From Software Security
To Cryptographic Research
In the ARM World

Roberto Avanzi
Architecture and Technology Group
roberto.avanzi@arm.com

Tel Aviv – 2018 LCD – April 29th, 2018

© ARM 2018
Prologue
Security vs. Cryptography

Are cryptography and security two distinct worlds? No!

Cryptography sits at the crossroads of security, mathematics, computer science, and engineering.
Security vs. Cryptography

Are cryptography and security two distinct worlds? No!
Cryptography sits at the crossroads of security, mathematics, computer science, and engineering.
Security vs. Cryptography

Are cryptography and security two distinct worlds? No!
Cryptography sits at the crossroads of security, mathematics, computer science, and engineering.
Security vs. Cryptography

Are cryptography and security two distinct worlds? No!
Cryptography sits at the crossroads of security, mathematics, computer science, and engineering.
Security vs. Cryptography

Are cryptography and security two distinct worlds? No!
Cryptography sits at the crossroads of security, mathematics, computer science, and engineering.
Security vs. Cryptography

Are cryptography and security two distinct worlds? No!
Cryptography sits at the crossroads of security, mathematics, computer science, and engineering.
Security vs. Cryptography

Are cryptography and security two distinct worlds? No!
Cryptography sits at the crossroads of security, mathematics, computer science, and engineering.
Security vs. Cryptography

Are cryptography and security two distinct worlds? No!
Cryptography sits at the crossroads of security, mathematics, computer science, and engineering.

This work fits here
But, what is security, actually?
Security deals with Security Problems (D’oh)

- A security Problem exists when an asset of a stakeholder can be compromised
- Your Whatsapp msgs – but also CCI, government / military secrets
- Depends on culture: It is a social construct
- An attacker invests resources to obtain some gain
- Goal: reduce attacker’s ROI – they have to invest $\geq (\text{asset’s value for attacker})$
  ... but not spend yourself $\geq (\text{asset’s value for stakeholder})$ (Constraints!)
- Cost can be human resources, silicon, memory, performance...
- Addressing security is always a business decision (!)
Security deals with Security Problems (D’oh)

- A security Problem exists when an asset of a stakeholder can be compromised
- Your Whatsapp msgs – but also CCI, government / military secrets
- Depends on culture: It is a social construct
- An attacker invests resources to obtain some gain
- Goal: reduce attacker’s ROI – they have to invest $\geq (\text{asset’s value for attacker})$
  ... but not spend yourself $\geq (\text{asset’s value for stakeholder})$ (Constraints!)
- Cost can be human resources, silicon, memory, performance...
- Addressing security is always a business decision (!)
Security deals with Security Problems (D’oh)

- A security Problem exists when an asset of a stakeholder can be compromised
- Your Whatsapp msgs – but also CCI, government / military secrets
  - Depends on culture: It is a social construct
  - An attacker invests resources to obtain some gain
  - Goal: reduce attacker’s ROI – they have to invest \( \geq \) (asset’s value for attacker)
    ... but not spend yourself \( \geq \) (asset’s value for stakeholder) (Constraints!)
- Cost can be human resources, silicon, memory, performance...
- Addressing security is always a business decision (!)
Security deals with Security Problems (D’oh)

- A security Problem exists when an asset of a stakeholder can be compromised.
- Your Whatsapp msgs – but also CCI, government/military secrets.
- Depends on culture: It is a social construct.

- An attacker invests resources to obtain some gain.
  
  Goal: reduce attacker’s ROI – they have to invest $ \geq (\text{asset’s value for attacker})$
  
  ... but not spend yourself $\geq (\text{asset’s value for stakeholder})$ (Constraints!)
- Cost can be human resources, silicon, memory, performance...
- Addressing security is always a business decision (!)
Security deals with Security Problems (D’oh)

- A security Problem exists when an asset of a stakeholder can be compromised
- Your Whatsapp msgs – but also CCI, government / military secrets
- Depends on culture: It is a social construct
- An attacker invests resources to obtain some gain
  - Goal: reduce attacker’s ROI – they have to invest \( \geq (\text{asset’s value for attacker}) \)
  - ... but not spend yourself \( \geq (\text{asset’s value for stakeholder}) \) (Constraints!)
- Cost can be human resources, silicon, memory, performance...
- Addressing security is always a business decision (!)
Security deals with Security Problems (D’oh)

- A security Problem exists when an asset of a stakeholder can be compromised
- Your Whatsapp msgs – but also CCI, government/military secrets
- Depends on culture: It is a social construct
- An attacker invests resources to obtain some gain
- Goal: reduce attacker’s ROI – they have to invest \( \geq (\text{asset’s value for attacker}) \)
  ... but not spend yourself \( \geq (\text{asset’s value for stakeholder}) \) (Constraints!)
- Cost can be human resources, silicon, memory, performance...
- Addressing security is always a business decision (!)
Security deals with Security Problems (D’oh)

- A security Problem exists when an asset of a stakeholder can be compromised
- Your Whatsapp msgs – but also CCI, government / military secrets
- Depends on culture: It is a social construct
- An attacker invests resources to obtain some gain
- Goal: reduce attacker’s ROI – they have to invest $\geq$ (asset’s value for attacker)
  ... but not spend yourself $\geq$ (asset’s value for stakeholder) (Constraints!)
- Cost can be human resources, silicon, memory, performance...
- Addressing security is always a business decision (!)
A security Problem exists when an asset of a stakeholder can be compromised

Your Whatsapp msgs – but also CCI, government/military secrets

Depends on culture: It is a social construct

An attacker invests resources to obtain some gain

Goal: reduce attacker’s ROI – they have to invest $\geq$ (asset’s value for attacker)
... but not spend yourself $\geq$ (asset’s value for stakeholder) (Constraints!)

Cost can be human resources, silicon, memory, performance...

Addressing security is always a business decision (!)
Security deals with Security Problems (D’oh)

- A security Problem exists when an asset of a stakeholder can be compromised
- Your Whatsapp msgs – but also CCI, government/military secrets
- Depends on culture: It is a social construct
- An attacker invests resources to obtain some gain
- Goal: reduce attacker’s ROI – they have to invest \( \geq (\text{asset’s value for attacker}) \)
  
  ... but not spend yourself \( \geq (\text{asset’s value for stakeholder}) \) (Constraints!)
- Cost can be human resources, silicon, memory, performance...
- Addressing security is always a business decision (!)
Constraints – in other words, what does lightweight mean?

Suppose you want to add a feature to a product, and this feature requires some cryptographic functionality in the critical path.

You start by investigating standards, then rest of literature.

If all solutions are too “heavy” – hence they make your product too bloated:

⇒ customers would not enable the feature
⇒ you can forget about implementing it in the first place
⇒ the feature will never see the light of the day!
Constraints – in other words, what does lightweight mean?

Suppose you want to add a feature to a product, and this feature requires some cryptographic functionality in the critical path.

You start by investigating standards, then rest of literature.

