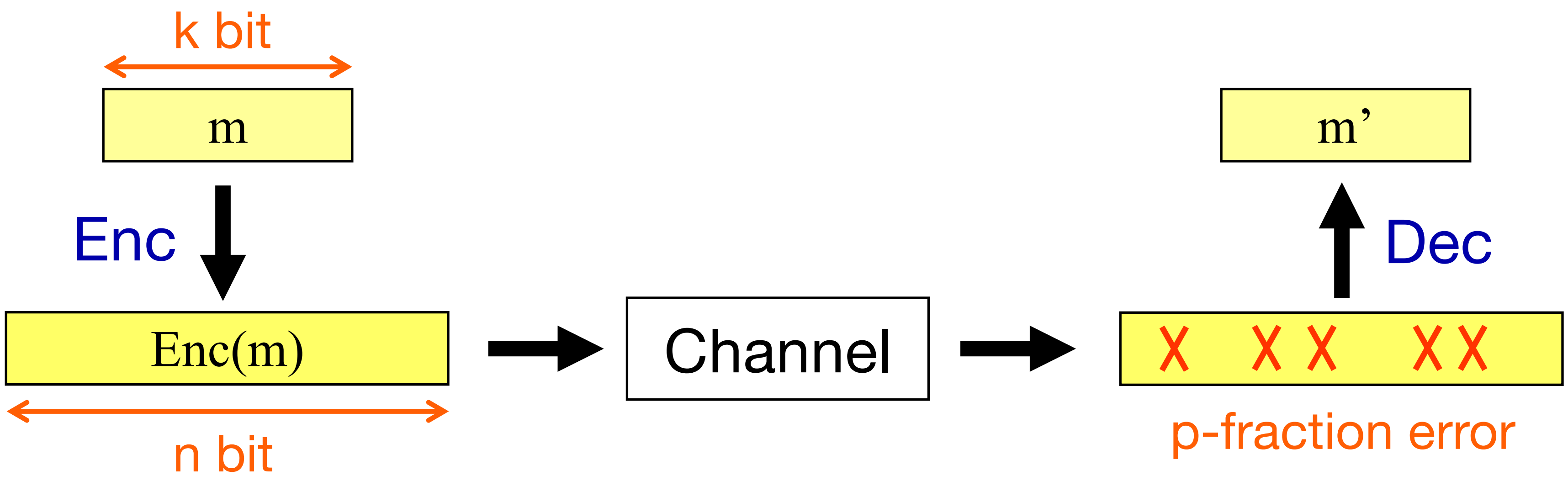
Error-Correcting Codes against Chosen-Codeword Attacks

Kenji Yasunaga
Kanazawa University, Japan

ICITS 2016 @ Tacoma, Washington, USA

Error-Correcting Codes



- **Goal:** Construct a code (Enc, Dec) that
  - corrects many errors (high error-rate p)
  - sends messages efficiently (high rate R = k/n)

→ Limitations depend on "Channel Models"

Channel Models

- Binary Symmetric Channel (BSC)
  - Each bit is independently flipped w.p. $p \in [0,1/2)$
  - Rate $R = 1 - h(p) - \varepsilon$ is achievable and optimal
  - $\exists$ efficient decoders [Forney'66][Arikan'09]

