Design and Cryptanalysis of a Customizable Authenticated Encryption Algorithm

A Master's Thesis Defense

Matthew J. Kelly

Rochester Institute of Technology

mjk7841@rit.edu

Supervised by

Alan Kaminsky, *Primary Advisor*
Marcin Łukowiak, *Primary Advisor*
Michael Kurdziel, *Advisor*
Reza Azarderakhsh, *Advisor*

Stanisław Radziszowski, *Team Member*

August 1, 2014

Overview

Why Authenticate?

- Encryption without authentication is generally insecure
- Several examples in recent history
  - Wired Equivalence Privacy (WEP) in 2001 [5]
  - SSL, IPSEC, and others based on CBC mode in 2002 [14]
- Encryption provides *confidentiality* only
- Authentication is needed for *data integrity* and assurance of message origin
  - Detect tampering or corruption of data
  - Ensure message came from expected sender

Authenticated Encryption

- Provide benefits of encryption and authentication in a single cryptographic primitive
- Process plaintext and produce ciphertext and a Message Authentication Code (MAC)
- AE is easy! Recipe:
  1. One secure block cipher (e.g. AES)
  2. One secure MAC generation function (e.g. HMAC)
  3. Mash them together: Encrypt-then-MAC, MAC-then-Encrypt, or Encrypt-and-MAC
- This naïve approach is called *generic composition*

Against Generic Composition

- Generic composition is far from ideal
  - Two unique keys
  - Not easy to use / not misuse resistant
  - Inefficient
- "Good" AE is more difficult to achieve

Better AE

- Desirable properties of AE algorithms in general:
  - Easy to use, since misuse can result in reduced security
  - Single key
  - Single pass
  - Support for Additional Authenticated Data (AAD / headers)
  - Support for intermediate tags (MACs)
  - No decryption mode requirement
- Government and military have more stringent requirements
  - Algorithms typically not in public domain

History of AE

- Jutla, 2000: Integrity Aware Cipher Block Chaining (IACBC) and Integrity Aware Parallelizable Mode (IAPM) [9]
    - Two keys, no support for AAD, highly patent encumbered
- Rogaway et al., 2001: Offset Codebook Mode (OCB) [13]
    - Requires decryption mode, patent encumbered
- Whiting et al., 2003: Counter with CBC-MAC (CCM) [15]
    - Two passes, only 128-bit block support

History of AE

- Kohno et al., 2004: Carter-Wegman + Counter (CWC) [10]
    - “Two” passes, prime field multiplication
- McGrew and Viega, 2004: Galois/Counter Mode (GCM) [12]
    - “Two” passes, binary GF multiplication, very popular
- Bellare et al., 2004: EAX Mode [1]
    - Two passes, slightly modified generic composition
- Whiting et al., 2005: Phelix [16]
    - Stream cipher based, broken by differential-linear attacks [18]

Present Day AE

- Sponge construction gaining popularity since KECCAK won SHA-3 in 2013
- "Duplexing" the sponge provides **excellent** potential for efficient AE
- ...which is why it has its own section
- CAESAR Competition is ongoing
  - First round out of three right now
  - Seven sponge-based AE algorithms
  - None customizable at an algorithmic level

Contributions

- Secure AE algorithm based on the sponge (duplex) construction
- Highly customizable within our security margin
  - We provide the guidelines
- Single key
- Single pass
- Support for intermediate tags (MACs)
- Support for 128- and 256-bit keys

Overview

Groups

- Set of elements G together with a binary operation $*$
- Satisfies following properties:
  1. *Associativity*. $(a * b) * c = a * (b * c)$ for all $a, b, c \in G$.
  2. *Closure*. $a * b \in G$ for all $a, b \in G$.
  3. *Identity*. There exists an element $e \in G$ such that $a * e = e * a = a$ for all $a \in G$.
  4. *Inverses*. For each $a \in G$ there exists $a^{-1} \in G$ such that $a * a^{-1} = a^{-1} * a = e$.