If all solutions are too “heavy” – hence they make your product too bloated:

⇒ customers would not enable the feature
⇒ you can forget about implementing it in the first place
⇒ the feature will never see the light of the day!
Constraints – in other words, what does lightweight mean?

Suppose you want to add a feature to a product, and this feature requires some cryptographic functionality in the critical path.

*You start by investigating standards, then rest of literature*

If all solutions are too “heavy” – hence they make your product too bloated:

⇒ customers would not enable the feature
⇒ you can forget about implementing it in the first place
⇒ the feature will never see the light of the day!
Constraints – in other words, what does lightweight mean?

Suppose you want to add a feature to a product, and this feature requires some cryptographic functionality in the critical path

You start by investigating standards, then rest of literature

If all solutions are too “heavy” – hence they make your product too bloated:

⇒ customers would not enable the feature
⇒ you can forget about implementing it in the first place
⇒ the feature will never see the light of the day!
Constraints – in other words, what does lightweight mean?

Suppose you want to add a feature to a product, and this feature requires some cryptographic functionality in the critical path

You start by investigating standards, then rest of literature

If all solutions are too “heavy” – hence they make your product too bloated:

$\Rightarrow$ customers would not enable the feature
$\Rightarrow$ you can forget about implementing it in the first place
$\Rightarrow$ the feature will never see the light of the day!
Constraints – in other words, what does lightweight mean?

Suppose you want to add a feature to a product, and this feature requires some cryptographic functionality in the critical path

You start by investigating standards, then rest of literature

If all solutions are too “heavy” – hence they make your product too bloated:

⇒ customers would not enable the feature
⇒ you can forget about implementing it in the first place
⇒ the feature will never see the light of the day!
Constraints – in other words, what does lightweight mean?

Suppose you want to add a feature to a product, and this feature requires some cryptographic functionality in the critical path.

You start by investigating standards, then rest of literature.

If all solutions are too “heavy” – hence they make your product too bloated:

⇒ customers would not enable the feature
⇒ you can forget about implementing it in the first place
⇒ the feature will never see the light of day!
Constraints – in other words, what does lightweight mean?

Suppose you want to add a feature to a product, and this feature requires some cryptographic functionality in the critical path.

You start by investigating standards, then rest of literature.

If all solutions are too “heavy” – hence they make your product too bloated:

⇒ customers would not enable the feature
⇒ you can forget about implementing it in the first place
⇒ the feature will never see the light of the day!
... what does lightweight mean?

For academia, Lightweight means to build the smallest/fastest/etc cipher of all
If it is broken, well, colleagues will get new papers ;-)  
But in industry we cannot accept (severe) cryptanalysis
For us, Lightweight is Rightweight, in other words, it just (!) has to fit the constraints. If it is not “minimal” but still fits, and contains some extra hedging against attacks, we are ok – leave some wiggle room (hence Tightweight is bad)
But it is still not an easy decision: We may still decide to cancel a feature
Only strong use cases (shall) drive development of new crypto...
... what does lightweight mean?

For academia, Lightweight means to build the smallest/fastest/etc cipher of all.

If it is broken, well, colleagues will get new papers ;-)  

But in industry we cannot accept (severe) cryptanalysis.

For us, Lightweight is Rightweight, in other words, it just (!) has to fit the constraints. If it is not “minimal” but still fits, and contains some extra hedging against attacks, we are ok – leave some wiggle room (hence Tightweight is bad).

But it is still not an easy decision: We may still decide to cancel a feature.

Only strong use cases (shall) drive development of new crypto...
… what does lightweight mean?

For academia, Lightweight means to build the smallest/fastest/etc cipher of all
If it is broken, well, colleagues will get new papers ;-) 

But in industry we cannot accept (severe) cryptanalysis

For us, Lightweight is Rightweight, in other words, it just (!) has to fit the constraints. If it is not “minimal” but still fits, and contains some extra hedging against attacks, we are ok – leave some wiggle room (hence Tightweight is bad)

But it is still not an easy decision: We may still decide to cancel a feature

Only strong use cases (shall) drive development of new crypto…
... what does lightweight mean?

For academia, Lightweight means to build the smallest/fastest/etc cipher of all
If it is broken, well, colleagues will get new papers ;-) 

But in industry we cannot accept (severe) cryptanalysis

For us, Lightweight is Rightweight, in other words, it just (!) has to fit the constraints. If it is not “minimal” but still fits, and contains some extra hedging against attacks, we are ok – leave some wiggle room (hence Tightweight is bad)

But it is still not an easy decision: We may still decide to cancel a feature

Only strong use cases (shall) drive development of new crypto...
... what does lightweight mean?

For academia, Lightweight means to build the smallest/fastest/etc cipher of all

If it is broken, well, colleagues will get new papers ;-)  

But in industry we cannot accept (severe) cryptanalysis

For us, Lightweight is Rightweight, in other words, it just (!) has to fit the constraints. If it is not “minimal” but still fits, and contains some extra hedging against attacks, we are ok – leave some wiggle room (hence Tightweight is bad)

But it is still not an easy decision: We may still decide to cancel a feature

Only strong use cases (shall) drive development of new crypto...
... what does lightweight mean?

For academia, Lightweight means to build the smallest/fastest/etc cipher of all. If it is broken, well, colleagues will get new papers ;-) 

But in industry we cannot accept (severe) cryptanalysis. For us, Lightweight is Rightweight, in other words, it just (!) has to fit the constraints. If it is not “minimal” but still fits, and contains some extra hedging against attacks, we are ok – leave some wiggle room (hence Tightweight is bad). But it is still not an easy decision: We may still decide to cancel a feature.

Only strong use cases (shall) drive development of new crypto...
... what does lightweight mean?

For academia, Lightweight means to build the smallest/fastest/etc cipher of all. If it is broken, well, colleagues will get new papers ;-) 

But in industry we cannot accept (severe) cryptanalysis. For us, Lightweight is Rightweight, in other words, it just (!) has to fit the constraints. If it is not “minimal” but still fits, and contains some extra hedging against attacks, we are ok – leave some wiggle room (hence Tightweight is bad).

But it is still not an easy decision: We may still decide to cancel a feature.

Only strong use cases (shall) drive development of new crypto...
Software Security

Memory Encryption
Software Security

Memory Encryption
Trouble with Software

Our devices contain information that are important to us: Credit card numbers, Pictures, Location...

Same for government / finance sector computers

The OS vendor and HW manufacturer have a reputation to defend

There are already sufficiently many stakeholders with different sets of assets to claim that a security problem exists, and motivate research into mitigations
Trouble with Software

Our devices contain information that are important to us: Credit card numbers, Pictures, Location...

Same for government / finance sector computers

The OS vendor and HW manufacturer have a reputation to defend

There are already sufficiently many stakeholders with different sets of assets to claim that a security problem exists, and motivate research into mitigations
Trouble with Software

Our devices contain information that are important to us: Credit card numbers, Pictures, Location...

