Meteor From Space

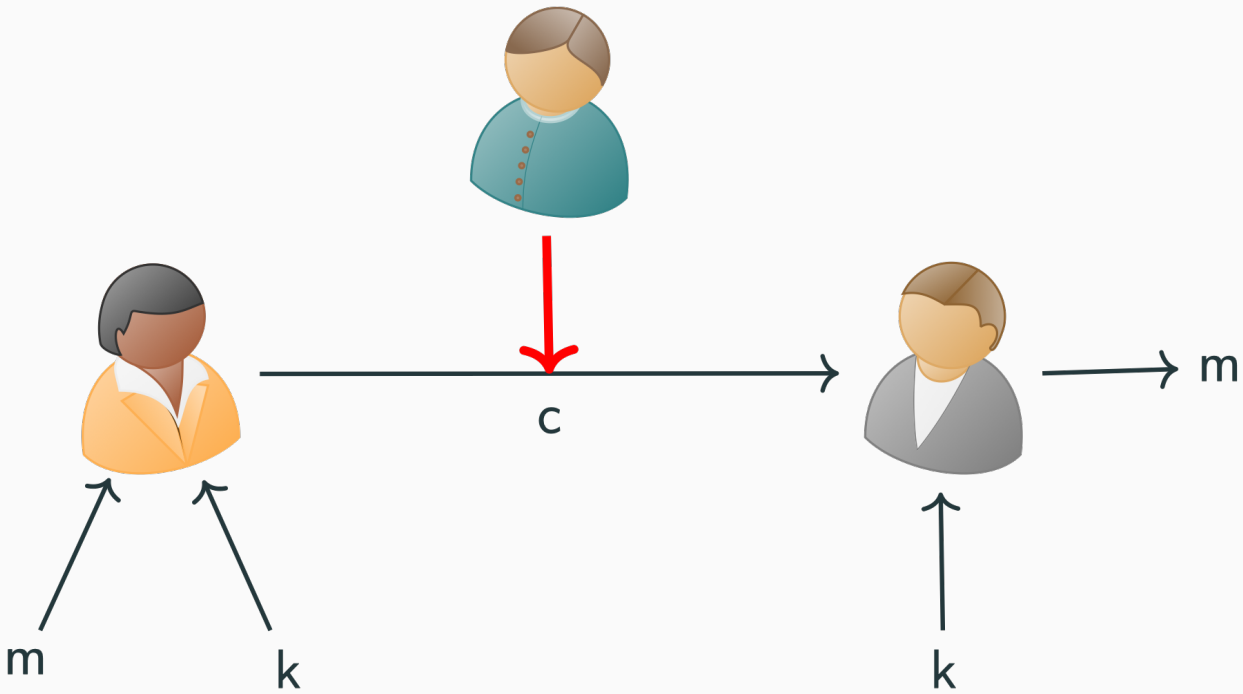
Cryptographically Secure Steganography For Realistic Distributions

