Hummingbird: Ultra-Lightweight Cryptography for Resource-Constrained Devices

Daniel Engels[2], Xinxin Fan[1], Guang Gong[1], Honggang Hu[1], and Eric Smith[2]

[1]Communication Security Lab (ComSec)
Department of Electrical and Computer Engineering
University of Waterloo
Email: x5fan@engmail.uwaterloo.ca

[2]Revere Security Corporation

RIM Seminar

Outline

1. Motivation
2. The Hummingbird Cryptographic Algorithm
   - Algorithm Specification
   - Security Analysis
3. Efficient Implementation of Hummingbird on Microcontrollers
   - Implementation Platforms
   - Software Implementation Results
4. Concluding Remarks

Security in Pervasive Computing

- A world of pervasive computing is just around the corner.
- Smart devices are penetrating into and impacting people's life.
  - access control
  - supply-chain management
  - home automation
  - healthcare
  - ......
- Smart devices usually have constrained capabilities in every aspect of computation, communication and storage.
- ⇒ (Ultra-)lightweight cryptographic primitives are needed.

Design Goals

- Main Goal: a novel (ultra-)lightweight cryptographic primitive for resource-constrained smart devices
  - Perform strong authentication and encryption
  - Provide other security functionalities
  - Efficient software and hardware implementations
- Three performance attributes:
  - The size of an implementation
  - The peak and the average power consumption
  - the time required to complete a computation

Basic Idea of Hummingbird Design

- Enigma belongs to a group of rotor-based crypto machines.
- The number of possible internal connections of Enigma has been estimated to be approximately $3 \cdot 10^{114}$.
- The basic idea of Hummingbird cipher lies in implementing extraordinarily large virtual rotors with custom block ciphers.

Figure: Enigma Machine

Outline

Top-Level Description of Hummingbird

- 4 identical block ciphers with 16-bit input and 16-bit output
- 4 16-bit registers acting as 4 rotors
- A 16-bit linear feedback shift register (LFSR)
- 256-bit key size
- 16-bit block size
- 80-bit internal state
- Extremely simple arithmetic and logic operations ($\oplus, \boxplus, \boxminus, \lll$)

Hummingbird Initialization Process

Nonce Initialization:

1. $RS1_0 = \text{NONCE}_1$
2. $RS2_0 = \text{NONCE}_2$
3. $RS3_0 = \text{NONCE}_3$
4. $RS4_0 = \text{NONCE}_4$

Four Iterations:

5. **for t = 0 to 3 do**

$$V12_t = E_{k_1}((RS1_t \boxplus RS3_t) \boxplus RS1_t)$$
$$V23_t = E_{k_2}(V12_t \boxplus RS2_t)$$
$$V34_t = E_{k_3}(V23_t \boxplus RS3_t)$$
$$TV_t = E_{k_4}(V34_t \boxplus RS4_t)$$
$$RS1_{t+1} = RS1_t \boxplus TV_t$$
$$RS2_{t+1} = RS2_t \boxplus V12_t$$
$$RS3_{t+1} = RS3_t \boxplus V23_t$$
$$RS4_{t+1} = RS4_t \boxplus V34_t$$

6. **end for**

LFSR Initialization:

7. $\text{LFSR} = TV_3 \mid 0\text{x}1000$

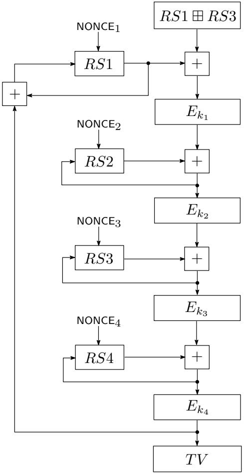


Figure: Initialization Process

Hummingbird Encryption Process

Block Encryption:

1. $V12_t = E_{k_1}(PT_i \boxplus RS1_t)$
2. $V23_t = E_{k_2}(V12_t \boxplus RS2_t)$
3. $V34_t = E_{k_3}(V23_t \boxplus RS3_t)$
4. $CT_i = E_{k_4}(V34_t \boxplus RS4_t)$

Internal State Updating:

5. $\mathrm{LFSR}_{t+1} \leftarrow \mathrm{LFSR}_t$
6. $RS1_{t+1} = RS1_t \boxplus V34_t$
7. $RS3_{t+1} = RS3_t \boxplus V23_t \boxplus \mathrm{LFSR}_{t+1}$
8. $RS4_{t+1} = RS4_t \boxplus V12_t \boxplus RS1_{t+1}$
9. $RS2_{t+1} = RS2_t \boxplus V12_t \boxplus RS4_{t+1}$

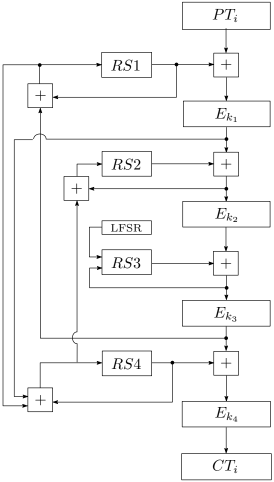


Figure: Encryption Process

Top-Level Specification of 16-bit Block Cipher

- Simple substitution-permutation network (SPN)
- 64-bit key size
- 16-bit block size
- Four 4 S-boxes (can use only one and repeat four times!)
- Simple linear transform for permutation layer
- 5 rounds (4 regular rounds + 1 final round)

