Permutation-based encryption, authentication and authenticated encryption

Joan DAEMEN\textsuperscript{1}

Joint work with
Guido BERTONI\textsuperscript{1}, Michaël PEETERS\textsuperscript{2} and Gilles VAN ASSCHE\textsuperscript{1}

\textsuperscript{1}STMicroelectronics \textsuperscript{2}NXP Semiconductors

DIAC 2012, Stockholm, July 6
Modern-day cryptography is block-cipher centric

- (Standard) hash functions make use of block ciphers
  - SHA-1, SHA-256, SHA-512, Whirlpool, RIPEMD-160, ...
  - So HMAC, MGF1, etc. are in practice also block-cipher based

- Block encryption: ECB, CBC, ...
- Stream encryption:
  - synchronous: counter mode, OFB, ...
  - self-synchronizing: CFB
- MAC computation: CBC-MAC, C-MAC, ...
- Authenticated encryption: OCB, GCM, CCM ...
Structure of a block cipher

Key Schedule

Data Path

Key

Data in

Expanded Key

Data out
Permutation-based encryption, authentication and authenticated encryption

Structure of a block cipher (inverse operation)
When is the inverse block cipher needed?

Indicated in red:

- Hashing and its modes HMAC, MGF1, ...
- **Block encryption**: ECB, CBC, ...
- **Stream encryption**:
  - synchronous: counter mode, OFB, ...
  - self-synchronizing: CFB
- **MAC computation**: CBC-MAC, C-MAC, ...
- **Authenticated encryption**: OCB, GCM, CCM ...

So a block cipher without inverse can do a lot!
Permutation-based encryption, authentication and authenticated encryption

Modern-day cryptography is block-cipher centric

Your typical block cipher

Block cipher internals
Permutation-based encryption, authentication and authenticated encryption

Modern-day cryptography is block-cipher centric

Designer’s view of a block cipher

**Designer’s view of a block cipher**

$n$-bit block cipher with $|K|$-bit key

$b$-bit permutation with $b = n + |K|

- obtained by repeating an invertible round function
- with an efficient inverse
- and no diffusion from data part to key part
Permutation-based encryption, authentication and authenticated encryption

Modern-day cryptography is block-cipher centric

How it is typically used

Hashing use case: Davies-Meyer compression function
Permutation-based encryption, authentication and authenticated encryption

Modern-day cryptography is block-cipher centric

Why limit diffusion from left to right?

Removing diffusion restriction not required in hashing
Permutation-based encryption, authentication and authenticated encryption

Modern-day cryptography is block-cipher centric

So iterated permutation is at the same time simpler and more efficient!

Simplifying the view: iterated permutation
Permutation-based encryption, authentication and authenticated encryption

Modern-day cryptography is block-cipher centric

Block cipher without inverse: wide permutation

Block cipher without inverse: wide permutation

- Previous applies to all modes where inverse is not needed
- Requirement of separate key schedule vanishes
- $n$-bit block cipher replaced by $b$-bit permutation with
  - $b = n + |K|$
- Permutation as a generalization of a block cipher
- Less is more!
Permutation-based construction: sponge

- \( f \): a \( b \)-bit permutation with \( b = r + c \)
  - efficiency: processes \( r \) bits per call to \( f \)
  - security: provably resists generic attacks up to \( 2^{c/2} \)
- Flexibility in trading rate \( r \) for capacity \( c \) or vice versa
What can we say about sponge security

- **Generic security:**
  - assuming $f$ has been chosen randomly
  - covers security against generic attacks
  - construction as sound as theoretically possible

- **Security for a specific choice of $f$:**
  - security proof is infeasible
  - Hermetic Sponge Strategy
  - design with attacks in mind
  - security based on absence of attacks despite public scrutiny
Permutation-based encryption, authentication and authenticated encryption

Applications

What can you do with a sponge function?

Regular hashing

- Pre-sponge permutation-based hash functions
  - Truncated permutation as compression function: Snefru [Merkle ’90], FFT-Hash [Schnorr ’90], ...MD6 [Rivest et al. 2007]
  - Streaming-mode: SUBTERRANEAN, PANAMA, RADIOGATÚN, Grindahl [Knudsen, Rechberger, Thomsen, 2007], ...
Message authentication codes

- Pre-sponge (partially) permutation-based MAC function: Pelican-MAC [Daemen, Rijmen 2005]
Applications

What can you do with a sponge function?