- For *abelian* groups, $a * b = b * a$ for all $a, b \in G$
- Common example: $(\mathbb{Z}, +)$, the integers under addition

Rings

- Set of elements R together with two binary operations $\cdot$ and $+$
- Call them multiplication and addition
- Satisfies following properties:
  1. R is an abelian group under addition; its identity is called 0.
  2. *Associativity*. Multiplication and addition are both associative.
  3. *Distributivity*. $a(b+c) = ab + ac$ and $(b+c)a = ba + ca$ for all $a, b, c \in R$; multiplication distributes over addition

- R is abelian if multiplication also commutes
- Common example: $(\mathbb{Z}, \cdot, +)$, the integers under addition and multiplication

Fields

- Set of elements $\mathbb{F}$ together with two binary operations $\cdot$ and $+$
- Satisfies following properties:
  1. $\mathbb{F}$ is an abelian ring.
  2. $\mathbb{F}$ is an abelian group under multiplication; its identity is called 1.

Galois Fields

- *Order* of an algebraic structure is the number of elements it contains
- Fields of finite order are called finite fields or *Galois fields* (GFs)
- Well-known result: all GFs are of prime power order
- Denoted $\mathbb{F}_{p^k}$ or $\mathrm{GF}(p^k)$
    - p: *characteristic* of the GF
    - k: *degree* of the GF
- Order of an element a: smallest integer k such that $a^k = e$
- Lagrange: order of an element divides order of the structure
- Cryptographers are mainly concerned with binary GFs ($p = 2$)

GF Element Representations

- Elements in $\mathrm{GF}(p^k)$ can be represented as polynomials modulo $f(x)$
- Where $f(x)$ is irreducible and $\deg(f(x)) = k$, and $\alpha_i \in \mathbb{Z}_p$

$$a = \alpha_{k-1}x^{k-1} + \alpha_{k-2}x^{k-2} + \ldots + \alpha_1 x + \alpha_0$$

- We also use binary (or hex) notation for binary GFs
- Example of some element $a \in \mathrm{GF}(2^{16})$:

$$\begin{aligned} a &= x^{15} + x^3 + x^2 + 1 \\ &\equiv \mathrm{0b1000\_0000\_0000\_1101} \\ &\equiv \mathrm{0x800d} \end{aligned}$$

GF Operations

- Multiplication: multiply polynomials as usual, reduce if degree of result $> \deg(f(x))$
  - Methods to optimize in software and hardware
- Addition: element-wise addition modulo p
  - For binary GF, $a + b \equiv a \,\mathrm{XOR}\, b$, denoted $a \oplus b$

Bitstrings

- Bitstring is a binary string; i.e. string of elements in $\mathbb{Z}_2$
- Example: $1011 \in \mathbb{Z}_2^4$
- Ordinary boolean operations apply: bitwise XOR, AND, etc.

Transformations

- *Transformation*: a function

$$t\colon X \to Y$$

- *Bijection*: one-to-one, onto transformation
- Bijections are *entropy-preserving*
- *Permutation*: bijection where domain X and codomain Y are equivalent
- Permutations on $\mathbb{Z}_2^n$ are central to this work

Confusion and Diffusion

- Shannon's notions of *confusion* and *diffusion* lay foundation for modern symmetric key cryptography
- *Confusion*: obscure relationship between plaintext and ciphertext
  - Example: substitutions
- *Diffusion*: dissipate redundancy of plaintext throughout ciphertext
  - Example: bitwise permutations

Overview

Sponge Construction

- Gaining popularity recently
  - Sponge-based KECCAK hash function won SHA-3 competition
- Provides way to generalize hash functions to have arbitrary length output
- Allows **many** other uses outside of hashing
- Built from underlying iterated *sponge permutation f*

