LEVERAGING TECHNOLOGY - THE JOINT IMPERATIVE
MILCOM 2015

Constructing Large S-boxes with Area Minimized Implementations

Christopher A. Wood, University of California Irvine
Stanisław Radziszowski, Rochester Institute of Technology
Marcin Lukowiak, Rochester Institute of Technology

woodc1@uci.edu, spr@cs.rit.edu, mxleec@rit.edu

October 26, 2015

Agenda

1. Introduction and preliminaries
2. Security properties of the S-Box
3. S-Box constructions
4. Computing the multiplicative inverse in a Galois field
5. Main results

The Importance of the S-Box

Why bother with S-boxes?

- They are typically the only nonlinear component in Substitution-Permutation algorithms
  - Reasonably sized linear systems of equations are easy to solve
  - Nonlinearity provides resistance to many known attacks
- They are used in the DES and AES
- They can be implemented "efficiently" in hardware and software
  - Secure implementations are not a part of this work. We assume precautions are taken to mitigate side-channel attacks.

Why Bother Going Bigger?

Why study 16-bit S-boxes?

- They might be useful in the future
  - Will the size of internal elements in the next block cipher standard bump up to 16-bits?
- To determine if such implementations are feasible or practical
  - LUTs for 16-bit S-boxes are out of the question.
- It may yield new perspectives of S-box constructions and implementation techniques

Galois Fields

- A Galois field $GF(\cdot)$ is just a finite field of p ($GF(p)$) or p^n ($GF(p^n)$) elements, where p is prime
- The field's characteristic is the smallest non-negative integer k such that $\underbrace{a + a + \cdots + a}_{k \text{ times}} = 0$
- We focus on Galois fields with characteristic 2 - $GF(2^n)$ - as elements are easily represented as bit strings in hardware and software
- We say that $GF(2^n)$ is an n degree extension of $GF(2)$ (the subfield)

Composite Galois Fields

Composite fields are simply fields composed of *more than one extension*!

Let $k = n \times m$

$GF((2^n)^m)$ (mod $p(v), q(w)$) is a composite field isomorphic to $GF(2^k)$ (mod $r(x)$) if the following are true:

- $GF(2^n)$ is an n degree extension of $GF(2)$ by an n degree irreducible polynomial $p(v)$ over $GF(2)$
- $GF((2^n)^m)$ is an m degree extension of $GF(2^n)$ by an m degree irreducible polynomial $q(w)$ over $GF(2^n)$
- $GF(2^k)$ is a k degree extension of $GF(2)$ by an k degree irreducible polynomial $r(x)$ over $GF(2)$

Bases of Galois Fields

It is natural to represent an element $\alpha \in GF(p^n)$ as a polynomial of the form

$$\alpha = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + ... + a_2x^2 + a_1x + a_0,$$

where $a_i \in GF(p)$ are the coefficients of the polynomial.

This is the polynomial basis representation.

Bases of Galois Fields (continued)

Let θ be a root of $p(x)$, the irreducible polynomial defining $GF(p^n)$, and let $\alpha \in GF(p^n)$

- Polynomial basis: $[\theta^0, \theta^i, \ldots, \theta^{n-1}]$

$$\alpha = a_{n-1}\theta^{n-1} + a_{n-2}\theta^{n-2} + \cdots + a_1\theta + a_0$$

- Normal basis: $[\theta^{p^0}, \theta^{p^1}, \ldots, \theta^{p^{n-1}}]$

$$\alpha = a_{n-1}\theta^{p^{n-1}} + a_{n-2}\theta^{p^{n-2}} + \cdots + a_{p^1}\theta^{p} + a_0\theta$$

(Note: $a_i \in GF(p)$)

S-Box Constructions

- Question 1: What constitutes a cryptographically strong S-box?
  - One that is not susceptible to known attacks

Common Attacks

- Linear cryptanalysis
- Differential cryptanalysis
- Algebraic attacks
- Interpolation attacks
- ... and more

Linear Cryptanalysis

Linear cryptanalysis exploits a high (or low) probability that some linear combination of input and output bits in the S-box is satisfied

What makes an S-box susceptible to this type of attack?

