Elastic Block Ciphers

Debbie Cook

Angelos Keromytis

Moti Yung

This work was done at Columbia University.

Lucent Technologies
Bell Labs Innovations

Warning Label

- This presentation contains:
  - No public key crypto
  - No protocols
  - Only one theoretical result (and it is on the last slide)
- But it is only 30 minutes

Overview

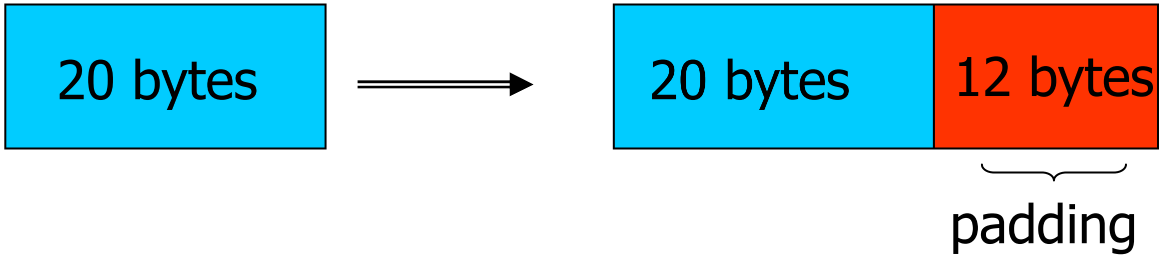
- § Introduce a method for building a variable length block cipher from any existing block cipher
  - – b bits to b+y bits, 0 ≤ y ≤ b
  - – Result called **elastic block cipher**
  - – Applications set block size
  - – Work proportional to block size
- § Relate the security of elastic version to that of the original version

Agenda

- Motivation
- Background and Related Work
- Elastic Block Cipher Algorithm
- Elastic Block Cipher Security
- Conclusions

Motivation

- Existing block ciphers work on fixed sized data blocks (typically 128 or 256 bits)
- Standard sizes provide design and implementation benefits …
- … but do not match up with real world data
- Data must be padded to an integral number of blocks



two full applications of 16 byte block cipher

Motivation

- Cryptographic support for variable size blocks provides
  - Reduced storage overhead (database applications)
  - Easier fit to MTU (IP applications)
  - Reduced cryptographic processing time

Database Example

Customer Table

User name – 20
Password – 20
First name – 15
Last name – 15
Phone – 16
Email – 50
Login date+time – 19
Expiration date+time – 19
Balance – 18
YTD payment – 18
Last visit date – 10
Key to address table - 10

Address Table

Customer id - 10
Street1 – 40
Street2 – 40
City – 30
State – 20
Zip – 10
Country - 4

Row: 154 bytes, 6 bytes padding ~3.8%
Fields: 17% unnecessary padding
24 bytes padding for fields < 16 bytes
30 bytes padding for fields > 16 bytes ~ 17%

Row: 230 bytes, 10 bytes padding ~ 4.3%
Fields: 40% unnecessary padding
14 bytes padding for fields < 16 bytes
92 bytes padding for fields > 16 bytes ~ 40% overhead

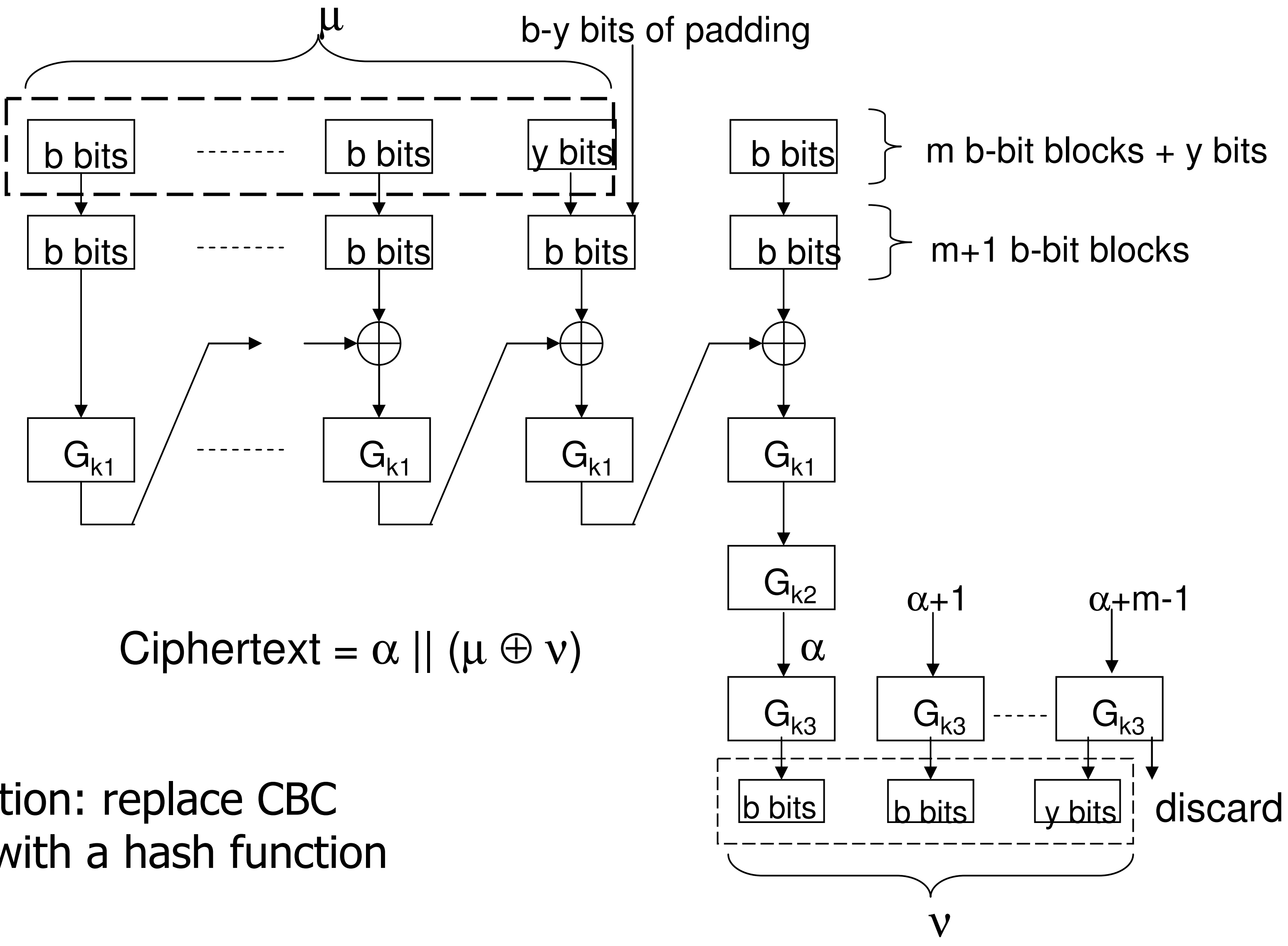
Motivation

- How to design elastic block ciphers
  - using existing block ciphers?
  - as a new block cipher?
- … while
  - maintaining provable resistance levels to attacks?
  - providing computational workload proportional to block size?