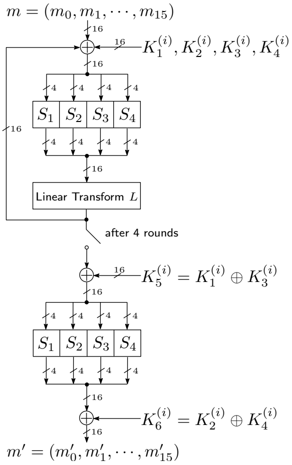


Figure: 16-bit Block Cipher

Outline

Security Analysis

- Differential Cryptanalysis
- Linear Cryptanalysis
- Structure Attack
- Algebraic Attack
- Cube Attack
- Slide and Related-key Attack
- Interpolation and Higher Order Differential Attack
- ......

Outline

Overview

- The compact version of Hummingbird is implemented.
  - A single 4×4 S-box S_1 is used four times in the 16-bit block cipher
- Two popular processors used in wireless sensor nodes:
  - 8-bit microcontroller ATmega128L from Atmel
  - 16-bit microcontroller MSP430 from Texas Instrument (TI)
- Two implementation variants for each platform:
  - Size optimized implementation
  - Speed optimized implementation

Outline

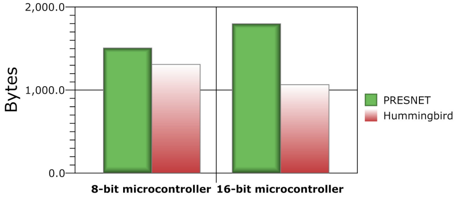
1. Motivation

2. The Hummingbird Cryptographic Algorithm
   - Algorithm Specification
   - Security Analysis

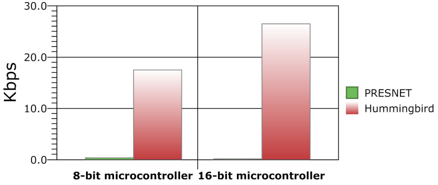
3. Efficient Implementation of Hummingbird on Microcontrollers
   - Implementation Platforms
   - Software Implementation Results

4. Concluding Remarks

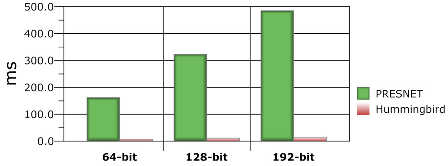
Size Optimized Implementation



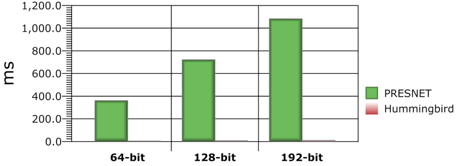
Memory Consumption



Throughput @ 4MHz

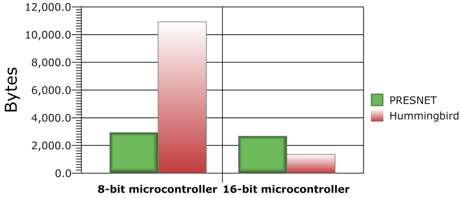


Overall Performance on 8-bit Microcontroller

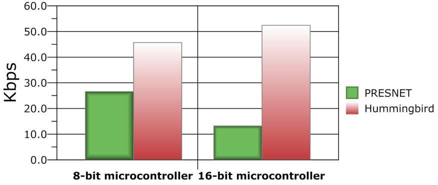


Overall Performance on 16-bit Microcontroller

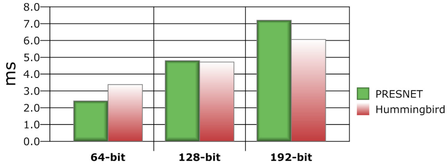
Speed Optimized Implementation



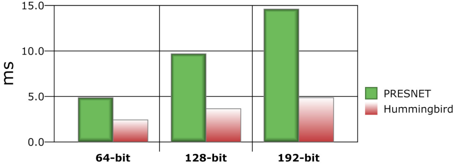
Memory Consumption



Throughput @ 4MHz



Overall Performance on 8-bit Microcontroller



Overall Performance on 16-bit Microcontroller

Concluding Remarks

- Hummingbird is a novel ultra-lightweight cryptographic algorithm, which is an elegant combination of block cipher and stream cipher.
- The hybrid structure adopted in Hummingbird can provide the designed security with small block size.
- Hummingbird seems to be resistant to the most common attacks to block ciphers and stream ciphers.
- Our experimental results show that after a system initialization procedure Hummingbird can achieve up to 147 and 4.7 times faster throughput for a size-optimized and a speed-optimized implementations, respectively.

Thanks for your attention!