

Day 4:

A Contemporary View of Symmetric Encryption

Thomas Ristenpart
University of Wisconsin

Some limitations in 90s and 00s viewpoint

- Encryption and authenticity as separate primitives leads to problems
 - Not enough guidance to implementers
 - Get the order wrong
 - Even with “right” generic composition, problems can arise
- Nonces can be confusing to developers
- Record layers are much more than “just” authenticated encryption

The expanding role of SE

- Recognition that the crypto should do more to help implementers
- Crypto needs to work around system constraints
 - Bad nonces should be protected against to best extent possible (defense-in-depth)
 - API should be simple
 - Systems often can't be redesigned (e.g., format-preservation, passwords)

Our game-plan today

We will build two widely needed primitives:

- **Authenticated encryption**
 - Contemporary viewpoint on AE
 - Two flavors (speed versus security trade-off)
- **Format-preserving encryption**
 - Used widely in industry for fixed-field encryption (credit card numbers, etc.)
 - Length-preserving encryption as special case
- **Time allowing:** other SE primitives such as message-locked encryption, honey encryption, password-based encryption

Lecture plan

1. Tweakable PRPs and shuffling
2. Nonce-based symmetric encryption
3. Format-preserving encryption
4. Further symmetric primitives

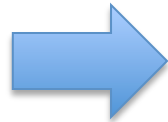
Tweakable PRPs and Shuffling

Thomas Ristenpart
University of Wisconsin

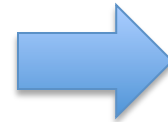
High level views of building symmetric encryption

Theory

One-way
function



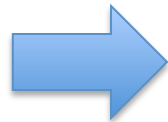
PRPs/PRFs



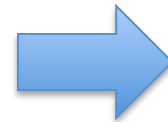
Encryption,
MACs

90s / 00s

Blockcipher
(assumed PRP)



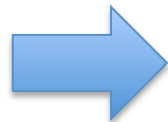
Encryption,
MACs



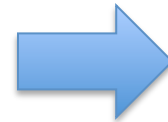
Authenticated-
Encryption

Contemporary

Blockcipher
(assume PRP)



Tweakable
PRPs



Authenticated-
Encryption,
Format-preserving
encryption

First up: the basic building blocks

- Recall PRFs and PRPs
- Feistel and Shuffling
 - Thorpe
- Tweakable block ciphers
 - Built from Thorp
- Mix-and-Cut
 - Pseudorandom separators
- Tweakable block ciphers from PRPs:
 - Simple LRW construction
 - $XE(X)$ constructions

Pseudorandom functions (PRFs)

- Keyed function family indistinguishable from random function

$$F : \{0, 1\}^k \times \{0, 1\}^* \rightarrow \{0, 1\}^n$$

- * is a bit misleading: usually fixed set, but we'll cheat for simplicity
- Examples:
 - CBC-mode using suitable block cipher
 - HMAC using suitable hash function
 - Theoretical constructions from any OWF

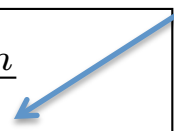
Pseudorandom functions (PRFs)

Can adversary distinguish between secret-keyed function and a random function?

Game PRF1_F $K \leftarrow_{\\$} \{0, 1\}^k$ $b' \leftarrow_{\\$} \mathcal{A}^{F_K}$ ret b'

Game PRF0_n $\rho \leftarrow_{\\$} \text{Func}(n)$ $b' \leftarrow_{\\$} \mathcal{A}^\rho$ ret b'

Pick a random
function with
range n bits



$$\mathbf{Adv}_F^{\text{prf}}(\mathcal{A}) = \left| \Pr \left[\text{PRF1}_F^{\mathcal{A}} \Rightarrow 1 \right] - \Pr \left[\text{PRF0}_n^{\mathcal{A}} \Rightarrow 1 \right] \right|$$

We will measure adversarial resources concretely : running time and queries
Proving “low” advantage for large resources translated as providing security

Pseudorandom permutations (PRPs)

- Keyed permutation family indistinguishable from random permutation

$$E : \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$$

$$\forall K, M . E^{-1}(K, E(K, M)) = M$$

- A **block cipher** is just a keyed family of permutations for which enciphering and deciphering are efficient
- Examples:
 - AES
 - 3DES
 - Luby-Rackoff using random functions

Pseudorandom permutations (PRPs)

Can adversary distinguish between secret-keyed permutation and random permutation?

Game PRP1_F

$K \leftarrow \$ \{0, 1\}^k$

$b' \leftarrow \$ \mathcal{A}^{E_K}$

ret b'

Game PRP0_n

$\pi \leftarrow \$ \text{Perm}(n)$

$b' \leftarrow \$ \mathcal{A}^\pi$

ret b'

Pick a random permutation with range n bits



$$\mathbf{Adv}_E^{\text{prp}}(\mathcal{A}) = \left| \Pr \left[\text{PRP1}_E^{\mathcal{A}} \Rightarrow 1 \right] - \Pr \left[\text{PRP0}_n^{\mathcal{A}} \Rightarrow 1 \right] \right|$$

We will measure concretely adversarial resources: running time and queries
Proving “low” advantage for large resources translated as providing security

PRP/PRF Switching lemma

- A PRP for largish n is already good PRF! (and visa-versa)

Theorem. *Fix n . Then for any adversary $\mathcal{A}$ making q oracle queries*

$$\left| \Pr \left[\text{PRP0}_n^{\mathcal{A}} \Rightarrow 1 \right] - \Pr \left[\text{PRF0}_n^{\mathcal{A}} \Rightarrow 1 \right] \right| \leq \frac{q^2}{2^n}$$

- Proof uses relatively straightforward birthday bounds
- Loosely speaking, means we can think of good block ciphers as either random functions or random permutations (when secretly keyed)

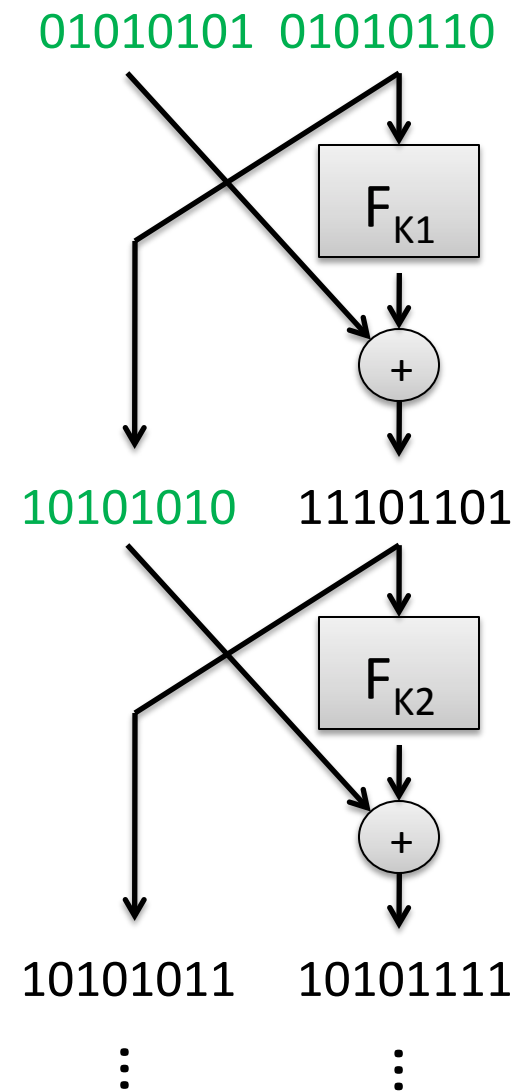
Revisiting a classical question

- How to build PRPs from PRFs?
- Why?
 - Cool theory (shuffling viewpoint on block ciphers)
 - Tweakable PRPs trivial given good PRP-from-PRF construction
 - Needed for format-preserving encryption since no good block ciphers of right block sizes

PRPs from PRFs

- Recall Feistel networks
 - Split input in half
 - Apply PRF to one half
 - XOR with left half
 - Drop right half down to left

This bound is not great. For $n = 128$ we have security only up to 2^{32} queries. Can we do better?

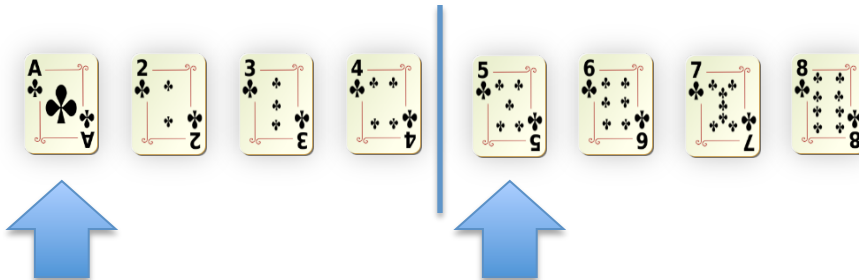
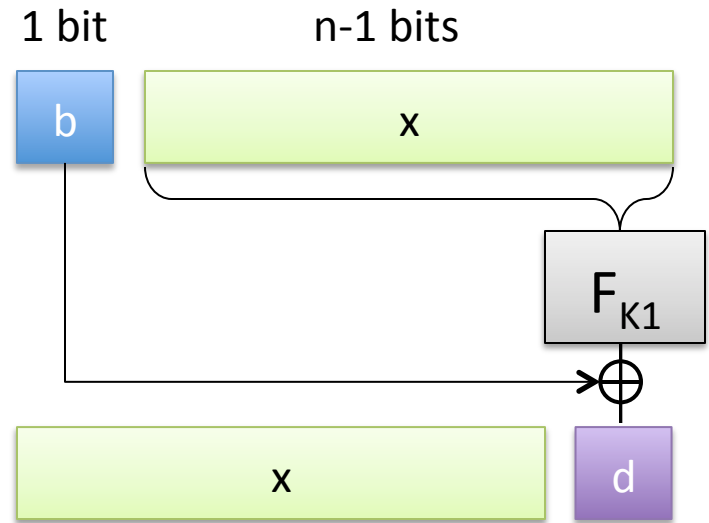


Theorem. Fix n and let Fe be a 3-round balanced Feistel network using three random functions $\rho_1, \rho_2, \rho_3 : \{0, 1\}^* \rightarrow \{0, 1\}^{n/2}$. Then for any adversary $\mathcal{A}$ making q oracle queries $\text{Adv}_{\text{Fe}}^{\text{prp}}(\mathcal{A}) \leq \frac{q^2}{2^{n/2}}$.

Time to shuffle!

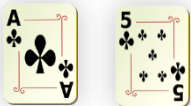
Consider a maximally unbalanced Feistel network.

This is the same as the so-called ***Thorpe shuffle*** [Morris, Rogaway, Stegers 09]



Flip random coin.

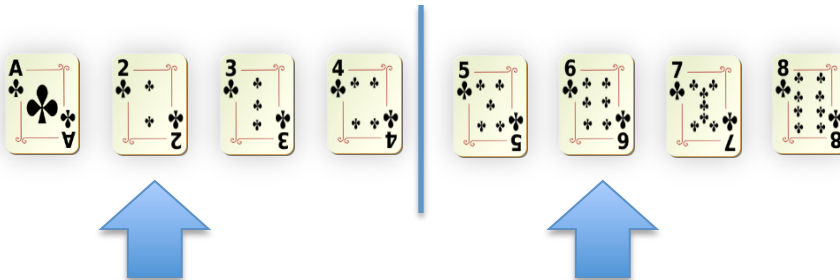
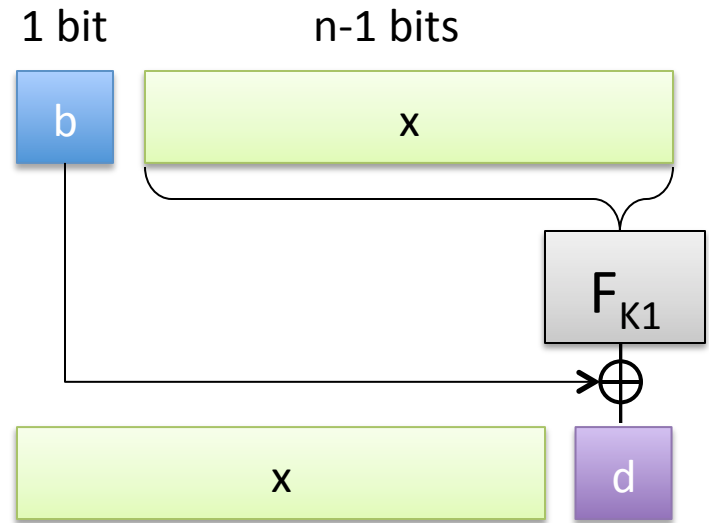
Heads put ace down first, tails 5. Say it is heads



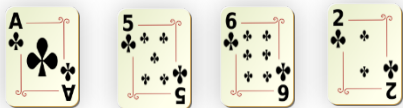
Time to shuffle!

Consider a maximally unbalanced Feistel network.

This is the same as the so-called ***Thorpe shuffle*** [Morris, Rogaway, Stegers 09]



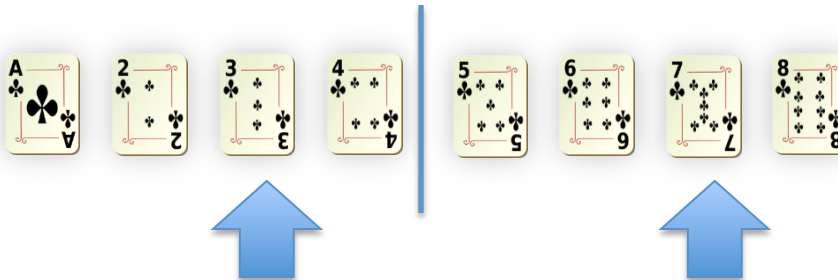
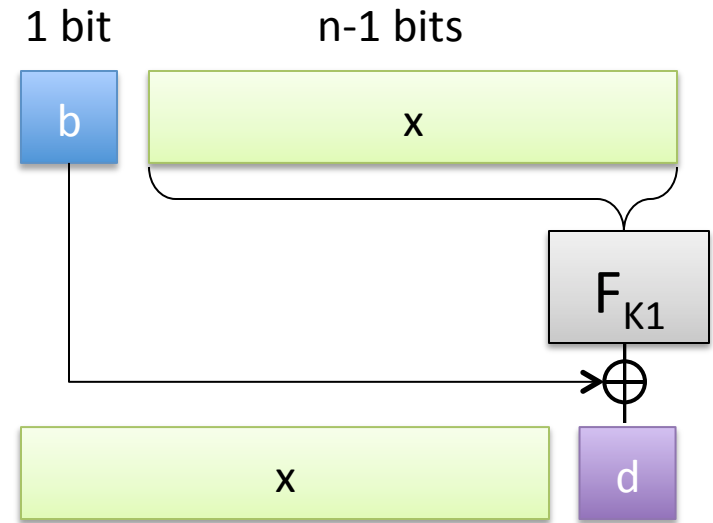
Now look at next two in each pile.
Flip a coin, and put them down in appropriate order



Time to shuffle!

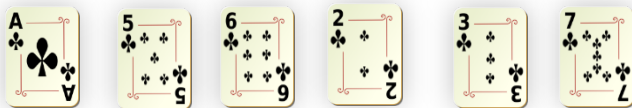
Consider a maximally unbalanced Feistel network.

This is the same as the so-called ***Thorpe shuffle*** [Morris, Rogaway, Stegers 09]



Now look at next two in each pile.

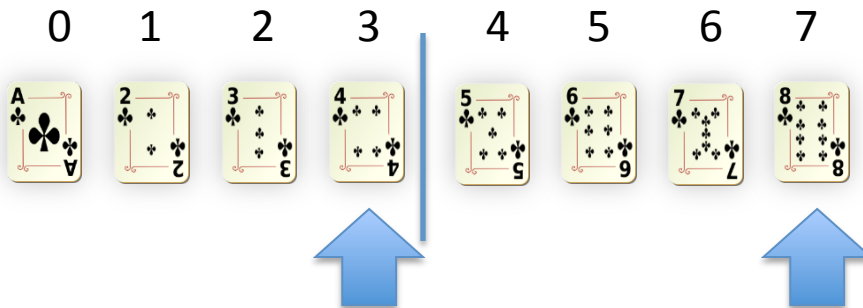
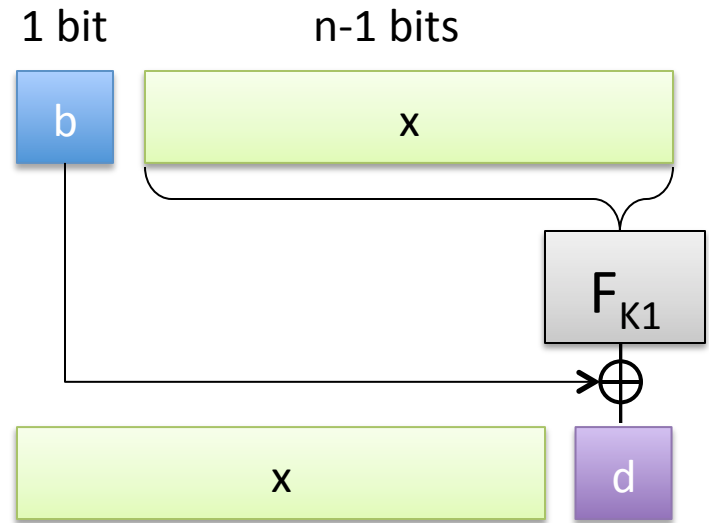
Flip a coin, and put them down in appropriate order



Time to shuffle!

Consider a maximally unbalanced Feistel network.

This is the same as the so-called **Thorpe shuffle** [Morris, Rogaway, Stegers 09]



One round of Thorp shuffle

Now think of position in deck as a numerical value:

5 gets mapped to 2

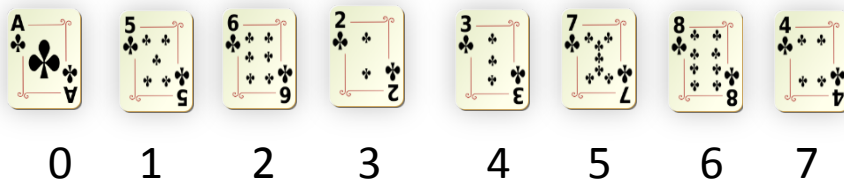
1 gets mapped to 3

...

This defines a permutation on 3 bits!

Now look at next two in each pile.

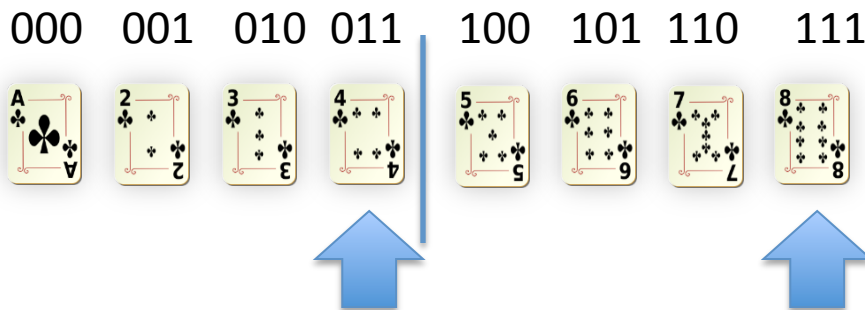
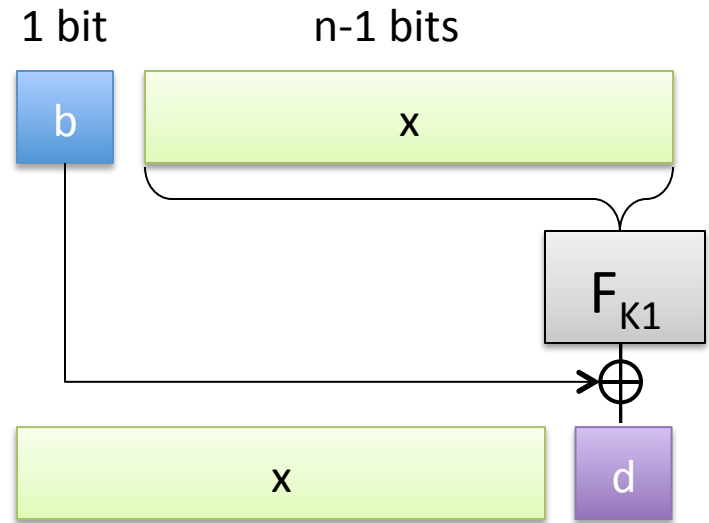
Flip a coin, and put them down in appropriate order



Time to shuffle!

Consider a maximally unbalanced Feistel network.

This is the same as the so-called **Thorpe shuffle** [Morris, Rogaway, Stegers 09]



One round of Thorpe shuffle

Now think of position in deck as a numerical value:

5 gets mapped to 2

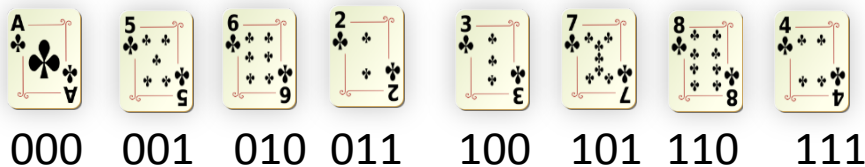
1 gets mapped to 3

...

This defines a permutation on 3 bits!

Now look at next two in each pile.

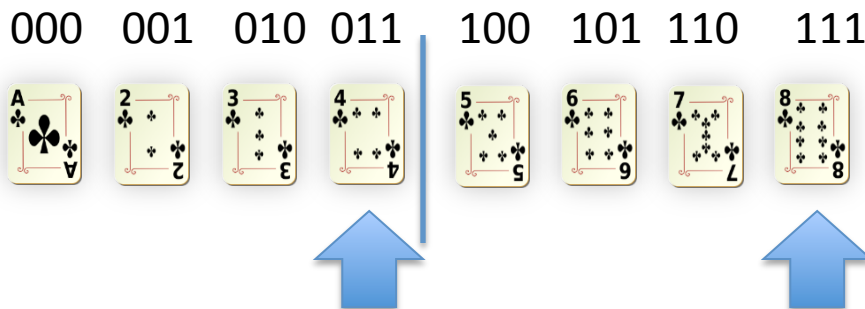
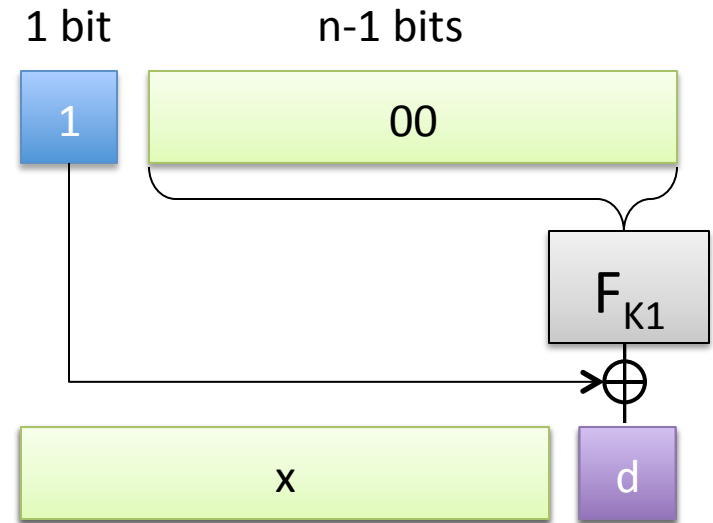
Flip a coin, and put them down in appropriate order



Time to shuffle!

Consider a maximally unbalanced Feistel network.

This is the same as the so-called **Thorpe shuffle** [Morris, Rogaway, Stegers 09]



One round of Thorp shuffle

Now think of position in deck as a numerical value:

5 gets mapped to 2

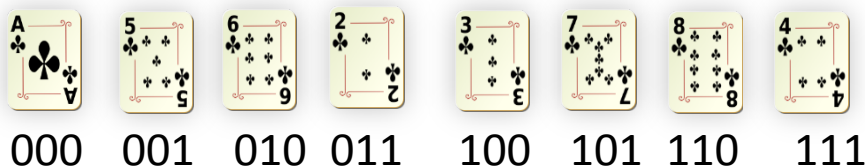
1 gets mapped to 3

...

This defines a permutation on 3 bits!

Now look at next two in each pile.

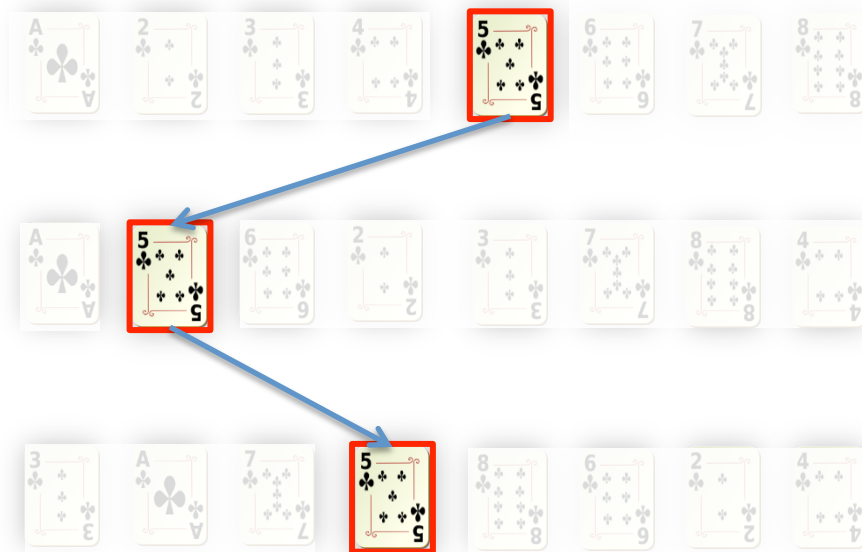
Flip a coin, and put them down in appropriate order



Do all card shuffles define block ciphers?

Sadly no, the shuffle must be ***oblivious***

Idea goes back to Naor



Example:

The traditional “Riffle” shuffle is not oblivious

Can trace position of a card without knowing what cards are in other positions, just knowing random coins related to this card

Balanced Feistel as Shuffle

$n = 4$ bits

Split deck into $2^{n/2}$ piles of size $2^{n/2}$

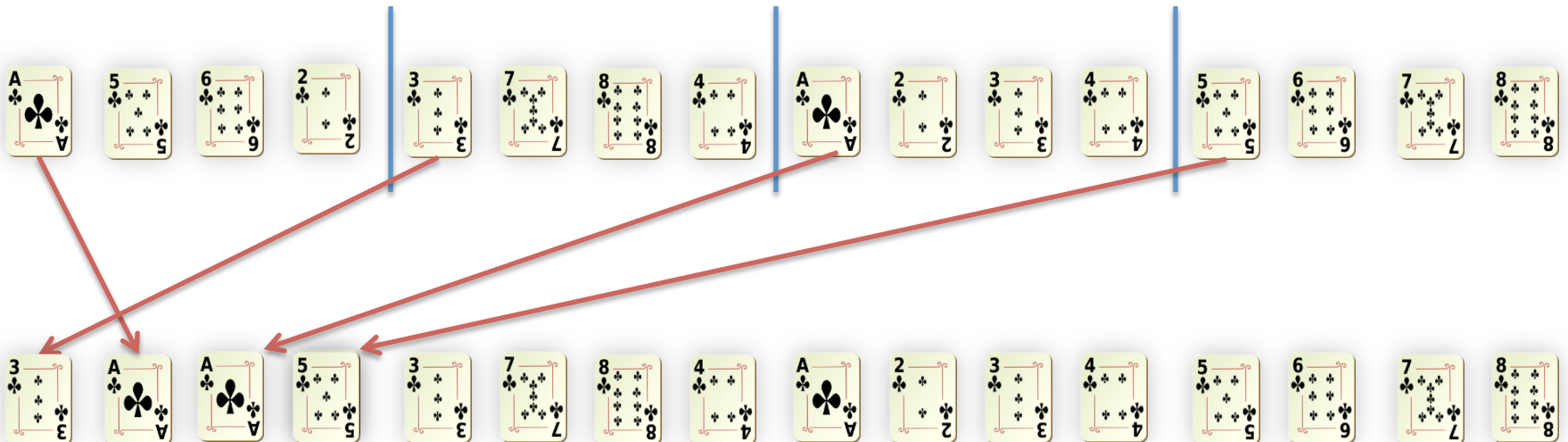
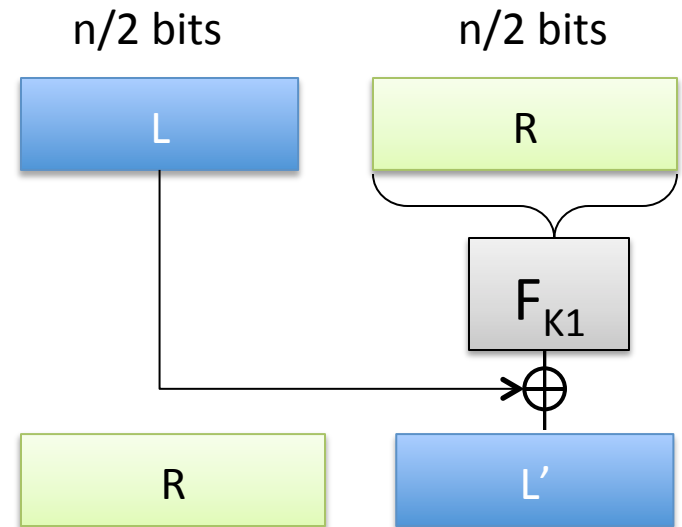
Take top card from each pile

Place them down in random order

Take next card from each pile

Place them down in random order

...



Provable results about shuffles

$$N = 2^n$$

Approach	Efficiency (# PRF calls)	Provable PRP Security
Balanced Feistel [LR 88]	3	$q \approx N^{1/4}$
Granboulin-Pornin [GP 07]	$O(\log^3 N)$	$q \approx N$
Thorp [MRS 09]	$O(\log N)$	$q \approx N^{(1-\epsilon)}$
Thorp [M 09]	$O(\log^3 N)$	$q \approx N$
Balanced Feistel [HR 10]	$O(6/\epsilon)$	$q \approx N^{(1-\epsilon)/2}$
Swap-or-Not [HMR 12]	$O(\log N)$	$q \approx (1-\epsilon)N$
Stefanov-Shi [SS 13]	$O(N^{1/2})$	$q \approx N$
Mix-and-Cut [RY 13]	$O(\log^2 N)$	$q \approx N$
Sometimes-Recurse [MR 13]	Expected $O(\log N)$	$q \approx N$

Block ciphers, Tweaks, and Shuffling

- Recall PRFs and PRPs
- Feistel and Shuffling
 - Thorpe
- Tweakable block ciphers
 - Built from Thorp
- Mix-and-Cut
 - Pseudorandom separators
- Tweakable block ciphers from PRPs:
 - Simple LRW construction
 - $XE(X)$ constructions

Tweakable block ciphers

- Add public “spice” or “tweak” to block ciphers
 - block cipher should appear to behave independently for each tweak
- First formalized by [Liskov, Rivest, Wagner 2003]
- First such block cipher is Hasty Pudding Cipher by [Schroeppele 1998]

$$\tilde{E} : \{0, 1\}^k \times \mathcal{T} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$$



Tweak space (some finite set)

Tweakable PRPs: definitions

A tweakable PRP is a map

$$\tilde{E} : \{0, 1\}^k \times \mathcal{T} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$$

$$\tilde{E}(K, M) = \tilde{E}_K^T(M)$$

$$\forall K, T, M . \tilde{D}_K^T(\tilde{E}_K^T(M)) = M$$



Inverse
algorithm

Must be efficiently computable in both directions

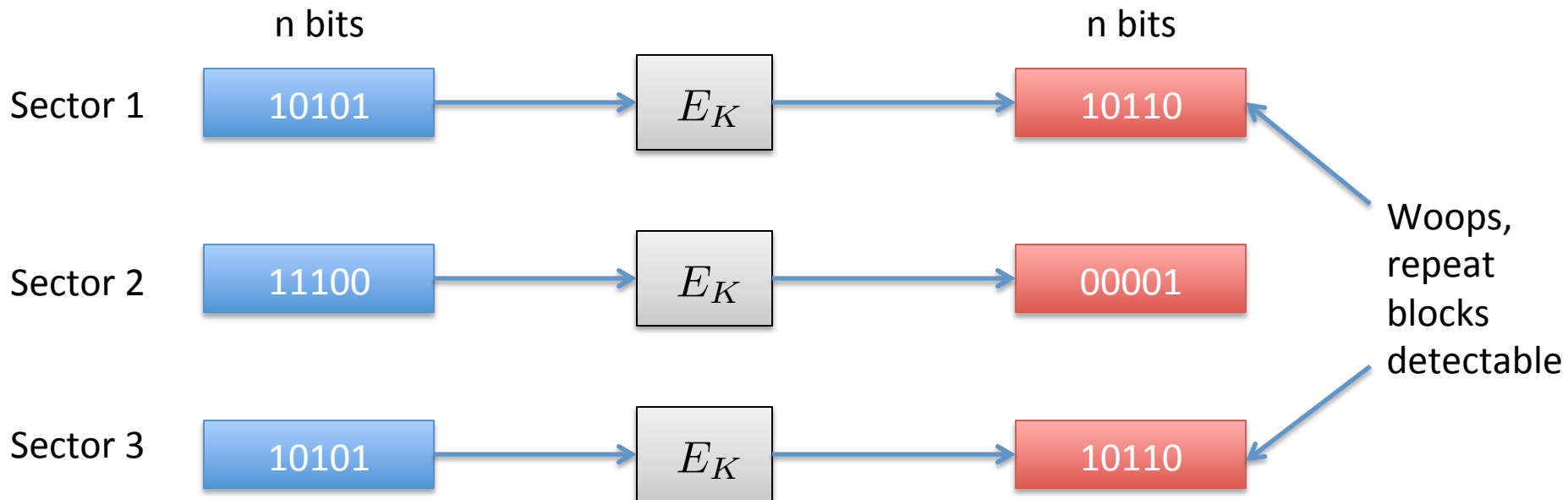
Tweakable PRPs: why?

Say we want to use an n -bit block cipher to encrypt n -bit disk sectors

Plaintext = n bits

Ciphertext = n bits

(In practice disk sectors are, e.g., 512 bytes.
We'll deal with it later)



Length-preservation means we can't use randomized encryption

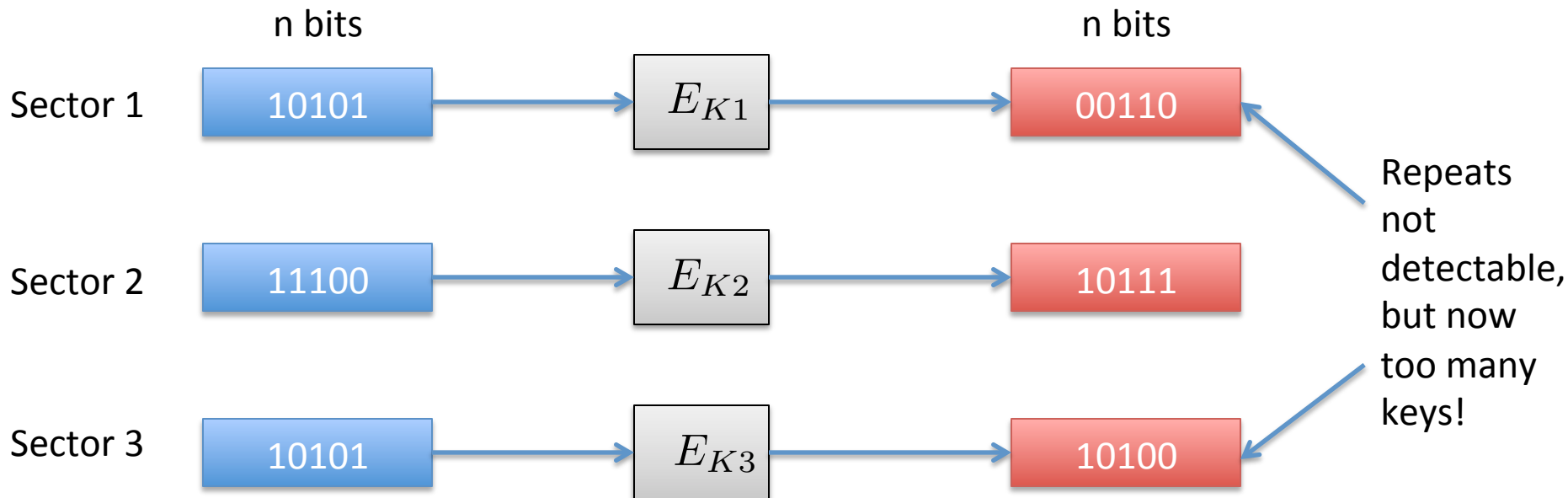
Tweakable PRPs: why?

Say we want to use an n-bit block cipher to encrypt n-bit disk sectors

Plaintext = n bits

Ciphertext = n bits

(In practice disk sectors are, e.g., 512 bytes.
We'll deal with it later)



Using independent keys with block cipher gives independent random permutations

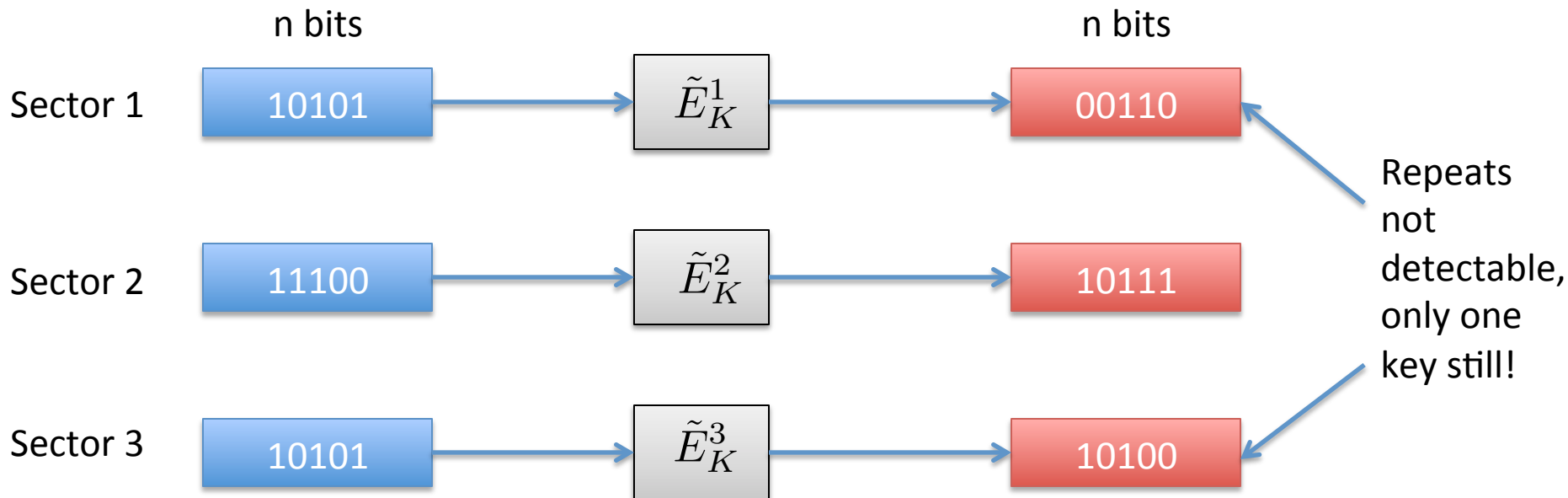
Tweakable PRPs: why?

Say we want to use an n-bit block cipher to encrypt n-bit disk sectors

Plaintext = n bits

Ciphertext = n bits

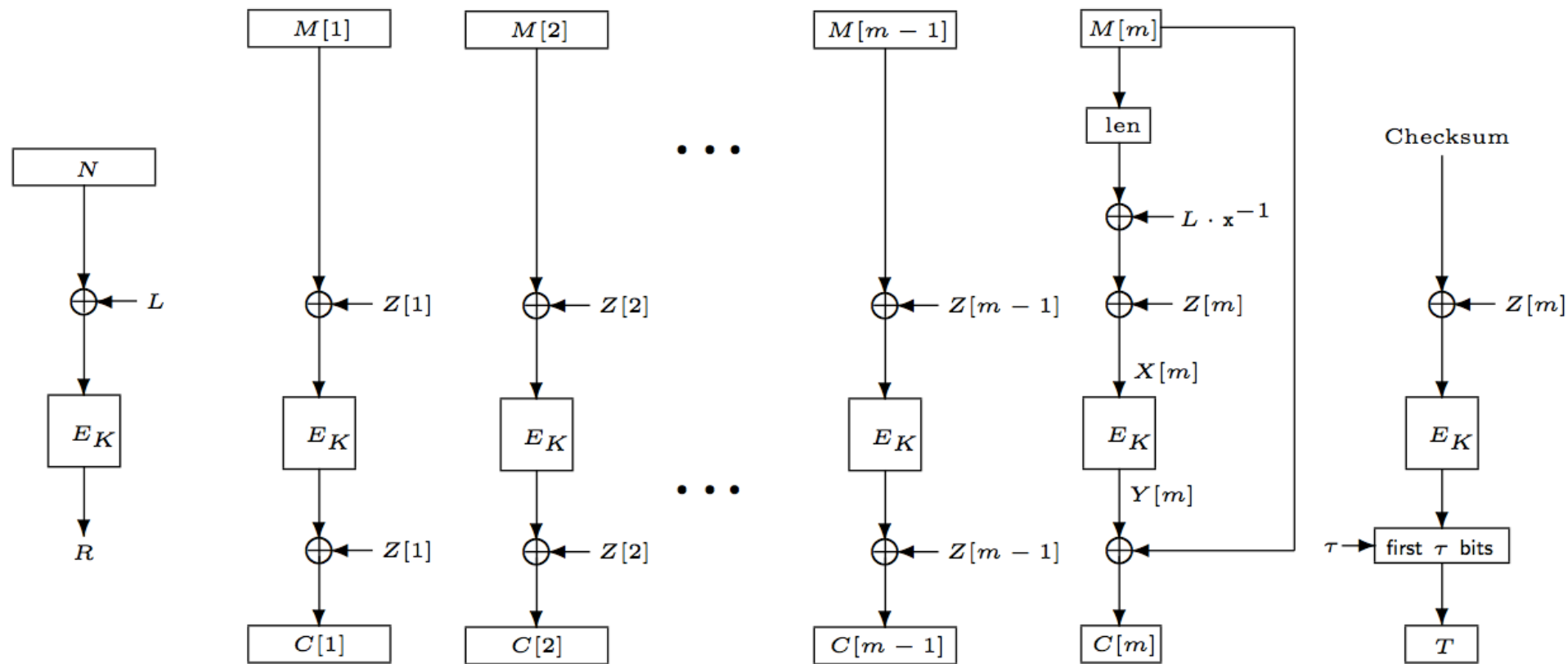
(In practice disk sectors are, e.g., 512 bytes.
We'll deal with it later)



Using distinct tweaks + good tweakable block cipher gives same effect as distinct keys. Tweaks can be public.