Stream encryption

- Similar to block cipher modes:
  - Long keystream per IV: like OFB
  - Short keystream per IV: like counter mode

- Independent permutation-based stream ciphers: Salsa and ChaCha [Bernstein 2007]
Mask generating function

What can you do with a sponge function?
Authenticated encryption: MAC generation
Authenticated encryption: encryption
Authenticated encryption: just do them both?
The duplex construction

Object: $D = \text{DUPLEX}[f, \text{pad}, r]$

- Requesting $\ell$-bit output $Z = D.\text{duplexing}(\sigma, \ell)$
- Generic security equivalent to that of sponge
SpongeWrap authenticated encryption

- Single-pass authenticated encryption
- Processes up to \( r \) bits per call to \( f \)
- Functionally similar to (P)helix [Lucks, Muller, Schneier, Whiting, 2004]
Key $K$, data header $A$ and data body $B$ of arbitrary length
- Confidentiality assumes unicity of data header
- Supports intermediate tags
Sponge functions: existing proposals to date

<table>
<thead>
<tr>
<th>Sponge function</th>
<th>Authors</th>
<th>Conference</th>
<th>Output sizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keccak</td>
<td>Bertoni, Daemen, Peeters, Van Assche</td>
<td>SHA-3 2008</td>
<td>25, 50, 100, 200, 400, 800, 1600</td>
</tr>
<tr>
<td>Quark</td>
<td>Aumasson, Henzen, Meier, Naya-Plasencia</td>
<td>CHES 2010</td>
<td>136, 176, 256</td>
</tr>
<tr>
<td>Photon</td>
<td>Guo, Peyrin, Poschmann</td>
<td>Crypto 2011</td>
<td>100, 144, 196, 256, 288</td>
</tr>
<tr>
<td>Spongent</td>
<td>Bogdanov, Knezevic, Leander, Toz, Varici, Verbauwhede</td>
<td>CHES 2011</td>
<td>88, 136, 176, 248, 320</td>
</tr>
</tbody>
</table>
The current perception

- Quark, Photon, Spongent: *lightweight hash functions*
- Lightweight is synonymous with low-area here
- Easy to see why. Let us target security strength $c/2$
  - Davies-Meyer block cipher based hash (*“narrow pipe”*)
    - chaining value (block size): $n \geq c$
    - input block size (key length): typically $k \geq n$
    - feedforward (block size): $n$
    - total state $\geq 3c$
- Sponge (*“huge state”*)
  - permutation width: $c + r$
  - $r$ can be made arbitrarily small, e.g. 1 byte
  - total state $\geq c + 8$
The current perception (continued)

One cryptographic expert’s opinion:

“The sponge construction is a pretty poor way to encrypt. One either gets high-speed but low security or low-speed and high security.”

- KECCAK showed that sponge can be secure and fast
- Keyed sponge still perceived as possible but inefficient
  - higher speed expected from MAC and stream encryption
  - competing proposals in keyed applications are faster
Permutations vs block ciphers

Unique block cipher features
- pre-computation of key schedule
  - storing expanded key costs memory
  - may be prohibitive in resource-constrained devices
- misuse resistance
  - issue: keystream re-use in stream encryption
  - not required if nonces are affordable or available
  - address it with decent nonce management

Unique permutation features
- diffusion across full state
- flexibility in choice of rate/capacity
Boosting keyed permutation modes

- Taking a closer look at rate/capacity trade-off
  - keyed generic security is $c - a$ instead of $c/2$
  - with $2^a$ ranging from data complexity down to 1
  - allows increasing the rate

- Distinguishing vulnerability in keyed vs unkeyed modes
  - in keyed modes attacker has less power
  - allows decreasing number of rounds in permutation

- Introducing dedicated variants
  - MAC computation
  - authenticated encryption strongly relying on nonces
Distinguishing attack setup

- $M$: online **data** complexity ($r$-bit blocks)
- $N$: offline **time** complexity (calls to $f$)

\[
\text{If } M = 2^a \ll 2^{c/2}
\]

