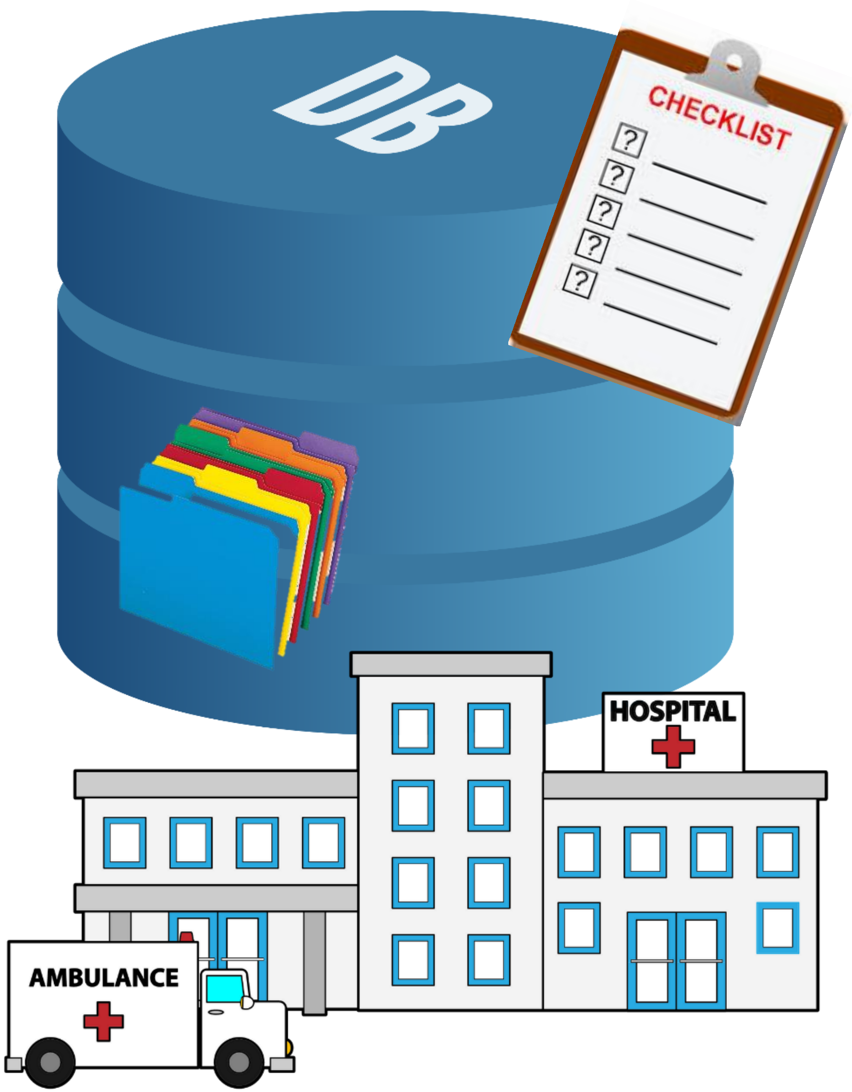
Format-Preserving Encryption
Part I: Introduction and Definitions

Mor Weiss

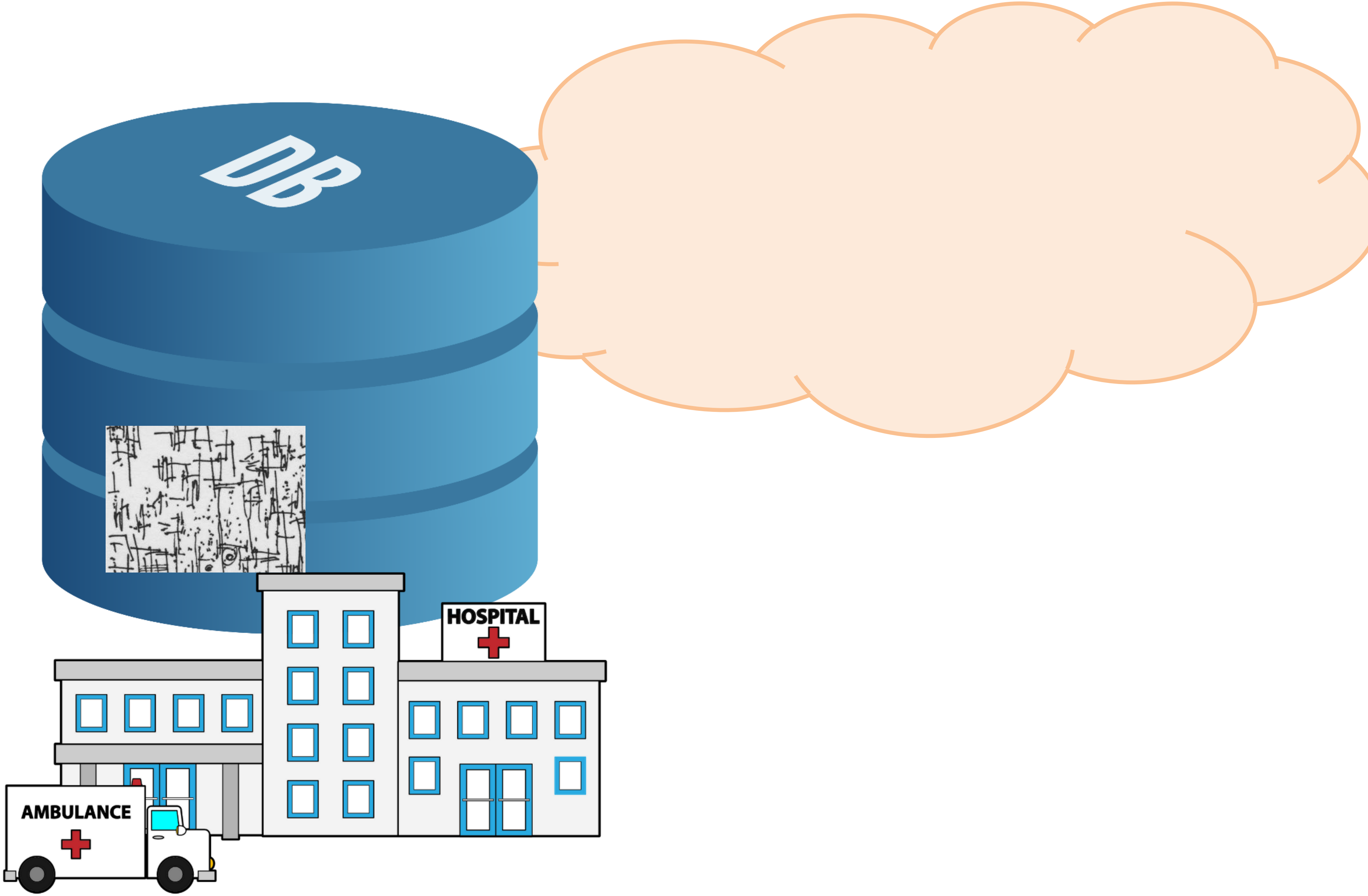
Technion

Winter School on Cryptography in the Cloud:
Verifiable Computation and Special Encryption
Bar-Ilan University

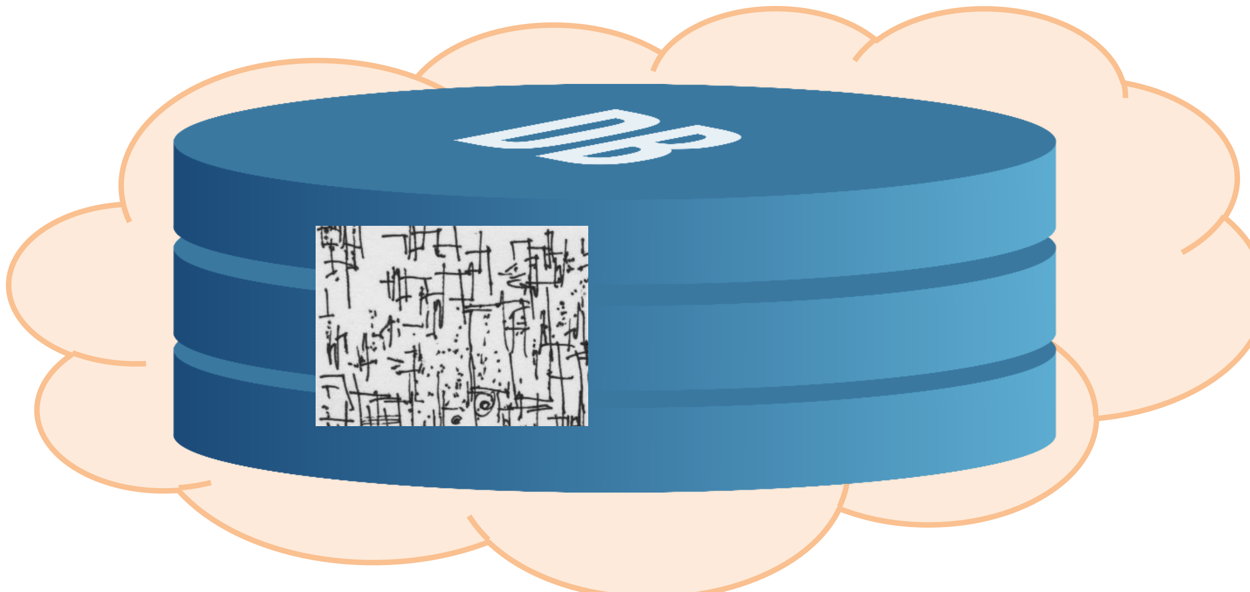
Why Format Preserving Encryption?



Why Format Preserving Encryption?

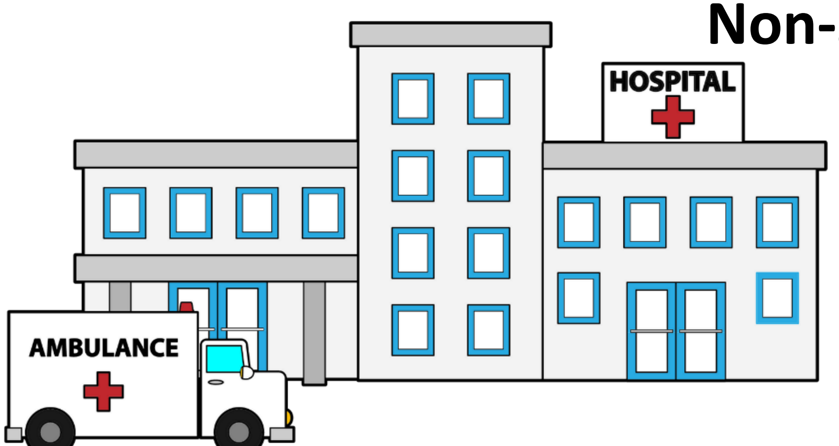


Why Format Preserving Encryption?

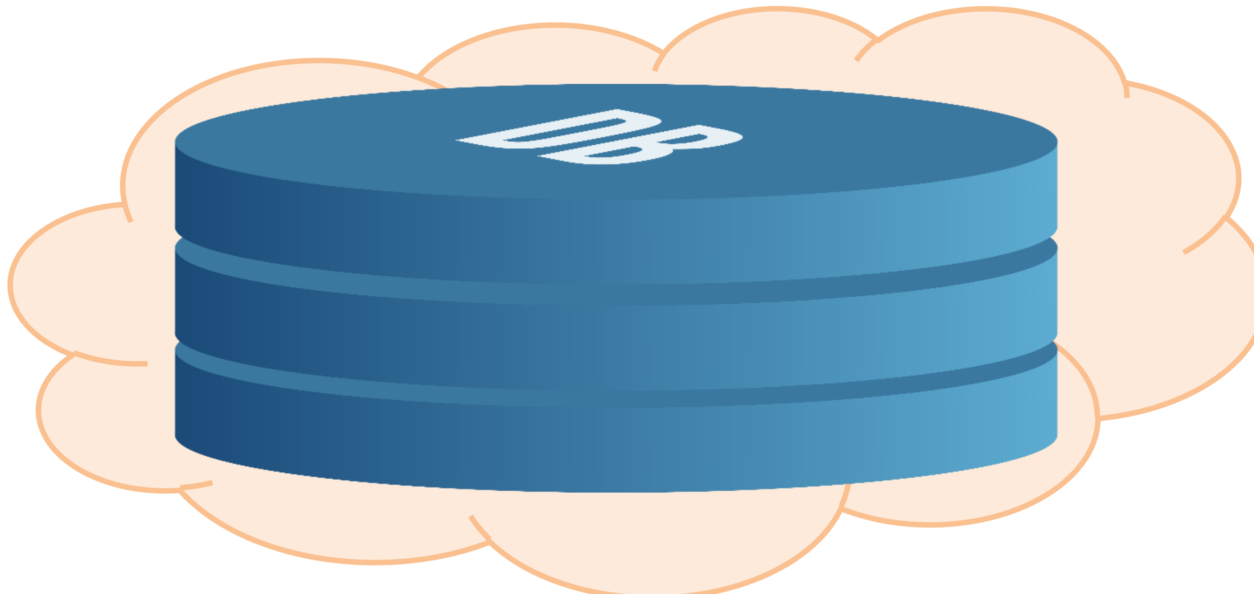


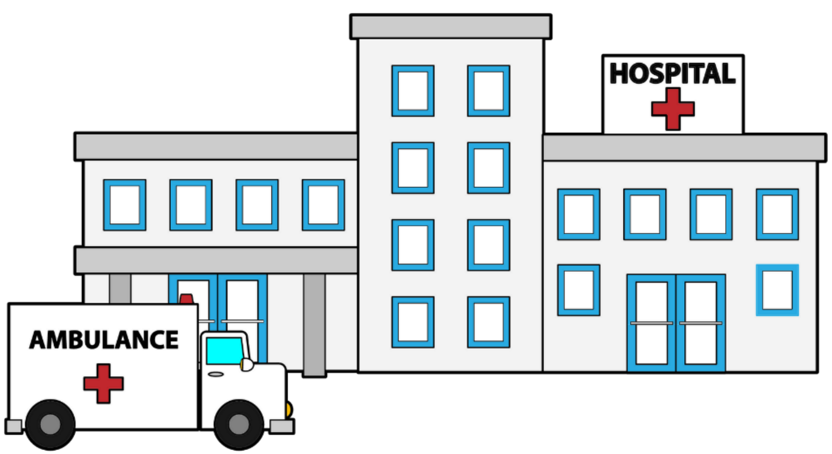
Problem (1): encrypted entry incompatible with database entry structure

Non-solution (1): generate new tables

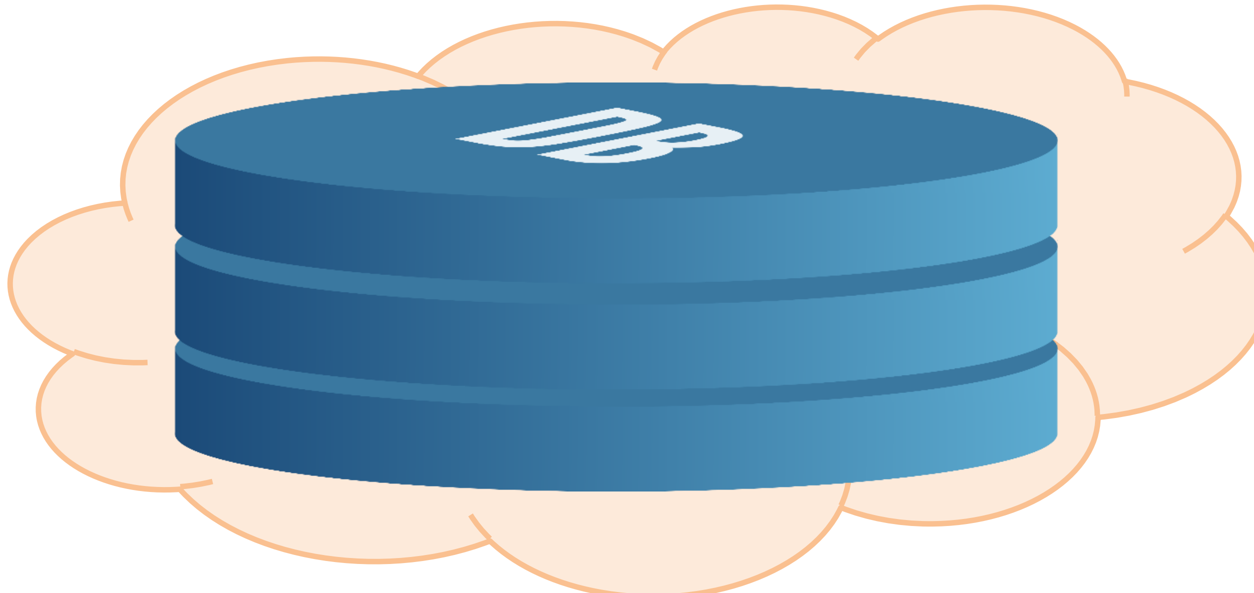


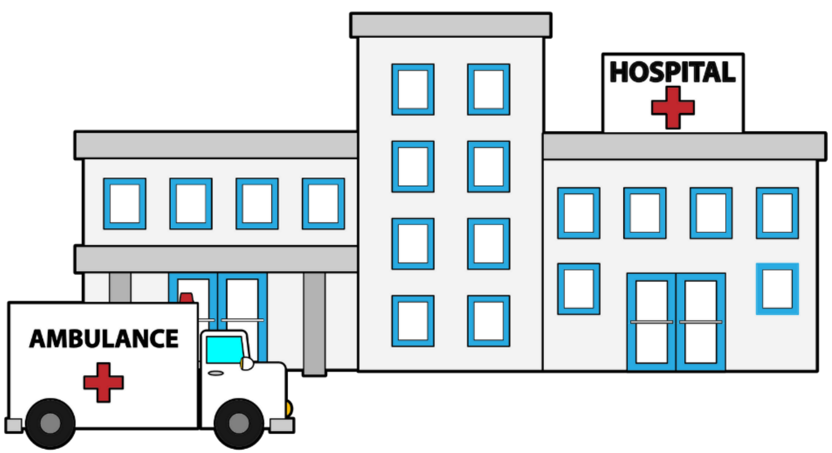
Why Format Preserving Encryption?



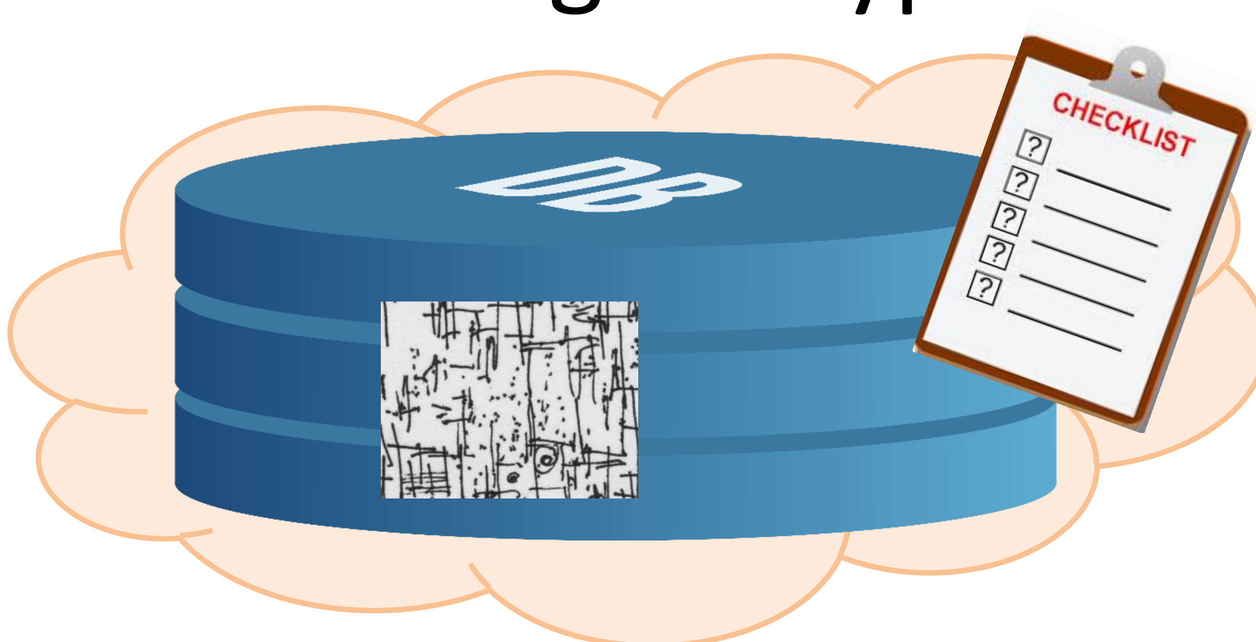


Why Format Preserving Encryption?