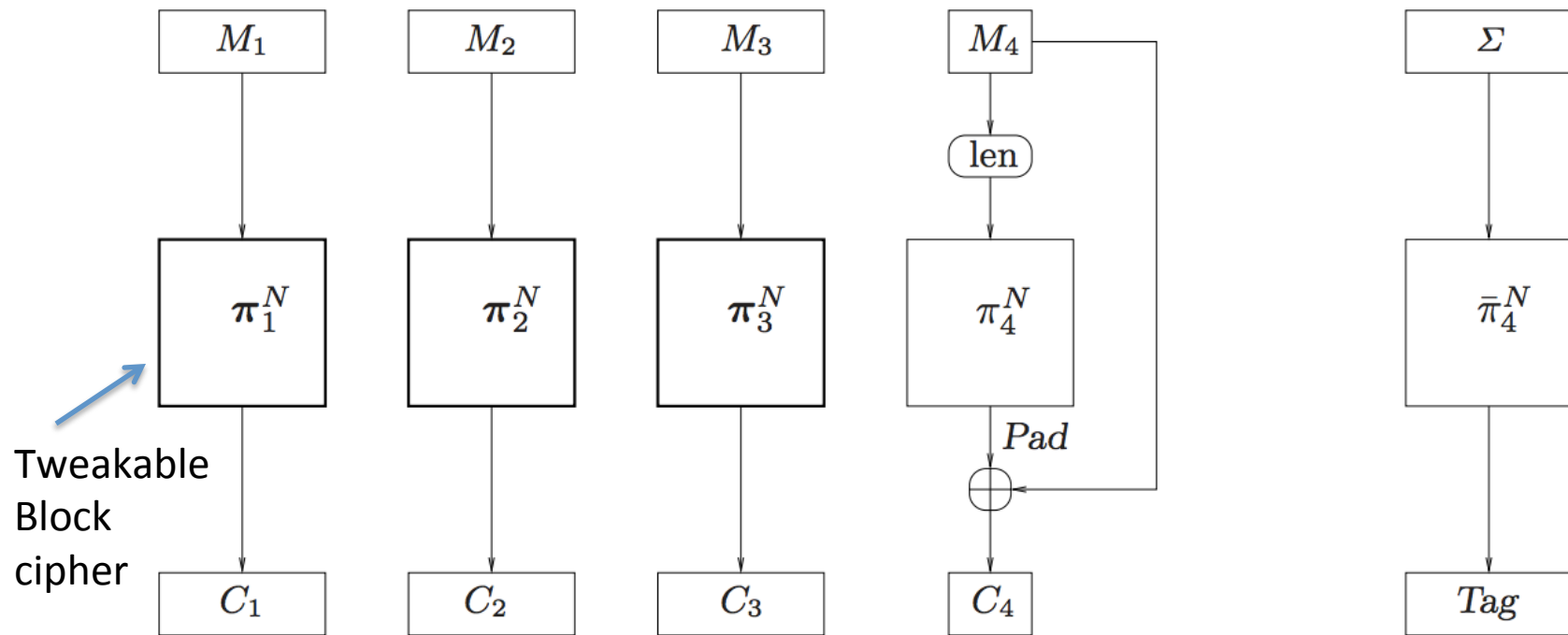
Tweakable PRPs: why?

Tweakable block ciphers are the “right” primitive for building higher level primitives such as AE



Tweakable PRPs: why?

Tweakable block ciphers are the “right” primitive for building higher level primitives such as AE



OCB AE mode, diagram from [Rogaway 2004]

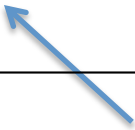
OCB was implicitly using tweakable block ciphers, making primitive explicit vastly simplifies treatment. Proof, in particular, becomes **very** simple

Tweakable PRPs (TPRPs)

Can adversary distinguish between tweakable BC for secret key and family of random permutations (one for each tweak)?

Game $\text{TPRP1}_{\tilde{E}}$

$K \leftarrow_{\$} \{0, 1\}^k$
 $b' \leftarrow_{\$} \mathcal{A}^{\tilde{E}_K(\cdot, \cdot)}$
ret b'



Game $\text{TPRP0}_{\mathcal{T}, n}$

$\tilde{\pi} \leftarrow_{\$} \text{Perm}(\mathcal{T}, n)$
 $b' \leftarrow_{\$} \mathcal{A}^{\tilde{\pi}(\cdot, \cdot)}$
ret b'



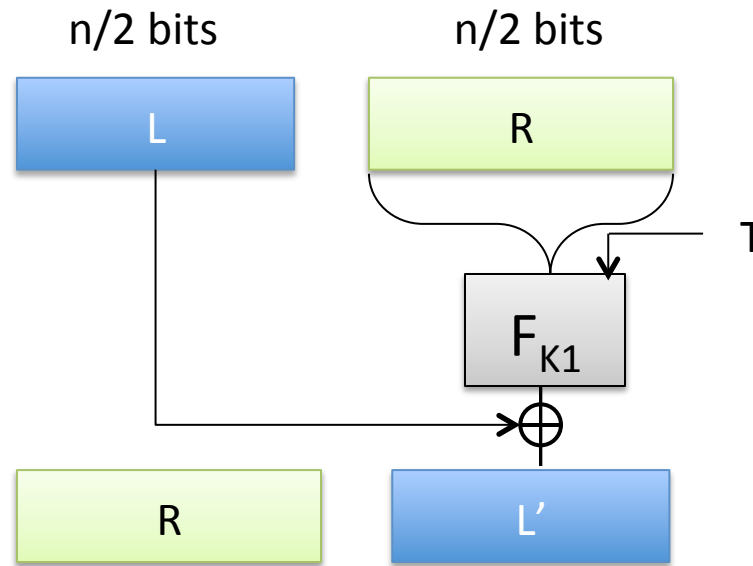
Adversary gets to choose tweaks and messages

$$\mathbf{Adv}_{\tilde{E}}^{\text{tprp}}(\mathcal{A}) = \left| \Pr \left[\text{TPRP1}_{\tilde{E}}^{\mathcal{A}} \Rightarrow 1 \right] - \Pr \left[\text{TPRP0}_{\mathcal{T}, n}^{\mathcal{A}} \Rightarrow 1 \right] \right|$$

Independent behavior for each tweak:

$\tilde{\pi}(T, M)$ and $\tilde{\pi}(T', M)$ are independent uniform bit strings

Adding tweaks to PRF-based constructions is straightforward



This gives a tweaked round of Feistel. Add tweak to all rounds.

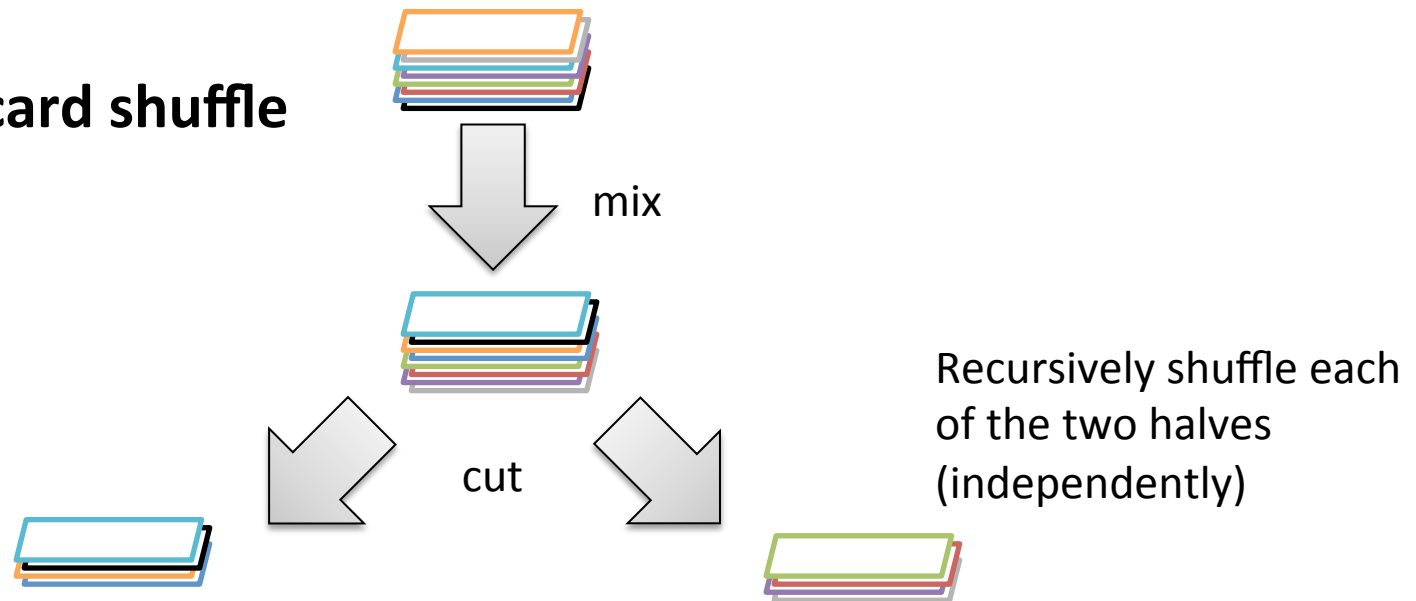
Each tweak gives rise to independent behavior

Provable results about shuffles

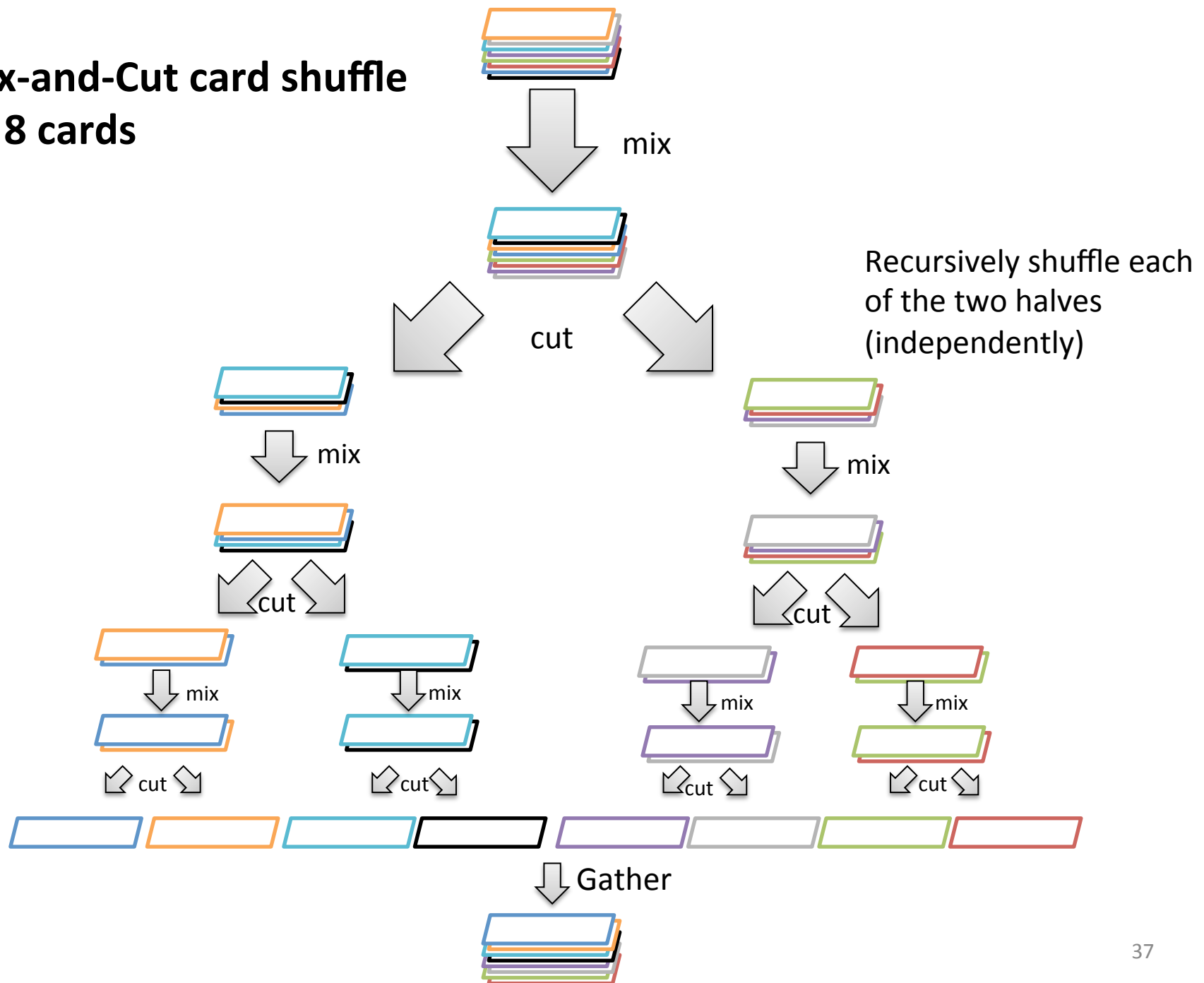
$$N = 2^n$$

Approach	Efficiency (# PRF calls)	Provable PRP Security
Balanced Feistel [LR 88]	3	$q \approx N^{1/4}$
Granboulin-Pornin [GP 07]	$O(\log^3 N)$	$q \approx N$
Thorp [MRS 09]	$O(\log N)$	$q \approx N^{(1-\epsilon)}$
Thorp [M 09]	$O(\log^3 N)$	$q \approx N$
Balanced Feistel [HR 10]	$O(6/\epsilon)$	$q \approx N^{(1-\epsilon)/2}$
Swap-or-Not [HMR 12]	$O(\log N)$	$q \approx (1-\epsilon)N$
Stefanov-Shi [SS 13]	$O(N^{1/2})$	$q \approx N$
Mix-and-Cut [RY 13]	$O(\log^2 N)$	$q \approx N$
Sometimes-Recurse [MR 13]	Expected $O(\log N)$	$q \approx N$

Mix-and-Cut card shuffle on 8 cards



Mix-and-Cut card shuffle on 8 cards



The Icicle Construction

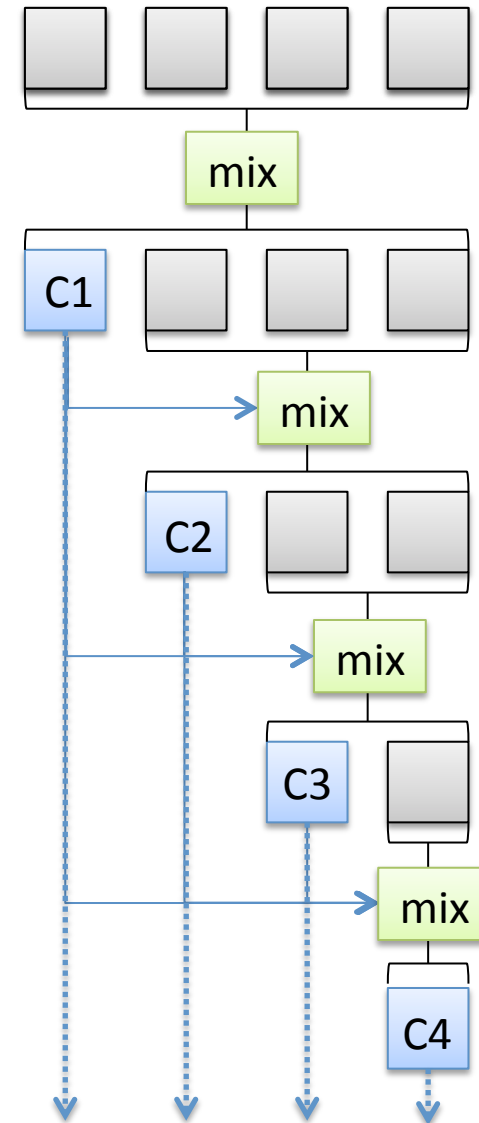
Example on 4 bits using 4 separators Z1 – Z4

C1, C2, C3, C4 “drip” down to output

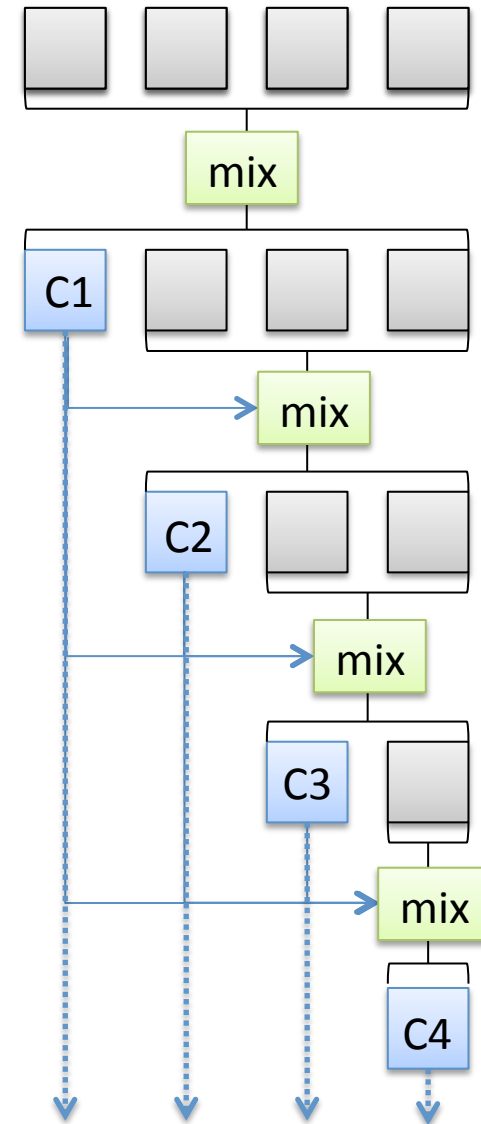
These bits are randomized by “mix”
Random choice of C1 = random choice of pile

Mixing needs to be independent depending on
which pile we’re in (indicated by bits-so-far-output)

This construction is implicit in [GP 07] cipher,
which uses perfect mix step (randomly picks C1)
based on recursive hypergeometric sampling
Super slow



The Icicle Construction



Can we instantiate mix with something more efficient?

Yes, to do so develop a framework for constructing fully-secure PRPs from

Pseudorandom Separators (PRSs)

a new primitive we define.

Can we instantiate mix with something more efficient?

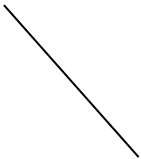
Yes, to do so develop a framework for constructing fully-secure PRPs from

Pseudorandom Separators (PRSs)

a new primitive we define.

A new framework

PRS  **PRP**


The icicle construction

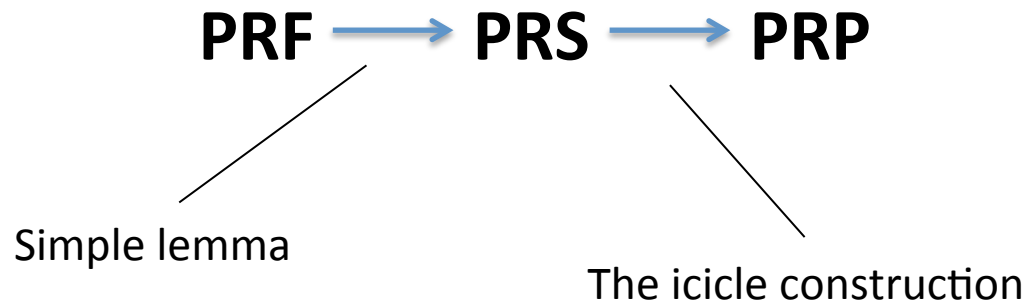
Can we instantiate mix with something more efficient?

Yes, to do so develop a framework for constructing fully-secure PRPs from

Pseudorandom Separators (PRSs)

a new primitive we define.

A new framework

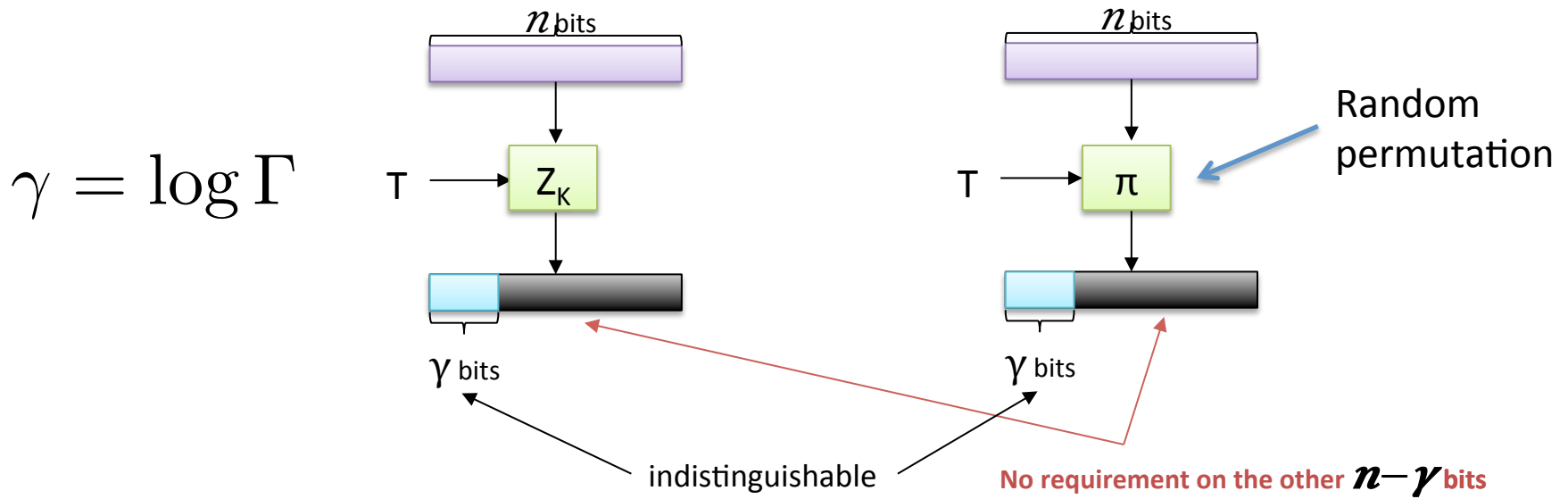


Pseudorandom Separators

Γ -pseudorandom separator (Γ -PRS) assigns cards pseudorandomly into Γ piles, but says nothing about ordering of cards in those piles.

Separator is keyed permutation $Z_K : \{0,1\}^n \rightarrow \{0,1\}^n$

Security game:



We will actually need separators that support tweaks, in the sense of tweakable block ciphers [LRW]. So separator is $Z_K : \mathcal{T} \times \{0,1\}^n \rightarrow \{0,1\}^n$

From 2-PRSs to PRPs: The Icicle Construction

Example on 4 bits using 4 separators $Z1 - Z4$

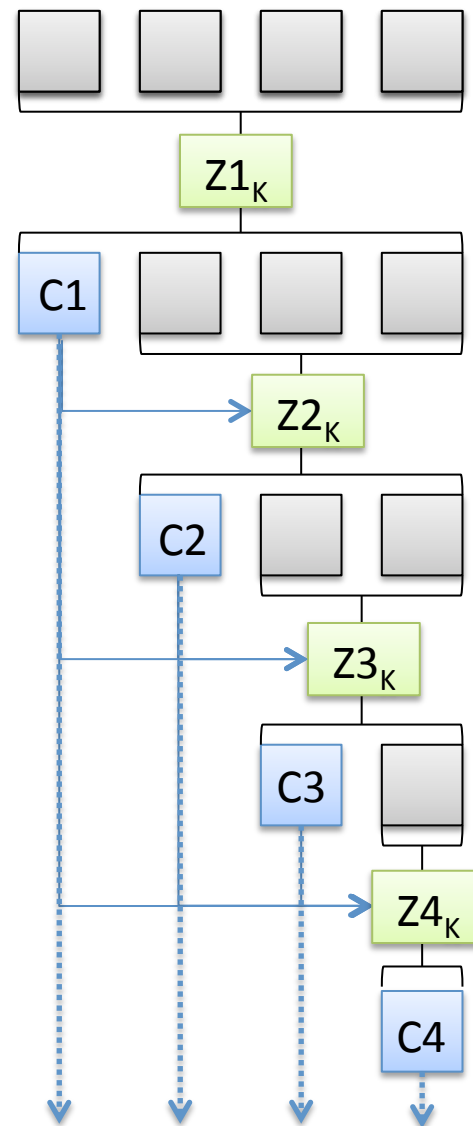
- 1) Apply $Z1$ to 4-bit input
- 2) Apply $Z2$ to 3 bits tweaked by $C1$
- 3) Apply $Z3$ to 2 bits tweaked by $C1 \parallel C2$
- 4) Apply $Z4$ to 1 bit tweaked by $C1 \parallel C2 \parallel C3$

$C1, C2, C3, C4$ “drip” down to output

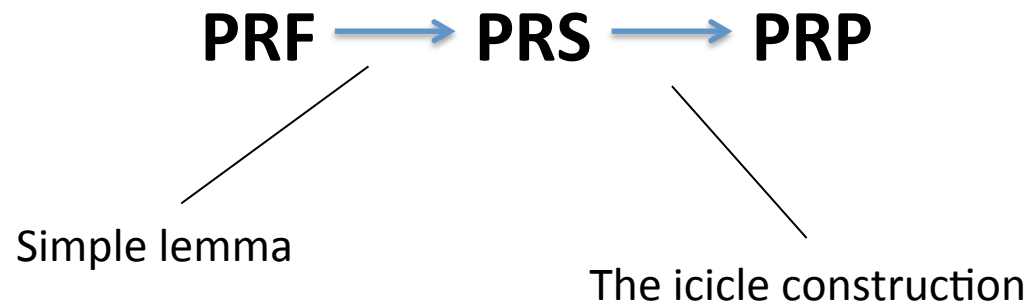
Tweaks are used to ensure separators act “independently” based on output so far

Theorem:

If Z 's are fully-secure PRSs, then the Icicle is a fully-secure PRP.



Our Framework



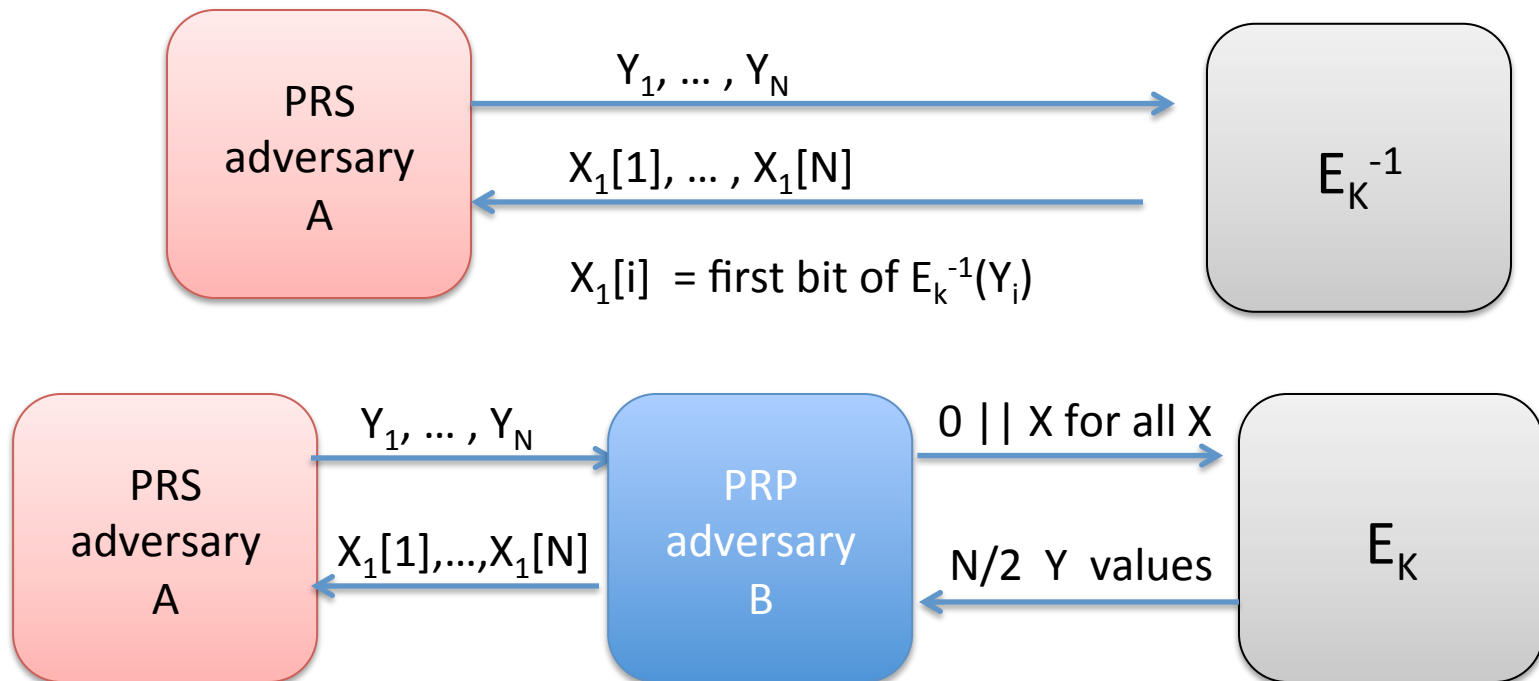
How to build PRSs: A simple lemma

Lemma:

If $E_K : N \rightarrow N$ is PRP-secure for $N/2$ queries, then E_K^{-1} is a fully secure 2-PRS

Proof sketch:

Reduction has to simulate N 2-PRS queries using only $N/2$ PRP queries



Putting it all together

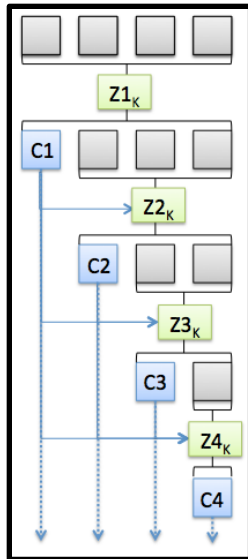
Lemma:

If $E_K : N \rightarrow N$ is PRP-secure for $q = N/2$ queries, then E_K^{-1} is a fully secure 2-PRS

Recall that Swap-or-Not [HMR 12] is secure for $(1-\epsilon)N$ queries.

Corollary:

$E_K = \text{Swap-or-Not}$ is a fully secure PRS using $r = O(\log N)$ rounds



Apply Icicle and we get fully-secure strong PRP using $O(\log^2 N)$ operations

Not usable in practice: $\sim 10,000$ rounds calls for $N \approx 2^{30}$

Faster PRSs?

Provable results about shuffles

Approach	Efficiency (# PRF calls)	Provable PRP Security
Balanced Feistel [LR 88]	3	$q \approx N^{1/4}$
Granboulin-Pornin [GP 07]	$O(\log^3 N)$	$q \approx N$
Thorp [MRS 09]	$O(\log N)$	$q \approx N^{(1-\epsilon)}$
Thorp [M 09]	$O(\log^3 N)$	$q \approx N$
Balanced Feistel [HR 10]	$O(6/\epsilon)$	$q \approx N^{(1-\epsilon)/2}$
Swap-or-Not [HMR 12]	$O(\log N)$	$q \approx (1-\epsilon)N$
Stefanov-Shi [SS 13]	$O(N^{1/2})$	$q \approx N$
Mix-and-Cut [RY 13]	$O(\log^2 N)$	$q \approx N$
Sometimes-Recurse [MR 13]	Expected $O(\log N)$	$q \approx N$

Big open question: $O(\log N)$ worse-case performance but full security?

Block ciphers, Tweaks, and Shufflin'

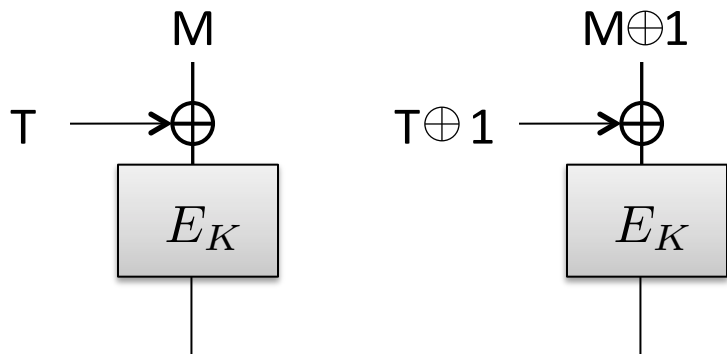
- Recall PRFs and PRPs
- Feistel and Shuffling
 - Thorpe
- Tweakable block ciphers
 - Built from Thorp
- Mix-and-Cut
 - Pseudorandom separators
- Tweakable block ciphers from PRPs:
 - Simple LRW construction
 - $XE(X)$ constructions

Fast TPRPs: Attempt 1

Given a good PRP (e.g., AES) how do we build a tweakable PRP?

Assume $\mathcal{T} = \{0, 1\}^n$

$$\tilde{E}(K, T, M) = E_K(T \oplus M)$$



Game TPRP1 $_{\tilde{E}}$

$K \leftarrow_{\$} \{0, 1\}^k$

$b' \leftarrow_{\$} \mathcal{A}^{\tilde{E}(\cdot, \cdot)}$

ret b'

Game TPRP0 $_{\tilde{E}}$

$\tilde{\pi} \leftarrow_{\$} \text{Perm}(\mathcal{T}, n)$

$b' \leftarrow_{\$} \mathcal{A}^{\tilde{\pi}(\cdot, \cdot)}$

ret b'

Can we break TPRP security? **Yup**

adversary $\mathcal{A}^{\mathcal{O}}$

$C \leftarrow \mathcal{O}(T, M)$

$C' \leftarrow \mathcal{O}(T \oplus 1, M \oplus 1)$

If $C = C'$ then Ret 1

Ret 0

$$\Pr [\text{TPRP1}_{\tilde{E}}^{\mathcal{A}} \Rightarrow 1] = 1$$

$$\Pr [\text{TPRP0}_{\mathcal{T}, n}^{\mathcal{A}} \Rightarrow 1] = 1/2^n$$

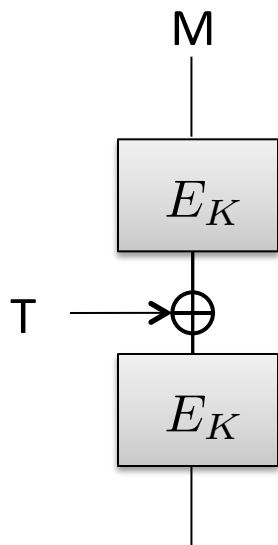
$$\text{Adv}_{\tilde{E}}^{\text{tprp}}(\mathcal{A}) = 1 - 1/2^n$$

Fast TPRPs: Attempt 2

Given a good PRP (e.g., AES) how do we build a tweakable PRP?

Assume $\mathcal{T} = \{0, 1\}^n$

$$\tilde{E}(K, T, M) = E_K(T \oplus E_K(M))$$



Game $\text{TPRP1}_{\tilde{E}}$

$K \leftarrow_{\$} \{0, 1\}^k$

$b' \leftarrow_{\$} \mathcal{A}^{\tilde{E}(\cdot, \cdot)}$

ret b'

Game $\text{TPRP0}_{\tilde{E}}$

$\tilde{\pi} \leftarrow_{\$} \text{Perm}(\mathcal{T}, n)$

$b' \leftarrow_{\$} \mathcal{A}^{\tilde{\pi}(\cdot, \cdot)}$

ret b'

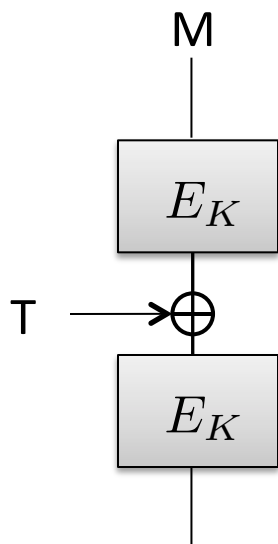
Can we break TPRP security? **Nope**

Fast TPRPs: Attempt 2

Given a good PRP (e.g., AES) how do we build a tweakable PRP?

Assume $\mathcal{T} = \{0, 1\}^n$

$$\tilde{E}(K, T, M) = E_K(T \oplus E_K(M))$$



Why does this work?

Intuition: For all M, T and M', T'

$$\Pr [M \oplus E_K(T) = M' \oplus E_K(T')]$$

is tiny.

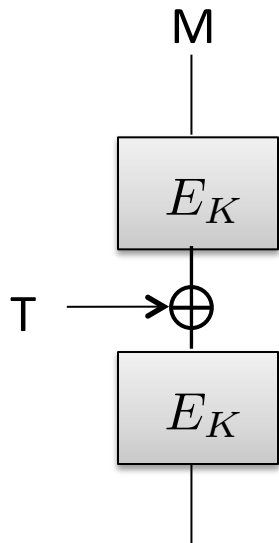
Can we break TPRP security? **Nope**

Fast TPRPs: Construction 1

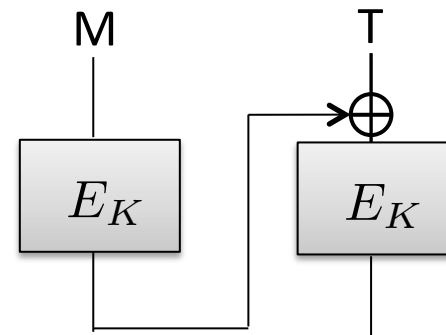
Let's sketch the proof.

Intuition:

Adversary can't force input to second E call to ever collide



Actually this construction is equivalent to CBC-MAC!



CBC-MAC[E](K,M | T)

Special case of [Bellare, Killian, Rogaway 99] gives that for any $\mathcal{A}$ there is $\mathcal{B}$ s.t.

$$\mathbf{Adv}_{\text{CBC-MAC}[E]}^{\text{prf}}(\mathcal{A}) \leq \mathbf{Adv}_E^{\text{prp}}(\mathcal{B}) + \frac{8q^2}{2^n}$$

Here $\mathcal{B}$ must make $2q$ queries when $\mathcal{A}$ makes q queries

Fast TPRPs: Construction 1

Let's sketch the proof.

Intuition:

Adversary can't force input to second E call to ever collide

$$\begin{aligned}
 \mathbf{Adv}_{\tilde{E}}^{\text{tprp}}(\mathcal{A}) &= \Pr [\text{TPRP1}_{\tilde{E}}^{\mathcal{A}} \Rightarrow 1] - \Pr [\text{TPRP0}_{\mathcal{T},n}^{\mathcal{A}} \Rightarrow 1] \\
 &\leq \Pr [\text{TPRP1}_{\tilde{E}}^{\mathcal{A}} \Rightarrow 1] - \Pr [\text{PRF0}_n^{\mathcal{A}} \Rightarrow 1] + \frac{q^2}{2^n} \\
 &= \mathbf{Adv}_{\text{CBC-MAC}[E]}^{\text{prf}}(\mathcal{B}) + \frac{q^2}{2^n} \\
 &\leq \mathbf{Adv}_E^{\text{prp}}(\mathcal{B}') + \frac{16q^2}{2^n} + \frac{q^2}{2^n} \\
 &= \mathbf{Adv}_E^{\text{prp}}(\mathcal{B}') + \frac{17q^2}{2^n}
 \end{aligned}$$

Extension of PRP/PRF switching to tweaks

CBC[E] and $\tilde{E}$ are the same

Bellare et al. result on CBC-MAC. (2q queries)

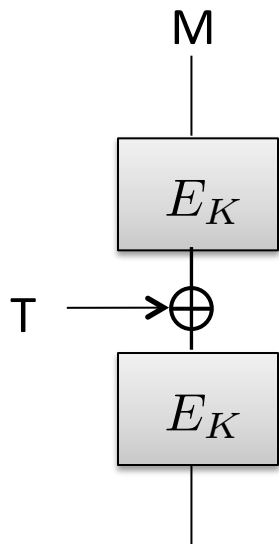
Addition

Fast TPRPs: Construction 2

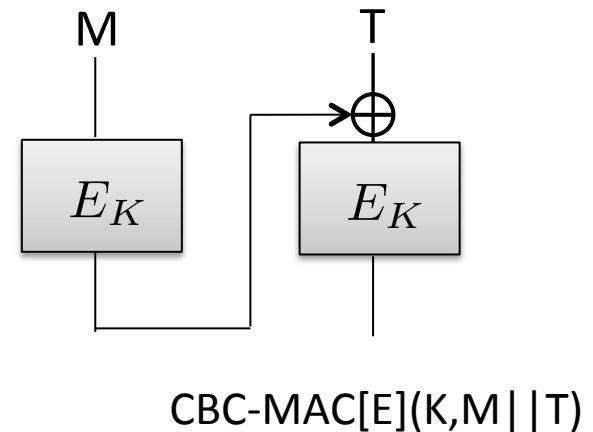
Given a good PRP (e.g., AES) how do we build a tweakable PRP?

Assume $\mathcal{T} = \{0, 1\}^n$

$$\tilde{E}(K, T, M) = E_K(T \oplus E_K(M))$$



Why can't we switch ordering of M and T in CBC-MAC interpretation?

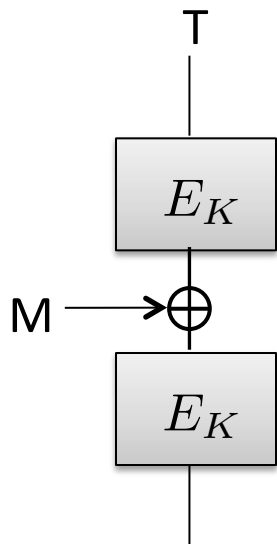


Fast TPRPs: Construction 2

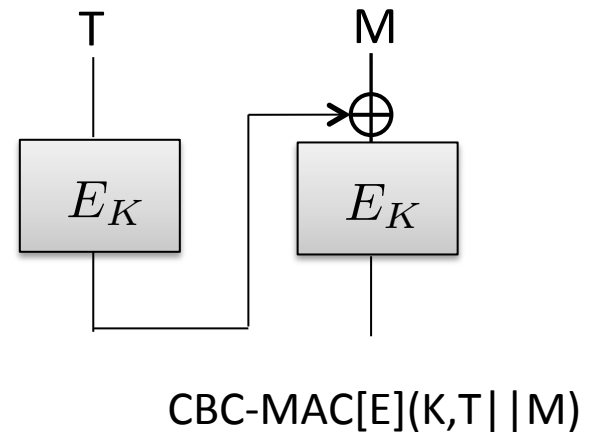
Given a good PRP (e.g., AES) how do we build a tweakable PRP?

Assume $\mathcal{T} = \{0, 1\}^n$

$$\tilde{E}(K, T, M) = E_K(M \oplus E_K(T))$$



Why can't we switch ordering of M and T in CBC-MAC interpretation?



We can! Same proof.

