Measuring the Cryptographic Security of S-Boxes with Efficient Implementations

Christopher A. Wood

April 8, 2013

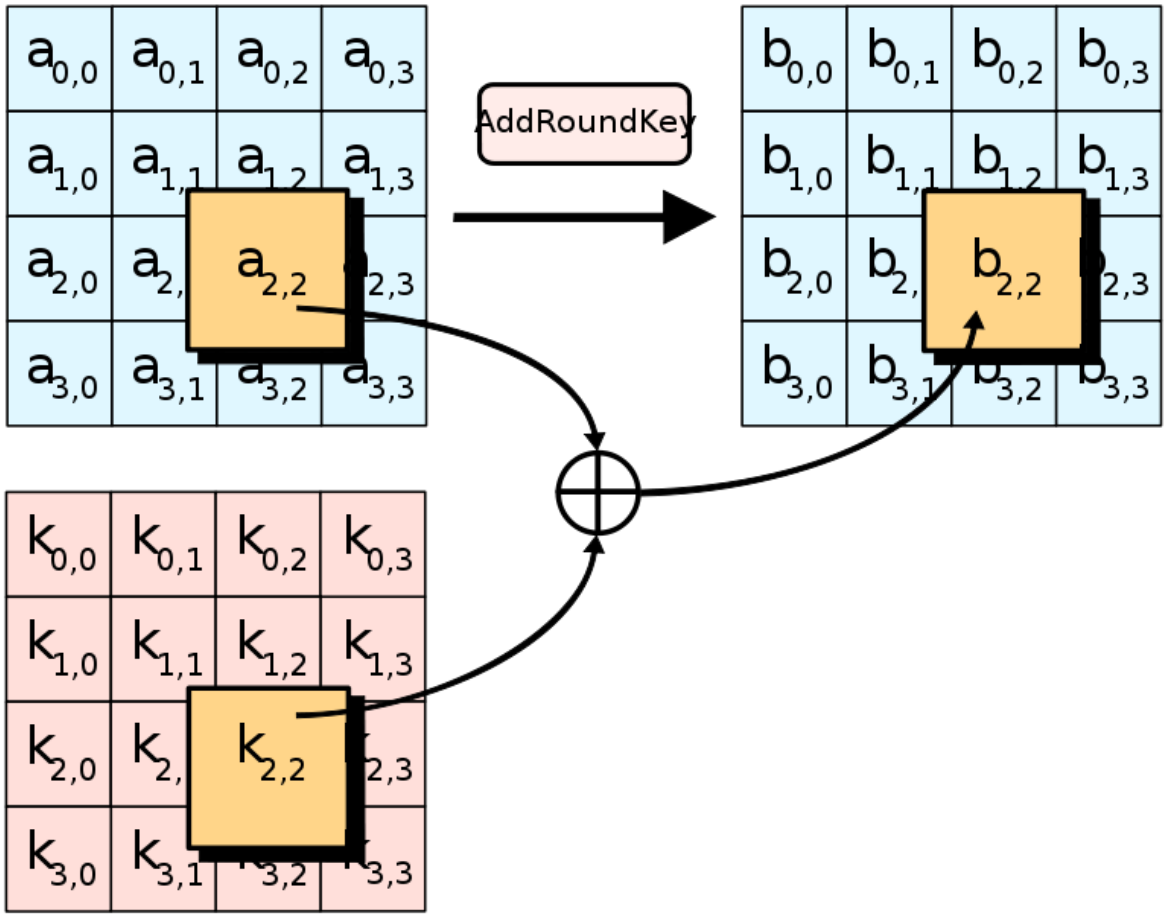
Agenda

1. Rijndael Review
2. Design Criteria and Implementation Considerations
3. Measuring S-box Security
   - Some Other Nonlinear Transformations
4. Implementation Strategies
   - Composite Fields
   - Composite Fields for Rijndael
   - Tower Field Isomorphic Functions
5. 16-bit S-boxes