Sponge Construction

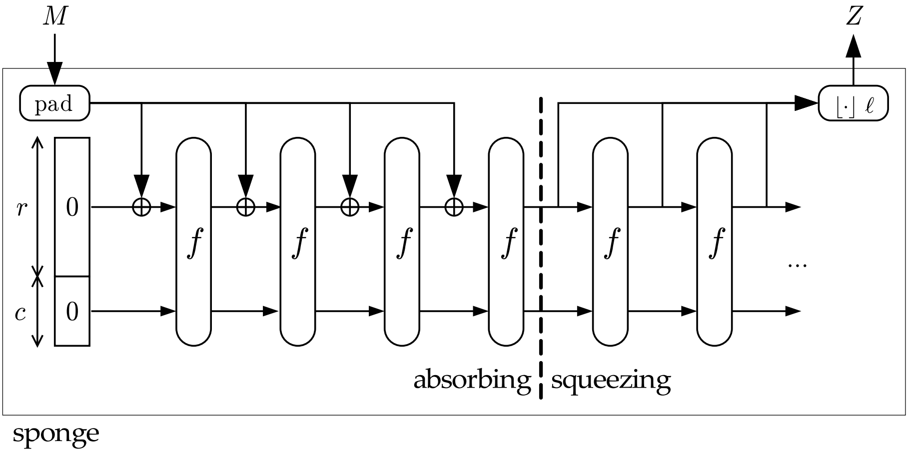

Figure : The sponge construction **sponge**[f, **pad**, r] [3]

Sponge Parameters

- State size / width, $b = r + c$
- Rate, r:
  - Exposed portion of state - *outer state*
  - Absorb and squeeze in r-bit blocks
  - Speed is mostly dependent on r
- Capacity, c:
  - Hidden portion of state - *inner state*
  - Security is mostly dependent on c

Simplified Sponge

- Ignore padding
- May be done at higher level in system
- Focus more / all design effort on underlying permutation

Sponge Applications

1. **Hashing**
   - Absorb plaintext - long
   - Squeeze out message digest - short
2. **Encryption**
   - Absorb key and IV - short
   - Squeeze out keystream - long
3. **MAC Generation**
   - Absorb key then message - long
   - Squeeze out MAC - short
4. **Others**
   - All without changing the overall construction!

Sponge for AE?

- Sponge can do encryption and MAC generation
- ...so can it do authenticated encryption?
- Yes! But:
    - No intermediate tags
    - State not maintained between calls
    - Would require re-initialization between calls
- Modify it slightly to get much more flexibility

Duplex Construction

- Maintains state between calls
- Construct a duplex object D
- Make calls to D.**duplexing**
- Arbitrary length inputs and outputs, including length zero
- *Blank call*: no input provided
- *Mute call*: no output requested

Duplex Illustrated

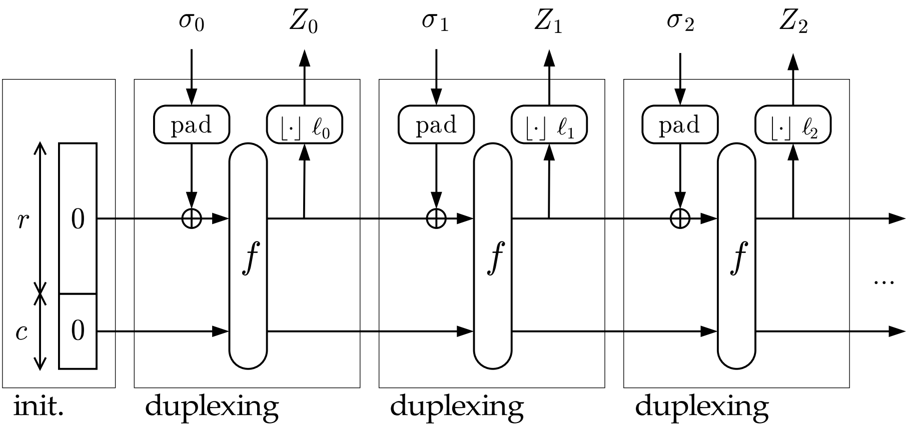

Figure : The duplex construction **duplex**[f, **pad**, r] [3]