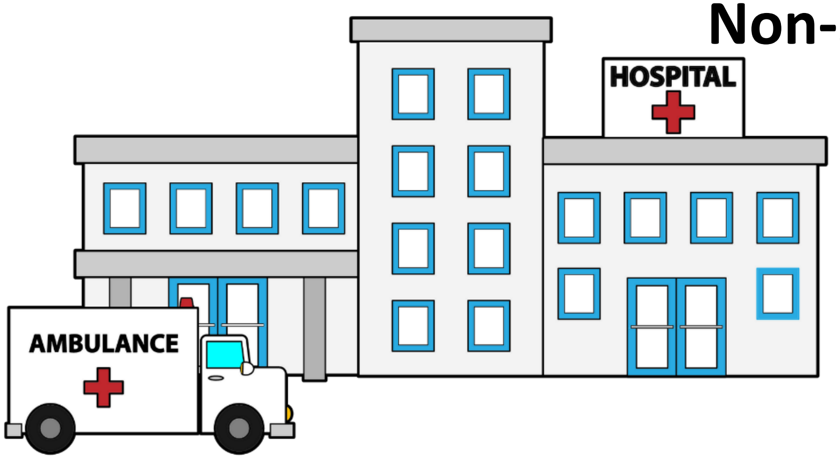


Why Format Preserving Encryption?



Problem (2): encrypted entry incompatible with applications using data

Non-solution (2): re-write applications



Session I: Outline

- Tweakable ciphers: motivation and definition
- Format Preserving Encryption (FPE):
  - Security definitions
  - Relations between definitions
- FPE constructions

Tweakable Encryption: Introduction

- In these sessions: all encryption schemes are **deterministic and private-key**
- **Deterministic Encryption Scheme** Π:
  - Message space $\mathcal{M}$
  - Randomized $KeyGen: \mathbb{N} \to \mathcal{K}$
  - *Deterministic* $E: \mathcal{K} \times \mathcal{M} \to \mathcal{C}$
  - *Deterministic* $D: \mathcal{K} \times \mathcal{C} \to \mathcal{M}$
- **Semantics:** correctness and secrecy
- **Notation:**
  - $E_K = E(K,\cdot)$
  - $D_K = D(K,\cdot)$
- "Unpredictability": only due to encryption key K
  - For random K, E_K "similar" to random permutation on $\mathcal{M}$

Tweakable Encryption: Introduction (2)

- Key-provided "unpredictability" insufficient for small $\mathcal{M}$
  - Example: credit card numbers

4385822056110982

Tweakable Encryption: Introduction (2)

- Key-provided "unpredictability" insufficient for small $\mathcal{M}$
  - Example: credit card numbers

4 38582 205611 0982

Issuer & bank number

Account number

Check digits

Tweakable Encryption: Introduction (2)

- Key-provided “unpredictability” insufficient for small $\mathcal{M}$
  - Example: credit card numbers

4 38582 $E_k(205611)$ 0982

Issuer & bank number

Account number

Check digits

Tweakable Encryption: Introduction (2)

- Key-provided "unpredictability" insufficient for small $\mathcal{M}$
  - Example: credit card numbers

4 38582 $E_k(205611)$ 0982

Tweakable Encryption: Introduction (2)

- Key-provided "unpredictability" insufficient for small $\mathcal{M}$
  - Example: credit card numbers

4 38582	$E_k(205611)$	0982
4 48539	205611	2836

Tweakable Encryption: Introduction (2)

- Key-provided “unpredictability” insufficient for small $\mathcal{M}$
  - Example: credit card numbers

4 38582 $E_k(205611)$ 0982

4 48539 $E_k(205611)$ 2836

- **Problem:** dictionary attacks allow decrypting unknown ciphertexts!
- **Want:** different plaintexts $\Rightarrow$ encryption uses different pseudorandom permutations
- **Solution:** “tweak” encryption using public info!

Tweakable Encryption: Definition

- Deterministic **Tweakable Encryption Scheme** Π [LRW`02]:
  - Message space $\mathcal{M}$
  - **Tweak space $\mathcal{T}$**
  - Randomized $KeyGen: \mathbb{N} \to \mathcal{K}$
  - *Deterministic* $E: \mathcal{K} \times \boldsymbol{\mathcal{T}} \times \mathcal{M} \to \mathcal{C}$
  - *Deterministic* $D: \mathcal{K} \times \boldsymbol{\mathcal{T}} \times \mathcal{C} \to \mathcal{M}$
- **Notation:**
  - $E_K^T = E(K, T, \cdot)$
  - $D_K^T = D(K, T, \cdot)$
- “Unpredictability”: ***still*** only due to encryption key K, but…
- … for random K, $E_K^{T_1}(\cdot), E_K^{T_2}(\cdot)$ “similar” to ***independent*** random permutations
- Tweaks give **family** of pseudorandom permutations
  - Different pseudorandom permutation for every plaintext
- Tweak fundamentally different than key
  - Provides **variability**, NOT **unpredictability**

Tweakable Encryption: Example

- Deterministic encryption is problematic in small domains
  - E.g., credit card numbers
- Before:

unencrypted

4 38582	205611	0982
4 48539	205611	2836
4 38582	$E_k(205611)$	0982
4 48539	$E_k(205611)$	2836

encrypted

Tweakable Encryption: Example

- Deterministic encryption is problematic in small domains
  - E.g., credit card numbers
- Before:

unencrypted

4 38582	205611	0982
4 48539	205611	2836
4 38582	849682	0982
4 48539	849682	2836

encrypted

Tweakable Encryption: Example

- Deterministic encryption is problematic in small domains
  - E.g., credit card numbers
- Before:

encrypted				unencrypted
	4 38582	205611	0982	
	4 48539	205611	2836	
	4 38582	849682	0982	
	4 48539	849682	2836	

- Tweaks solve the problem
  - All available public info used as tweak
- Now:

4 38582	$E_K^{4385820982}(205611)$	0982
4 48539	$E_K^{4485392836}(205611)$	2836

Tweakable Encryption: Example

- Deterministic encryption is problematic in small domains
  - E.g., credit card numbers
- Before:

	4 38582	205611	0982
	4 48539	205611	2836
	4 38582	849682	0982
	4 48539	849682	2836

unencrypted

encrypted

- Tweaks solve the problem
  - All available public info used as tweak
- Now:

$\alpha \rightarrow$ 4 38582 $\quad E_K^{(\alpha,\beta)}(205611) \quad$ 0982 $\leftarrow \beta$

$\alpha' \rightarrow$ 4 48539 $\quad E_K^{(\alpha',\beta')}(205611) \quad$ 2836 $\leftarrow \beta'$

Tweakable Encryption: Example

- Deterministic encryption is problematic in small domains
  - E.g., credit card numbers
- Before:

unencrypted

4 38582	205611	0982
4 48539	205611	2836
4 38582	849682	0982
4 48539	849682	2836

encrypted

- Tweaks solve the problem
  - All available public info used as tweak
- Now:

tweak plaintext

$\alpha \rightarrow$ 4 38582 $\quad E_K^{\boxed{(\alpha,\beta)}}(\boxed{205611}) \quad$ 0982 $\leftarrow \beta$

$\alpha' \rightarrow$ 4 48539 $\quad E_K^{(\alpha',\beta')}(205611) \quad$ 2836 $\leftarrow \beta'$

Tweakable Encryption: Example

- Deterministic encryption is problematic in small domains
  - E.g., credit card numbers
- Before:

unencrypted

4 38582	205611	0982
4 48539	205611	2836
4 38582	849682	0982
4 48539	849682	2836

encrypted

- Tweaks solve the problem
  - All available public info used as tweak
- Now:

4 38582	237849	0982
4 48539	967395	2836

Tweakable Encryption: History

- Tweakable block ciphers [LRW`02] use tweak to
  - Design better "modes of operation"
    - Instead of a fixed IV
  - Improve efficiency
    - Instead of replacing encryption key
- In small domains: tweaks are ***essential***!
- Many formats for which format preserving encryption is needed are small
  - Social security numbers (SSNs), credit card numbers (CCNs),…

Format-Preserving Encryption

Format-Preserving Encryption (FPE): Introduction

- Standard encryption maps messages to "garbage", causing
  - Applications using data to crash
  - Tables designed to store data unsuitable for storing encrypted data
- Sometimes plaintext properties should be preserved
- **Want:** $\mathcal{M} = \mathcal{C}$
  - i.e., E_K^T is a permutation over $\mathcal{M}$
- $\mathcal{M}$ is union of messages over all supported formats
  - Supported formats are called "slices"
- **Examples:**
  - $\mathcal{M} = \mathrm{SSNs} \cup \mathrm{CCNs} \cup \mathrm{Dates} \cup \{1, \ldots, N\}$
  - $\mathcal{M} = \cup_{n \in \mathbb{N}} \{0,1\}^n$
  - $\mathcal{M} = \cup_{n \in \mathbb{N}} \mathbb{Z}_n$

FPE: Syntactic Definition

- **Format-Preserving Encryption (FPE)** Π [BRRS`09]:
  - **Format space $\mathcal{N}$**
  - Message space $\mathcal{M} = \cup_{N \in \mathcal{N}} \mathcal{M}_N$
    - All $\mathcal{M}_N$'s are finite
  - Tweak space $\mathcal{T}$
  - Randomized $KeyGen: \mathbb{N} \rightarrow \mathcal{K}$
  - *Deterministic* $E: \mathcal{K} \times \mathcal{T} \times \boldsymbol{\mathcal{N}} \times \mathcal{M} \rightarrow \mathcal{M} \cup \{\perp\}$
    - $\perp \notin \mathcal{M}$
    - $E(K, T, N, m) = \perp$ denotes encryption error ($m \notin \mathcal{M}_N$)
      - Failure depends only on N, m and **not** on K, T
    - $E(K, T, N, \cdot)$ is a permutation **over** $\boldsymbol{\mathcal{M}_N}$
  - *Deterministic* $D: \mathcal{K} \times \mathcal{T} \times \boldsymbol{\mathcal{N}} \times \mathcal{M} \rightarrow \mathcal{M} \cup \{\perp\}$
- **Notation:**
  - $E_K^{T,N} = E(K, T, N, \cdot)$
  - $D_K^{T,N} = D(K, T, N, \cdot)$

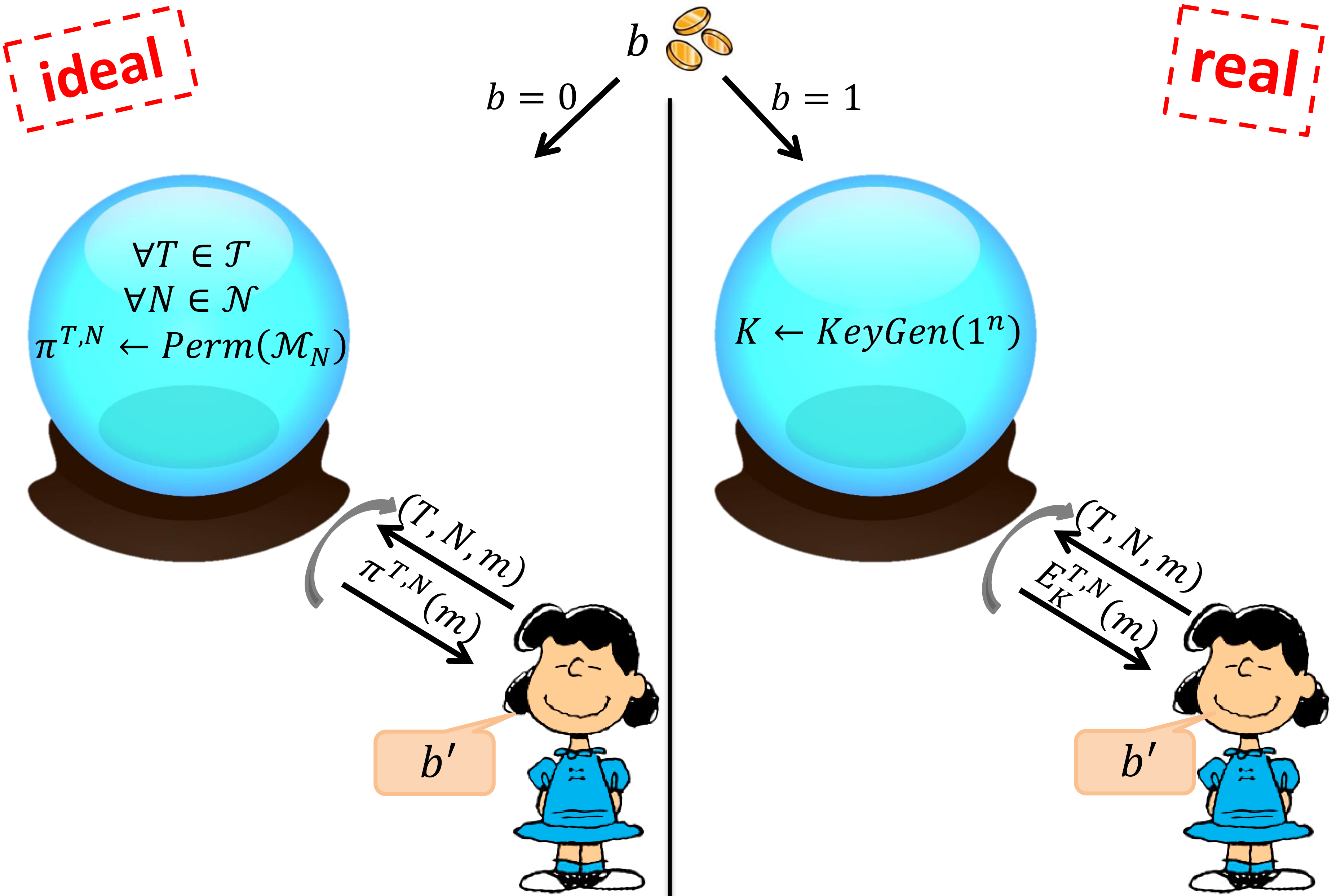
FPE: Semantic Definition

- **Correctness:** for every $K \in \mathcal{K}$, every $T \in \mathcal{T}$, every $N \in \mathcal{N}$ and every $m \in \mathcal{M}_N$

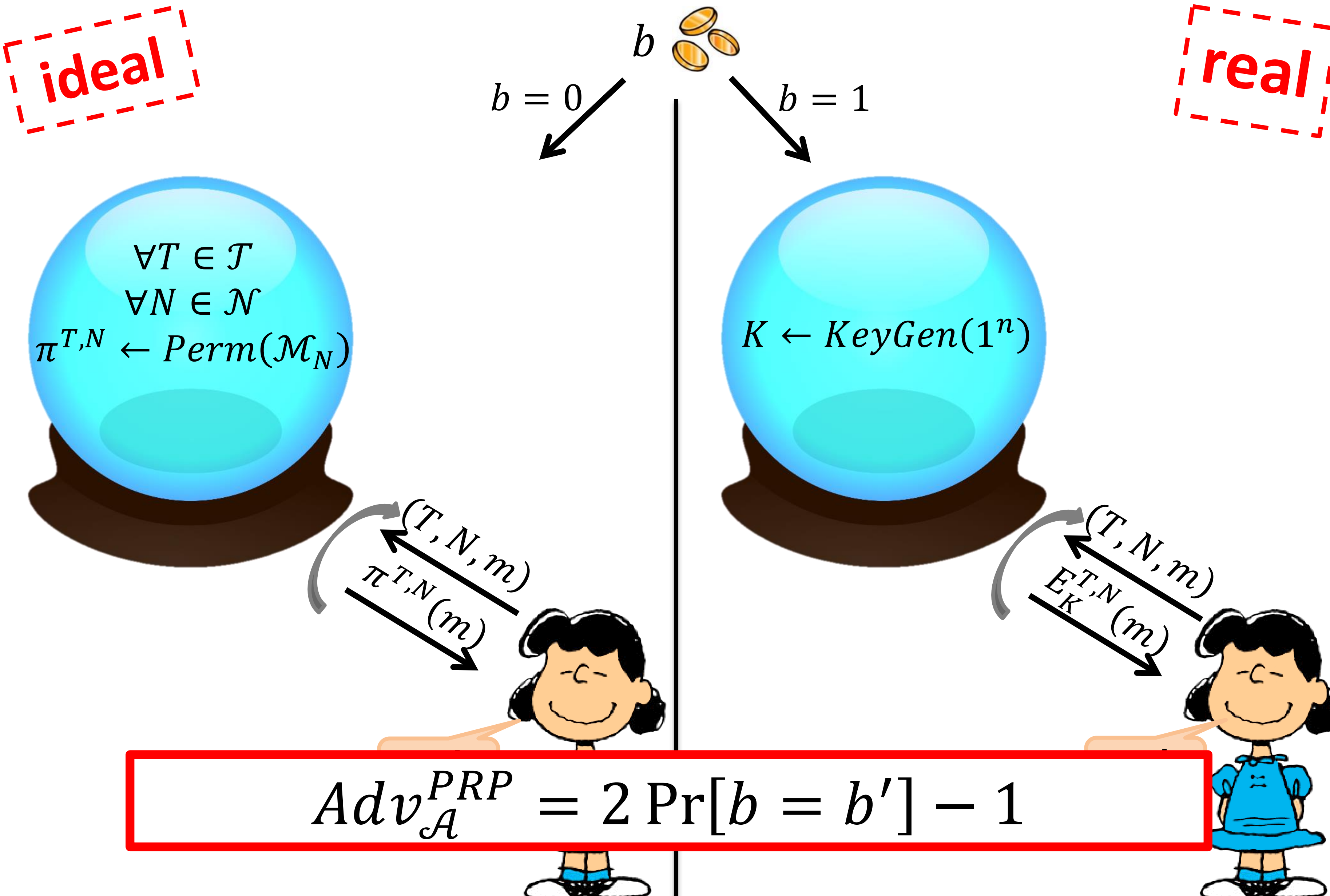
$$D_K^{T,N}\left(E_K^{T,N}(m)\right) = m$$

- **Security:**
  - Hierarchy of security notions [BRRS`09]
  - Strongest: **Pseudo-Random Permutation (PRP)** security
    - K random $\Rightarrow E_K^{T,N}$ close to pseudorandom permutation on $\mathcal{M}_N$