Same for government / finance sector computers

The OS vendor and HW manufacturer have a reputation to defend

There are already sufficiently many stakeholders with different sets of assets to claim that a security problem exists, and motivate research into mitigations
Trouble with Software

Our devices contain information that are important to us: Credit card numbers, Pictures, Location...

Same for government / finance sector computers

The OS vendor and HW manufacturer have a reputation to defend

There are already sufficiently many stakeholders with different sets of assets to claim that a security problem exists, and motivate research into mitigations
Return oriented programming is about borrowing code chunks from an existing program or library and executing them instead of executing code on the stack

- Enough of such gadgets will most likely be Turing complete
- Prepare a sequence of stack frames pointing to these gadgets
- Each return instruction will then jump to the next one
- Smash this sequence of stack frames on the stack
- RETURN – This starts a domino effect
Code Reuse Attacks

Return oriented programming is about borrowing *code chunks* from an existing program or library and executing them instead of executing code on the stack.

- Enough of such *gadgets* will most likely be Turing complete
  - Prepare a sequence of stack frames pointing to these gadgets
  - Each return instruction will then jump to the next one
  - Smash this sequence of stack frames on the stack
  - RETURN – This starts a domino effect
Code Reuse Attacks

Return oriented programming is about borrowing code chunks from an existing program or library and executing them instead of executing code on the stack

- Enough of such gadgets will most likely be Turing complete
- Prepare a sequence of stack frames pointing to these gadgets
  - Each return instruction will then jump to the next one
  - Smash this sequence of stack frames on the stack
  - RETURN – This starts a domino effect
Code Reuse Attacks

Return oriented programming is about borrowing code chunks from an existing program or library and executing them instead of executing code on the stack

- Enough of such gadgets will most likely be Turing complete
- Prepare a sequence of stack frames pointing to these gadgets
- Each return instruction will then jump to the next one
  - Smash this sequence of stack frames on the stack
  - RETURN – This starts a domino effect
Code Reuse Attacks

Return oriented programming is about borrowing code chunks from an existing program or library and executing them instead of executing code on the stack.

- Enough of such gadgets will most likely be Turing complete.
- Prepare a sequence of stack frames pointing to these gadgets.
- Each return instruction will then jump to the next one.
- Smash this sequence of stack frames on the stack.
- RETURN – This starts a domino effect.
Code Reuse Attacks

Return oriented programming is about borrowing *code chunks* from an existing program or library and executing them instead of executing code on the stack

- Enough of such *gadgets* will most likely be Turing complete
- Prepare a sequence of stack frames pointing to these gadgets
- Each return instruction will then jump to the next one
- Smash this sequence of stack frames on the stack
- RETURN – This starts a domino effect
Countermeasures

- Various mitigations today, such as ASLR, table based CFI, canaries, pointer/stack mangling, shadow stacks
  
  ... not as widely deployed
  ... can be difficult to integrate
  ... can have non-trivial performance / code size impact
  ... can inhibit debugging

- There have been several proposals for encrypting the stack so far – but all with extremely high performance penalties
Countermeasures

- Various mitigations today, such as ASLR, table based CFI, canaries, pointer/stack mangling, shadow stacks
  - not as widely deployed
  - can be difficult to integrate
  - can have non-trivial performance / code size impact
  - can inhibit debugging

- There have been several proposals for encrypting the stack so far – but all with extremely high performance penalties
Countermeasures

- Various mitigations today, such as ASLR, table based CFI, canaries, pointer/stack mangling, shadow stacks

  ... not as widely deployed
  ... can be difficult to integrate
  ... can have non-trivial performance / code size impact
  ... can inhibit debugging

- There have been several proposals for encrypting the stack so far – but all with extremely high performance penalties
Some of my colleagues at Qualcomm (Can Acar, Arvind Krishnaswamy, and Robert Turner) started to think of solutions to this problem...
Pointer authentication comes to the rescue

- Optional ARMv8.3-A extension
- Detects illicit modification of pointers (and data structures)
  ... can be used to catch ROP, etc
  ... simple to integrate
  ... with minimal code size / performance impact
- Backwards compatible subset
  ... binaries using some features can run on any ARMv8-A CPU (without protection)
  ... so distributions only need one set of binaries
The idea behind pointer authentication 1/4

Observations: you do not need to encrypt addresses

While some code is executing, until it returns or calls back, it is in control

So just verify these addresses in the code itself (google CFI, LLVM CFI)

So, let us use nifty crypto with a nifty name

Let us define a short MAC for pointers

Call it Pointer Authentication Code: PAC
The idea behind pointer authentication 1/4

Observations: you do not need to encrypt addresses

While some code is executing, until it returns or calls back, it is in control

So just verify these addresses in the code itself (google CFI, LLVM CFI)

So, let us use nifty crypto with a nifty name

Let us define a short MAC for pointers

Call it Pointer Authentication Code: PAC
The idea behind pointer authentication 2/4

In ROP you return from functions you did not enter properly

Function prologue

```c
void f(const char *inbuf, int len) {
    char buf[8];

    // insert PAC(LR,key,SP) in unused bits of LR
    // store FP and LR on stack
    // SP += 0x40 (local variables)
}
```

Function epilogue

- SP -= 0x40 (local variables)
- load FP and LR from stack
- verify PAC in LR using SP as “tweak”
- if PAC wrong then panic
- (delete PAC bit field in LR) and RET

The directly higher execution level controls the key
The PAC is inserted into unused bits of the address (more later)
The idea behind pointer authentication 2/4

In ROP you return from functions you did not enter properly

**Function prologue**

```c
void f(const char *inbuf, int len) {
    char buf[8];
    - insert PAC(LR,key,SP) in unused bits of LR
    - store FP and LR on stack
    - SP += 0x40 (local variables)
```

**Function epilogue**

- SP -= 0x40 (local variables)
- load FP and LR from stack
- verify PAC in LR using SP as “tweak”
- if PAC wrong then panic
- (delete PAC bit field in LR) and RET

The directly higher execution level controls the key
The PAC is inserted into unused bits of the address (more later)
The idea behind pointer authentication 3/4

**Forward path**

**Taking an address of a pointer**

```c
int lt(...);
int (*x)(...);
x = lt;
```

- `x0 ← PC`
- `x3 ← x0`
- insert PAC(x3,key,C=context) in x3

**Taking indirect branch**

```c
int (*x)(...);
x(...);
```

- verify the PAC in x3 with tweak C is correct
  - (x3 will be junk if PAC invalid)
- JUMP to x3

The context is a value that depends on the “non connected” parts of the graph of possible calls in the software
The idea behind pointer authentication 3/4

Forward path

Taking an address of a pointer

int lt(...);
int (*x)(...);
x = lt;

- x0 ← PC
- x3 ← x0
- insert PAC(x3,key,C=context) in x3

Taking indirect branch

int (*x)(...);
x(...);

- verify the PAC in x3 with tweak C is correct
  (x3 will be junk if PAC invalid)
- JUMP to x3