Low nonlinearity

Differential Cryptanalysis

Differential cryptanalysis exploits output differences that occur with high probability for each round of a cipher

An ideal block cipher will have perfectly uniform differentials

What makes an S-box susceptible to this type of attack?

"High" differential uniformity

Other Attacks

Algebraic attacks (XL and XSL)

- Goal: define the cipher as (large) system of *linearized* equations with key-dependent coefficients to solve
- Relevant metric: algebraic immunity

Interpolation attacks

- Goal: model the cipher as a high-order polynomial with key-dependent coefficients and solve
- Relevant metric: algebraic complexity

S-Box Constructions

- ~~Question 1: What constitutes a cryptographically strong S-box?~~
  - ~~One that is not susceptible to known attacks~~
- Question 2: How do we build cryptographically strong S-boxes?
  - Choose constructions that have *known* and *measurable* security properties

Cryptographically Significant Power Mappings

Important properties of power mappings over $GF(2^n)$:

- Of the form $f(x) = x^d$
- Only bijective if $\gcd\{d, 2^n - 1\} = 1$

Cryptographically Significant Power Mappings (continued)

There are many known cryptographically significant power mappings with ideal nonlinearity and differential uniformity:

- Gold: $2^k + 1$, $\gcd\{k, n\} = 1$ for some $1 \leq k \leq 2^n - 1$
- Kasami: $2^{2k} - 2^k + 1$, $\gcd\{k, n\} = 1$ for some $1 \leq k \leq n/2$
- Dobbertin: $2^{4k+3k+2k+k} - 1$ over $GF(2^n)$ with $n = 5k$
- Niho: $2^m + 2^{m/2} - 1$ over $GF(2^n)$ with $n = 2m + 1$ and m even, $2^m + 2^{(3m+1)/2} - 1$ over $GF(2^n)$ with $n = 2m + 1$ and m odd
- Welch: $2^m + 3$ over $GF(2^n)$ with $n = 2m + 1$
- Inverse: $-1 \equiv 2^n - 2$

Choosing the Right Mapping

What if $n = 16$?

- ~~Gold: $2^k + 1$, $\gcd\{k, n\} = 1$ for some $1 \leq k \leq 2^n - 1$~~
- ~~Kasami: $2^{2k} - 2^k + 1$, $\gcd\{k, n\} = 1$ for some $1 \leq k \leq n/2$~~
- ~~Dobbertin: $2^{4k+3k+2k+k} - 1$ over $GF(2^n)$ with $n = 5k$~~
- ~~Niho: $2^m + 2^{m/2} - 1$ over $GF(2^n)$ with $n = 2m + 1$ and m even, $2^m + 2^{(3m+1)/2}$ over $GF(2^n)$ with $n = 2m + 1$ and m odd~~
- ~~Welch: $2^m + 3$ over $GF(2^n)$ with $n = 2m + 1$~~

The inverse mapping is the only remaining candidate!

Affine Transformation Criteria

- Affine transformations are used to increase the complexity of the algebraic representation of the S-box.
- AES-like S-boxes $S(x)$ have the form

$$S(x) = A(P(x))$$
$$S^{-1}(x) = P^{-1}(A^{-1}(x))$$

  where $P(x)$ is the inverse power mapping and $A(x)$ is the affine transformation
- The maximum algebraic complexity (number of terms in the interpolation polynomial) of AES-like S-boxes over $GF(2^n)$ is $n + 1$

Searching for Affine Transformations

Non-deterministic procedure for finding suitable affine transformations

Input: $GF(2^n), d, k$

1. Generate a random matrix $\mathbf{A}$ over $GF(2)$
2. If $det(\mathbf{A}) \neq 0$ go to step 1.
3. Generate a random element $c \in GF(2^n)$ with weight k
4. Iterate over every element $e \in GF(2^n)$ and compute $y = \mathbf{A}(e^d) + c$ and $y^{-1} = (\mathbf{A}^{-1}(y + c))^{d^{-1}}$. If e yields a fixed point with $\mathbf{A}$ and c, go to step 1. Store y and y^{-1} in maps S and S^{-1}.
5. Perform Lagrangian interpolation to obtain $p(y)$ and $p^{-1}(y)$ for S and S^{-1}.
6. If $\#p(y) > n$ and $\#p^{-1}(y) > n$, return $\mathbf{A}$ and c, else go to step 1.