Jeremy Boy

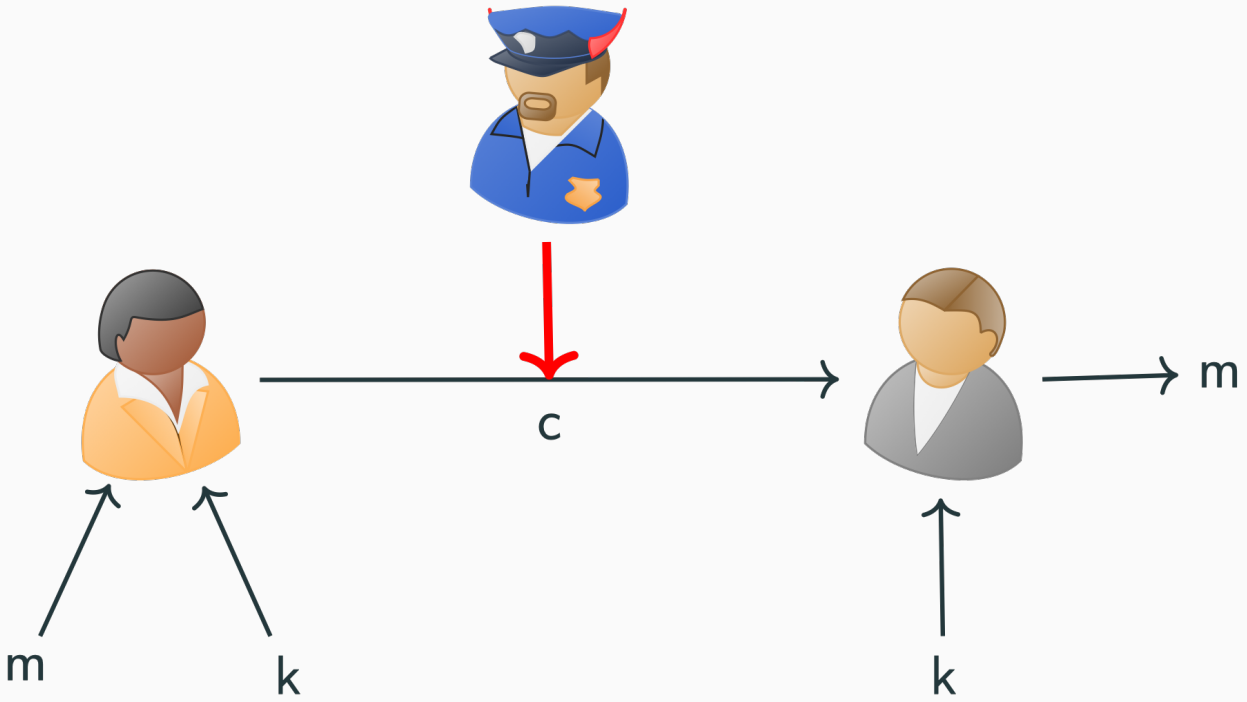
2022-05-16

Universitt zu Lbeck

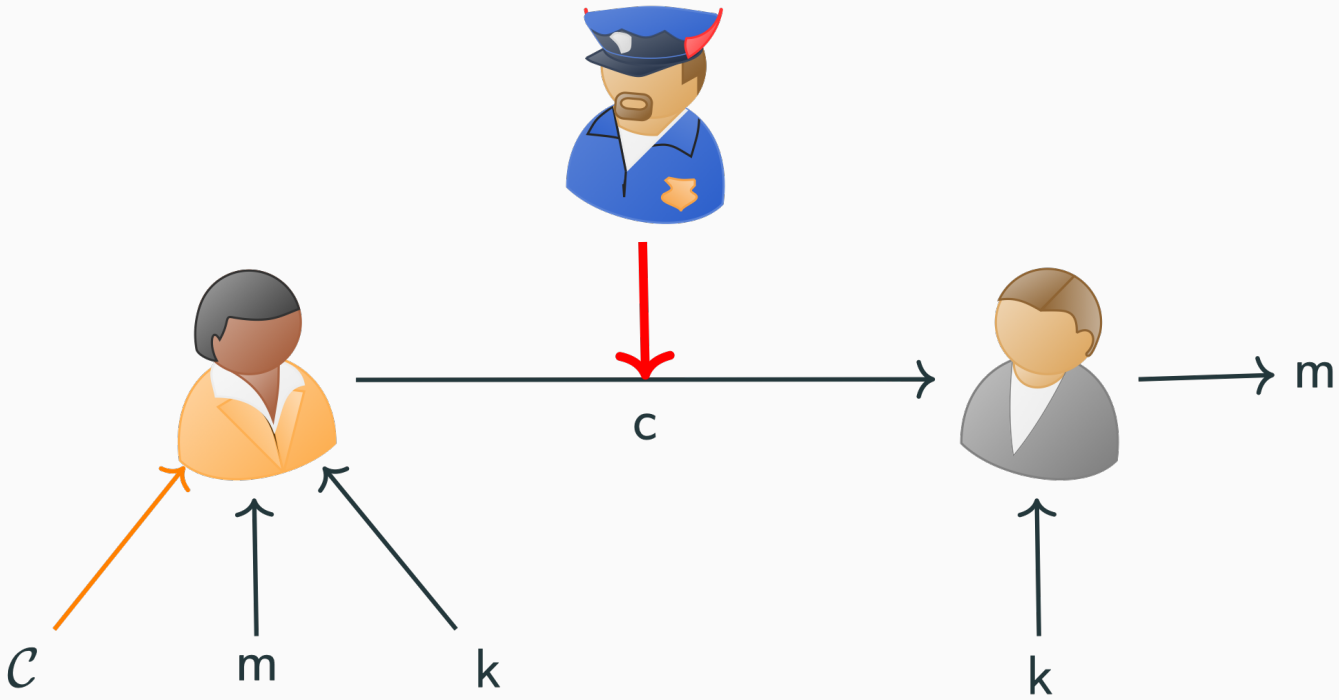
Introduction



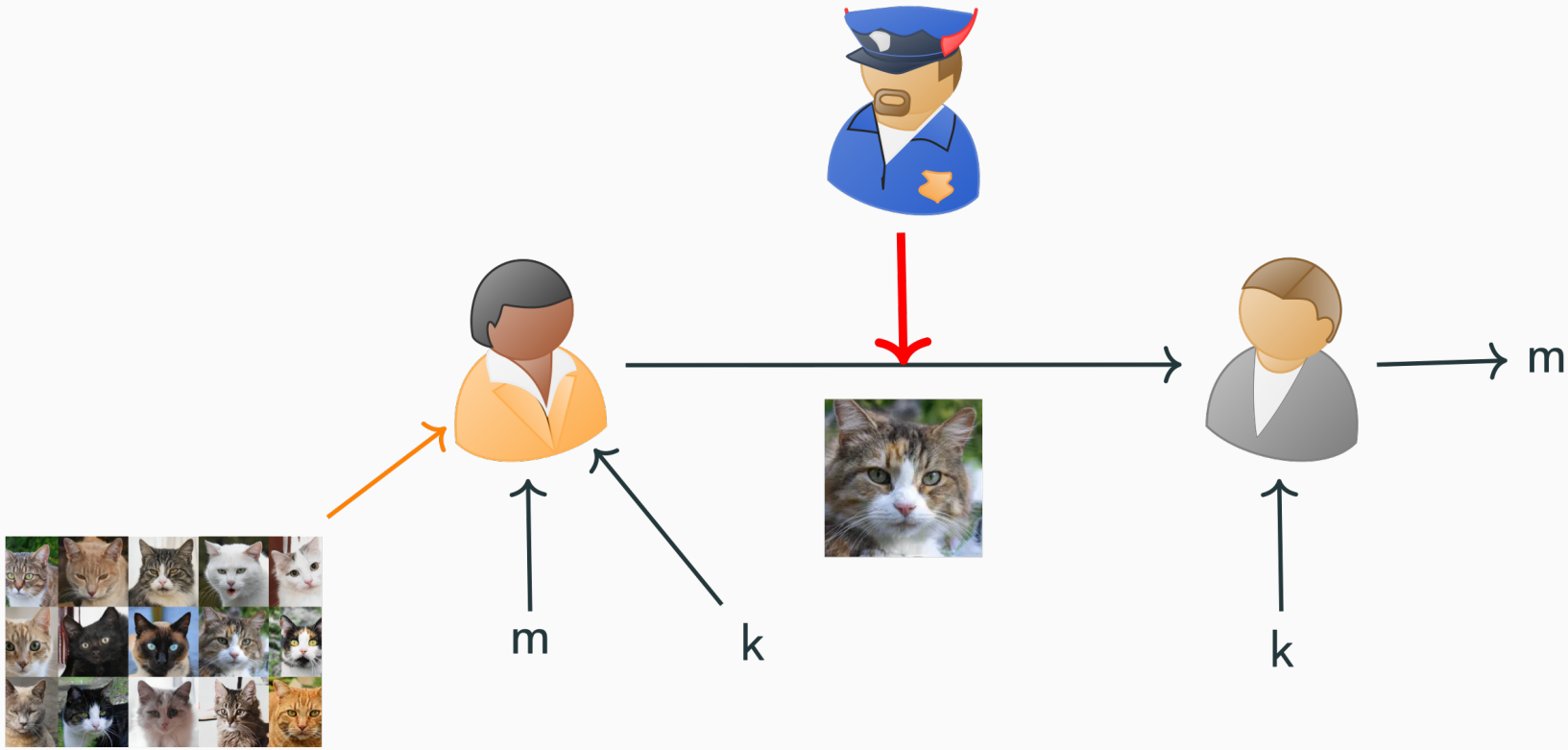
Introduction



Introduction



Introduction



Meteor From Space

1. Overview
2. Generative Neural Networks
3. Ranged Randomness Recoverable Sampling Scheme (RRRSS)
4. From RRRSS To Stegosystem
5. Security And Performance
6. Live Demo
7. Conclusion

Overview

Overview

- Classical Steganography not applicable to realistic distributions

Overview

- Classical Steganography not applicable to realistic distributions
- Use *generative approximation* of natural language to hide information

Overview

- Classical Steganography not applicable to realistic distributions
- Use *generative approximation* of natural language to hide information
- Encrypt hiddentext m using pre-shared key k to get ciphertext r