Previous Variable Length Designs

- Use of existing cipher as a black box
  - *"On the Construction of Variable Length Input Ciphers",* [Bellare and Rogaway '99]
  - *"Efficient Constructions of Variable Length Input Ciphers",* [Patel, et. al, '04]
- Design from scratch
  - *Hasty Pudding Cipher* [Schroeppel]
  - *Cellular Message Encryption Algorithm*
- Modification of modes – ciphertext stealing

Variable Length Input [BR99]

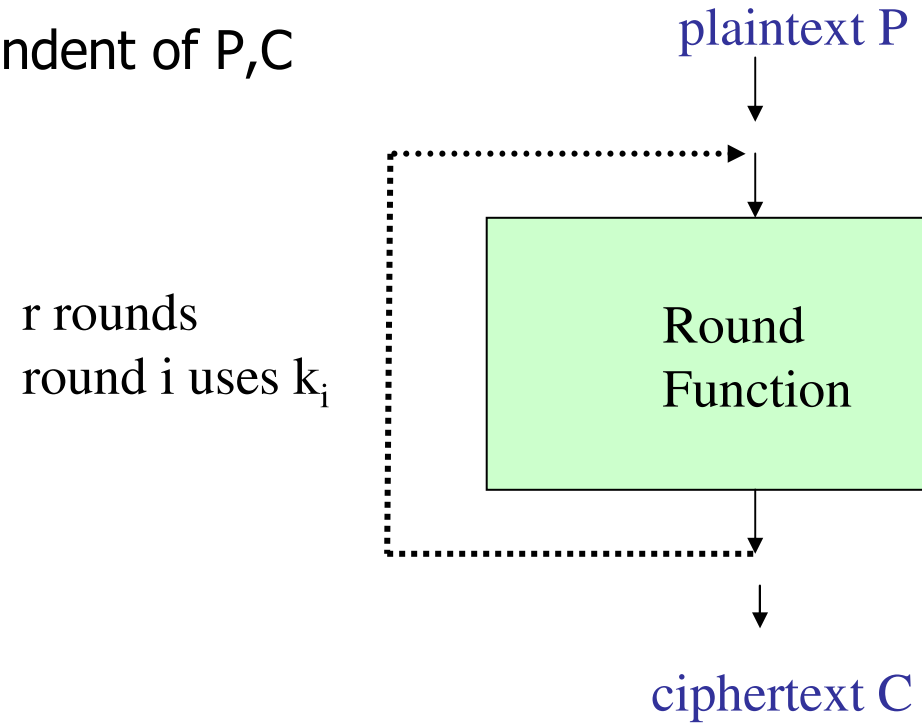


Ciphertext = $\alpha \,||\, (\mu \oplus \nu)$

Modification: replace CBC portion with a hash function

Typical Block Cipher Structure

- P,C are fixed length (*e.g.* 128 or 256 bits)
- Secret key, K, expanded via a function called a key schedule to create round keys $k_1, k_2, \ldots k_r$
- Key schedule independent of P,C

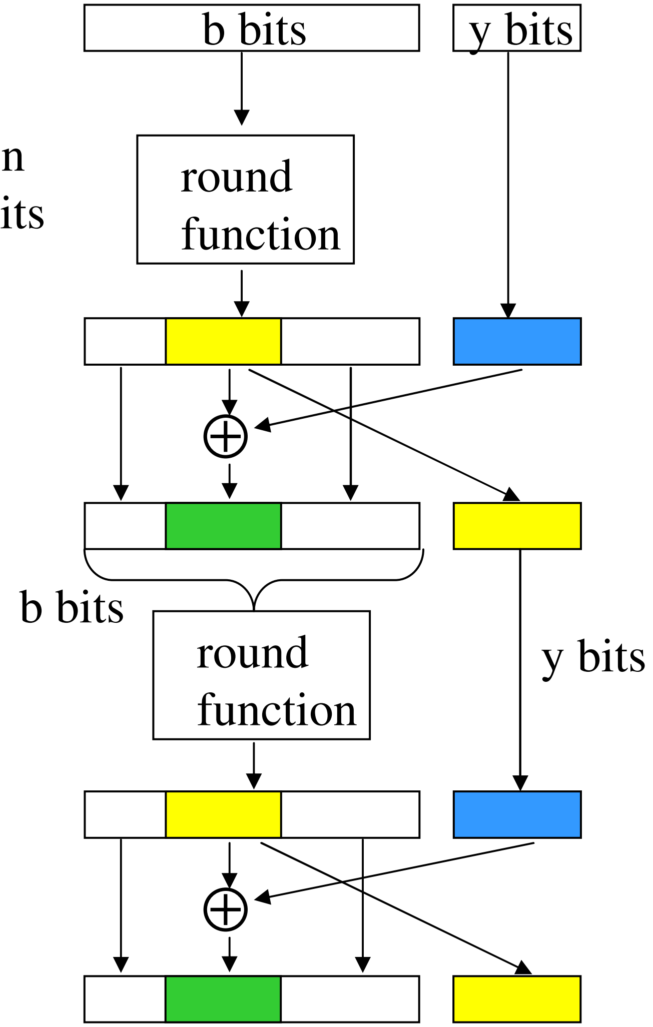


Algorithm Overview

- Creates a variable length block cipher, *G'*, from existing block cipher, *G*
  - If *G* accepts blocks of length b, *G'* accepts blocks of length b+y, for $0 \leq y \leq b$
- Works for any *G* structured as a series of rounds
  - *e.g.* AES, Camellia, DES, Misty1, RC6 …
- Falls between black-box and design from scratch approaches
- Computational workload increases proportionally to block size
- Security related to that of original cipher - reduction between elastic and original versions

