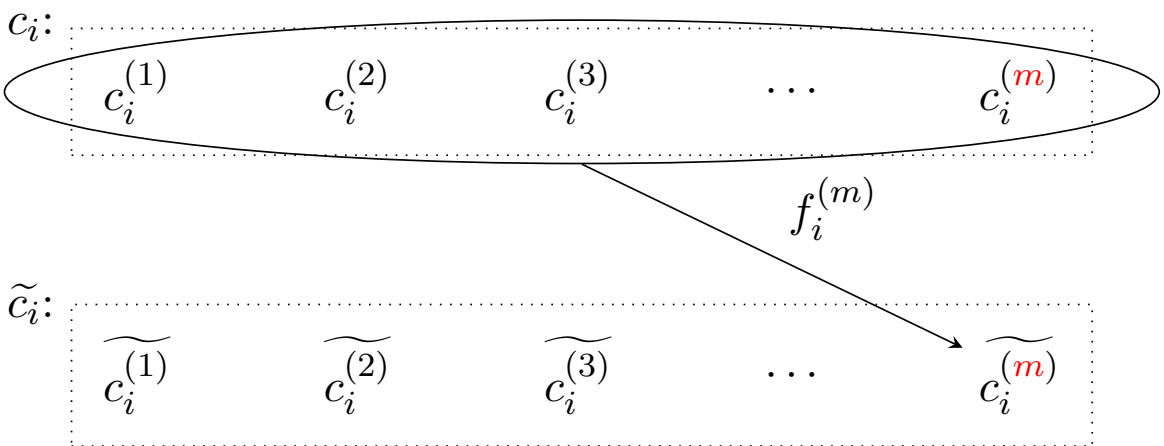
k-Lookahead Tampering Model



k-Lookahead Tampering Model



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

Our Contributions

Our Contributions

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

Upper Bound

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

Our Contributions

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

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 $\ell/n \leqslant 1 - 1/k$

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

Related Works: Split-State

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

k	Work	Best Rate
10	Chattopadhyay and Zuckerman [FOCS-14]	(small) const.
4	Kanukurthi, Obbattu and Sekar [TCC-17]	1/3
3	This work Kanukurthi, Obbattu and Sekar [EUROCRYPT-18]	1/3
2	Dziembowski, Kazana and Obremski [CRYPTO-13] Aggarwal, Dodis and Lovett [STOC-14] Aggarwal, Dodis, Kazana and Obremski [STOC-15] Li [STOC-17]	$1/\log n$

Upper Bound

Theorem 1

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

Upper Bound

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

Upper Bound

Theorem 1

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

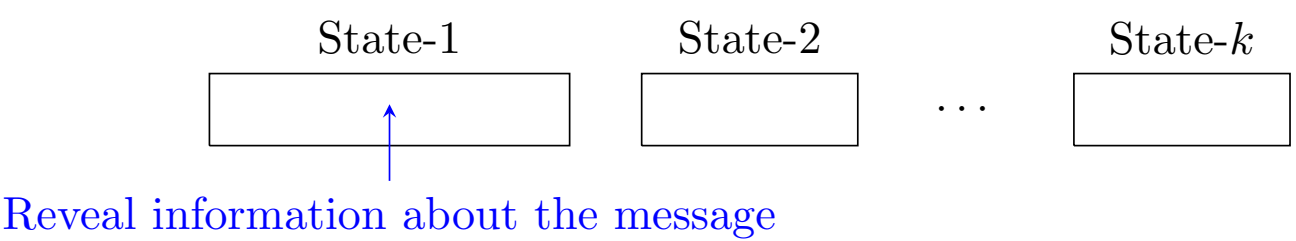
- We prove it for the weakest model, where tampering function tampers one-bit at a time
- Cheraghchi and Guruswami [ITCS-14] showed that if the rate is higher than $1 - 1/k$, there will exist a distinguisher $\mathcal{D}$ and two messages m_0, m_1 such that $\mathcal{D}$ can use the longest state to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