The context is a value that depends on the “non connected” parts of the graph of possible calls in the software
The idea behind pointer authentication 4/4

ROP becomes possible if the attacked can guess the PAC; Suppose PAC is 7 bits

Brute force: for 7 bits it is 128 choices

A ROP payload consists of several gadgets

If you have to guess 10 tags, then you must guess 70 bits

If you assume you guessed the first 7 bits correctly, the space for key may be reduced somewhat

For each additional “guessed” PAC choices may be further reduced

Choose primitive that remains robust with a few know inputs
The PAC Instructions
New instructions to sign and authenticate pointers
... against a user-chosen (dynamic) context
... e.g. return address is valid for a given stackframe
... architecture provides mechanism, not policy

Uses a Pointer Authentication Code (PAC)
... authentication metadata stored within pointer
... so no additional space required
Each PAC is derived from:
- A pointer value
- A 64-bit context value
- A 128-bit secret key

PAC algorithm $P$ can be:
- QARMA$^1$
- IMPLEMENTATION DEFINED

Instructions hide the algorithm details

$^1$We’ll talk about it later
Keys

- **Secret** 128-bit value
  - ... inhibit prediction / forging of PACs

- Held in system registers
  - ... can be used, but not read/written at EL0 (userspace)
  - ... limited risk of disclosure / modification

- Several keys:
  - APIAKey, APIBKey (instruction pointers)
  - APDAKey, APDBKey (data pointers)
  - APGAKey (data)
Pointers in AArch64

<table>
<thead>
<tr>
<th>63</th>
<th>56</th>
<th>55</th>
<th>54</th>
<th>VA_SIZE</th>
<th>VA_SIZE-1</th>
</tr>
</thead>
</table>

- address
- reserved
- low/high
- tag/reserved
Pointers in AArch64 (with authentication)

- PAC embedded in reserved pointer bits
  - e.g. 7 bits with 48-bit VA with tagging
  - leaving remaining bits intact

![Diagram of pointer structure with PAC embedded in reserved bits]

© ARM 2018
Pointers in AArch64 (with authentication)

- PAC embedded in reserved pointer bits
  - e.g. 15 bits with 48-bit VA without tagging
  - leaving remaining bits intact

```
 63  56  55  54  VA_SIZE  VA_SIZE-1  0
 PAC | PAC |
```

- Address
- Reserved
- Low/high
- Reserved
Operations: sign

- PAC* instructions **sign** pointers with PACs
- Result is not a usable pointer
Operations: authenticate

- **AUT* instructions authenticate PACs**
- If PAC matches, result is the original pointer
- If PAC doesn’t match, result is an invalid pointer → faults upon use
Operations: strip

- XPAC* instructions strip PACs
- Result is the original pointer
- No authentication is performed
Using the PAC Instructions
ROP vulnerable code

```assembly
stp fp, lr, [sp, #-FRAME_SIZE]!
mov fp, sp

< function body >

ldp fp, lr, [sp], #FRAME_SIZE
ret lr
```
ROP protection

```
pacia lr, sp
stp fp, lr, [sp, #-FRAME_SIZE]!
mov fp, sp

< function body >

ldp fp, lr, [sp], #FRAME_SIZE
autia lr, sp
ret lr
```
ROP protection (backwards compatible – use NOP instruction space)

```
paciasp
stp fp, lr, [sp, #-FRAME_SIZE]!
mov fp, sp

< function body >

ldp fp, lr, [sp], #FRAME_SIZE
autiasp
ret lr
```
Other uses

- Many potential uses / contexts:
  - Locally-scoped pointers / stackframe
  - PLTs / PLT address (dynamic link time)
  - Opaque pointers / logical type, owner

- Architecture provides mechanism, not policy

- Needs careful consideration of reuse attacks
  - Need to avoid signing gadgets
  - Very short gadget chains may still be doable with sufficiently many retries
  - May require distinct keys for distinct purposes / processes
Trouble
The Quest for a Suitable Cryptographic Primitive

Constraints:

- The instructions should be cryptographically strong (or at least not completely breakable) and do not introduce annoying bubbles in the pipeline: Latency is the main constraint
- Area – and thus power – is not a (huge) concern; Efficient SW implementation is not a goal

AES: clearly out of the question, as latency is $\geq 20$ cycles! (Consider the pipelines of ARM cores)

PRINCE: the latency of one instance is ok ($\approx 3$-$5$ pipeline stages). But we needed two 64-bit inputs: the pointer and a context. PRINCE offers only one (the plaintext). All provably secure constructions to get a hash function for $\approx 120$ bits from PRINCE more than doubled the latency.

SIPHASH. Accepts arbitrarily long input. But: even slower...

Back to square one?
The Quest for a Suitable Cryptographic Primitive

Constraints:

- The instructions should be cryptographically strong (or at least not completely breakable) and
do not introduce annoying bubbles in the pipeline: Latency is the main constraint
- Area – and thus power – is not a (huge) concern; Efficient SW implementation is not a goal

AES: clearly out of the question, as latency is $\geq 20$ cycles! (Consider the pipelines of ARM cores)
PRINCE: the latency of one instance is ok ($\approx 3$-5 pipeline stages). But we needed two 64-bit inputs: the pointer and a context. PRINCE offers only one (the plaintext). All provably secure constructions to get a hash function for $\approx 120$ bits from PRINCE more than doubled the latency.
SIPHASH. Accepts arbitrarily long input. But: even slower...

Back to square one?
The Quest for a Suitable Cryptographic Primitive

Constraints:

- The instructions should be cryptographically strong (or at least not completely breakable) and do not introduce annoying bubbles in the pipeline: Latency is the main constraint
- Area – and thus power – is not a (huge) concern; Efficient SW implementation is not a goal

AES: clearly out of the question, as latency is $\geq 20$ cycles! (Consider the pipelines of ARM cores)

PRINCE: the latency of one instance is ok ($\approx 3$-5 pipeline stages). But we needed two 64-bit inputs: the pointer and a context. PRINCE offers only one (the plaintext). All provably secure constructions to get a hash function for $\approx 120$ bits from PRINCE more than doubled the latency.

SIPHASH. Accepts arbitrarily long input. But: even slower...

Back to square one?
The Quest for a Suitable Cryptographic Primitive

Constraints:

- The instructions should be cryptographically strong (or at least not completely breakable) and do not introduce annoying bubbles in the pipeline: Latency is the main constraint.
- Area – and thus power – is not a (huge) concern; Efficient SW implementation is not a goal.

AES: clearly out of the question, as latency is \( \geq 20 \) cycles! (Consider the pipelines of ARM cores).

PRINCE: the latency of one instance is ok (\( \approx 3-5 \) pipeline stages). But we needed two 64-bit inputs: the pointer and a context. PRINCE offers only one (the plaintext). All provably secure constructions to get a hash function for \( \approx 120 \) bits from PRINCE more than doubled the latency.

SIPHASH. Accepts arbitrarily long input. But: even slower...