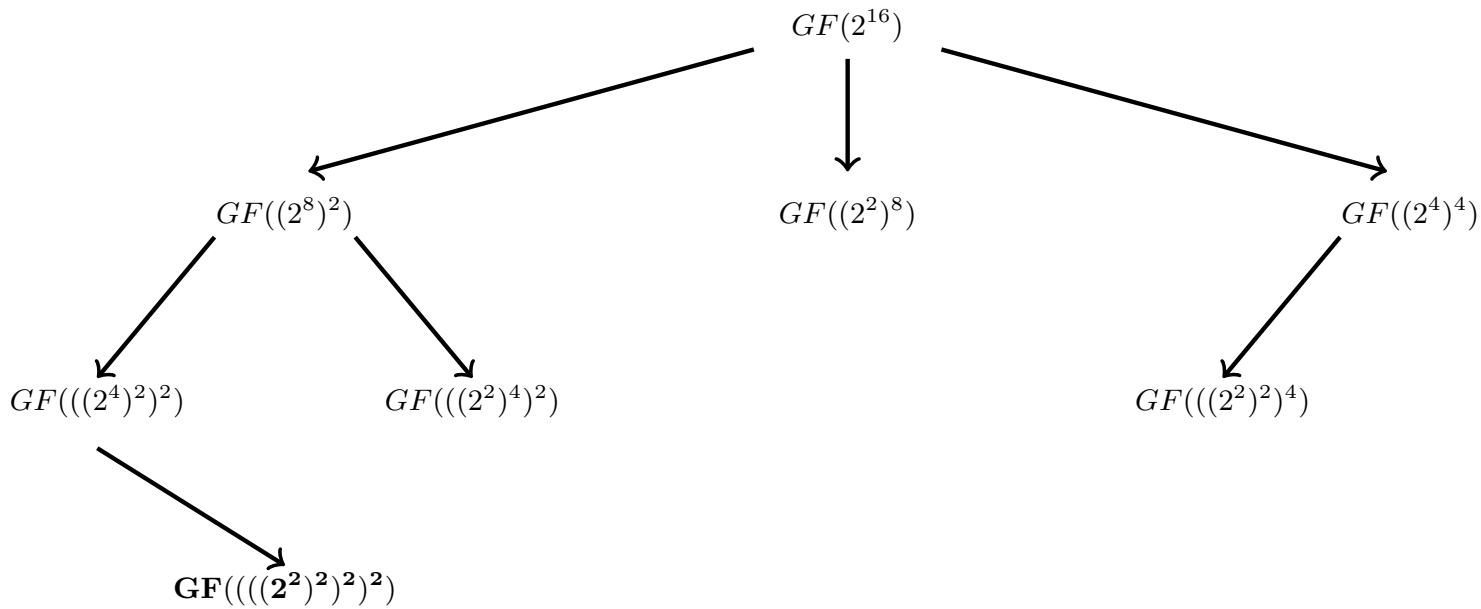
Efficient S-Box Computations

- ~~Question 1: What constitutes a cryptographically strong S-box?~~
  - ~~One that is not susceptible to known attacks~~
- ~~Question 2: How do we build cryptographically strong S-boxes?~~
  - ~~Choose constructions that have *known* and *measurable* security properties~~
- Question 3: How can we implement these S-boxes efficiently in hardware?
  1. Decompose the S-box calculation into bitwise operations
  2. Minimize the logic involved in these operations

Computing the Multiplicative Inverse

- Extended Euclidean algorithm
- Fermat's Little Theorem: $x^{-1} \equiv x^{2^n-2}$ in $GF(2^n)$
- **Reduction to subfield inversion using composite fields**
  - Decomposition to arithmetic over $GF(2)$
  - Itoh-Tsujii inversion algorithm (omitted)

Isomorphic Composite Fields



We focus on $GF((((2^2)^2)^2)^2)$.