Upper Bound

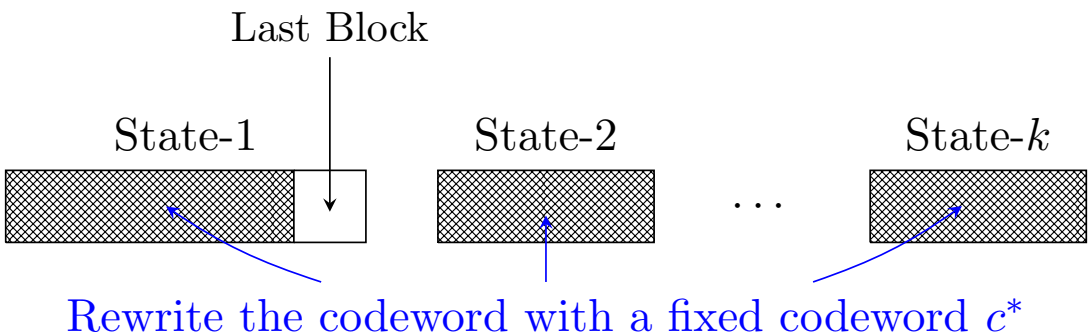
Theorem 1

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 $\mathcal{D}$ and two messages m_0, m_1 such that $\mathcal{D}$ can use the longest state to distinguish the encoding of m_0 and m_1

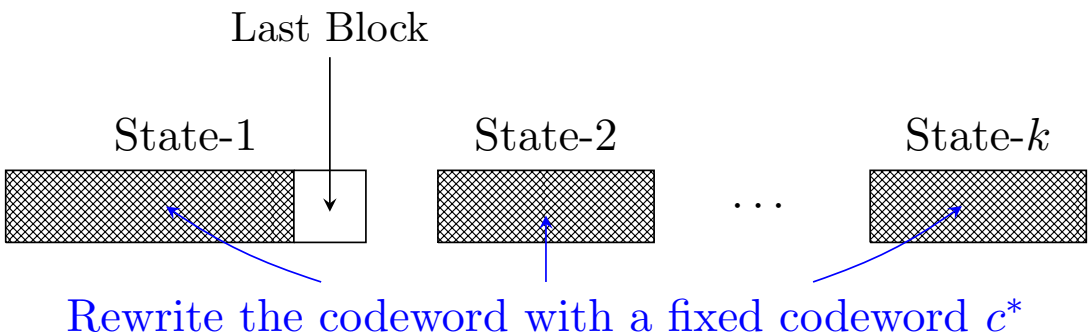


Upper Bound



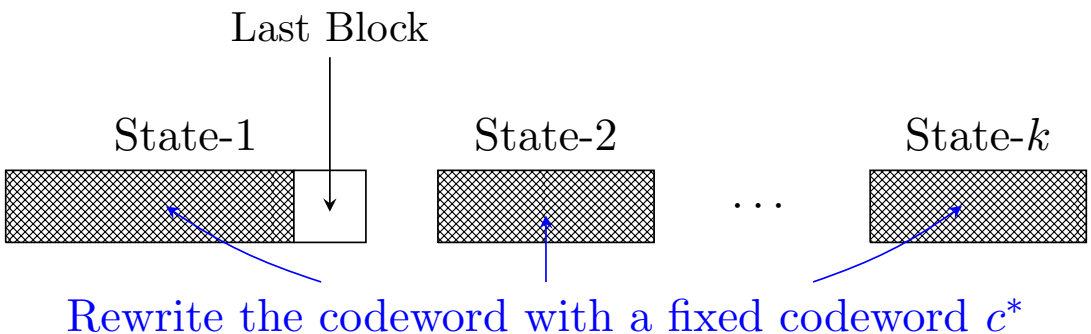
- Except for the last block of the longest state, we will rewrite the codeword to with a fixed codeword c^*, which encodes a fixed message $\mathrm{Dec}(c^*) = m^*$

Upper Bound



- 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 $\mathrm{Dec}(c^*) = m^*$
- 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