Overview

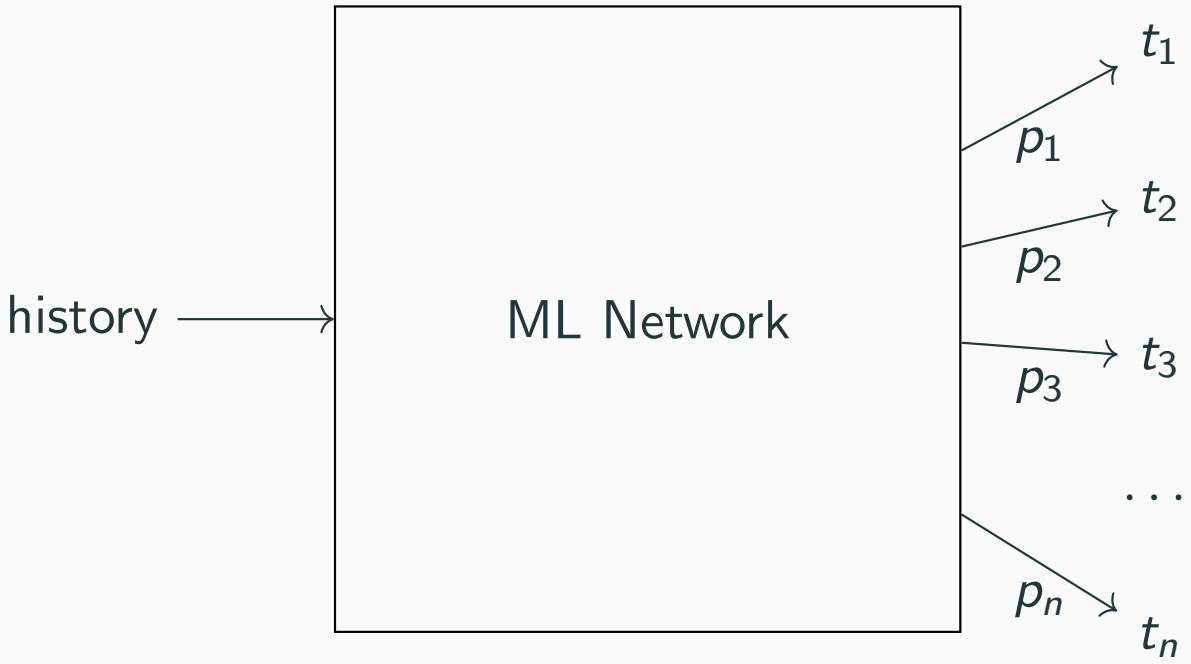
- Classical Steganography not applicable to realistic distributions
- Use *generative approximation* of natural language to hide information
- Encrypt hiddentext m using pre-shared key k to get ciphertext r
- Alice: manipulate sampling from $\mathcal{C}$ using r to create stegotext s

Overview

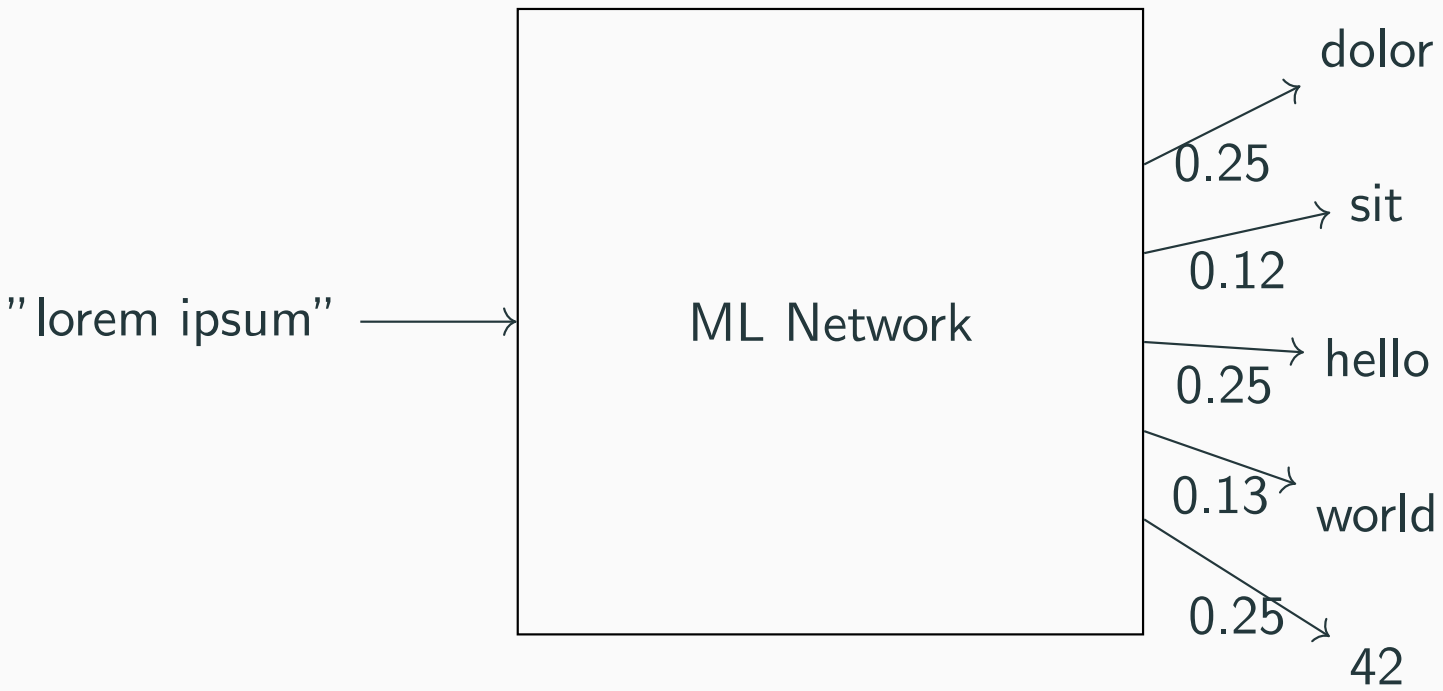
- Classical Steganography not applicable to realistic distributions
- Use *generative approximation* of natural language to hide information
- Encrypt hiddentext m using pre-shared key k to get ciphertext r
- Alice: manipulate sampling from $\mathcal{C}$ using r to create stegotext s
- Bob: recover r from s and decrypt m using k