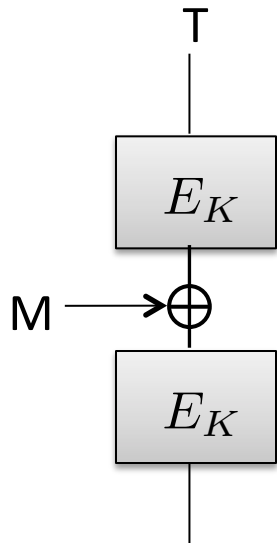
This will be faster in some cases. When?

Fast TPRPs

Given a good PRP (e.g., AES) how do we build a tweakable PRP?

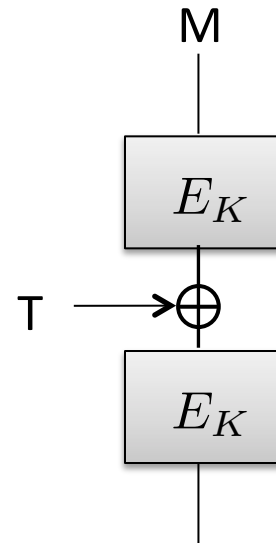
Assume $\mathcal{T} = \{0, 1\}^n$

$$\tilde{E}(K, T, M) = E_K(T \oplus E_K(M))$$



Fast for fixed tweak

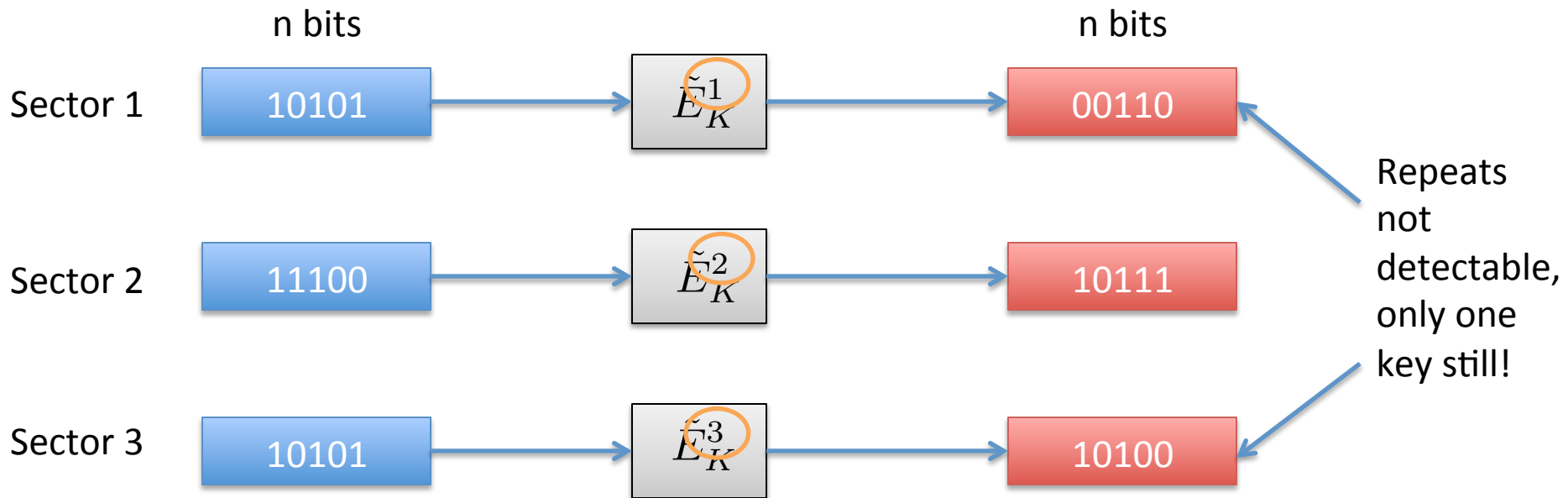
$$\tilde{E}(K, T, M) = E_K(M \oplus E_K(T))$$



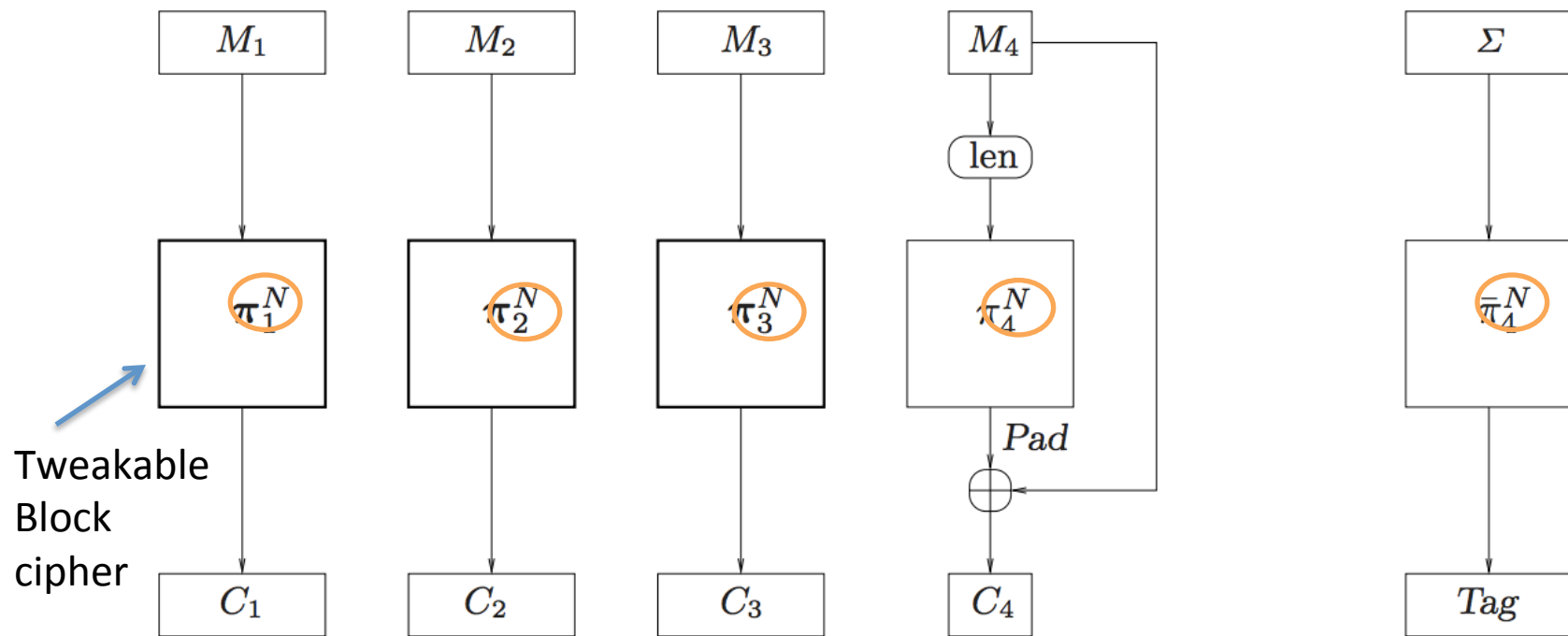
Fast for fixed message

Can we get best of both worlds? Yes, for special cases.

Recall motivating applications



Recall motivating applications



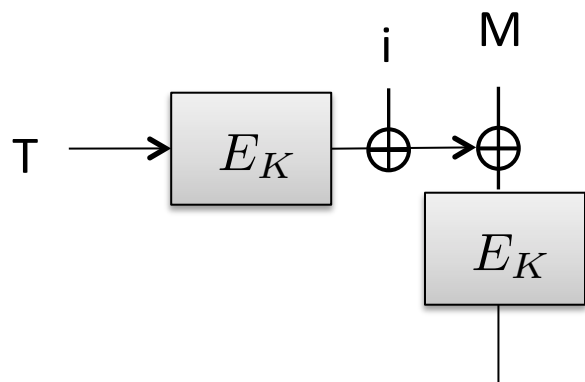
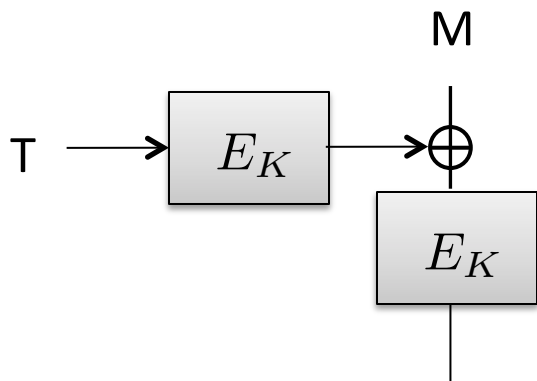
For many applications tweakable BC with specialized tweak space suffices

Towards faster TPRPs

Given a good PRP (e.g., AES) how do we build a tweakable PRP?

Assume $\mathcal{T} = \{0, 1\}^n \times [0..2^n - 2]$

$$\tilde{E}(K, T, M) = E_K(T \oplus E_K(M))$$



Ideal to have something like this, where i can be an integer

But this is insecure!

Essentials on $\text{GF}(2^n)$

- $\text{GF}(2^n)$ is Finite (Galois) Field over n -bit strings
- Represent point a equivalently as:
 - Polynomial $a(x) = a_{n-1}x^{n-1} + \cdots + a_1x + a_0$
 - Integer
$$a = \sum_{i=0}^{n-1} a_i 2^i$$
 - Bit string $a = a_{n-1} \cdots a_1 a_0$

Essentials on GF(2ⁿ)

- GF(2ⁿ) is Finite (Galois) Field over n-bit strings
- Addition = bitwise xor of points (as bit strings)
- Multiplication:
 - fix primitive poly. For n = 128 use
$$x^{128} + x^7 + x^2 + x + 1$$
 - Multiple points *ac* by treating as polynomials, multiplying formal polys, and take remainder mod fixed prim poly
- Benefit: $x = 2$ ($a = 0^{126}10$) generates multiplicative subgroup of GF(2ⁿ)

Essentials on $\text{GF}(2^n)$

Multiplication by 2 is fast!

$$\begin{aligned} 2(x) \cdot a(x) &= (x) \cdot (a_{n-1}x^{n-1} + \dots + a_1x + a_0) \\ &= a_{n-1}x^n + \dots + a_1x^2 + a_0x \end{aligned}$$

Divide by primitive polynomial and take remainder

If $a_{n-1} = 1$ then:

left-shift a by 1 and add prim poly

If $a_{n-1} = 0$ then:

left-shift a by 1

One shift and one conditional xor!

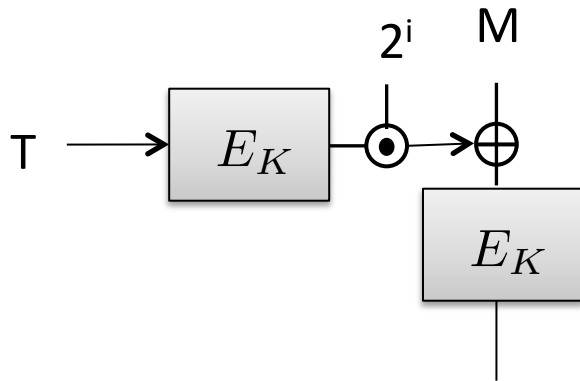
Fast TPRPs: XE construction

[Rogaway 2004]

Given a good PRP (e.g., AES) how do we build a tweakable PRP?

Assume $\mathcal{T} = \{0, 1\}^n \times [0..2^n - 2]$

$$\tilde{E}(K, (T, i), M) = E_K(M \oplus \Delta) \quad \Delta = E_K(T) \cdot 2^i$$



Multiplication in $GF(2^n)$ using primitive polynomial. This means that

$2Y, 4Y, 8Y, \dots$

are all distinct! (Think $Y = E_K(T)$)

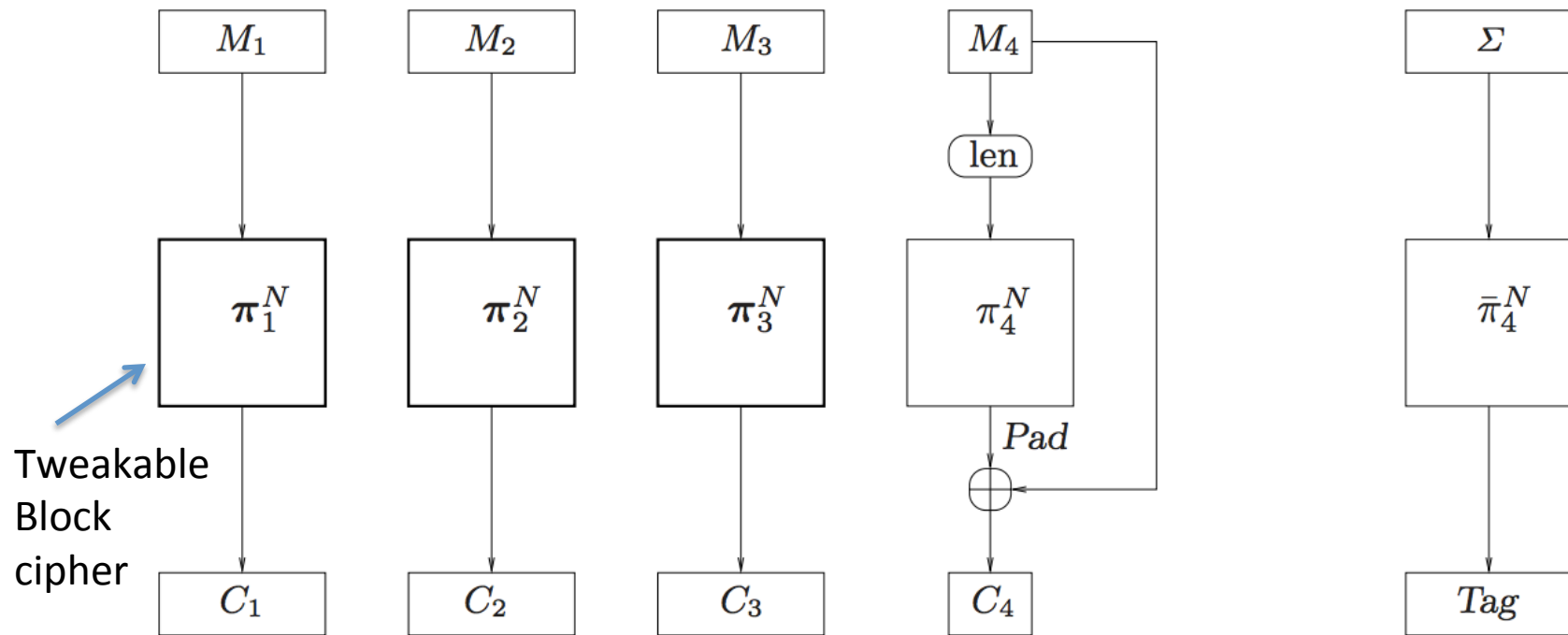
Proof by extension of previous arguments

But is this fast? Yes in an amortized sense:

Computing $2Y$ given Y is 1 shift and 1 conditional xor!

Fast TPRPs: XE construction

Using XE in a mode in many settings is fast. Compute $E_K(N)$ once and then only have to do doublings



So far: only secure against chosen-plaintext attacks

Can adversary distinguish between tweakable BC for secret key and family of random permutations (one for each tweak)?

Game $\text{TPRP1}_{\tilde{E}}$

$K \leftarrow_{\$} \{0, 1\}^k$
 $b' \leftarrow_{\$} \mathcal{A}^{\tilde{E}_K(\cdot, \cdot)}$
ret b'

Game $\text{TPRP0}_{\mathcal{T}, n}$

$\tilde{\pi} \leftarrow_{\$} \text{Perm}(\mathcal{T}, n)$
 $b' \leftarrow_{\$} \mathcal{A}^{\tilde{\pi}(\cdot, \cdot)}$
ret b'

Adversary gets to choose tweaks and messages

$$\mathbf{Adv}_{\tilde{E}}^{\text{tprp}}(\mathcal{A}) = \left| \Pr \left[\text{TPRP1}_{\tilde{E}}^{\mathcal{A}} \Rightarrow 1 \right] - \Pr \left[\text{TPRP0}_{\mathcal{T}, n}^{\mathcal{A}} \Rightarrow 1 \right] \right|$$

Independent behavior for each tweak:

$\tilde{\pi}(T, M)$ and $\tilde{\pi}(T', M)$ are independent uniform bit strings

Tweakable Strong PRPs (TSPRPs)

Can adversary distinguish between tweakable BC for secret key and its inverse and family of random permutations (one for each tweak) and inverses?

Game TSPRP1 $\tilde{E}$

$K \leftarrow_{\$} \{0, 1\}^k$
 $b' \leftarrow_{\$} \mathcal{A}^{\tilde{E}_K(\cdot, \cdot), \tilde{D}_K(\cdot, \cdot)}$
ret b'

Game TSPRP0 $\mathcal{T}, n$

$\tilde{\pi} \leftarrow_{\$} \text{Perm}(\mathcal{T}, n)$
 $b' \leftarrow_{\$} \mathcal{A}^{\tilde{\pi}(\cdot, \cdot), \tilde{\pi}^{-1}(\cdot, \cdot)}$
ret b'

Adversary gets to choose tweaks and messages

$$\mathbf{Adv}_{\tilde{E}}^{\text{tsprp}}(\mathcal{A}) = \left| \Pr [\text{TSPRP1}_{\tilde{E}}^{\mathcal{A}} \Rightarrow 1] - \Pr [\text{TSPRP0}_{\mathcal{T}, n}^{\mathcal{A}} \Rightarrow 1] \right|$$

A sometimes useful rule of thumb in building Strong PRPs:

Do what you did for PRPs, twice

... and then spend >2x the time checking/proving it works

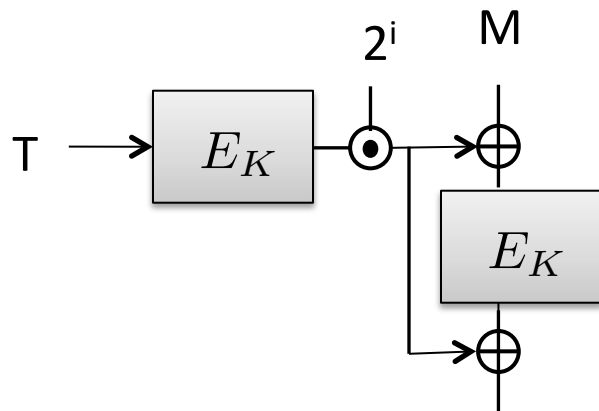
Fast TPRPs: XEX construction

[Rogaway 2004]

Given a good PRP (e.g., AES) how do we build a tweakable PRP?

Assume $\mathcal{T} = \{0, 1\}^n \times [0..2^n - 2]$

$$\tilde{E}(K, (T, i), M) = E_K(M \oplus \Delta) \oplus \Delta \quad \Delta = E_K(T) \cdot 2^i$$



Ok we did it twice. Done?

Not quite: can't use tweak $(T, 0)$ for any T

adversary $\mathcal{A}^{\mathcal{O}, \mathcal{O}^{-1}}$

$N \leftarrow \mathcal{O}^{-1}((0^n, 0), 0^n)$
 $C \leftarrow \mathcal{O}((0^n, 1), 2N)$
 $C' \leftarrow \mathcal{O}((0^n, 2), 4N)$
 If $C = C'$ then Ret 1
 Ret 0

Fast TPRPs: XEX construction

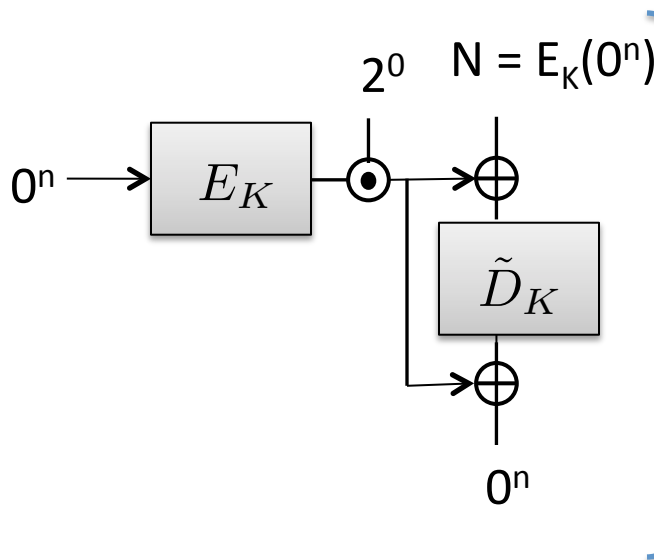
[Rogaway 2004]

Given a good PRP (e.g., AES) how do we build a tweakable PRP?

Assume $\mathcal{T} = \{0, 1\}^n \times [0..2^n - 2]$

Just don't allow 0

$$\tilde{E}(K, (T, i), M) = E_K(M \oplus \Delta) \oplus \Delta \quad \Delta = E_K(T) \cdot 2^i$$



Ok we did it twice. Done?

Not quite: can't use tweak $(T, 0)$ for any T

adversary $\mathcal{A}^{\mathcal{O}, \mathcal{O}^{-1}}$

$$N \leftarrow \mathcal{O}^{-1}((0^n, 0), 0^n)$$

$$C \leftarrow \mathcal{O}((0^n, 1), 2N)$$

$$C' \leftarrow \mathcal{O}((0^n, 2), 4N)$$

If $C = C'$ then Ret 1
Ret 0

Fast TPRPs: XEX construction

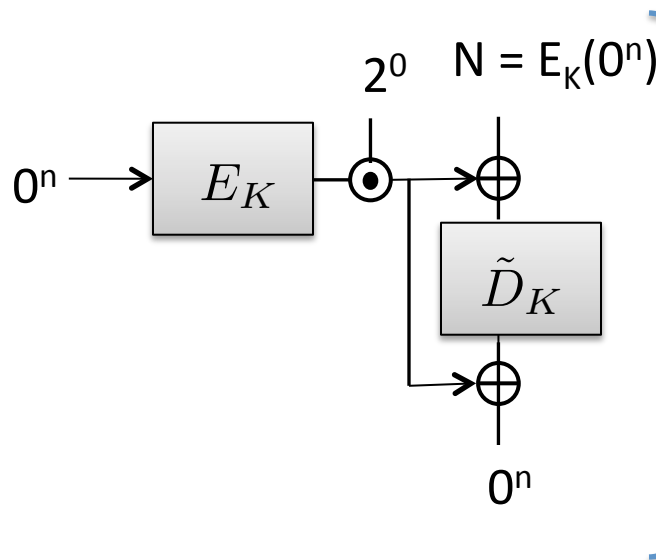
[Rogaway 2004]

Given a good PRP (e.g., AES) how do we build a tweakable PRP?

Assume $\mathcal{T} = \{0, 1\}^n \times [1..2^n - 2]$

Just don't allow 0

$$\tilde{E}(K, (T, i), M) = E_K(M \oplus \Delta) \oplus \Delta \quad \Delta = E_K(T) \cdot 2^i$$



Ok we did it twice. Done?

Not quite: can't use tweak $(T, 0)$ for any T

adversary $\mathcal{A}^{\mathcal{O}, \mathcal{O}^{-1}}$

$$N \leftarrow \mathcal{O}^{-1}((0^n, 0), 0^n)$$

$$C \leftarrow \mathcal{O}((0^n, 1), 2N)$$

$$C' \leftarrow \mathcal{O}((0^n, 2), 4N)$$

If $C = C'$ then Ret 1
Ret 0

Block ciphers, Tweaks, and Shufflin'

- Recall PRFs and PRPs
- Feistel and Shuffling
 - Thorpe
- Tweakable block ciphers
 - Built from Thorp
- Mix-and-Cut Shuffle
 - Pseudorandom separators
- Tweakable block ciphers from PRPs:
 - Simple LRW construction
 - $XE(X)$ constructions

Lecture plan

1. Tweakable PRPs and shuffling
2. Nonce-based symmetric encryption
3. Format-preserving encryption
4. Further symmetric primitives

Lecture 2:

Nonce-based Symmetric Encryption

Thomas Ristenpart
University of Wisconsin

Lecture plan

1. Tweakable PRPs and shuffling
2. Nonce-based symmetric encryption
3. Format-preserving encryption
4. Further symmetric primitives

Outline

- Nonce-based Symmetric Encryption
 - Security notions
- OCB (Offset Codebook Mode)
- 2-pass Modes

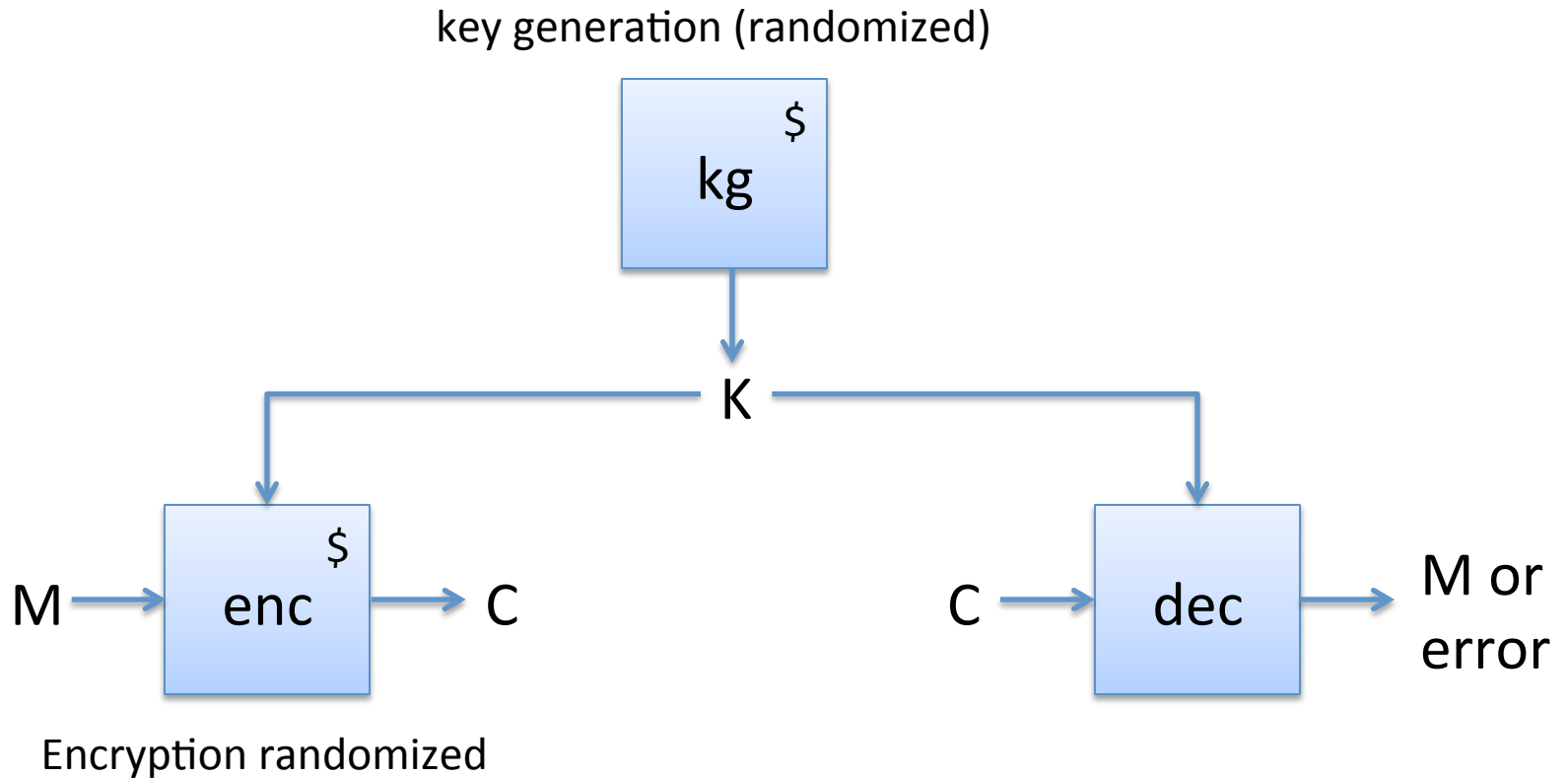
Deficiencies in prior treatments

- Encryption and authenticity as separate primitives leads to problems
 - Get the order wrong
 - Hard to implement properly
- Nonces are a problem in practice
 - Bad randomness
 - Counters get replayed
 - Failure modes are bad (e.g., CTR mode)
- Speed is very important
 - Generic composition = 2 cryptographic passes

The expanding role of SE

- Recognition that the crypto should do more to help implementers
- Crypto needs to work around system constraints
 - Bad nonces should be protected against to best extent possible
 - API should be simple
 - Systems often can't be redesigned (e.g., format-preservation, passwords)

Symmetric encryption

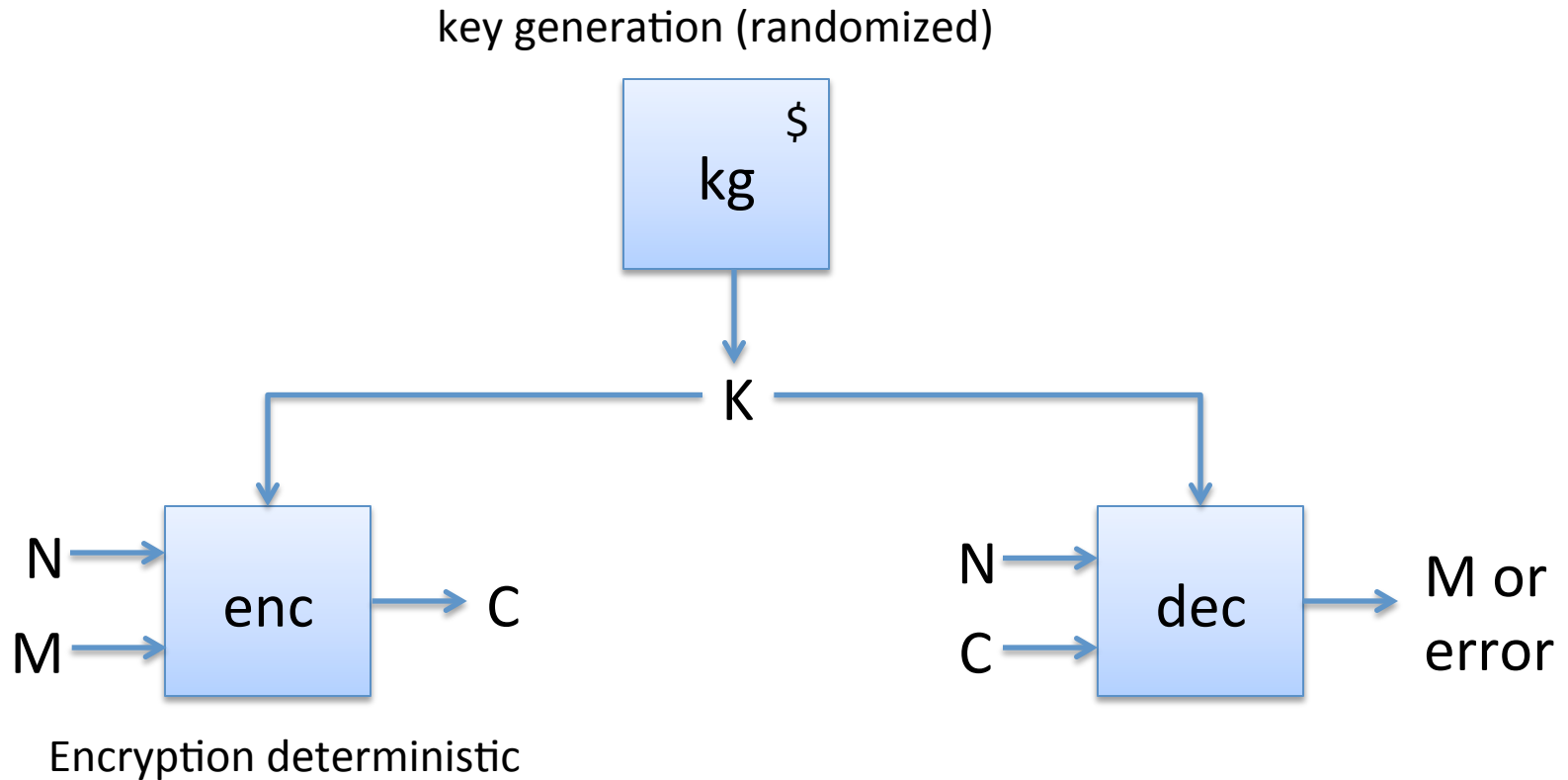


$SE = (kg, enc, dec)$

$$\forall M . \Pr [\text{dec}(K, \text{enc}(K, M)) = M] = 1$$

Probability over coins of kg and enc

Nonce-based symmetric encryption

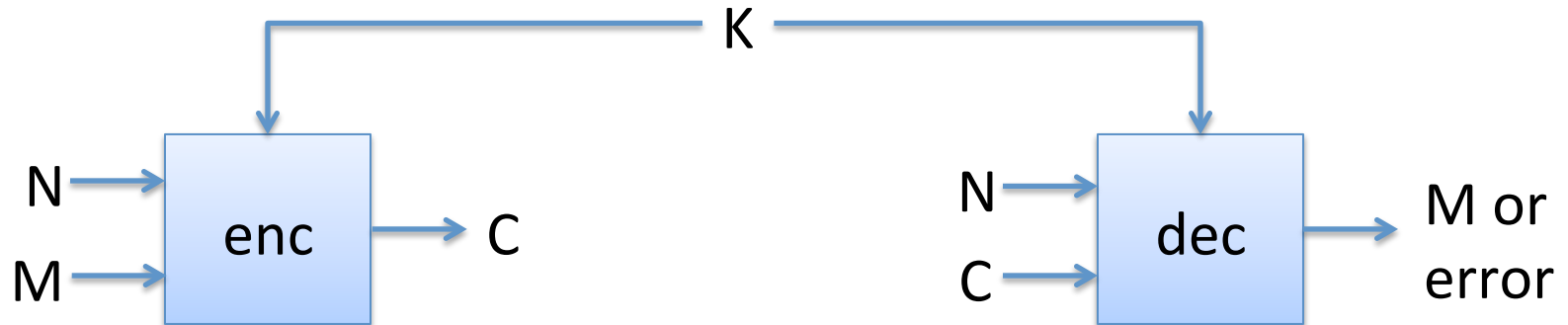


$$SE = (kg, enc, dec)$$

$$\forall N, M . \Pr [\text{dec}(K, N, \text{enc}(K, N, M)) = M]$$

Probability over coins of kg

Associated data



In most applications of encryption, need to consider plaintext data that should be cryptographically bound to ciphertext

- Called associated data
- Think of N now as a vector of bit strings
- Security notions are same (constructions can treat long N)

For simplicity we'll just think of N as a short nonce

Nonce-based security: two flavors

Game REAL_{SE}

$K \leftarrow_{\$} \{0, 1\}^k$

$b' \leftarrow_{\$} \mathcal{A}^{\text{Enc}}$

ret b'

$\text{Enc}(N, M)$

$C \leftarrow \text{enc}(K, N, M)$

Ret C

Game RAND_{SE}

$K \leftarrow_{\$} \{0, 1\}^k$

$b' \leftarrow_{\$} \mathcal{A}^{\text{Enc}}$

ret b'

$\text{Enc}(N, M)$

$C \leftarrow \text{enc}(K, N, M)$

$C \leftarrow_{\$} \{0, 1\}^{|C|}$

Ret C

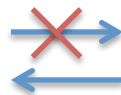
$$\text{Adv}_{\text{SE}}^{\text{ind\$}}(\mathcal{A}) = \left| \Pr [\text{REAL}_{\text{SE}}^{\mathcal{A}} \Rightarrow 1] - \Pr [\text{RAND}_{\text{SE}}^{\mathcal{A}} \Rightarrow 1] \right|$$

Nonce-respecting adversaries: all queries unique N

Repetition-respecting adversaries: all queries unique (N, M)

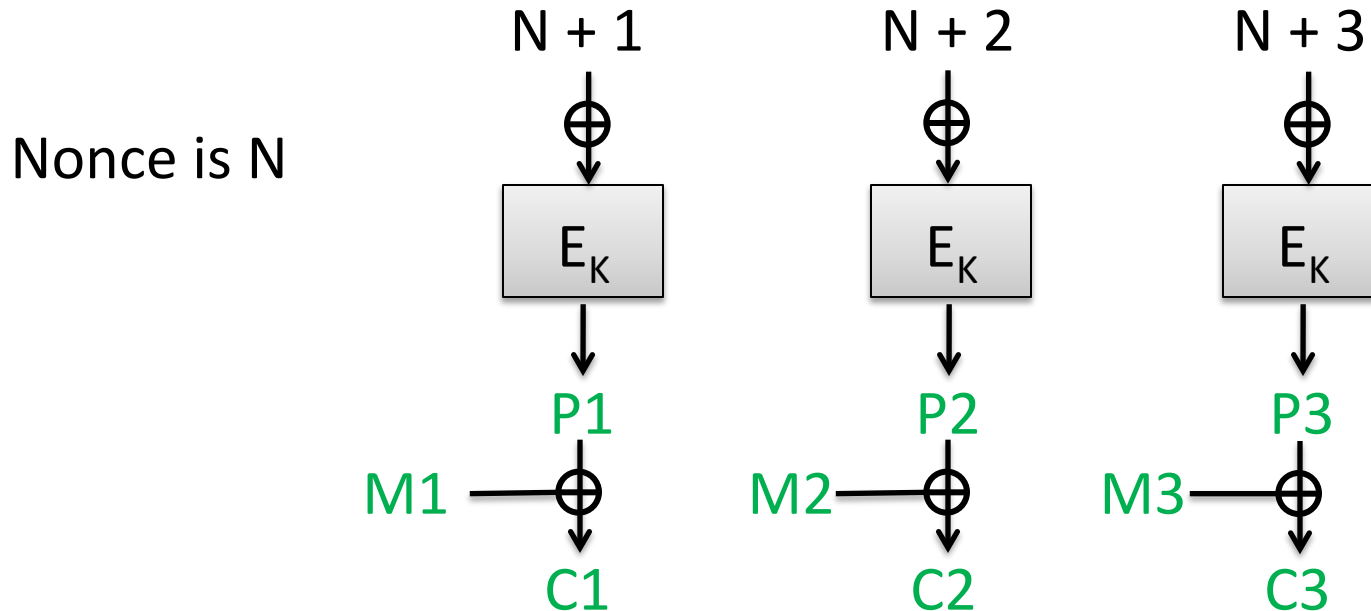
Also called deterministic AE (DAE) security

Ind\$
nonce-
respecting



Ind\$
repetition-
respecting

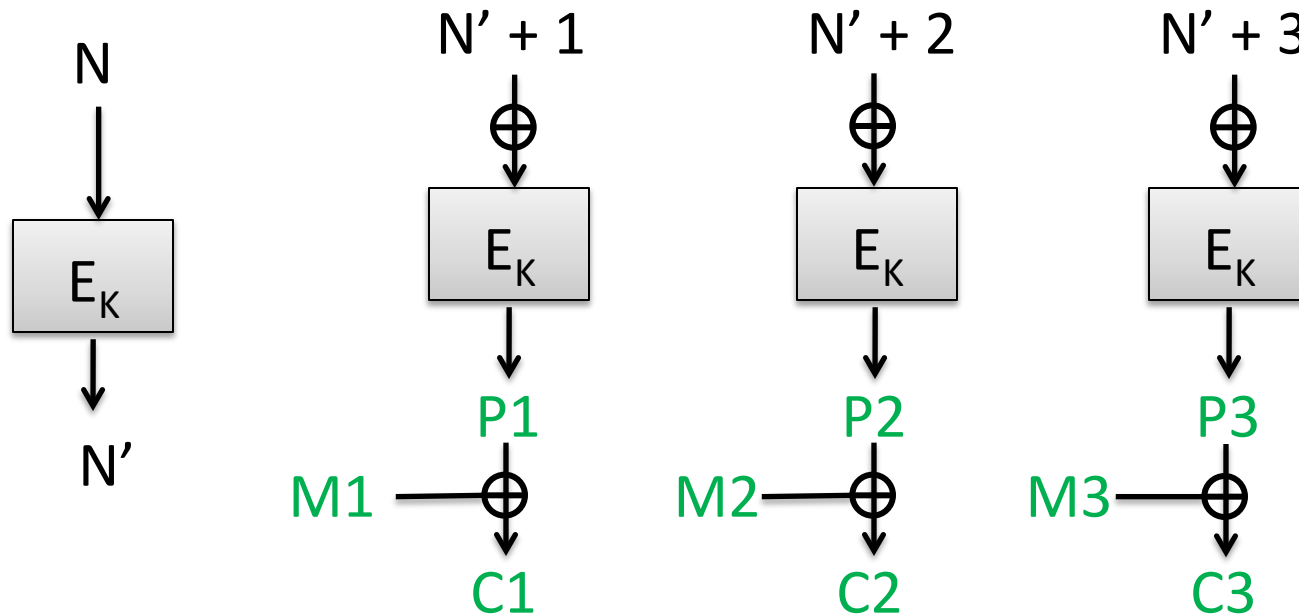
Example: CTR mode using nonces



Is this secure against nonce-respecting adversaries? No!

Query $C0, C1, C2 \leftarrow \text{Enc}(0^n, 0^{2n})$
Query $C0', C1', C2' \leftarrow \text{Enc}(0^{n-1}1, 0^{2n})$
If $C1' = C2$ then Ret 1
Ret 0

Example: CTR mode using nonces



Is this secure against nonce-respecting adversaries? Yes!

Is this secure against repetition-respecting adversaries? No!

```
Query  $C_0, C_1, C_2 \leftarrow \text{Enc}(0^n, 0^{2n})$   
Query  $C_0', C_1', C_2' \leftarrow \text{Enc}(0^n, 0^n 1^n)$   
If  $C_1' = C_1$  then Ret 1  
Ret 0
```

Confidentiality for nonce-based SE

- Nonce-respecting Ind\$:
 - Hides all partial info about plaintexts assuming nonces are never repeated
 - If nonces repeated security may break down
 - Traditional Ind\$/IND-CPA: choose nonce randomly
- Repetition-respecting Ind\$:
 - Also called deterministic AE security
 - Scheme is indistinguishable from a random injection
 - Implies nonce-respecting Ind\$
 - If nonces repeated security still pretty strong (just reveal repeat plaintexts)

Nonce-based SE CTXT integrity

Game CTXT_{SE}

$K \xleftarrow{\$} \{0, 1\}^k$
 $b' \xleftarrow{\$} \mathcal{A}^{\text{Enc, Forge}}$

ret false

$\text{Enc}(N, M)$

$C \leftarrow \text{enc}(K, N, M)$

$\mathcal{C} \leftarrow \mathcal{C} \cup \{(N, C)\}$

Ret C

$\text{Dec}(N, C)$

$M \leftarrow \text{dec}(K, N, M)$

If $(N, C) \notin \mathcal{C}$ and $M \neq \perp$ then

Ret true

Ret M

$$\text{Adv}_{\text{SE}}^{\text{ctxt}}(\mathcal{A}) = \Pr [\text{CTXT}_{\text{SE}}^{\mathcal{A}} \Rightarrow \text{true}]$$

Can't forge ciphertexts for new nonces

Provides authenticity

Is nonce-based CTR-mode secure against CTXT adversaries?