Upper Bound



- 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 $\mathrm{Dec}(c^*) = m^*$
- 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

$$\Pr\left[\mathrm{Tamper}_f^{m_0} = m^*\right] \quad \text{and} \quad \Pr\left[\mathrm{Tamper}_f^{m_1} = m^*\right]$$

will be significantly different

NMC against 2-lookahead

NMC against 2-lookahead

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

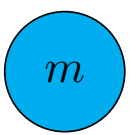
NMC against 2-lookahead

NMC against 2-lookahead

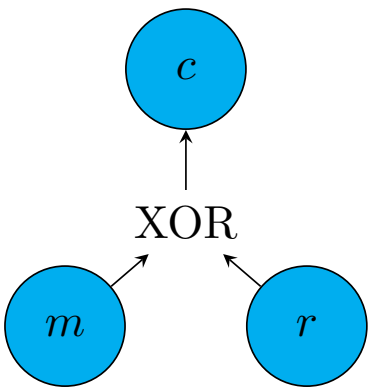
There exists an efficient non-malleable code, with negligible simulation error, against the 2-lookahead tampering with rate 1/3

- Kanukurthi, Obbattu and Sekar [TCC-17]'s construction of four-state NMC is the starting point of our construction

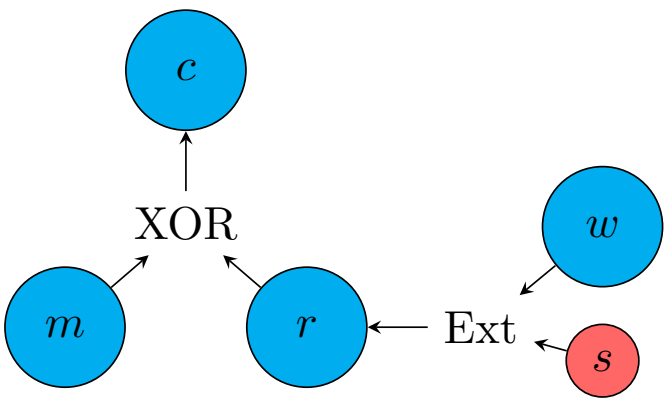
Construction of Kanukurthi, Obbattu and Sekar [TCC-17]



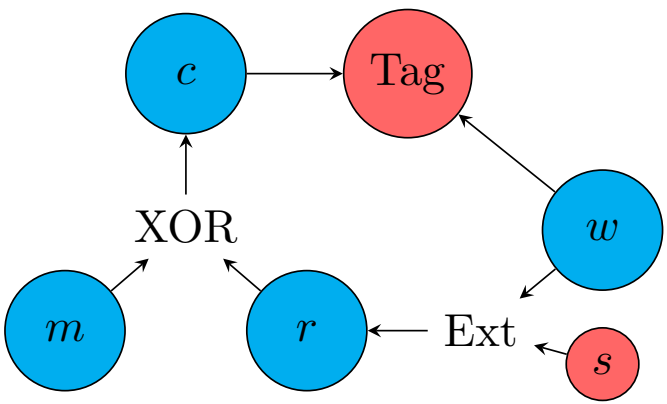
Construction of Kanukurthi, Obbattu and Sekar [TCC-17]



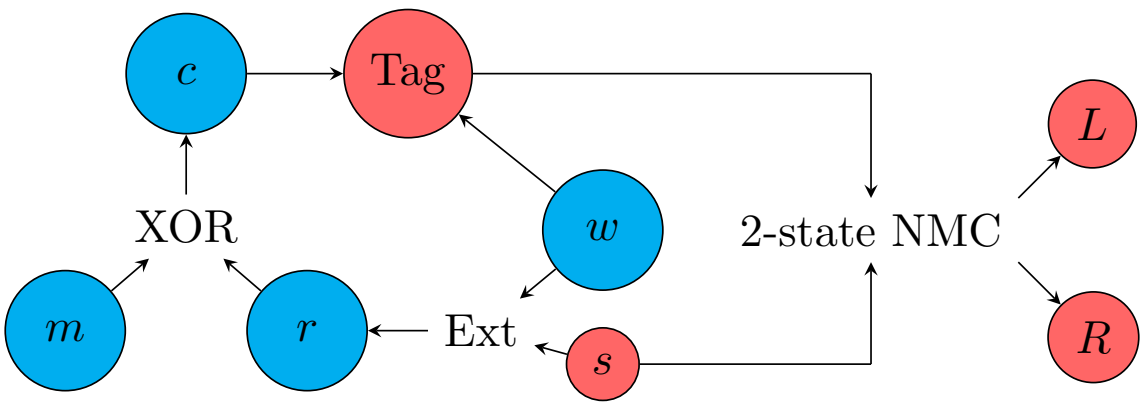
Construction of Kanukurthi, Obbattu and Sekar [TCC-17]