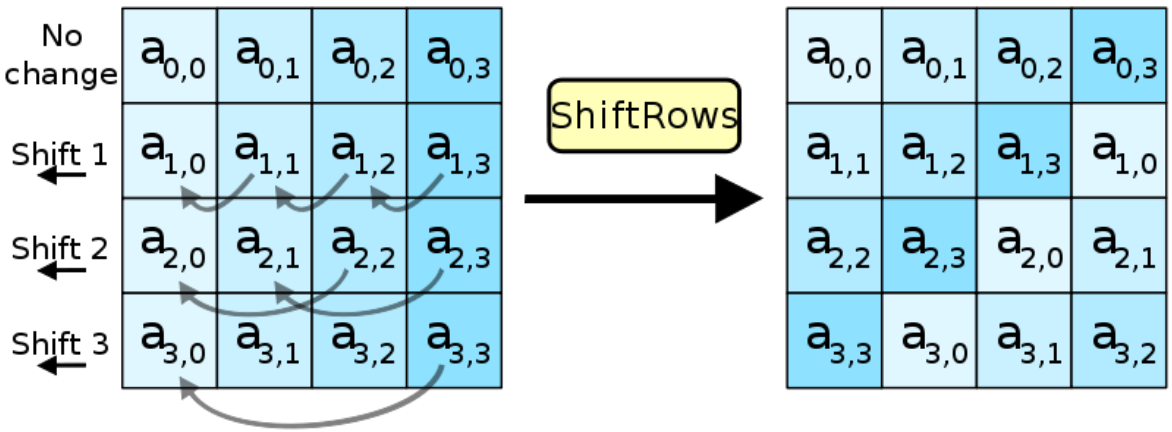
Rijndael - The Advanced Encryption Standard

- Four main operations
  - Add round key
  - Shift rows
  - Substitute bytes
  - Mix columns

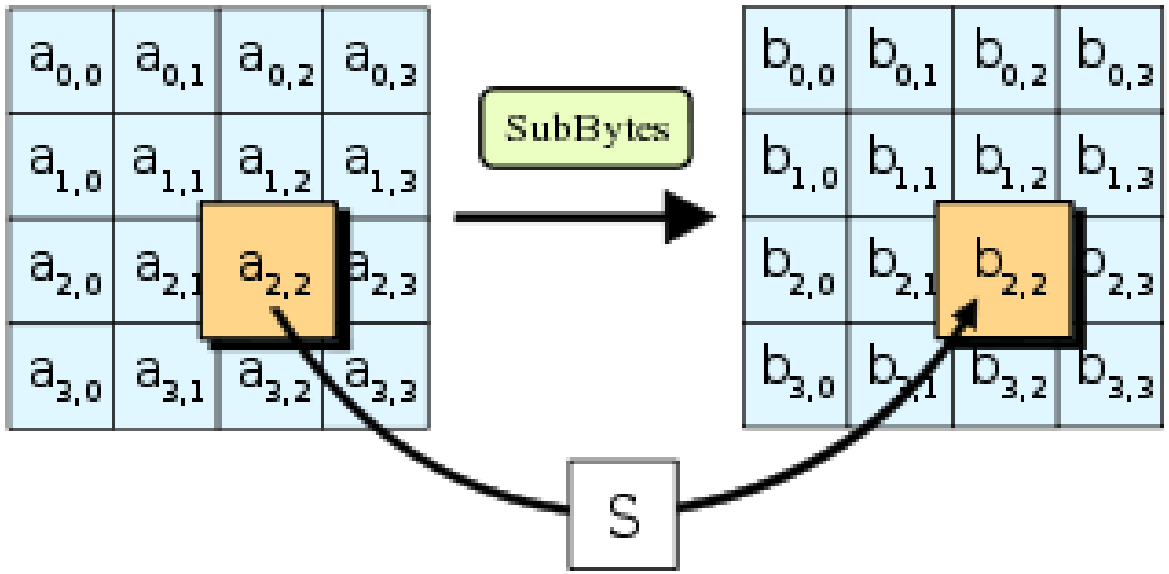
Add Round Key



Shift Rows


No change
Shift 1
Shift 2
Shift 3
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
ShiftRows
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$ $a_{1,0}$
$a_{2,2}$ $a_{2,3}$ $a_{2,0}$ $a_{2,1}$
$a_{3,3}$ $a_{3,0}$ $a_{3,1}$ $a_{3,2}$