General Structure

round function
maps b to b bits



b bits

y bits

round
function

b bits

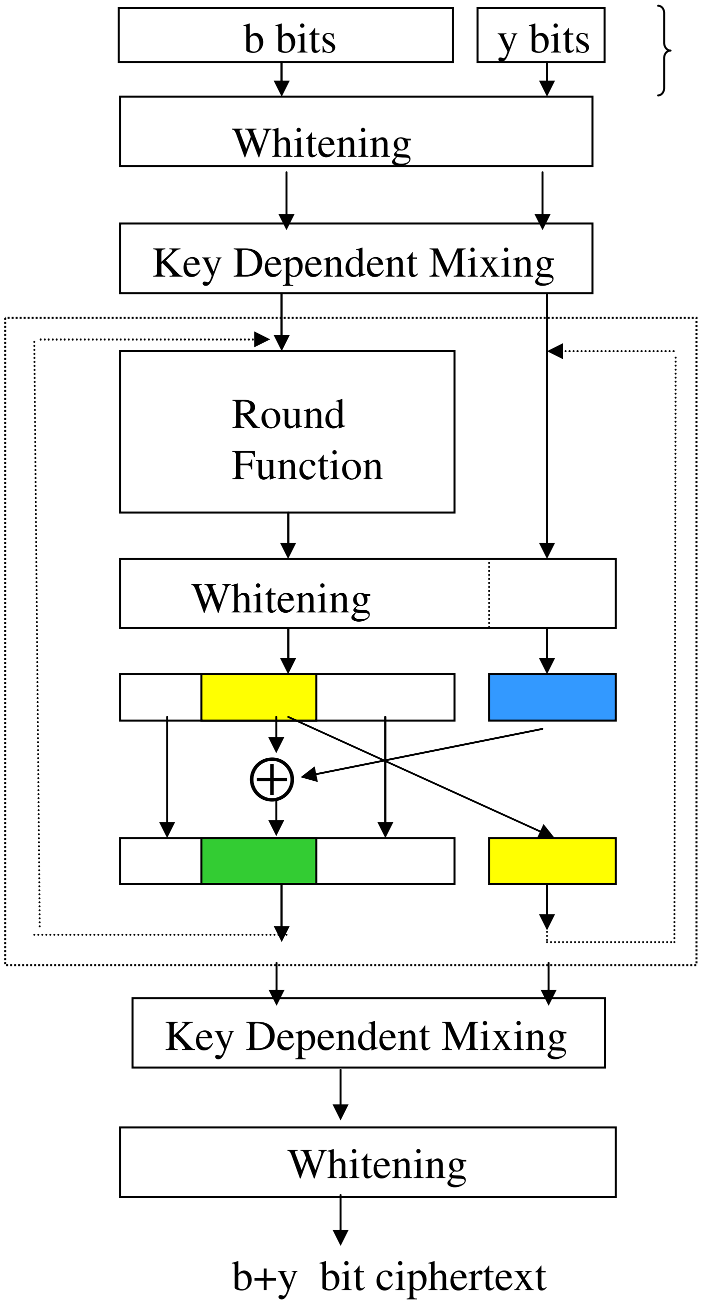
round
function

y bits

y bits XORed
and swapped

The y bit positions chosen
from the b bits for the XOR
and swap can vary per round.

Elastic Block Cipher Structure



plaintext b+y bits, $0 \leq y \leq b$ bits

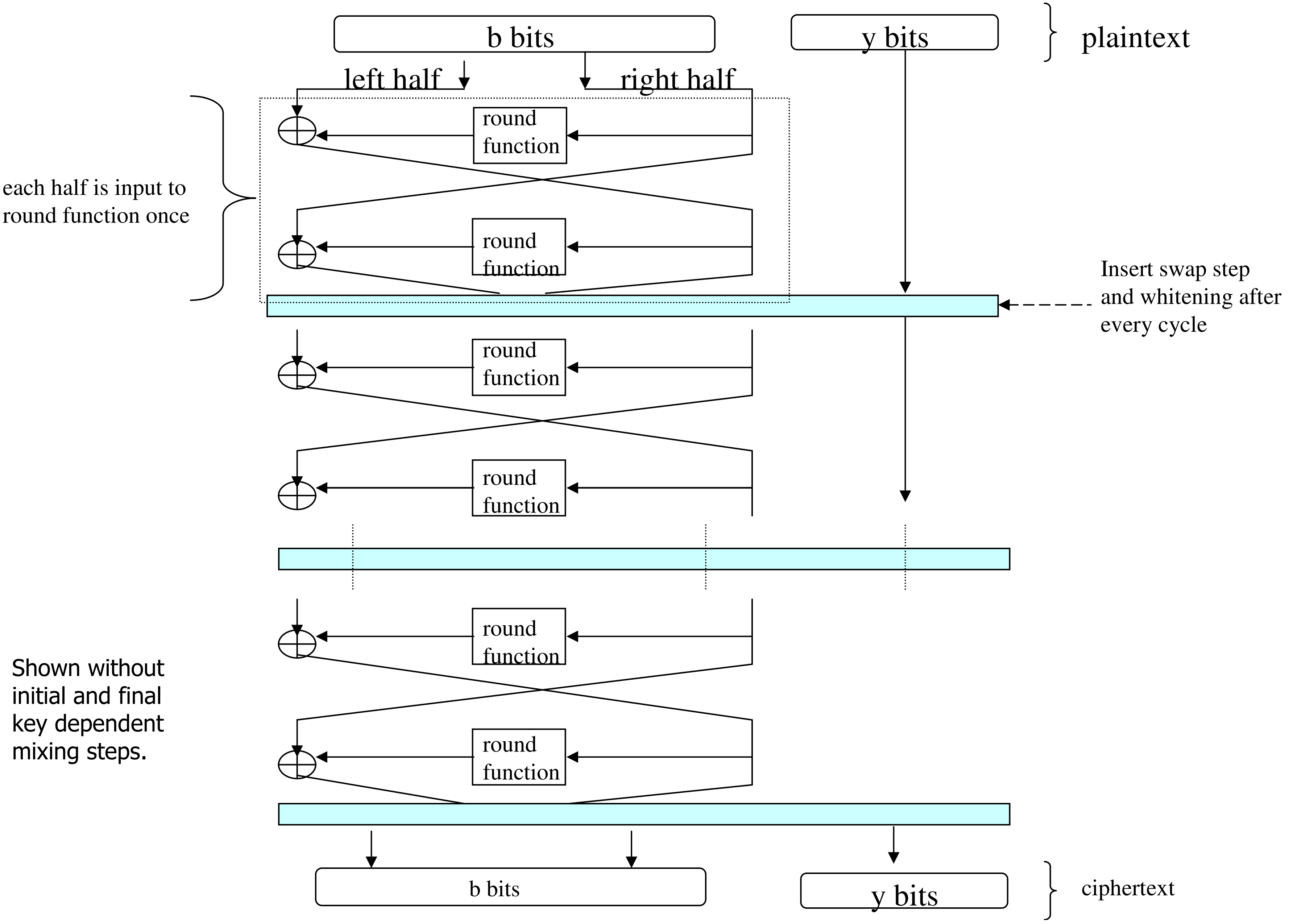
Round function

Total # of rounds $= r + \lceil ry/b \rceil$

r rounds in original cipher

Addition to round to swap y bits.
XOR y bits left out of round with
y bits that were in the round and
swap the two segments
Exact bit positions used from round’s output
in the swap varies by round.
Swap can be omitted after last round.