No!

Security for SE schemes

- Nonce-respecting Ind\$ + CTXT
 - Not as robust, but allows faster schemes
 - Example: OCB
- Repetition-respecting Ind\$ + CTXT
 - Stronger security target
 - Example: SIV
- Can also give equivalent all-in-one security definitions.

All-in-one notions

Game REAL_{SE}

$K \leftarrow_{\$} \{0, 1\}^k$
 $b' \leftarrow_{\$} \mathcal{A}^{\text{Enc}, \text{Dec}}$
 ret b'

$\text{Enc}(N, M)$
 $C \leftarrow \text{enc}(K, N, M)$
 ret C

$\text{Dec}(N, C)$
 Ret $\text{Dec}(N, C)$

Game RAND_{SE}

$K \leftarrow_{\$} \{0, 1\}^k$
 $b' \leftarrow_{\$} \mathcal{A}^{\text{Enc}}$
 ret b'

$\text{Enc}(N, M)$
 $C \leftarrow \text{enc}(K, N, M)$
 $C \leftarrow_{\$} \{0, 1\}^{|C|}$
 ret C

$\text{Dec}(N, C)$
 Ret $\perp$

See [Rogaway, Shrimpton 2006]

Simple strengthening of CCA security to also prevent forgeries

CTXT + Nonce-resp. Ind\$ $\Leftrightarrow$
 Nonce-resp. All-in-one Ind\$

CTXT + Repeat-resp. Ind\$ $\Leftrightarrow$
 Repeat-resp. All-in-one Ind\$

Nonce-respecting:

Never repeat N to Enc

Can't query $\text{Dec}(N, C)$ if (N, C) from $\text{Enc}(N, M)$

Repeat-respecting:

Never repeat (N, M) to Enc

Can't query $\text{Dec}(N, C)$ if (N, C) from $\text{Enc}(N, M)$

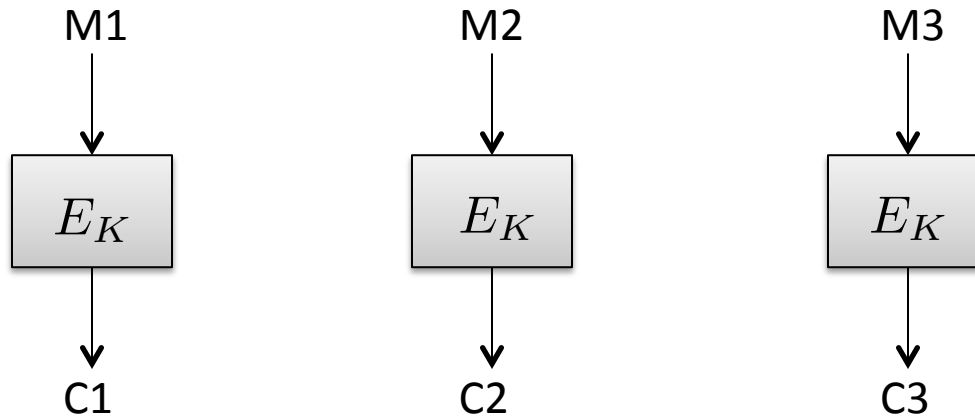
Reflection on security goals for SE

- 1 primitive, as many security properties as possible
 - Often we are treating orthogonal properties
 - CTXT (unforgeability), Ind\$ (confidentiality)
 - We can't build one scheme per property
- More simply:
 - target “robustness” / “defense-in-depth”

Outline

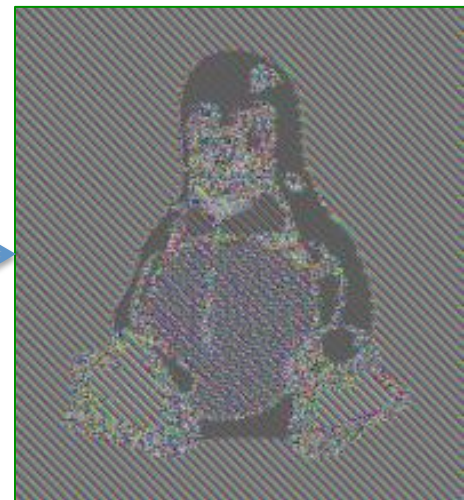
- Nonce-based Symmetric Encryption
 - Security notions
- OCB (Offset Codebook Mode)
- SIV (Synthetic IV) and Encode-then-encipher

Towards Nonce-based AE



Encrypted with ECB

Images courtesy
of Wikipedia

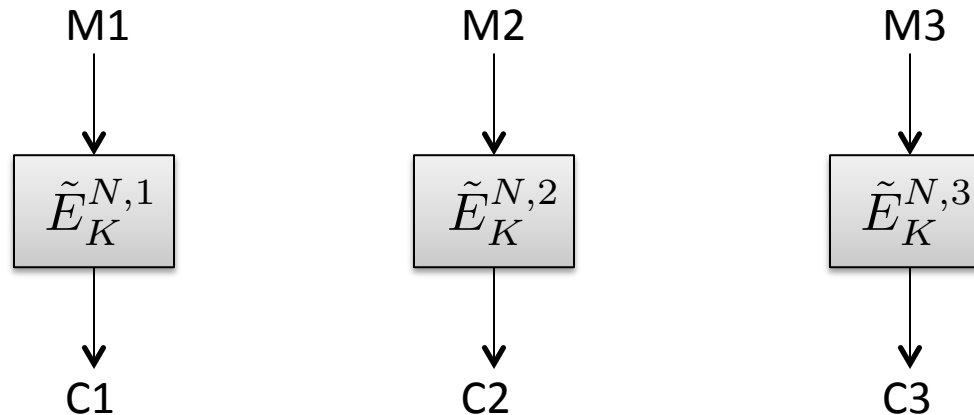


CPA-core of OCB mode

Tweakable block ciphers can fix all this!

Let N be an n -bit nonce

and use XE mode (or other suitable tweakable BC):



Remember our tweakable block cipher mode XE:

$$\mathcal{T} = \{0, 1\}^n \times [0..2^n - 2]$$

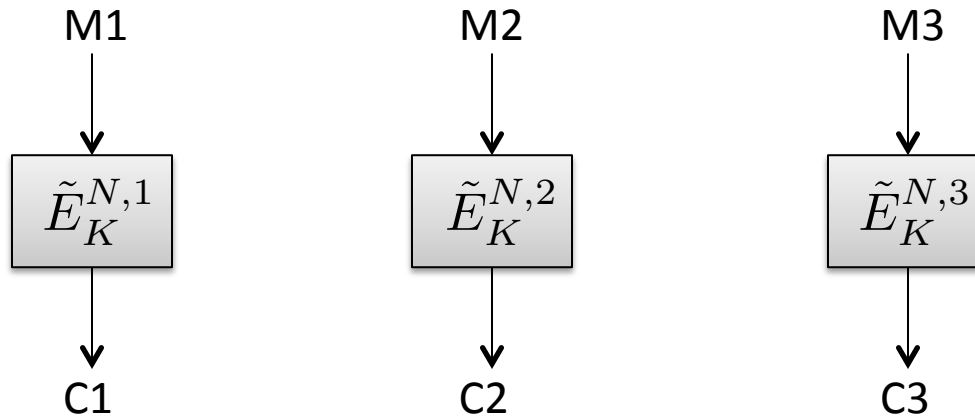
$$\tilde{E}(K, (T, i), M) = E_K(M \oplus \Delta) \quad \Delta = E_K(T) \cdot 2^i$$

CPA-core of OCB mode

Tweakable block ciphers can fix all this!

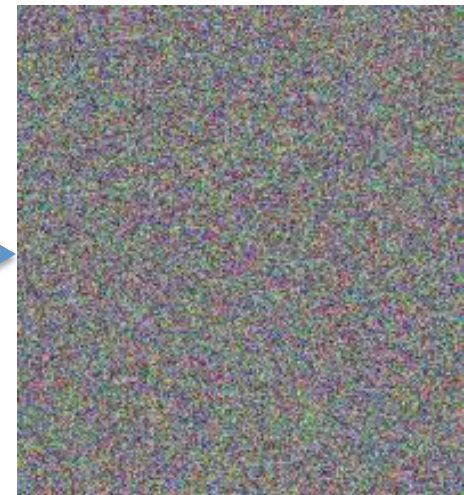
Let N be an n -bit nonce

and use XE mode (or other suitable tweakable BC):



Encrypted with OCB

Images courtesy
of Wikipedia



Nonce-resp. Ind\$ of OCB CPA-core

Theorem. *Let OCB be the SE scheme defined above using a tweakable block cipher $\tilde{E} : \{0, 1\}^k \times \mathcal{T} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ with $\mathcal{T} = \{0, 1\}^n \times [0..2^n - 2]$. Then for any nonce-respecting adversary $\mathcal{A}$ making q oracle queries with m -block messages and running in time t we give an explicit adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}_{\text{OCB}}^{\text{ind\$}}(\mathcal{A}) \leq \mathbf{Adv}_{\tilde{E}}^{\text{tprp}}(\mathcal{B})$$

$\mathcal{B}$ runs in time $t + \mathcal{O}(mq)$ and makes mq queries.

Standard reduction to switch from real TBC to family of random permutations

As long as nonces never repeat, then tweak inputs are unique to every use, selecting distinct random permutations

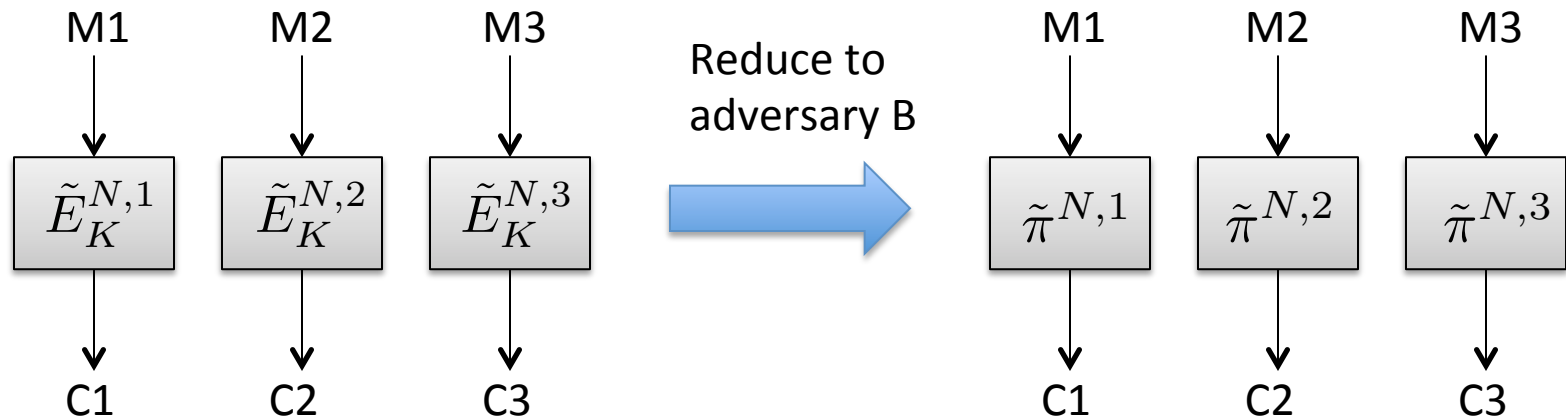
Outputs are uniformly random

Nonce-resp. Ind\$ of OCB CPA-core

Theorem. Let OCB be the SE scheme defined above using a tweakable block cipher $\tilde{E} : \{0, 1\}^k \times \mathcal{T} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ with $\mathcal{T} = \{0, 1\}^n \times [0..2^n - 2]$. Then for any nonce-respecting adversary $\mathcal{A}$ making q oracle queries with m -block messages and running in time t we give an explicit adversary $\mathcal{B}$ such that

$$\mathbf{Adv}_{\text{OCB}}^{\text{ind\$}}(\mathcal{A}) \leq \mathbf{Adv}_{\tilde{E}}^{\text{tprp}}(\mathcal{B})$$

$\mathcal{B}$ runs in time $t + \mathcal{O}(mq)$ and makes mq queries.



Each random perm used on at most one point
C1, C2, C3 all independent uniform points

How to get CTXT integrity?

Game CTXT_{SE}

$K \leftarrow_{\$} \{0, 1\}^k$
 $b' \leftarrow_{\$} \mathcal{A}^{\text{Enc, Forge}}$

ret false

$\text{Enc}(N, M)$

$C \leftarrow \text{enc}(K, N, M)$

$\mathcal{C} \leftarrow \mathcal{C} \cup \{(N, C)\}$

Ret C

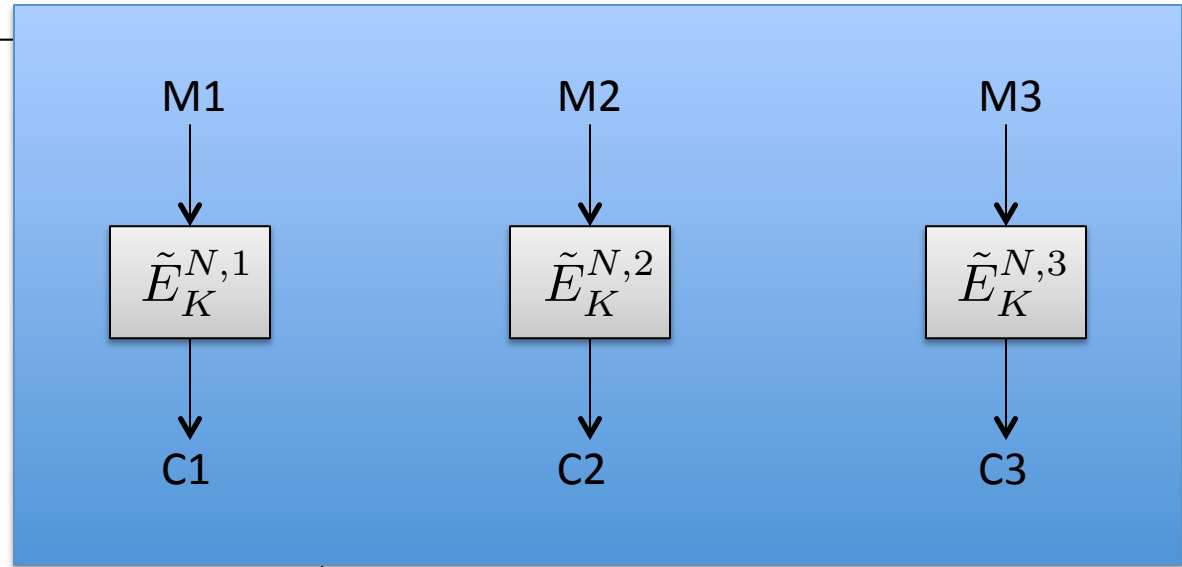
$\text{Dec}(N, C)$

$M \leftarrow \text{dec}(K, N, M)$

If $(N, C) \notin \mathcal{C}$ and $M \neq \perp$ then

Ret true

Ret M



Trivially CTXT insecure as every $3n$ -bit string is a valid ciphertext

Ideas for getting CTXT?

How to get CTXT integrity?

Game CTXT_{SE}

$K \leftarrow_{\$} \{0, 1\}^k$

$b' \leftarrow_{\$} \mathcal{A}^{\text{Enc, Forge}}$

ret false

$\text{Enc}(N, M)$

$C \leftarrow \text{enc}(K, N, M)$

$\mathcal{C} \leftarrow \mathcal{C} \cup \{(N, C)\}$

Ret C

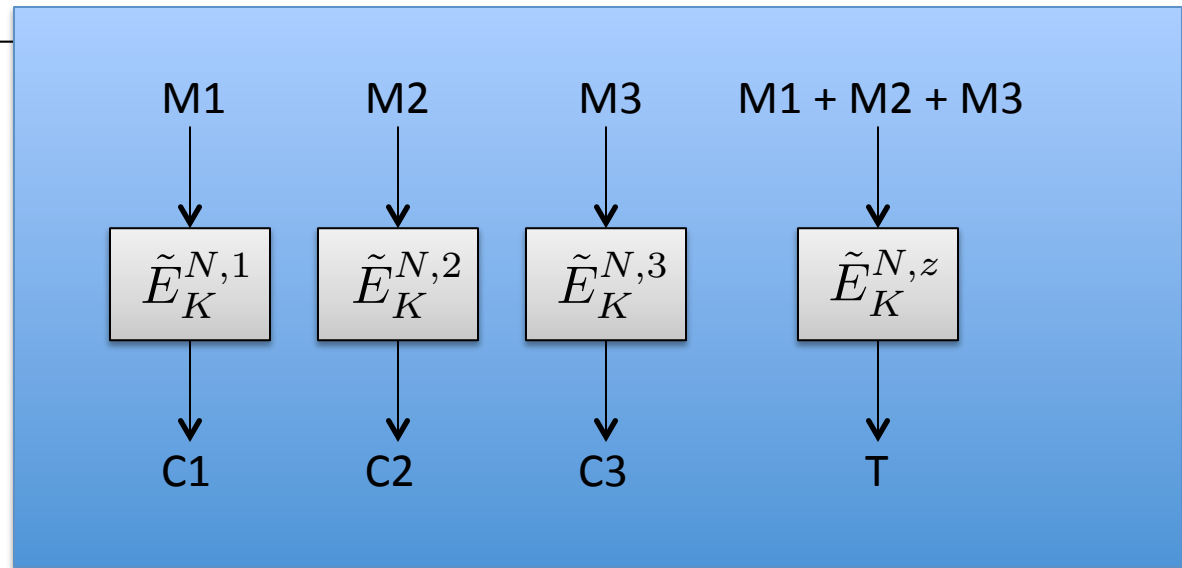
$\text{Dec}(N, C)$

$M \leftarrow \text{dec}(K, N, M)$

If $(N, C) \notin \mathcal{C}$ and $M \neq \perp$ then

Ret true

Ret M



Add integrity checksum.

Tweak N,z used for checksum call must be distinct from all other uses of TBC

Here M1 + M2 + M3 is XOR of message blocks.

Why not C1 + C2 + C3? (Insecure)

How to get CTXT integrity?

Game CTXT_{SE}

$K \leftarrow_{\$} \{0, 1\}^k$
 $b' \leftarrow_{\$} \mathcal{A}^{\text{Enc, Forge}}$

ret false

$\text{Enc}(N, M)$

$C \leftarrow \text{enc}(K, N, M)$

$\mathcal{C} \leftarrow \mathcal{C} \cup \{(N, C)\}$

Ret C

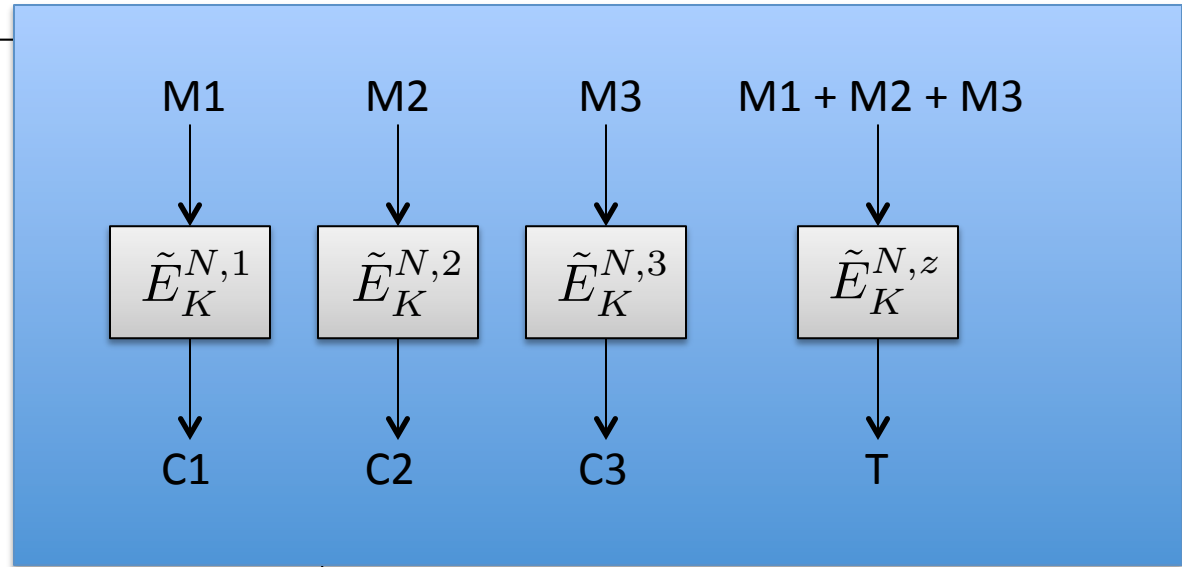
$\text{Dec}(N, C)$

$M \leftarrow \text{dec}(K, N, M)$

If $(N, C) \notin \mathcal{C}$ and $M \neq \perp$ then

Ret true

Ret M



New N means:

inverse of T random

New C means:

one of M1, M2, M3, inverse of T random

Either way, low prob. of hitting a correct T

How to get CTXT integrity?

Theorem. *Let OCB be the SE scheme defined above using a tweakable block cipher $\tilde{E} : \{0, 1\}^k \times \mathcal{T} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ with $\mathcal{T} = \{0, 1\}^n \times [0..2^n - 2]$. Then for any adversary $\mathcal{A}$ making q oracle queries with m -block messages and running in time t we give an explicit adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}_{\text{OCB}}^{\text{ctxt}}(\mathcal{A}) \leq \mathbf{Adv}_{\tilde{E}}^{\text{tsprp}}(\mathcal{B}) + \frac{1}{2^n - 1}$$

$\mathcal{B}$ runs in time $t + \mathcal{O}(mq)$ and makes mq queries.

I described simplified OCB: only works with fixed-length messages of mn bits
(don't implement as I've described!)

Full OCB handles:

- Messages that aren't multiple of n bits
- Uses slightly more complex XEX TBC construction for super fast tweaking
- See [Rogaway 2004]

OCB repetition-resp. Ind\$ Security?

Game REAL_{SE}

$K \leftarrow_{\$} \{0, 1\}^k$

$b' \leftarrow_{\$} \mathcal{A}^{\text{Enc}}$

ret b'

$\text{Enc}(N, M)$

$C \leftarrow_{\$} \text{enc}(\bar{K}, N, M)$

Ret C

Game RAND_{SE}

$K \leftarrow_{\$} \{0, 1\}^k$

$b' \leftarrow_{\$} \mathcal{A}^{\text{Enc}}$

ret b'

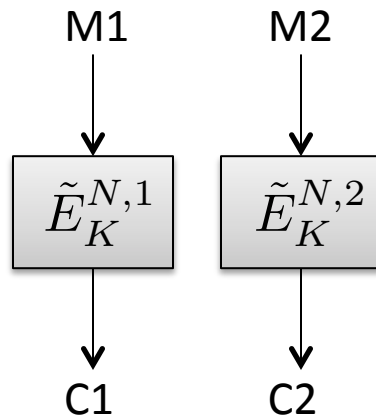
$\text{Enc}(N, M)$

$C \leftarrow_{\$} \text{enc}(\bar{K}, N, M)$

$C \leftarrow_{\$} \{0, 1\}^{|C|}$

Ret C

Repetition-respecting adversaries: all queries unique (N,M)



Adversary $\mathcal{A}$:

$C1, C2 \leftarrow \text{Enc}(N, 0^n 0^n)$

$C1', C2' \leftarrow \text{Enc}(N, 0^n 1^n)$

If $C1 = C1'$ then Ret 1

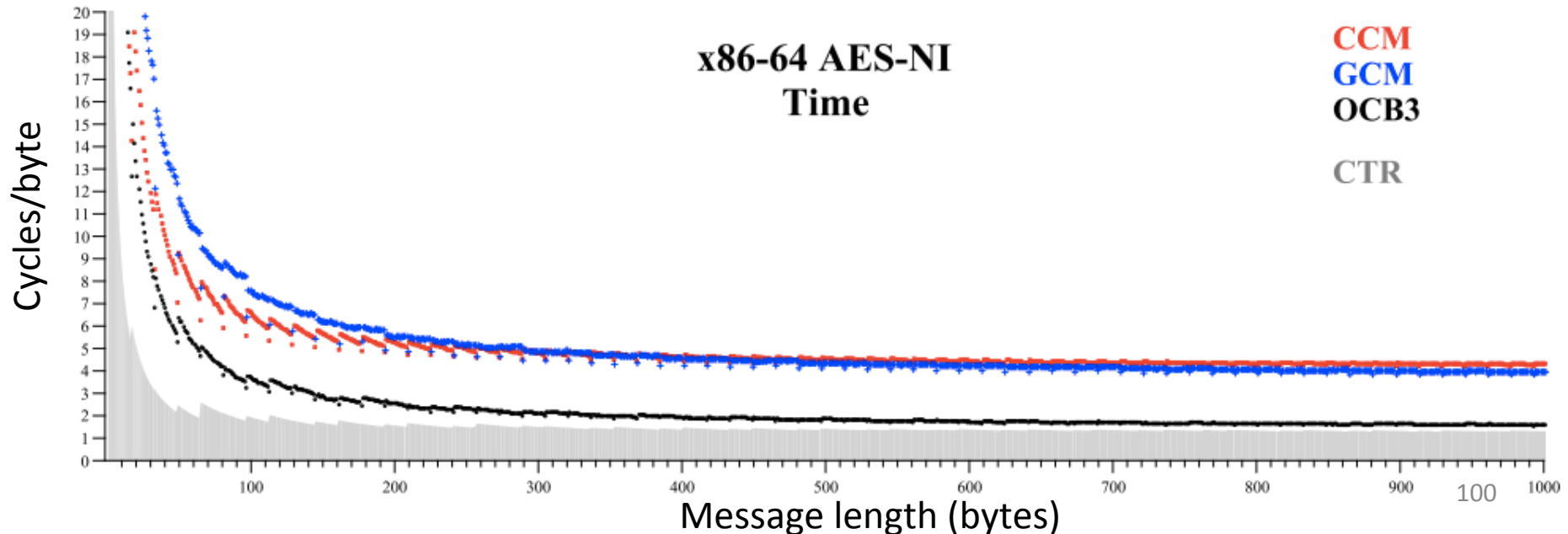
Ret 0

$$\text{Adv}_{\text{OCB}}^{\text{ind\$}}(\mathcal{A}) = 1 - \frac{1}{2^n}$$

Recap of OCB

- OCB gives fast, single pass nonce-respecting scheme
- OCB doesn't give repetition-resp. security
- OCB remains fastest AES-based AE scheme

From [Krovetz, Rogaway 2011]:

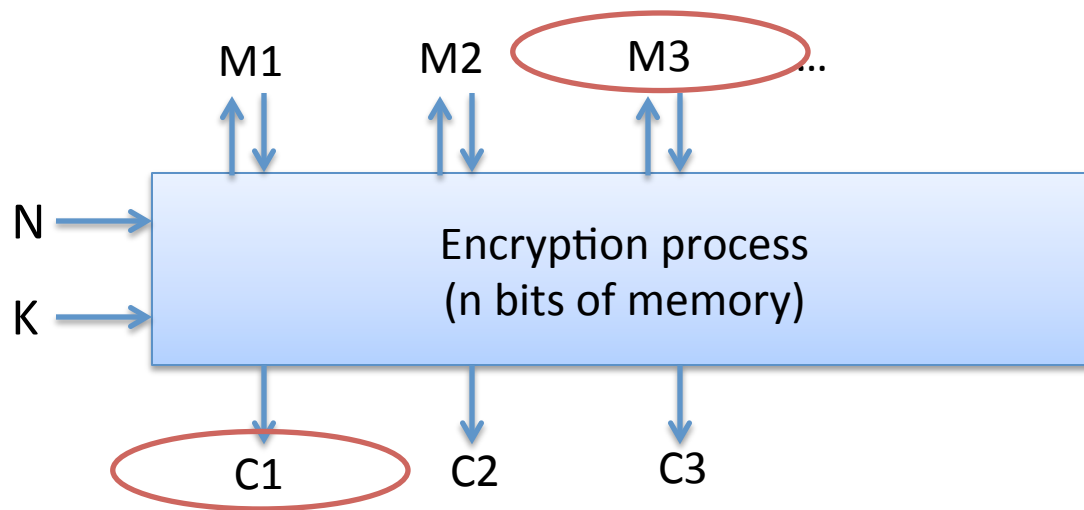


Outline

- Nonce-based Symmetric Encryption
 - Security notions
- OCB (Offset Codebook Mode)
- SIV (Synthetic IV) and Encode-then-encipher

Single pass can't give repetition-respecting security

Single pass means process each message block once, constant memory state



Game REAL_{SE}

$K \leftarrow_{\$} \{0, 1\}^k$

$b' \leftarrow_{\$} \mathcal{A}^{\text{Enc}}$

ret b'

$\text{Enc}(N, M)$

$C \leftarrow_{\$} \text{enc}(K, N, M)$

Ret C

Game RAND_{SE}

$K \leftarrow_{\$} \{0, 1\}^k$

$b' \leftarrow_{\$} \mathcal{A}^{\text{Enc}}$

ret b'

$\text{Enc}(N, M)$

$C \leftarrow_{\$} \text{enc}(K, N, M)$

$C \leftarrow_{\$} \{0, 1\}^{|C|}$

Ret C

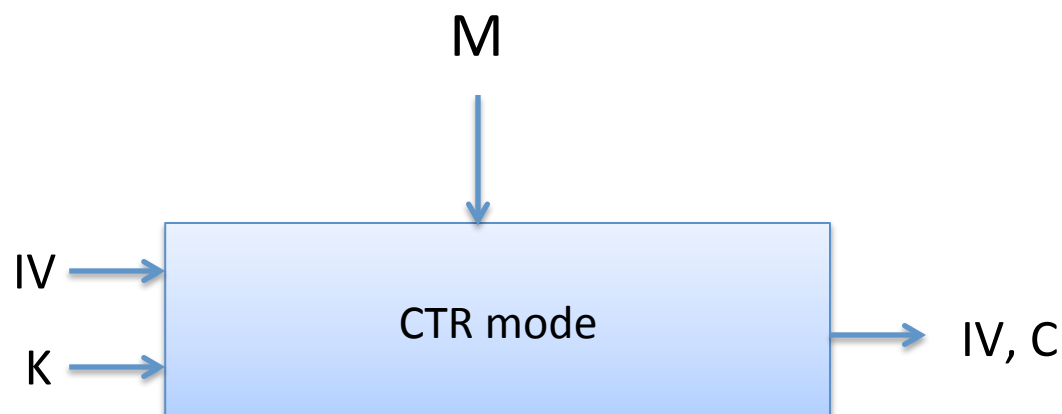
Must output ciphertext bits before processing all message bits
BUT: Can never be repetition-resp Ind\$ secure!!!

Two-pass modes

- Single-pass mode can't achieve repetition-resp. Ind\$
- We will see 2 two-pass modes
 - SIV mode
 - Encode-then-Encipher

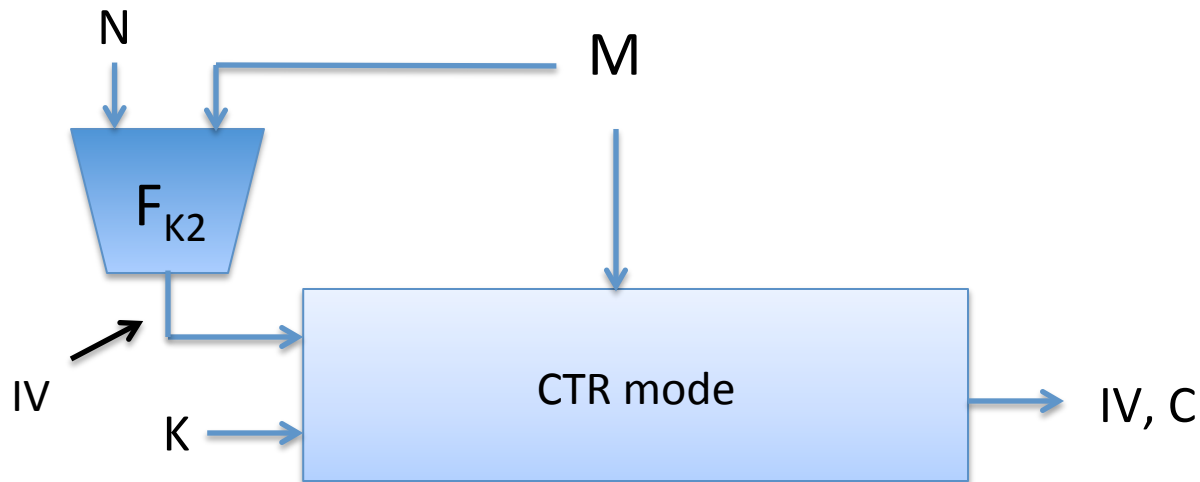
Two-pass construction (SIV mode)

[Rogaway, Shrimpton 2006]



Two-pass construction (SIV mode)

[Rogaway, Shrimpton 2006]



Apply PRF to nonce and message to derive new nonce for use with encryption

Decryption: check the IV included in C

Intuition for Ind $\$$ security:

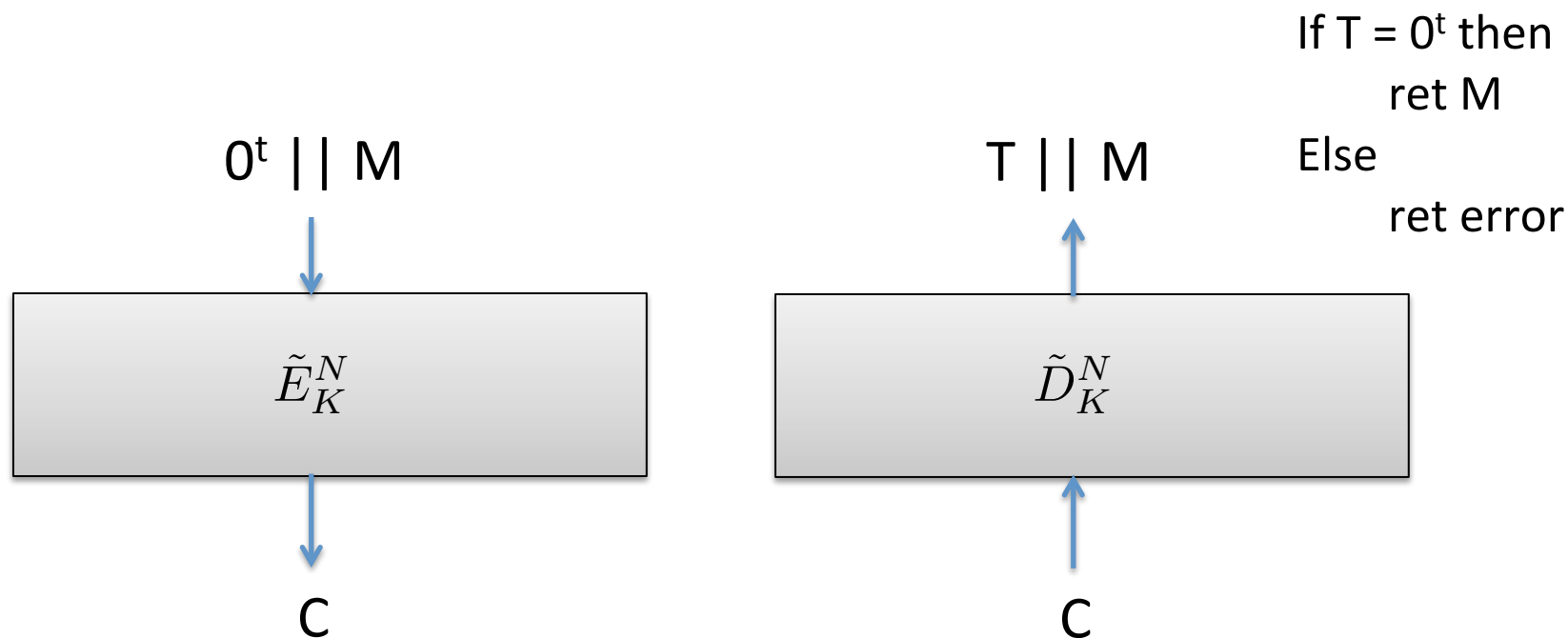
N, M never repeat, guaranteed random IV fed to encryption

Intuition for CTXT security:

change to any ciphertext component changes M or IV

Encode-then-encipher

Say we have “wide-block” tweakable BC secure as SPRP that accepts arbitrary-length inputs



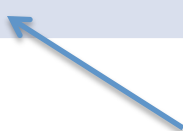
Original analysis (without tweaks) in [Bellare, Rogaway 1997]
Intuition: TBC randomizes everything, low probability of hitting 0^t
TBC on long inputs are all two-pass (we will see examples soon)

Any security difference between SIV and Encode-Encipher?

- Both meet Repetition-resp. Ind\$ + CTXT
 - What do you think?
- Encode-Encipher also provides Strong PRP
 - This seems to add robustness
- Crisply clarifying this issue is a good question for future research

A high level summary

Scheme	# of cryptographic passes	Security
OCB	1	Nonce-resp.
SIV	2	Repeat-resp.
Encode-then-Encipher	3	Strong PRP / Repeat-resp.



We'll see this in next lecture

Recap

- Make SE more robust
 - Any nonce works (don't need random one)
 - Nonce failures should be handled gracefully
 - Possible with repetition-resp. security
 - SE based on strong PRP may be even better
- Target easy-to-use interfaces
 - `Encrypt()`, `Mac()` versus `Encrypt(N,M)`
- Tweakable PRPs good starting point for building such SE schemes

Lecture plan

1. Tweakable PRPs and shuffling
2. Nonce-based symmetric encryption
3. Format-preserving encryption
4. Further symmetric primitives

Lecture 3:

Format-preserving encryption and special cases

Thomas Ristenpart
University of Wisconsin

Lecture plan

1. Tweakable PRPs and shuffling
2. Nonce-based symmetric encryption
3. Format-preserving encryption
4. Further symmetric primitives

Outline

- Motivation for FPE
- Formalization of FPE
- Rank-Encipher-Unrank
 - Ranking for regular expressions
 - Integer FPE
- Integer FPE schemes for small domains
- Disk-sector / FPE schemes for large domains

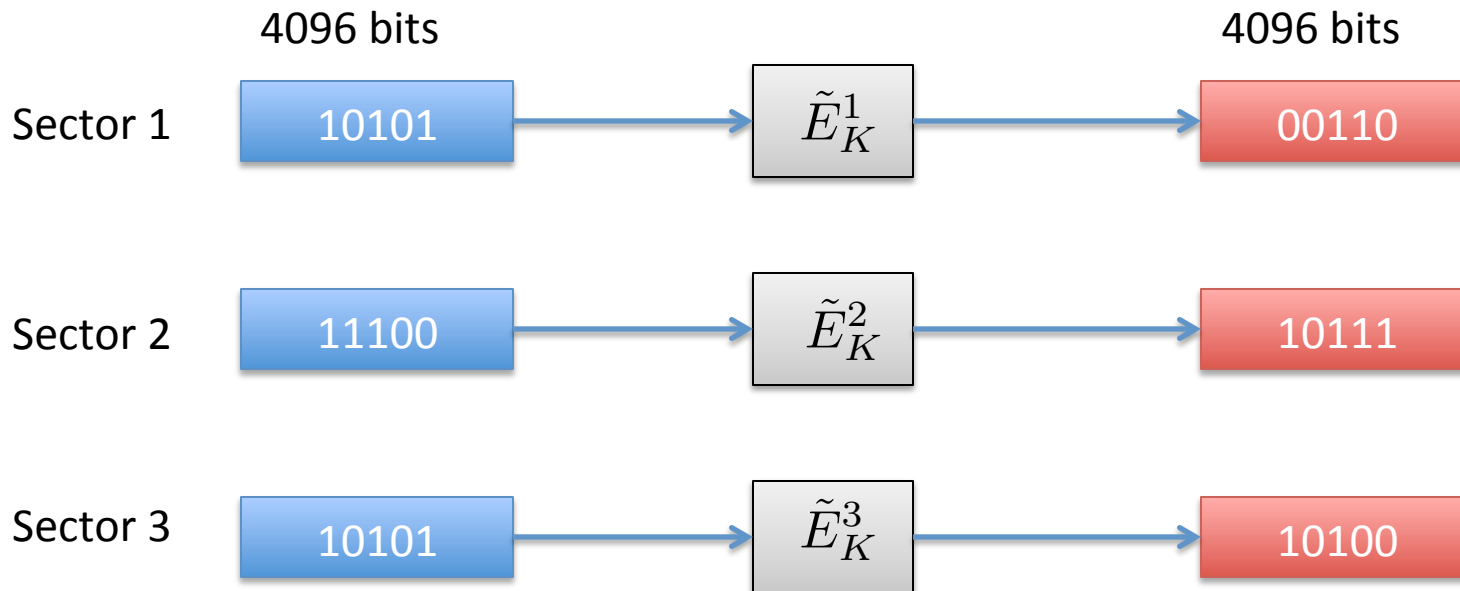
Example #1: Disk sector encryption

Say we want to do full-disk encryption sector-by-sector

Plaintext = 4096 bytes

Ciphertext = 4096 bytes

How to build TBC on 4096 bits
given BC with $n = 128$?



Example #2: Data field encryption

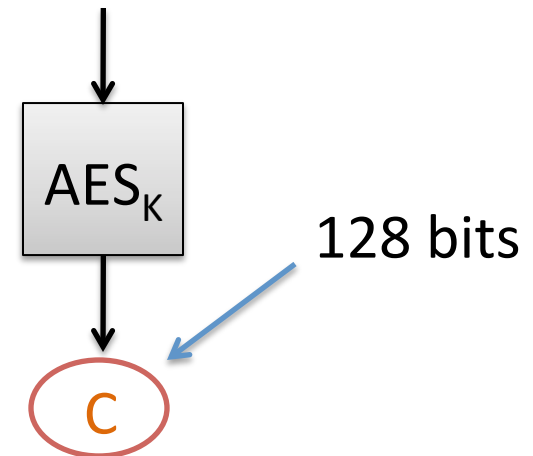
Jane Doe	1343-1321-1231-2310
Thomas Ristenpart	9541-3156-1320-2139
John Jones	5616-2341-2341-1210
Eve Judas	2321-4232-1340-1410

← Database schemas and software require ≤ 16 decimal digits and valid Luhn checksum

$$\text{AES}_K : \{0,1\}^{128} \longrightarrow \{0,1\}^{128}$$

Ciphertexts are too big for replacing plaintext within database!