Substitute Bytes



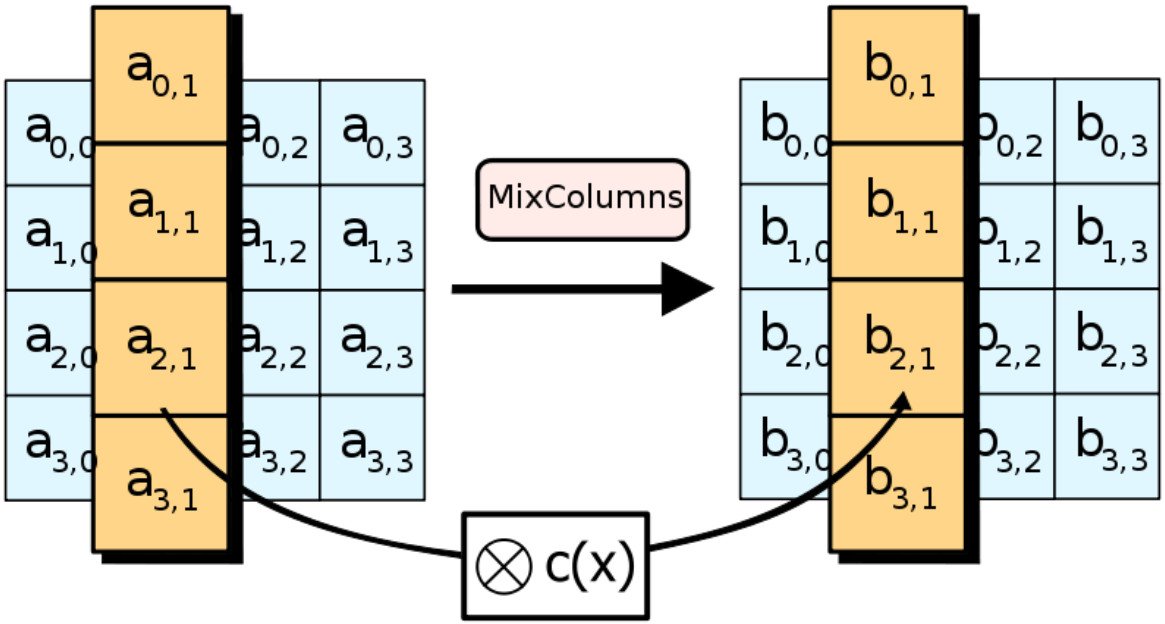
Substitute Bytes

This affine transformation can also be represented algebraically as follows

$$b'_i = b_i \oplus b_{(i+4) mod 8} \oplus b_{(i+5) mod 8} \oplus b_{(i+6) mod 8} \oplus b_{(i+7) mod 8} \oplus c_i$$

where i is the ith bit of the input byte b and $c = \langle 01100011 \rangle$.

Mix Columns



Mix Columns MDS Matrix

$$\begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Algorithm security fundamentals

- Strive for high confusion and diffusion
  - *Confusion* - complex relationship between the secret key and ciphertext
  - *Diffusion* - dissipation of plaintext bits throughout ciphertext bits

Implementation Considerations

- Design against common cryptanalysis techniques
- Linear transformations
  - Linear permutations
  - Circular shifts
  - Modular addition
- Nonlinear transformations
  - **S-boxes**
  - ARX functions
  - Chaotic recurrence relations

S-boxes

- Bijective functions from $\mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$.
- Designed for optimal nonlinearity and algebraic complexity.
  - Maximize the avalanche property of the S-box for all input/output pairs.
  - Minimize the differential propagation probability
  - Maximize complexity of the algebraic expression for the S-box in $\mathbb{F}_2^n$

S-boxes

- Random and fixed structure (i.e. Rijndael s-box) designs have been proposed based on susceptibility to differential cryptanalysis
  - Fixed structure are more beneficial for security analysis and proof purposes
- Various construction criteria have been proposed
  - Nydberg (91) - *"A perfect nonlinear S-box is a substitution transformation with evenly distributed directional derivatives."*
  - Dawson and Tavares (91) - static and dynamic criteria supporting claim for high branch numbers and avalanche property
  - ...

Measuring security

- Linear behavior
  - Exhibit avalanche effect and adherence to Strict Avalanche Criterion (SAC)
- Nonlinear behavior
  - Branch number
  - Direct measurement of nonlinear behavior

Avalanche effect

A function $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ exhibits the *avalanche effect* if and only if

$$\sum_{x \in \mathbb{F}_2^n} \text{wt}(f(x) \oplus f(x \oplus c_i^n)) = n2^{n-1}, *$$

for all $i(1 \leq i \leq n)$, where $c_i^n = [0, 0, \ldots, 1, \ldots, 0]$ (where a 1 is in the nth position of the vector of cardinality n).

*wt indicates the Hamming Weight function

SAC

A function $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ satisfies the *Strict Avalanche Critertion (SAC)* if for all $i(1 \leq i \leq n)$ the following equation holds:

$$\sum_{x \in \mathbb{F}_2^n} f(x) \oplus f(x \oplus c_i^n) = (2^{n-1}, 2^{n-1}, ..., 2^{n-1})$$

This simply means that $f(x) \oplus f(x \oplus c_i^n)$ is balanced for every pair of elements in $\mathbb{F}_2^n$ with Hamming distance of 1.