Generative Neural Networks

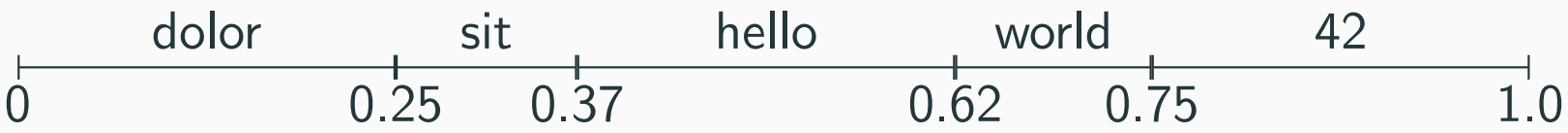
Generative Neural Networks



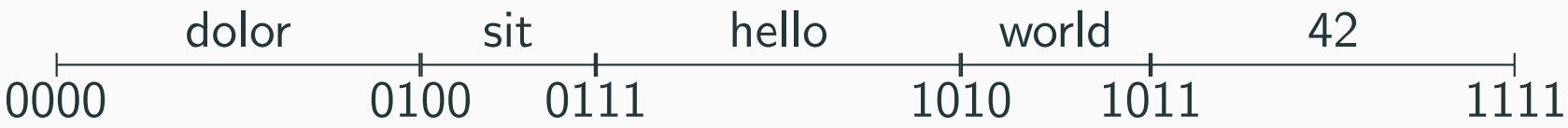
Generative Neural Networks



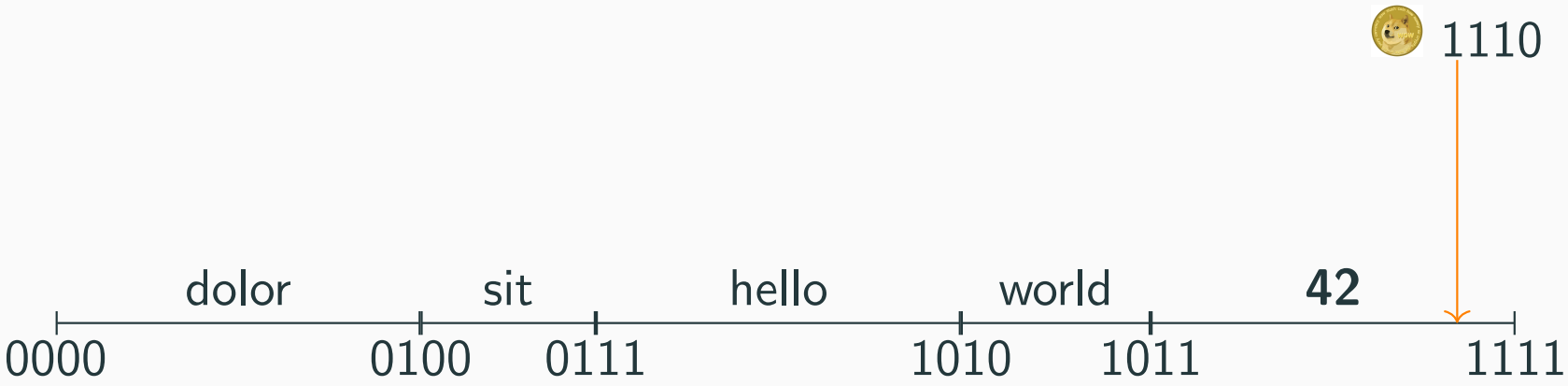
Generative Neural Networks



Generative Neural Networks



Generative Neural Networks



Ranged Randomness Recoverable Sampling Scheme (RRRSS)

Ranged Randomness Recoverable Sampling Scheme

- Let $\mathcal{D}$ be a probability distribution and $\beta \in \mathbb{N}$

$\mathcal{R} := \{r \in \{0,1\}^{\beta} \mid \mathsf{Sample}^{\beta}_{\mathcal{D}}(\mathcal{H}, r) = s\}$

Ranged Randomness Recoverable Sampling Scheme

- Let $\mathcal{D}$ be a probability distribution and $\beta \in \mathbb{N}$
- We call $(\mathcal{D}, \beta)$ a RRRSS on distribution $\mathcal{D}$ with precision β, if:

$\mathcal{R} := \{r \in \{0,1\}^{\beta} \mid \mathsf{Sample}^{\beta}_{\mathcal{D}}(\mathcal{H}, r) = s\}$