$M = 2321-4232-1345-1415$



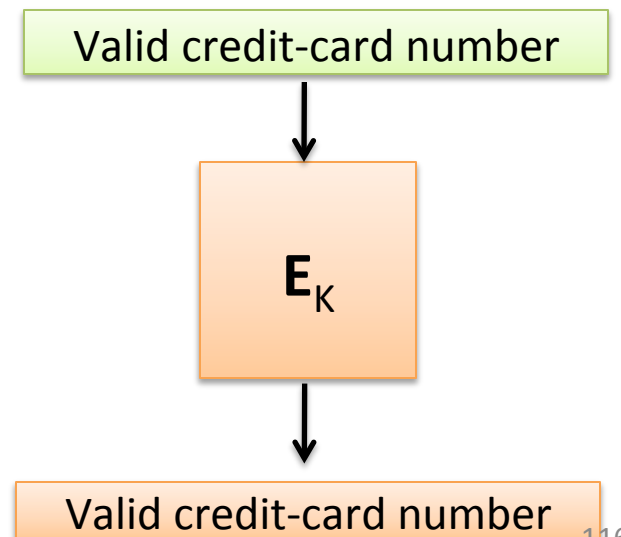
Example #2: Data field encryption

Jane Doe	
Thomas Ristenpart	
John Jones	
Eve Judas	

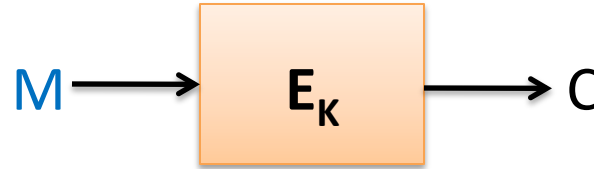
← Database schemas and software require ≤ 16 decimal digits and valid Luhn checksum

Encryption tool whose **ciphertexts** are also credit-card numbers

$$E_K : [0..9]^{16} \rightarrow [0..9]^{16}$$



Format-preserving encryption (FPE)



Enjoys format **N**

Enjoys format **N**

Disk sectors / payment card numbers just two examples
Some others:

- 1) Valid addresses for a certain country
- 2) 4096-byte disk sectors
- 3) Assigned Social Security Numbers (9 digits, without leading 0 or 9)
- 4) Composition of (1) and (3)

History of FPE:

FIPS 74 (1981) --- DES to encipher strings over some fixed alphabet (e.g. decimal digits)

Brightwell and Smith (1997) --- datatype-preserving encryption

Hasty pudding cipher (1998) --- variable block-length blockcipher

Ad-hoc
approaches

Black, Rogaway (2002) --- special case of FPE (arbitrary set encryption)

Voltage, Semtek, etc. --- promote use of commercial FPE technologies

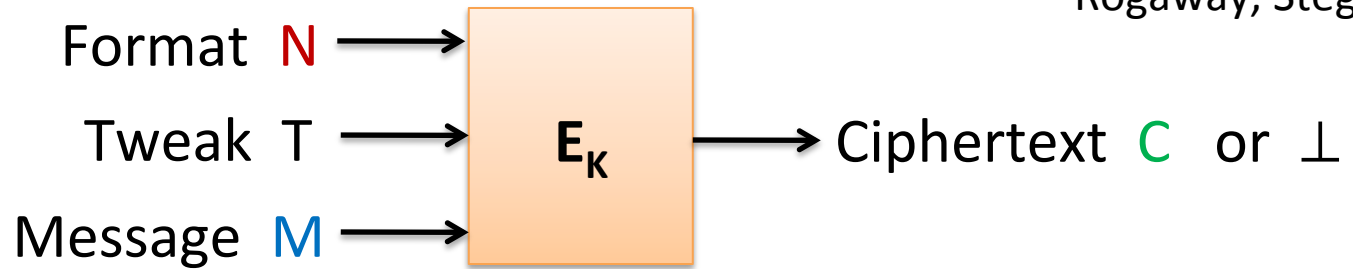
Spies (2008) --- FFSEM submitted to NIST as potential standard for FPE. Based on Black, Rogaway techniques

Bellare, Ristenpart, Rogaway, Stegers (2009) --- formal treatment of FPE

Bellare, Rogaway, Spies (2009) --- FFX proposed standard submitted to NIST

Format-preserving encryption (FPE)

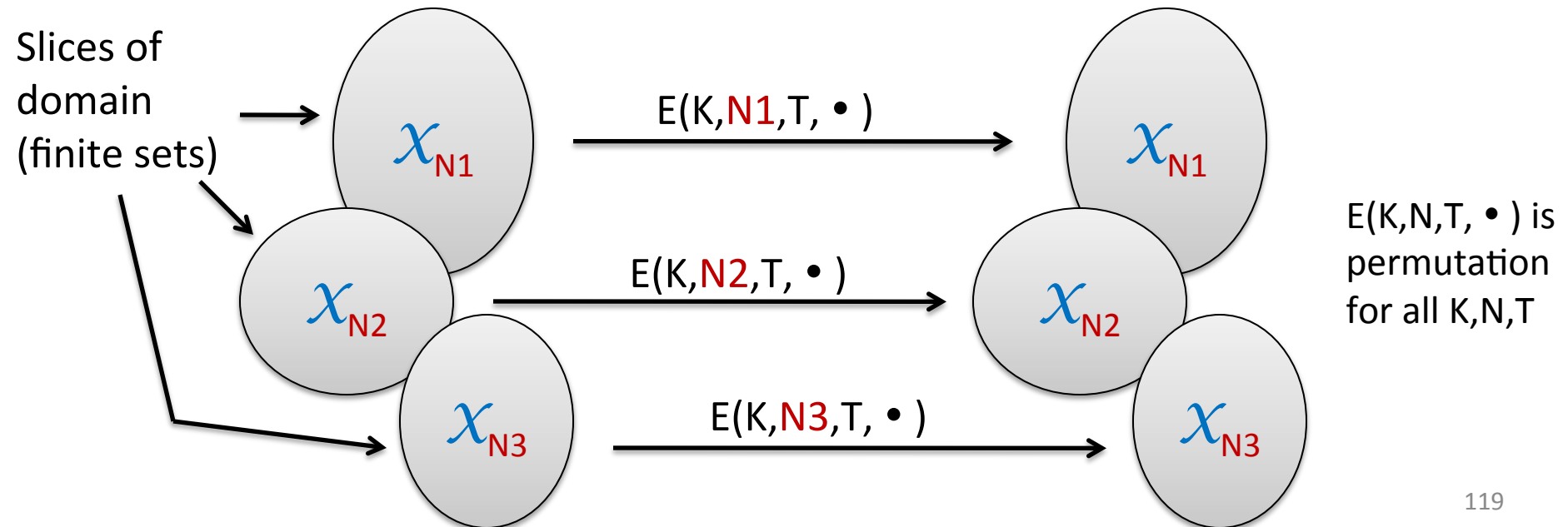
[Bellare, Ristenpart,
Rogaway, Stegers '09]



$\mathcal{N} = \{N_1, N_2, \dots\}$ is format space (set of formats handled)

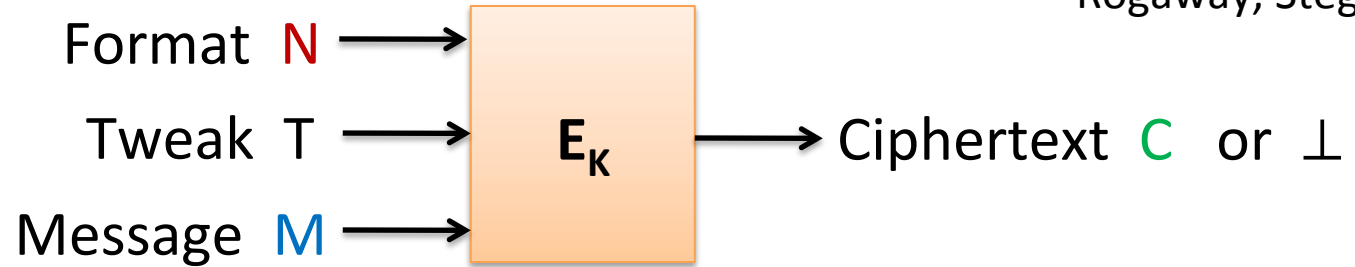
Domain is $\mathcal{X} = \{x_N\}_{N \in \mathcal{N}}$

Tweaks [LRW05] required



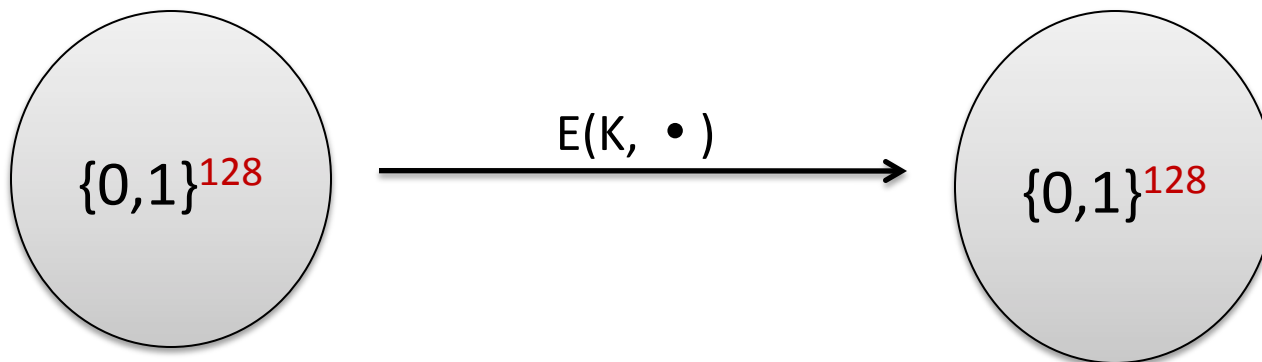
Format-preserving encryption (FPE)

[Bellare, Ristenpart,
Rogaway, Stegers '09]



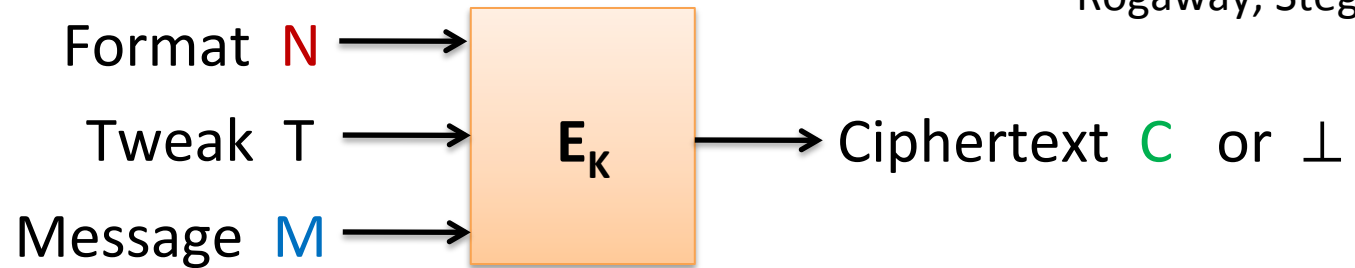
AES is an FPE (with singleton tweak space)

Domain is $\mathcal{X} = \{x_{128}\}$ where $x_{128} = \{0,1\}^{128}$



Format-preserving encryption (FPE)

[Bellare, Ristenpart,
Rogaway, Stegers '09]

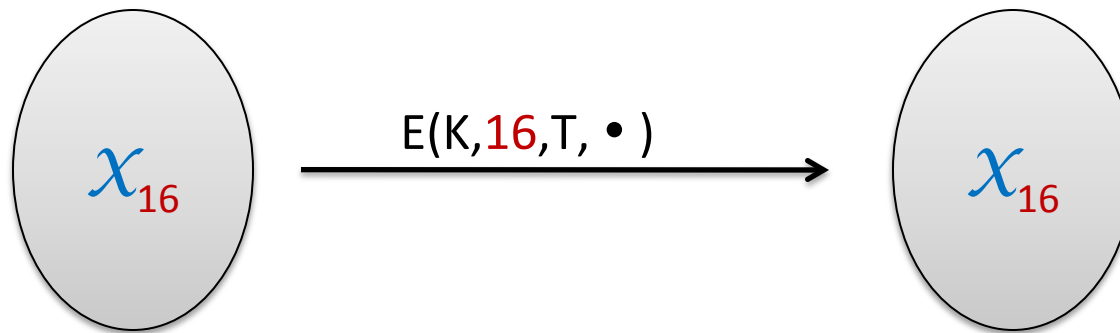


Encrypting Credit Card Numbers (CCNs) with valid Luhn digit

Domain is $\mathcal{X} = \{x_N\}_{N \in \mathcal{N}}$ where $\mathcal{N} = \{12, 13, 14, 15, \dots, 20\}$

$x_N = \{X \in \{0, 1, \dots, 9\}^N \mid \text{LuhnOK}(X)\}$

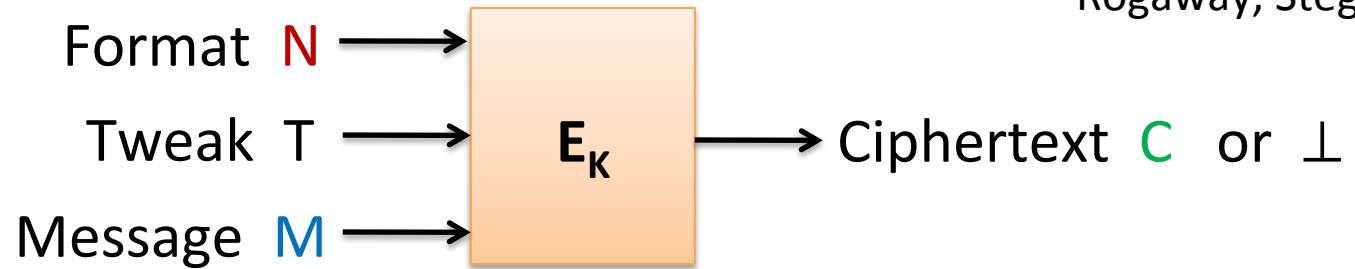
Example
slice:



All 16-digit CCNs
with valid Luhn digit

Format-preserving encryption (FPE)

[Bellare, Ristenpart,
Rogaway, Stegers '09]



Encrypting members of a **regular language** L

Domain is $\mathcal{X} = \{\mathcal{X}_N\}_{N \in \mathcal{N}}$ where $\mathcal{N} = \{0, 1, 2, 3, \dots\}$
 $\mathcal{X}_N = \{0, 1\}^N \cap L$

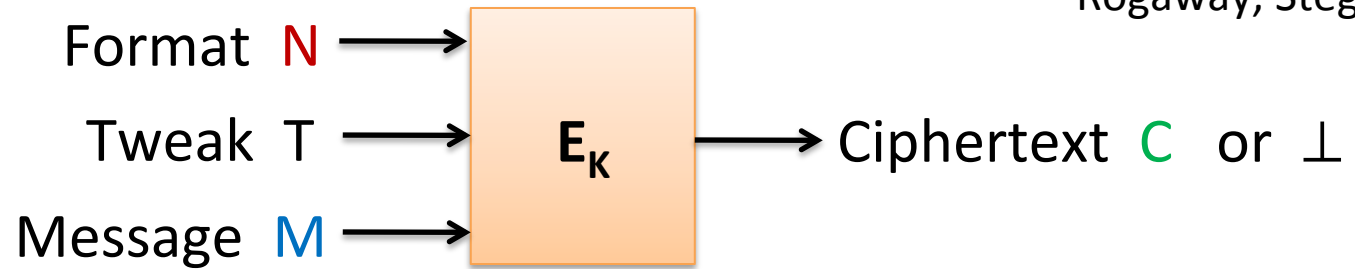
Example
slice:



All 130-bit strings in L

Format-preserving encryption (FPE)

[Bellare, Ristenpart,
Rogaway, Stegers '09]

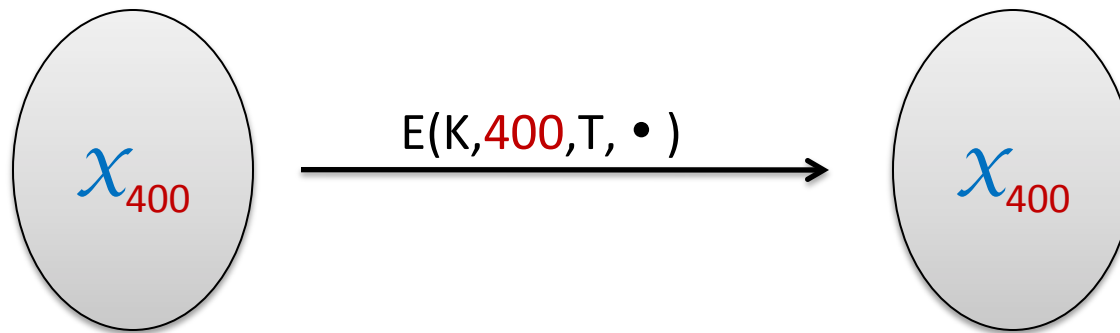


Encrypting **integers**

We call this Integer FPE

Domain is $\mathcal{X} = \{x_N\}_{N \in \mathcal{N}}$ where $\mathcal{N} = \{1, 2, 3, \dots\}$
 $x_N = \mathbb{Z}_N$

Example
slice:



Any number between 0 and 399

Security for FPE

Game $\text{SPRP1}_{\hat{E}}$

$K \leftarrow_{\$} \{0, 1\}^k$
 $b' \leftarrow_{\$} \mathcal{A}^{\hat{E}_K(\cdot, \cdot, \cdot), \hat{D}_K(\cdot, \cdot, \cdot)}$
ret b'

Game $\text{SPRP0}_{\mathcal{X}, \mathcal{T}}$

for $(N, T) \in \mathcal{X} \times \mathcal{T}$
 $\hat{\pi}(N, T, \cdot) \leftarrow_{\$} \text{Perm}(\mathcal{X}_N)$
 $b' \leftarrow_{\$} \mathcal{A}^{\hat{\pi}(\cdot, \cdot, \cdot)}$
ret b'

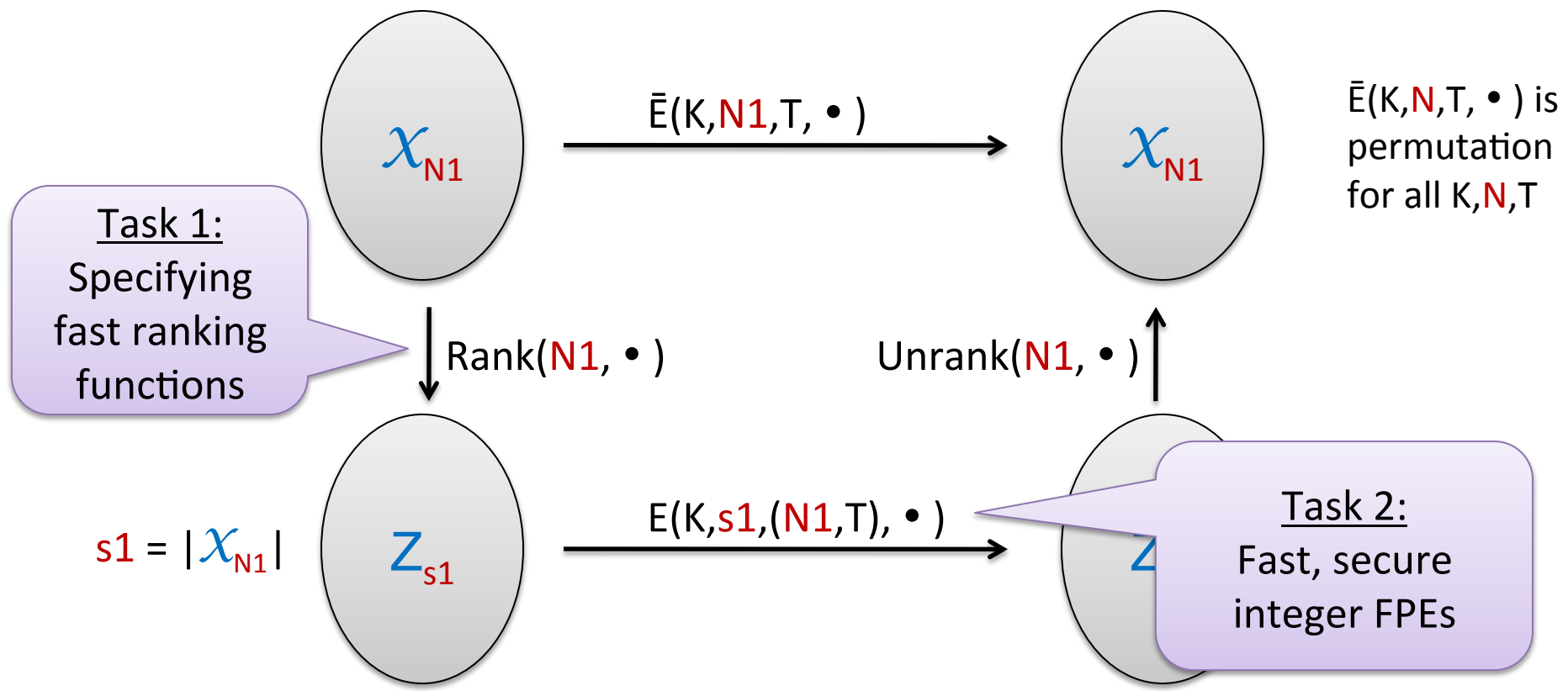
$$\mathbf{Adv}_{\hat{E}}^{\text{prp}}(\mathcal{A}) = \left| \Pr [\text{SPRP1}_{\hat{E}}^{\mathcal{A}} \Rightarrow 1] - \Pr [\text{SPRP0}_{\mathcal{X}, \mathcal{T}}^{\mathcal{A}} \Rightarrow 1] \right|$$

- Allows adversarial format, tweaks
- PRP security variant disallows inverse queries
- Attacker can detect repeats if same tweak, format, message
- Special case is strong tweakable PRP security we've already seen

General construction of FPE schemes: Rank-then-Encipher

$\mathcal{N} = \{N1, N2, \dots\}$ some arbitrary format space

Domain is $\mathcal{X} = \{x_N\}_{N \in \mathcal{N}}$



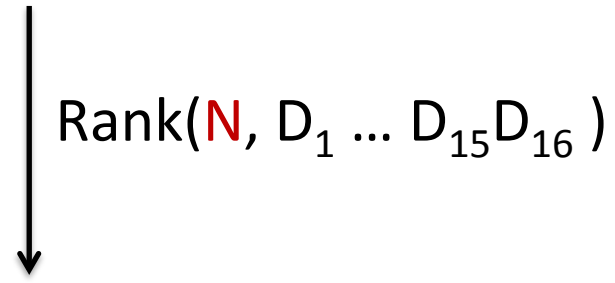
Just need integer FPE for $\mathcal{N} = \{s1, s2, \dots\}$ where $s1 = |x_{N1}|$, $s2 = |x_{N2}|$, ...

$\bar{E}$ inherits security of E . Proof is easy

Specialized constructions

Last digit of CC number
is checksum fully determined
by preceding 15 digits

$$X_N = \{ X \in \{0,1,\dots,9\}^{16} \mid \text{LuhnOK}(X) \}$$



- 1) Compute number **X** that is base-10 represented by $D_1 \dots D_{15}$
- 2) Return **X**

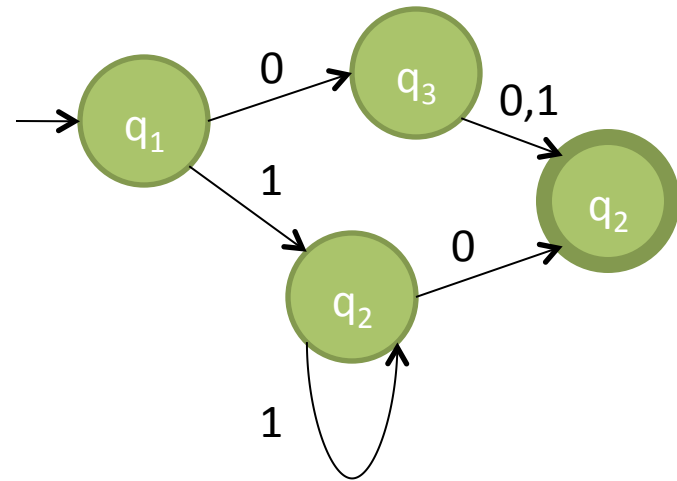
$$Z_{10^{15}}$$



- 1) Compute base-10 representation of **X** as $D_1 \dots D_{15}$
- 2) Solve for Luhn digit D_{16}
- 3) Return $D_1 \dots D_{15} D_{16}$

$$X_N$$

General constructions for regular languages



Regular languages naturally describe many formatting constraints!

$\mathcal{X} = L$ for any regular language L

$\mathcal{N} = \{0, 1, 2, 3, \dots\}$

Let $\mathcal{X}_N = \{0, 1\}^N \cap L$

[BRRS09] gives polytime
 $\text{Rank}(\mathcal{N}, \bullet)$, $\text{Unrank}(\mathcal{N}, \bullet)$
 following [Goldberg, Sipser '85]
 Based on dynamic programming

Requires DFA
 representation of L .
 Starting from regex is
 PSPACE-complete

Q = number of DFA states

Σ = the alphabet

$N_{\max}$ = length of longest string

N = length of string to be ranked

Precomputation:

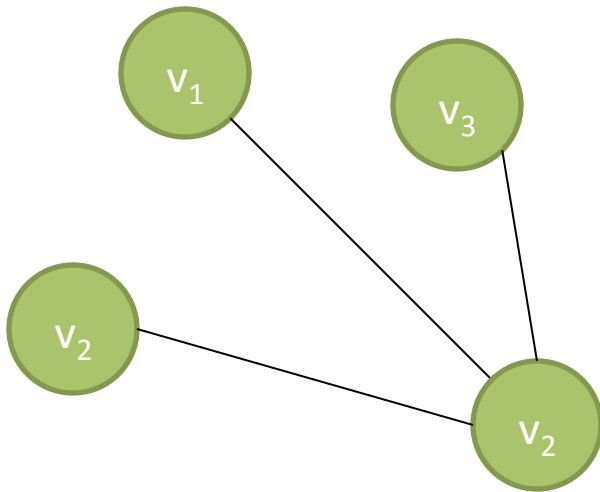
$c \cdot N_{\max} \cdot |\Sigma| \cdot Q$ time, $c \cdot N_{\max} \cdot Q$ space

rank: $O(N)$

unrank: $O(N \cdot |\Sigma|)$

Limits of Rank-then-Encipher approach

Theorem [BRRS 09] Suppose there exists a one-way function. Then there is a domain $\mathcal{X} = \{\mathcal{X}_N\}$ that admits a PRP-secure FPE scheme but for which $\{\mathcal{X}_N\}$ cannot be efficiently ranked.



$\mathcal{N}$ = simple, undirected graphs G

$\mathcal{X}_N$ = all proper k -colorings of G for $k = 2d+1$
(d is max degree of any vertex)

[Bubley et al. '99] prove that counting number of k -colorings is #P-complete



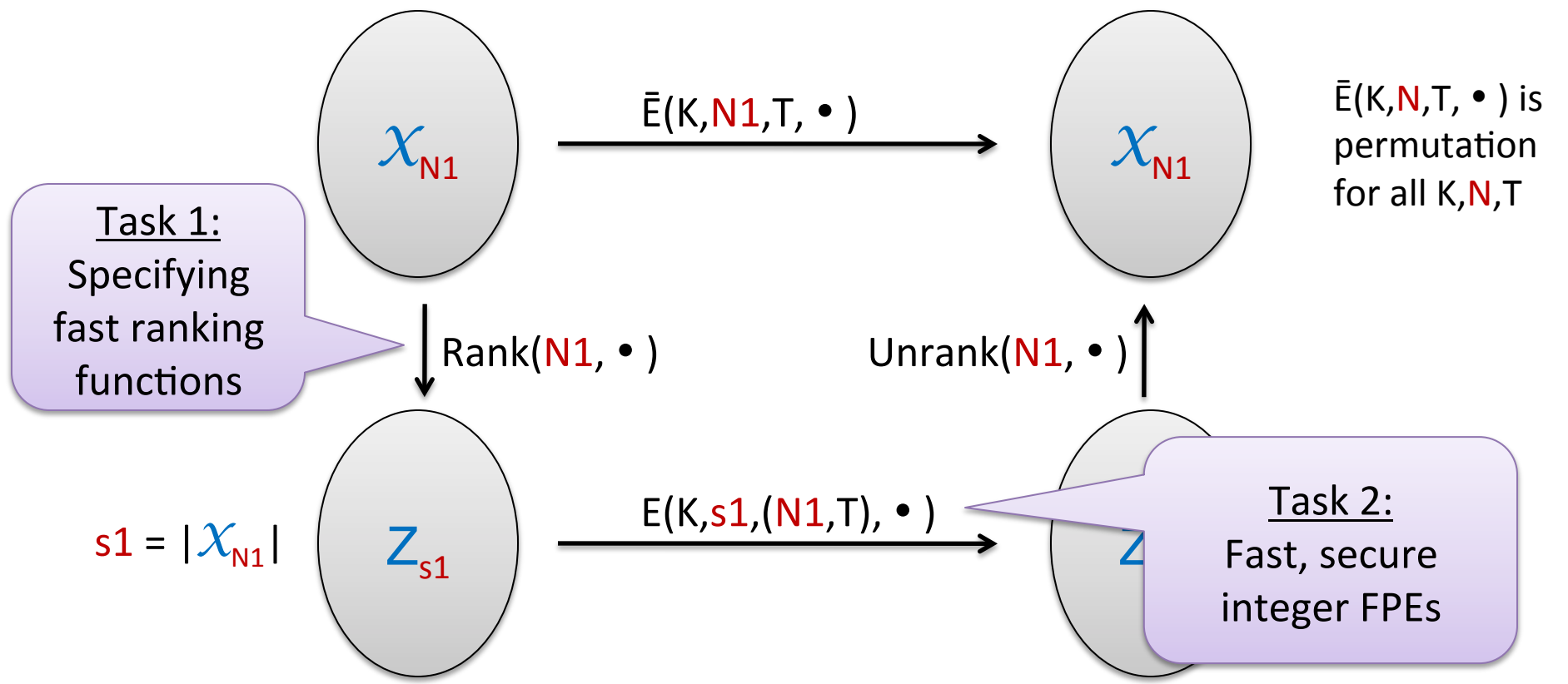
Can't exist efficient $\text{Unrank}(N, \cdot)$ if $P \neq \#P$

Modify [Jerrum '95] results to give direct FPE on proper k -colorings

General construction of FPE schemes: Rank-then-Encipher

$\mathcal{N} = \{N1, N2, \dots\}$ some arbitrary format space

Domain is $\mathcal{X} = \{x_N\}_{N \in \mathcal{N}}$



Just need integer FPE for $\mathcal{N} = \{s1, s2, \dots\}$ where $s1 = |x_{N1}|$, $s2 = |x_{N2}|$, ...

$\bar{E}$ inherits security of E . Proof is easy

Rank-then-Encipher Task 2: Integer FPE schemes

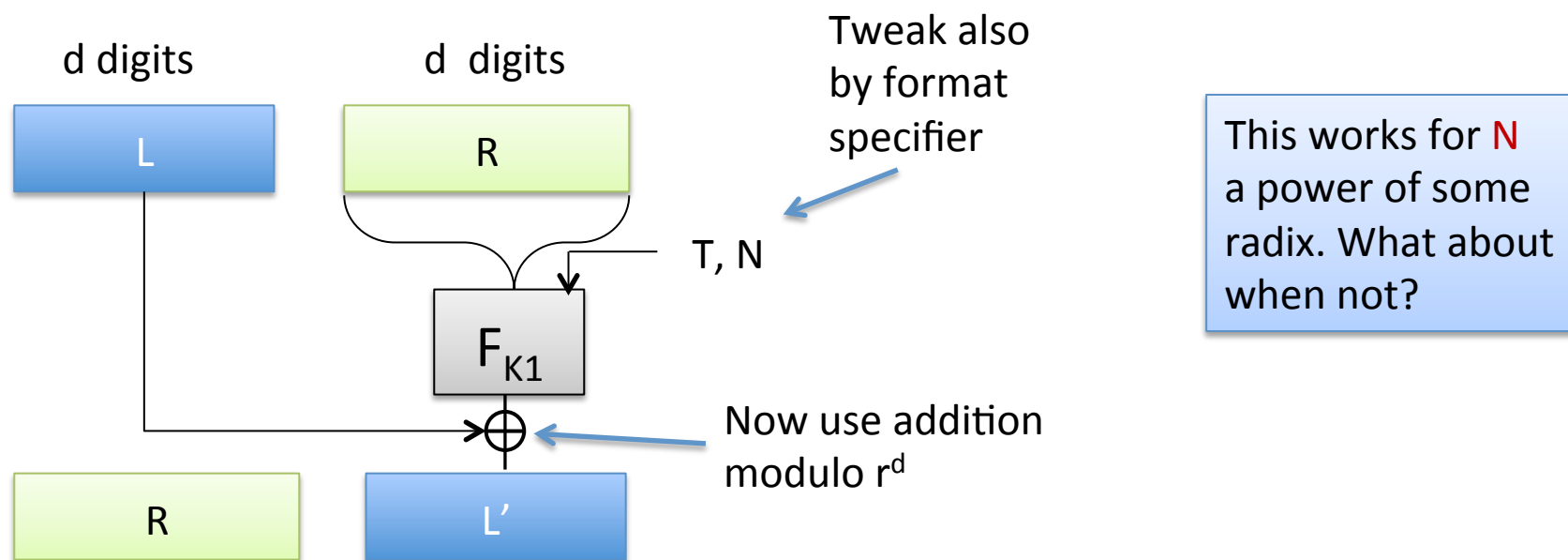
- Required domain is $\mathcal{X} = \{Z_N\}_{N \in \mathcal{N}}$ where $\mathcal{N} = \{1, 2, 3, \dots\}$
- We will focus on one slice: cipher on Z_N for some $N \in \mathcal{N}$
 - Build using conventional n-bit block cipher (AES)
 - Target conventional strong PRP security

$N < 2^n$	“Small-block encryption” problem	Example: $n = 128$ $N = 10^9 \approx 2^{30}$	Shuffling algorithms from last segment	
$N \geq 2^n$	“Wide-block encryption” problem	Example: $n = 128$ $N = 4096 * 8$	Fastest solutions are of EME2, CMC, TET, ... style (stay tuned)	Security: $q \approx O(2^{n/2})$ Efficiency: $O(\log N)$ AES calls

Rank-then-Encipher Task 2: Integer FPE schemes

Required domain is $\mathcal{X} = \{Z_N\}_{N \in \mathcal{N}}$ where $\mathcal{N} = \{1, 2, 3, \dots\}$

- Used to operating on bit strings, but N often not power-of-2
- Many shuffling constructions work for arbitrary N
 - Mix-and-cut, Swap-or-Not, Sometimes-Recurse, etc.
 - So far a bit slow for practical deployment
- In practice: FFX mode based on some radix (e.g., $r = 10$)



FFX specification heuristically suggests 10 rounds of Feistel. (Fast)

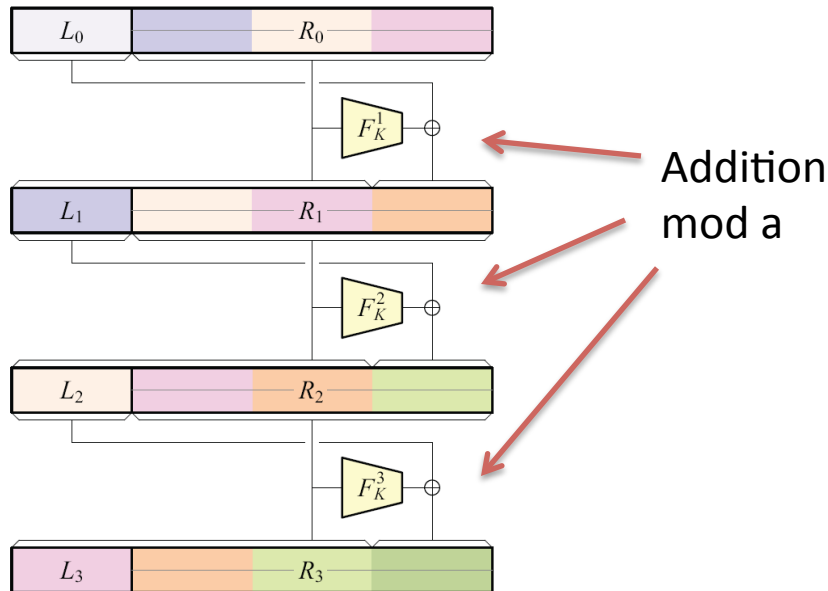
Unbalanced Feistel-based FPEs

Format space is $\mathcal{N} = \{(a,b) \mid 2 \leq a \leq b\}$ and $\chi_{(a,b)} = \mathbb{Z}_{ab}$

Split input $M \in \mathbb{Z}_{ab}$ into $L_0 \in \mathbb{Z}_a$ and $R_0 \in \mathbb{Z}_b$

Apply R rounds of an unbalanced Feistel network

FE1

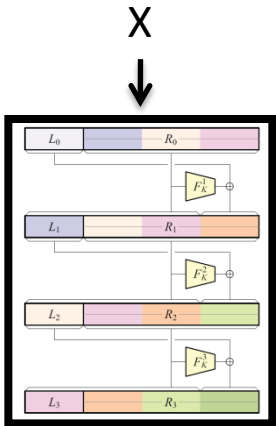


What if N is prime?

[Hoang, Rogaway 2010] show security up to $\sim N/a$ queries for enough rounds.
Probably can get by with fewer rounds than they suggest

Use cycle-walking [Black, Rogaway 2002]

to go from $\mathcal{N} = \{(a,b) \mid 2 \leq a \leq b\}$ to $\mathcal{N} = \{1,2,3,\dots\}$



To encrypt message $X \in \mathbb{Z}_N$

Choose a, b so that $ab > N$

$$Y = \text{FE1}(K, (a, b), T, X)$$

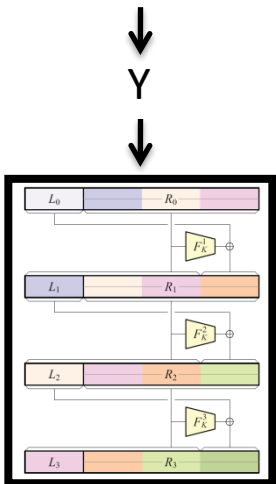
Is $Y \in \mathbb{Z}_N$? Yes, stop and output Y

Worst-case running time:

$$O(N)$$

Expected running time:

$$O(N/ab)$$



$$Y' = \text{FE1}(K, (a, b), T, Y)$$

Is $Y' \in \mathbb{Z}_N$? Yes, stop and output Y'

⋮

⋮

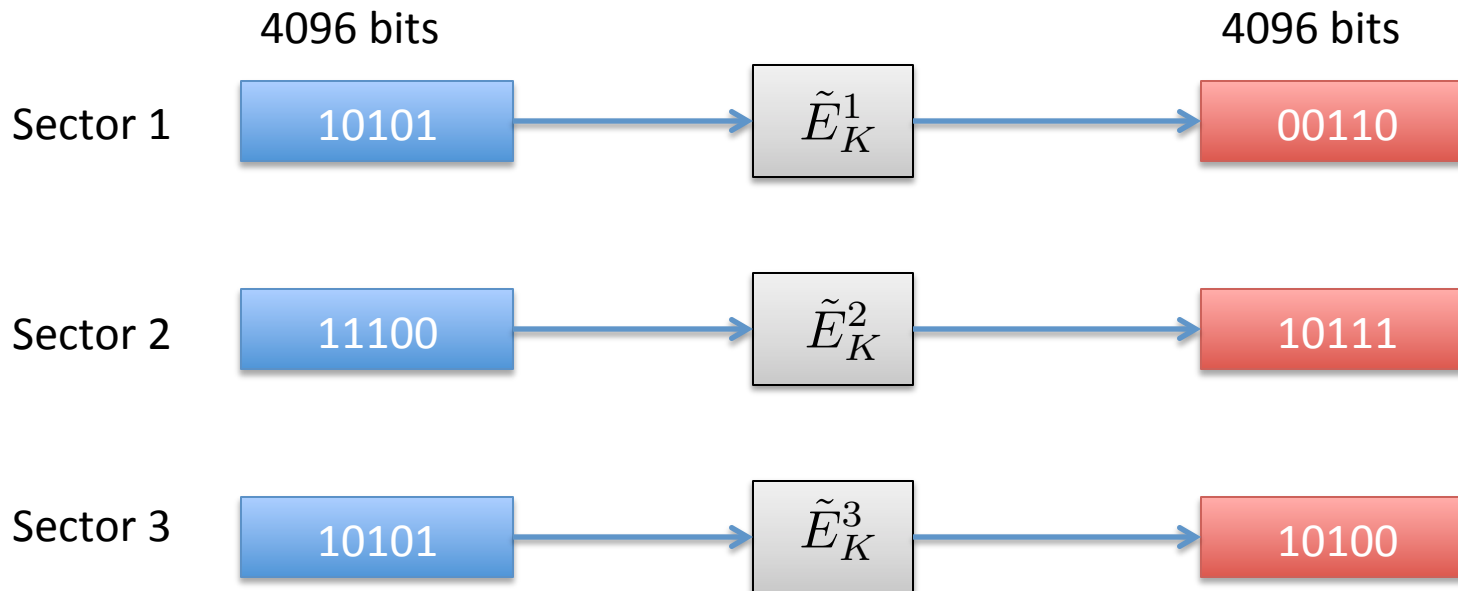
- Required domain is $\mathcal{X} = \{Z_N\}_{N \in \mathcal{N}}$ where $\mathcal{N} = \{1, 2, 3, \dots\}$
- We will focus on one slice: cipher on Z_N for some $N \in \mathcal{N}$
 - Build using conventional n-bit block cipher (AES)
 - Target conventional strong PRP security

$N < 2^n$	“Small-block encryption” problem	Example: $n = 128$ $N = 10^9 \approx 2^{30}$	Shuffling algorithms from last segment	
$N \geq 2^n$	“Wide-block encryption” problem	Example: $n = 128$ $N = 4096$	Fastest solutions are of EME2, CMC, TET, ... style (stay tuned)	Security: $q \approx O(2^{n/2})$ Efficiency: $O(\log N)$ AES calls

Length-preserving encryption for large domains

Required domain is $\mathcal{X} = \{\{0,1\}^N\}_{N \in \mathcal{N}}$ where $\mathcal{N} = \{128, 129, \dots\}$
(128 because of AES)

How to build TBC on 4096 bits
given BC with $n = 128$?