Pseudo-Random Permutation (PRP) security



Pseudo-Random Permutation (PRP) security

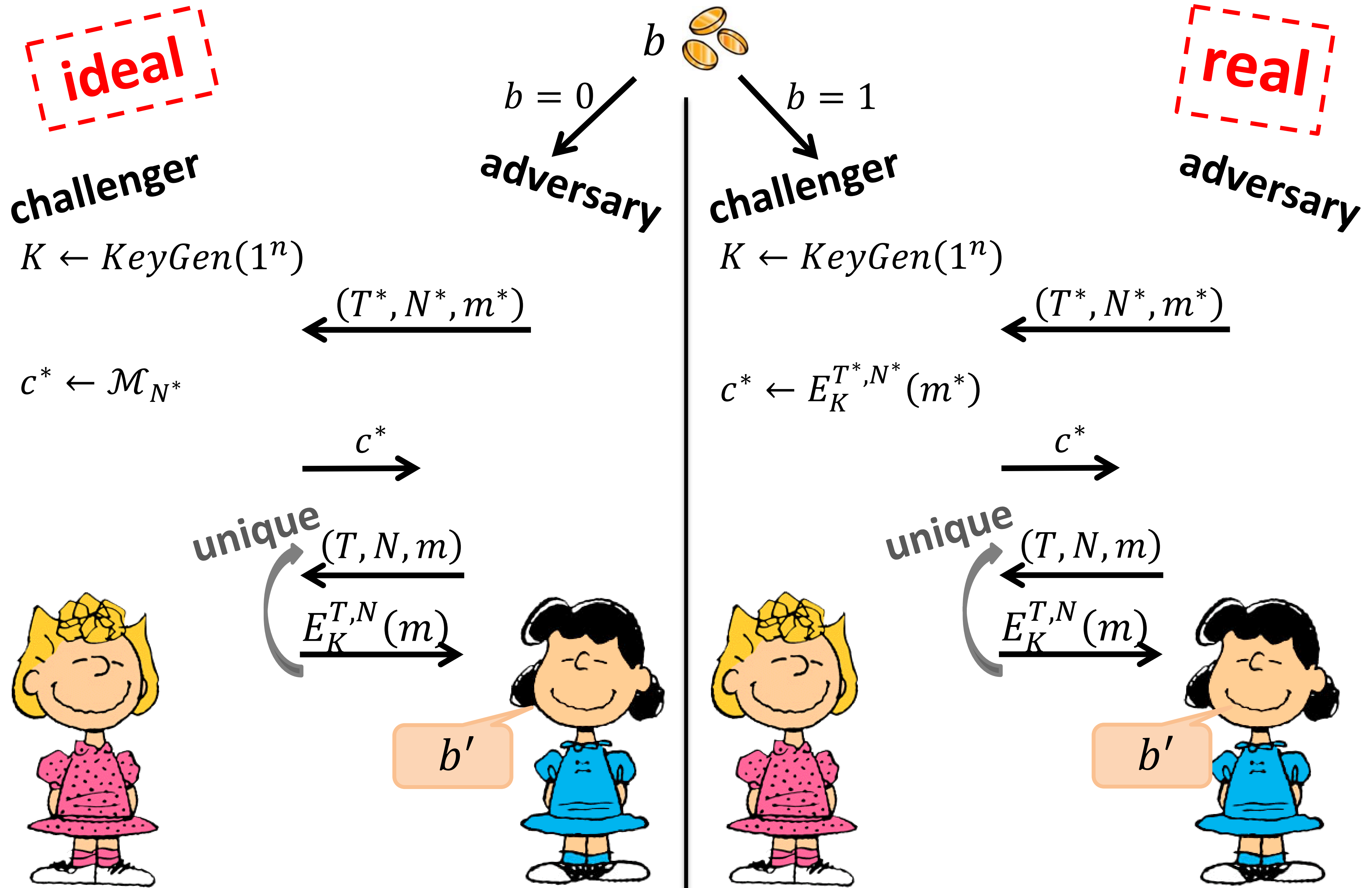


$$Adv_{\mathcal{A}}^{PRP} = 2 \Pr[b = b'] - 1$$

FPE: Security Definitions (2)

- Hierarchy of security notions [BRRS`09]
- Strongest: **Pseudo-Random Permutation (PRP)** security
  - K random $\Rightarrow E_K^{T,N}$ close to pseudorandom permutation on $\mathcal{M}_N$
  - Guaranteed security against (improbable) attacks incurs expensive overhead
  - "Overkill" for typical applications
- **Single Point Indistinguishability (SPI)** security
  - Adversary cannot distinguish encryption of **single** point of its choice from random
  - Analogous to PRF and PRP security notions [GGM`84, DM`00, MRS`09]

Single Point Indistinguishability (SPI) Security



Single Point Indistinguishability (SPI) Security

b 

$b = 0$ | $b = 1$

ideal

challenger | **adversary**

$K \leftarrow KeyGen(1^n)$

$\xleftarrow{(T^*, N^*, m^*)}$

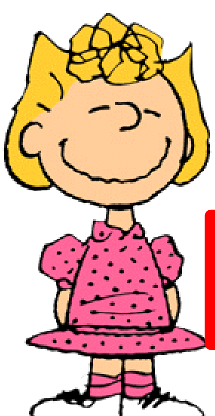
$c^* \leftarrow \mathcal{M}_{N^*}$

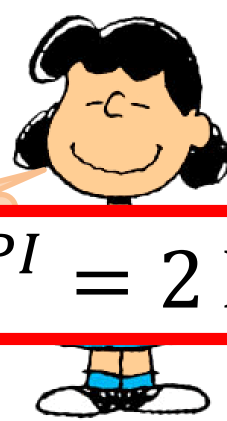
$\xrightarrow{c^*}$

unique

$\xleftarrow{(T, N, m)}$

$\xrightarrow{E_K^{T,N}(m)}$





real

challenger | **adversary**

$K \leftarrow KeyGen(1^n)$

$\xleftarrow{(T^*, N^*, m^*)}$

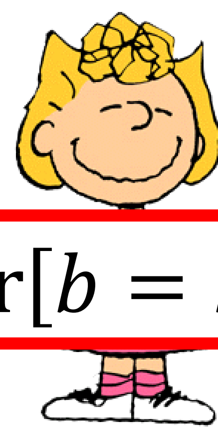
$c^* \leftarrow E_K^{T^*, N^*}(m^*)$

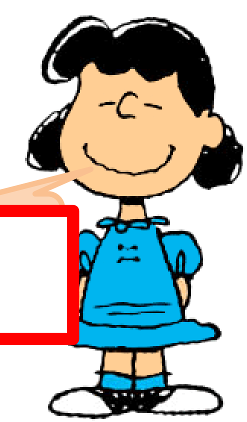
$\xrightarrow{c^*}$

unique

$\xleftarrow{(T, N, m)}$

$\xrightarrow{E_K^{T,N}(m)}$





$$\boxed{Adv_{\mathcal{A}}^{SPI} = 2\Pr[b = b'] - 1}$$

Why SPI?

- **Pseudo-Random Permutation (PRP)**
  - Adversary cannot distinguish encryption oracle from random permutations
  - $Adv_{\mathcal{A}}^{PRP} = 2\Pr[b = b'] - 1$
- **Single Point Indistinguishability (SPI)**
  - Adversary cannot distinguish encryption of **single** point of its choice from random
    - Even given encryption oracle
  - $Adv_{\mathcal{A}}^{SPI} = 2\Pr[b = b'] - 1$
- Equivalent notions, SPI easier to work with
- $\boldsymbol{PRP \Rightarrow SPI}$: $Adv_{\mathcal{A}}^{SPI} \leq \mathbf{2} \cdot Adv_{\mathcal{A}'}^{PRP} + \frac{\boldsymbol{q}}{\boldsymbol{M}}$
  - q = number of queries of PRP adversary
  - M = minimal size of supported format
- $\boldsymbol{SPI \Rightarrow PRP}$: $Adv_{\mathcal{A}}^{PRP} \leq \boldsymbol{q} \cdot Adv_{\mathcal{A}'}^{SPI} + \frac{\boldsymbol{q^2}}{\boldsymbol{M}}$

FPE: Security Definitions (3)

- Hierarchy of security notions [BRRS`09]
- Strongest: **Pseudo-Random Permutation (PRP)** security
  - K random $\Rightarrow E_K^{T,N}$ close to pseudorandom permutation on $\mathcal{M}_N$
- **Single Point Indistinguishability (SPI)** security
  - Adversary cannot distinguish encryption of **single** point of its choice from random
- **Message Privacy (MP)** security
  - "Format-preserving" analog of semantic security
  - Challenge ciphertext c^* *practically* no help in computing $f(m^*)$
  - **Randomized** encryption: "practically" = no help
  - **Deterministic** encryption: "practically" = encryption oracle equivalent to equality oracle

Message Privacy (MP) Security

ideal

challenger | **adversary**

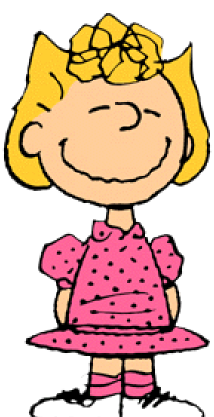
$\xleftarrow{\mathcal{D}}$

Same as in real world

$(T^*, N^*, m^*) \leftarrow \mathcal{D}$

$\xrightarrow{T^*, N^*}$

$q \left(\begin{array}{l} \xleftarrow{m?} \\ \xrightarrow{\text{Yes\\No}} \end{array} \right.$



Same as in real world

f, z

real

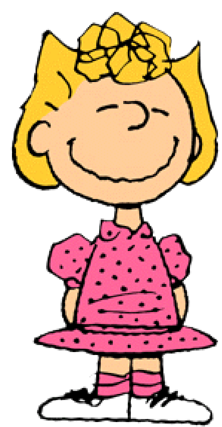
challenger | **adversary**

$\xleftarrow{\mathcal{D}}$

$K \leftarrow KeyGen(1^n)$
$(T^*, N^*, m^*) \leftarrow \mathcal{D}$
$c^* \leftarrow E_K^{T^*, N^*}(m^*)$

$\xrightarrow{T^*, N^*, c^*}$

$q \left(\begin{array}{l} \xleftarrow{(T, N, m)} \\ \xrightarrow{E_K^{T,N}(m)} \end{array} \right.$



f, z

Guess for $f(m^*)$

Message Privacy (MP) Security

ideal

challenger **adversary**

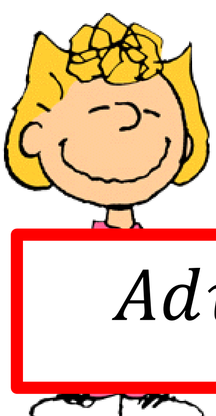
Same as in real world

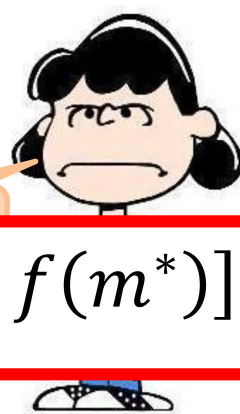
$\xleftarrow{\mathcal{D}}$

$(T^*, N^*, m^*) \leftarrow \mathcal{D}$

$\xrightarrow{T^*, N^*}$

$q \left(\begin{array}{l} \xleftarrow{m?} \\ \xrightarrow{\text{Yes\No}} \end{array} \right.$





Same as in

real

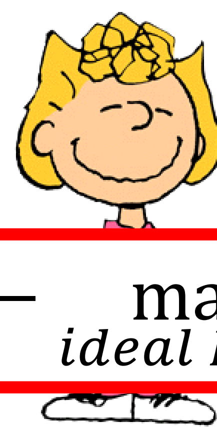
challenger **adversary**

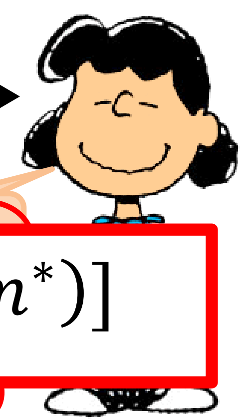
$\xleftarrow{\mathcal{D}}$

$K \leftarrow KeyGen(1^n)$
$(T^*, N^*, m^*) \leftarrow \mathcal{D}$
$c^* \leftarrow E_K^{T^*, N^*}(m^*)$

$\xrightarrow{T^*, N^*, c^*}$

$q \left(\begin{array}{l} \xleftarrow{(T, N, m)} \\ \xrightarrow{E_K^{T,N}(m)} \end{array} \right.$





$f(m^*)$

$$Adv_{\mathcal{A}}^{MP} = \Pr_{real}[z = f(m^*)] - \max_{ideal\ Lucy} \Pr[z = f(m^*)]$$

FPE: Security Definitions (4)

- Hierarchy of security notions [BRRS`09]
- Strongest: **Pseudo-Random Permutation (PRP)** security
  – K random $\Rightarrow E_K^{T,N}$ close to pseudorandom permutation on $\mathcal{M}_N$
- **Single Point Indistinguishability (SPI)** security
  – Adversary cannot distinguish encryption of **single** point of its choice from random
- **Message Privacy (MP)** security
  – "Format-preserving" analog of semantic security
  – Challenge ciphertext c^* practically no help in computing $f(m^*)$
- Weakest: **Message Recovery (MR)** security
  – Adversary cannot **completely** recover challenge plaintext

Message Recovery (MR) Security

ideal

challenger **adversary**

$\xleftarrow{\mathcal{D}}$

Same as in real world

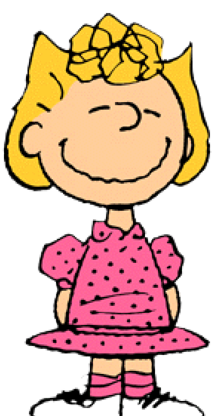
$(T^*, N^*, m^*) \leftarrow \mathcal{D}$

$\xrightarrow{T^*, N^*}$

q

$\xleftarrow{m?}$

$\xrightarrow{\text{Yes\No}}$



$\boldsymbol{m'}$

real

challenger **adversary**

$\xleftarrow{\mathcal{D}}$

$K \leftarrow KeyGen(1^n)$

$(T^*, N^*, m^*) \leftarrow \mathcal{D}$

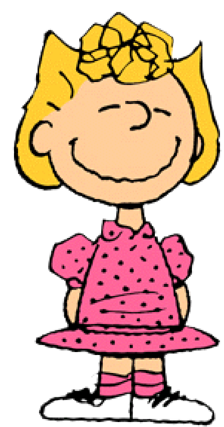
$c^* \leftarrow E_K^{T^*, N^*}(m^*)$

$\xrightarrow{T^*, N^*, c^*}$

q

$\xleftarrow{(T, N, m)}$

$\xrightarrow{E_K^{T,N}(m)}$



$\boldsymbol{m'}$

Message Recovery (MR) Security

ideal

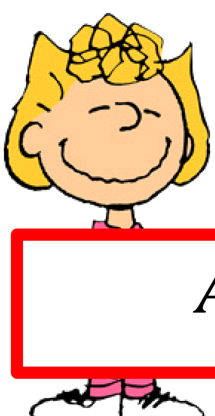
challenger **adversary**

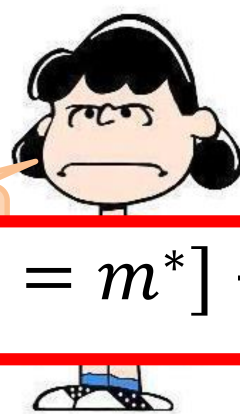
$\xleftarrow{\mathcal{D}}$ Same as in real world

$(T^*, N^*, m^*) \leftarrow \mathcal{D}$

$\xrightarrow{T^*, N^*}$

q ($\xleftarrow{m?}$ $\xrightarrow{\text{Yes\textbackslash No}}$)





real

challenger **adversary**

$\xleftarrow{\mathcal{D}}$

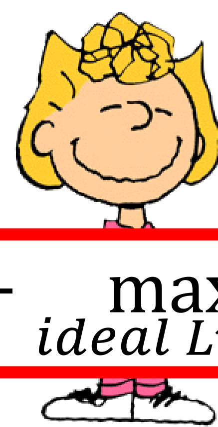
$K \leftarrow KeyGen(1^n)$

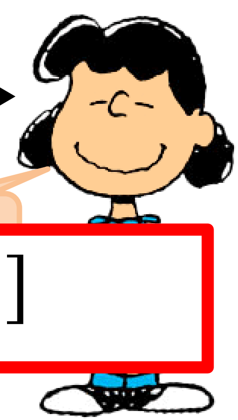
$(T^*, N^*, m^*) \leftarrow \mathcal{D}$

$c^* \leftarrow E_K^{T^*, N^*}(m^*)$

$\xrightarrow{T^*, N^*, c^*}$

q ($\xleftarrow{(T, N, m)}$ $\xrightarrow{E_K^{T,N}(m)}$)





$$\boxed{Adv_{\mathcal{A}}^{MR} = \Pr_{real}[m' = m^*] - \max_{ideal\ Lucy} \Pr[m' = m^*]}$$

FPE: Security Definitions (5)

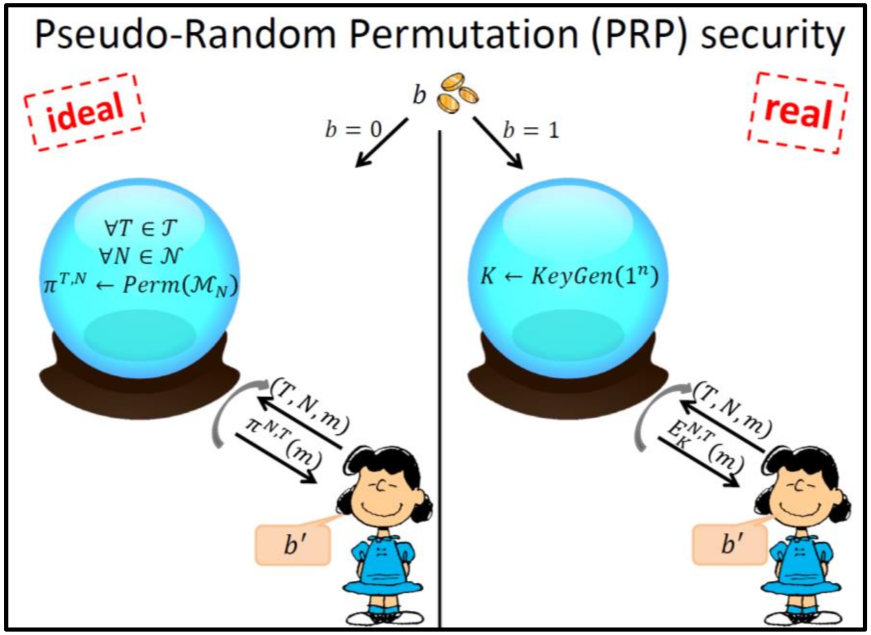
- Hierarchy of security notions [BRRS`09]
  - Pseudo-Random Permutation (PRP)
  - Single Point Indistinguishability (SPI)
  - Message Privacy (MP)
  - Message Recovery (MR)
- Similar to IND-Distinct**CPA** security
- Extends to stronger IND-Distinct**CCA** security:
  - **Strong-PRP:**
    - Real world: adversary given $D_K^{T,N}$
    - Ideal world: adversary given $(\pi^{T,N})^{-1}$
  - **Strong SPI, MP, MR:**
    - Real world: adversary given $D_K^{T,N}$
    - Ideal world: simulator given no additional oracle
- We will work with IND-CPA notions (no decryption oracle)