Ranged Randomness Recoverable Sampling Scheme

- Let $\mathcal{D}$ be a probability distribution and $\beta \in \mathbb{N}$
- We call $(\mathcal{D}, \beta)$ a RRRSS on distribution $\mathcal{D}$ with precision β, if:
  1. $\text{Sample}_{\mathcal{D}}^{\beta}(\mathcal{H}, r) \to s$. On history $\mathcal{H}$ and randomness $r \in \{0,1\}^{\beta}$ sample an output s from $\mathcal{D}$

$\mathcal{R} := \{r \in \{0,1\}^{\beta} \mid \text{Sample}_{\mathcal{D}}^{\beta}(\mathcal{H}, r) = s\}$

Ranged Randomness Recoverable Sampling Scheme

- Let $\mathcal{D}$ be a probability distribution and $\beta \in \mathbb{N}$
- We call $(\mathcal{D}, \beta)$ a RRRSS on distribution $\mathcal{D}$ with precision β, if:
  1. $\text{Sample}^{\beta}_{\mathcal{D}}(\mathcal{H}, r) \to s$. On history $\mathcal{H}$ and randomness $r \in \{0,1\}^{\beta}$ sample an output s from $\mathcal{D}$
  2. $\text{Recover}^{\beta}_{\mathcal{D}}(\mathcal{H}, s) \to \mathcal{R}$. On history $\mathcal{H}$ and sample s, output the set $\mathcal{R}$ of possible values for r

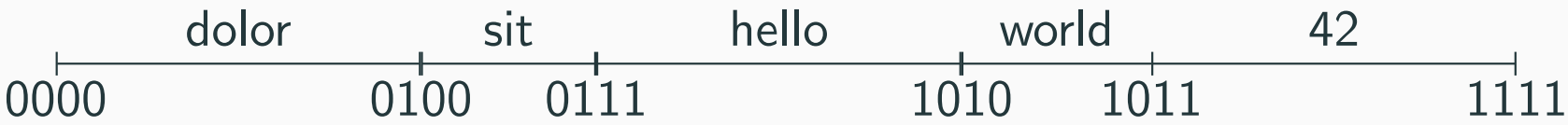
$\mathcal{R} := \{r \in \{0,1\}^{\beta} \mid \text{Sample}^{\beta}_{\mathcal{D}}(\mathcal{H}, r) = s\}$

Ranged Randomness Recoverable Sampling Scheme

$$\text{Sample}^{4}_{\mathcal{D}}(\text{"lorem ipsum"}, 1110) = \ ?$$

$\text{Sample}^{\beta}_{\mathcal{D}}(\mathcal{H}, r) \to s$. On history $\mathcal{H}$ and randomness $r \in \{0,1\}^{\beta}$ sample an output s from $\mathcal{D}$

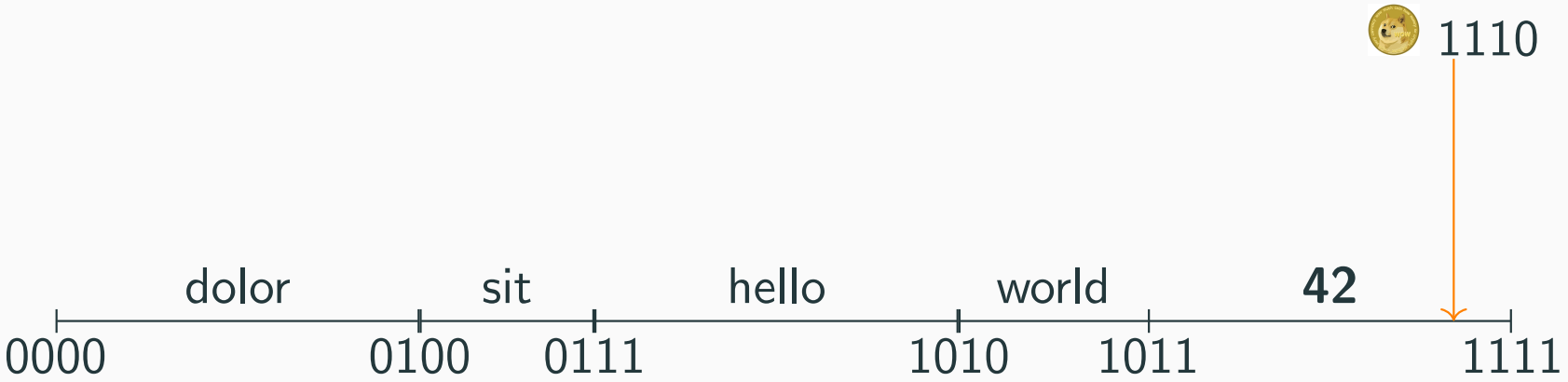
Ranged Randomness Recoverable Sampling Scheme



$$\text{Sample}^{4}_{\mathcal{D}}(\text{"lorem ipsum"}, 1110) = \ ?$$

$\text{Sample}^{\beta}_{\mathcal{D}}(\mathcal{H}, r) \to s$. On history $\mathcal{H}$ and randomness $r \in \{0,1\}^{\beta}$ sample an output s from $\mathcal{D}$