Expected time complexity is about \(\min(2^{c-a-1}, 2^{|K|})\)
Intuition behind $2^{c-a-1}$

CICO problem:
- given $r$ input and $r$ output to $f$, determine remaining $c$ bits
- expected workload: $2^c$ computations of $f$
Intuition behind $2^{c-a-1}$

Multi-target CICO problem (with multiplicity $\mu$):
- $\mu$ instances with same partial $r$-bit input
- expected workload: $2^c / \mu$ computations of $f$
Permutation-based encryption, authentication and authenticated encryption

Boosting keyed permutation modes

Taking a closer look at rate/capacity trade-off

**Intuition behind $2^{c-a-1}$**

Multi-target CICO problem (with multiplicity $\mu$):

- $\mu$ instances with same partial $r$-bit input
- expected workload: $2^c / \mu$ computations of $f$
Intuition behind $2^{c-a-1}$

Multi-target CICO problem (with multiplicity $\mu$):
- $\mu$ instances with same partial $r$-bit input
- expected workload: $2^c / \mu$ computations of $f$
Intuition behind $2^{c-a-1}$: multiplicity

- **Multiplicity $\mu$:**
  - # CICO instances with same $r$-bit part
  - Upper bound: $\mu \leq 2^a$

- In most modes attacker cannot force high multiplicity
  - MAC computation: absolute input unknown
  - keystream generation: each $r$-bit input different
  - authenticated encryption, passive attacker

- Counting on collisions in $r$-bit (input or output) part
  - If $a \ll r$, multiplicity $\mu$ small
  - if $a > r$, multiplicity $\mu$ of order $2^{a-r}$
Numeric example

Say we have the following requirements:
- we have a permutation with width 200 bits
- we want to realize different functions
- desired security strength: 80 bits
- we assume active adversary, limited to $2^{48}$ data complexity

Collision-resistant hashing: $c = 2 \times 80 \Rightarrow r = 40$
SpongeWrap: $c = 80 + 48 + 1 \Rightarrow r = 71$
MAC computation: $c = 80 \Rightarrow r = 120$
Unkeyed modes weaker than keyed modes?

- **MD5 hash function** [Rivest 1992]
  - unkeyed: collisions usable in constructing fake certificates [Stevens et al. 2009]
  - keyed: very little progress in 1st pre-image generation

- **PANAMA hash and stream cipher** [Clapp, Daemen 1998]
  - unkeyed: instantaneous collisions [Daemen, Van Assche 2007]
  - keyed: stream cipher unbroken till this day

- **Keccak crypto contest with reduced-round challenges**
  - unkeyed: collision challenges up to 4 rounds broken [Dinur, Dunkelman, Shamir 2012]
  - keyed: 1st pre-image challenges up to 2 rounds broken [Morawiecki 2011]
KECCAK-$f$: the permutations in KECCAK

Operates on 3D state:

- (5 x 5)-bit slices
- $2^\ell$-bit lanes
- param. $0 \leq \ell < 7$

Round function with 5 steps:
- $\theta$: mixing layer
- $\rho$: inter-slice bit transposition
- $\pi$: intra-slice bit transposition
- $\chi$: non-linear layer
- $\iota$: round constants

- Lightweight, but high diffusion
- # rounds: $12 + 2\ell$ for $b = 2^\ell 25$
  - 12 rounds in KECCAK-$f[25]$
  - 24 rounds in KECCAK-$f[1600]$
- High safety margin, even if unkeyed
KECCAK: reference versions

- **KECCAK with default parameters:** KECCAK[]
  - width $b = 1600$: largest version
  - rate $r = 1024$: a round number
  - gives generic security strength $c/2 = 288$ bits
  - roughly 7% slower than the KECCAK SHA-3 256-bit candidate
  - For performance see eBash, Athena, XBX, etc.