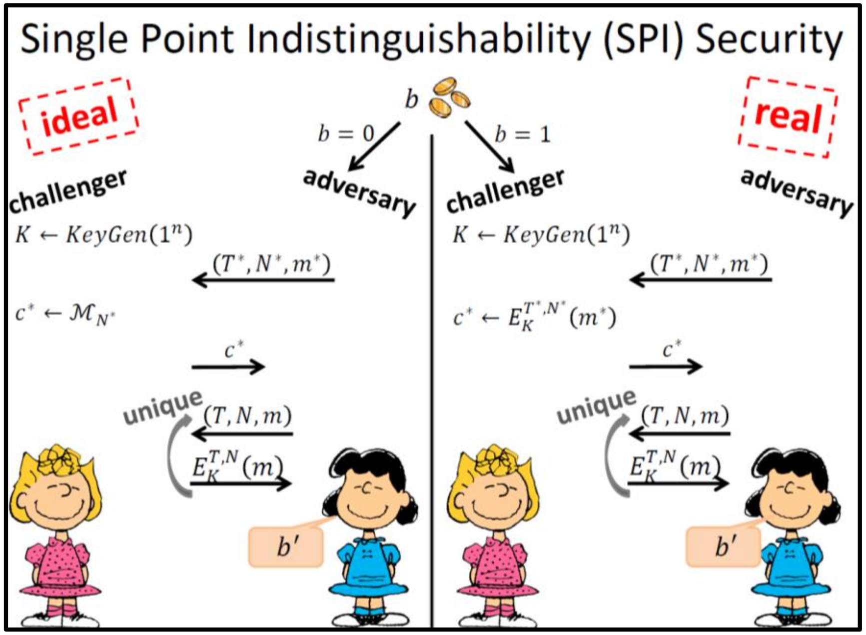
Relations Between Security Definitions

$$\boldsymbol{PRP \Leftrightarrow SPI \Rightarrow MP \Rightarrow MR}$$

PRP: Pseudo Random Permutation
SPI: Single Point Indistinguishability
MP: Message Privacy
MR: Message Recovery

- $PRP \Rightarrow SPI: Adv_{\mathcal{A}}^{SPI} \leq 2 \cdot Adv_{\mathcal{A}'}^{PRP} + \frac{q}{M}$
- $SPI \Rightarrow PRP: Adv_{\mathcal{A}}^{PRP} \leq q \cdot Adv_{\mathcal{A}'}^{SPI} + \frac{q^2}{M}$





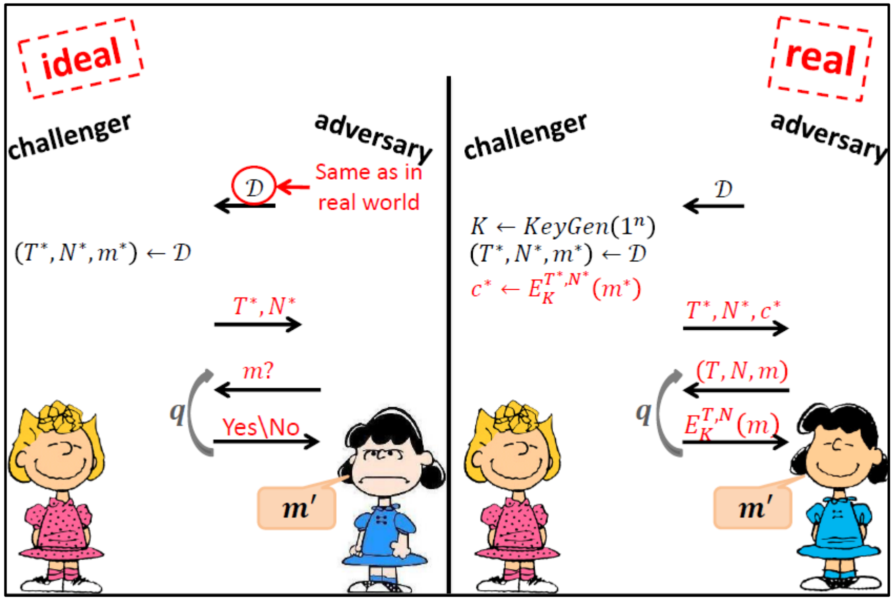
Relations Between Security Definitions (2)

$$PRP \Leftrightarrow SPI \Rightarrow MP \Rightarrow MR$$

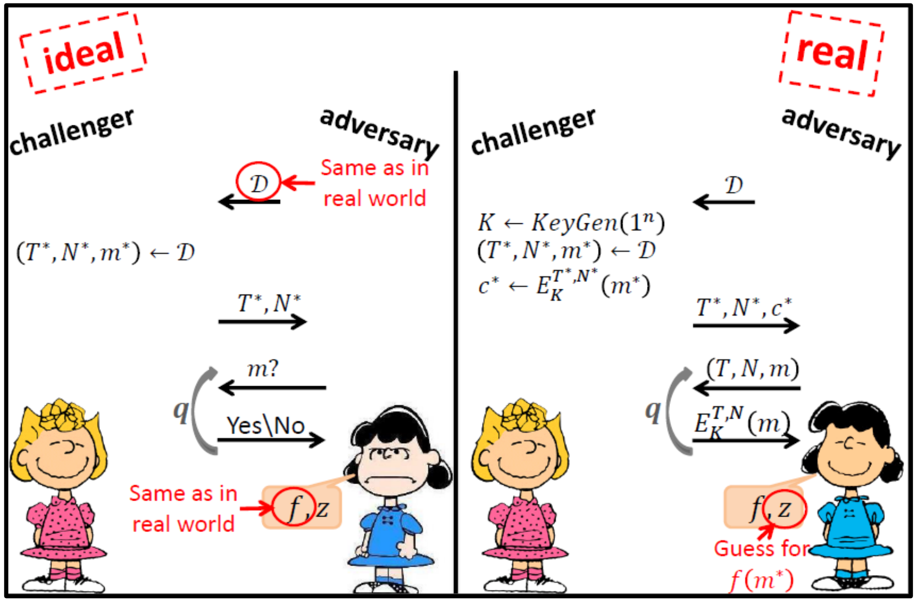
PRP: Pseudo Random Permutation
SPI: Single Point Indistinguishability
MP: Message Privacy
MR: Message Recovery

- $MP \Rightarrow MR$: $Adv_{\mathcal{A}}^{MR} \leq Adv_{\mathcal{A}'}^{MP}$
  - MR special case of MP: $\mathcal{A}'$ chooses f = identity function

MR



MP



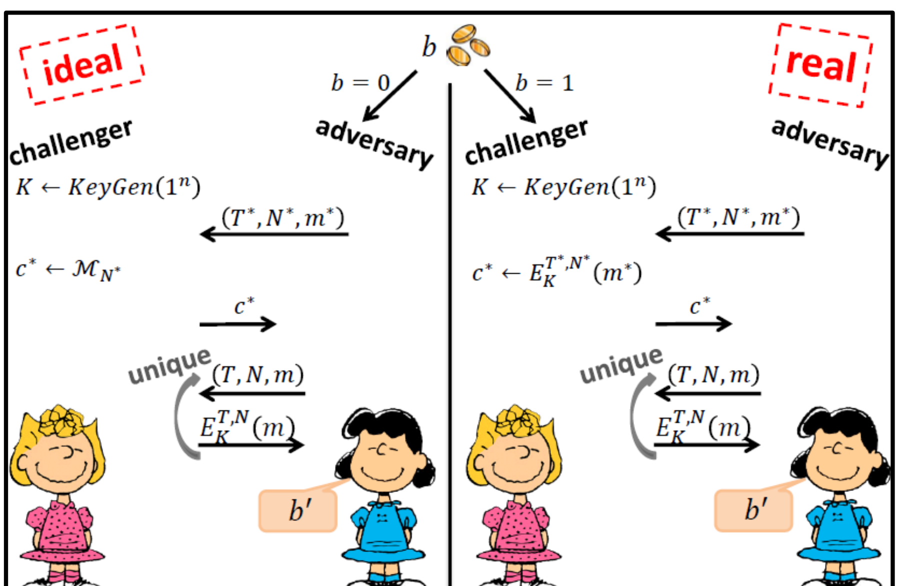
Relations Between Security Definitions (3)

$$PRP \Leftrightarrow SPI \Rightarrow MP \Rightarrow MR$$

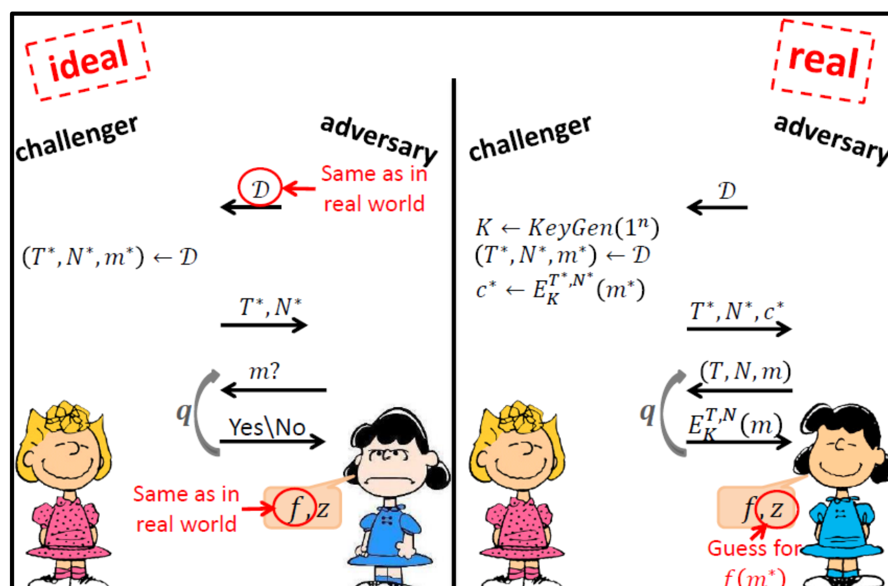
PRP: Pseudo Random Permutation
SPI: Single Point Indistinguishability
MP: Message Privacy
MR: Message Recovery

- $SPI \Rightarrow MP$: $Adv_{\mathcal{A}}^{MP} \leq Adv_{\mathcal{A}'}^{SPI}$

SPI



MP



Relations Between Security Definitions (4)

$$PRP \Leftrightarrow SPI \not\Leftarrow MP \not\Leftarrow MR$$

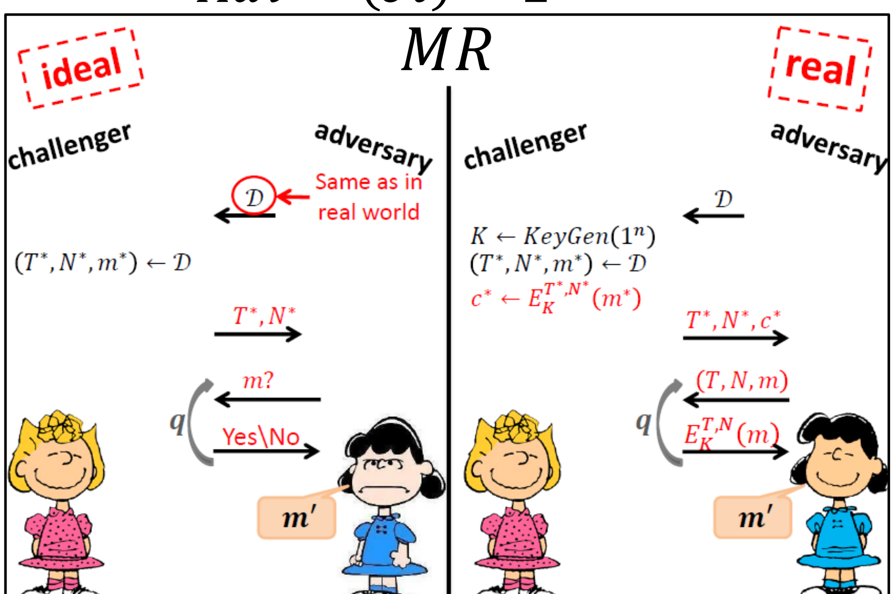
PRP: Pseudo Random Permutation

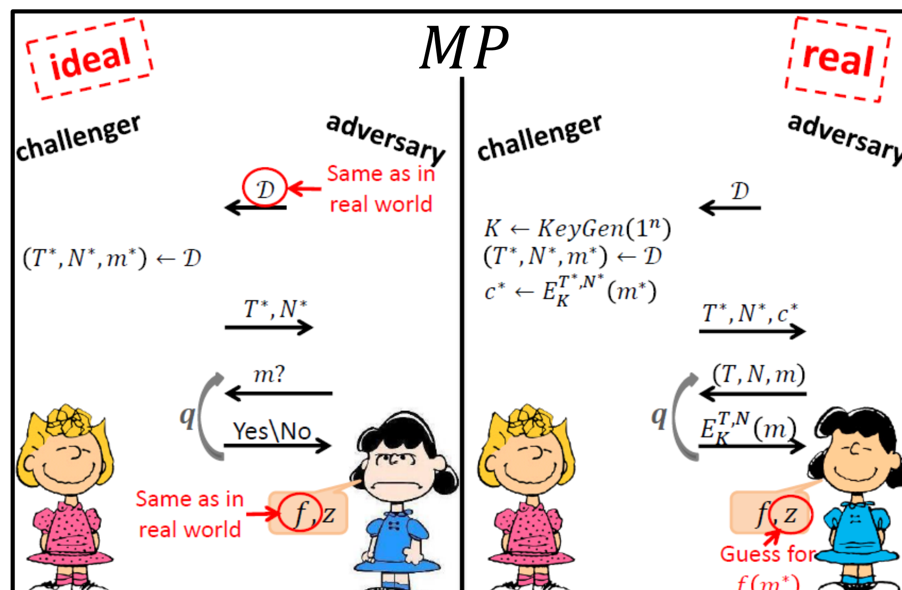
SPI: Single Point Indistinguishability

MP: Message Privacy

MR: Message Recovery

- $MR \nRightarrow MP$:
  - E.g., $\mathcal{M}_N = \{0,1\}^N$ for all $N \in \mathbb{N}$, and encryption is:
    - Identity on first plaintext bit
    - Pseudorandom on other plaintext bits
  - $Adv^{MP}(\mathcal{A}) = 1/2$ ← Compare $\mathcal{A}$ to "best possible" ideal $\mathcal{A}$
  - $Adv^{MR}(\mathcal{A}) \approx 2^{-(N-1)}$





Relations Between Security Definitions (5)

$$\boldsymbol{PRP \Leftrightarrow SPI \notin MP \notin MR}$$

PRP: Pseudo Random Permutation
SPI: Single Point Indistinguishability
MP: Message Privacy
MR: Message Recovery

- $MP \nRightarrow SPI$:
  - e.g., encryption has "fixed point" $m_N \in \mathcal{M}_N$ for every N and every K, T:
    - $\pi_K^{T,N}$ is pseudorandom permutation over $\mathcal{M}_N \setminus \{m_N\}$
    - $m \neq m_N \;\; \Rightarrow \;\; E_K^{T,N}(m) = \pi_K^{T,N}(m)$
    - $E_K^{T,N}(m_N) = m_N$

Relations Between Security Definitions (6)

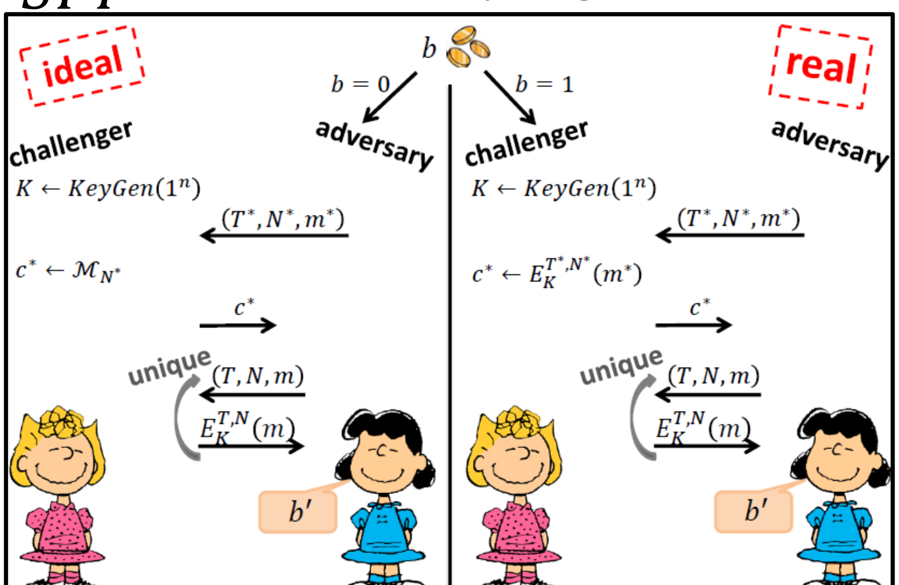
$$PRP \Leftrightarrow SPI \notin MP \notin MR$$

PRP: Pseudo Random Permutation **MP:** Message Privacy

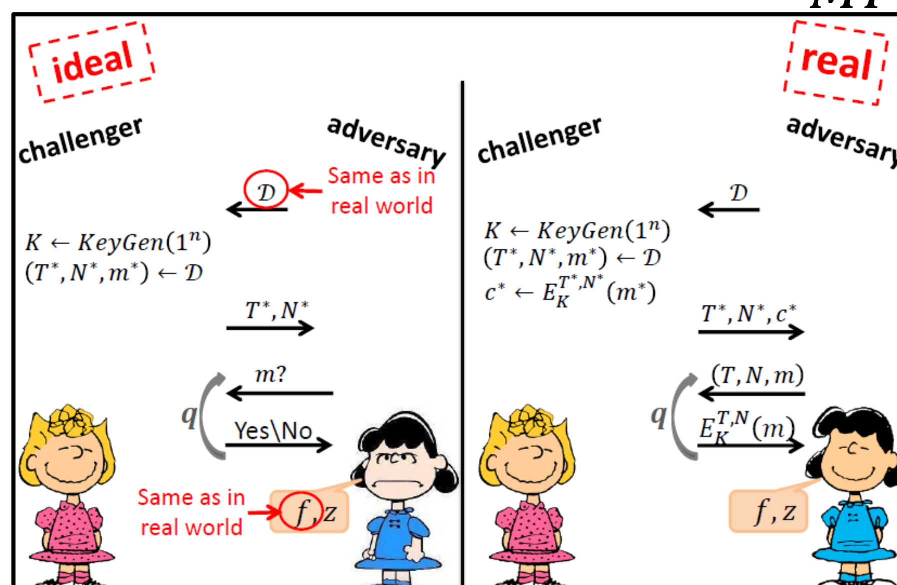
SPI: Single Point Indistinguishability **MR:** Message Recovery

- $MP \not\Rightarrow SPI$:
  - e.g., encryption has "fixed point" $m_N \in \mathcal{M}_N$ for every N and every K, T
  - $\mathcal{A}^{SPI}$ chooses challenge plaintext (T, N, m_N) for maximal $|\mathcal{M}_N|$
    - Has advantage $1 - \frac{1}{|\mathcal{M}_N|}$
  - "Best" $\mathcal{A}^{MP}$: choose "easy to guess" f or m
    - Also "easy to guess" in ideal world