S-box specific nonlinear measurements

The nonlinearity of an n-bit S-Box from $\mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ can be measured by

$$P_S = \max_{0 \neq a,b} |\{x \in \mathbb{F}_2^n : S(x+a) - S(x) = b\}|$$

where $a, b \in \mathbb{F}_2^n$.

Bent Functions

The correlation between a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ and a linear function $x \mapsto u \cdot x$ is defined as

$$c_f(u) = \frac{1}{2^n}(|\{x \in \mathbb{F}_2^n : f(x) = u \cdot x\}| - |\{x \in \mathbb{F}_2^n : f(x) \neq u \cdot x\}|)$$

A Boolean function is thus called *bent* if

$$|c_f(u)| = 2^{\frac{-n}{2}},$$

for all $u \in \mathbb{F}_2^n$. Note that n must be even in order for f to be bent.

Vector Bent Functions

- Perfect nonlinearity of Boolean functions strongly correlates to cryptographic strength
- Nydberg's "perfect nonlinear functions" - vectorial Boolean functions

A vector function $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is said to be *bent* if

- $w \cdot f_i$ is bent for all $w \neq 0, 1 \leq i \leq m$.
- f is perfectly nonlinear ($f(x+\alpha) = f(x)$ is uniformly distributed as x varies, for all fixed $\alpha \in \mathbb{F}_2^n - \{0\}$).

APN S-boxes

A function $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ is said to be *almost perfect nonlinear* (APN) if

$$|\{x : f(x+\alpha) + f(x) = \beta\}| \leq 2,$$

for all fixed $\alpha \in \mathbb{F}_2^n - \{0\}$. Some examples include:

- $f : \mathbb{F}_2^n \to \mathbb{F}_2^n, f(x) = x^3$
- $f : \mathbb{F}_2^n \to \mathbb{F}_2^n, f(x) = x^{2^k+1}$ (i.e. any odd power exponent)

Differentially δ-Uniform S-box functions

A function $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ is said to be *differentially δ-uniform* if

$$|\{x : f(x+\alpha) + f(x) = \beta\}| \leq \delta,$$

Small values for δ are desirable - indicates higher degree of nonlinearity.

ARX functions

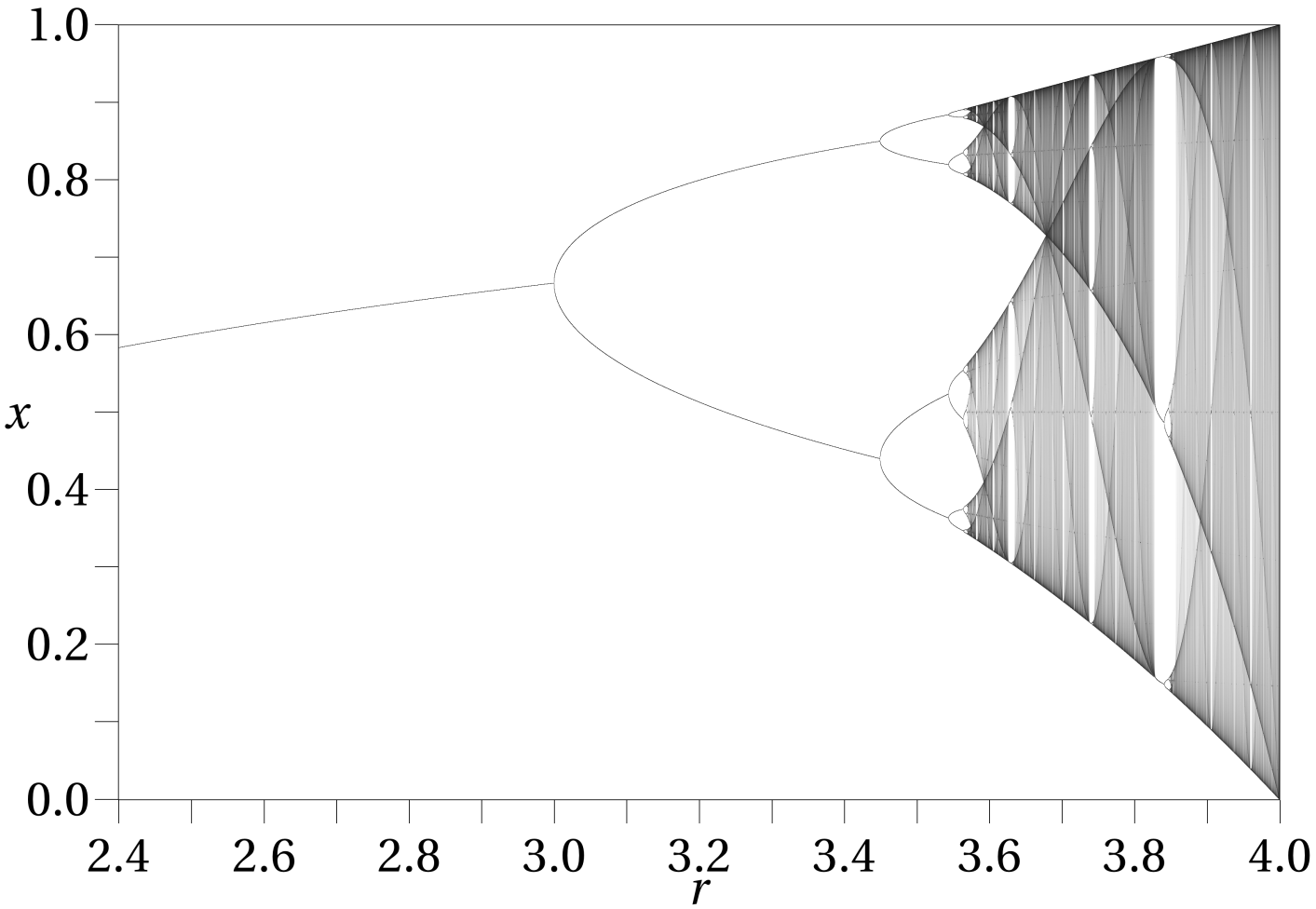
- Nonlinear functions consisting of a combination of addition, bitwise rotation, and bitwise XOR operations
- Analysis of differential propagation is difficult
  - Differential properties of sub-operations need to be considered ($\text{adp}^{\oplus}$, xdp^{+})
- Most susceptible to rotational cryptanalysis
  - Threefish was attacked using a combination of rotational cryptanalysis with a rebound attack (Khovratovich et al, 2010) - led to adjustment of Threefish rotation constants