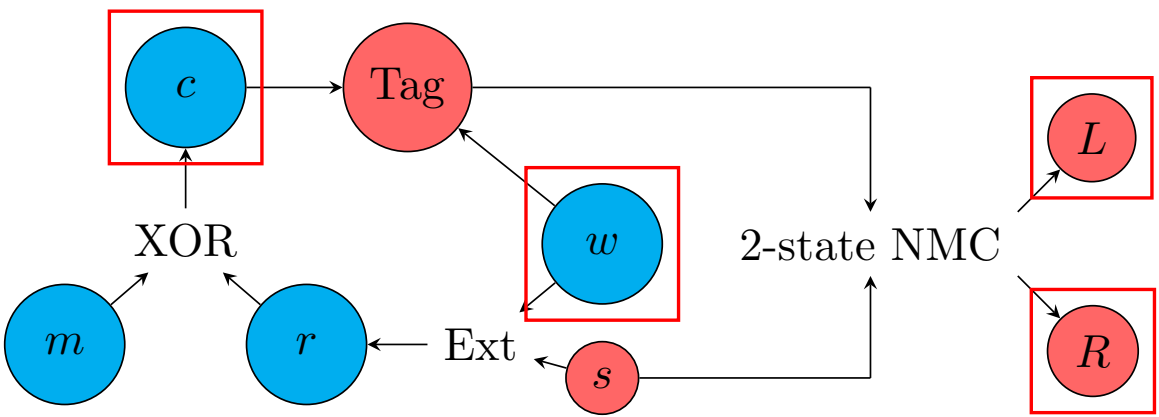
Construction of Kanukurthi, Obbattu and Sekar [TCC-17]



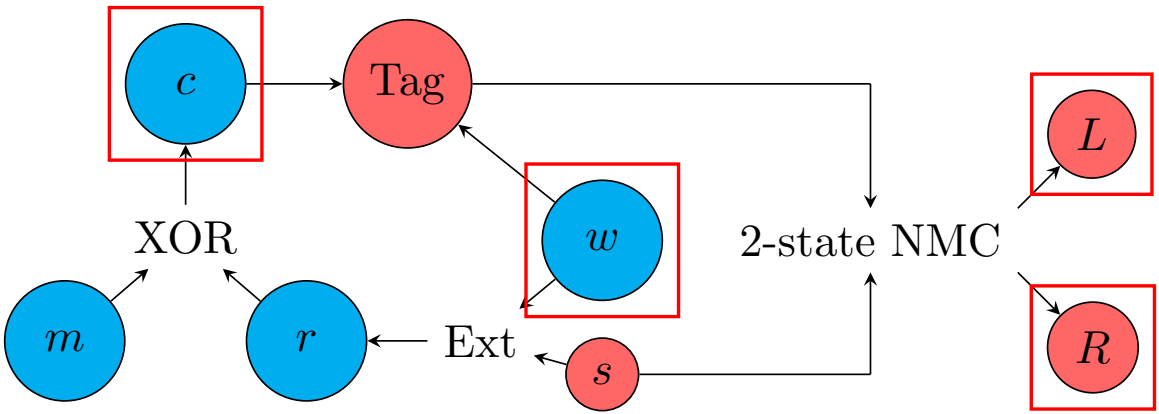
Construction of Kanukurthi, Obbattu and Sekar [TCC-17]