Duplex Expanded

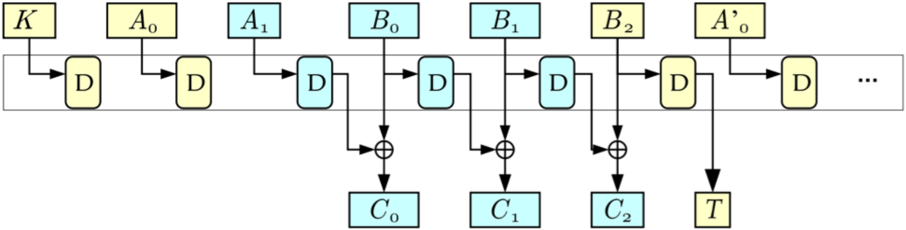

Figure : K = Key, A = Header, B = Body [2]

Duplex for AE

1. Easy to use
2. Single key required
3. Single-pass for encryption and authentication
4. Support for intermediate tags
5. Support for Additional Authenticated Data (AAD, or headers)
6. Secure against generic attacks
7. Ability to trade off speed and security by adjusting r

Generic Security

- Duplex construction can be reduced to sponge construction
- Sponge construction is secure against *generic attacks*
    - Attacks which don't exploit specific properties of f
- If f is secure, so is the sponge construction it lives in

Keyed Sponge Security

- Keyed sponges are more secure than unkeyed [4]
- Lower bound by Jovanovic et. al in 2014 [8]:

$$\min(2^{(r+c)/2}, 2^{c}, 2^{|K|}).$$

- Allows designers to properly choose c for desired generic security

Overview

Algorithm Overview

- Based on the simplified duplex construction with parameters:
  - Width: $b = 512$
  - Rate: $r = 128$
  - Capacity: $c = 384$
- Most design effort into pseudorandom permutation f

Permutation Overview

- Permutation consists of several *rounds*
  - $R = 10$ for 128-bit key
  - $R = 16$ for 256-bit key
- Each round consists of 4 steps
  - Substitution
  - Bitwise Permutation
  - Mixer
  - Add Round Constant
- Each step has a fairly specific purpose

Single Round Illustrated

See thesis document...sorry!

Substitution Step

- *S-box*: random-looking map from n-bit inputs to m-bit outputs
- Main source of *confusion*
- We use 32 identical 16×16 S-boxes
- Could be randomly generated mappings with nice properties
    - Not suitable for hardware
- Instead based on invertible operations in a GF (similar to AES)

S-box

- Taken from Chris Wood's MS thesis (CS Dept., 2013) [17]
- Multiplicative inversion in $\mathrm{GF}(2^{16})$ with irreducible polynomial

$$p(x) = x^{16} + x^5 + x^3 + x + 1$$

- Followed by affine transformation
  - Increase algebraic complexity
- Only 1238 XOR gates and 144 AND gates in hardware

S-box Equation

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

Inverse S-box Equation

$$\left[\begin{pmatrix}
0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
x_{15} \\ x_{14} \oplus 1 \\ x_{13} \\ x_{12} \\ x_{11} \\ x_{10} \oplus 1 \\ x_9 \\ x_8 \oplus 1 \\ x_7 \oplus 1 \\ x_6 \\ x_5 \oplus 1 \\ x_4 \oplus 1 \\ x_3 \\ x_2 \oplus 1 \\ x_1 \oplus 1 \\ x_0 \oplus 1
\end{pmatrix}\right]^{-1}$$

Bitwise Permutation

- Provides *long-range diffusion*
- Aims to increase min number of active S-boxes
- Could be random mapping with nice properties
- We define it using an affine function with nice properties
- Allows for compact representation

$$\pi(x) = 31x + 15 \pmod{512}$$

where $x \in \mathbb{Z}_{512}$ is the bit index

Bitwise Permutation Properties

1. Sends all 16 outputs of each S-box to 16 different mixers
2. Has no fixed points: $\pi(x) \neq x$ for any x
3. High order; does not repeat within R rounds
4. No lower order bits

We found 384 permutations defined by affine functions that satisfy these properties.