Elastic Version of Feistel Network



Shown without initial and final key dependent mixing steps.

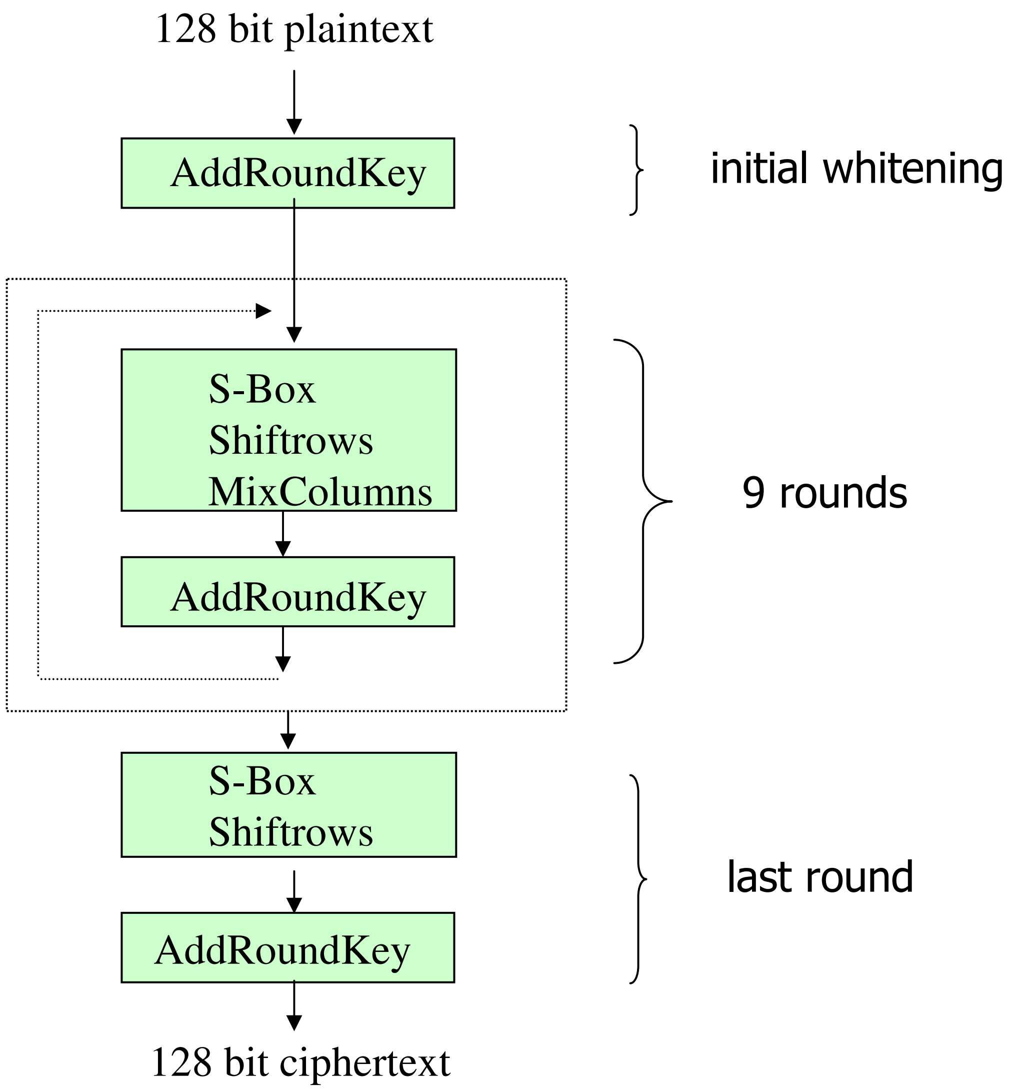
Notes on Algorithm

- Decryption
  - Round function replaced by its inverse
  - In the case of Feistel network, this is just complete cycle in reverse
- Diffusion
  - Every bit impacts 2nd through last round of *G'*
  - If complete diffusion in *G* takes i rounds, it takes at most i+1 rounds in *G'*
- Key schedule
  - *G'* needs more expanded key bits than *G*
  - Use a stream cipher