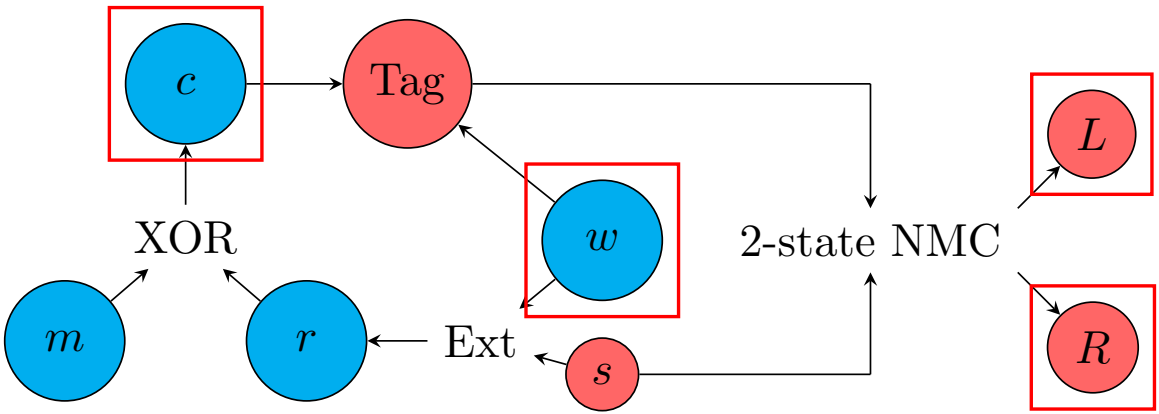
Construction of Kanukurthi, Obbattu and Sekar [TCC-17]



Our Modification

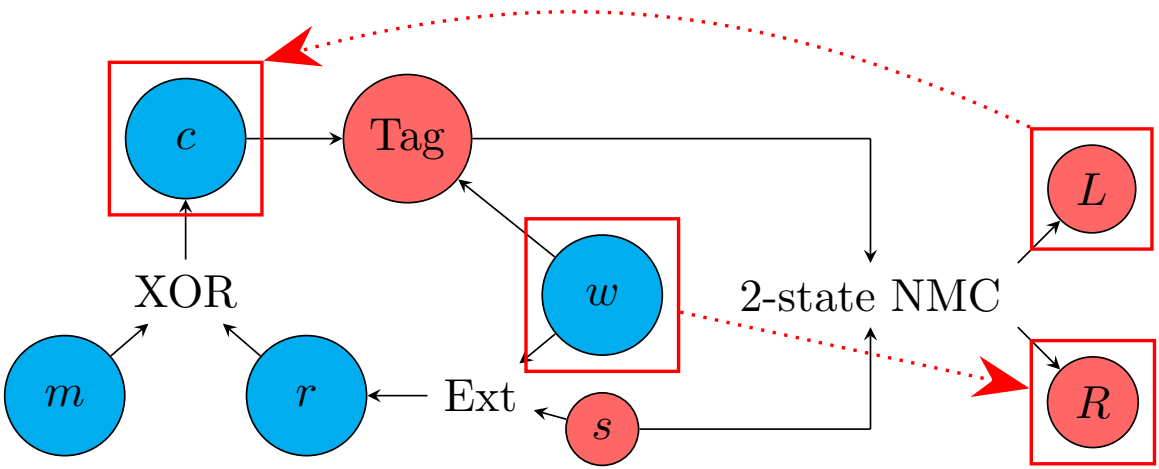


Our Modification



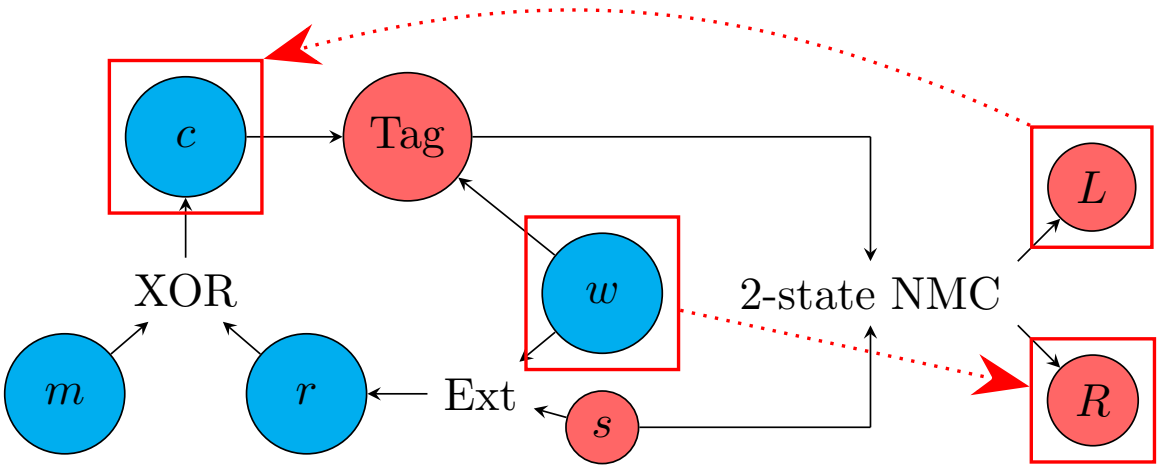
- Now, instead of storing c, w, L, R individually, we are going to merge some states and store it as $\underrightarrow{(w, R)}$ and $\underrightarrow{(L, c)}$