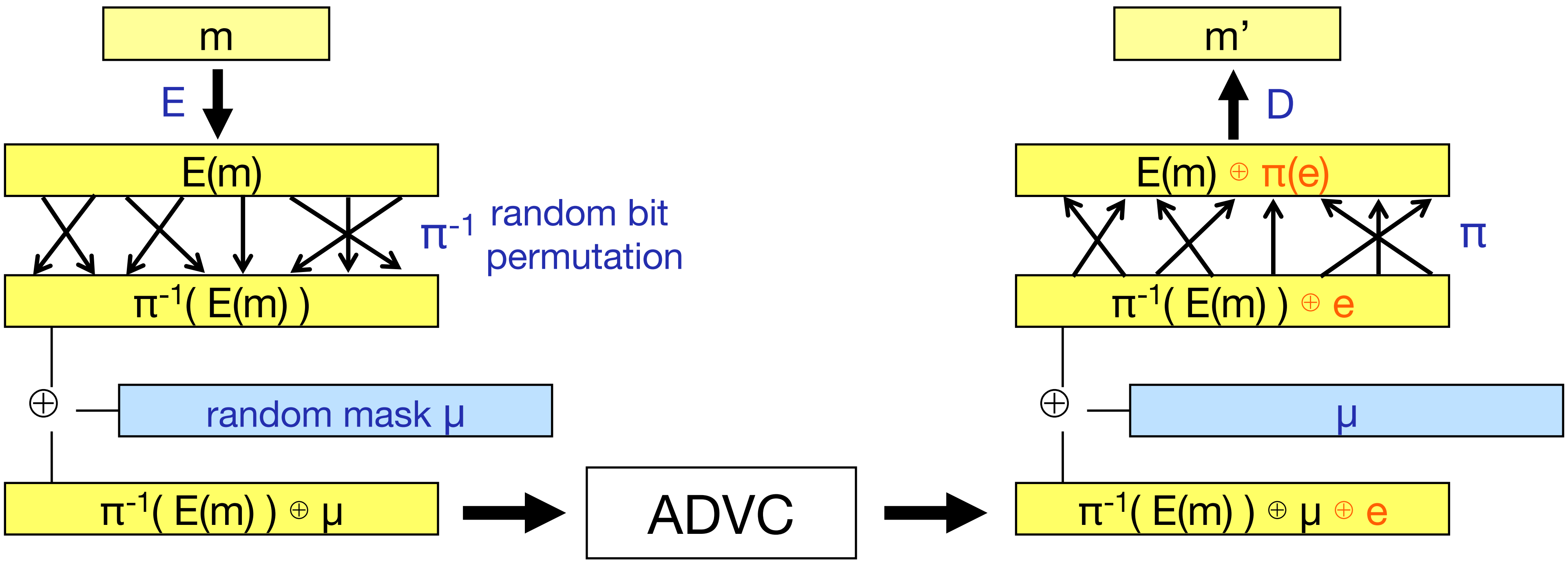
$h(p) = -p \log(p) - (1-p) \log(1-p)$

- Adversarial Channel (ADVC)
  - Worst-case error e is introduced s.t. $w_H(e) \leq pn$
  - Random codes achieve rate $R = 1 - h(2p)$
    - Optimality/efficient-decoders are open problems

Lipton's Reduction [Lipton'94]

Code for BSC is sufficient for ADVC in Secret-Key Setting

- Lipton's scheme using BSC code (E, D), SK = (π, μ)



- Worst-case error "e" → random error "π(e)"
- μ is used to conceal π from Channel

On Lipton's Scheme

- Achieves only one-time security
  - Sending t messages needs t secret keys
  - Similar to One-Time Pad Encryption

- Modern cryptography requires schemes that are
  - many-time secure with single secret-key
  - secure in more powerful attack scenarios