PRESENT Attack

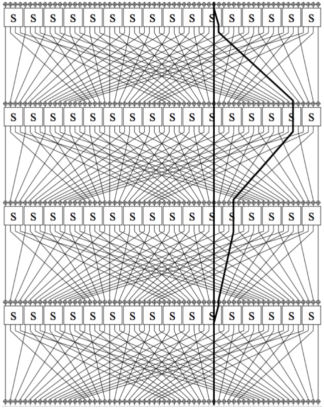

Figure : Minimization of PRESENT active S-boxes

Mixer

- Increase *branch number* of round to 3
  - Verified via SAT solver analysis
  - Tools courtesy of Alan Kaminsky
- Provide local diffusion
- Defined by invertible matrix multiplication in $\mathrm{GF}(2^{16})$
- Irreducible polynomial:

$$p(x) = x^{16} + x^5 + x^3 + x^2 + 1$$

Mixer Equations

- Input two words A and B
- Forward:

$$\begin{pmatrix} A' \\ B' \end{pmatrix} = \begin{pmatrix} 1 & x \\ x & x+1 \end{pmatrix} \begin{pmatrix} A \\ B \end{pmatrix}$$

- Inverse:

$$\begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} a & b \\ b & c \end{pmatrix} \begin{pmatrix} A' \\ B' \end{pmatrix}$$

where

$$a = x^{15} + x^{14} + x^{12} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x + 1$$
$$b = x^{14} + x^{13} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^3 + 1$$
$$c = x^{15} + x^{13} + x^{12} + x^{10} + x^9 + x^7 + x^6 + x^3 + x.$$

Mixer in Hardware

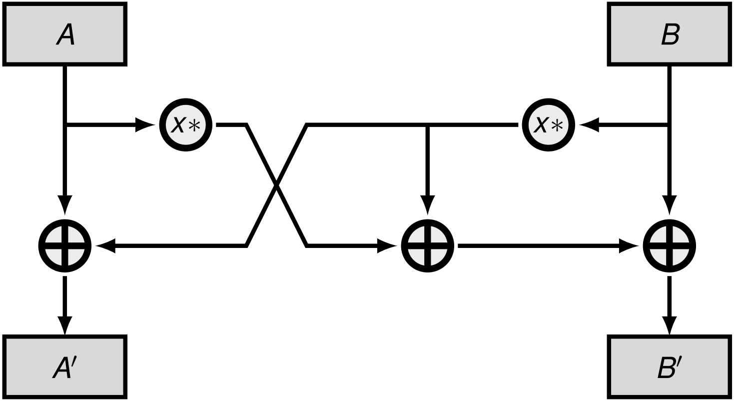

Figure : Hardware implementation of forward mixer function.

$x*$ in Hardware

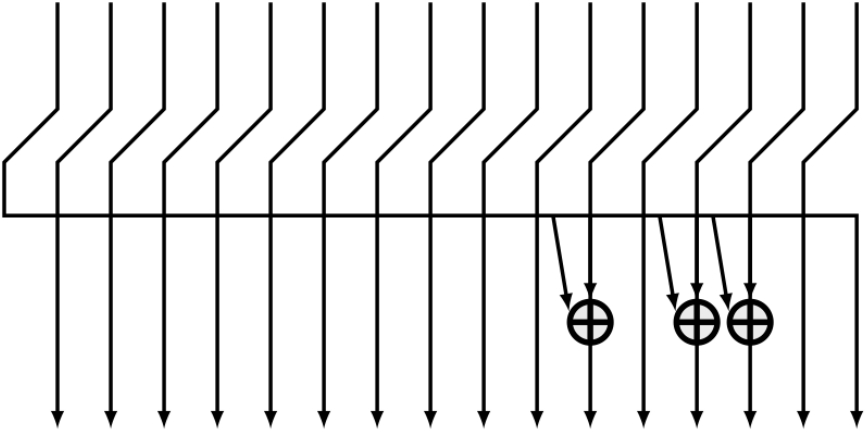

Figure : Hardware implementation of the $x*$ function. The leftmost bit is the MSB.