Galois Field Arithmetic Decompositions

With composite fields, coefficient arithmetic is performed over the subfield.

How do we go about determining the complexity of all relevant arithmetic operations?

Define arithmetic in the first extension of $GF(2)$ - $GF(2^2)$ - and then walk up the tower to $GF((((2^2)^2)^2)^2)$, constructed as follows:

- $GF(2^2)/p(v) = v^2 + v + 1$
- $GF((2^2)^2)/q(w) = w^2 + w + \Sigma$
- $GF(((2^2)^2)^2)/r(x) = x^2 + x + \Pi$
- $GF((((2^2)^2)^2)^2)/s(y) = y^2 + y + \Lambda$

Inversion in $GF(2^2)$

Inversion in $GF(2)$ is trivial, so we start with the first extension:

$$GF(2^2)/p(v) = v^2 + v + 1$$

For $\delta \in GF(2^2)$ and $\gamma \in GF(2)$. We may compute δ^{-1} as follows:

- Polynomial basis $[1, V]$

$$\delta = \gamma_1 v + \gamma_2$$
$$\delta^{-1} = \gamma_1 v + (\gamma_1 + \gamma_2)$$

- Normal basis $[V, V^2]$ (by Fermat's Little Theorem)

$$\delta = \gamma_1 v^2 + \gamma_2 v$$
$$\delta^{-1} = \gamma_2 v^2 + \gamma_1 v$$

Overall: Polynomial basis requires one XOR and normal basis is "free" (bit swap)

Inversion With More Extensions

Things become more complicated when $GF(2^2)$ is extended to larger fields.

Let $\epsilon(=\delta_1 w+\delta 2)\in GF((2^2)^2)/q(w)=w^2+w+\Sigma$, $\delta\in GF(2^2)$.

- Polynomial basis $[1,W]$

$$\epsilon^{-1}=\delta_1(\delta_2^2+\delta_1\delta_2+\delta_1^2\Sigma)^{-1}w+(\delta_1+\delta_2)(\delta_2^2+\delta_1\delta_2+\delta_1^2\Sigma)^{-1}$$

- Normal basis $[W,W^4]$

$$\epsilon^{-1}=((\delta_1\delta_2+(\delta_1+\delta_2)^2\Sigma)^{-1}\delta_2)w^4+((\delta_1\delta_2+(\delta_1+\delta_2)^2\Sigma)^{-1}\delta_1)w$$

$GF((2^2)^2)$ General Arithmetic Results

Subfield arithmetic costs ((A)dditions, (M)ultiplications, (Sq)uares, (I)nversions, (SS)quare-scales, (Sc)ales) for finite field arithmetic operations in $GF((2^2)^2)$ using polynomial and normal bases.

Operation	*Polynomial Basis*	*Normal Basis*
Inverse	$3M + 2A + I + 1SS$	$3M + 2A + I + 1SS$
Add	$2A$	$2A$
Multiply	$3M + 4A + 1Sc$	$3M + 4A + 1Sc$
Square	$2Sq + Sc + A$	$3A + 2Sq + Sc$

Scaling (and square-scaling) can be further optimized to account for known constants

Counting Gates for $GF((((2^2)^2)^2)^2)$ Inversion

Assume a basis $[1, V]$, $[1, W]$, $[1, X]$, and $[1, Y]$ and coefficients $\Sigma = v$, $\Pi = (v + 1)w + v$, $\Lambda = vwx + (w + v)$