Our Modification



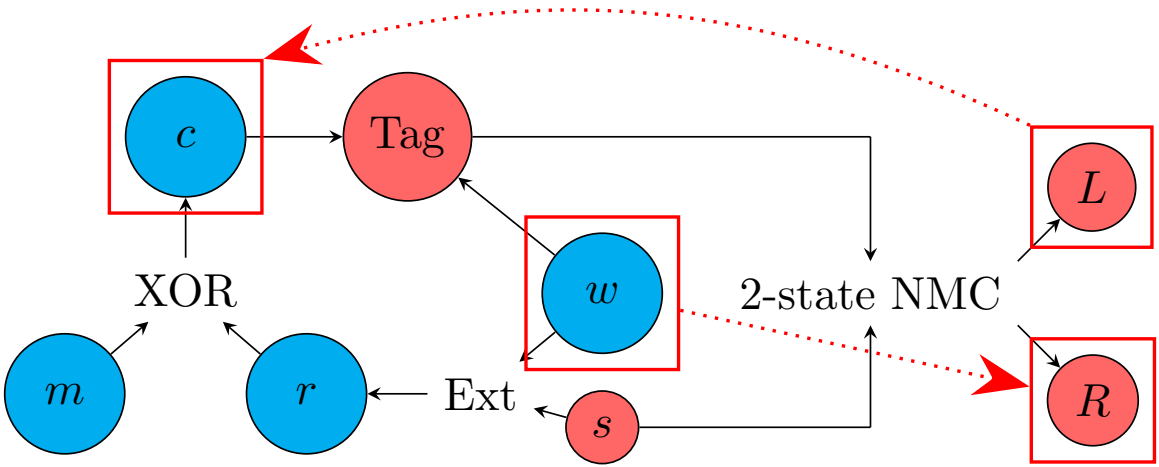
- Now, instead of storing c, w, L, R individually, we are going to merge some states and store it as $\underrightarrow{(w, R)}$ and $\underrightarrow{(L, c)}$
- This results in two issues:
  - The tampering on R now depends on w
  - The tampering on c now depends on L

Our Modification



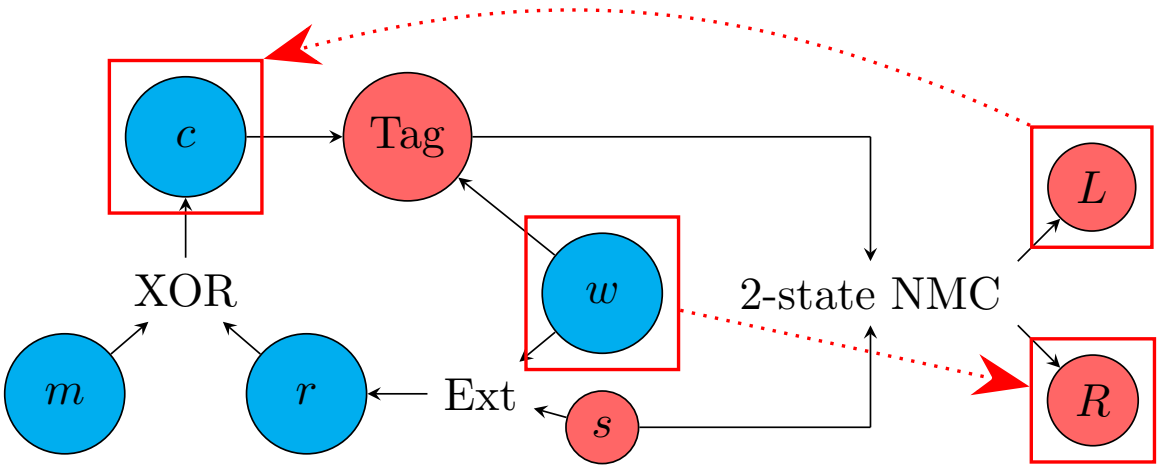
- Our codeword: $\underrightarrow{(w, R)}$ and $\underrightarrow{(L, c)}$