SPI



MP



Recap

- **Tweakable ciphers:** parameterized by key K *and tweak* T
  - Tweak "equivalent" to using pseudorandom permutation *family*
  - Essential when encrypting small domains
- Format Preserving Encryption: preserves message format
  - Hierarchy of security definitions: $PRP \Leftrightarrow SPI \Rightarrow MP \Rightarrow MR$

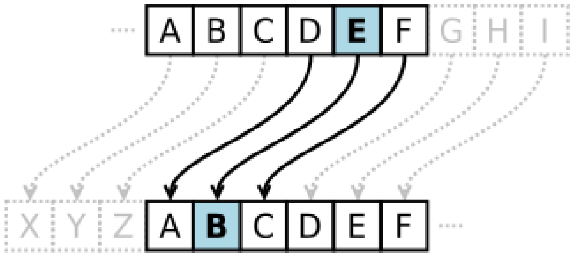
    PRP: Pseudo Random Permutation
    SPI: Single Point Indistinguishability
    MP: Message Privacy
    MR: Message Recovery

    - $PRP \Rightarrow SPI$: $Adv_{\mathcal{A}}^{SPI} \leq 2 \cdot Adv_{\mathcal{A}'}^{PRP} + \frac{q}{M}$
    - $SPI \Rightarrow PRP$: $Adv_{\mathcal{A}}^{PRP} \leq q \cdot Adv_{\mathcal{A}'}^{SPI} + \frac{q^2}{M}$
    - $SPI \Rightarrow MP \Rightarrow MR$ with tight bounds
    - $MR \nRightarrow MP \nRightarrow SPI$

Constructions

What We Know About FPE

- First* FPE



- AES
- Term coined by Terence Spies, Voltage Security's CTO
- First formal definitions due to [BRRS`09]
- Constructions for specific formats
  - Social Security Numbers (SSNs) [Hoo`11]
  - Credit Card Numbers (CCNs)
  - Dates [LJLC`10]
  - …
- **Drawbacks:**
  - Designed for specific formats
  - New encryption techniques, little (if any) security analysis
  - Often inconsistent with syntactic definition
- Interested in schemes for **general** formats
  - Starting point: schemes for integral domains

Format-Preserving Encryption Part II: Integral Domains

Session II: Outline

- Integral and “almost-integral” domains
- Feistel Networks and Generalized Feistel Network
- Integer-FPE constructions from Feistel Networks
- Integer-FPE standards

Integer-FPEs

- In many cases, interested in encrypting integral domains
  - E.g., credit-card numbers
- FPEs for integral (and “almost integral”) domains useful for encrypting general formats
  - Stay tuned…
- **Int-FPE**: FPE for integral domain $\mathbb{Z}_M$ [BR`02,BRRS`09]
- Also interested in FPEs for **“almost integral” domains**

$$\mathcal{M} = \{0,1,\ldots,m-1\}^n \text{ for } n, m \in \mathbb{N}$$

  - Methods described as early as 1981
  - FFX [BRS`10], BPS [BPS`10] under NIST consideration
- We will refer to both as “int-FPE”
- Many constructions based on Feistel Networks

Integer-FPE: Constructions

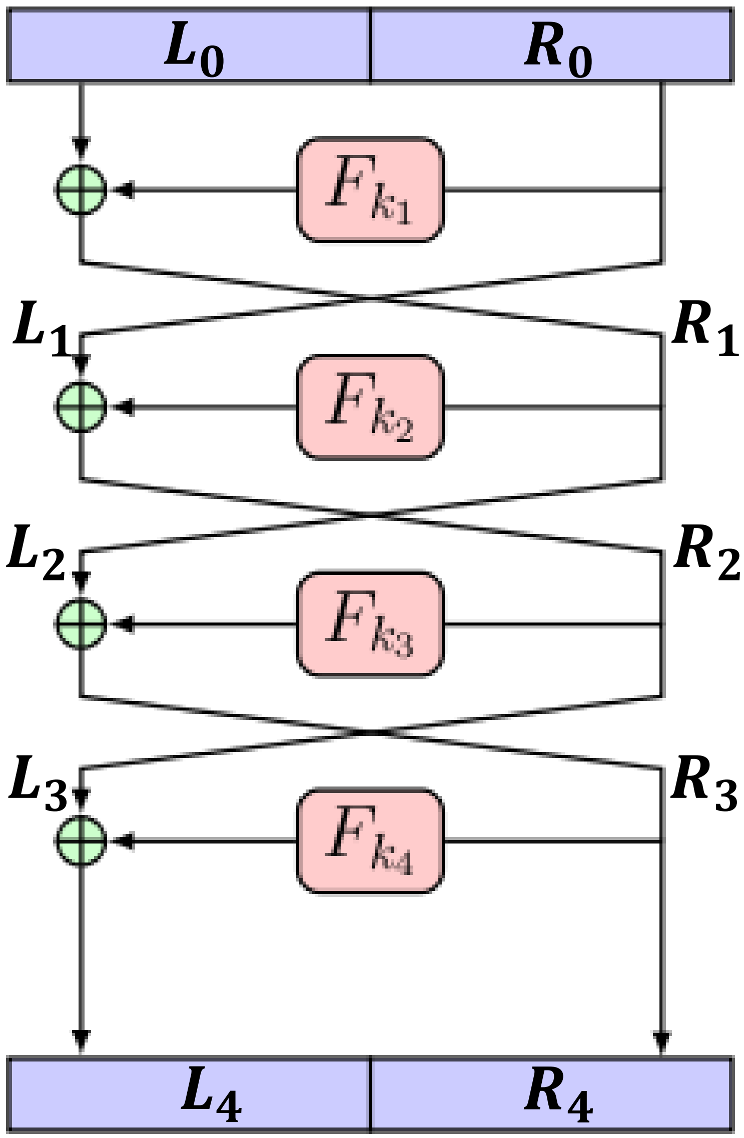
- **"Tiny" domains $\mathbb{Z}_M$:** spending $O(M)$ time\space is feasible
  - Using card shuffles [Dur`98,FY`38,Knu`69,MO`63,San`98]
  - Using block ciphers [BR`02]
- **"Small" domains $\mathbb{Z}_M$ or $\{\mathbf{0},\mathbf{1},\ldots,\boldsymbol{m}-\mathbf{1}\}^{\boldsymbol{n}}$:**

  $M, m^n \leq$ domain of underlying block cipher
  - Based on Feistel networks for
    - $\mathbb{Z}_{ab}$ [BR`02,BRRs`09]
    - $\{0,1\}^n$ [Fei`74,AB`96,Luc`96,SK`96]
    - $\{0,1,\ldots,m-1\}^n$ [BRS`10,BPS`10])
  - Based on card shuffling for $\mathbb{Z}_M$
    - Obtained as special case of Feistel network, or inefficient [Tho`73,GP`07]
- **"Huge" domains $\{\mathbf{0},\mathbf{1}\}^{\boldsymbol{n}}$:**

  $2^n >$ domain of underlying block cipher
  - Constructions based on block ciphers, e.g., [ZMI`89,Hal`04,HR`04,MF`07,CS`08,Sar`08,SAR`11]

Feistel-Based Integer FPEs

Feistel Networks [Smi`71,Fei`74,FNS`75]

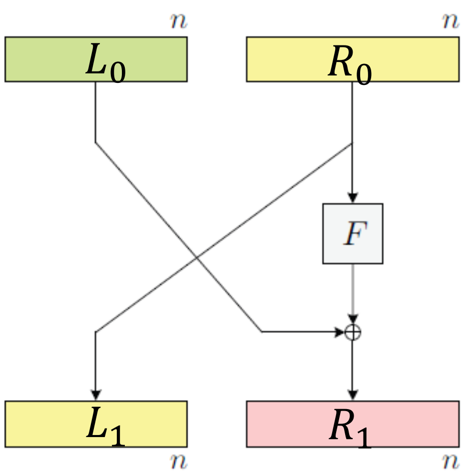


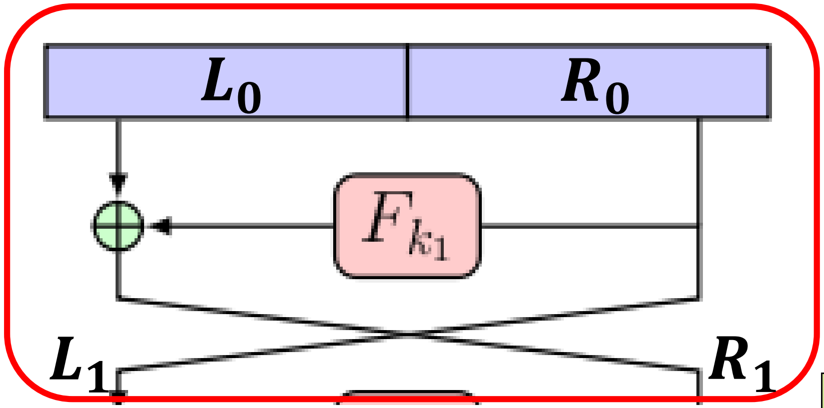
m n

Feistel Networks [Smi`71,Fei`74,FNS`75]

Balanced:

$|L_i| = |R_i|$

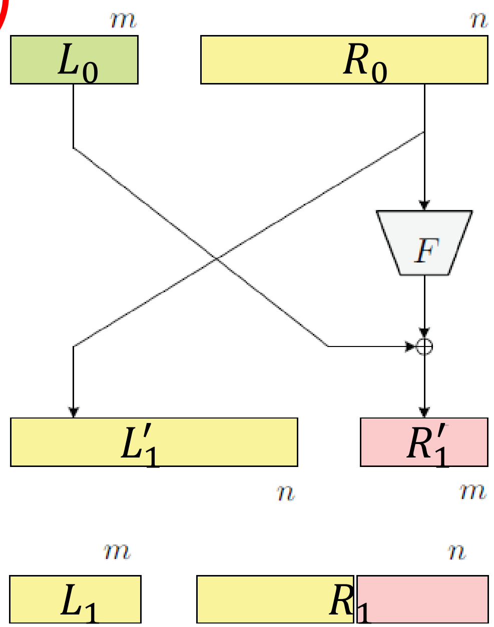


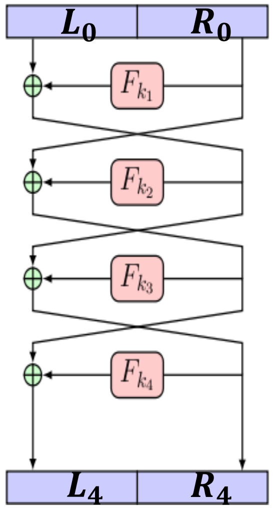


$$R_1 = F_{k_1}(R_0) \oplus L_0$$
$$L_1 = R_0$$

Unbalanced:

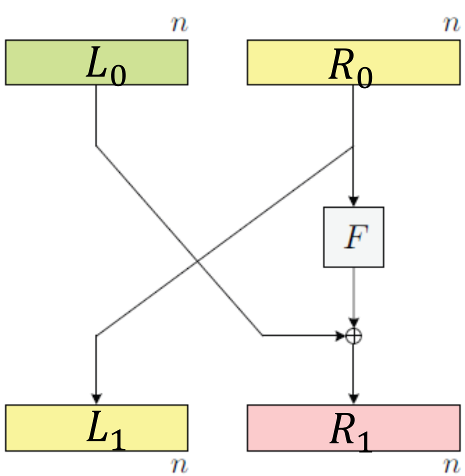
$|L_i| \neq |R_i|$



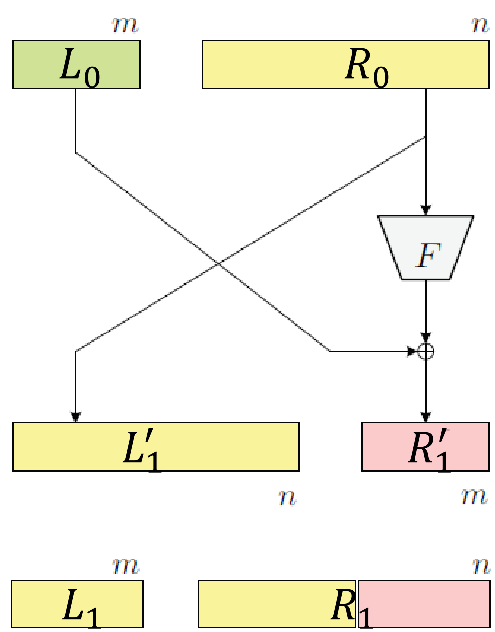


Feistel Networks (2)

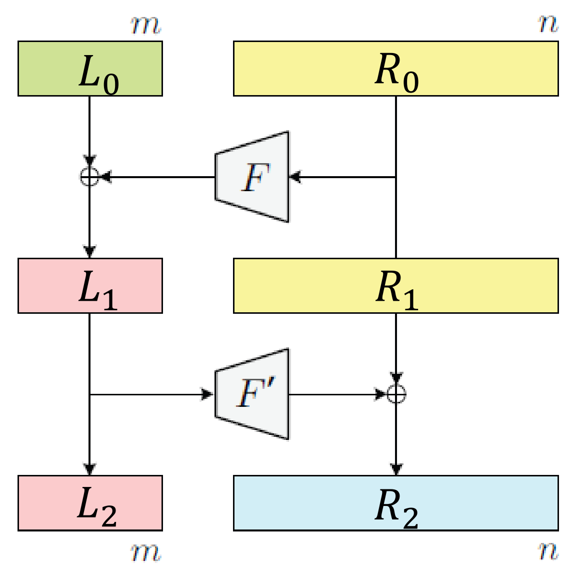
Balanced:
$|L_i| = |R_i|$



Unbalanced:
$|L_i| \neq |R_i|$



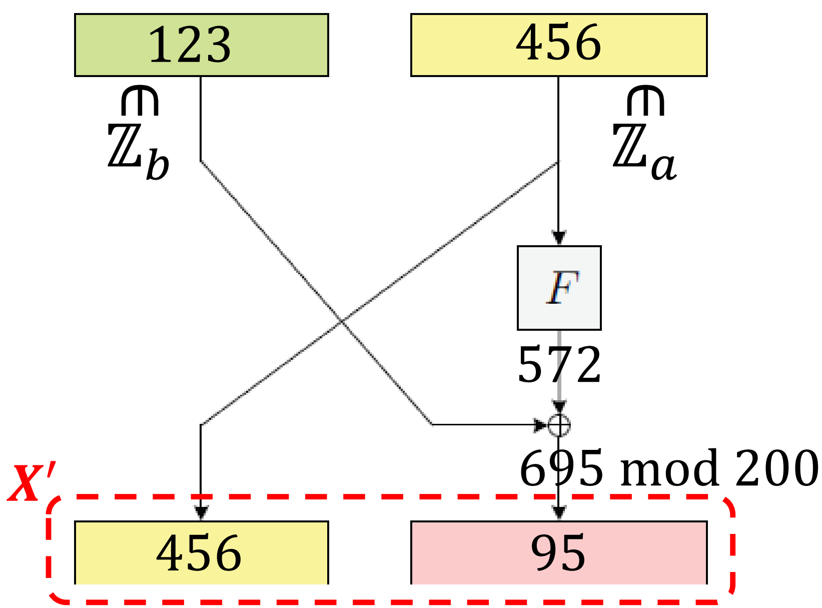
Alternating:
$|L_i| \neq |R_i|$



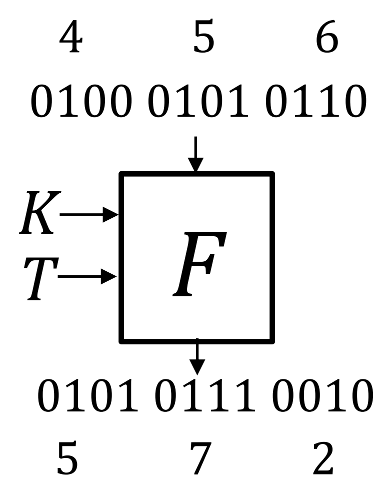
Generalized Feistel Networks

- Classic Feistel networks defined over bit strings
- First generalized to integral domains $\mathbb{Z}_{ab}$ by [BR`02]
  - Used alternating Feistel
- Tweakable Feistel for $\mathbb{Z}_{ab}$ described in [BRRS`09] (**FE1** and **FE2**)
  - Tweakable round function F
  - Tweak of F includes all public info (round #, provided tweak, format)
  - Use either alternating of unbalanced Feistel
- Operations computed modulo b
- **Example:** $a = 500, b = 200$ (generally, $b \leq a$), input $\boxed{61956}$

$$61956 = 123 \cdot a + 456$$



$$X' = 456 \cdot b + 95$$

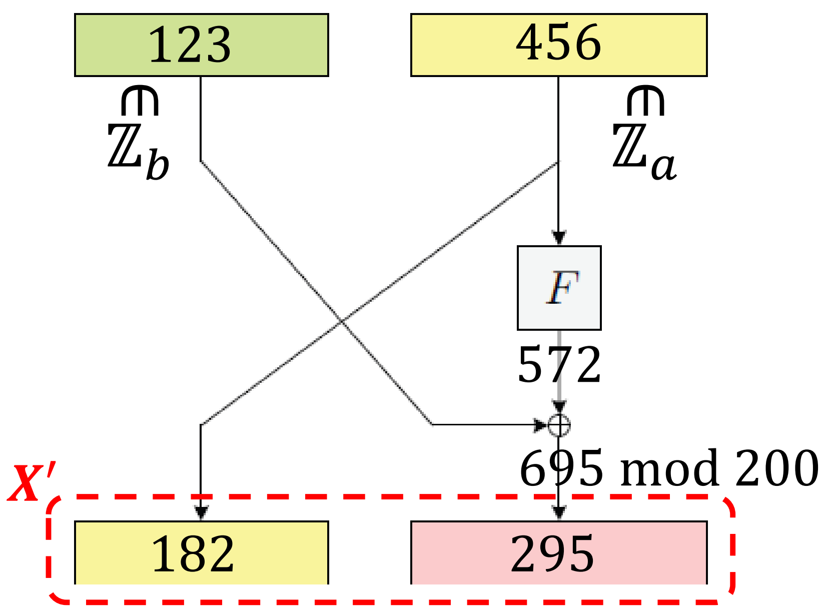


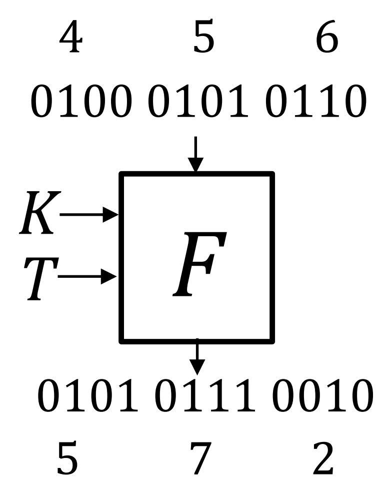
Generalized Feistel Networks

- Classic Feistel networks defined over bit strings
- First generalized to integral domains $\mathbb{Z}_{ab}$ by [BR`02]
  - Used alternating Feistel
- Tweakable Feistel for $\mathbb{Z}_{ab}$ described in [BRRS`09] (**FE1** and **FE2**)
  - Tweakable round function F
  - Tweak of F includes all public info (round #, provided tweak, format)
  - Use either alternating of unbalanced Feistel
- Operations computed modulo b

$$61956 = 123 \cdot a + 456$$

- **Example:** $a = 500, b = 200$ (generally, $b \leq a$), input $\boxed{61956}$

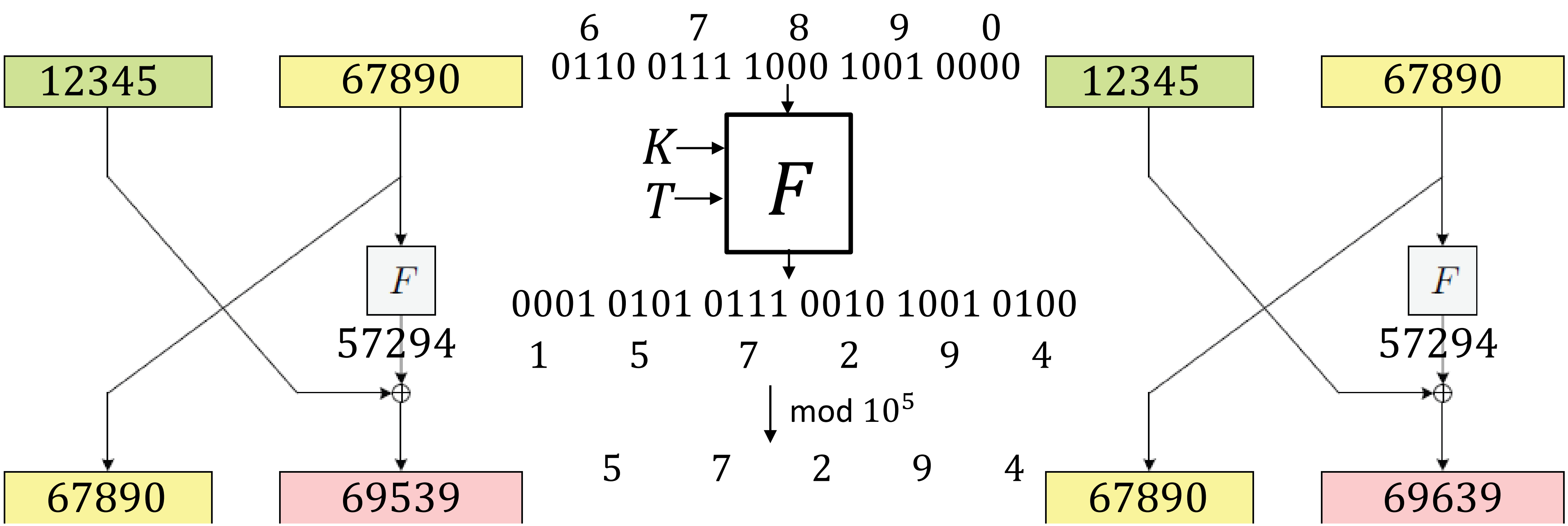




$$X' = 456 \cdot b + 95$$
$$= 182 \cdot a + 295$$

Generalized Feistel Networks (2)