Back to square one?
The Quest for a Suitable Cryptographic Primitive

Constraints:

- The instructions should be cryptographically strong (or at least not completely breakable) and do not introduce annoying bubbles in the pipeline: Latency is the main constraint.
- Area – and thus power – is not a (huge) concern; Efficient SW implementation is not a goal.

AES: clearly out of the question, as latency is $\geq 20$ cycles! (Consider the pipelines of ARM cores).

PRINCE: the latency of one instance is ok ($\approx 3$-5 pipeline stages). But we needed two 64-bit inputs: the pointer and a context. PRINCE offers only one (the plaintext). All provably secure constructions to get a hash function for $\approx 120$ bits from PRINCE more than doubled the latency.

SIPHASH. Accepts arbitrarily long input. But: even slower...

Back to square one?
The Quest for a Suitable Cryptographic Primitive

Constraints:

- The instructions should be cryptographically strong (or at least not completely breakable) and do not introduce annoying bubbles in the pipeline: Latency is the main constraint
- Area – and thus power – is not a (huge) concern; Efficient SW implementation is not a goal

AES: clearly out of the question, as latency is $\geq 20$ cycles! (Consider the pipelines of ARM cores)

PRINCE: the latency of one instance is ok ($\approx 3$-5 pipeline stages). But we needed two 64-bit inputs: the pointer and a context. PRINCE offers only one (the plaintext). All provably secure constructions to get a hash function for $\approx 120$ bits from PRINCE more than doubled the latency.

SIPHASH. Accepts arbitrarily long input. But: even slower...

Back to square one?
Truncate a Tweakable Block cipher?

Idea: whereas a block cipher has two inputs (plaintext and key), a tweakable block cipher (TBC) has three (plaintext, key, and tweak)

In a TBC the permutation is determined by the key and the tweak, and the cipher must remain secure even if the tweak is known or under adversarial control!

If we truncate an ideal block cipher (a PRP) to at most half of its bits, then it becomes indistinguishable from an ideal hash function – actually, a PRF


Of course we shall deal with a concrete – not ideal! – block cipher, and in fact with a concrete, tweakable, and lightweight (!) block cipher...
Truncate a Tweakeable Block cipher?

Idea: whereas a block cipher has two inputs (plaintext and key), a tweakable block cipher (TBC) has three (plaintext, key, and tweak)

In a TBC the permutation is determined by the key and the tweak, and the cipher must remain secure even if the tweak is known or under adversarial control!

If we truncate an ideal block cipher (a PRP) to at most half of its bits, then it becomes indistinguishable from an ideal hash function – actually, a PRF


Of course we shall deal with a concrete – not ideal! – block cipher, and in fact with a concrete, tweakable, and lightweight (!) block cipher...
Truncate a Tweakeable Block cipher?

Idea: whereas a block cipher has two inputs (plaintext and key), a tweakable block cipher (TBC) has three (plaintext, key, and tweak)

In a TBC the permutation is determined by the key and the tweak, and the cipher must remain secure even if the tweak is known or under adversarial control!

If we truncate an ideal block cipher (a PRP) to at most half of its bits, then it becomes indistinguishable from an ideal hash function – actually, a PRF


Of course we shall deal with a concrete – not ideal! – block cipher, and in fact with a concrete, tweakable, and lightweight (!) block cipher...
The Road to QARMA
Security Requirements

For software security we do not need $2^{128}$ security, and 64-bit blocks are sufficient

Like with PRINCE we ask for time $\times$ data $\geq 2^{128-\varepsilon}$ – so a minimum time $2^{64-\varepsilon}$ to collect the data or to run the attack

After all, in a PQ world, a key search would last $2^{64}$ anyway

For memory encryption we double values, to aim at where the AES-128 currently stays (while AES-256 would be too bloated)
Security Requirements

For software security we do not need $2^{128}$ security, and 64-bit blocks are sufficient.

Like with PRINCE we ask for time $\times$ data $\geq 2^{128-\varepsilon}$ – so a minimum time $2^{64-\varepsilon}$ to collect the data or to run the attack.

After all, in a PQ world, a key search would last $2^{64}$ anyway.

For memory encryption we double values, to aim at where the AES-128 currently stays (while AES-256 would be too bloated).
Security Requirements

For software security we do not need $2^{128}$ security, and 64-bit blocks are sufficient.

Like with PRINCE we ask for time $\times$ data $\geq 2^{128-\varepsilon}$ – so a minimum time $2^{64-\varepsilon}$ to collect the data or to run the attack.

After all, in a PQ world, a key search would last $2^{64}$ anyway.

For memory encryption we double values, to aim at where the AES-128 currently stays (while AES-256 would be too bloated).
I took the train from Munich to Bochum ... and MANTIS was born

The first idea was to take PRINCE...

... add the tweak at each round, whose nibbles are shuffled by a permutation $\gamma$, and adopt/adapt the round function from MIDORI to offset increase of rounds
I took the train from Munich to Bochum ... and MANTIS was born

The first idea was to take PRINCE...

... add the tweak at each round, whose nibbles are shuffled by a permutation $\gamma$, and adopt/adapt the round function from MIDORI to offset increase of rounds
I took the train from Munich to Bochum ... and MANTIS was born

The first idea was to take PRINCE...

... add the tweak at each round, whose nibbles are shuffled by a permutation $\gamma$, and adopt/adapt the round function from MIDORI to offset increase of rounds
I took the train from Munich to Bochum ... and MANTIS was born

The first idea was to take PRINCE...

... add the tweak at each round, whose nibbles are shuffled by a permutation $\gamma$, and adopt/adapt the round function from MIDORI to offset increase of rounds...
I took the train from Munich to Bochum ... and MANTIS was born

Texts / tweak / state = vectors of sixteen 4-bit cells / $4 \times 4$ matrices

$\tau, h =$ Cell Shuffles, $M =$ Involutory Almost MDS $4 \times 4$ matrix, $S =$ S-Box layer

$\tau \circ M \circ S$ related to MIDORI round function – lighter than PRINCE’s to offset the additional rounds
Beyond MANTIS

What happened next is normal in crypto research
First there were doubts about the 4-round SuperBox in the middle,
then some partners disliked the (re)use of MIDORI components
So I had to go back to the drawing board: Boring unless we spice it with mathematics

1. New structure
2. Better diffusion matrices
3. Better S-Boxes (and new heuristics to find them)
4. Provide a 128-bit variant with 256-bit key

Shortly after that, security margins of MANTIS started to erode a bit

Outcome: MANTIS has a new cousin ...
Beyond MANTIS

What happened next is normal in crypto research
First there were doubts about the 4-round SuperBox in the middle, then some partners disliked the (re)use of MIDORI components

So I had to go back to the drawing board: Boring unless we spice it with mathematics

1. New structure
2. Better diffusion matrices
3. Better S-Boxes (and new heuristics to find them)
4. Provide a 128-bit variant with 256-bit key

Shortly after that, security margins of MANTIS started to erode a bit

Outcome: MANTIS has a new cousin ...
Beyond MANTIS

What happened next is normal in crypto research
First there were doubts about the 4-round SuperBox in the middle, then some partners disliked the (re)use of MIDORI components

So I had to go back to the drawing board: Boring unless we spice it with mathematics

1. New structure
2. Better diffusion matrices
3. Better S-Boxes (and new heuristics to find them)
4. Provide a 128-bit variant with 256-bit key

Shortly after that, security margins of MANTIS started to erode a bit

Outcome: MANTIS has a new cousin ...
... a cipher partly designed on the slopes of Mount Carmel ...