Chaotic recurrence relations

- Chaotic systems are defined by:
  - Sensitivity to initial conditions
  - Topologically mixing (i.e. covers entire state space)
  - Dense (and long) periodic orbits
- Some recurrence relations exhibit "chaotic" behavior (e.g. the Logistic Map)

$$x_{n+1} = rx_n(1 - (x_n))$$

Chaos in the Logistic Map



Changing Gears

Now that we've measure the security and chosen the right S-box, we must now implement it!

Composite Fields

A *composite field* is a pair

$$\{GF(2^n), Q(y) = y^n + \sum_{i=0}^{n-1} q_i y^i, q_i \in GF(2)\}$$

$$\{GF((2^n)^m), P(x) = x^m + \sum_{i=0}^{m-1} p_i x^i, p_i \in GF(2^n)\},$$

where $GF(2^n)$ is constructed from $GF(2)$ by $Q(y)$, and $GF((2^n)^m)$ is constructed from $GF(2^n)$ by $P(x)$. Also, $GF((2^n)^m)$ is a degree m extension of $GF(2^n)$.

Rijndael S-box

The S-box in Rijndael is defined as follows:

$$F(x) = Ax^{-1} \oplus b,$$

where $x, b \in GF(2^8)$.

Computing the inverse of x with combinational logic in hardware is *very* expensive. What can we do?

Computing the Inverse with Composite Fields

Every element in a field $GF(2^{nm})$ can be represented by a polynomial with coefficients from $GF(2^n)$ using an irreducible polynomial of the form $x^2 + Ax + B$ (we assume $m = 2$). Thus, if $\alpha \in GF(2^{nm})$, and $\alpha = bx + c$, where $b, c \in GF(2^n)$, then:

$$\alpha^{-1} = (bx + c)^{-1} = b(b^2B + bcA + c^2)^{-1} + (c + bA)(b^2B + bcA + c^2)^{-1}$$

Now we compute the inverse over $GF(2^n)$, leading to less required hardware resources.