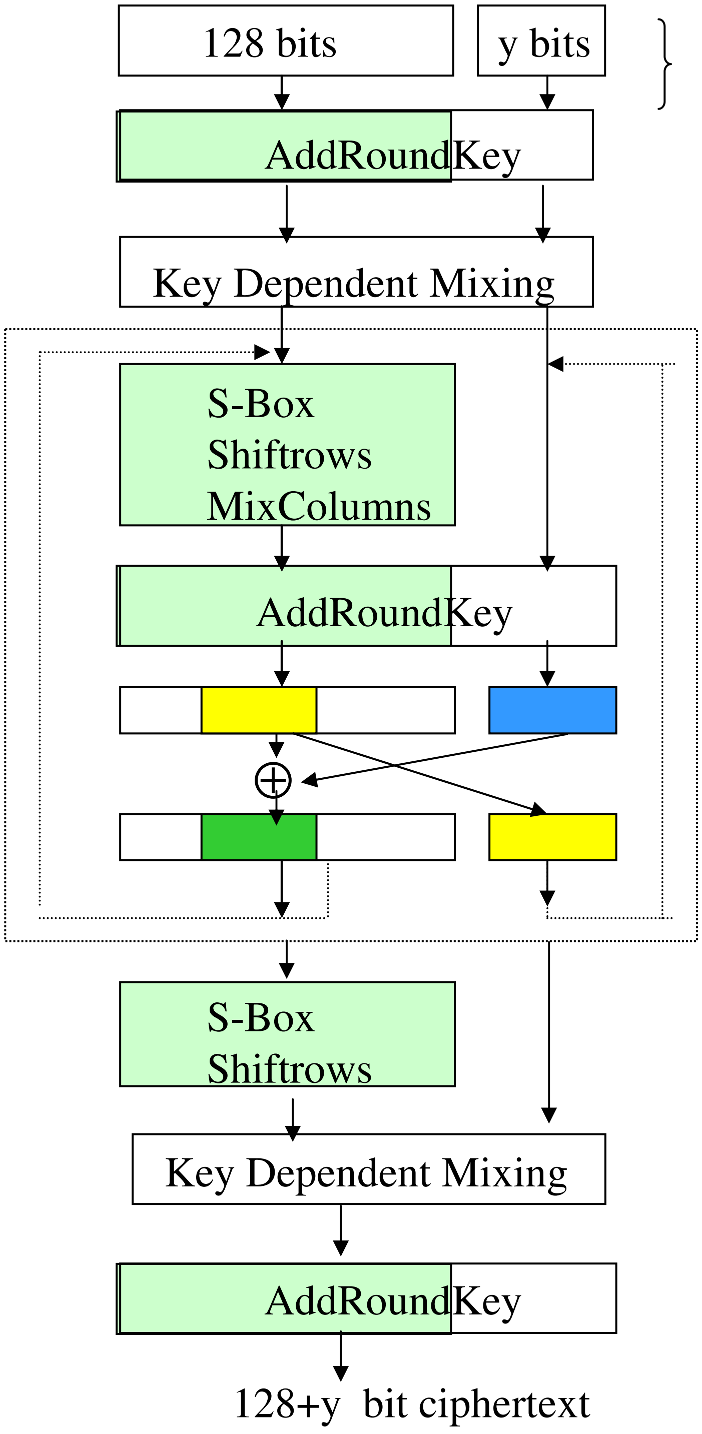
Examples

- Created 4 examples from existing block ciphers
  - AES, MISTY1, Camellia, RC6
  - Demonstrated general applicability of method
- Measured performance
  - Compared to padding
- Measured randomness of output
  - No obvious flaws – passed at same rates or higher than fixed-length versions
  - Tests used by NIST in AES competition

Example: AES – 128 bit block



Elastic Version of AES



Plaintext 128+y bits, $0 \le y < 128$ bits

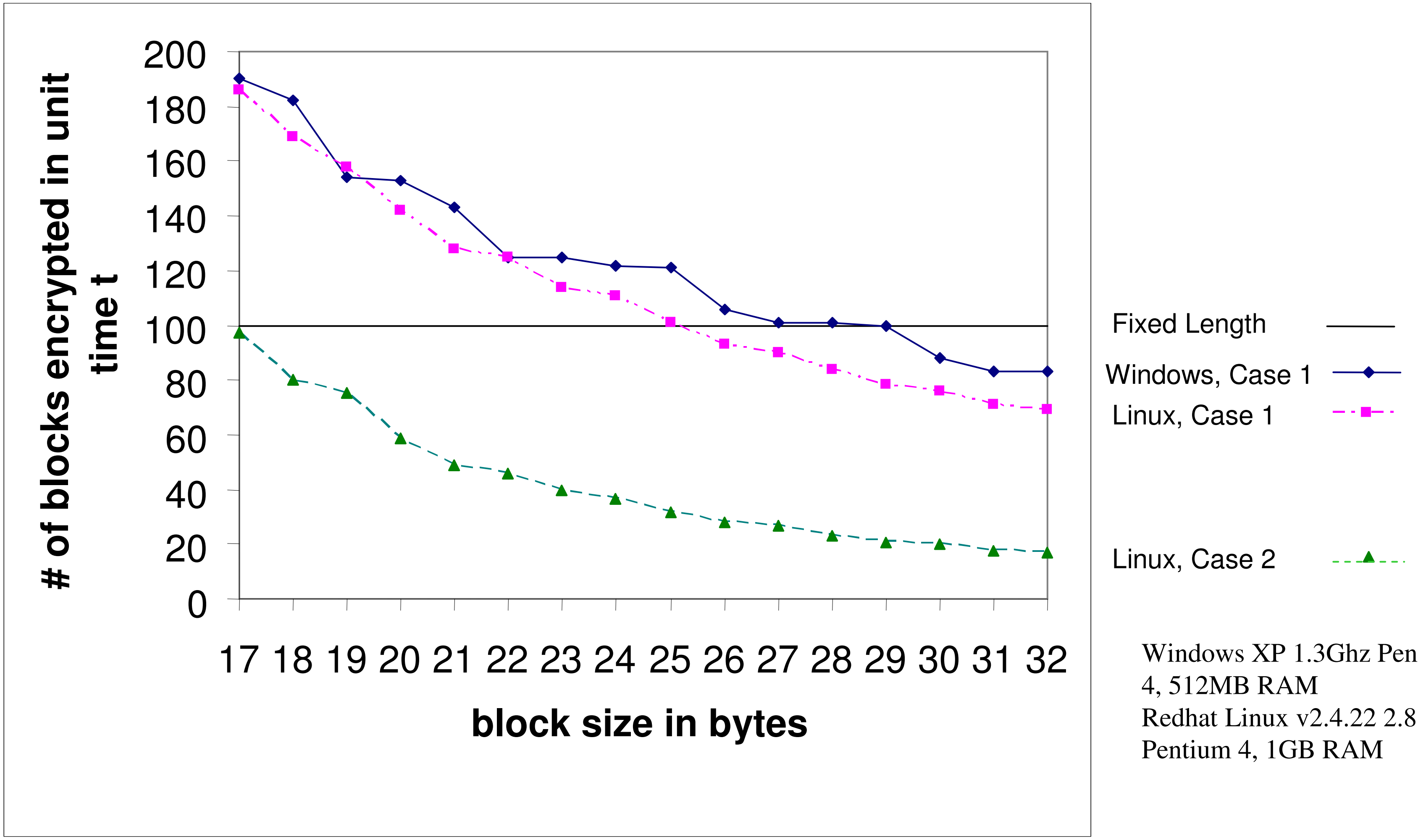
AES round, excluding last round

Total # of rounds $= \lceil 10(128+y)/128 \rceil$

Addition to round to swap y bits.
XOR y bits left out of round with
y bits that were in the round and
swap the two segments
Exact bit positions used from round’s output
in the swap varies by round.

last round

Elastic AES Performance



Windows XP 1.3Ghz Pentium 4, 512MB RAM
Redhat Linux v2.4.22 2.8 GHz Pentium 4, 1GB RAM

Case 1: steps within round performed in sequence, table lookups for MixColumns
Case 2: entire round performed as 32-bit table lookups and XORs