  ⇒ Chosen-Ciphertext Attack (CCA) security

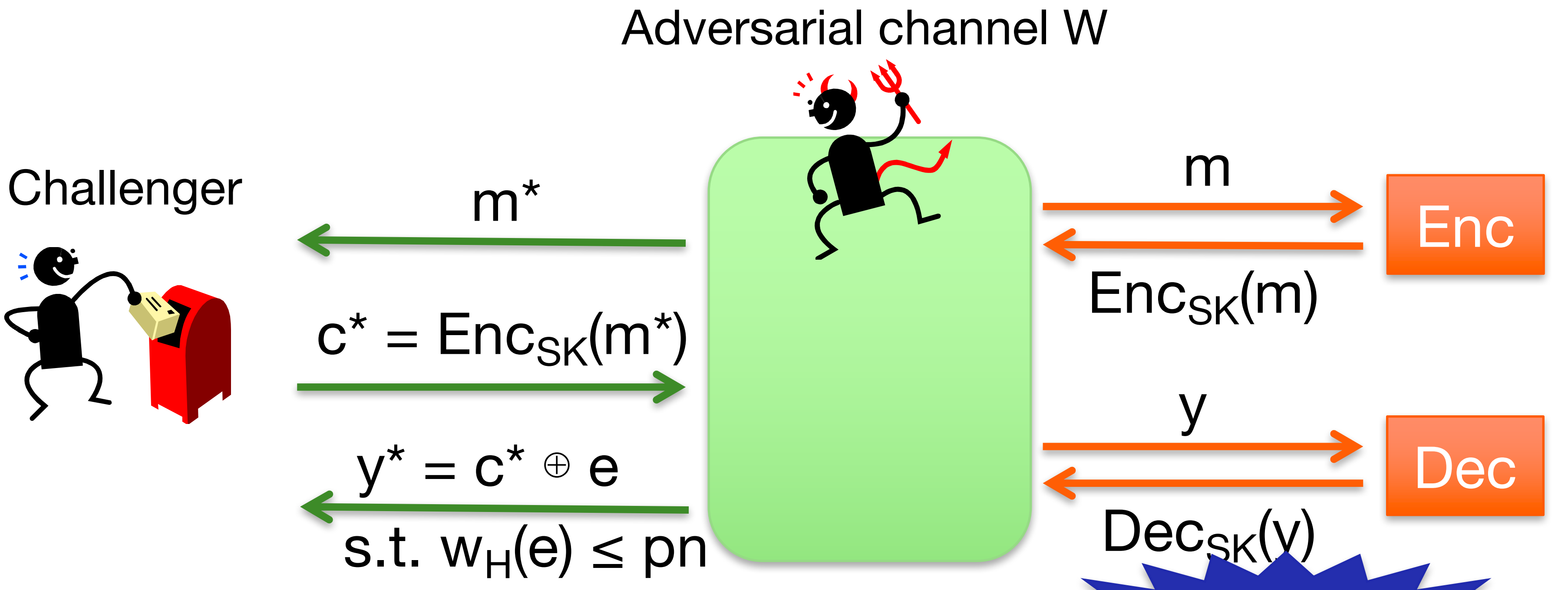
This Work

- Introduce Chosen-Codeword Attack (CCA) security for error-correcting codes
  - Enc/Dec oracles are available to channels

- Construct optimal-rate CCA-secure code
  - Based on Guruswami-Smith code [GS'10] for computationally bounded channels
  - Assuming OWF
  - Secret-key setting

Chosen-Codeword Attack (CCA) Security

- In error-correcting game, Adversarial channel can adaptively access to Enc/Dec oracles



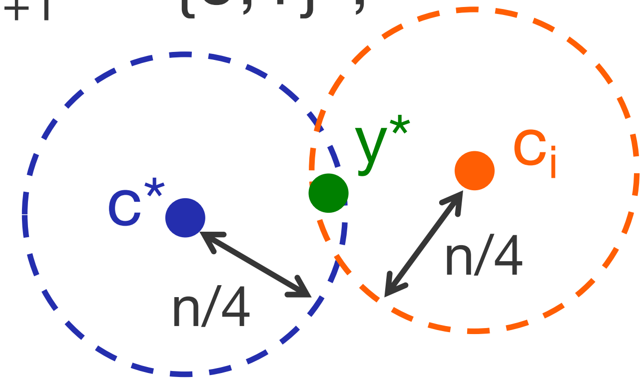
CCA secure $\Leftrightarrow$ Pr[W wins] ≈ 0

$\Leftrightarrow$ Pr[$\mathrm{Dec}_{SK}(y^*) \neq m^*$] ≈ 0

Impossible to achieve

Impossibility

- In CCA game, W can obtain polynomially-many valid codewords $c_1, c_2, \dots$
- [Plotkin bound] $\forall$ strings $x_1, \dots, x_{2n+1} \in \{0,1\}^n$, $\exists$ i, j s.t. $\mathrm{dist}_H(x_i, x_j) < n/2$



- Given valid $c^*, c_1, c_2, \dots,$ W can find c_i (w.p. $1/n^2$) s.t. $\mathrm{dist}_H(c^*, c_i) < n/2$

→ W can find y^* s.t. $\mathrm{dist}_H(c^*, y^*) \le n/4$, $\mathrm{dist}_H(y^*, c_i) \le n/4$