Our Modification



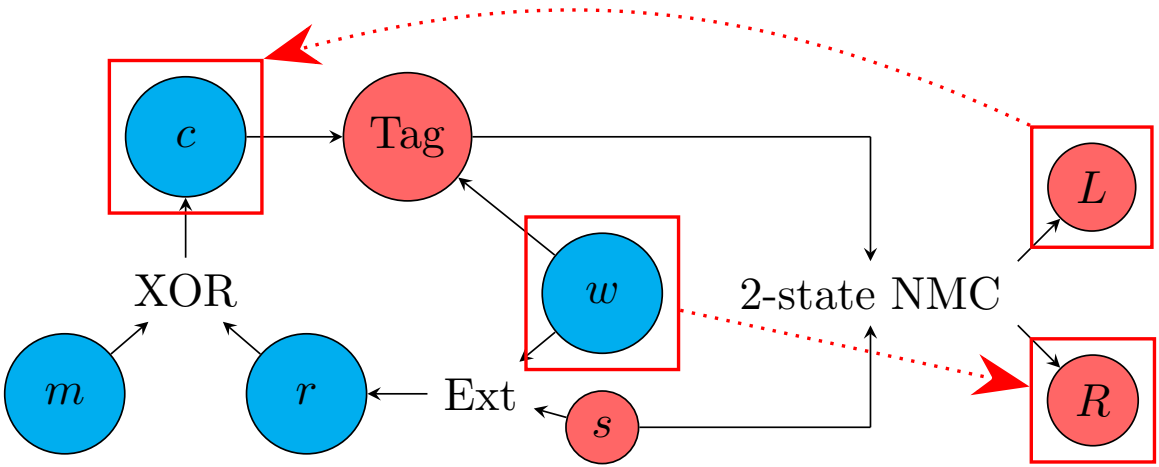
- Our codeword: $\overrightarrow{(w, R)}$ and $\overrightarrow{(L, c)}$
- The tampering on R influence only Tag and seed s.