Composite Field Representations

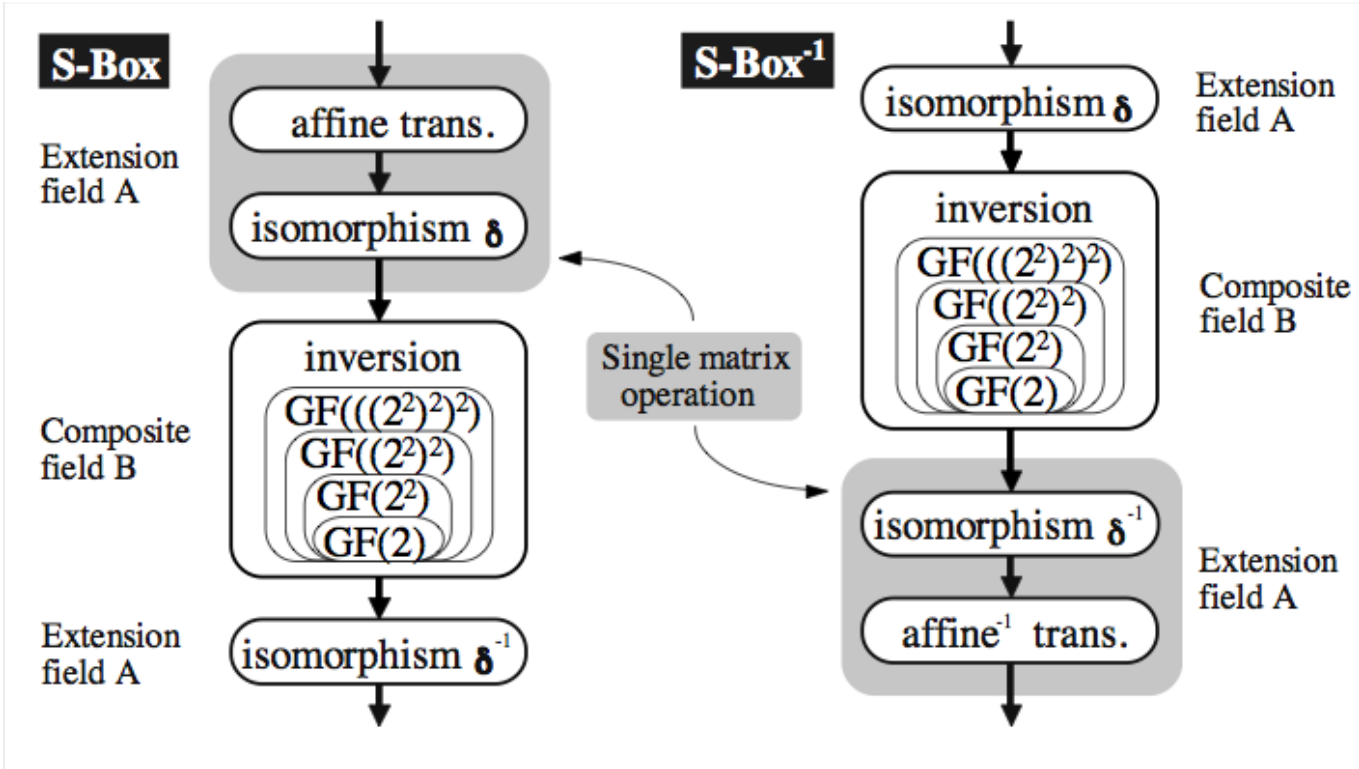
Research has systematically examined all composite field extensions for $GF(2^8)$, seeking to minimize the area.

$$GF(2^4) \rightarrow GF((2^4)^2)$$
$$GF(2^2) \rightarrow GF((2^2)^4)$$
$$GF(2^2) \rightarrow -GF((2^2)^2) \rightarrow GF(((2^2)^2)^2)$$

A systematic evaluation must check all tower field extensions **and** all irreducible polynomials.