- Feistel for $\mathbb{Z}_M, M = ab$: given format size M, requires factoring M
  - Highly inefficient for large M!
- Can we avoid factoring?
- Feistel for $\{0,1,\ldots,m-1\}^n$ for $n, m \in \mathbb{N}$ [BRS`10,BPS`10]
  - Operations computed coordinate-wise or block-wise (mod $m^{|R|}$)
- **Example:** $m = n = 10$, $|L| = |R| = 5$, input 1234567890



coordinate-wise: mod 10

block-wise: mod 10^5

Generalized Feistel Networks (3)

- Feistel for $\mathbb{Z}_M, M = ab$: given format size M, requires factoring M
  - Highly inefficient for large M!
- Can we avoid factoring?
- Feistel for $\{0,1,\dots,m-1\}^n$ for $n, m \in \mathbb{N}$ [BRS`10,BPS`10]
  - Operations computed coordinate-wise of block-wise (mod $m^{|R|}$)
- **Efficiency:** no factoring
- Generalized Feistel networks ⇒ int-FPE for domains $\mathbb{Z}_M, M = ab$ and $\{0,1,\dots,m-1\}^n$
  - Main issue: choosing network parameters
    - Round function, # rounds, operation and network type…

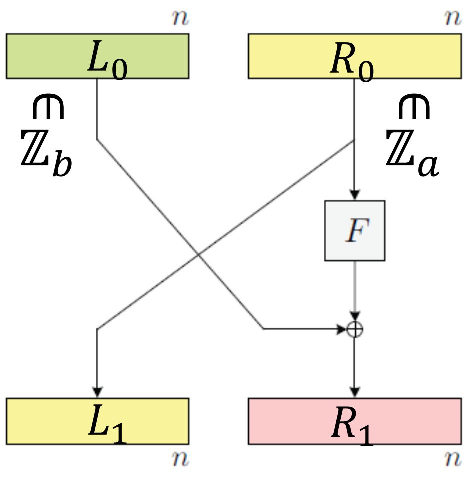
Security of Feistel Networks

- Main approach for block cipher constructions
- Intensively studied for over 3 decades
  - Security proofs (e.g., [LR`88, Mau`92, NR`97, Vau`98, Pat`98, MP`03, Pat`03, MRS`09, Pat`10, LP`12])
  - Attacks (e.g., [Pat`01, Pat`04, PNB`06, PNB`07])
- **Security measure:** PRP or strong-PRP security (random round functions)
  - Also: attacks exploiting round function structure, or allowing adversary oracle access to round functions
- **Parameters of interest:**
  - # queries
  - Running time
- Parameters of interest influence choice of round number
- Huge gap between security guarantees and known attacks
  - In part due to *highly* inefficient information theoretic attacks
  - Major open problem!

Security of Generalized Feistel Networks

- Generalized Feistel (almost) as secure as standard Feistel
  - But not as well studied
- **Standard Feistel:** security follows from pseudo-randomness of F
- **Generalized Feistel:**
  - Output z of F pseudorandom in $2^n > |\mathbb{Z}_a|$
  - Output used in generalized Feistel is $\boldsymbol{z} \mathbf{\ mod\ } \boldsymbol{a}$
- Mod operation preserves pseudo-randomness [BRRS`09]:

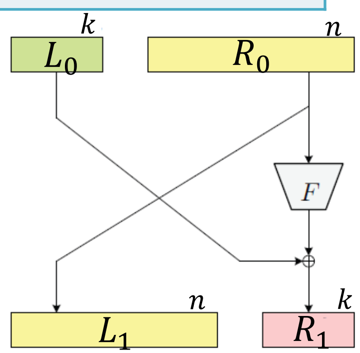
$(z \bmod a)$ is $\frac{a}{2^{n-2}}$-statistically close to random $z' \in_R \mathbb{Z}_a$



Security of Generalized Feistel Networks (2)

Domain	Network type	Number queries q	Number rounds r
$\mathbb{Z}_M, M = ab$ $a > b$	unbalanced, contracting	$q \approx a^{1-\epsilon}$	$r = O\left(\frac{\lceil \log_b a \rceil}{\epsilon}\right)$
$\mathbb{Z}_M, M = ab$ $a \leq b$	unbalanced, expanding	$q \approx a^{1-\epsilon}$	$r = O\left(\frac{\lceil \log_a b \rceil}{\epsilon}\right)$
$\{0,1,\ldots,m-1\}^N$ $N = 2n$	balanced	$q \approx m^{n(1-\epsilon)}$	$r = O\left(\frac{1}{\epsilon}\right)$
$\{0,1,\ldots,m-1\}^N$ $N = n + k, n > k$	unbalanced, contracting*	$q \approx m^{n(1-\epsilon)}$	$r = O\left(\frac{n}{\epsilon k}\right)$
$\{0,1,\ldots,m-1\}^N$ $N = n + k, n \leq k$	unbalanced, expanding**	$q \approx m^{n(1-\epsilon)}$	$r = O\left(\frac{n}{\epsilon k}\right)$

Security bounds from [HR`10]
*bound improves with imbalance
** bound deteriorates with imbalance



Int-FPE (Soon To Be*) Standards

- **Recall:** generalized Feistel networks ⇒ int-FPE for domains $\mathbb{Z}_M, M = ab$ and $\{0,1,\ldots,m-1\}^n$
  - Main issue: choosing network parameters
    - Round function, # rounds, operation and network type…
- Two Feistel-based int-FPE schemes for $\{0,1,\ldots,m-1\}^n$ currently under NIST consideration:
  - FFX [BRS`10]
  - BPS [BPS`10]

FFX [BRS`10]

- **Highly parameterized:**
  - **Format structure:** m, n $(100 \leq m^n \leq 2^{128})$
    - No mode of operation
  - **Round function** F (and key space)
    - E.g., CBC-MAC, CMAC, HMAC
  - **# rounds, tweak space**
    - Tweak should include all public info
  - **Network structure:** alternating\unbalanced; block\coordinate-wise operation; imbalance factor
- **Security goal:** strong-PRP against $m^n - 2$ queries in time < exhaustive key search
  - "Suggested" (conservative) # rounds based on known results
  - Shorter input $\Rightarrow$ more rounds
- Variants for useful domains:
  - FFX-A2: bit strings, lengths 8-128 (12-36 rounds)
  - FFX-A10: decimal strings, lengths 4-36 (12-24 rounds)

BPS [BPS`10]

- **Construction parameters:**
  - **Format structure:** m, n for any $m, n \in \mathbb{N}$
    - Mode of operation for long messages (# blocks $\leq 2^{16}$)
  - **Round function** F (and key space)
    - E.g., AES, TDES, SHA-2
  - **# rounds** (even $\geq$ 8)
- **Construction constants:**
  - **Tweak space:** $\{0,1\}^{64}$
    - Tweak should include all public info (long tweaks hashed)
  - **Network structure:**
    - Alternating, maximally balanced
    - Coordinate-wise operation in Feistel, block-wise in mode of operation
  - **Mode of operation:** CBC (block size = $2 \times \log_m \left\lfloor 2^{\text{input length to } F \text{ minus } 32} \right\rfloor$)
- **Security goal:** PRP-security against m^n queries (no time bound)
  - "Suggested" # rounds based on known attacks and security analysis
  - # rounds fixed to 8 for all input lengths

Int-FPE (Soon To Be*) Standards (2)

	FFX [BRS`10]	BPS [BPS`10]
Domain size $M = m^n$	$M \leq 2^{128}$ (*)	arbitrary
Security goal	strong-PRP $q = m^n - 2$ $T <$ exhaustive key search	PRP $q = m^n$ no time bound
Efficiency	more rounds (conservative bounds) more calls to F (defeat strong attacks)	8 rounds (less conservative) less calls to F (strong attacks outside of security goal)
Suggested F	CBC-MAC, CMAC, HMAC	AES, TDES, SHA-2
Flexibility (tweak space and network structure)	✓ user defined	✗ fixed

Format-Preserving Encryption Part III: General Formats

Post-Lunch Recap

- Format Preserving Encryption (FPE):
  - Preserves message format
  - Tweakable, deterministic, private key
  - Useful for:
    - Storing data at remote servers
    - Running applications for (unencrypted data) on encrypted data
- Hierarchy of security notions: $PRP \Leftrightarrow SPI \Rightarrow MP \Rightarrow MR$
- Int-FPE based on generalized Feistel networks
  - For $\mathbb{Z}_M, M = ab$
  - For $\{0,1, \ldots, m-1\}^n$

Session III: Outline

- Techniques for general-format FPE
- Natural FPE construction: analysis and insecurities
- FPE constructions for general formats:
  - From regular expressions and relaxed ranking
  - From bottom-up framework and (standard) ranking

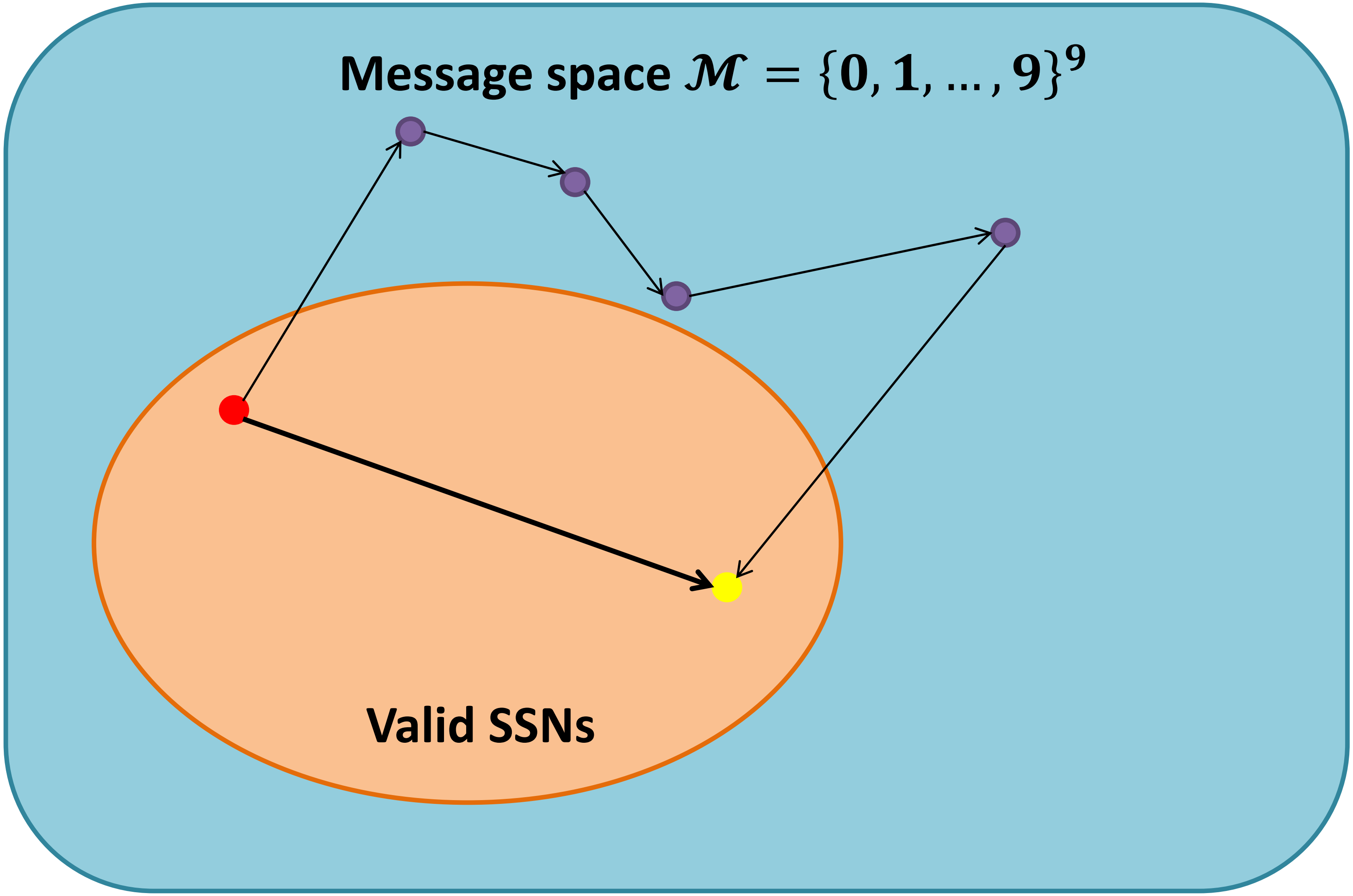
Techniques for General-Format FPE (Part 1)

- How to encrypt social security numbers (SSNs)?
  - Subset of $\{0,1,\dots,9\}^9$
  - Additional constraints
- We have FPE for $\mathcal{M} = \{0,1,\dots,9\}^9$
- Can get FPE for SSNs from FPE for $\mathcal{M}$:
- Use cycle walking [SO`98,BR`02]

"if at first you don't succeed, pick yourself up and try again"

  - Use "standard" FPE for $\mathcal{M} = \{0,1,\dots,9\}^9$
  - Repeat until ciphertext is valid SSN

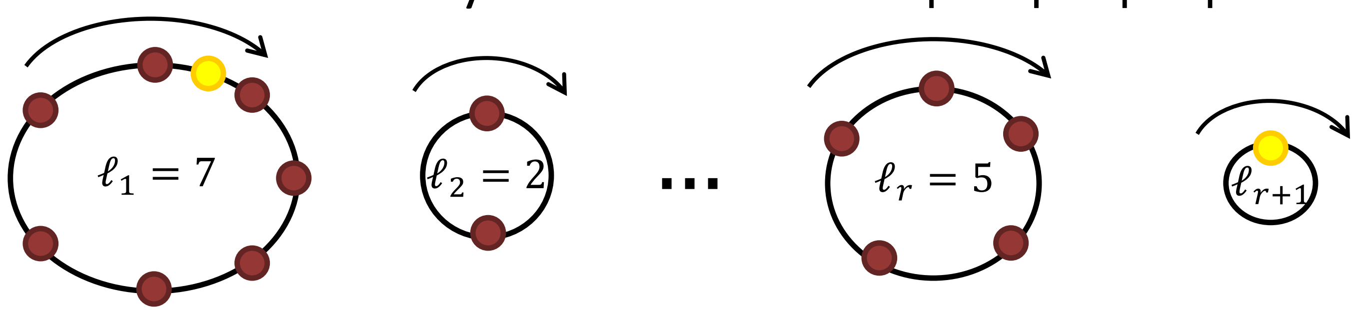
Cycle Walking



Cycle Walking: Security Analysis

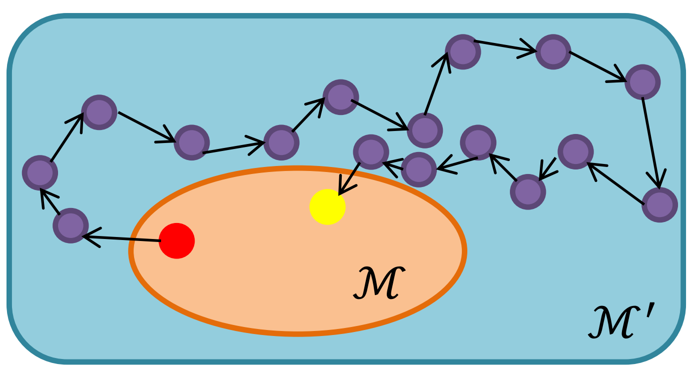
- **Want:** FPE for $\mathcal{M}$
  - Encryption "looks like" random permutation on $\mathcal{M}$
- **Have:** ***ideal*** FPE for $\mathcal{M}'$, $\mathcal{M} \subseteq \mathcal{M}'$ with encryption E'_K
  - Ideal FPE: each permutation on $\mathcal{M}'$ induced by *single* key
- $E_K^{\mathcal{CW}}$:= apply cycle walking to E'_K until ciphertext in $\mathcal{M}$
- For random K, $E_K^{\mathcal{CW}}$ is ***random permutation*** on $\mathcal{M}$ [BR`02]
  - Enough to show all permutations π on $\mathcal{M}$ obtained by same number of keys K
  - Adding one element $x \in \mathcal{M}' \setminus \mathcal{M}$ to π: $\Sigma_{i=1}^{r} \ell_i + 1$ options
    - $|\mathcal{M}| + 1$ options of adding x
  - General case follows by induction on $k = |\mathcal{M}'| - |\mathcal{M}|$

π



Cycle Walking: Efficiency Analysis

- $\mathcal{M} \subseteq \mathcal{M}'$
- $E_K^{\mathcal{CW}}$ for $\mathcal{M}$ obtained from cycle walking on E'_K for $\mathcal{M}'$
- Single $E_K^{\mathcal{CW}}$ call requires ***on average*** $\frac{|\mathcal{M}|}{|\mathcal{M}'|}$ calls to E'_K
  - No timing attacks due to repeated encryption [BRRS`09] (cycle length independent of plaintext)
- No bound on actual efficiency
  - But… for “good” FPE (=like PRP) on $\mathcal{M}'$: bound close to average