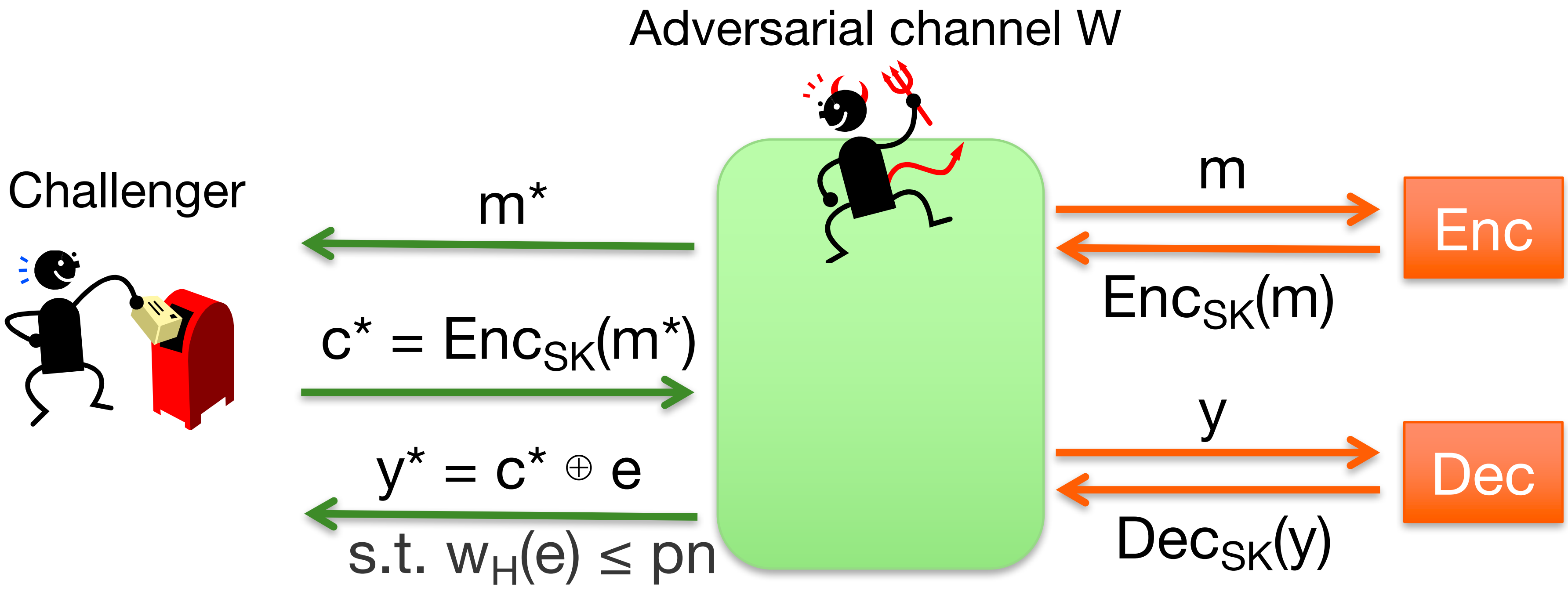
→ W can win by submitting y^* if $p \ge 1/4$

Unique decoding is impossible for $p \ge 1/4$

⇒ List decoding

Chosen-Codeword Attack (CCA) Security

- Unique decoding → List decoding



CCA secure $\Leftrightarrow$ Pr[W wins] $\approx$ 0

$\Leftrightarrow$ Pr[$m^* \notin L \mid L \leftarrow \mathrm{Dec}_{SK}(\, y^* \,)$] $\approx$ 0

