# SIMON, SPECK and NISTIR 8114

Presented by
Louis Wingers, National Security Agency

NIST Lightweight Workshop 2016
October 17-18, 2016

# The Big Picture

What are the goals of a lightweight cryptography standard?

A major goal is securing the “Internet of Things”, which consists of many small, constrained devices operating in heterogeneous networks. Technology is changing rapidly, and we can’t predict what future devices will look like. Flexibility is important to cope with unforeseen circumstances.

We take a broad view of the IoT — devices don’t necessarily need to communicate through the Internet (or even a hub).

# Profiles

The “profiles” approach taken in the NISTIR document seems sensible in the near term. But in the long term, it might run counter to the goal of supporting IoT security.

- It could lead to fragmentation, where different algorithms are identified for every use case. If we take a longer view, support for heterogeneous networks should be of prime importance, where flexible algorithms would be preferred.
- Identification of very specific thresholds for size, power, throughput, latency, etc., could lead to standards which are obsolete even before they are published.

# Profiles

It might be better to have more generic profiles, created with input from the industries involved, and which will remain relevant in 10 years or more. For this, it's best if they're not strongly tied to current systems.

Care should be taken so that profiles aren't misused, i.e., written to carve out a space for a particular algorithm. Characteristics should not be unnecessarily restricted.

# Profiles

On sample profile #2: Why is a 20 ns latency required for the automobile CAN application, when car systems operate on a time scale of tens of milliseconds— a million times slower?

Even if a profile makes sense today, will it make sense in 2030?

In 2030 it seems reasonable that the time to deploy an airbag will still be in milliseconds, even if the CAN no longer operates at 50 MHz (i.e. 20 ns per clock cycle).

The platforms will change, the physics won't.

# Block Size

Should there be a minimum block size?

- Small block sizes might be necessary for many highly constrained applications. This is why so many lightweight ciphers have 64-bit blocks.
- Small block sizes have security issues as the amount of encrypted plaintext approaches $O(2^{n/2})$. While the key is not recoverable, plaintext may be leaked.
- How do we prevent misuse of a cipher with a small block size? Once standardized a cryptographic algorithm may be used in unintended ways. Are written guidelines enough?

# Block Size

- Using a block cipher in a tweak mode solves these issues. This is easy for some ciphers, e.g., SKINNY.
- Classical block ciphers can be made into tweakable block ciphers in various ways.[1] Some performance degradation may occur.
- When not used in a tweak mode, restricting the amount of data to be encrypted is effective. For many lightweight applications the amount of data encrypted for a fixed key will be relatively small. Can data restrictions be enforced?

What's the cutoff — 32, 48, 64, 80, 96 bits?

[1] How to Build Fully Secure Tweakable Blockciphers from Classical Blockciphers by Wang, Guo, Zhang, Zhao and Gu, eprint 2016/876

# Key Size and Security

NISTIR 8114 states (line 451): The security against key-recovery attacks should be at least 112 bits.

Even-Mansour schemes such as PRINCE and Chaskey assume a weakened security model. Allowing $2^d$ data, the key can be recovered with $2^{k-d}$ work. For a 128-bit key, a $2^{16}$ block data limit must be enforced (across all users on common key).

Mavromati showed that with $2^{32}$ different keys each producing $2^{32}$ plain/cipher pairs, we can recover 2 of the keys with $O(2^{64})$ total work[2]. The Chaskey documentation recommends a data limit of $2^{48}$ message blocks, so this is a reasonable amount of data for each key.

---

[2] C. Mavromati, "Key-recovery attacks against the MAC algorithm Chaskey", ePrint 2015/811, 2015.

# Key Size and Security

Some questions about Even-Mansour schemes —

- What do these designs (developed in a weaker security model) buy us relative to more traditional designs?
- Will developers be confused by what a 128-bit key means? Will they think that a 128-bit Even-Mansour block cipher with a 128-bit key has security equal to AES-128?
- Should there be a minimum amount of data which a block cipher needs to be able to encrypt under a fixed key?

# Security Update

SIMON and SPECK were released in June 2013.
About 60 cryptanalysis papers.

| Algorithm | Atk/Tot | Margin | Algorithm | Atk/Tot | Margin |
|---|---|---|---|---|---|
| SIMON 64/96 | 30/42 | 29% | SPECK 64/96 | 19/26 | 27% |
| SIMON 64/128 | 31/44 | 30% | SPECK 64/128 | 20/27 | 26% |
| SIMON 96/96 | 37/52 | 29% | SPECK 96/96 | 20/28 | 29% |
| SIMON 96/144 | 38/54 | 30% | SPECK 96/144 | 21/29 | 28% |
| SIMON 128/128 | 49/68 | 28% | SPECK 128/128 | 23/32 | 28% |
| SIMON 128/192 | 51/69 | 26% | SPECK 128/192 | 24/33 | 27% |
| SIMON 128/256 | 53/72 | 26% | SPECK 128/256 | 25/34 | 26% |

AES-128 has a 30% security margin (7 of 10 rounds attacked).

Design rationale found in “The SIMON and SPECK Lightweight Block Ciphers”, DAC 2016.

# Software and Hardware

For hardware, we think the most important attributes are

- size (related to power and cost)
- throughput (related to energy).

These seem to be important for some RFID applications.

For software, code size, RAM usage and throughput are all important.

# Algorithm Flexibility

SIMON was optimized for hardware
SPECK was optimized for software

Nevertheless,

- SIMON is among the best algorithms in software
- SPECK is among the best algorithms in hardware

They prove that it is not necessary to design highly specialized algorithms for specific platforms.