Our Modification



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

Our Modification

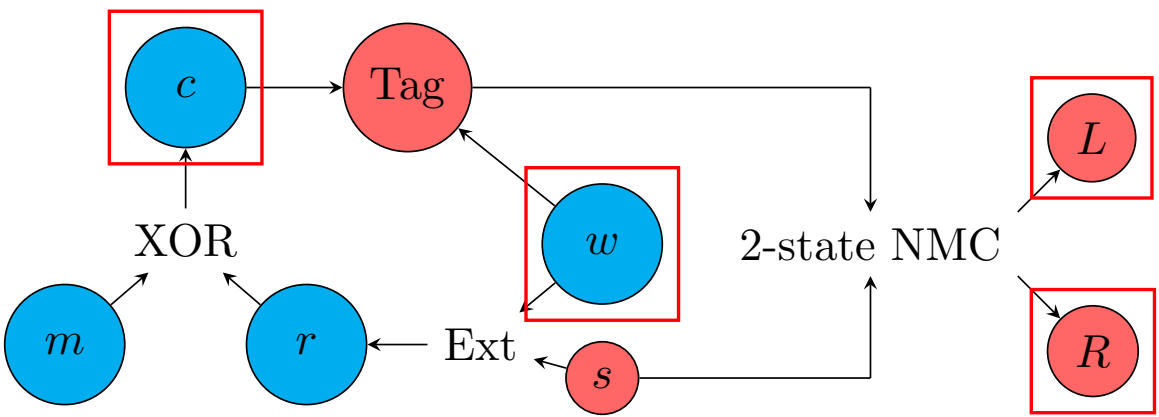


- Our codeword: $\overrightarrow{(w, R)}$ and $\overrightarrow{(L, c)}$
- The tampering on R influence only Tag and seed s. We view this as additional leakage on w
- We use an additional property of Aggarwal, Dodis and Lovett [STOC-14]'s construction, called *augmented non-malleability* (identified by 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

3-state NMC

NMC against 3-split-state

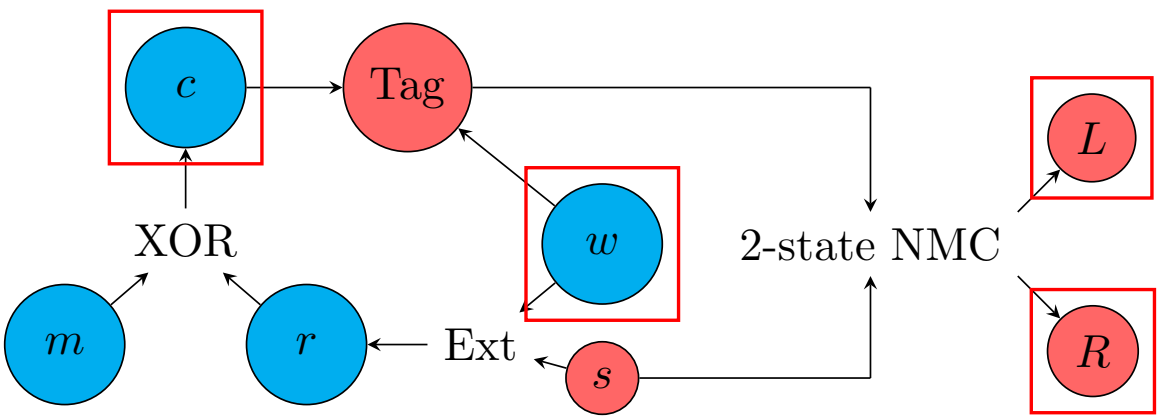
There exists an efficient non-malleable code, with negligible simulation error, against the 3-split-state tampering with rate 1/3



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

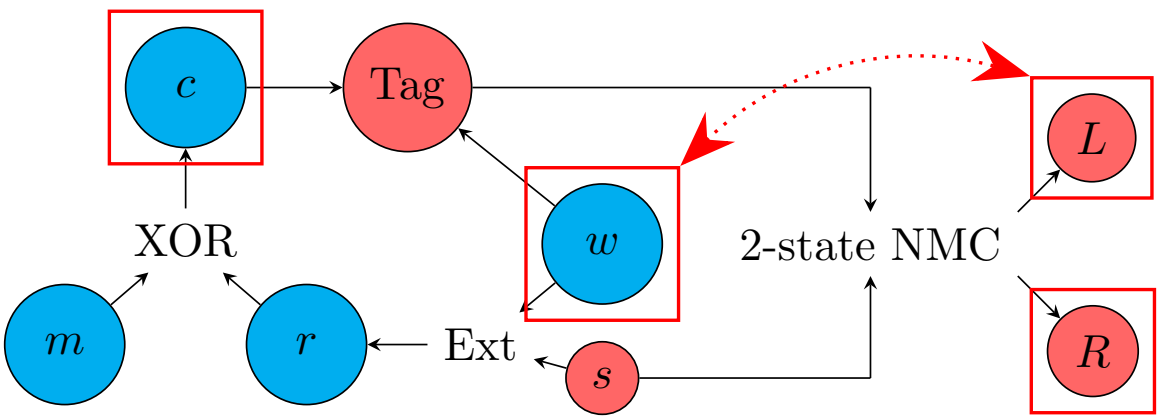


- Our 3-state codeword: c, $\underbrace{(L, w)}$ and R

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, $\underbrace{(L, w)}$ 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 $1 - 1/k$
- 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!