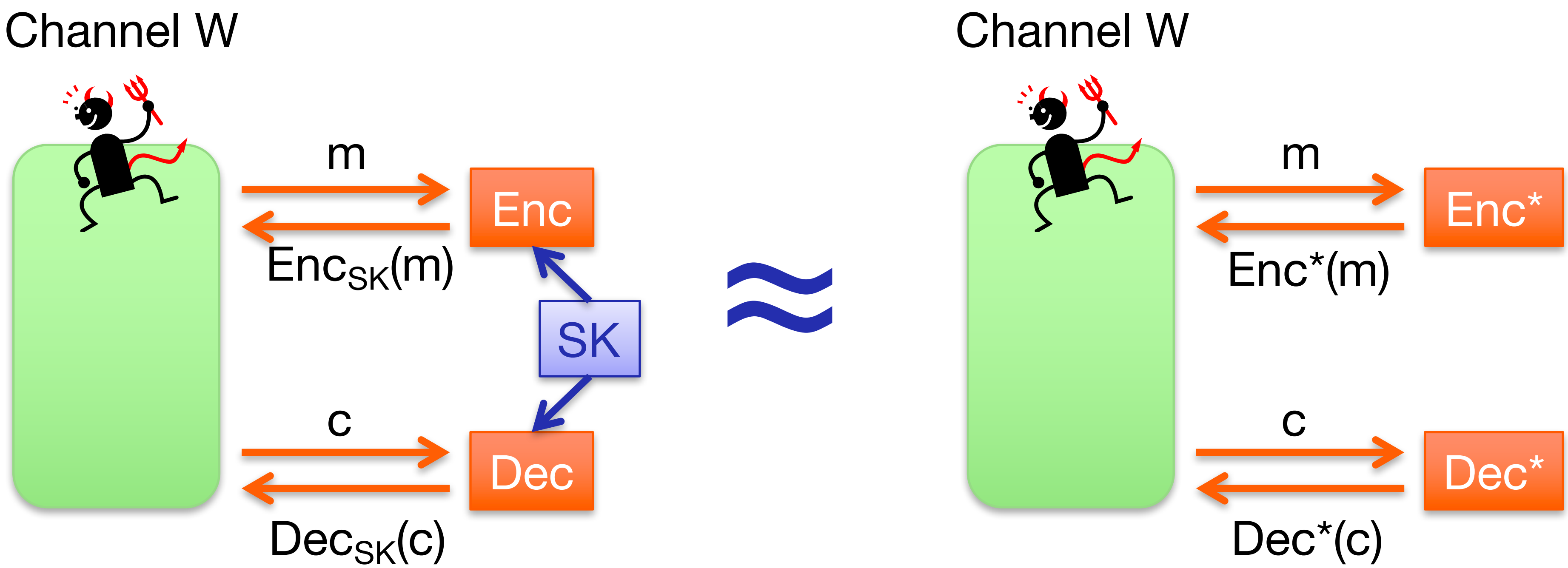
Code Construction

- Guruswami-Smith code [GS'10]
  - Optimal-rate list-decodable code for n^c-time channels for any $c > 0$
  - No setting (secret key or public key) is needed
  - Assuming pseudorandom codes (PRC)
    - PRC C ⇔ (1) list decodable (2) C(m) is pseudorandom
    - Probabilistic construction [GS'10]

    ➔ Explicit construction in "secret-key" setting
- Our approach:
  - Modify explicit GS code in SK setting to have CCA security