- **KECCAK[r=40, c=160]**
  - width $b = 200$: small state
  - $c = 160$, generic security strength 80 bits
  - gives rate of $r = 40$
  - roughly 2.4 more work per input/output bit than KECCAK
KECCUP: reduced-round versions of KECCAK

- For keyed modes use reduced-round versions of KECCAK-f
  - called KECCUP\([r, c, n]\) and KECCUP-f\([b, n]\)
  - we assume that the multiplicity is below \(2^{64}\)
- Same can be done for any iterated permutation
  - Quark, Photon, Spongent
  - JH’s E8
  - Gröstl’s P512, Q512, P1024, Q1024
  - ECHO, Cubehash, etc.
  - block cipher with fixed key: e.g., Rijndael
Keyed sponge and duplex with KECCUP

Some KECCUP varieties that we think are reasonable:

| width $b$ | strength $|K|$ | capacity $c$ | rate $r$ | # rounds | speedup |
|-----------|----------------|--------------|-----------|-----------|---------|
| 1600      | 128            | 192          | 1408      | 10        | 3.3     |
| 1600      | 256            | 320          | 1280      | 11        | 2.7     |
| 200       | 80             | 144          | 56        | 9         | 2.8     |
| 200       | 128            | 192          | 8         | 6         | 0.6     |
Introducing dedicated variants

- Sponge and duplex are generic constructions
  - flexible and multi-purpose
  - do not exploit mode-specific adversary limitations
- MAC computation
  - before squeezing adversary has no information about state
  - relaxes requirements on $f$ during absorbing
- Authenticated encryption in presence of nonces
  - nonce can be used to *decorrelate* computations
MAC: take a look at Pelican-MAC [Daemen, Rijmen, 2005]

- Block cipher based MAC
  - application of Alred
  - based on Rijndael (AES)
  - permutation-based absorbing

- Speed: for long messages:
  - 4 rounds per 128 bits
  - 2.5 times faster than AES

- Security rationale
  - key recovery: block cipher
  - secret state recovery:
    - block cipher at the end
    - hardness of inner collisions
    - relies on low MDP of AES 4R
    - security claims with $2^a \leq 2^{60}$
Permutation-based encryption, authentication and authenticated encryption

- Boosting keyed permutation modes
- Introducing dedicated variants

The donkeySponge MAC construction

- Usage of full state width $b$ during absorbing
- Reduced number of rounds during init and absorbing
- Truncated permutation instead of final block cipher
Applying donkeySponge to KECCUP

- KECCUP proposed values:
  - $n_{\text{init}} = 3$: sufficient to make all state bits depend on the key
  - $n_{\text{absorb}} = 6$: dictated by MDP estimation
  - $n_{\text{squeeze}} = 12$: dictated by chosen-input-difference attacks

- $b = 1600$ and $|K| = 256$: gains factor 6.25
- $b = 200$ and $|K| = 128$: gains factor 15
The monkeyDuplex construction

- For authenticated encryption and keystream generation
- Initialization: key, nonce and strong permutation
- Reduced number of rounds in duplex calls
Permutation-based encryption, authentication and authenticated encryption

Boosting keyed permutation modes
Introducing dedicated variants

monkeyDuplex rationale

- **Initialization**
  - decorrelates states for different nonces
  - is assumed to rule out differential attacks

- **Remaining attack: state reconstruction**
  - high rate: solving CICO problem
  - low rate: multiple iterations of $f$ must be considered
  - Number of rounds to span: $n_{unicity}$
Some monkeyDuplex KECCUP varieties

- $n_{\text{init}} = 12$: dictated by chosen-input-difference attacks
- For $b = 200$ we propose $n_{\text{duplex}} = 1$: streaming mode
- For $b = 1600$ we propose $2r > b$: blockwise mode

| $b$   | $|K|$ | $c$  | $r$   | $n_{\text{duplex}}$ | $n_{\text{unicity}}$ | speedup |
|-------|------|------|-------|----------------------|-----------------------|---------|
| 1600  | 256  | 320  | 1280  | 8                    | 8                     | 3.75    |
| 200   | 80   | 184  | 16    | 1                    | 12                    | 7.2     |
Conclusions

- Iterated permutations
  - versatile cryptographic primitives
  - more flexible modes than with block ciphers

- Permutation-based keyed modes can be boosted
  - generic security: reducing capacity from $2|K|$ to $|K| + a$
  - permutation-specific security: reducing $\#$ rounds
  - mode-specific security: dedicated constructions
Permutation-based encryption, authentication and authenticated encryption

That's it, folks!

Questions?

Thanks for your attention!

More information on
http://keccak.noekeon.org/
http://sponge.noekeon.org/