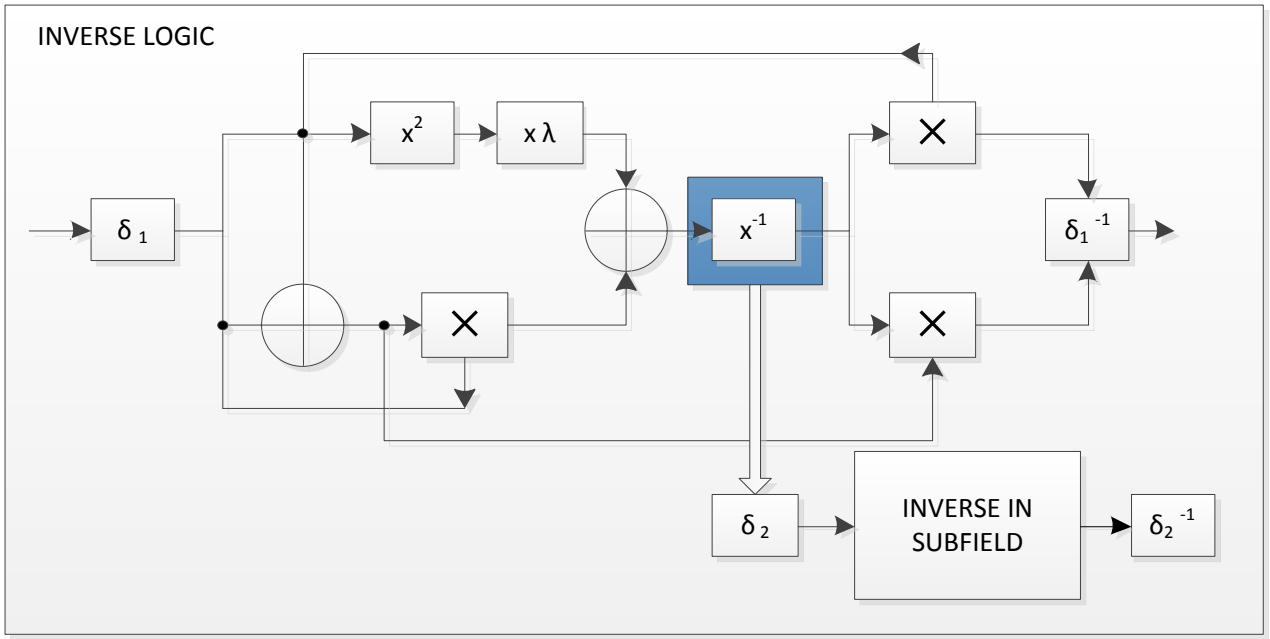
8-Bit S-Boxes

Satoh tower field design - $GF(((2^2)^2)^2)$



16-Bit S-Boxes

Proposed design - $GF((((2^2)^2)^2)^2)$



Question: How deep should this recursion go? What yields the minimal hardware area?

Tower Field Isomorphic Functions

The isomorphic functions are constructed as follows (assume we're constructing a function for mapping $GF(2^{nm}) \rightarrow GF((2^n)^m)$):

- Find two generators α and β ($\alpha \in GF(2^{nm})$ and $\beta \in GF((2^n)^m)$), where α and β are roots of the same primitive irreducible polynomial.
- Map $\alpha^k \rightarrow \beta^k$ for $1 \leq k \leq 2^{nm}$ (mapping basis elements of $GF(2^{nm}) \rightarrow GF((2^n)^m)$). If the mapping doesn't hold group homomorphism, find the next generator β and repeat.

Isomorphic Function Generation - Primitive Irreducible Polynomials

The isomorphic mappings are constructed as follows (assume we're constructing a function for mapping $GF(2^{nm}) \rightarrow GF((2^n)^m)$):

- Find two generators α and β ($\alpha \in GF(2^{nm})$ and $\beta \in GF((2^n)^m)$), where α and β are roots of two respective primitive polynomials.
- Map α^k to β^k for $1 \leq k \leq 2^{nm}$.
- Multiplication and addition homomorphism is guaranteed.
  - $\alpha^i \times \alpha^j = \alpha^{i+j} = \beta^{i+j} = \beta^i \times \beta^j$
  - $\alpha^t = \alpha^i + \alpha^j \rightarrow \beta^t = \beta^j + \beta^j$

Isomorphic Function Generation - Exhaustive Searches

- Find two generators α and β ($\alpha \in GF(2^{nm})$ and $\beta \in GF((2^n)^m)$).
- Map α^k to β^k for $1 \leq k \leq 2^{nm}$ (multiplication homomorphism holds)
- For all $0 \leq i \leq 2^{nm} - 1$ check to see if $\alpha^i + 1 \rightarrow \beta^i + 1$.
- Multiplication and addition homomorphism is now guaranteed.
  - $\alpha^i \times \alpha^j = \alpha^{i+j} = \beta^{i+j} = \beta^i \times \beta^j$
  - $\alpha^t = \alpha^i + \alpha^j = \alpha^i \times (1 + \alpha^{j-i}) \rightarrow \beta^t = \beta^i \times (1 + \beta^{j-i}) = \beta^i + \beta^j$

Matrix Transformation **T**

The matrix **T** can be generated with the following algorithm.