Ranged Randomness Recoverable Sampling Scheme



$$\text{Sample}^4_{\mathcal{D}}(\text{"lorem ipsum"}, 1110) = \text{"42"}$$

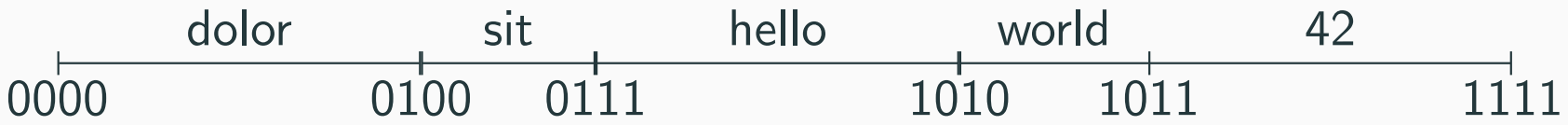
$\text{Sample}^{\beta}_{\mathcal{D}}(\mathcal{H}, r) \to s$. On history $\mathcal{H}$ and randomness $r \in \{0,1\}^{\beta}$ sample an output s from $\mathcal{D}$

Ranged Randomness Recoverable Sampling Scheme

$$\text{Recover}^{4}_{\mathcal{D}}(\text{"lorem ipsum"}, \text{"42"}) = \ ?$$

$\text{Recover}^{\beta}_{\mathcal{D}}(\mathcal{H}, s) \to \mathcal{R}$. On history $\mathcal{H}$ and sample s, output the set $\mathcal{R}$ of possible values for r

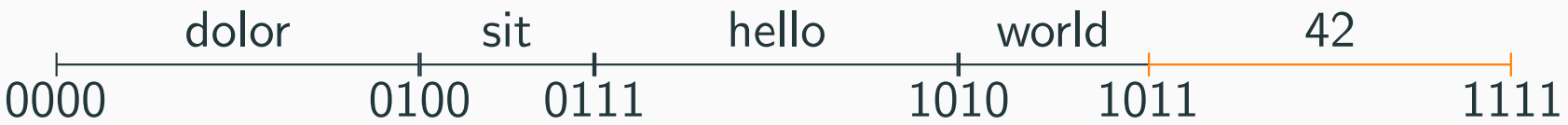
Ranged Randomness Recoverable Sampling Scheme



$$\text{Recover}^{4}_{\mathcal{D}}(\text{"lorem ipsum"}, \text{"42"}) = \ ?$$

$\text{Recover}^{\beta}_{\mathcal{D}}(\mathcal{H}, s) \to \mathcal{R}$. On history $\mathcal{H}$ and sample s, output the set $\mathcal{R}$ of possible values for r

Ranged Randomness Recoverable Sampling Scheme



$$\text{Recover}^4_{\mathcal{D}}(\text{"lorem ipsum"}, \text{"42"}) = \{1011,$$
$$1100,$$
$$1101,$$
$$1110,$$
$$1111\}$$

$\text{Recover}^{\beta}_{\mathcal{D}}(\mathcal{H}, s) \to \mathcal{R}$. On history $\mathcal{H}$ and sample s, output the set $\mathcal{R}$ of possible values for r

From RRRSS To Stegosystem

From RRRSS To Stegosystem

- Use m to determine randomness used in sampling

From RRRSS To Stegosystem

- Use m to determine randomness used in sampling
- Problem: m might be biased and can be observed by Warden

From RRRSS To Stegosystem

- Use m to determine randomness used in sampling
- Problem: m might be biased and can be observed by Warden
- Solution: at each iteration, encrypt the next β bits of m using PRG to get random value $r = m_\beta \oplus \text{PRG.next}(k)$

From RRRSS To Stegosystem

- Use m to determine randomness used in sampling
- Problem: m might be biased and can be observed by Warden
- Solution: at each iteration, encrypt the next β bits of m using PRG to get random value $r = m_\beta \oplus \text{PRG.next}(k)$
- Use r to sample $s = \text{Sample}_{\mathcal{D}}^{\beta}(\mathcal{H}, r)$

From RRRSS To Stegosystem

- Use m to determine randomness used in sampling
- Problem: m might be biased and can be observed by Warden
- Solution: at each iteration, encrypt the next β bits of m using PRG to get random value $r = m_\beta \oplus \text{PRG.next}(k)$
- Use r to sample $s = \text{Sample}_{\mathcal{D}}^{\beta}(\mathcal{H}, r)$
- We have hidden a prefix of r (and therefore m) in s