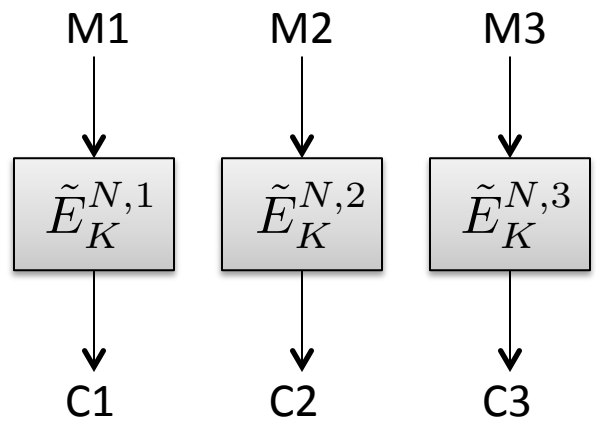


Studied extensively due to need for disk-sector encryption

Length-preserving encryption for large domains

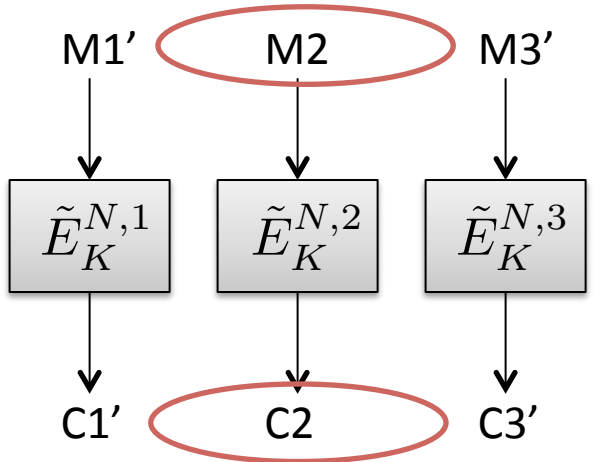
Required domain is $\mathcal{X} = \{ \{0,1\}^N \}_{N \in \mathcal{N}}$ where $\mathcal{N} = \{128,129,\dots\}$
(128 because of AES)

Split 4096 bit sector across multiple invocations of XEX



What's wrong with OCB core?
(say N is tweak = sector #)

Each ciphertext block not affected
by all plaintext blocks



What does this leak?

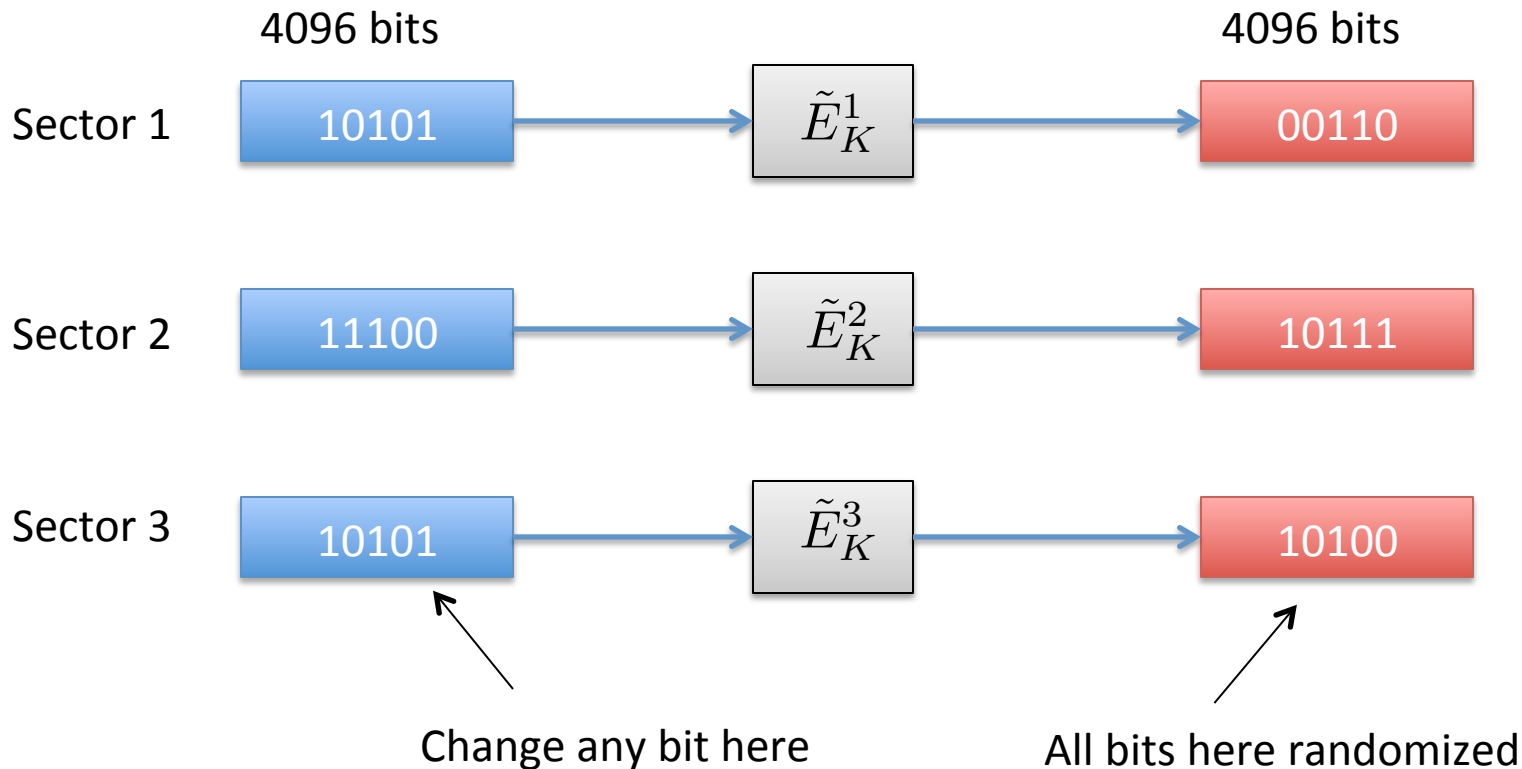
Multiple ciphertexts leak n-bit diffs

Practitioners seem to think this is ok:
XTS mode standardizes variant of this
XTS used in many products

Length-preserving encryption for large domains

Required domain is $\mathcal{X} = \{\{0,1\}^N\}_{N \in \mathcal{N}}$ where $\mathcal{N} = \{128, 129, \dots\}$
(128 because of AES)

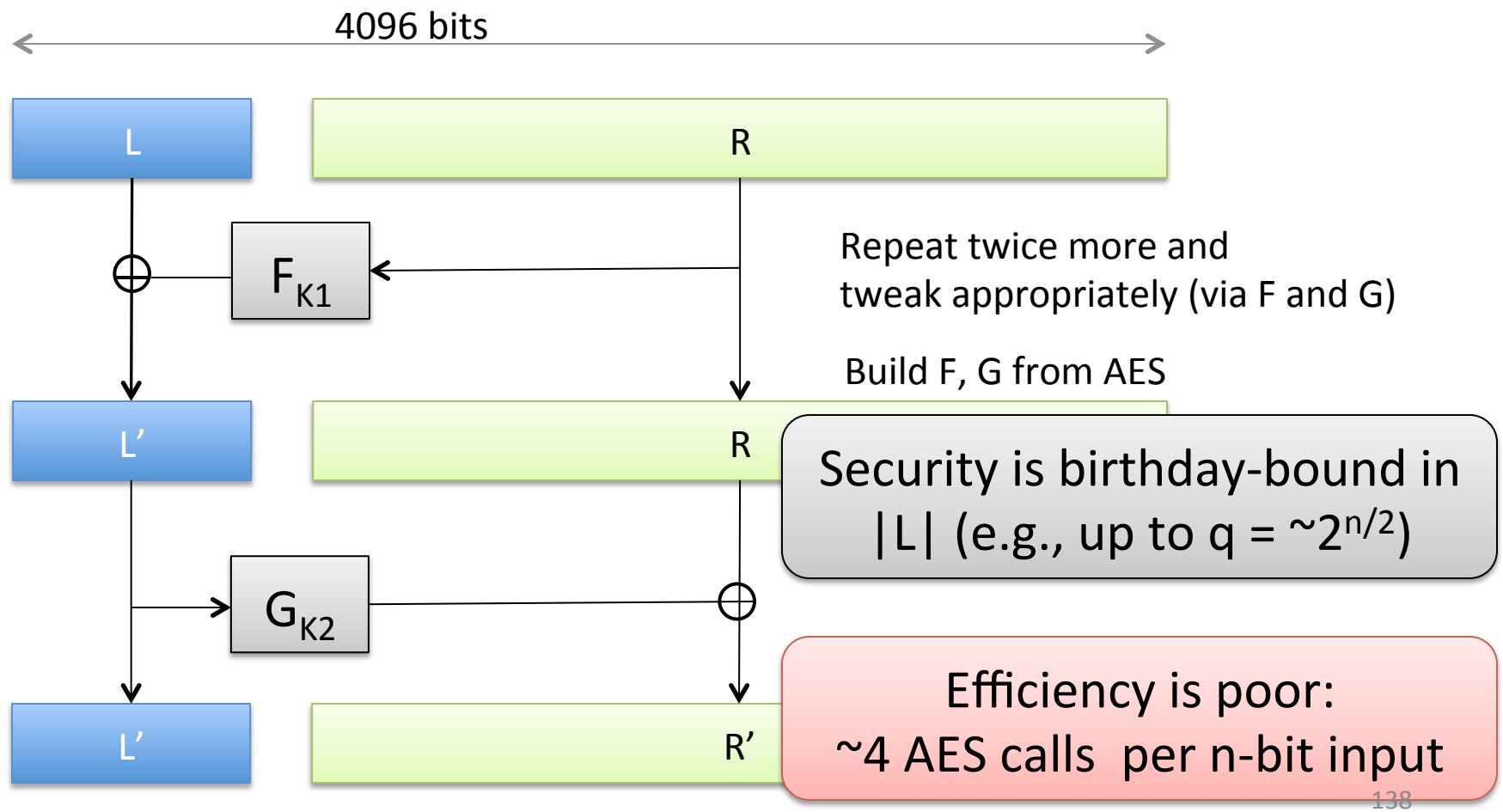
We can do better by building wide-block length-preserving ciphers



Length-preserving encryption for large domains

Required domain is $\mathcal{X} = \{ \{0,1\}^N \}_{N \in \mathcal{N}}$ where $\mathcal{N} = \{128, 129, \dots\}$
(128 because of AES)

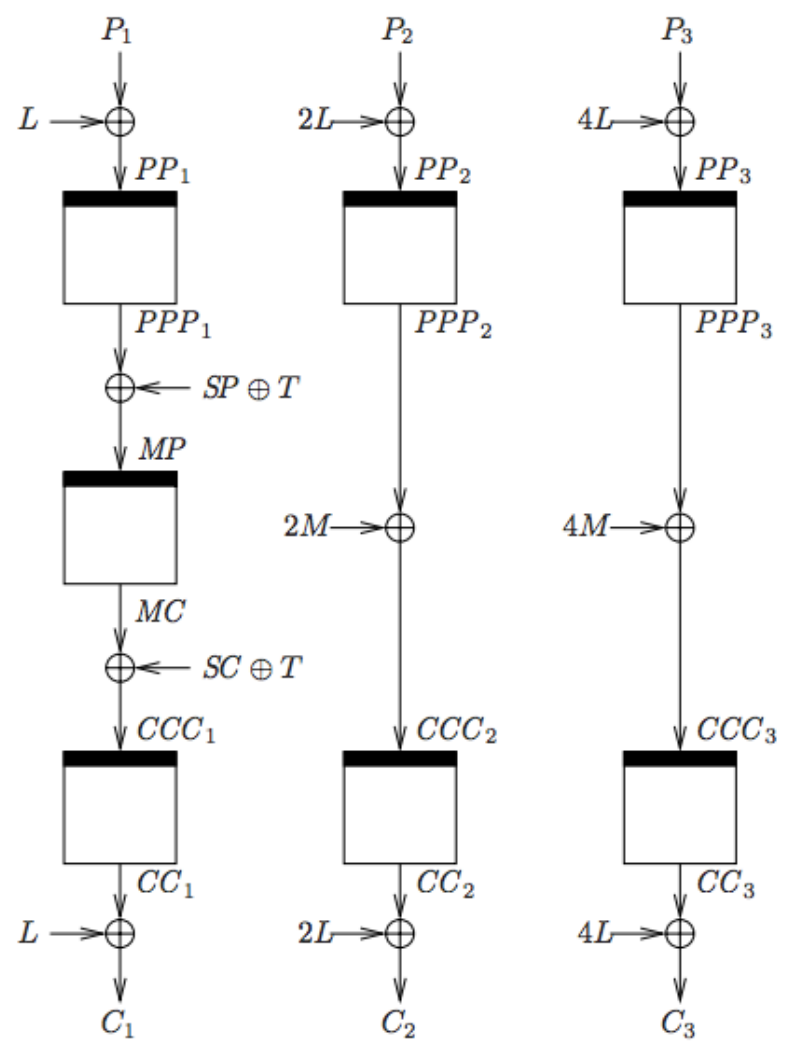
Simple solution is 4 rounds of a type of unbalanced Feistel



Length-preserving encryption for large domains

Required domain is $\mathcal{X} = \{ \{0,1\}^N \}_{N \in \mathcal{N}}$ where $\mathcal{N} = \{128,129,\dots\}$
(128 because of AES)

Faster modes: EME (~2 block ciphers per n-bit input)



$$\begin{aligned} SP &= PPP_2 + PPP_3 \\ SC &= CCC_2 + CCC_3 \\ M &= MP + MC \end{aligned}$$

Provably secure up to $\sim 2^{n/2}$ blocks encrypted in total

Proof is super complex
Recast using TBC and get simpler proof?

Diagram from [Halevi , Rogaway 2003]

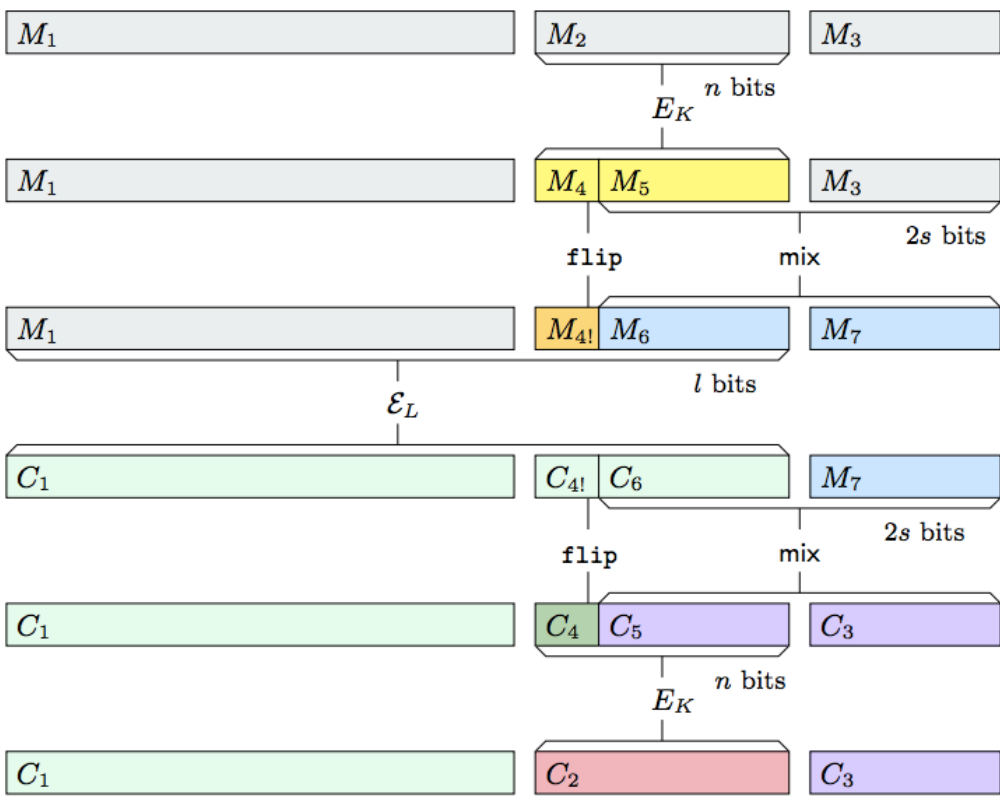
Length-preserving encryption for large domains

Required domain is $\mathcal{X} = \{ \{0,1\}^N \}_{N \in \mathcal{N}}$ where $\mathcal{N} = \{128,129,\dots\}$
(128 because of AES)

Faster modes: EME (~2 block ciphers per n-bit input)

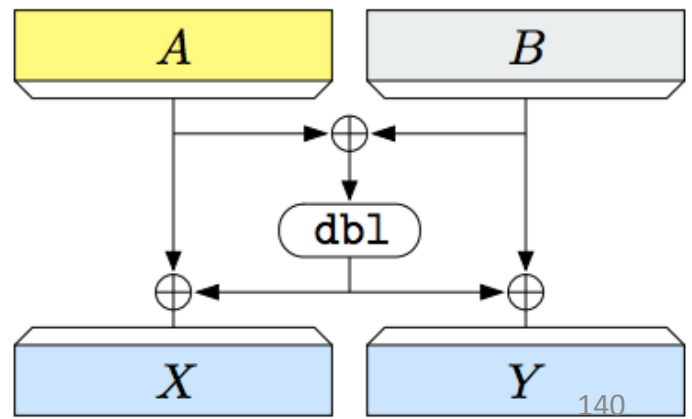
Extending to arbitrary bit-lengths? EME* does.

A generic approach with XLS (eXtension by Latin Squares)



$|M_1| = jn \text{ for } j \geq 0$
 $|M_2| = n$
 $|M_3| < n$

mix is a multipermutation on 2s bits:



Orthogonal Latin Squares

Latin square: each column and each row include all symbols

Orthogonal Latin square: two Latin squares for which $(A[i,j], B[i,j])$ never repeats for distinct i,j

A			
00	01	10	11
01	00	11	10
10	11	00	01
11	10	01	00

B			
00	10	11	01
01	11	10	00
10	00	01	11
11	01	00	10

$$\text{mix}((i,j)) = (A[i,j], B[i,j])$$

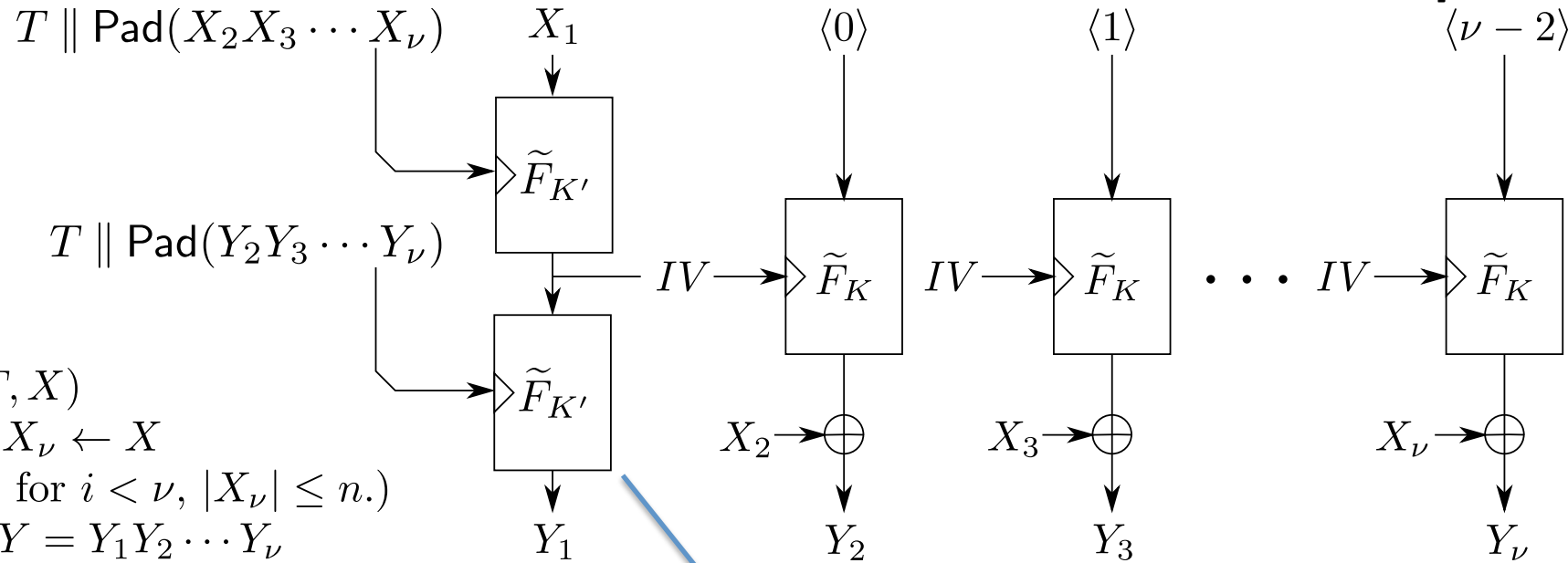
Sufficiently good approximation of OLS is:

$$\text{mix}(i,j) = (i + \text{rol}(i+j), j + \text{rol}(i+j))$$

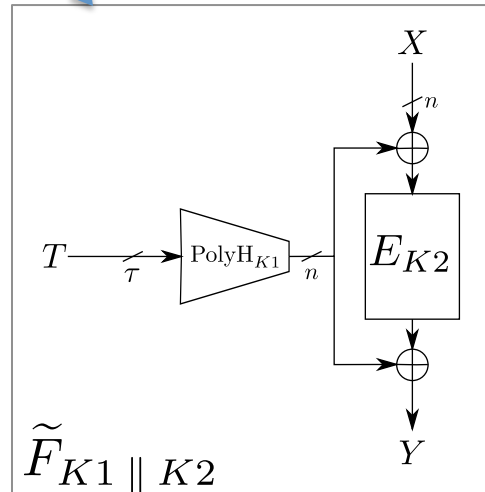
Where $+$ is xor and rol is circular-rotate left by one bit

Protected IV mode (TCT1)

[Shrimpton,
Terashima 2013]



~1 BC call +
 ~2 finite field multiplications
 per n-bit input



Outline

- Motivation for FPE
- Formalization of FPE
- Rank-Encipher-Unrank
 - Ranking for regular expressions
 - Integer FPE
- Integer FPE schemes for small domains
- Disk-sector / FPE schemes for large domains

Lecture plan

1. Tweakable PRPs and shuffling
2. Nonce-based symmetric encryption
3. Format-preserving encryption
4. Further symmetric primitives

Lecture 4:

Further symmetric primitives

Thomas Ristenpart
University of Wisconsin

Our game-plan today

We will build two widely needed primitives:

- **Nonce-based symmetric encryption**
 - Contemporary viewpoint on SE
 - Two flavors (speed versus security trade-off)
- **Format-preserving encryption**
 - Used widely in industry for fixed-field encryption (credit card numbers, etc.)
 - Length-preserving encryption as special case
- **Time allowing:** advanced SE primitives such as message-locked encryption, honey encryption, password-based encryption,

Further SE Primitives

- Dealing with passwords
 - PKCS#5 constructions
- Multi-instance security
- Honey encryption
- Format-transforming encryption

These address deficiencies in conventional encryption schemes in various contexts

Examples of deficiencies:



Deep packet inspection systems can **block** protocols

↳ Encryption tools are easy to detect

Format-transforming encryption to trick DPI



Dropbox **has access** to your data

↳ Encryption doesn't allow deduplication to save space

Message-locked encryption to support dedup



Lastpass uses password-based encryption that can be **cracked**

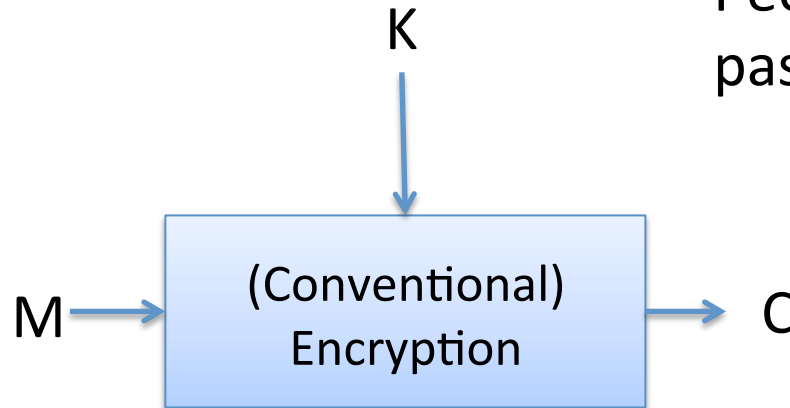
↳ Encryption reveals when wrong key is used to decrypt

PBKDFs and honey-encryption

Password-based encryption

secret password K

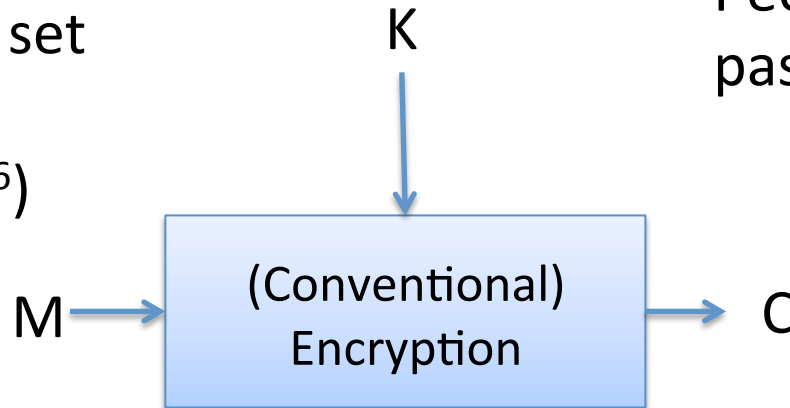
People don't pick good passwords



Password-based encryption

secret password K
drawn from set
of size q
(e.g., $q = 10^6$)

People don't pick good
passwords



Brute force attack given C :

$M_1 \leftarrow \text{Decrypt}(K_1, C)$

$M_2 \leftarrow \text{Decrypt}(K_2, C)$

$M_3 \leftarrow \text{Decrypt}(K_3, C)$

...

$M_q \leftarrow \text{Decrypt}(K_q, C)$



abufdsjklfeqfdsj

hgjkzalcfewjiofw

beertimeat5man

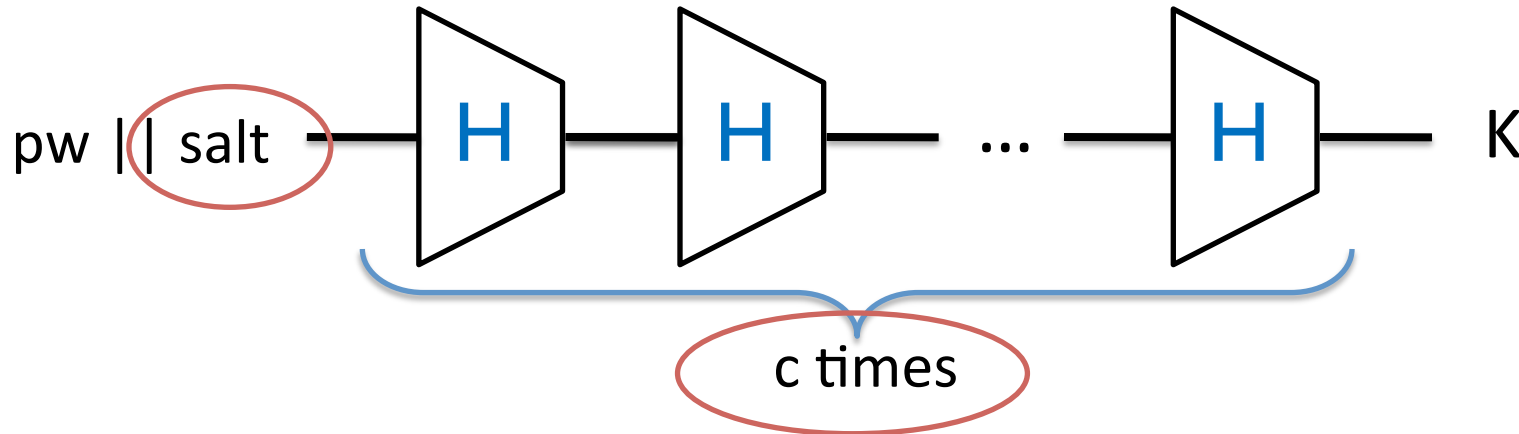
...

tyeiragjzjfjdajsal

How to deal with offline brute-force attacks?

- Slow them down by making encryption/
decryption slower
 - Standard practice of using hash chains / salts
 - PKCS#5 standardizes this
 - Used widely in practice
- Make it hard to pick out correct plaintext
 - Folklore idea
 - Very recently: honey encryption

PKCS #5 : Password based cryptography



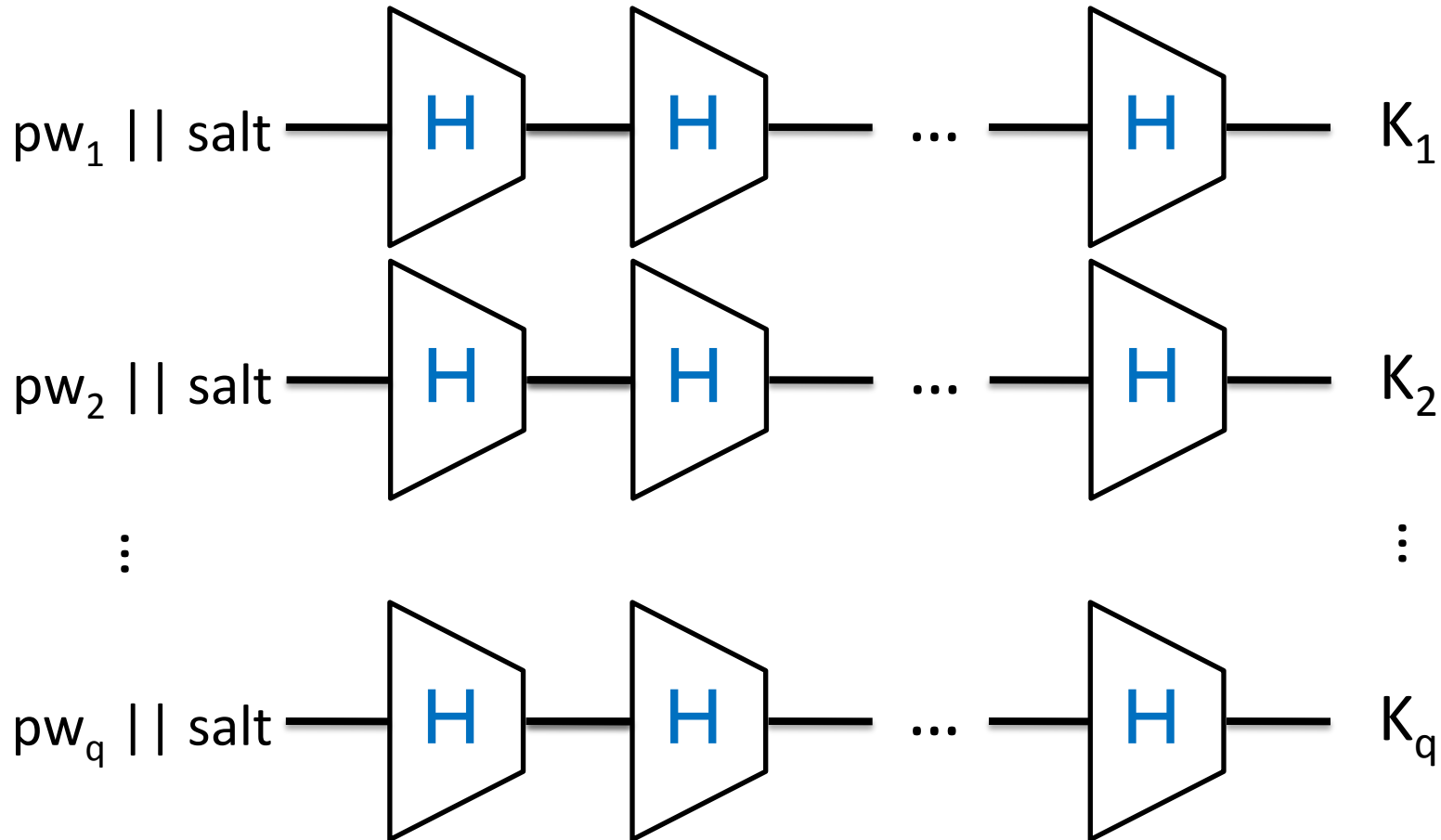
$H : \{0,1\}^* \rightarrow \{0,1\}^n$ is cryptographic hash function (e.g., SHA-256)

K truncated if needed

```
PB-Encrypt(pw, M)
salt  $\leftarrow \{0,1\}^s$ 
K  $\leftarrow H^c(\text{pw} || \text{salt})$ 
C  $\leftarrow \text{Encrypt}(K, M)$ 
Return salt || C
```

Used widely: Winzip, OpenOffice, Mac OS X FileVault, TrueCrypt, WiFi WPA (PBKDF), ...

Intuition for no short-cut attacks

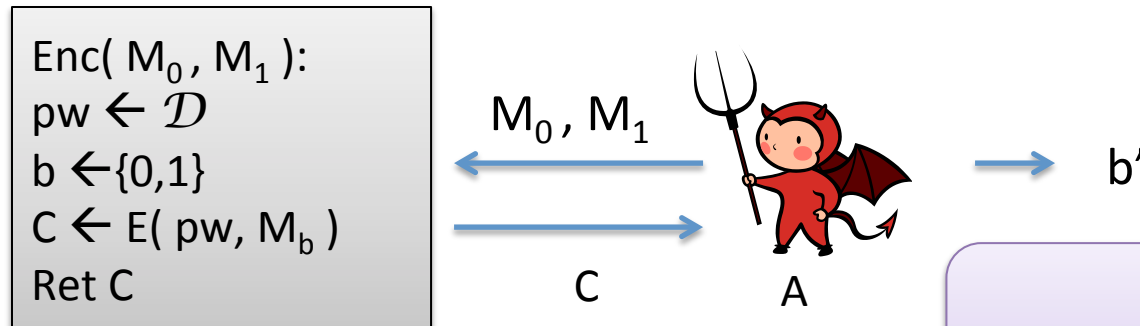


If H is “ideal” (a random oracle) then probability of any two chains overlapping is small. Surprisingly complex proof, though.

Proving benefit of iterations

Let $\mathcal{D}$ be set of possible passwords (dictionary)
IND-CPA security is traditional goal here
(shown as single-query version for simplicity):

```
PB-Encrypt(pw, M)
salt  $\leftarrow \{0,1\}^s$ 
 $K \leftarrow H^c(\text{pw} || \text{salt})$ 
 $C \leftarrow \text{Encrypt}(K, M)$ 
Return salt || C
```



$$\text{Adv}(E, A) = \Pr[A \Rightarrow b] - 1/2$$

Salts play no role in
IND-CPA security!
What are they for?

Theorem: PB-Encrypt is such that for all IND-CPA adversaries A

$$\text{Adv}(\text{PB-Encrypt}, A) < q / cN$$

where $N = |\mathcal{D}|$, ignoring negligible terms and small constants, and modeling H as a random oracle

An easy brute-force attack works in time close to cN

The role of salts

- Make precomputation attacks harder/not work
 - Rainbow tables are nice time-memory trade-off
 - Long salts prevent this, since need to know salt before attacking system
- Make breaking encryptions under independent passwords harder
 - Attacking encryptions of m independent passwords require m times the work

Multi-instance attack setting

Adversary gets encryptions $C_1, \dots, C_m$
for unknown messages $M_1, \dots, M_m$ under
passwords $pw_1, \dots, pw_m$ independently
chosen from $\mathcal{D}$ with $N = |\mathcal{D}|$

```
PB-Encrypt(pw, M)
salt  $\leftarrow \{0,1\}^s$ 
K  $\leftarrow H^c(pw || \text{salt})$ 
C  $\leftarrow \text{Encrypt}(K, M)$ 
Return salt || C
```

Salting slows down best **known** attacks:

With salts:

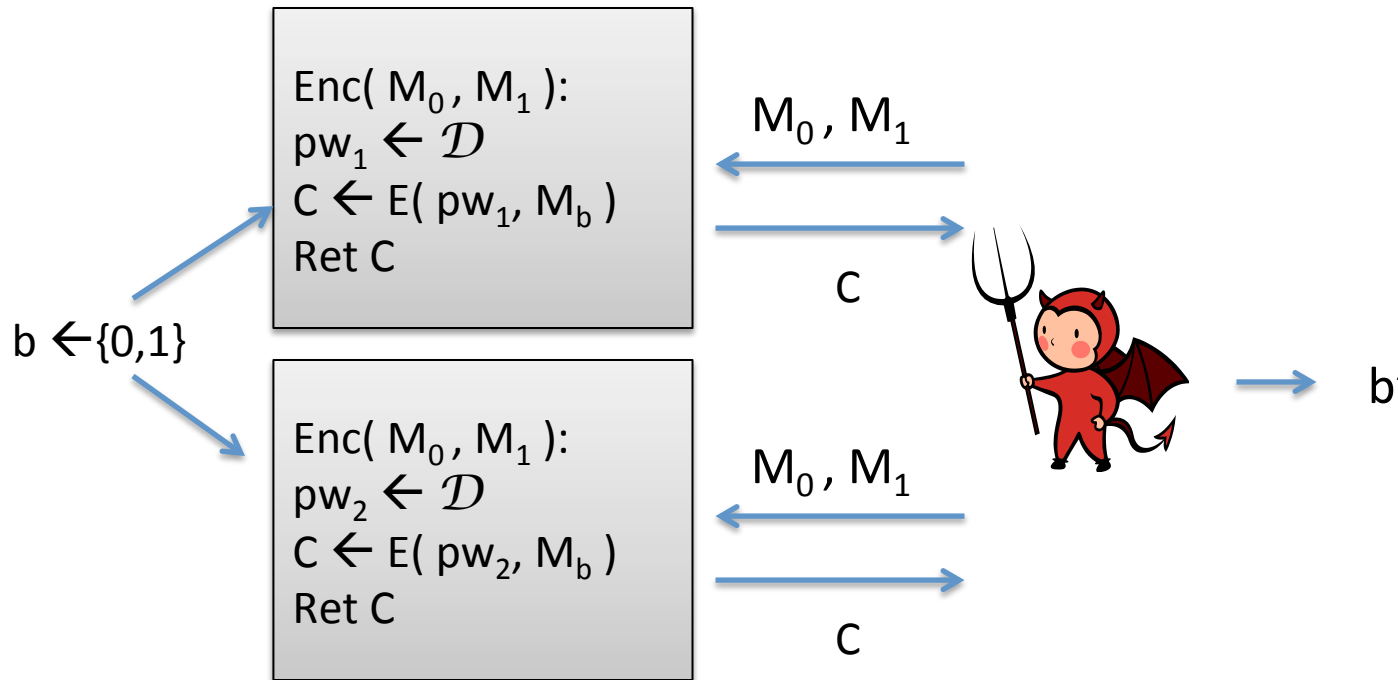
attack in time $O(mcN)$ recovers **all** m passwords

Without salts:

attack in time $O(cN + mN)$ recovers **all** m passwords

Can we **prove** that “breaking m encryptions”
requires mCN time?