- Inversion: 78 gates
- Multiplication: 81 gates
- Square-scale: 81 gates (none of our optimizations apply)

```
> F2 := GF(2);
> Poly2<V> := PolynomialRing(F2);
> P := V^2 + V + 1;
> F4<v> := ext<F2 | P>;
> Poly4<W> := PolynomialRing(F4);
> Q := W^2 + W + v;
> F16<w> := ext<F4 | Q>;
> Poly16<X> := PolynomialRing(F16);
> R := X^2 + X + ((v + 1)*w + v);
> F256<x> := ext<F16 | R>;
> Poly256<Y> := PolynomialRing(F256);
> S := Y^2 + Y + (v*w*x + (w + v));
> F6K<y> := ext<F256 | S>;
> gatesInv16(P, Q, R, S, v, ((v + 1)*w + v), \
        (v*w*x + (w + v)), 1, v, 1, w, 1, x, 1, y);
398
```

ASIC S-Box Implementations

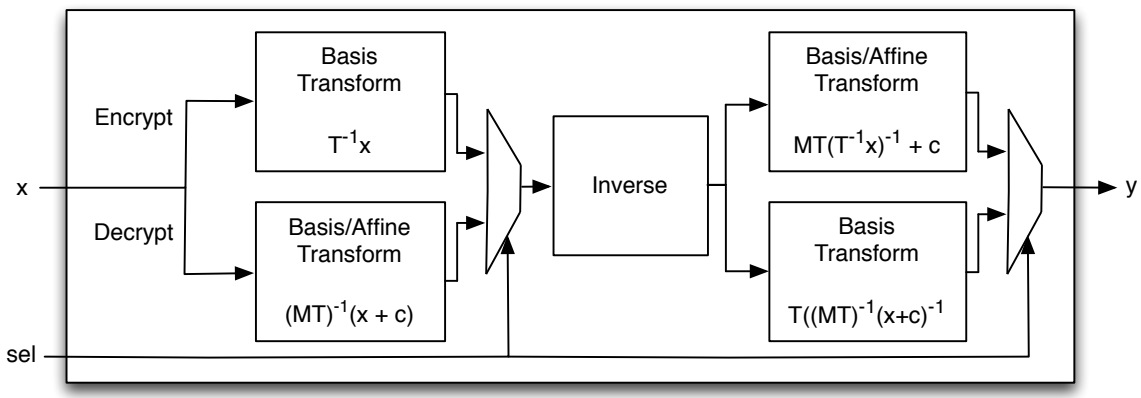
There are really two options for implementing an S-box circuit:

1. Implement the forward and inverse S-boxes separately
   - Redundant inversion circuits
2. Merge the forward and inverse S-box circuits together
   - Share an inverter at the cost of some MUXs

ASIC S-Box Implementations

There are really two options for implementing an S-box circuit:

1. Implement the forward and inverse S-boxes separately
   - Redundant inversion circuits
2. Merge the forward and inverse S-box circuits together
   - Share an inverter at the cost of some MUXs



16-Bit S-Box Results

One of our best candidates has the following parameters:

- Field polynomial: $t(v) = v^{16} + v^5 + v^3 + v + 1$
- Basis sets: $[1, V]$, $[1, W]$, $[1, X]$, and $[Y^{256}, Y]$
- Coefficients: $\Sigma = v$, $\Pi = vw + v$, and $\Lambda = (vw + v)x + w$
- Our (unoptimized) version of this S-box requires 1238 XOR gates and 144 AND gates

16-Bit S-Box Definition

$$S(x) = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_{15} \\ x_{14} \\ x_{13} \\ x_{12} \\ x_{11} \\ x_{10} \\ x_9 \\ x_8 \\ x_7 \\ x_6 \\ x_5 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \\ x_0 \end{pmatrix}^{-1} + \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

16-Bit S-Box Basis Change Matrices

$$\mathbf{T}^{-1} = \begin{pmatrix}
1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1
\end{pmatrix}$$

Conclusion

We talked about

- The motivation for constructing larger S-boxes
- Ways to construct large S-boxes
- Ways to minimize large S-boxes
- (A subset of) our **programmatically**-generated results

Questions?

Fire away!