Add Round Constant

- Very simple; its own inverse
- Bitwise XOR 512-bit round constant into state
- Reduces symmetry in state
- Prevents against *slide attacks*
- RCs generated as follows:

$$RC_i = \textbf{SHA3-512}(\textbf{ASCII}(i))$$

Customization

- Different S-box from Wood's thesis
  - Recalculate maximum linear bias and differential probabilities
- Different bitwise permutation out of the 384 we found
- Different matrix for mixer
  - Verify branch number is 3 using SAT tools
- Different round constants

Overview

Types of Cryptanalysis

- Secure against generic attacks
- Need to analyze permutation
- Two major attack types:
  1. Differential and linear attacks
  2. Algebraic attacks
- In addition: “other” attacks, statistical tests

Goal of Cryptanalysis

- Cryptanalysis goal: find “shortcut” attacks on a system
  - Attacks faster than generic brute force
  - Exploit non-random behavior of system
- Our goal: prove / reason that such attacks are negligibly probable
- Especially generic differential/linear and algebraic attacks
  - Many (most?) attacks are derived from these

Differential Cryptanalysis

- *Difference*: $\Delta X = X' \oplus X''$
- Feed ΔX through system and obtain output ΔY
- *Differential*: $(\Delta X, \Delta Y)$
- Differentials have associated probabilities ($1/2^n$ for ideal system)
- Exploit high (or low) probability differentials
- First look at S-box
- *Active S-box*: S-box being estimated during an attack

Resistance to Differential Attacks

- Our S-box has max differential probability $p_{D,max} = 2^{-14}$
  - Found using Kaminsky's analysis program
- Our round has branch number $\mathcal{B}_D = 3$
  - Minimum of 3 active S-boxes across rounds
- Probability of difference in either output: $p_{D,out} = 2^{-15}$

Resistance to Differential Attacks

- Worst-case probability of propagating difference over 2 rounds

$$(p_{D,max})^{\mathcal{B}_D} \cdot p_{D,out}$$

Rounds	Worse Case Differential Probability
2	2^{-57}
4	2^{-114}
6	2^{-171}
8	2^{-228}
10	2^{-285}
12	2^{-342}
14	2^{-399}
16	2^{-456}

- $R = 6$ sufficient for 128-bit key
- $R = 10$ sufficient for 256-bit key

Linear Cryptanalysis

- Similar to differential cryptanalysis in many ways
- Estimate behavior of system using linear expressions

$$\left(\bigoplus_{i=1}^{16} X_i\right) = \left(\bigoplus_{i=1}^{16} Y_i\right)$$

- Ideal *linear probability*: $p = 1/2$
- Concerned with *linear bias*: $\epsilon = p - 1/2$
  - Deviation from ideal probability

Resistance to Linear Attacks

- Result: linear branch number = differential branch number
  - Sufficient condition: mixer matrix is symmetric [6]
- Our S-box has maximum linear bias: $\epsilon_{L,max} = 2^{-8}$
- Combine biases of linearly active S-box using Piling-Up Lemma

$$\epsilon = 2^{n-1} \prod_{i=1}^{n} \epsilon_i,$$

- where $n = \mathcal{B}_L = 3$ and $\epsilon_i = \epsilon_{L,max} = 2^{-8}$

Resistance to Linear Attacks

- Matsui: # PT/CT pairs needed is approximately ϵ^{-2} [11]

Rounds	Worst Case Linear Bias	PT/CT Pairs Required
2	2^{-22}	2^{44}
4	2^{-44}	2^{88}
6	2^{-66}	2^{132}
8	2^{-88}	2^{176}
10	2^{-110}	2^{220}
12	2^{-132}	2^{264}
14	2^{-154}	2^{308}
16	2^{-176}	2^{352}

- $R = 6$ sufficient for 128-bit key
- $R = 12$ sufficient for 256-bit key

Algebraic Attacks

- Unlike differential/linear attacks, not probabilistic in nature
- Goal: find mathematical models of a system that are always true
- Example: compact representation of AES by Ferguson et. al [7]
- Initial raised huge alarm in cryptographic community

Resistance to Algebraic Attacks

- Finding models is “easy”
- Solving models is the problem
- Models are necessarily highly nonlinear because of S-box
- We’re good at linear systems, so attempt to reduce to linear
- So far, no practical attacks due to S-box complexity
- Our S-box is even larger than AES with higher algebraic complexity
- Conclusion: algebraic attacks extremely unlikely