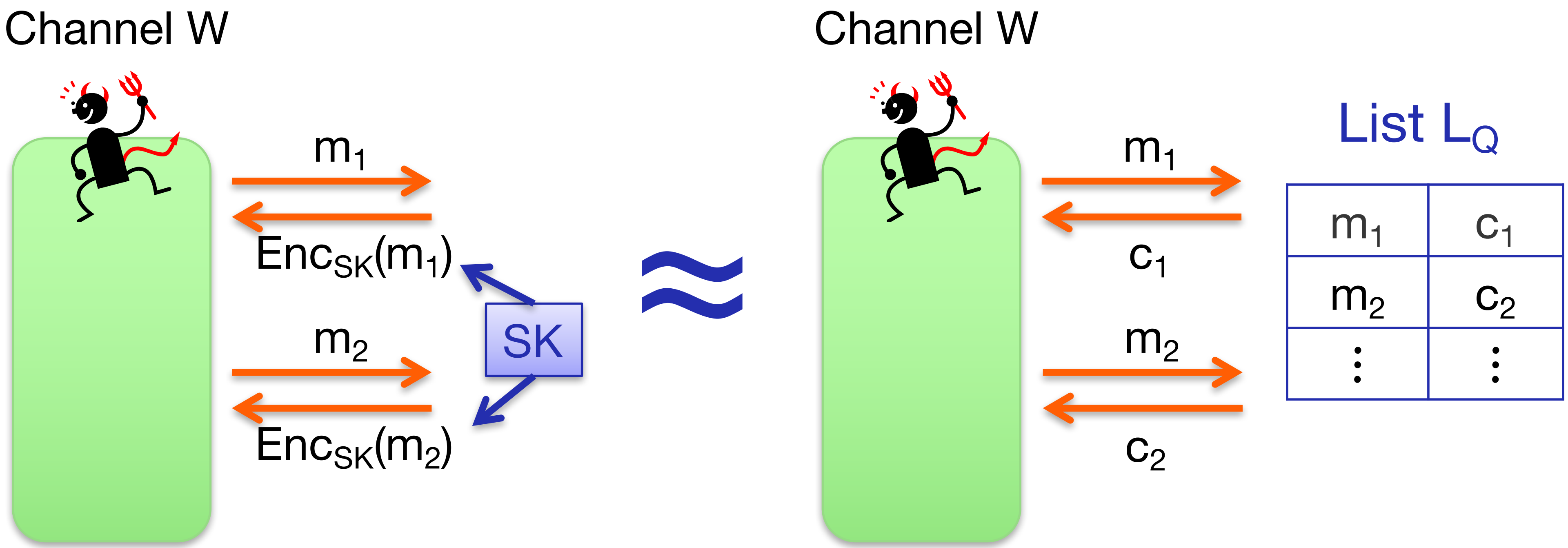
Ideas of the Construction

- Need to simulate Enc/Dec oracles w/o secret key



How to simulate Enc oracle

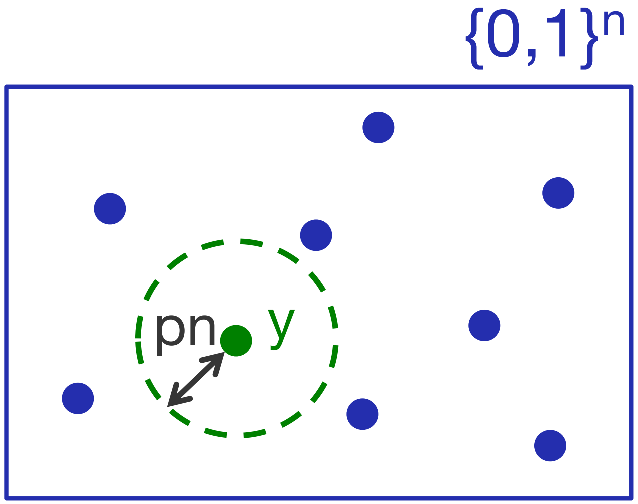
- If $Enc_{SK}(m)$ is pseudorandom, Enc is simulatable
  - For query m_i, reply with randomly chosen c_i



- GS codewords are pseudorandom. Done!

How to simulate Dec oracle

- On query y, need to reply with $L(y) = \{ m : dist_H(y, Enc_{SK}(m)) \leq pn \}$
  - How to deal with exponentially-many $Enc_{SK}(\{0,1\}^k)$?



Fact: $\forall y \in \{0,1\}^n$, given $M=2^k$ random $c_1, \ldots, c_M \in \{0,1\}^n$