Security

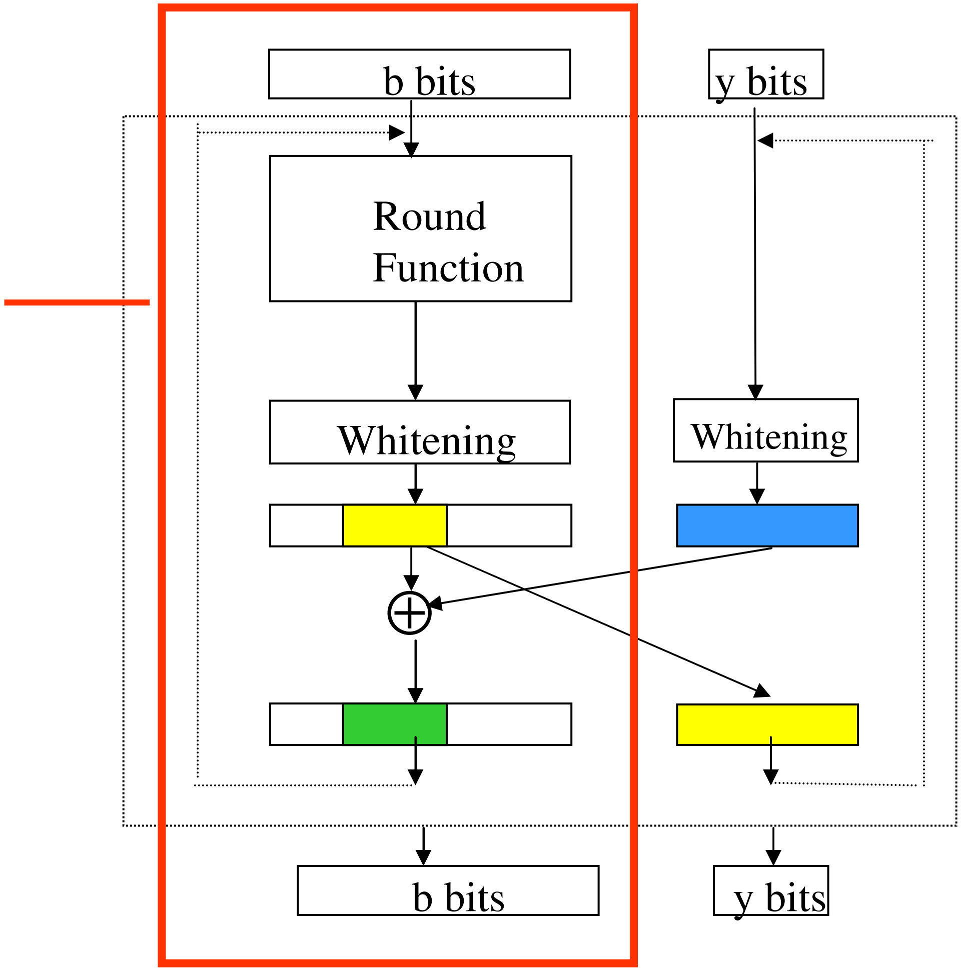
- General
  - Relate security of *G'* against key recovery to that of *G* via a reduction
  - All concrete attacks (linear, differential …) attempt (round) key recovery
  - Independent of specific cipher
  - Allows the security of elastic version to be defined in terms of original *b* bit block size version of the cipher - avoid analyzing the elastic version against every attack
- Independently, considered specific attacks
  - linear and differential cryptanalysis

Relation between Security of *G'* and *G*

- If there exists an attack on *G'* that allows the round keys to be determined for r consecutive rounds, then there exists an attack on *G* with r rounds that finds the round keys for *G* and uses polynomial many resources as the attack on *G'.*
- Assumptions:
  - Attack on r' rounds of *G'* implies attack on r < r' round version of *G'*
  - Expanded key bits depend only on the key and not on the plaintext. This is true of ciphers used in practice.
- Notes:
  - Whitening
  - Keys bits in *G'* taking on same values as in *G*
  - Analyze *G'* without initial/final mixing steps

Intuition: *G* in *G’*

Round of *G*
(plus whitening)



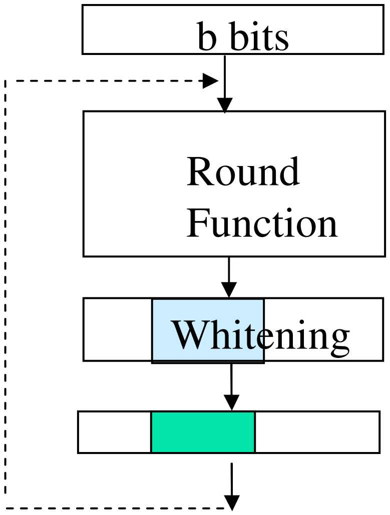
Round of *G′*

- Instance of *G* embedded in *G’*
- Allows round keys for *G’* to be converted to round keys for *G*

Attack Conversion

- Two methods:
  - Independent of selection process for swapping bits
- First method:
  - Given a set $\{(P_i', C_i')\}$ of size n and round keys for *G'*, find round keys for *G* that correspond to a set $\{(P_j, C_j)\}$ of size $\leq n$
  - Less work than second attack
  - But only useful when y is small
- Second method:
  - Given a set $\{(P_j, C_j)\}$ of size n for *G*, use the attack on *G'* to find round keys for *G* corresponding to $\{(P_j, C_j)\}$