# Small ASIC Comparisons

We compare hardware performance of SPECK against SKINNY (a good hardware-optimized design).

| | Area (GE) | Tput@100kHz (kbps) |
|---|---|---|
| SIMON 64/128 | 958 | 4.2 |
| SPECK 64/128 | 996 | 3.4 |
| SKINNY 64/128 | 1172 | 2.0 |

# Small ASIC Comparisons

We compare hardware performance of SPECK against SKINNY (a good hardware-optimized design).

| | Area (GE) | Tput@100kHz (kbps) |
|---|---|---|
| SIMON 64/128 | 958 | 4.2 |
| SPECK 64/128 | 996 | 3.4 |
| SKINNY 64/128 | 1172 | 2.0 |

| | | |
|---|---|---|
| SIMON 64/128 | 1416 | 139.1 |
| SPECK 64/128 | 1247 | 27.6 |
| SKINNY 64/128 | 1399 | 8.1 |

For comparable or smaller size —
SPECK has higher throughput than SKINNY.

# Small ASIC Comparisons

For 128-bit variants

| | Area (GE) | Tput@100kHz (kbps) |
|---|---|---|
| SIMON 128/128 | 1242 | 5.7 |
| SPECK 128/128 | 1280 | 3.0 |
| SKINNY 128/128 | 1481 | 1.8 |

# Small ASIC Comparisons

For 128-bit variants

| | Area (GE) | Tput@100kHz (kbps) |
|---|---|---|
| SIMON 128/128 | 1242 | 5.7 |
| SPECK 128/128 | 1280 | 3.0 |
| SKINNY 128/128 | 1481 | 1.8 |

| | | |
|---|---|---|
| SIMON 128/128 | 2097 | 182.9 |
| SPECK 128/128 | 1732 | 48.5 |
| SKINNY 128/128 | 1840 | 14.7 |

| | | |
|---|---|---|
| AES-128 | 2400 | 56.6 |

Up to around 60 MHz, SPECK seems to outperform SKINNY.
Beyond that SKINNY will probably have better ASIC efficiency.

# Fast FPGA Comparison

On Virtex-7 XC7VX330T

| | Area (Luts) | FF's | T'put (Gbps) |
|---|---|---|---|
| SIMON 64/128 | 3298 | 4290 | 45.5 |
| SPECK 64/128 | 4369 | 4940 | 37.8 |
| SKINNY 64/128 | 4247 | 6720 | 25.8 |

For comparable or smaller size —
SPECK has higher throughput than SKINNY.

# Fast FPGA Comparison

For 128-bit variants

| | Area (Luts) | FF's | T'put (Gbps) |
|---|---|---|---|
| SIMON 128/128 | 8579 | 8898 | 87.5 |
| SPECK 128/128 | 10237 | 8151 | 56.1 |
| SKINNY 128/128 | 13389 | 10048 | 41.0 |

# Microcontroller Comparisons

| | AVR | | | MSP | | | |
|---|---|---|---|---|---|---|---|
| cipher | ROM | RAM | Time | ROM | RAM | Time | FOM |
| Chaskey-8 | 770 | 84 | 1597 | 490 | 86 | 1351 | 4.7 |
| SPECK 64/96 | 448 | 53 | 2829 | 328 | 48 | 1959 | 4.8 |
| SPECK 64/128 | 452 | 53 | 2917 | 332 | 48 | 2013 | 4.8 |
| Chaskey-16 | 770 | 84 | 2413 | 492 | 86 | 2064 | 5.3 |
| SIMON 64/96 | 600 | 57 | 4269 | 460 | 56 | 2905 | 6.5 |
| SIMON 64/128 | 608 | 57 | 4445 | 468 | 56 | 3015 | 6.7 |
| LEA | 906 | 80 | 4023 | 722 | 78 | 2814 | 7.5 |
| RECTANGLE 64/128 | 602 | 56 | 4381 | 480 | 54 | 2651 | 8.0 |
| RECTANGLE 64/80 | 606 | 56 | 4433 | 480 | 54 | 2651 | 8.0 |
| +15 more entries | | | | | | | |

FOM score also based on ARM implementation (not shown). Smaller FOM is better.

FELICS table (scenario 2) encrypting 128 bits in counter mode. Note that SIMON (a hardware-optimized design) scores very highly. It surpasses many software designs (e.g., LEA (FOM 7.5) and PRIDE (FOM 24.0)).

# Fast x86 Comparison

Counter mode (long messages).
Numbers in cycles/byte (smaller is better).

| Algorithm | Westmere | Haswell |
|---|---|---|
| SPECK 64/128 | 2.59 | 1.11 |
| SIMON 64/128 | 4.13 | 1.57 |
| SKINNY 64/128 | 7.45 | 2.47 |

# Fast x86 Comparison

Counter mode (long messages).
Numbers in cycles/byte (smaller is better).

| Algorithm | Westmere | Haswell |
|---|---|---|
| SPECK 64/128 | 2.59 | 1.11 |
| SIMON 64/128 | 4.13 | 1.57 |
| SKINNY 64/128 | 7.45 | 2.47 |

| Algorithm | Westmere | Haswell |
|---|---|---|
| SPECK 128/128 | 2.91 | 1.29 |
| SIMON 128/128 | 6.14 | 2.26 |
| SKINNY 128/128 | N/A | 3.66 |

SIMON outperforms SKINNY.

# Parting thoughts

- The goals of a lightweight standard should be carefully considered.
- There is value in selecting algorithms which perform well in multiple environments, rather than “stovepipe” algorithms.
- Security models, block sizes, and data limits are important, and usage guidance should be given. Security should align with NIST’s stated security targets.

Questions/Comments?