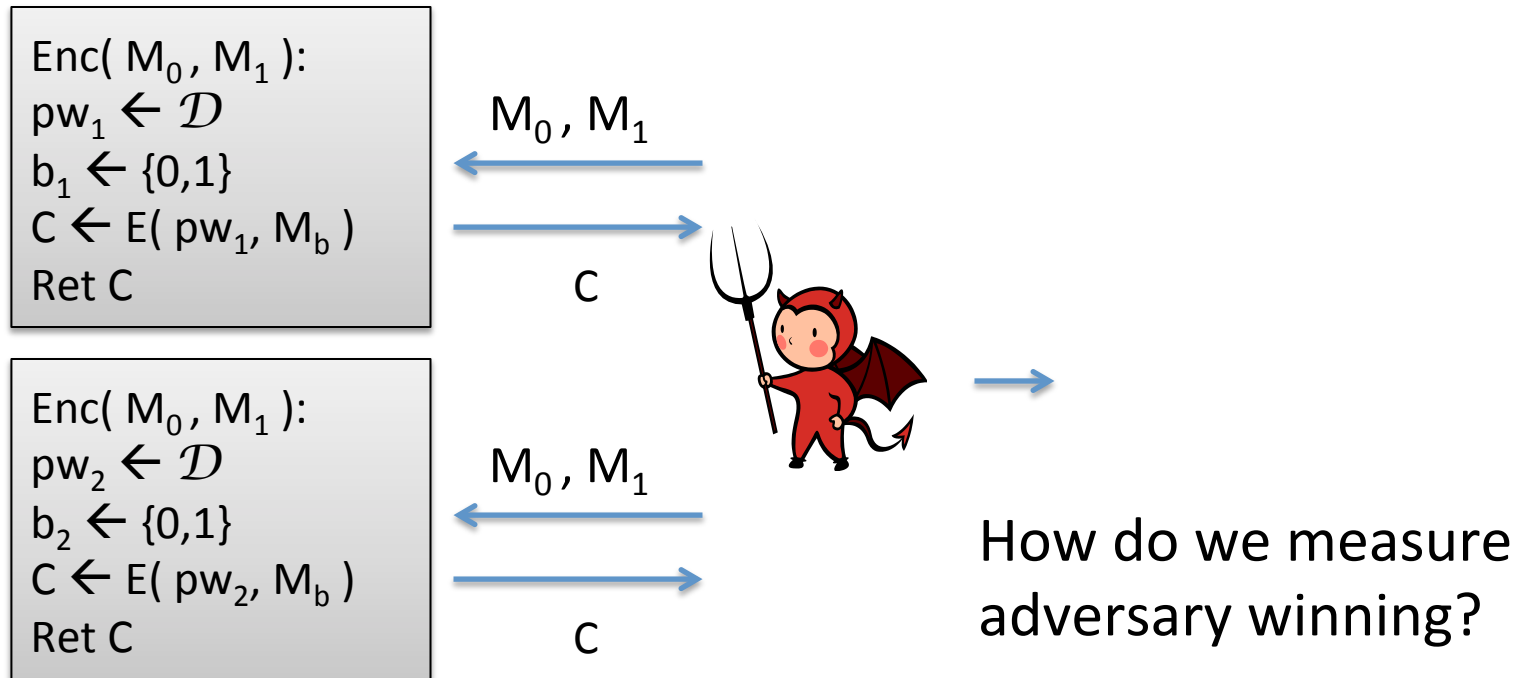
Defining two-instance encryption security: try 1



$$\text{Adv}(A) = \Pr[A \Rightarrow b] - 1/2$$

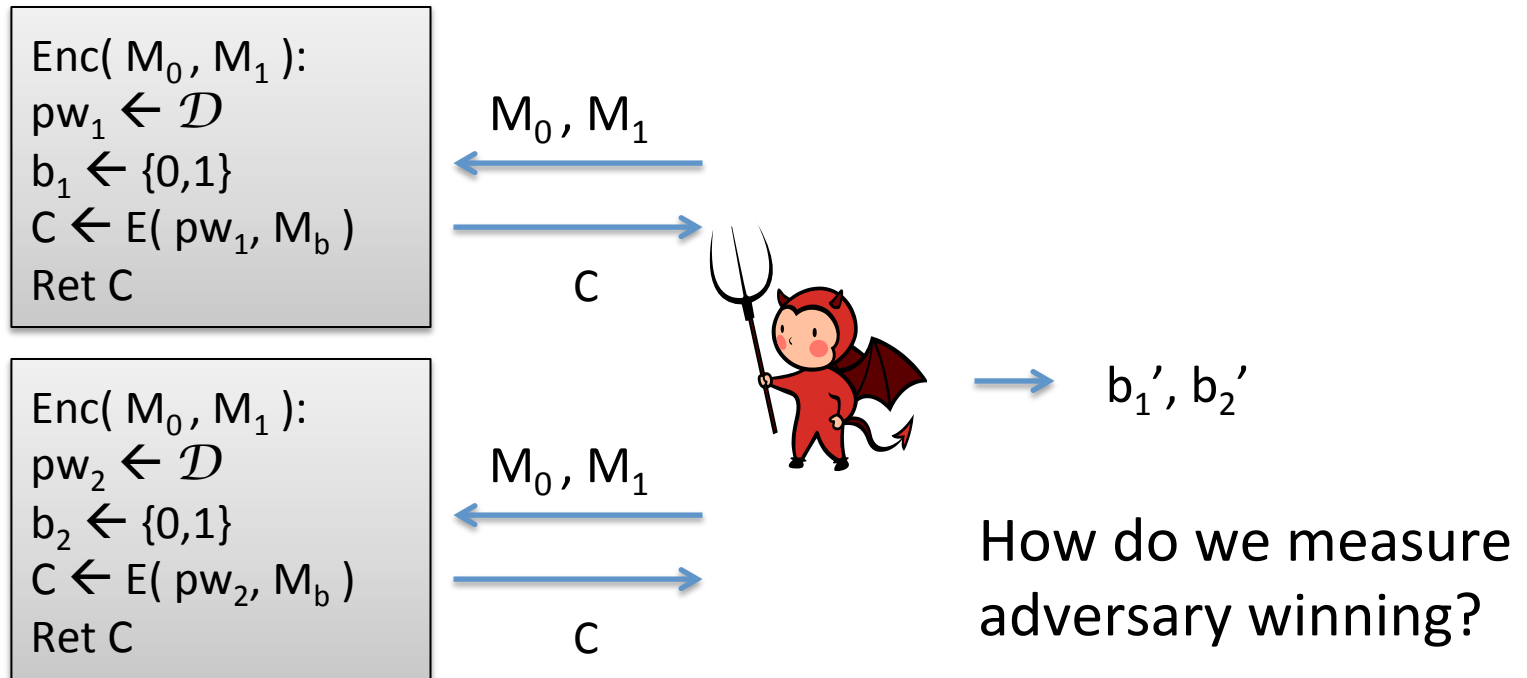
This is the multi-user setting of [Bellare, Boldyreva, Micali 2000]
But it doesn't work for us: recovering pw_1 or pw_2 reveals b

Defining two-instance encryption security: try 2



Use independent challenge bits in two different instances

Defining two-instance encryption security: try 2

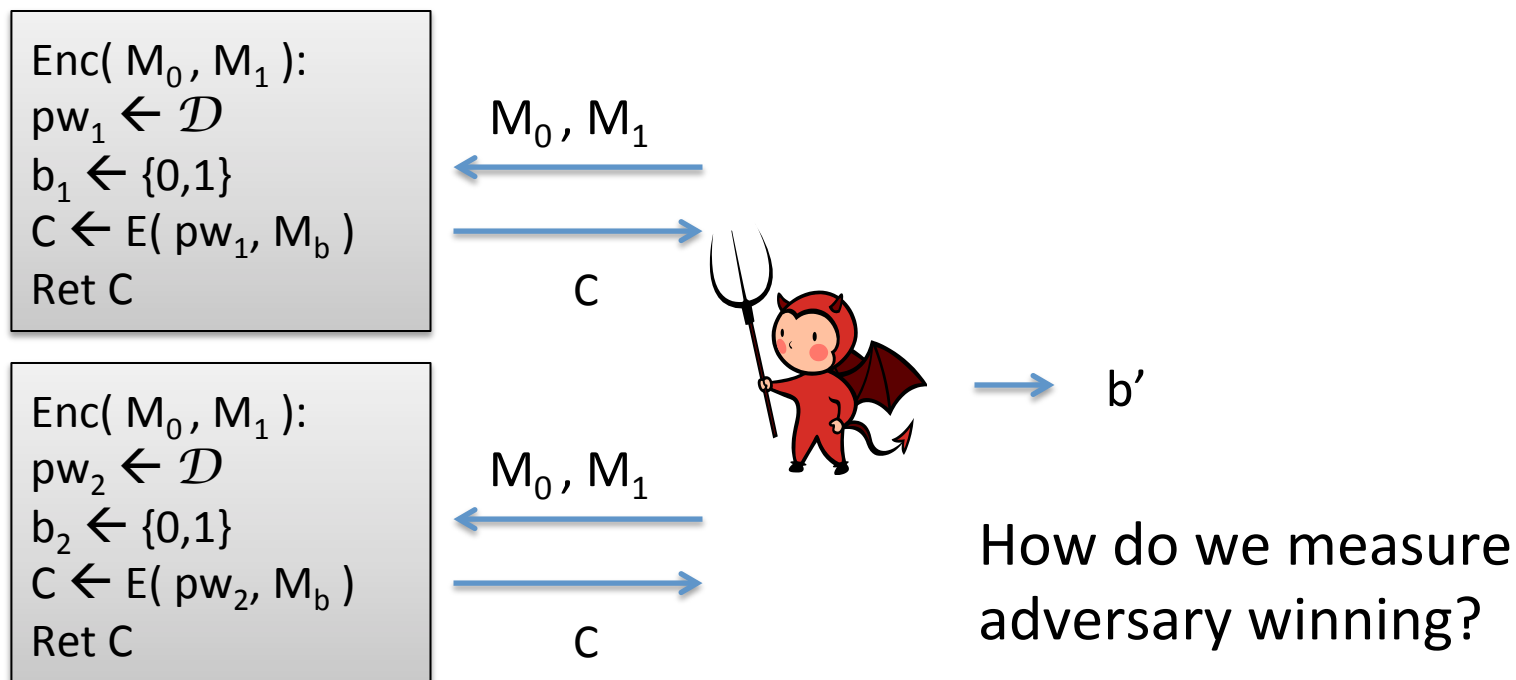


$$\text{Adv}(A) = \Pr[A \Rightarrow (b_1, b_2)] - 1/4$$

Use independent challenge bits in two different instances

AND measure: recover pw_1 and guess b_2 gives $\text{Adv}(A) = \frac{1}{2} - \frac{1}{4} = \frac{1}{4}$

Defining two-instance encryption security: try 3



$$\text{Adv}(A) = \Pr[A \Rightarrow (b_1 \oplus b_2)] - 1/2$$

Use independent challenge bits in two different instances

AND measure: recover pw_1 and guess b_2 gives $\text{Adv}(A) = \frac{1}{2} - \frac{1}{4} = \frac{1}{4}$

XOR measure: recover pw_1 and guess b_2 gives $\text{Adv}(A) = 0$

This is simplified view of multi-instance security notions

Adaptive corruptions (attacker can adaptively corrupt games, learning, e.g., password)

Extends to primitives beyond encryption

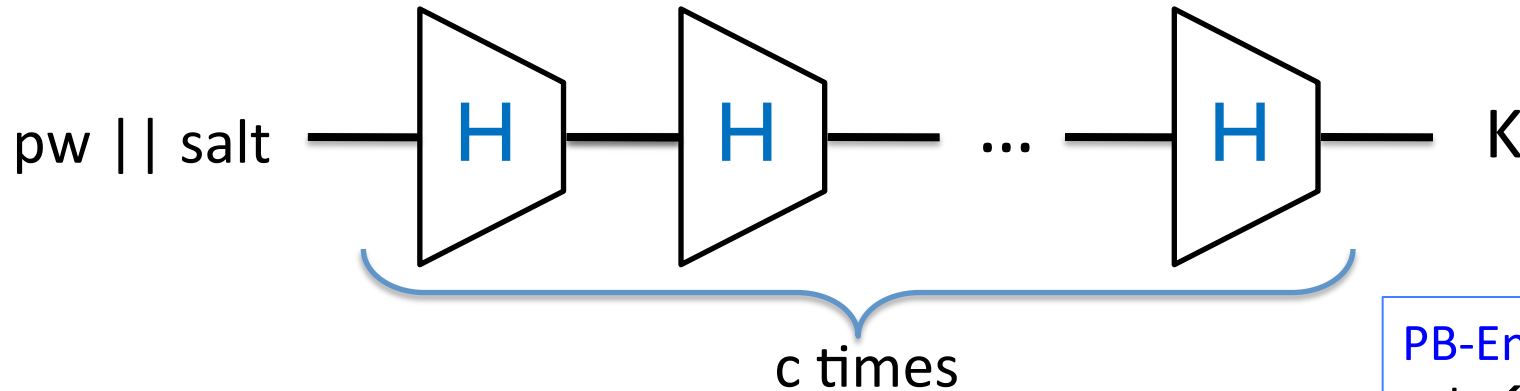
Connections with Yao-style hardness amplification literature

Multi-instance setting surfaces technical challenges

Hybrid arguments no longer easy and incur factor 2^m loss in advantage

Details in [Bellare, R., Tessaro 2012]

Back to passwords and PKCS #5...



Introduce indistinguishability-style
simulation-based PBKDF notion
to overcome difficulties of multi-instance setting

PB-Encrypt(pw, M)
salt $\leftarrow \{0,1\}^s$
 $K \leftarrow H^c(\text{pw} || \text{salt})$
 $C \leftarrow \text{Encrypt}(K, M)$
Return salt || C

Theorem: **PB-Encrypt** is such that for all m-IND-CPA adversaries A there exists an adversary B such that

$$\text{Adv}(\text{PB-Encrypt}, A) < \frac{q}{mcN} + m \text{Adv}(\text{Encrypt}, B) + \frac{q^2}{2^n} + \frac{q^2}{2^s}$$

where $N = |\mathcal{D}|$, ignoring small constants, and modeling H as a random oracle

How to deal with offline brute-force attacks?

- Slow them down by making encryption/
decryption slower
 - Standard practice of using hash chains / salts
 - PKCS#5 standardizes this
 - Used widely in practice
- Make it hard to pick out correct plaintext
 - Folklore idea (one paper in 99 looked at this)
 - Very recently: honey encryption

Password-based encryption

- PKCS#5 standard:
 - Slow down decryption by lots of hashing and salts
 - Provably works ...
 - ... but only slows down previous attack to $O(mcN)$

In practice, N is too small

6-digit lower-case alphanumeric
passwords

$$N \approx 2^{31}$$

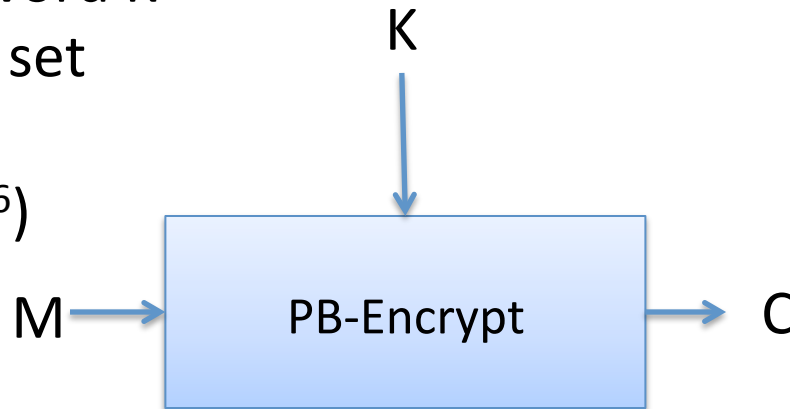
$$c = 10000$$

$$q \approx 2^{44} \text{ to succeed}$$

LastPass ****
The Last Password You'll Ever Need.

Password-based encryption

secret password K
drawn from set
of size q
(e.g., $q = 10^6$)



What if we could build encryption so that:

Brute force attack given C :

$M_1 \leftarrow \text{Decrypt}(K_1, C)$

$M_2 \leftarrow \text{Decrypt}(K_2, C)$

$M_3 \leftarrow \text{Decrypt}(K_3, C)$

...

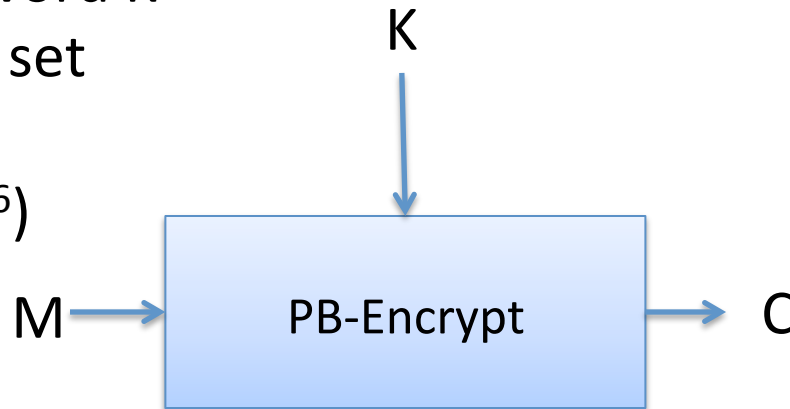
$M_q \leftarrow \text{Decrypt}(K_q, C)$



abufdsjklfeqfdsj
hgjkzalcfewjiofw
beertimeat5man
...
tyeiragjzjfjdajsal

Password-based encryption

secret password K
drawn from set
of size q
(e.g., $q = 10^6$)



What if we could build encryption so that:

Brute force attack given C :

$M_1 \leftarrow \text{Decrypt}(K_1, C)$

$M_2 \leftarrow \text{Decrypt}(K_2, C)$

$M_3 \leftarrow \text{Decrypt}(K_3, C)$

...

$M_q \leftarrow \text{Decrypt}(K_q, C)$



yehudarocks

yabbadabbado

beertimeat5man

...

isthislectureoveryet

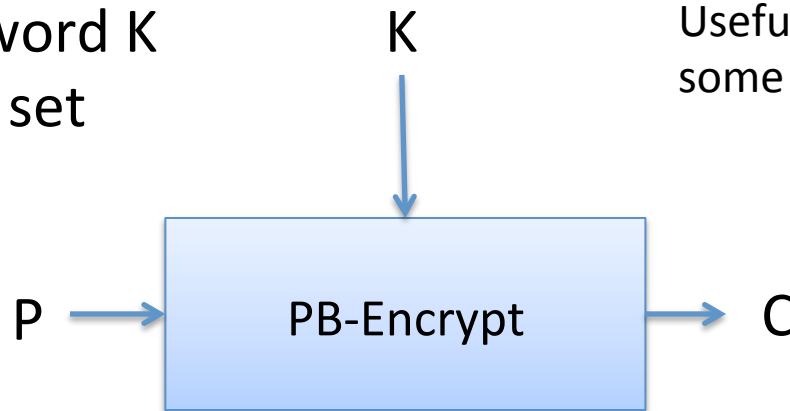
Honey encryption

- Same API as password-based encryption schemes
 - Secure in conventional sense
- But use special encodings to ensure that decrypting ciphertext with **wrong** key yields fresh sample from some target distribution
- Attacker can't figure out which is right message

Honey encryption for prime numbers

secret password K
drawn from set
of size q

Useful to store secret keys for
some crypto systems (RSA) [HK99]



1024-bit prime number

Run primality tests to see which is prime. FP
probability about $1 / 1024$ for each candidate

Brute force attack given C :

$M_1 \leftarrow \text{Decrypt}(K_1, C)$

$M_2 \leftarrow \text{Decrypt}(K_2, C)$

$M_3 \leftarrow \text{Decrypt}(K_3, C)$

...

$M_q \leftarrow \text{Decrypt}(K_q, C)$

100

321849

9883

...

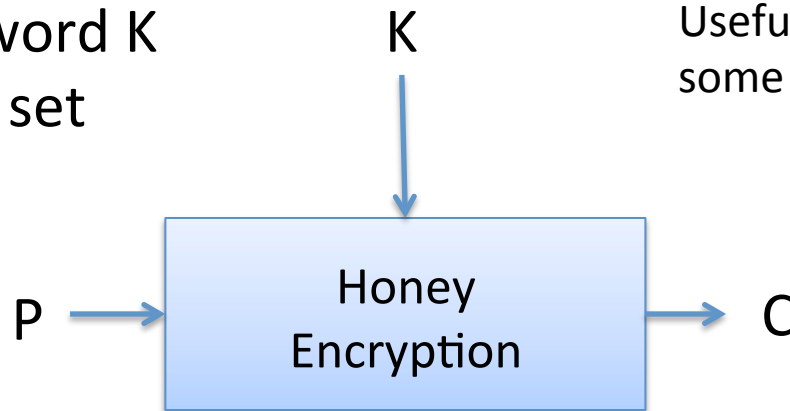
16



Honey encryption for prime numbers

secret password K
drawn from set
of size q

Useful to store secret keys for
some crypto systems (RSA) [HK99]



1024-bit prime number

All outputs of decryption are uniformly distributed prime numbers!

Brute force attack given C :

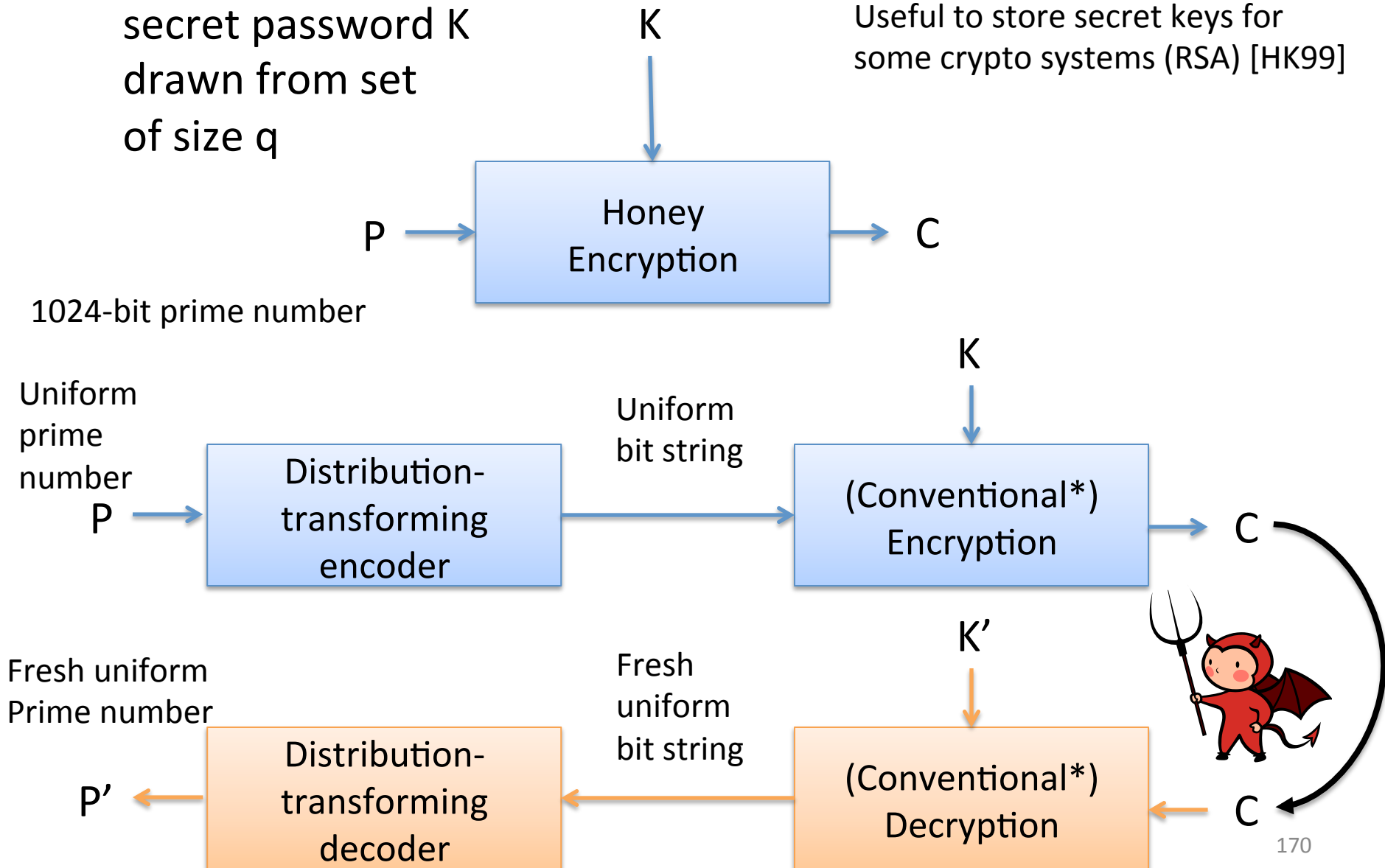
$M_1 \leftarrow \text{Decrypt}(K_1, C)$	102953
$M_2 \leftarrow \text{Decrypt}(K_2, C)$	56431
$M_3 \leftarrow \text{Decrypt}(K_3, C)$	9883
...	...
$M_q \leftarrow \text{Decrypt}(K_q, C)$	26171



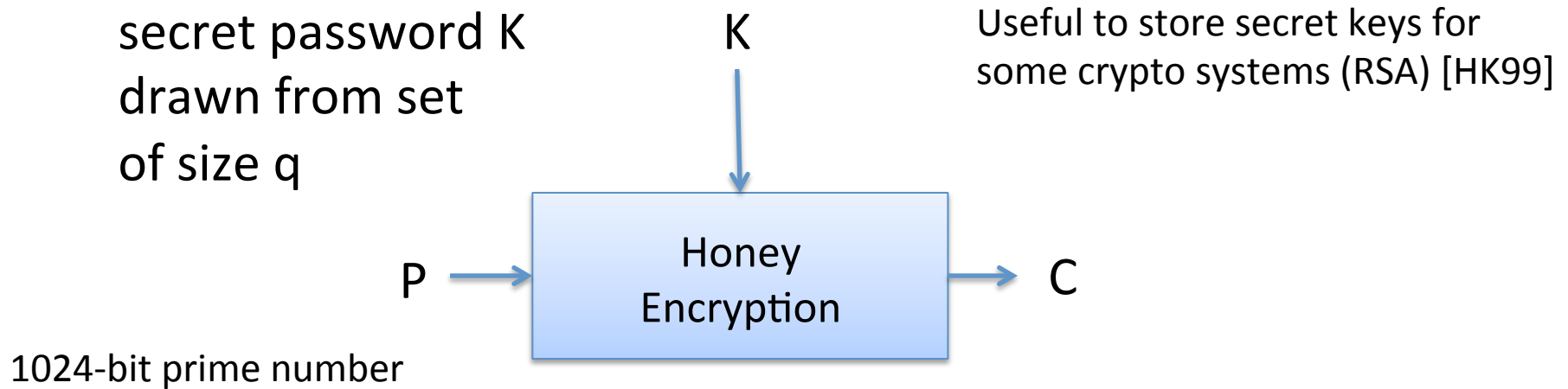
Honey encryption for prime numbers

secret password K
drawn from set
of size q

Useful to store secret keys for
some crypto systems (RSA) [HK99]



Honey encryption for prime numbers



Thm (roughly). No attacker **A** can recover message with probability better than $1 / q$

Proof requires interesting non-standard balls-and-bins analyses

Security bound essentially optimal!

Honey encryption

- Only have DTEs for some messages types
 - Uniform prime numbers
 - Credit-card numbers
- Want to build ones for messages being
 - Passwords (to help out poor Lastpass)
 - Others?

Examples of deficiencies:



Deep packet inspection systems can **block** protocols

- Encryption tools are easy to detect
- Format-transforming encryption to trick DPI



Dropbox **has access** to your data

- Encryption doesn't allow deduplication to save space
- Message-locked encryption to support dedup



Lastpass uses password-based encryption that can be **cracked**

- Encryption reveals when wrong key is used to decrypt
- PBKDFs and honey-encryption

Protocol identification via deep-packet inspection (DPI)



Check packet contents against ***regular expressions***

```
/^(\x16\x03[\x00\x01\x02]..\x02...\x03[\x00\x01\x02]|...? .*)/
```

Free translation: Does packet include “I’m TLS 1.1” ?

DPI users want to identify protocol X

X = TLS or Tor then throttle connection

X = HTTP then leave it alone

X = ??? then throttle traffic



DPI systems can classify encrypted protocols

TLS 1.1 ...

A43FB89CD213F31456

Encryption doesn't attempt to hide its presence

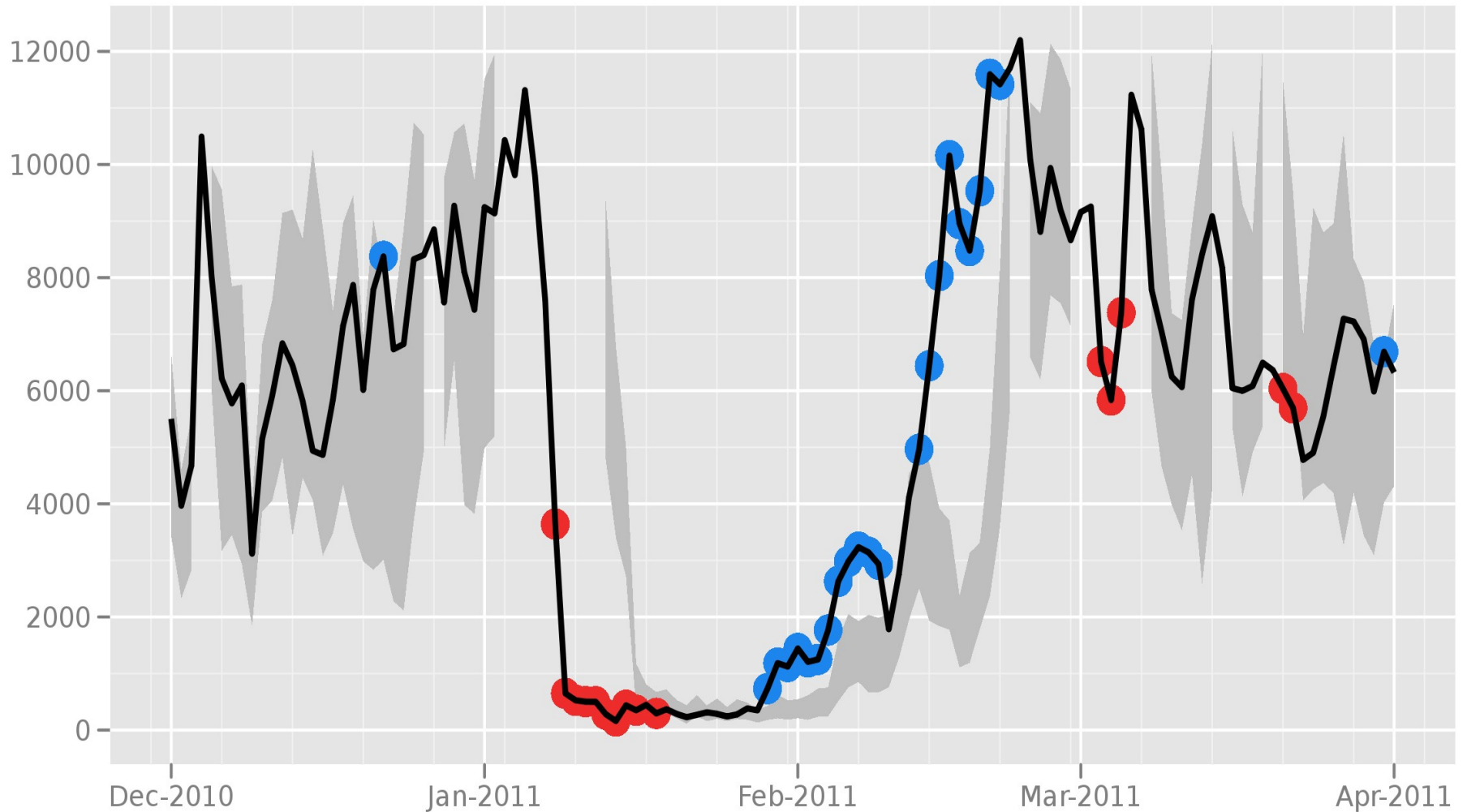


Iran reportedly blocking encrypted Internet traffic

The Iranian government is reportedly blocking access to websites that use the ...

by Jon Brodtkin - Feb 10 2012, 9:44pm IST

Directly connecting users from the Islamic Republic of Iran



The Tor Project - <https://metrics.torproject.org/>

Scenario:

DPI system only allows HTTP traffic unfettered



Stegonagraphy (e.g., Stegotorus): embed bits into HTTP messages

- Too slow for practical use (56k modem anyone?)

Obsfproxy (built into Tor): encrypt all bits sent over network (no plaintext bits)

- Really fast
- But DPI will flag traffic as ???

Want way to force DPI to classify traffic incorrectly as HTTP
So-called “misclassification attacks” against DPI

Surveying modern DPI systems

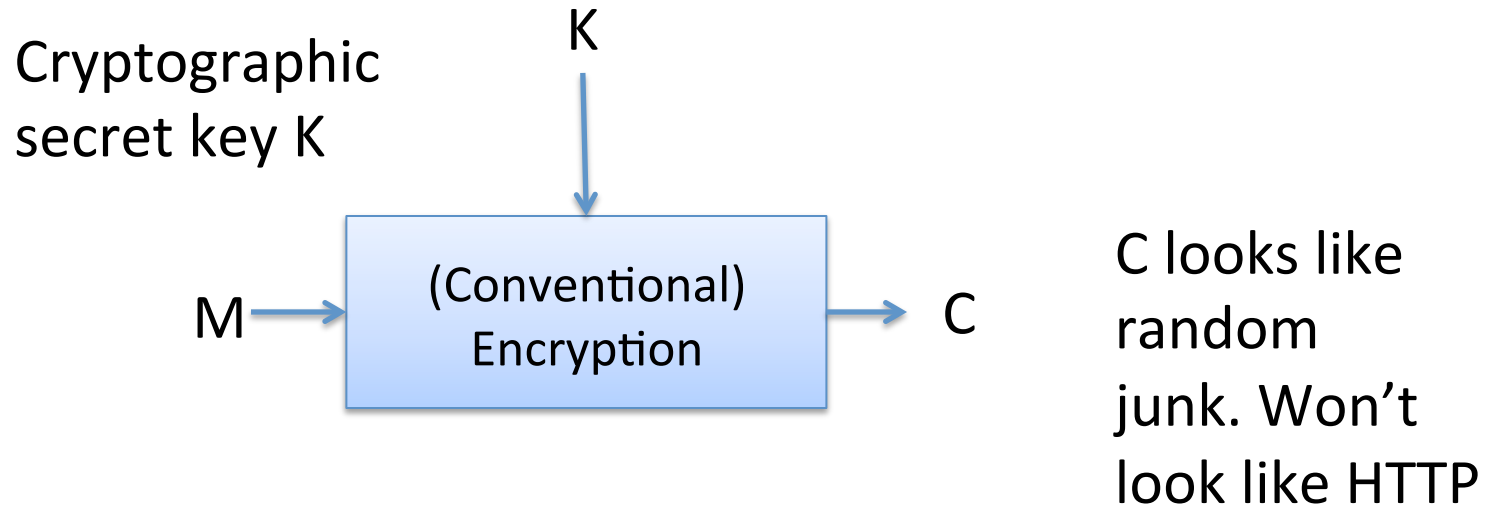


System	Look at ports?	TCP stream reassembly	Uses regex's	Use's C/C++
AppID	Yes	No	Yes	No
L7-filter	Yes	No	Yes	No
Yaf	Yes	Yes	Yes	No
Bro	Yes	Yes	Yes	Yes
nProbe	No	Yes	Not explicitly	Yes
Proprietary*	Yes	Yes	?	?

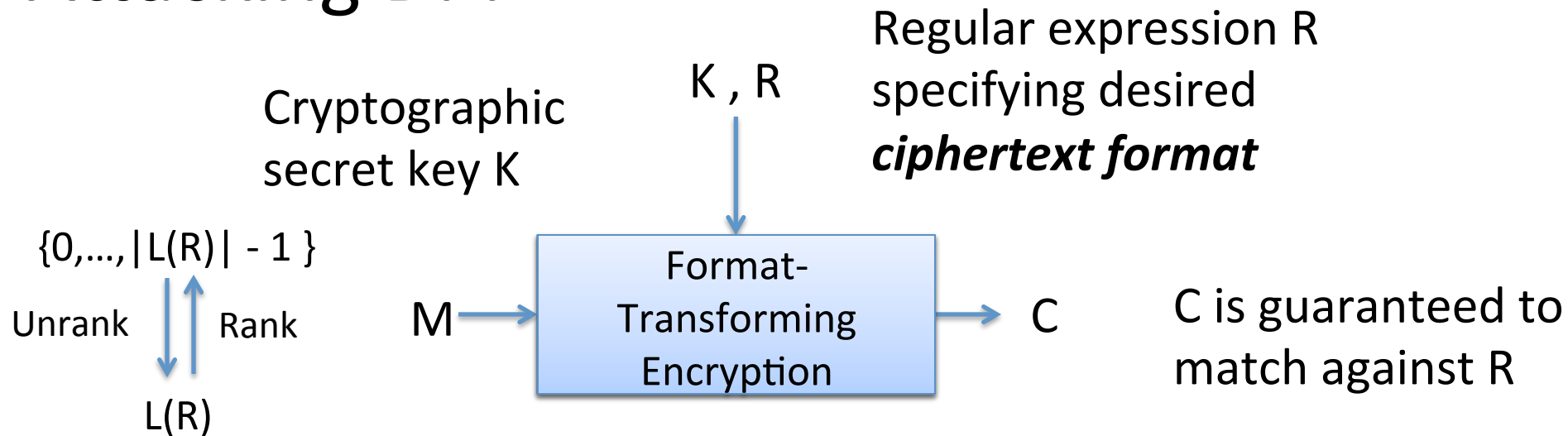
* Hint: it's a serious product (~\$10k) and similar ones seem to be used in Iran.

Can we build encryption schemes that fool regex-based systems?

Attacking DPI



Attacking DPI



Ranking for DFAs (1985).
But want it now for regexes:

[Goldberg-Sipser '85]
[BRRS '09]

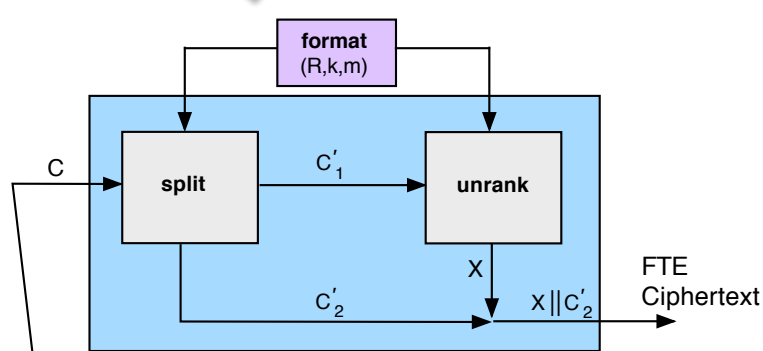
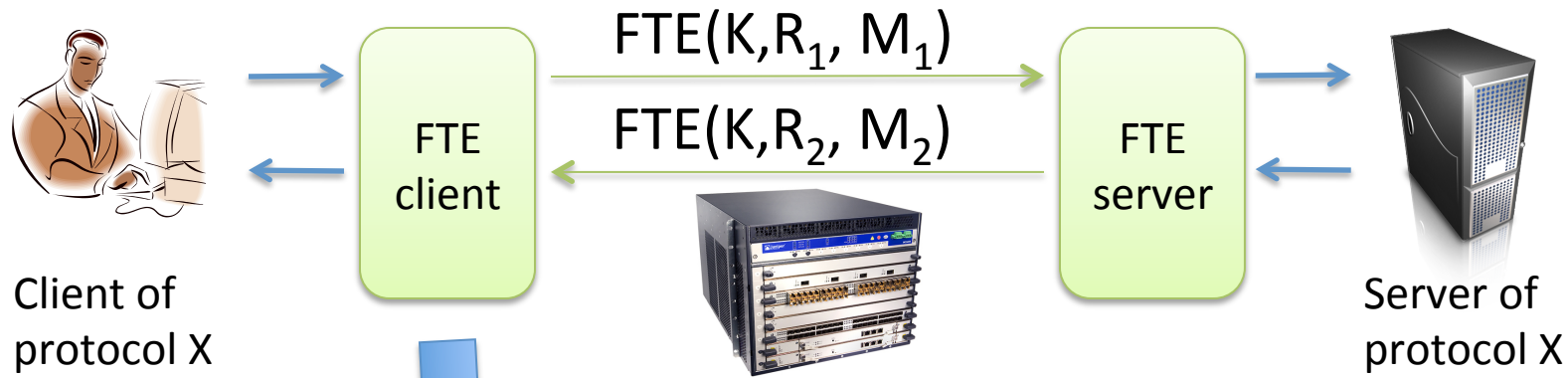


Exponential blow-up in worst case. We show that here it is ok.

Approximate ranking for NFAs

[Luchaup, Jha, R., Shrimpton –
In preparation 2013]

We build a complete FTE record layer and proxy system



- Encrypts stream of data with AE using shared keys
- Splits up ciphertext into unrankable chunks
- Library of regexes specifying formats
- Handles regex negotiation with server
- Many efficiency hurdles

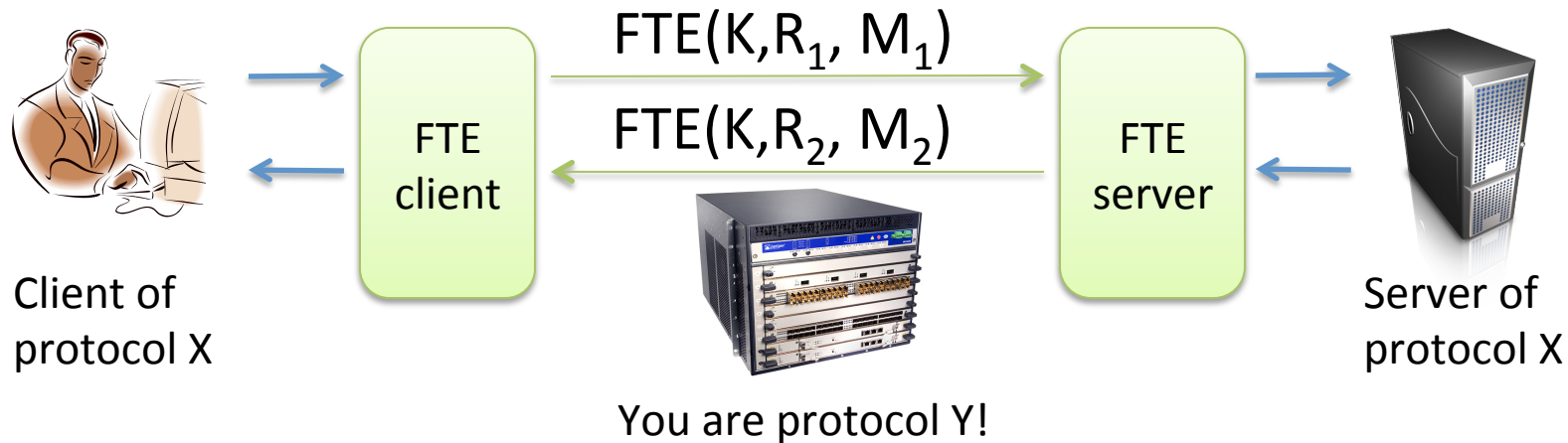
Encrypted Record

$CTR_{K1}(|M| || M)$

$HMAC_{K2}(\cdot)$

T

We build a complete FTE record layer and proxy system

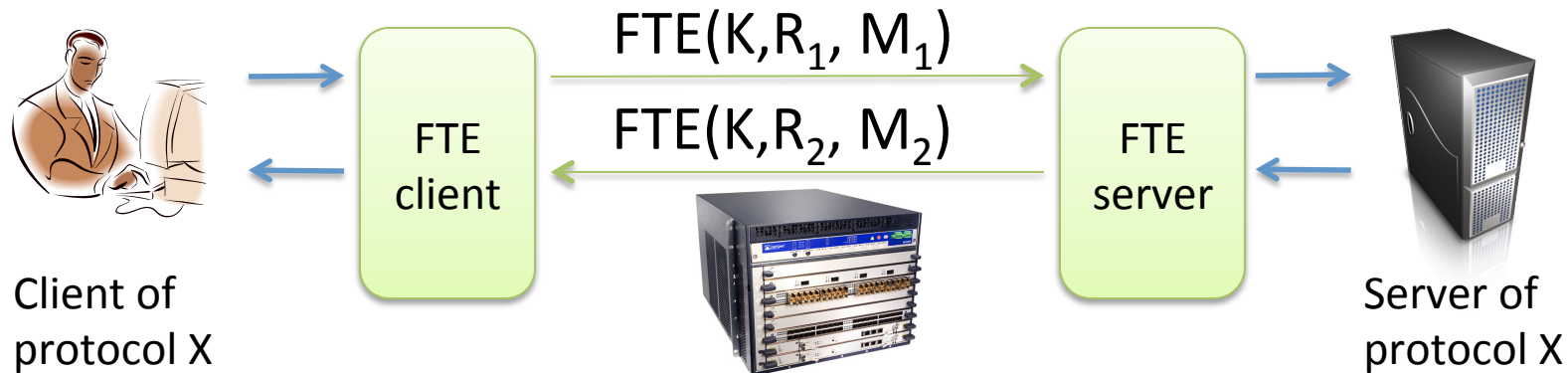


Want to trick DPI into thinking we're protocol Y $\neq$ X
Where do we get R_1 and R_2 ???