- Let β be a generator of $GF((2^n)^m)$ such that $\alpha^i \in GF(2^{nm})$ is mapped to β^i for all $0 \leq i \leq 2^{nm} - 1$ (α forms a basis of $GF((2^n)^m)$).
- Compute $\alpha^0, \alpha^1, \alpha^2, \ldots, \alpha^{nm-1}$.
- Define the columns of **T** as the transpose of each nm-dimensional bit vector of these powers:

$$\mathbf{T} = \begin{bmatrix} (\alpha^{nm-1})^T & \ldots & (\alpha^1)^T (\alpha^0)^T \end{bmatrix}$$

An Example

- $\alpha = x$ and $\beta = xy$
- $(x^7 + x^6 + x^5 + x^2 + x + 1) \rightarrow [(x^3 + x^2 + x + 1)y + (x^3 + x^2 + 1)]$
- $\mathbf{T} = \begin{bmatrix} (xy^{nm-1})^T & \dots & (xy^1)^T (xy^0)^T \end{bmatrix}$

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

Another Example

- $\alpha = x$ and $\beta = xy$ (same homomorphic mapping)
- $(x^6) \rightarrow [(x^2)y + (x^3 + x^2 + 1)]$
- $\mathbf{T} = \begin{bmatrix} (xy^{nm-1})^T & \dots & (xy^1)^T (xy^0)^T \end{bmatrix}$

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

Different Tower Field Decompositions for 16-bit S-boxes

$$(1)\ GF(2^{16}) \rightarrow GF((2^8)^2)$$
$$(2)\ GF(2^{16}) \rightarrow GF((2^4)^4)$$
$$(3)\ GF(2^{16}) \rightarrow GF((2^2)^8)$$

We need only study these decompositions - optimal tower-field decompositions for smaller fields are in the literature.

Multiplicative Inverse Calculations

The derivation gets messy very quick...

$$(b*x^3+c*x^2+d*x+e)*(f*x^3+g*x^2+h*x+i)= \\ k*(x^4+A*x^3+B*x^2+C*x+D)+1$$

$$f \rightarrow \frac{1}{x^3\left(e+dx+cx^2+bx^3\right)}\left(1-ei+Dk-ehx-dix+Ckx\right) \\ \left(-egx^2-dhx^2-cix^2+Bkx^2-dgx^3-chx^3-bix^3+Akx^3-cgx^4\right) \\ \left(-bhx^4+kx^4-bgx^5\right)$$

Are there easier ways to calculate the inverse?

Finding Capable Polynomials

- Primitive polynomials always work for the mapping
  - Let α be a primitive root of the field $F_2[x]/P(x)$ and $P(\alpha) = 0$
  - $P(x)$ is therefore a *primitive polynomial*
  - Powers α (which are linearly independent) can be used to form a standard basis
- Exhaustive generate all primitive polynomials up to degree 16 (there exists $a_p(n) = \frac{\phi(p^n-1)}{n}$ polynomials for degree n)
- Choose the one that has the lowest transformation and multiplication/inverse costs

Finding Capable Polynomials (cont'd)

Degree	Primitive Polynomials
1	$x+1$
2	x^2+x+1
3	x^3+x+1, x^3+x^2+1
4	x^4+x^3+1, x^4+x+1
5	x^5+x^2+1, $x^5+x^3+x^2+x+1$, $x^5+x^4+x^3+x+1$

For $n = 16$, we have $a_2(16) = \frac{\phi(2^16-1)}{n} = 2048$ different primitive polynomials.

Choosing the Right Transform

- Let $\mathbf{T}^*$ be the optimal transformation matrix in the set of transformations $\mathscr{T}$.
- The "cost" of transforms is the number of 1s in the matrix $\mathbf{T}^*$.
- The "cost" of the inverse is dependent on the polynomial selection.

$$T^* = \min_{T_i \in \mathscr{T}} \{C(transform) + C(inverse) + C(invTransform)\}$$

- Examine all primitive polynomials for $P(x)$, $Q(x)$, $R(x)$
- Generate all possible composite field transformations
- Pick the one with least cost

Exhaustively Searching All S-boxes

- Loop over invertible binary matrices and all constants for affine transformation
- For each valid mapping, measure the cryptographic strength using the Boolean function analysis software
- Pick the one with the best properties

An Interesting Case

- Nyberg's Power Mapping: $F(x) = x^{2^k+1}$
- These functions are 2-differentially uniform with a $\mathcal{N}_l$ equal to precisely $2^{n-1} - 2^{\frac{n-1}{2}}$.
  - That's better than the inverse mapping $F(x) = x^{-1}$
- In a normal basis, this reduces to squaring (which is free) and multiplications
- For hardware, does this yield a more efficient **and** more secure mapping for 16-bit S-boxes?