Cycle Walking: Summary

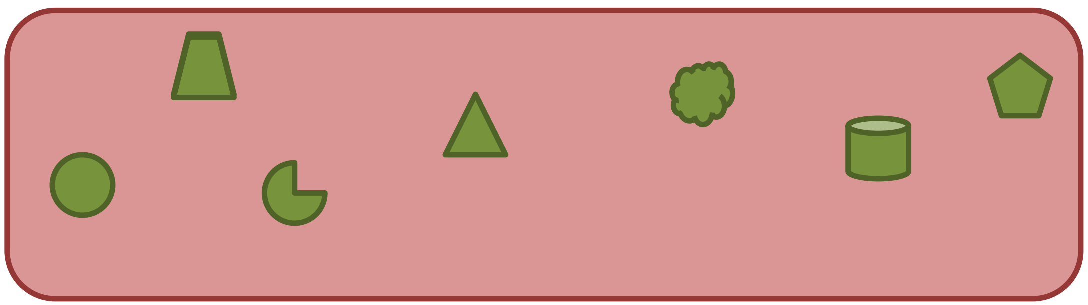
- $\mathcal{M} \subseteq \mathcal{M}'$
- $E_K^{\mathcal{CW}}$ for $\mathcal{M}$ obtained from cycle walking on E'_K for $\mathcal{M}'$
- **Cons:** Efficiency loss
  - single $E_K^{\mathcal{CW}}$ call = multiple E'_K calls
  - Average, not worst case, bound
  - Even average bound (typically ≈ 2) sometimes too expensive
- **Pros:** can use known schemes (e.g., Feistel)
  - Inherit security

But… can also be obtained ***without*** cycle walking

- Would like to avoid when possible
  - E.g., design dedicated int-FPE schemes

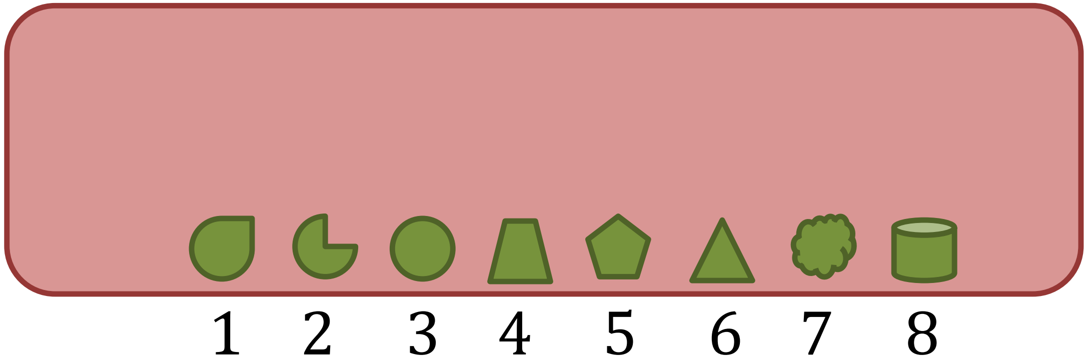
Techniques for General-Format FPE (Part 2)

- Rank-then-Encipher (RtE) [BRRS`09]: **general-format** FPEs from **int**-FPE
  - Order $\mathcal{M}$ arbitrarily: $\mathbf{rank}: \mathcal{M} \to \{1, .., M\}$



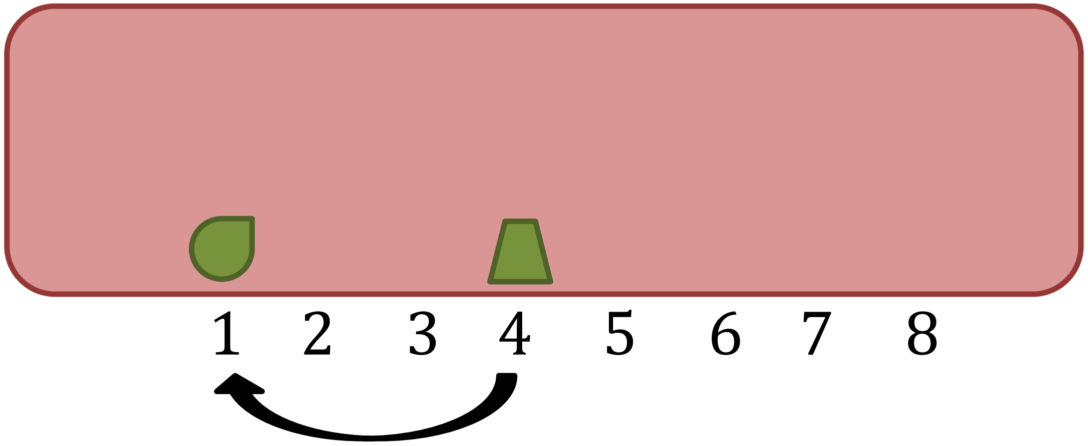
Techniques for General-Format FPE (Part 2)

- Rank-then-Encipher (RtE) [BRRS`09]: **general-format** FPEs from **int**-FPE
  - Order $\mathcal{M}$ arbitrarily: $\mathbf{rank}: \mathcal{M} \rightarrow \{1, .., M\}$



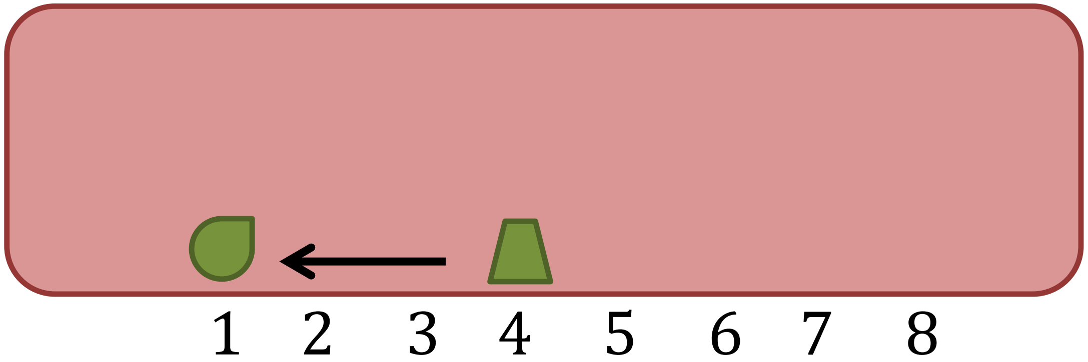
Techniques for General-Format FPE (Part 2)

- Rank-then-Encipher (RtE) [BRRS`09]: **general-format** FPEs from **int**-FPE
  - Order $\mathcal{M}$ arbitrarily: $\mathbf{rank}$: $\mathcal{M} \to \{1, .., M\}$
  - To encrypt message m:
    - **Rank *m*:** $i = \mathrm{rank}(m)$
    - **Encipher *i*:** $j = intE(K, i)$
    - **Unrank *j*:** $c = \mathrm{rank}^{-1}(j)$



Techniques for General-Format FPE (Part 2)

- Rank-then-Encipher (RtE) [BRRS`09]: **general-format** FPEs from **int**-FPE
  - Order $\mathcal{M}$ arbitrarily: $\mathbf{rank}$: $\mathcal{M} \rightarrow \{1, .., M\}$
  - To encrypt message m:
    - **Rank *m*:** $i = \mathrm{rank}(m)$
    - **Encipher *i*:** $j = intE(K, i)$
    - **Unrank *j*:** $c = \mathrm{rank}^{-1}(j)$



Techniques for General-Format FPE (Part 2)

- Rank-then-Encipher (RtE) [BRRS`09]: **general-format** FPEs from **int**-FPE
  - Order $\mathcal{M}$ arbitrarily: $\mathbf{rank}$: $\mathcal{M} \rightarrow \{1, .., M\}$
  - To encrypt message m:
    - **Rank $\boldsymbol{m}$:** $i = \mathrm{rank}(m)$
    - **Encipher $\boldsymbol{i}$:** $j = intE(K, i)$
    - **Unrank $\boldsymbol{j}$:** $c = \mathrm{rank}^{-1}(j)$
- Security: from security of int-FPE
  - rank not meant to, and does not, add security
- Efficiency: only if $\mathrm{rank}, \mathrm{unrank}$ are efficient
- **Main challenge:** design efficient ranking procedures
  - "Meta" technique for regular languages [BRRS`09]

Constructing General-Format FPE

- **Goal:** design FPE supporting general formats
  - E.g., SSNs, CCNS, dates, names, addresses…
- **Main tool:** RtE Technique
- **Main challenges:**
  - Designing efficient ranking procedures
  - Representing formats

Simplification-Based FPE [MYHC`11,MSP`11]

- Represent formats as union of simpler sub-formats
  - Messages interpreted as strings
  - $\mathcal{M}$ divided into subsets $\mathcal{M}_1, \ldots, \mathcal{M}_k$ defined by
    - Length
    - Index-specific character sets
- Encrypt each $\mathcal{M}_i$ ***separately*** using Rank-then-Encipher
  - Ranking computed using generalizes decimal counting method

$\mathcal{F}_{name}$: format of valid names

Name: 1-4 space-separated words

Word: upper case letter followed by 1-15 lower case letters

Subsets:

$\mathcal{M}_1$ contains Al

$\mathcal{M}_2$ contains Tal

...

$\mathcal{M}_{15}$ contains Muthuramakrishna

$\mathcal{M}_{16}$ contains El Al

$\boxed{\mathcal{M}_5 \text{ contains Migel}}$

Simplification-Based FPE: Security Concerns

- **The problem:** encryption preserves ***message-specific*** properties
  - Length and character type at each location
  - John Doe can encrypt Jane Lee but not Johnnie Smith
- Scheme insecure both in theory and practice [WRB`15]
  - **Practice:** experimental results
    - Ciphertext usually completely reveals plaintext
    - Worse than not encrypting at all…
  - **Theory:** scheme is MR (message recovery) insecure
    - Implies insecurity according to all FPE security notions

Message Recovery (MR)

ideal

challenger **adversary**

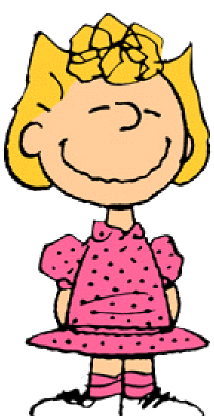
$\xleftarrow{\mathcal{D}}$ Same as in real world

$(T^*, N^*, m^*) \leftarrow \mathcal{D}$

$\xrightarrow{T^*, N^*}$

q $\xleftarrow{m?}$

$\xrightarrow{\text{Yes\textbackslash No}}$



$\boldsymbol{m'}$

real

challenger **adversary**

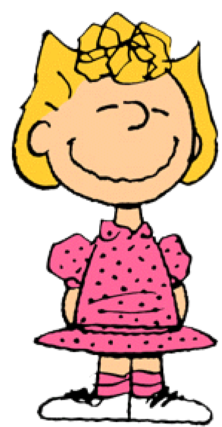
$\xleftarrow{\mathcal{D}}$

$K \leftarrow KeyGen(1^n)$
$(T^*, N^*, m^*) \leftarrow \mathcal{D}$
$c^* \leftarrow E_K^{T^*, N^*}(m^*)$

$\xrightarrow{T^*, N^*, c^*}$

q $\xleftarrow{(T, N, m)}$

$\xrightarrow{E_K^{T,N}(m)}$



$\boldsymbol{m'}$

Message Recovery (MR)

ideal

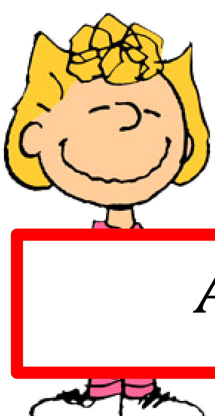
challenger | **adversary**

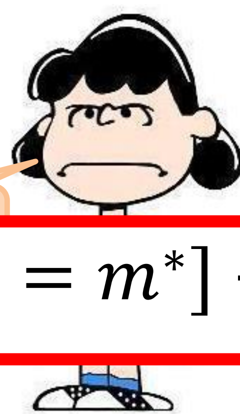
$\xleftarrow{\mathcal{D}}$ Same as in real world

$(T^*, N^*, m^*) \leftarrow \mathcal{D}$

$\xrightarrow{T^*, N^*}$

q ($\xleftarrow{m?}$ $\xrightarrow{\text{Yes\\No}}$)





real

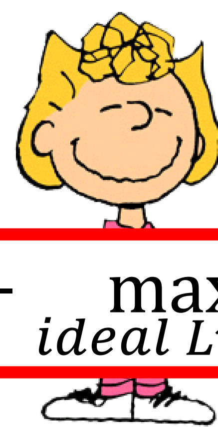
challenger | **adversary**

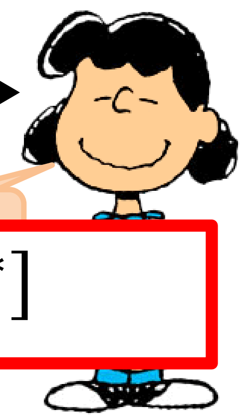
$\xleftarrow{\mathcal{D}}$

$K \leftarrow KeyGen(1^n)$
$(T^*, N^*, m^*) \leftarrow \mathcal{D}$
$c^* \leftarrow E_K^{T^*,N^*}(m^*)$

$\xrightarrow{T^*, N^*, c^*}$

q ($\xleftarrow{(T,N,m)}$ $\xrightarrow{E_K^{T,N}(m)}$)





$$\boxed{Adv_{\mathcal{A}}^{MR} = \Pr_{real}[m' = m^*] - \max_{ideal\ Lucy} \Pr[m' = m^*]}$$

Simplification-Based FPE: MR-insecurity

Warm-up example: attacking sparse formats

- $\mathcal{M} = \{m_1, \ldots, m_n\}$, $|m_i|$'s are unique
- Ciphertext reveals message length ($\Rightarrow$ reveals message)
  - In this case: $E_K^{T,N}(m) = m$
- The adversary $\mathcal{A}$:
  - Picks $\mathcal{D} =$ uniform distribution over $\mathcal{M}$
  - Given c^*, guesses c^*
  - Makes no queries!
  - $\Pr[\mathcal{A} \text{ wins}] = 1$
- Best ideal-world adversarial strategy: random guess
  - $\Pr[ideal\mathcal{A} \text{ wins}] = \frac{1}{|\mathcal{M}|}$
- Adversarial advantage: $1 - \frac{1}{|\mathcal{M}|} \rightarrow^{|\mathcal{M}| \rightarrow \infty} 1$

General case: "sparsify" the format

- For every possible length, $\mathcal{A}$ selects a single message
- Picks uniforms distribution over these messages

Simplification-Based FPE: Take-Home Message

- "Natural" method of representing formats is insecure
- **Reason:** encryption preserves ***message-specific*** properties

FPE "wish list"

- **Functionality (and efficiency):**
  - *Simple* method of representing formats
  - *Efficient* rank, unrank procedures
    - In particular: minimize cycle walking
- **Security:** preserve *only format-specific* properties
  - Hide *all message-specific* properties

RtE-Based FPE for General Formats

- Two concurrent works [LDJRS`14,WRB`15], differ in focus and design
- **Focus:** Developer- or user-oriented
- **Design:** representing formats, ranking methods
  - Both schemes based on RtE (Rank-then-Encipher)
  - libFTE [LDJRS`14]:
    - Developer-oriented
    - Represent formats using regular expressions
    - Extend RtE method to allow efficient ranking
  - GFPE [WRB`15]:
    - User-oriented
    - Represent formats using bottom-up framework
    - Use standard RtE

libFTE [LDJRS`14]

- Library for format-preserving and format transforming encryption
- Developer-oriented: developer needed to…
  - Choose "right" scheme to use (using "Configuration assistant")
    - Several schemes (with different parameters) available
  - Define new formats
- **Structure:**
  - Represent formats with Regular Expressions (Regexes)
    - Expressions limited to lengths in range $\{n_{\min}, n_{\max}\}$
  - Ranking from automatons
  - Int-FPE using FFX-A2 (FFX over bit strings)
- **Main challenge:** efficient $\mathrm{rank}, \mathrm{unrank}$ algorithms

Ranking in libFTE

- Format represented as regular expressions (regexes)
  $\Rightarrow$ need a method of ranking regexes
- – **Ranking:** bijection from $\mathcal{M}$ to $\{1, \ldots, |\mathcal{M}|\}$
  - Only useful when bijection easy to compute
- – (Exact) ranking may be an overkill
- – Suffices to achieve a relaxed ranking notion
- – **Relaxed ranking** [LDJRS`14]:

$$\text{map } \mathcal{M} \text{ to } \{1, \ldots, M'\}, |\mathcal{M}| < M'$$

  - $\mathbf{r}\mathrm{rank}: \mathcal{M} \to \{1, \ldots, M'\}$ injective
  - $\mathrm{un}\mathbf{r}\mathrm{rank}: \{1, \ldots, M'\} \to \mathcal{M}$ surjective
  - For all $m \in \mathcal{M}$, $\mathrm{un}\mathbf{r}\mathrm{rank}\left(\mathbf{r}\mathrm{rank}(m)\right) = m$

Ranking in libFTE (2)

- Format represented as regular expressions (regexes)
  ⇒ need a method of ranking regexes

(Highly Informal) Automata Theory Crash Course

- Automatons, and regexes, used to represent sets $\mathcal{M}$
- Automatons are graphs: $m \in \mathcal{M}$ represented through paths in graph
  - **Deterministic** finite automaton (**DFA**)
  - **Nondeterministic** finite automaton (**NFA**)
- Regexes equivalent to automatons:
  - Regex-to-NFA transformation in **linear** time
  - Regex-to-DFA transformation in **exponential** time (this is tight!)