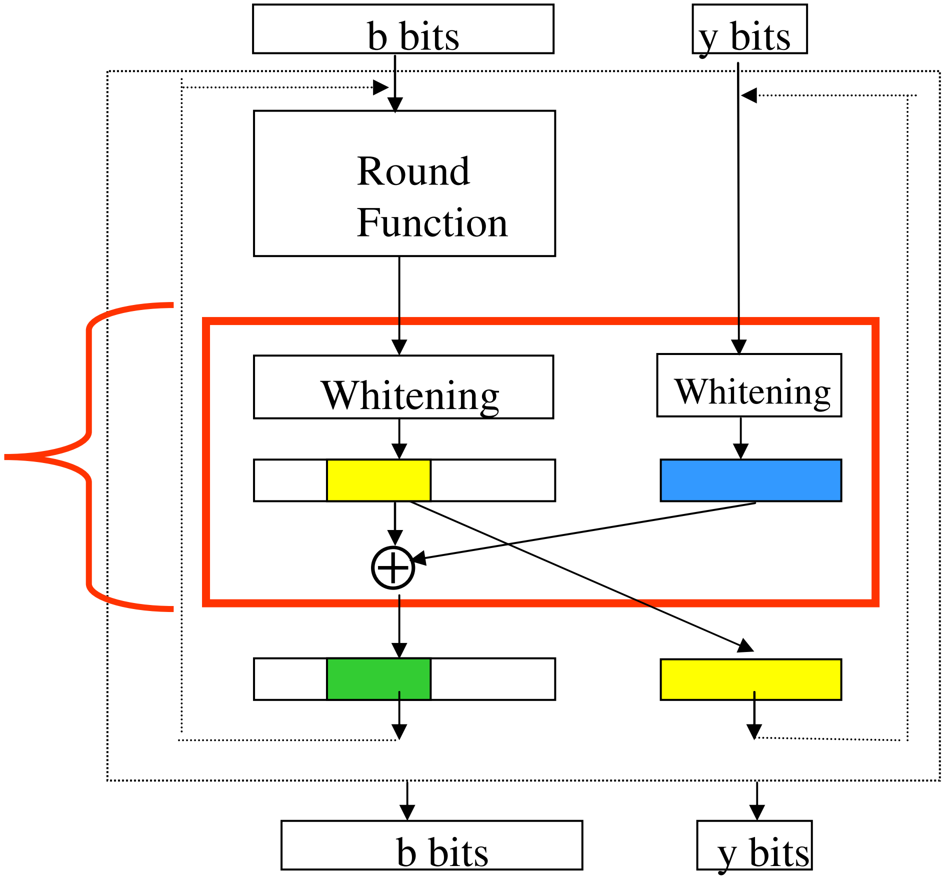
Method 1


b bits

Round Function

Whitening

Round of *G*


b bits

y bits

Round Function

Whitening

Whitening

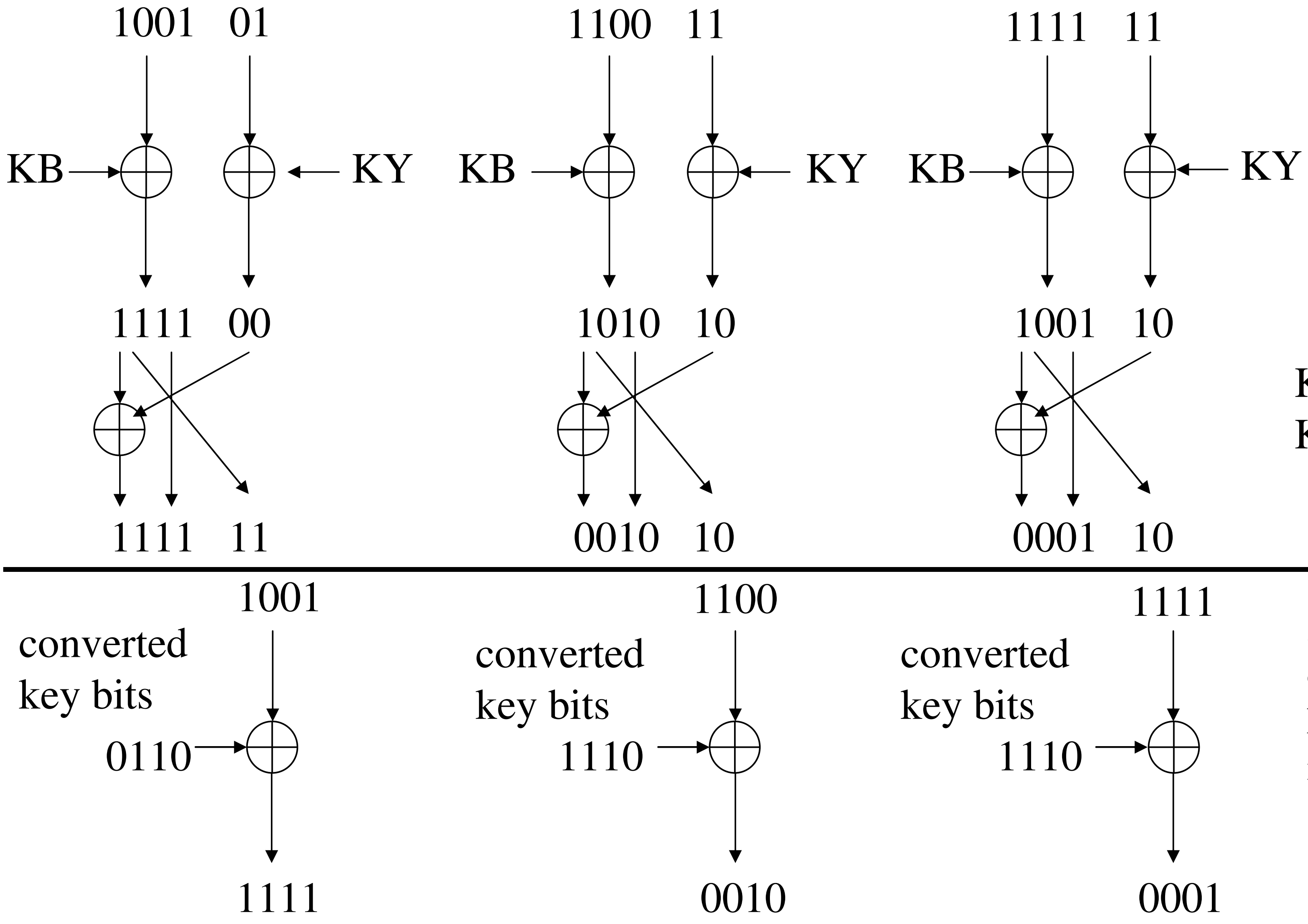
$\oplus$

b bits

y bits

Round of *G′*

Conversion of Whitening Bits from *G'* to *G*



KB = 0110
KY = 01

2 of 3 converted keys are identical

Method 1

- n plaintext, ciphertext pairs $\{(P_j,C_j)\}$ for *G'* (formed from $\{(P_j,C_j)\}$ for *G* with y bits appended)
- Initial whitening and round 1: discard rightmost y whitening bits from *G'*, use remaining round key bits in *G*
- For i= 2 to r-1:
  - Convert round key from *G'* to round key for *G* for each of n plaintexts: collapse swap and whitening step, copy any key bits internal to the round function
  - At most 2^y possible whitening values result for *G*
  - Use result occurring the max # of times as the whitening for *G*
  - Use any key bits from *G'* internal to the round function in *G*
  - $n_{i+1} \geq n_i/2^y$ (P,C) pairs left
- Round r: discard rightmost y whitening bits from *G'*, use remaining round key bits in *G*

Method 1

- Worst case:
  - Apply attack on *G’* once
  - Conversion of round key bits is linear
  - Find keys for set of $\geq n/2^{y(r-2)}$ (P,C) pairs
  - Requires $y(r-2) < b$