Qualcomm

ARM

Authenticator

Q + A + R + M + A

Roberto M. Avanzi

(and it might badly affect my karma)
The Design of QARMA
QARMA has a new Structure

Whitening key derivation is s.t. $w^0 \leftrightarrow w^1$ and $w^0 \leftrightarrow w^0 + w^1$ both 1-1 (orthomorphism)

It is a 3-round, 2-key, alternating-key (non ideal) Even-Mansour scheme

(TD tradeoff may increase from $TD \geq n-\varepsilon$ to $TD \geq 2^{\frac{3}{2}n-\varepsilon}$, however we stick with the more conservative one)
QARMA Encryption (QARMA_r has $2r + 2$ rounds)

Texts / tweak / state = vectors of sixteen 4-bit cells / $4 \times 4$ matrices

$\tau$, $h$ = Cell Shuffles; $M$ = Involutory Almost MDS matrix; $S$ = 16 S-Boxes; $\omega$ = LSFR $\times$ 7
QARMA Encryption (QARMA_r has 2r + 2 rounds)

Texts / tweak / state = vectors of sixteen 4-bit cells / 4 × 4 matrices

τ, h = Cell Shuffles; M = Involutory Almost MDS matrix; S = 16 S-Boxes; ω = LSFR × 7
QARMA Decryption

Texts / tweak / state = vectors of sixteen 4-bit cells / $4 \times 4$ matrices

$\tau, h = \text{Cell Shuffles}; M = \text{Involuntary Almost MDS matrix}; S = 16 \text{ S-Boxes}; \omega = \text{LSFR} \times 7$

Decrypt with: $k^0 \leftrightarrow k^0 \oplus \alpha$, swap $w^0$ and $w^1$
QARMA Decryption

Texts / tweak / state = vectors of sixteen 4-bit cells / 4 × 4 matrices
τ, h = Cell Shuffles; M = Involutory Almost MDS matrix; S = 16 S-Boxes; ω = LSFR × 7
Decrypt with: $k^0 \mapsto k^0 \oplus \alpha$, swap $w^0$ and $w^1$, replace $k^1 \mapsto M \cdot k^1$
Texts / tweak / state = vectors of sixteen 4-bit cells / 4 x 4 matrices
\( \tau, h = \text{Cell Shuffles}; M = \text{Involutory Almost MDS matrix}; S = 16 \text{ S-Boxes}; \omega = \text{LSFR} \times 7 \)
Decrypt with: \( k^0 \mapsto k^0 \oplus \alpha, \) swap \( w^0 \) and \( w^1, \) replace \( k^1 \mapsto M \cdot k^1 \)
Impact of new central construction

Use of whitening key(s) instead of core key thwarts reflection attacks
Non involutory, keyed Pseudo-Reflector also makes reflection attacks more difficult
\( \tau \) and \( \bar{\tau} \) around it improve diffusion, kill 4-round SuperBox
Various Improvements

No need to provide (many) details. There’s a paper! (FSE 2017)

- New diffusion matrix
- New S-Boxes with better cryptographic properties (and a bit slower as well, TANSTAAFL)

So far (I am keeping my fingers crossed) these components are keeping the type of attacks on (reduced round) MANTIS away. And:

- Also a 128-bit version!
The Design of QARMA

The Diffusion Matrix
MIDORI and MANTIS: Almost MDS Matrix $circ(0, 1, 1, 1)$

Represent state as a matrix of sixteen $m$-bit cells:

$$IS = \begin{pmatrix}
  s_0 & s_1 & s_2 & s_3 \\
  s_4 & s_5 & s_6 & s_7 \\
  s_8 & s_9 & s_{10} & s_{11} \\
  s_{12} & s_{13} & s_{14} & s_{15}
\end{pmatrix}$$

Consider matrices that operates on columns (i.e. on the left) of form

$$M = circ(0, 1, 1, 1) = \begin{pmatrix}
  0 & 1 & 1 & 1 \\
  1 & 0 & 1 & 1 \\
  1 & 1 & 0 & 1 \\
  1 & 1 & 1 & 0
\end{pmatrix} \quad (*)$$
The **MIDORI** and **MANTIS** Matrix

\[
\begin{pmatrix}
0 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 \\
1 & 1 & 1 & 0 \\
\end{pmatrix}
\times
\begin{pmatrix}
v_0 \\
v_1 \\
\vdots \\
\vdots \\
\end{pmatrix}
= 
\begin{pmatrix}
v_1 \oplus \cdots \\
v_0 \oplus \cdots \\
(\nu_0 \oplus \nu_1) \oplus \cdots \\
(\nu_0 \oplus \nu_1) \oplus \cdots \\
\end{pmatrix}
\]

Two S-Boxes copied, two additions identical – characteristics propagate unchanged and easily controlled.
QARMA: Almost MDS Matrix over $R = \mathbb{F}_2[X]/(X^m + 1) = R[\rho]$

Consider matrices that operates on columns (i.e. on the left) of form

$$M = \text{circ}(0, \rho^a, \rho^b, \rho^c) = \begin{pmatrix}
0 & \rho^a & \rho^b & \rho^c \\
\rho^c & 0 & \rho^a & \rho^b \\
\rho^b & \rho^c & 0 & \rho^a \\
\rho^a & \rho^b & \rho^c & 0
\end{pmatrix}$$

Note that $\rho^a$ means circular rotation of $a$ places to the left.

*Remark:* Multiplication by $\rho^a$ in $\mathbb{F}[\rho]$ is cheaper than any multiplication in $\mathbb{F}_2^m \setminus \{0, 1\}$. In fact, these matrices are as expensive (area, latency) as the $\{0, 1\}$-matrices (well, there is some additional wire crossing).

Depth = 2 XOR, Area between $6m$ and $8m$. 

© ARM 2018
It’s a ring (that is not a field, it has zero divisors)
Classification (Here be Theorems!)

Let \( R = \mathbb{F}_2[\rho] \) be the quotient ring \( \mathbb{F}_2[X]/(X^m + 1) \), \( m \geq 2 \).