- (1) Easy to manually craft
- (2) Get from DPI themselves
- (3) Learn from traffic samples

We build regexes for variety of “cover” protocols:
 $Y = \text{HTTP, SSH, SMB, SIP, RTSP}$

FTE forces protocol misidentification

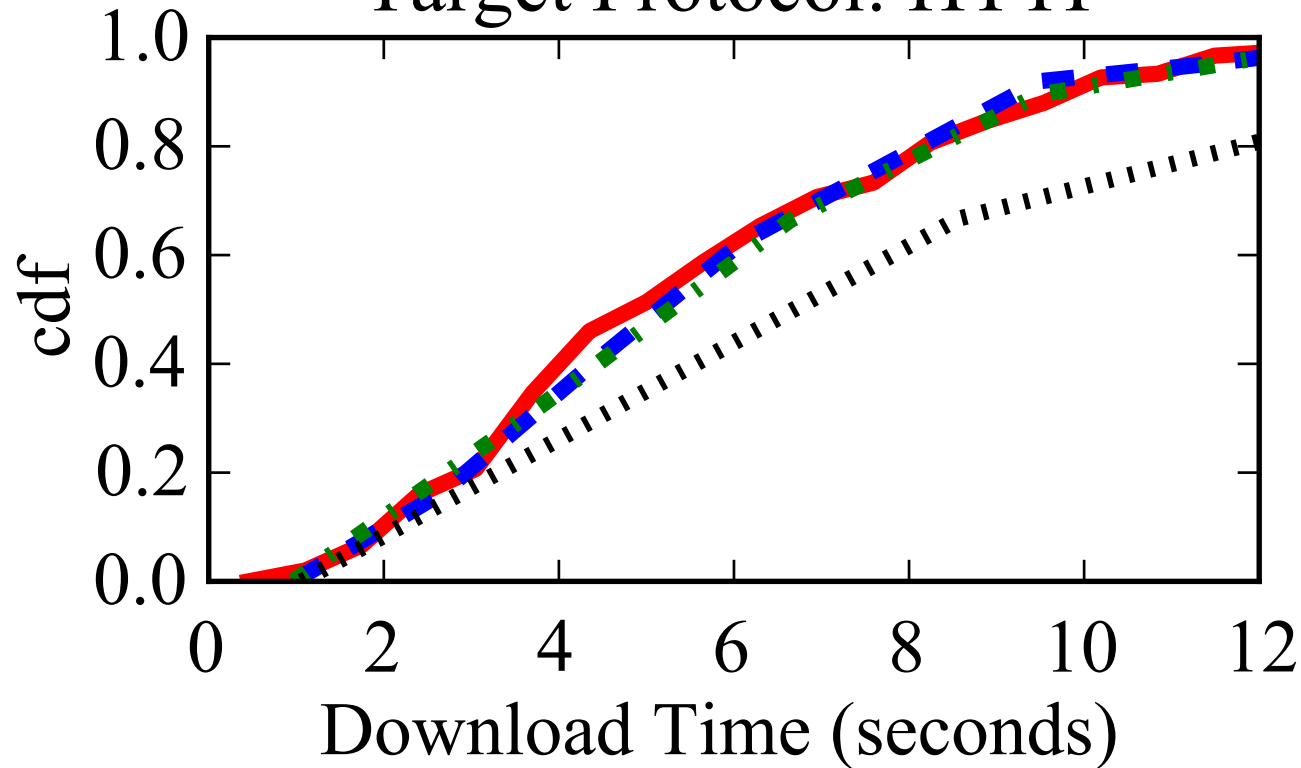


Tests with gets on Alexa Top 50 sites (X = mix of HTTPS/HTTP)

R_1 R_2 set to HTTP, SSH, SMB, and more. When do we trick DPI ?

System	DPI-derived regex's	Manual regex's	Learned regex's
AppID	Always	Always	Always
L7-filter	Always	Always	Always
Yaf	Always	Always	Always
Bro	Sometimes	Always	Always
nProbe	Never	Always	Almost always
Proprietary	Always	Always	Always

Target Protocol: HTTP

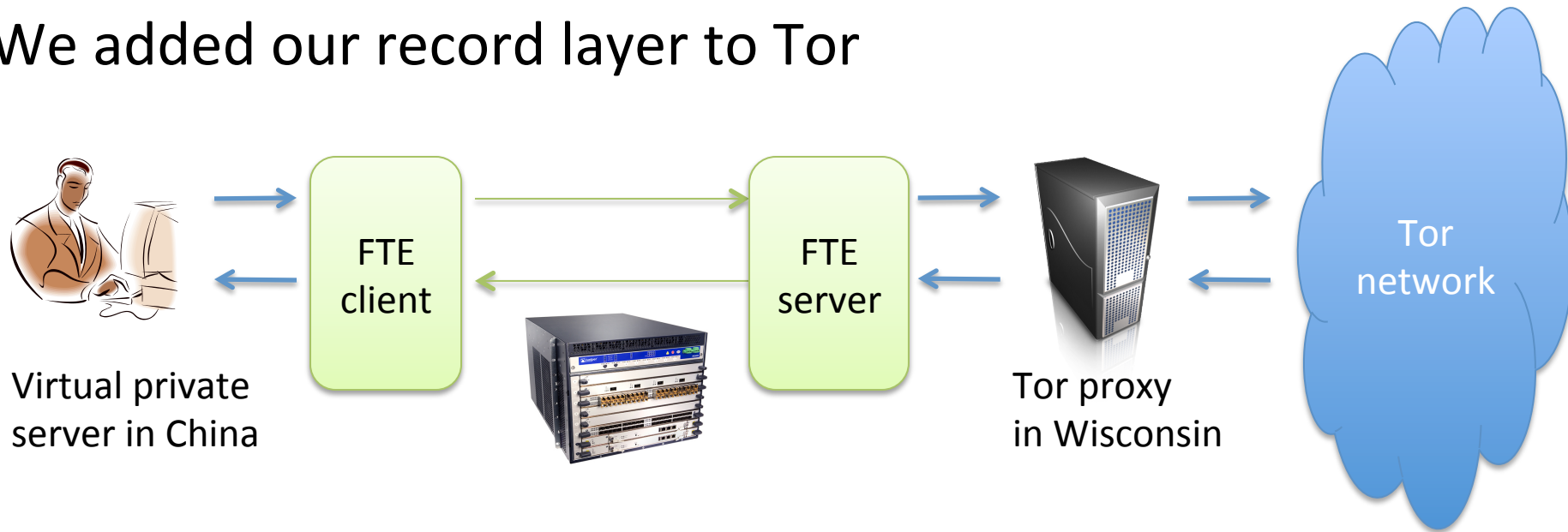


Top 50 Alexa websites

- socks-over-ssh
- intersection
- manual
- auto

42 Mbps goodput (manual/extracted regexes)
2 Mbps goodput (learned regexes)
58 Mbps goodput baseline

We added our record layer to Tor



We rented virtual private server in China. Setup Tor proxy in WI

Confirmed vanilla Tor blocked on this setup

FTE never blocked as we expected

Working with Tor team to put FTE in main bundle

Examples of deficiencies:



Deep packet inspection systems can **block** protocols

- Encryption tools are easy to detect
- Format-transforming encryption to trick DPI



Dropbox **has access** to your data

- Encryption doesn't allow deduplication to save space
- Message-locked encryption to support dedup



Lastpass uses password-based encryption that can be **cracked**

- Encryption reveals when wrong key is used to decrypt
- PBKDFs and honey-encryption

Deduplication



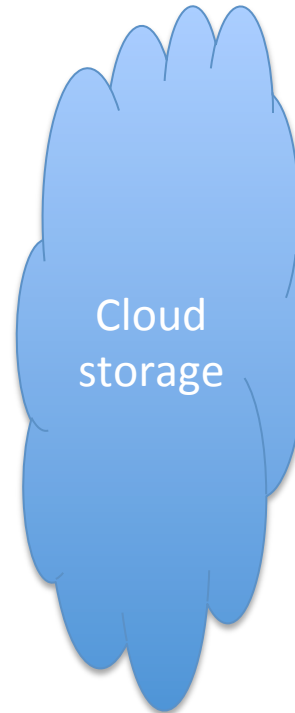
User A

“myFile”, 010101...



User B

“theFile”, 010101...



User	Filename	Contents
A	myFile	010101...
B	theFile	010101...

Dropbox saves on storage by storing (logically) only one copy of file contents

Deduplication

Find duplicate files and remove redundant copies

Dedup doesn't work with conventional client-side encryption

$$C_A = E(K_A, M)$$



User A

“myFile”, C_A



$$C_B = E(K_B, M)$$



User B

“theFile”, C_B



User	Filename	Contents
A	myFile	C_A
B	theFile	C_B

Service must
store both
 C_A and C_B

Even when file contents M same for both users, C_A and C_B will appear to be independent, random bit strings



EMC²



Deduplication saves
space

$$K_A \leftarrow \text{KeyGen}()$$

$$C_A \leftarrow E(K_A, M)$$

Encryption doesn't
allow deduplication



TUESDAY, APRIL 12, 2011

How Dropbox sacrifices user privacy for cost savings

by Christopher Soghoian



Corporations outsource storage/backups and deduplication
But: data must be encrypted before it leaves network



Prior work



(Distributed) storage literature:

[Batten et al. '01]
[Douceur et al. '02]
[Cox et al. '02]
[Cooley et al. '04]
[Killijian et al. '06]
[Wilcox-O'Hearn, Warner '08]
[Storer et al. '08]
... (many more)

Convergent
encryption (CE)

Systems:

Flud
TahoeFS
Ciphertite
GNUnet

Companies:

bitcasa

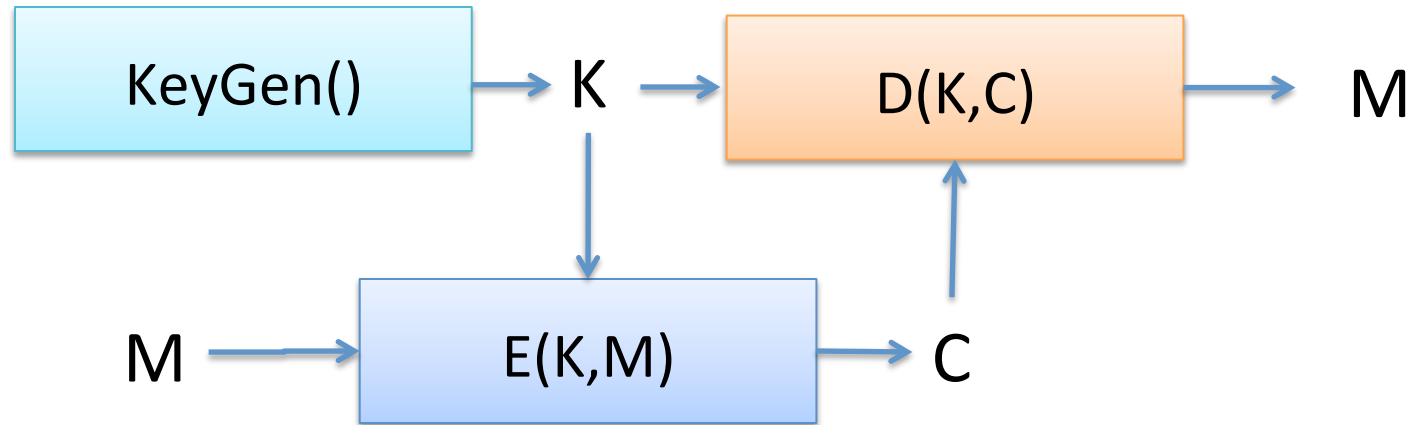
Crypto literature:

(this space intentionally left blank)

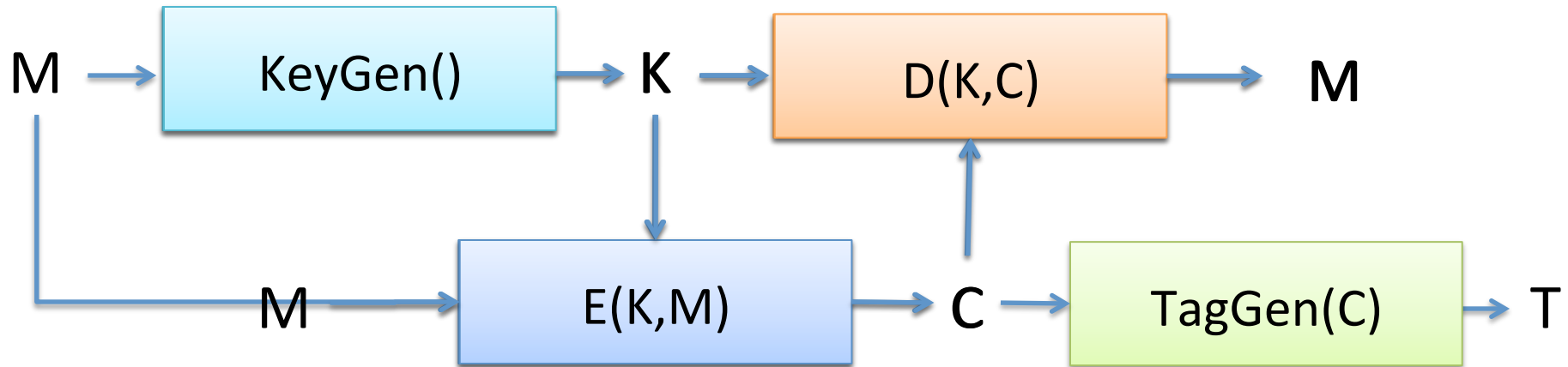
What security is achieved?

Are there other approaches?

Conventional symmetric encryption has
3 algorithms:



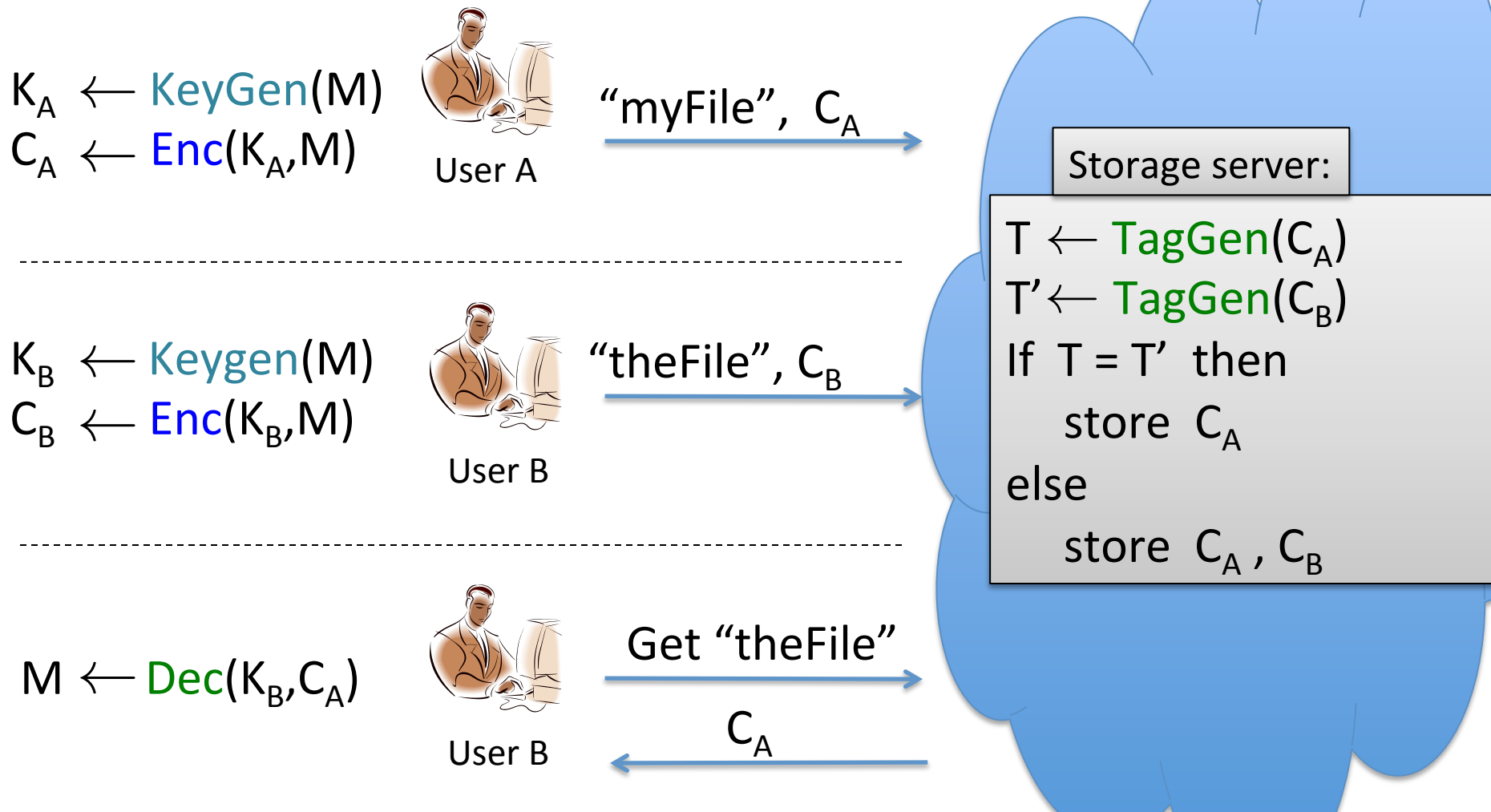
We formalize a new cryptographic primitive: Message-Locked Encryption (MLE)



- **Keygen**, **Encrypt** may be randomized
- **TagGen**, **Dec** are deterministic

The big idea:
message is “shared secret material” used to derive keys

Using MLE with deduped storage



K_A, K_B can be encrypted and stored using conventional scheme
Space savings when: $|K_A|, |K_B| \ll |M|$

Message privacy security?

Let $\{M_1, \dots, M_m\}$ be set of possible messages

Let $C \leftarrow \text{Enc}(\text{KeyGen}(M_i), M_i)$ for random i and give adversary C

BruteForce(C):

$T \leftarrow \text{TagGen}(C)$

For $j = 1$ to m do

$K_j \leftarrow \text{KeyGen}(M_j)$

$C_j \leftarrow \text{Enc}(K_j, M_j)$

$T_j \leftarrow \text{TagGen}(C_j)$

If $T = T_j$ then

Return M_j



Works against any scheme

Runs in time $O(m)$

Privacy for **MLE** schemes only possible
for unpredictable messages

Convergent encryption

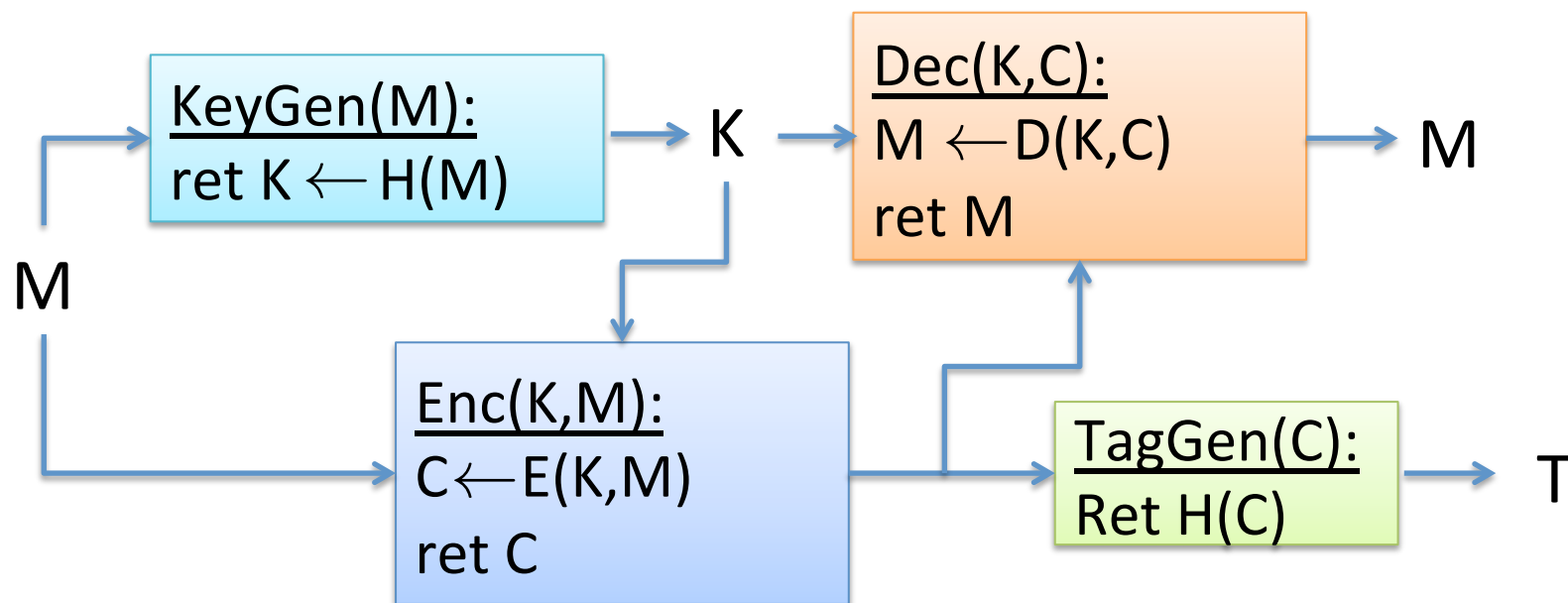
[Douceur et al. 2002]

[Pettitt '96] [Clarke et al. '00]

[Wilcox-O'Hearn '00]

Deterministically encrypt M under cryptographic hash $H(M)$

CE as an MLE scheme:

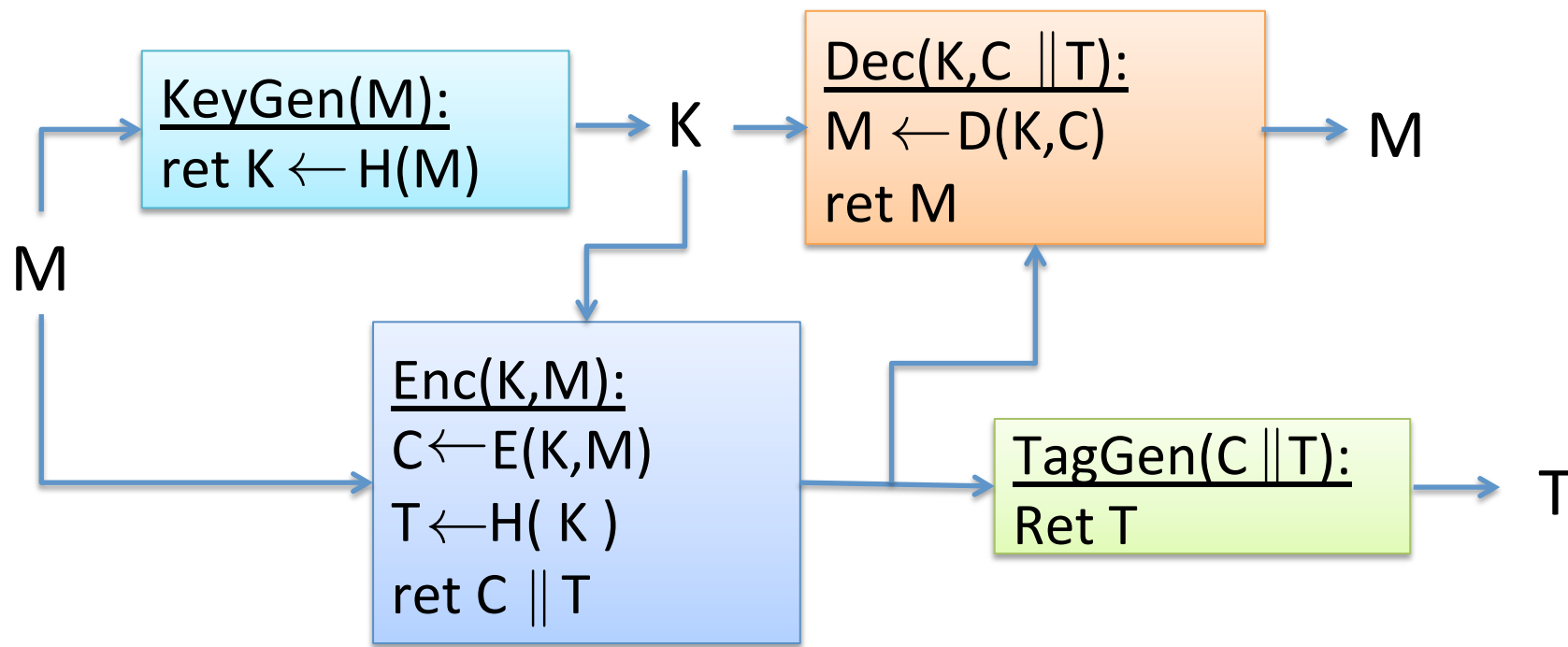


E is deterministic symmetric scheme E (decryption via D)
(e.g., CTR-mode AES with constant IV)

Non-triviality: $|K| = 128$ bits while M can be arbitrary length

Hash-and-CE (HCE) scheme

Used in TahoeFS,
elsewhere



In paper two new schemes:

- Hash-and-CE 2 with tag check (**HCE2**)
- Randomized CE (**RCE**) that achieves single-pass **MLE**

Why? All three schemes are faster than CE

Security analysis of fast MLE schemes

In-use CE and variant HCE + 2 new schemes HCE2 and RCE

2 new privacy definitions

2 new integrity definitions

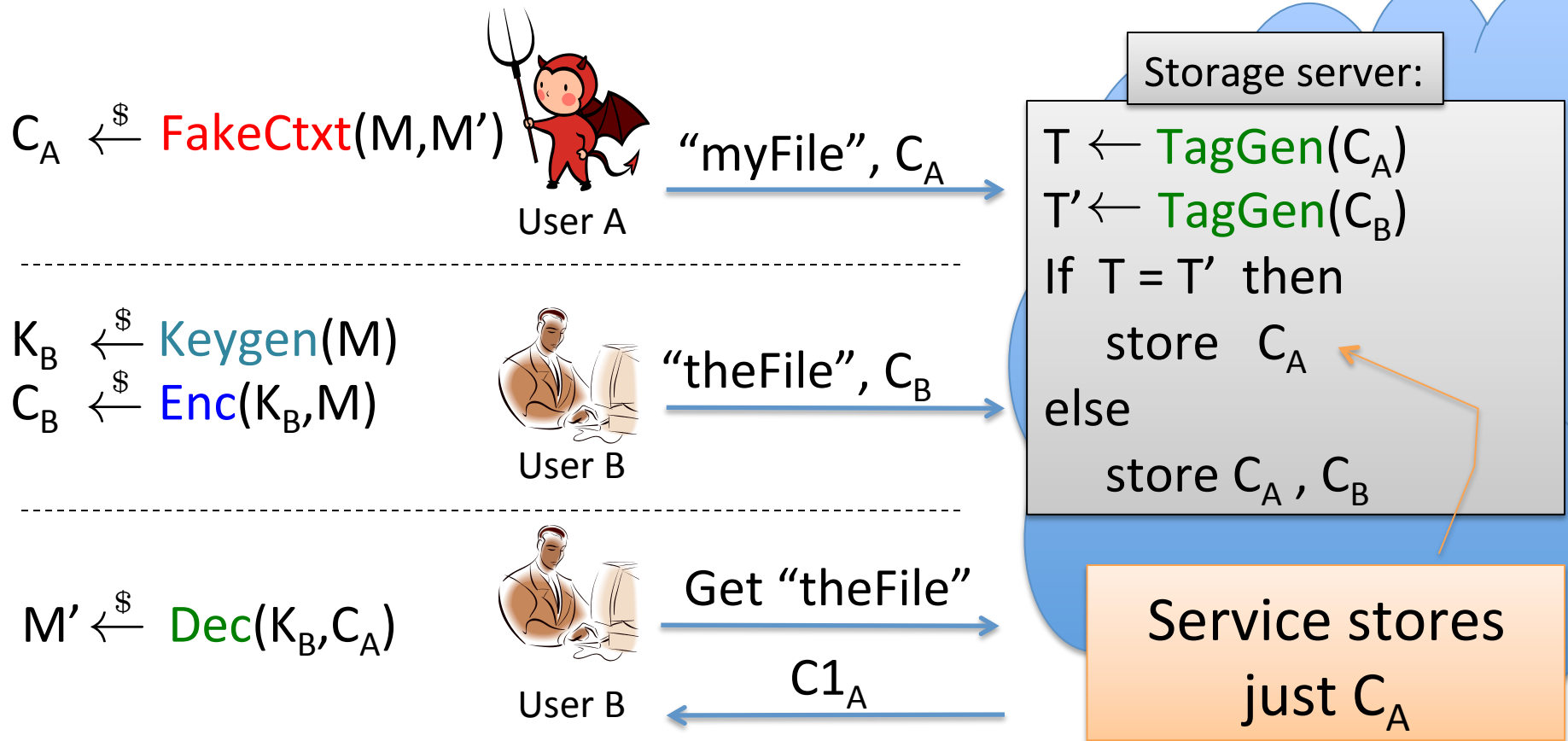
Scheme	KeyGen + Enc + TagGen time	PRV-CDA	PRV\$-CDA	TC	STC
CE	11.8 cpb	Yes	Yes	Yes	Yes
HCE	6.6 cpb	Yes	Yes	No	No
HCE2	6.6 cpb	Yes	Yes	Yes	No
RCE	6.5 cpb	Yes	Yes	Yes	No

Proofs of message privacy for unpredictable message spaces

Proofs of, or attacks against, integrity

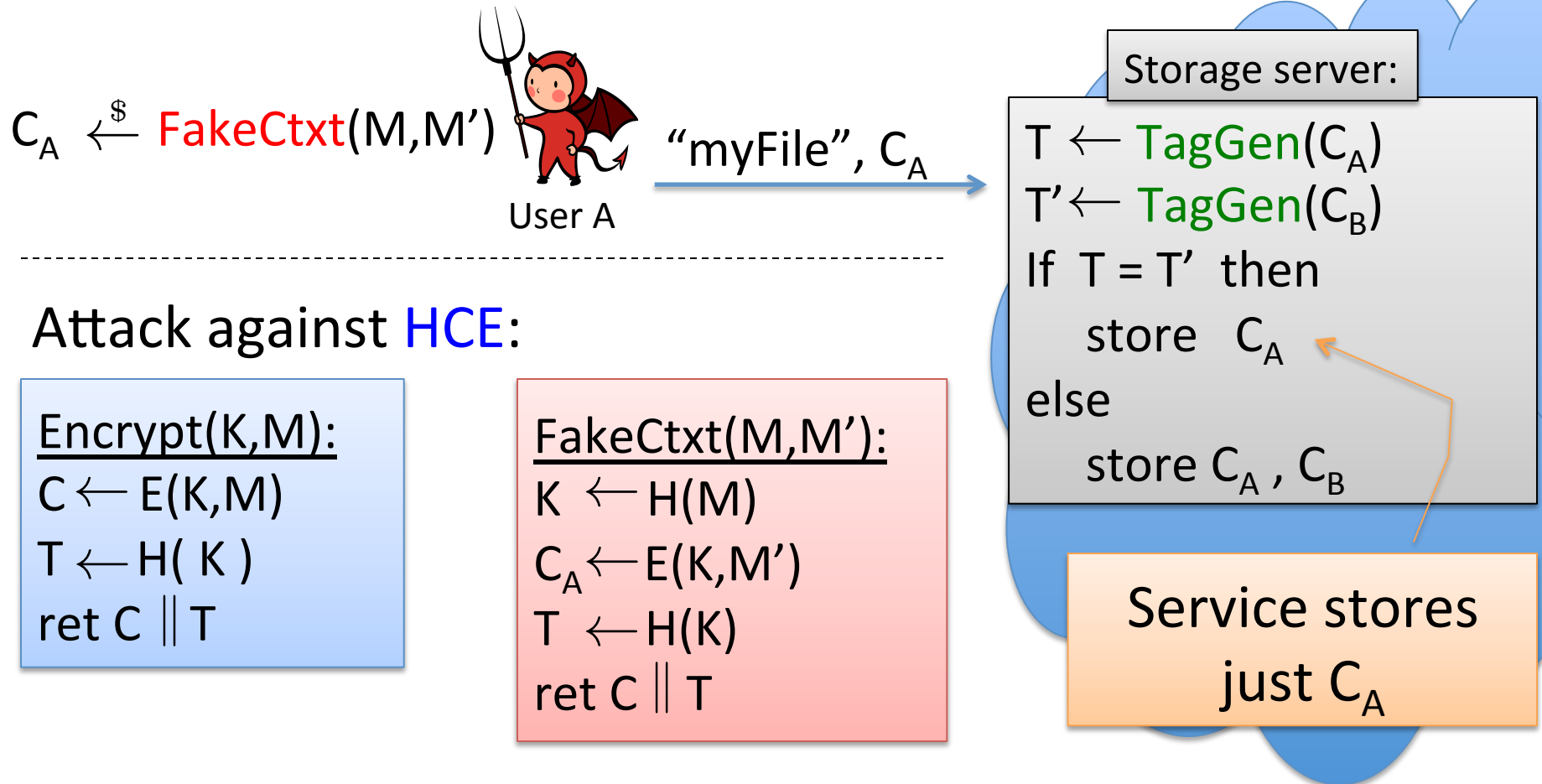
Attack against HCE used by TahoeFS that erases user's messages

Duplicate faking attacks and MLE integrity



- 1) Adversary knows M , makes fake C_A , uploads it
- 2) User B uploads honestly-generated C_B , server dedups
- 3) User B later gets back corrupted file!

Duplicate faking attacks and MLE integrity



Similar attack in [Storer et al. '08], but vulnerabilities not realized

Weaker attack would just make decryption fail

Security analysis of fast MLE schemes

In-use CE and variant HCE + 2 new schemes HCE2 and RCE

2 new privacy definitions

2 new integrity definitions

Scheme	KeyGen + Enc + TagGen time	PRV-CDA	PRV\$-CDA	TC	STC
CE	11.8 cpb	Yes	Yes	Yes	Yes
HCE	6.6 cpb	Yes	Yes	No	No
HCE2	6.6 cpb	Yes	Yes	Yes	No
RCE	6.5 cpb	Yes	Yes	Yes	No

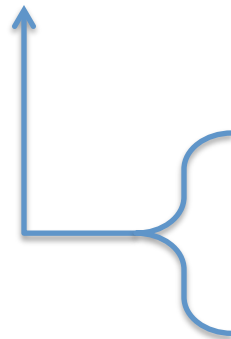
Proofs of message privacy for unpredictable message spaces

Proofs of, or attacks against, integrity

Attack against HCE used by TahoeFS that erases user's messages

MLE leaks nothing about messages...
if messages are unpredictable

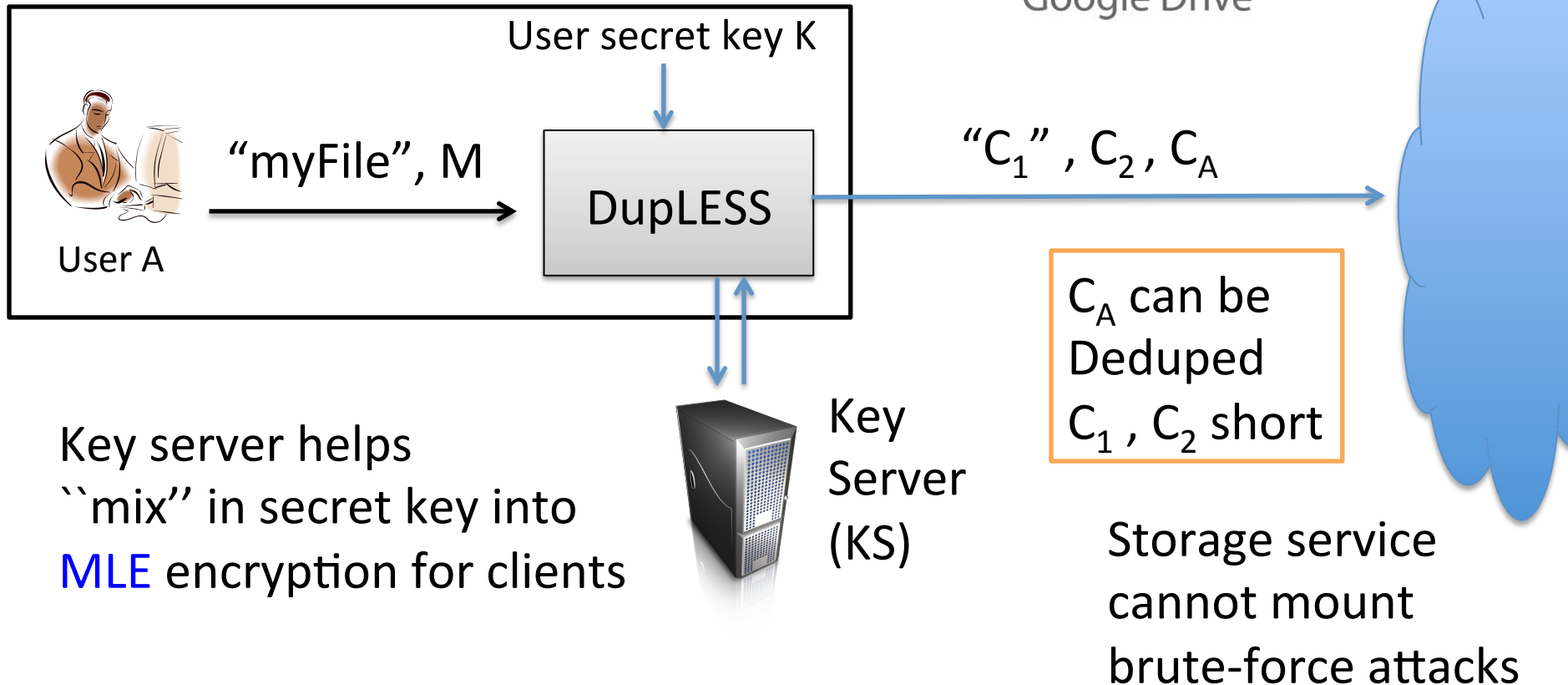
Attacker recovers M given $\text{Enc}(\text{KeyGen}(M), M)$
in time $O(m)$ when m is # of possible messages



- 1) Sometimes $m = 2$
- 2) Hard for defenders to determine m

DupLESS (DuplicateLess Encryption for Simple Storage)

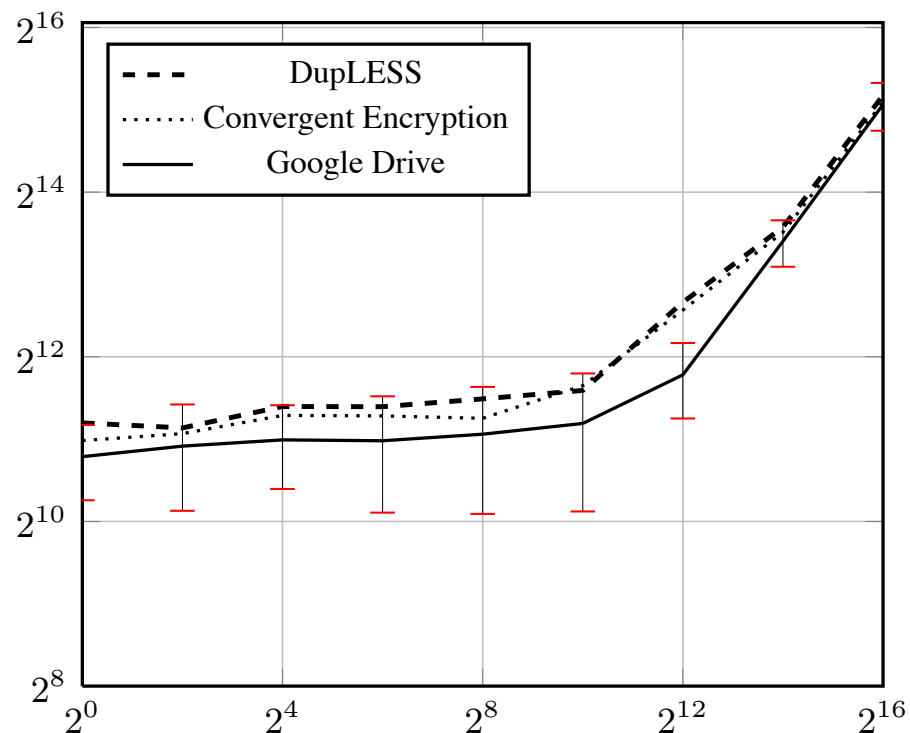
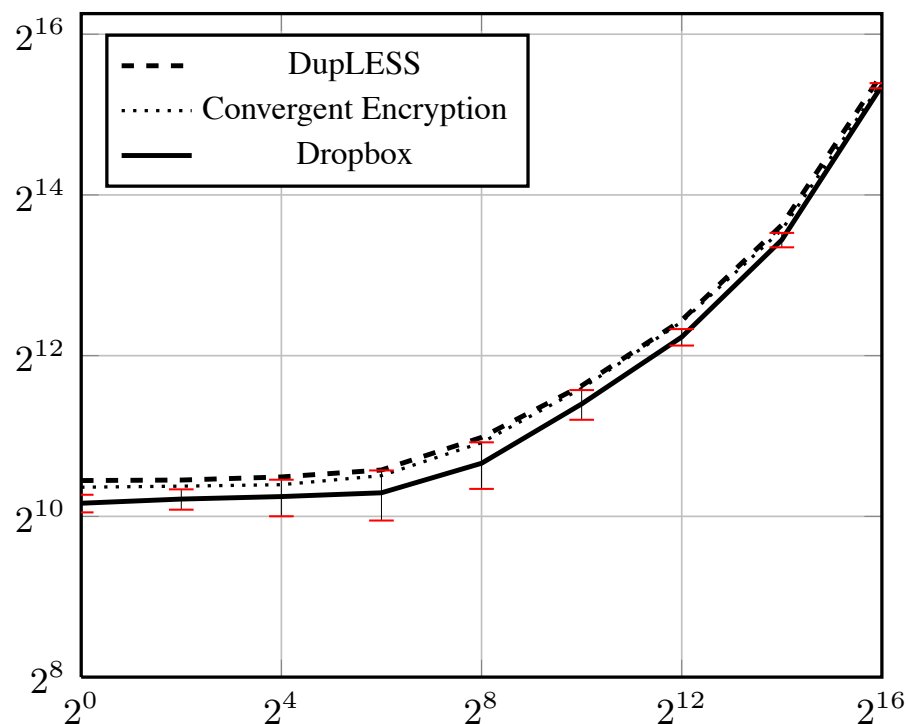
Provides protection against
brute-force attacks



Key server helps
"mix" in secret key into
MLE encryption for clients

Oblivious PRF protocol:
KS learns nothing about M

DupLESS performance for storage



X-axis: File size (KB)
Y-axis: Put time (ms)

Code for DupLESS available at: <http://cseweb.ucsd.edu/~skeelvec/dupless/>

Examples of deficiencies:



Deep packet inspection systems can **block** protocols

- Encryption tools are easy to detect
- Format-transforming encryption to trick DPI



Dropbox **has access** to your data

- Encryption doesn't allow deduplication to save space
- Message-locked encryption to support dedup



Lastpass uses password-based encryption that can be **cracked**

- Encryption reveals when wrong key is used to decrypt
- PBKDFs and honey-encryption

Some concluding thoughts

- Typical case: no right security definition
 - Righter ones and wronger ones
 - One primitive, multiple security goals
 - We don't yet know all the security goals
- Crypto should be designed to support applications, not other way around
 - Nonce-based SE, FPE, MLE, PW-based encryption
 - Industry has lots of really cool crypto problems
- Understanding systems key
 - First order approximation: no one in practice will care unless you implement it

Our game-plan today

We will build two widely needed primitives:

- **Nonce-based symmetric encryption**
 - Contemporary viewpoint on SE
 - Two flavors (speed versus security trade-off)
- **Format-preserving encryption**
 - Used widely in industry for fixed-field encryption (credit card numbers, etc.)
 - Length-preserving encryption as special case
- **Time allowing:** advanced SE primitives such as message-locked encryption, honey encryption, password-based encryption