Statistical Testing

- Not a substitute for real cryptanalysis!
- But can give insight into behavior
- We used STS and manual avalanche testing
- STS results: no statistical anomalies
    - Similar to most AES candidates
- Avalanche testing: full diffusion after 3 rounds

Overview

Conclusions

- AE is an extremely popular and important topic right now
- We provide our own novel solution
- Provides nearly all desired properties of AE:
  - Easy to use
  - Single key, single pass
  - Header support
  - Intermediate MACs
- **Easily** customizable on per-user / application basis

Future Work

- Actual hardware implementation
- More cryptanalysis - always
- Complete analysis of all 16-bit S-boxes by Wood
    - Easier customization
- Find larger matrices with maximum branch number (MDS matrices)
    - Decrease rounds
- Much more!

Questions

?

Bibliography I

[1] Mihir Bellare, Phillip Rogaway, and David Wagner.
The EAX Mode of Operation.
In *Fast Software Encryption*, pages 389–407. Springer, 2004.

[2] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.
Duplexing the sponge: single-pass authenticated encryption and other applications, August 2010.
http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/Aug2010/documents/presentations/DAEMEN_SpongeDuplexSantaBarbaraSlides.pdf.

[3] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.
Cryptographic Sponge Functions, 2011.
http://http://sponge.noekeon.org/CSF-0.1.pdf.

[4] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.
On the Security of the Keyed Sponge Construction, 2011.
http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/Aug2010/documents/papers/VANASSCHE_SpongeKeyed.pdf.

[5] Nikita Borisov, Ian Goldberg, and David Wagner.
Intercepting Mobile Communications: The Insecurity of 802.11.
In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, pages 180–189. ACM, 2001.

[6] Joan Daemen and Vincent Rijmen.
The Design of Rijndael: AES-The Advanced Encryption Standard.
Springer, 2002.

[7] Niels Ferguson, Richard Schroeppel, and Doug Whiting.
A Simple Algebraic Representation of Rijndael.
In *Selected Areas in Cryptography*, pages 103–111. Springer, 2001.

Bibliography II

[8] Philipp Jovanovic, Atul Luykx, and Bart Mennink.
Beyond $2^{c/2}$ Security in Sponge-Based Authenticated Encryption Modes.
Cryptology ePrint Archive, Report 2014/373, 2014.
http://eprint.iacr.org/.

[9] Charanjit S. Jutla.
Encryption Modes with Almost Free Message Integrity.
In *Advances in Cryptology —- EUROCRYPT 2001*, pages 529–544. Springer, 2001.

[10] Tadayoshi Kohno, John Viega, and Doug Whiting.
CWC: A high-performance conventional authenticated encryption mode.
In *Fast Software Encryption*, pages 408–426. Springer, 2004.

[11] Mitsuru Matsui.
Linear Cryptanalysis Method for DES Cipher.
In *Advances in Cryptology—EUROCRYPT'93*, pages 386–397. Springer, 1994.

[12] David McGrew and John Viega.
The Galois/Counter Mode of Operation (GCM), 2004.
http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-spec.pdf.

[13] Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz.
OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption.
ACM Transactions on Information and System Security (TISSEC), 6(3):365–403, 2003.

[14] Serge Vaudenay.
Security Flaws Induced by CBC Padding—Applications to SSL, IPSEC, WTLS...
In *Advances in Cryptology—EUROCRYPT 2002*, pages 534–545. Springer, 2002.

Bibliography III

[15] Doug Whiting, Russ Housley, and Niels Ferguson.
Counter with CBC-MAC (CCM).
IETF Request for Comments: 3610, 2003.

[16] Doug Whiting, Bruce Schneier, Stefan Lucks, and Frédéric Muller.
Phelix: Fast Encryption and Authentication in a Single Cryptographic Primitive, 2005.
http://schneier.com/paper-phelix.pdf.

[17] Christopher A. Wood.
Large Substitution Boxes with Efficient Combinational Implementations.
Master's thesis, Rochester Institute of Technology, 2013.

[18] Hongjun Wu and Bart Preneel.
Differential-Linear Attacks Against the Stream Cipher Phelix.
In *Fast Software Encryption*, pages 87–100. Springer, 2007.