Consider matrices

\[
M = \text{circ}(0, \rho^a, \rho^b, \rho^c) = \begin{pmatrix}
0 & \rho^a & \rho^b & \rho^c \\
\rho^c & 0 & \rho^a & \rho^b \\
\rho^b & \rho^c & 0 & \rho^a \\
\rho^a & \rho^b & \rho^c & 0
\end{pmatrix}
\]

We classify:

1. The matrices \( M \) which are invertible.
2. The matrices \( M \) which have inverse of the same form.
3. The involutory ones (\( M = M^{-1} \)).
Choice of Matrices for QARMA

Ideally pick $M = \text{circ}(0, \rho^a, \rho^b, \rho^c)$ such that all $x - y$ with $x \neq y \in \{a, b, c\}$ different mod $m$, so that common sums of two S-Box outputs in same result column have different relative shifts.

Only possible for $m = 8$, partially for $m = 4$. Examples with $m = 8$:

\[
\begin{pmatrix}
0 & \rho & \rho^4 & \rho^5 \\
\rho^5 & 0 & \rho & \rho^4 \\
\rho^4 & \rho^5 & 0 & \rho \\
\rho & \rho^4 & \rho^5 & 0
\end{pmatrix}
\times
\begin{pmatrix}
v_0 \\
v_1 \\
\vdots \\
\vdots
\end{pmatrix}
= 
\begin{pmatrix}
(v_1 \ll 1) \oplus \cdots \\
(v_0 \ll 5) \oplus \cdots \\
(v_0 \ll 4) \oplus (v_1 \ll 5) \oplus \cdots \\
(v_0 \ll 1) \oplus (v_1 \ll 4) \oplus \cdots
\end{pmatrix}
\Delta = 1
\Delta = 3
\]

But at least common rotation different...

Then next S-Box layer more likely to disrupt characteristics (linear, differential, also higher order, including saturation), or at least to avoid the same to be copy-and-pasted onto two output boxes.

Select values heuristically, minimising fixpoints (also under simple linear maps) over 1.5 rounds.
Choice of Matrices for QARMA

Ideally pick $M = \text{circ}(0, \rho^a, \rho^b, \rho^c)$ such that all $x - y$ with $x \neq y \in \{a, b, c\}$ different mod $m$, so that common sums of two S-Box outputs in same result column have different relative shifts.

Only possible for $m = 8$, partially for $m = 4$. Examples with $m = 8$:

$$
\begin{pmatrix}
0 & \rho & \rho^4 & \rho^5 \\
\rho^5 & 0 & \rho & \rho^4 \\
\rho^4 & \rho^5 & 0 & \rho \\
\rho & \rho^4 & \rho^5 & 0
\end{pmatrix}
\begin{pmatrix}
v_0 \\
v_1 \\
v_2 \\
\ldots
\end{pmatrix}
= 
\begin{pmatrix}
(v_0 \ll 5) \oplus (v_2 \ll 1) \\
\ldots \\
(v_0 \ll 1) \oplus (v_2 \ll 5) \\
\ldots
\end{pmatrix}
\Delta = 4

But at least common rotation different...

Then next S-Box layer more likely to disrupt characteristics (linear, differential, also higher order, including saturation), or at least to avoid the same to be copy-and-pasted onto two output boxes.

Select values heuristically, minimising fixpoints (also under simple linear maps) over 1.5 rounds.
Choice of Matrices for QARMA

Ideally pick $M = \text{circ}(0, \rho^a, \rho^b, \rho^c)$ such that all $x - y$ with $x \neq y \in \{a, b, c\}$ different mod $m$, so that common sums of two S-Box outputs in same result column have different relative shifts.

Only possible for $m = 8$, partially for $m = 4$. Examples with $m = 8$:

$$
\begin{pmatrix}
0 & \rho & \rho^4 & \rho^5 \\
\rho^5 & 0 & \rho & \rho^4 \\
\rho^4 & \rho^5 & 0 & \rho \\
\rho & \rho^4 & \rho^5 & 0
\end{pmatrix}
\times
\begin{pmatrix}
v_0 \\
\cdots \\
v_2 \\
\cdots
\end{pmatrix}
= \begin{pmatrix}
(v_0 \ll 5) \oplus (v_2 \ll 1) \\
\cdots \\
(v_0 \ll 4) \oplus \cdots \\
(v_0 \ll 1) \oplus (v_2 \ll 5) \oplus \cdots
\end{pmatrix}
\Delta = 4
$$

But at least common rotation different...

Then next S-Box layer more likely to disrupt characteristics (linear, differential, also higher order, including saturation), or at least to avoid the same to be copy-and-pasted onto two output boxes. Select values heuristically, minimising fixpoints (also under simple linear maps) over 1.5 rounds.
Lower bounds on the number of active S-Boxes

To make search for linear/differential characteristics more difficult, I define new matrices.

### Class I Diffusion Matrices

<table>
<thead>
<tr>
<th>$r$</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>32</td>
<td>50</td>
<td>64</td>
<td>$\geq$ 70</td>
<td>$\geq$ 76</td>
<td>$\geq$ 82</td>
<td>$\geq$ 100</td>
<td>$\geq$ 114</td>
<td>$\geq$ 124</td>
</tr>
<tr>
<td>Rel. Tweak</td>
<td>16</td>
<td>24</td>
<td>34</td>
<td>44</td>
<td>52</td>
<td>60</td>
<td>$\geq$ 64</td>
<td>$\geq$ 70</td>
<td>$\geq$ 78</td>
</tr>
</tbody>
</table>

### Class II Diffusion Matrices

<table>
<thead>
<tr>
<th>$r$</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>32</td>
<td>50</td>
<td>60</td>
<td>$\geq$ 68</td>
<td>$\geq$ 74</td>
<td>$\geq$ 82</td>
<td>$\geq$ 98</td>
<td>$\geq$ 112</td>
<td>$\geq$ 120</td>
</tr>
<tr>
<td>Rel. Tweak</td>
<td>14</td>
<td>24</td>
<td>30</td>
<td>42</td>
<td>48</td>
<td>58</td>
<td>$\geq$ 62</td>
<td>$\geq$ 68</td>
<td>$\geq$ 76</td>
</tr>
</tbody>
</table>

### Comparison: MANTIS (Class I Diffusion Matrix and no ShuffleCells in Reflector)

<table>
<thead>
<tr>
<th>$r$</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>32</td>
<td>46</td>
<td>60</td>
<td>$\geq$ 68</td>
<td>$\geq$ 76</td>
<td>$\geq$ 82</td>
</tr>
<tr>
<td>Rel. Tweak</td>
<td>12</td>
<td>20</td>
<td>34</td>
<td>44</td>
<td>50</td>
<td>56</td>
</tr>
</tbody>
</table>
Choice

Criteria:

- Effectiveness to disrupt copy-and-past of differential characteristics
- Number of active S-Boxes (this reduces to the classes seen in previous slide)
- Number of fixed points (not necessarily minimal, but close)
- Involutory – better heuristics for resistance against invariant subspace attacks

For QARMA-64

\[ M = \tilde{M} = \text{circ}(0, \rho, \rho^2, \rho) \]

For QARMA-128

\[ M = \tilde{M} = \text{circ}(0, \rho, \rho^4, \rho^5) \]
The Design of QARMA

The S-Boxes
**S-Box Search Heuristics**

Most important property in our context: total latency

Logic synthesis of a circuit is expensive and slow. Cannot synthesise billions of S-boxes.