From RRRSS To Stegosystem

- Use m to determine randomness used in sampling
- Problem: m might be biased and can be observed by Warden
- Solution: at each iteration, encrypt the next β bits of m using PRG to get random value $r = m_\beta \oplus \text{PRG.next}(k)$
- Use r to sample $s = \text{Sample}_{\mathcal{D}}^{\beta}(\mathcal{H}, r)$
- We have hidden a prefix of r (and therefore m) in s
- Use $\text{Recover}_{\mathcal{D}}^{\beta}(\mathcal{H}, s) = R$, calculate prefix of R

From RRRSS To Stegosystem

- Use m to determine randomness used in sampling
- Problem: m might be biased and can be observed by Warden
- Solution: at each iteration, encrypt the next β bits of m using PRG to get random value $r = m_\beta \oplus \text{PRG.next}(k)$
- Use r to sample $s = \text{Sample}_{\mathcal{D}}^{\beta}(\mathcal{H}, r)$
- We have hidden a prefix of r (and therefore m) in s
- Use $\text{Recover}_{\mathcal{D}}^{\beta}(\mathcal{H}, s) = R$, calculate prefix of R
- Decrypt m from recovered bits using k

From RRRSS To Stegosystem

Algorithm 5: $\text{Encode}^{\beta}_{\mathcal{M}}$

Input: Key k_{prg}, Plaintext Message m, History $\mathcal{H}$
Output: Stegotext Message c

$c \leftarrow \varepsilon, n \leftarrow 0$
while $n < |m|$ **do**
  $mask \leftarrow \text{PRG.Next}(k_{prg})$
  $r \leftarrow m[n : n + \beta] \oplus mask$
  $c_i \leftarrow \text{Sample}^{\beta}_{\mathcal{M}}(\mathcal{H}, r)$
  $\mathcal{R} \leftarrow \text{Recover}^{\beta}_{\mathcal{M}}(\mathcal{H}, c_i)$
  $n_i \leftarrow \text{LenPrefix}^{\beta}(\mathcal{R})$
  $c \leftarrow c \| c_i, n \leftarrow n + n_i, \mathcal{H} \leftarrow \mathcal{H} \| c_i$
Output c

From RRRSS To Stegosystem

Algorithm 6: $\text{Decode}^{\beta}_{\mathcal{M}}$

Input: Key k_{prg}, Stegotext Message c, History $\mathcal{H}$
Output: Plaintext Message m

$x \leftarrow \varepsilon$
Parse c as $\{c_0, c_1, \ldots, c_{|c|-1}\}$
for $i \in \{0, 1, \ldots, |c| - 1\}$ **do**
    $\mathcal{R} \leftarrow \text{Recover}^{\beta}_{\mathcal{M}}(\mathcal{H}, c_i)$
    $x_i \leftarrow \text{Prefix}^{\beta}(\mathcal{R})$
    $mask \leftarrow \text{PRG.Next}(k_{prg})$
    $x \leftarrow x \| (x_i \oplus mask[0 : |x_i|])$
    $\mathcal{H} \leftarrow \mathcal{H} \| c_i$
Output x

Security And Performance

Security And Performance

- Meteor proven secure (by reduction from PRG)

Security And Performance

- Meteor proven secure (by reduction from PRG)
- Ca. 4 bits of hiddentext data per stegotext word

Security And Performance

- Meteor proven secure (by reduction from PRG)
- Ca. 4 bits of hiddentext data per stegotext word
- Encoding/Decoding a 160 byte message takes:

Security And Performance

- Meteor proven secure (by reduction from PRG)
- Ca. 4 bits of hiddentext data per stegotext word
- Encoding/Decoding a 160 byte message takes:
  - 7 seconds on GPU (NVIDIA TITAN X)

Security And Performance

- Meteor proven secure (by reduction from PRG)
- Ca. 4 bits of hiddentext data per stegotext word
- Encoding/Decoding a 160 byte message takes:
  - 7 seconds on GPU (NVIDIA TITAN X)
  - 41 seconds on CPU (Intel Core i7-6700)

Security And Performance

- Meteor proven secure (by reduction from PRG)
- Ca. 4 bits of hiddentext data per stegotext word
- Encoding/Decoding a 160 byte message takes:
  - 7 seconds on GPU (NVIDIA TITAN X)
  - 41 seconds on CPU (Intel Core i7-6700)
  - 473 seconds on Mobile (iPhone X)

Live Demo

Live Demo

Meteor Live Demo

Conclusion

Conclusion

- We can use generative neural networks to build cryptographically secure stegosystems

Conclusion

- We can use generative neural networks to build cryptographically secure stegosystems
- Generated texts are hardly distinguishable from actual human texts (and will even become better)

Conclusion

- We can use generative neural networks to build cryptographically secure stegosystems
- Generated texts are hardly distinguishable from actual human texts (and will even become better)
- We can achieve competitive performance using symmetric encryption