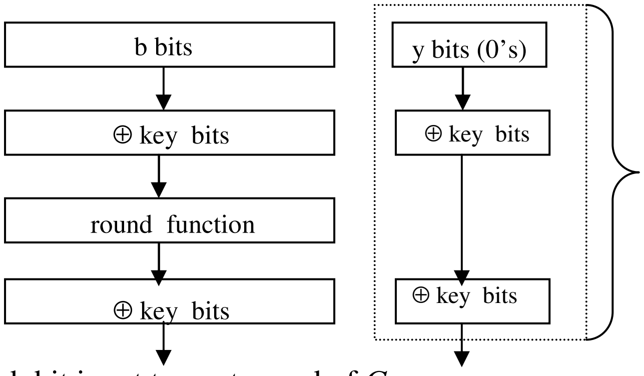
Method 2

- Let $\{(P_j,C_j)\}$ denote a set of n plaintext, ciphertext pairs for *G*
- Round key conversion from *G'* to *G*:
  - Key bits used internal to round function copied
  - Rightmost y bits of whitening deleted, rest copied
- Form (P_j',C_j') pairs for *G'* :
  - $P_j' = P_j$ || constant set of y bits (*e.g.* all 0's)
  - $C_j' = C_j$ || variable rightmost y bits
- Repeated reduced-round attacks on *G'* :
  - Find round keys for *G* one round at a time

Method 2

- Use attack on *G'* to find round keys 0, 1 of *G*:
  - Solve for round keys 0,1 of *G'*
  - Convert to round keys for *G* (*i.e.* copy and discard rightmost y bits in whitening steps)

Rounds 0 and 1 of *G*'
prior to first swap.



b bits

⊕ key bits

round function

⊕ key bits

y bits (0's)

⊕ key bits

⊕ key bits

y bits have no impact
on leftmost b bits
prior to swap

b bit input to next round of *G*

- For i = 2 to r:
  - Obtain output of round i-1 of *G*, append constant bits to form input to r-i+1 round version of *G'*
  - Solve reduced round version of *G'* for 1st round key
  - Convert round key from *G'* to ith round key for *G*

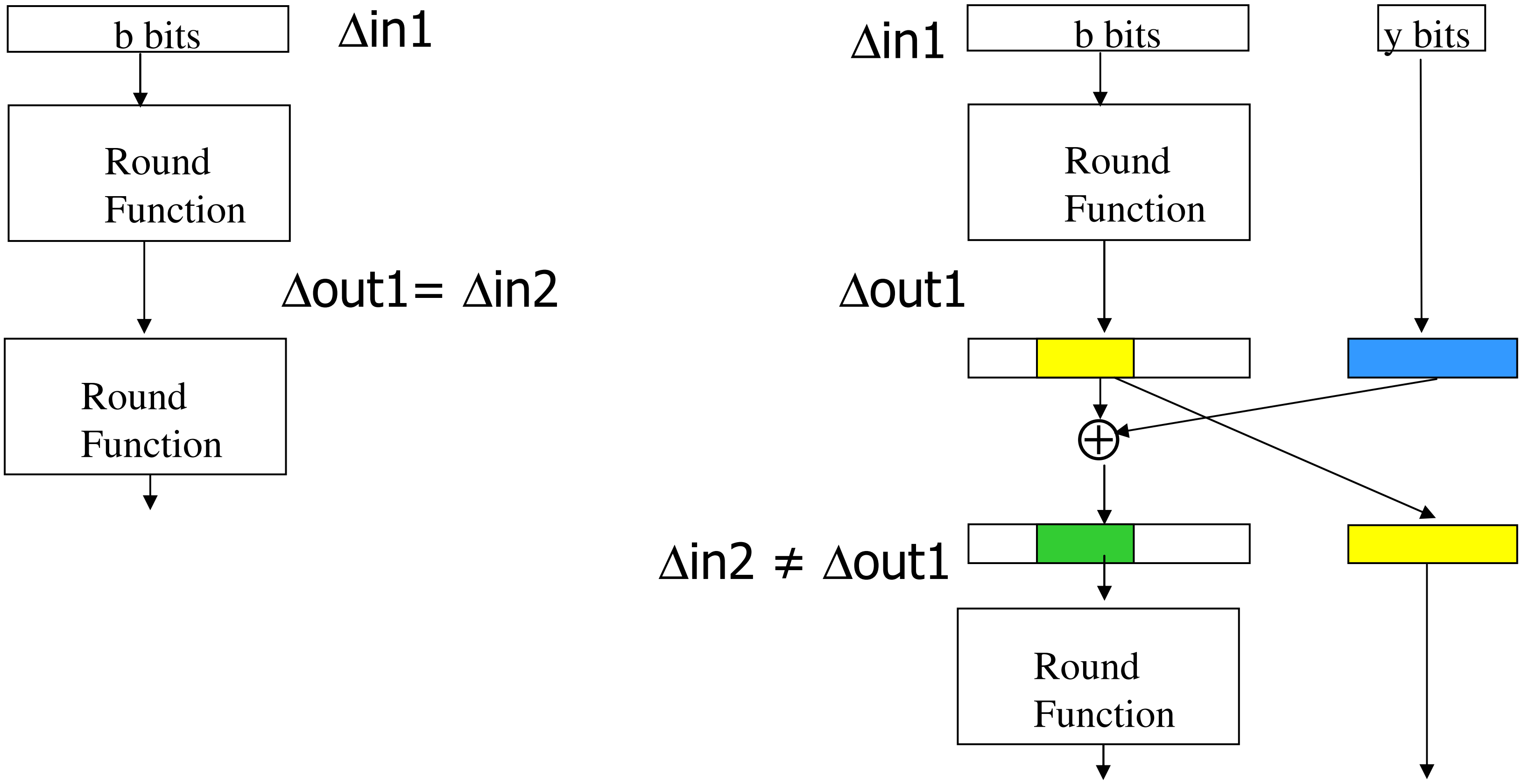
Method 2

- Worst case:
  - n applications of *G*
  - r attacks on reduced round versions of *G'* requiring a maximum of n(r+1)r/2 rounds of *G'*

Algebraic Attacks

- Algebraic-based cryptanalysis
  - Linear and non-linear cryptanalysis
  - Equations relating P,C,K bits
- Can convert equations (attack) for *G'* to equations for *G*
  - Steps between rounds are linear
  - Independent of specific cipher
  - Polynomial time and memory

Differential Cryptanalysis



View Δ as Δb || Δy
Determine possible Δs resulting from swap

Differential Cryptanalysis

- Defined method for tracking difference based on b-bit and y-bit portions of the block
  - Is $\Delta b = 0$? Is $\Delta y = 0$?
- Used differential bounds of round function from *G*
- Computer model to generate cases for elastic version of AES
  - Show probability *r'* round characteristic holds is $\leq \mathbf{2^{-(b+y)}}$
  - Bounded 1 to 8 rounds for each case to derive overall bound

Summary

- Provides a method for creating elastic block ciphers from existing block ciphers
  - Computational workload proportional to block size
- Approach is between black-box and design from scratch
- Relate the security of elastic version to original version - avoid analyzing from scratch
- Other results
  - Creation of variable length PRPs and SPRPs from fixed-length PRPs
  - Modes of encryption