- "Meta" ranking technique from DFA [BRRS`09]
  - Order paths in DFA, map $m \in \mathcal{M}$ to index of corresponding path
  - **Too inefficient!**
- *Relaxed* ranking from NFA in polynomial time [LDJRS`14]
  - In NFA, (possibly) more than one path for $m \in \mathcal{M}$
  - Find one such path efficiently through implicit graph representation
- libFTE supports DFA-based ranking and NFA-based relaxed ranking

libFTE: Tools and Algorithms

- Configuration assistant helps developer choose appropriate scheme
  - Randomized\deterministic, DFA-based\NFA-based ranking…
- “Appropriate” schemes chosen according to input parameters
  - Format, memory threshold for encryption\ranking…
  - Assistant runs tests to evaluate time and memory performance
- Developer chooses preferred scheme from list

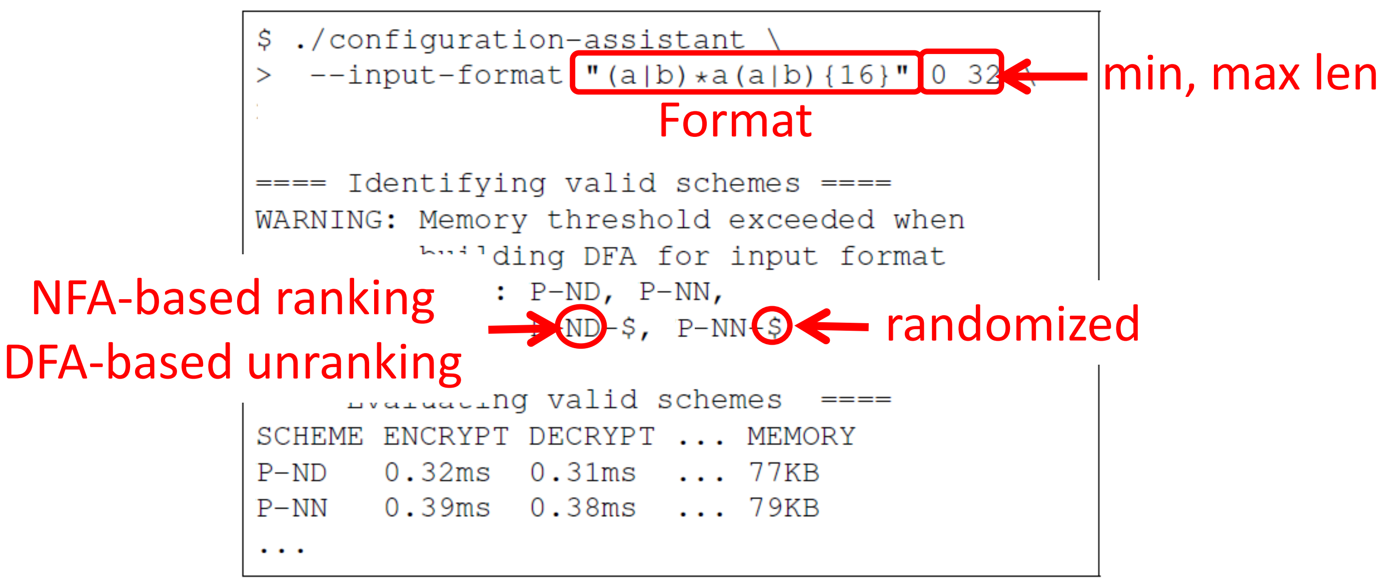
```
$ ./configuration-assistant \
>  --input-format "(a|b)*a(a|b){16}" 0 32 \


==== Identifying valid schemes ====
WARNING: Memory threshold exceeded when
         building DFA for input format
VALID SCHEMES: P-ND, P-NN,
               P-ND-$, P-NN-$

==== Evaluating valid schemes  ====
SCHEME ENCRYPT DECRYPT ... MEMORY
P-ND   0.32ms  0.31ms  ... 77KB
P-NN   0.39ms  0.38ms  ... 79KB
...
```

libFTE: Tools and Algorithms

- Configuration assistant helps developer choose appropriate scheme
  - Randomized\deterministic, DFA-based\NFA-based ranking…
- “Appropriate” schemes chosen according to input parameters
  - Format, memory threshold for encryption\ranking…
  - Assistant runs tests to evaluate time and memory performance
- Developer chooses preferred scheme from list



libFTE: Implementation Notes

- Encryption\decryption performance (runtime and memory consumption) determined by:
  - Chosen scheme (DFA or NFA-based ranking)
  - Chosen representation of format (!)
- Unclear how to find scheme + format representation optimizing performance
  - Even given performance estimate of assistant
  - Bad performance due to bad regex or bad format?

Scheme	Input/Output Format R	α	β	DFA/NFA States	Memory Required	Encrypt (ms)	Decrypt (ms)
P-DD	`(a|b)*`	0	32	2	4KB	0.18	0.18
	`(a|b)*a(a|b){16}`	16	32	131,073	266MB	0.25	0.21
	`(a|a|b){16}(a|b)*`	16	32	18	36KB	0.19	0.18
	`(a|b){1024}`	1,024	1,024	1,026	34MB	1.2	1.2
P-NN	`(a|b)*`	0	32	3	6KB	0.36	0.35
	`(a|b)*a(a|b){16}`	16	32	36	73KB	0.61	0.60
	`(a|a|b){16}(a|b)*`	16	32	51	103KB	1,340	1,340
	`(a|b){1024}`	1,024	1,024	2,049	68MB	6.6	6.6

libFTE: Implementation Notes

- Encryption\decryption performance (runtime and memory consumption) determined by:
  - Chosen scheme (DFA or NFA-based ranking)
  - Chosen representation of format (!)
- Unclear how to find scheme + format representation optimizing performance
  - Even given performance estimate of assistant
  - Bad performance due to bad regex or bad format?

Scheme	Input/Output Format			DFA/NFA States	Memory Required	Encrypt (ms)	Decrypt (ms)
	R	α	β				
P-DD	`(a|b)*`	0	32	2	4KB	0.18	0.18
	`(a|b)*a(a|b){16}`	16	32	131,073	266MB	0.25	0.21
	`(a|a|b){16}(a|b)*`	16	32	18	36KB	0.19	0.18
	`(a|b){1024}`	1,024	1,024	1,026	34MB	1.2	1.2
P-NN	`(a|b)*`	0	32	3	6KB	0.36	0.35
	`(a|b)*a(a|b){16}`	16	32	36	73KB	0.61	0.60
	`(a|a|b){16}(a|b)*`	16	32	51	103KB	1,340	1,340
	`(a|b){1024}`	1,024	1,024	2,049	68MB	6.6	6.6

GFPE [WRB`15]

- User-oriented
  - Part of a larger system used by the end-user
- Encryption\decryption using RtE, supporting int-FPE for:
  - $\mathbb{Z}_M$ (proven security, no cycle walking, inefficient for large formats)
  - $\{0,1,\dots,m-1\}^n$ (no security proofs, requires cycles walking, efficient for large formats)
- **Main challenge:** user-friendly format representation
- **Structure:** formats represented using bottom-up framework
  - **"Basic" building-blocks (primitives)**
    - Usually "rigid" formats
      - SSNs, CCNs, dates, set of valid strings, fixed-length strings…
    - Also "less rigid" formats (e.g., variable-length strings)
  - **Operations** used to construct complex formats
    - Operations preserve the "parsing property"

GFPE: Representing Formats

- **"Basic" building-blocks (primitives):**
  - $\mathcal{F}_{upper}$ = {A,B,…,Z}
  - $\mathcal{F}_{lower}$ = length-k lower-case letter strings, $1 \leq k \leq 15$
  - $\mathcal{F}_{ssn}$ =SSNs
- **Operations:**
  - *Concatenation:*
    - $\mathcal{F} = \mathcal{F}_1 \cdot \ldots \cdot \mathcal{F}_k$
      - Words: $\mathcal{F}_{word} = \mathcal{F}_{upper} \cdot \mathcal{F}_{lower}$
    - $\mathcal{F} = \mathcal{F}_1 \cdot d_1 \cdot \mathcal{F}_2 \cdot \ldots \cdot d_{n-1} \cdot \mathcal{F}_n$ ($d_1, \ldots, d_{n-1}$ are delimiters)
  - *Range:* $\mathcal{F} = (\mathcal{F}_1 \cdot d)^k, min \leq k \leq max$
    - Names: $\mathcal{F}_{name} = (\mathcal{F}_{word} \cdot space)^k$ for $1 \leq k \leq 4$
  - *Union:* $\mathcal{F} = \mathcal{F}_1 \cup \cdots \cup \mathcal{F}_k$
    - "Names or SSNs": $\mathcal{F} = \mathcal{F}_{name} \cup \mathcal{F}_{ssn}$

Example: Representing Addresses

name **house #** **street** **city** **zip** **state**

- $\mathcal{F}_{name} = (\mathcal{F}_{word} \cdot space)^k$ for $1 \leq k \leq 4$ (range)
- $\mathcal{F}_{num} = \{1, \ldots, 100\}$ (integral domain)
- $\mathcal{F}_{zip} = \{0,1, \ldots, 9\}^5$ (fixed length string)
- $\mathcal{F}_{state}$ = set of valid state abbreviations
- Valid addresses obtained through concatenation:

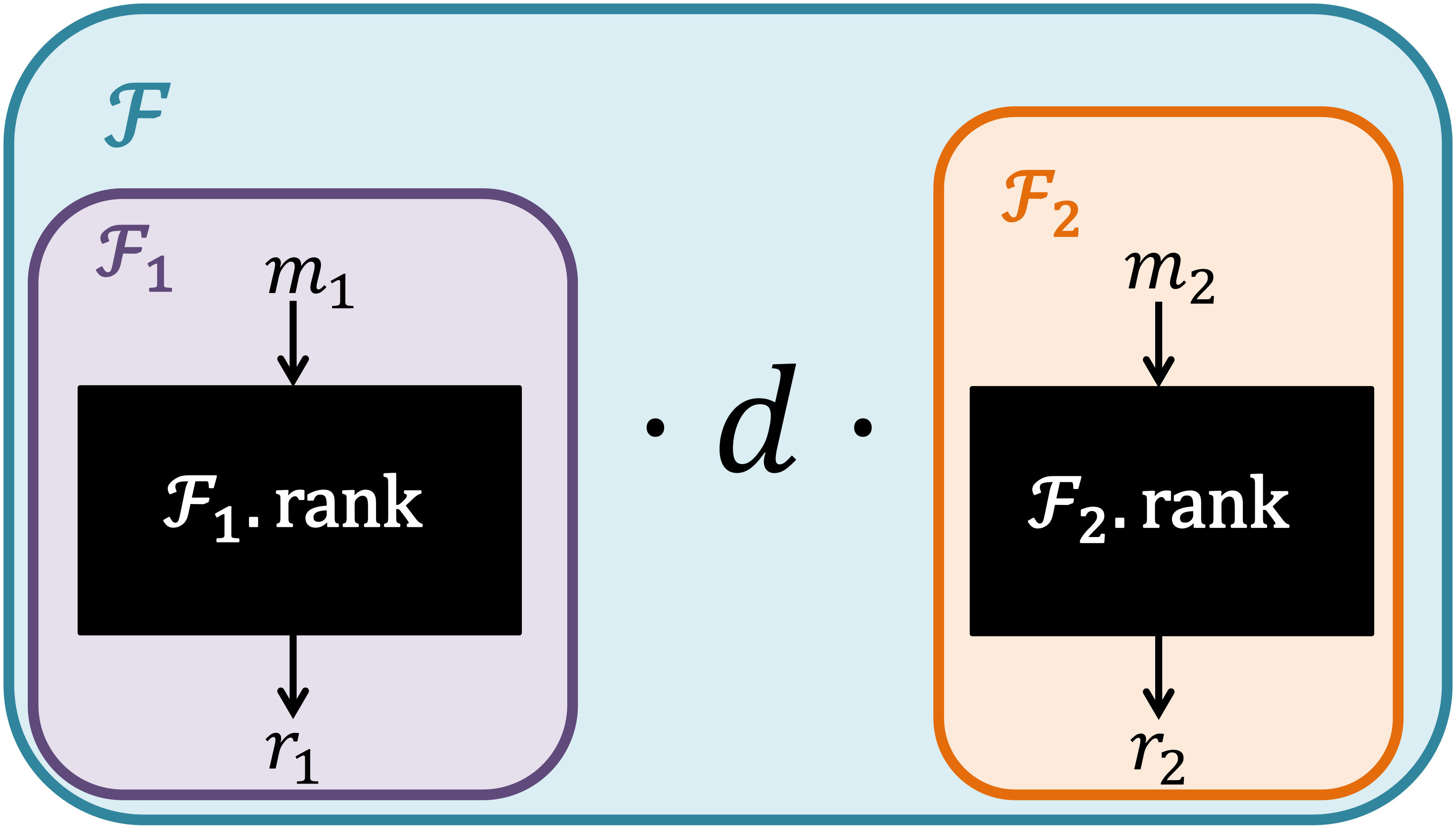
$$\mathcal{F}_{add} = \underbrace{\mathcal{F}_{name}}_{\text{name}} \cdot \underbrace{\mathcal{F}_{num}}_{\text{house \#}} \cdot \underbrace{\mathcal{F}_{name}}_{\text{street}} \cdot \underbrace{\mathcal{F}_{name}}_{\text{city}} \cdot \underbrace{\mathcal{F}_{zip}}_{\text{zip}} \cdot \underbrace{\mathcal{F}_{state}}_{\text{state}}$$

GFPE: Ranking

- Define ranking for **primitives** and **operations**
- Rank of compound formats computed top-down:
  - Parse string to components
  - Delegate substring ranking to format components
  - “Glue” ranks together using ranking for operations

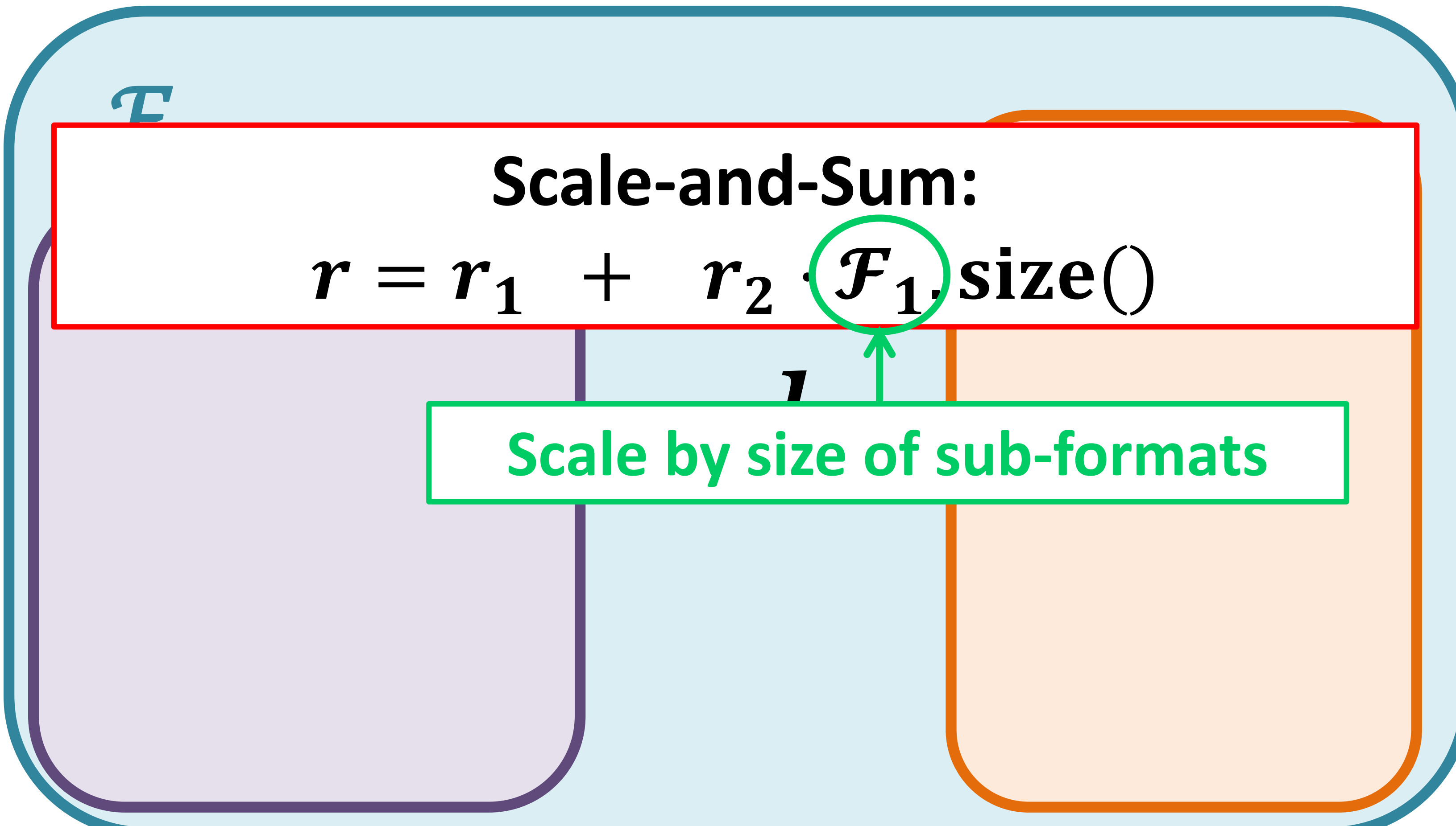
Example: Ranking Concatenation

$$\mathcal{F} = \mathcal{F}_1 \cdot d \cdot \mathcal{F}_2$$
$$m = m_1 \cdot d \cdot m_2$$



Example: Ranking Concatenation

$$\mathcal{F} = \mathcal{F}_1 \cdot d \cdot \mathcal{F}_2$$



GFPE: Supporting Large Formats

- Scheme supports int-FPEs for $\mathbb{Z}_M$ [BR`02,BRRS`09]
- Requires factoring M $\Rightarrow$ inefficient for large M's!
- **Supporting large formats:** keep formats small
  - Divide large formats
  - Minimize security loss by "hiding" ***message-specific*** properties:
    - Division according to ***format structure*** ← **Main challenge!**
    - Maximizing sub-format size
  - $maxSize$ determined by user-defined performance constraints

Example: Dividing Address Format

Name **house #** **street** **city** **zip** **state**

- Valid addresses obtained through concatenation:

$$\mathcal{F}_{add} = \boxed{\underbrace{\mathcal{F}_{name}}_{\text{name}} \cdot \underbrace{\mathcal{F}_{num}}_{\text{house \#}}} \cdot \boxed{\underbrace{\mathcal{F}_{name}}_{\text{street}} \cdot \underbrace{\mathcal{F}_{name}}_{\text{city}}} \cdot \boxed{\underbrace{\mathcal{F}_{zip}}_{\text{zip}} \cdot \underbrace{\mathcal{F}_{state}}_{\text{state}}}$$

- Jane Doe 23 | Delaford New York | 12345 NY
- Jane Doe 23 | Delaford Berkeley | 12345 CA
- Smaller $maxSize \Rightarrow$ further division
  - E.g., $\mathcal{F}_{name}$ divided according to number of words in name

GFPE: Supporting Large Formats (2)

- Scheme supports int-FPEs for $\mathbb{Z}_M$ [BR`02,BRRS`09]
- requires factoring M $\Rightarrow$ inefficient for large M's!
- **Supporting large formats:** keep formats small
  - Divide large formats
  - Minimize security loss by "hiding" ***message-specific*** properties:
    - $\boxed{\text{Division according to } \textbf{\textit{format structure}}}$ ← **Main challenge!**
    - Maximizing sub-format size
  - $maxSize$ determined by user-defined performance constraints
- Introduces complications in ranking and unranking
  - Generalize rank, unrank to lists of ranks
- Minimal security loss (according to experimental results)

FPEs for General Formats: Summary

	libFTE [LDJRS'14]	GFPE [WRB`15]
Underlying int-FPE	FFX (any FPE for $\{0,1,\ldots,m-1\}^n$)	FFX or FE1 (any FPE for $\mathbb{Z}_M$ or $\{0,1,\ldots,m-1\}^n$)
Designed for	developers	end users
Encryption type	deterministic\ randomized	deterministic
Format representation	regular expressions	bottom-up framework
Security guarantee	same as underlying int-FPE	same as underlying int-FPE
Encryption type	FPE + format transforming	FPE
Performance	depends on scheme and format representation	uniform
Expressiveness	not clear how to efficiently represent though computations	representation thorough computation is possible
Open source?	Yes: Python, C++, JavaScript	No

Format Preserving Encryption (FPE): Summary

- FPE preserves plaintext format under encryption
- Useful when adding encryption layer to existing schemes
- Int-FPEs based on generalized Feistel networks
  - Two constructions under NIST consideration for standardization
- Techniques for general-format FPE:
  - Cycle walking
  - Rank-then-Encipher (RtE)
- FPE for general formats constructed from int-FPE
  - Using RtE on top of int-FPE
  - Comparable security and performance


THANK YOU