Every c_i lies outside Ball(y, pn) with high probability

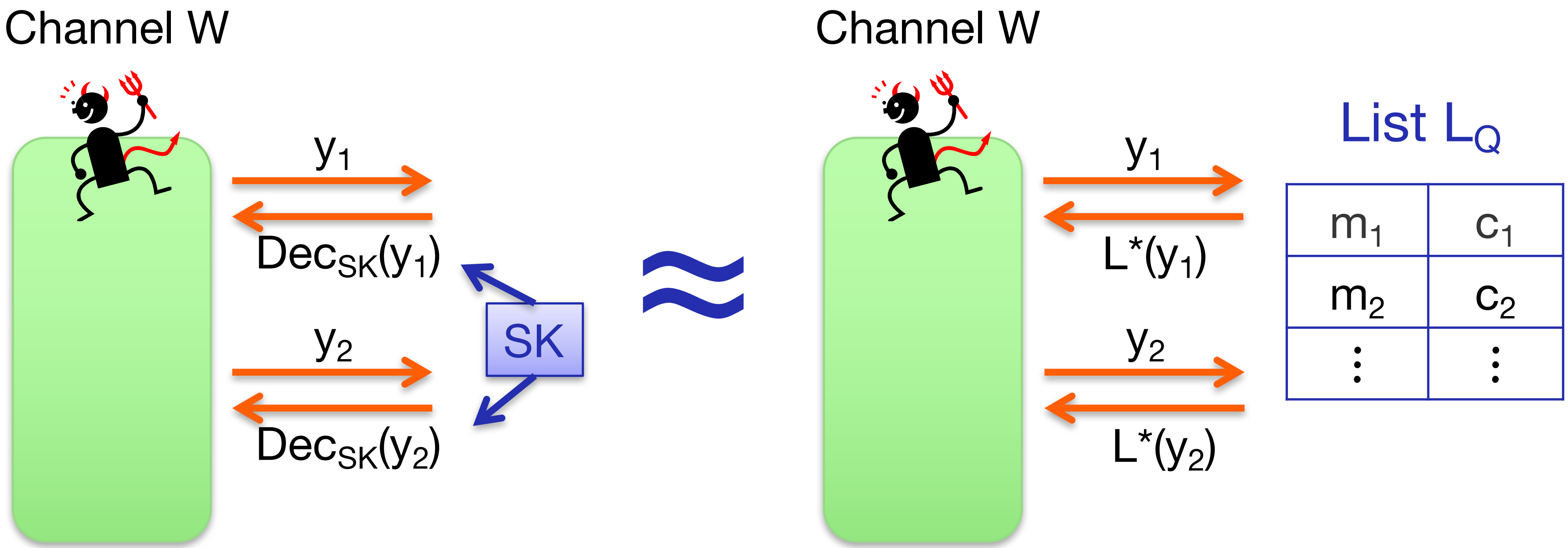
$\Leftrightarrow$

$$\Pr[\ \forall\ c_i,\ dist_H(y, c_i) > pn\ ] = (\ 1 - |Ball(y, pn)| / 2^n\ )^M \approx 1 - 2^{-\varepsilon n}$$

where $R = 1 - h(p) - \varepsilon$

How to simulate Dec oracle

- On query y, sufficient to reply with
  $L^*(y) = \{ m_i : \mathrm{dist}_H(y, c_i) \leq pn \wedge (m_i, c_i) \in L_Q \}$



- W may generate codewords w/o querying Enc
  → Prevented by adding MAC tag to messages

Other Contribution

- Simplify GS code construction by using cryptographic tools

GS Construction	→	Our Construction
randomness-efficient sampler		PRG
t-wise ind. perm. generator		PRG + simple generator
t-wise ind. string generator		PRG
t-wise-error correcting code		BSC code

Main Theorem

Assuming OWF,
$\forall$ p $\in$ (0,1/2), ε > 0, c > 0,
$\exists$ explicit CCA-secure code with $R = 1 - h(p) - \varepsilon$
that corrects p-fraction errors introduced by
n^c-time channels in SK setting

- Encoder/Decoder run in poly(n)-time

Future Work

- CCA security for unbounded poly-time channels
  - Need PRC secure for unbounded poly-time
- Construction in other settings, PK/CRS
  - Need PRC in PK/CRS setting

Thank you

Pseudorandom Codes (PRC)

- PRC : $\{0,1\}^{Rb} \times \{0,1\}^b \rightarrow \{0,1\}^b$
  1. (1/2 – ε, L)-list decodable for any ε > 0:
     $\Leftrightarrow \forall y \in \{0,1\}^b, \exists d \leq L$ codewords $c_1, \ldots, c_d$
     s.t. $\text{dist}(y, c_i) \leq (1/2 - \varepsilon)b$
  2. $\text{PRC}(m; U_b)$ is pseudorandom

- Probabilistic construction of [GS'10]
  - $\text{PRC}(m; r) = C(m) \oplus G(r)$,
    C is (1/2 – ε, L)-list decodable code, G is PRG
  - If $G : \{0,1\}^{O(\log n)} \rightarrow \{0,1\}^{O(\log n)}$ is randomly chosen,
    G is secure for n^c-time adversaries w.h.p.

Ingredients of the Construction (1/2)

- p-error correcting code REC: $\{0,1\}^{R'n'} \to \{0,1\}^{n'}$
  - correcting p-fraction random errors
  - $n' = k + \lambda, \ \ \lambda = k^{1/2}$
  - $\exists$ explicit codes with $R' = 1 - h(p) - \varepsilon$

- Reed-Solomon code RS: $\{0,1\}^{3\lambda} \to F^{\kappa}q$
  - list-recovering property
  - erasure decoding property

- Pseudorandom code PRC : $\{0,1\}^{R_2 b} \times \{0,1\}^{b} \to \{0,1\}^{b}$
  - $\exists$ in the secret-key setting

Ingredients of the Construction (2/2)

- MAC (Tag, Vrfy) with Tag_{SK}: $\{0,1\}^k \rightarrow \{0,1\}^\lambda$

- PRG G : $\{0,1\}^n \rightarrow \{0,1\}^{p(n)}$ for any poly p(n)
  - To generate
    (1) a random bit-permutation π over [n']
    (2) a pseudorandom mask μ
    (3) a set of random samples $V \subseteq [t]$
    each with $\lambda = k^{1/2}$-bit seed

- PRF $F = \{F_s : \{0,1\}^n \rightarrow \{0,1\}^n\}_s$
  - To make Enc deterministic
    by using $F_s(m)$ as random coins for GS code