Idea: apply crude heuristics based on Quine-McCluskey to bound the depth of individual output bits. Take max. Minimise it over all searched S-Boxes.

Use variant of Prissette’s algorithm to enumerate involutions with a predetermined subset of fixed points.
S-Box Search Heuristics

Most important property in our context: total latency

Logic synthesis of a circuit is expensive and slow. Cannot synthesise billions of S-boxes.

Idea: apply crude heuristics based on Quine-McCluskey to bound the depth of individual output bits. Take max. Minimise it over all searched S-Boxes.

Use variant of Prissette’s algorithm to enumerate involutions with a predetermined subset of fixed points.
S-Box Search Heuristics

Three S-Boxes selected

1. One with same parameters and depth as MIDORI’s, but slightly larger area, only 2 fixed points (curiosity: all such S-Boxes have 2 fixed points whose values have Hamming distance 1).

2. Similar one but such that one bit input differences always give same number of $n$-bit output differences for each $n = 1, 2, 3, 4$ – for each fixed input bit.

3. The original PRINCE S-Box.
### The Three S-Boxes (Using new Search Heuristics)

<table>
<thead>
<tr>
<th>S-Box</th>
<th>MIDORI Direct</th>
<th>MIDORI Inverse</th>
<th>PRINCE Direct</th>
<th>PRINCE Inverse</th>
<th>QARMA $\sigma_0$ Direct</th>
<th>QARMA $\sigma_0$ Inverse</th>
<th>QARMA $\sigma_1$ Direct</th>
<th>QARMA $\sigma_1$ Inverse</th>
<th>QARMA $\sigma_2$ Direct</th>
<th>QARMA $\sigma_2$ Inverse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. prob. of a differential</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
</tr>
<tr>
<td># with max. probability</td>
<td>24</td>
<td>15</td>
<td>15</td>
<td>18</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Max. bias of a lin. approx.</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
</tr>
<tr>
<td># with max. bias</td>
<td>36</td>
<td>30</td>
<td>30</td>
<td>32</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>Algebraic Degree</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td># components of deg 3, 2</td>
<td>12, 3</td>
<td>15, 0</td>
<td>15, 0</td>
<td>14, 1</td>
<td>15, 0</td>
<td>15, 0</td>
<td>15, 0</td>
<td>15, 0</td>
<td>15, 0</td>
<td>15, 0</td>
</tr>
<tr>
<td>Fixed Points</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Minimal depth (GE)</td>
<td>3.5</td>
<td>5</td>
<td>4.5</td>
<td>3.5</td>
<td>4</td>
<td>4.5</td>
<td>4</td>
<td>4.5</td>
<td>4</td>
<td>4.5</td>
</tr>
<tr>
<td>Minimal area (GE)</td>
<td>12.8</td>
<td>20.2</td>
<td>19</td>
<td>14.17</td>
<td>16.5</td>
<td>20.2</td>
<td>19</td>
<td>20.2</td>
<td>19</td>
<td>20.2</td>
</tr>
</tbody>
</table>
### The Three S-Boxes (Using new Search Heuristics)

<table>
<thead>
<tr>
<th>S-Box</th>
<th>MIDORI Direct</th>
<th>MIDORI Inverse</th>
<th>PRINCE Direct</th>
<th>PRINCE Inverse</th>
<th>QARMA $\sigma_0$</th>
<th>QARMA $\sigma_1$</th>
<th>QARMA $\sigma_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. prob. of a differential</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
</tr>
<tr>
<td># with max. probability</td>
<td>24</td>
<td>15</td>
<td>15</td>
<td>18</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Max. bias of a lin. approx.</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
</tr>
<tr>
<td># with max. bias</td>
<td>36</td>
<td>30</td>
<td>30</td>
<td>32</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>Algebraic Degree</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td># components of deg 3</td>
<td>12, 3</td>
<td>15, 0</td>
<td>15, 0</td>
<td>14, 1</td>
<td>15, 0</td>
<td>15, 0</td>
<td>15, 0</td>
</tr>
<tr>
<td>Fixed Points</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Minimal depth (GE)</td>
<td>3.5</td>
<td>5</td>
<td>4.5</td>
<td>3.5</td>
<td>4</td>
<td>4.5</td>
<td>4</td>
</tr>
<tr>
<td>Minimal area (GE)</td>
<td>12.8</td>
<td>20.2</td>
<td>19</td>
<td>14.17</td>
<td>16.5</td>
<td>20.2</td>
<td>19</td>
</tr>
</tbody>
</table>

$\sigma_0$ is similar to MIDORI’s S-Box but is has better cryptographic properties (all parameters that can be improved are improved), same latency, and slightly larger area.
### The Three S-Boxes (Using new Search Heuristics)

<table>
<thead>
<tr>
<th>S-Box</th>
<th>MIDORI</th>
<th>PRINCE</th>
<th>QARMA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Direct</td>
<td>Inverse</td>
<td>(\sigma_0)</td>
</tr>
<tr>
<td>Max. prob. of a differential</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
</tr>
<tr>
<td># with max. probability</td>
<td>24</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Max. bias of a lin. approx.</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
</tr>
<tr>
<td># with max. bias</td>
<td>36</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>Algebraic Degree</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td># components of deg 3, 2</td>
<td>12, 3</td>
<td>15, 0</td>
<td>15, 0</td>
</tr>
<tr>
<td>Fixed Points</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Minimal depth (GE)</td>
<td>3.5</td>
<td>5</td>
<td>4.5</td>
</tr>
<tr>
<td>Minimal area (GE)</td>
<td>12.8</td>
<td>20.2</td>
<td>19</td>
</tr>
</tbody>
</table>

\(\sigma_1\) is optimal and involutory, and has properties that may make side channel attacks more difficult
The Three S-Boxes (Using new Search Heuristics)

<table>
<thead>
<tr>
<th>S-Box</th>
<th>MIDORI</th>
<th>PRINCE</th>
<th>QARMA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Direct</td>
<td>Inverse</td>
<td>Direct</td>
</tr>
<tr>
<td>Max. prob. of a differential</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
</tr>
<tr>
<td># with max. probability</td>
<td>24</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Max. bias of a lin. approx.</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
</tr>
<tr>
<td># with max. bias</td>
<td>36</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>Algebraic Degree</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td># components of deg 3, 2</td>
<td>12, 3</td>
<td>15, 0</td>
<td>15, 0</td>
</tr>
<tr>
<td>Fixed Points</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Minimal depth (GE)</td>
<td>3.5</td>
<td>5</td>
<td>4.5</td>
</tr>
<tr>
<td>Minimal area (GE)</td>
<td>12.8</td>
<td>20.2</td>
<td>19</td>
</tr>
</tbody>
</table>

$\sigma_2$ comes from the PRINCE selection
The Design of QARMA

The 128-bit Variant with a 256-bit Key
The 8-bit S-Box for QARMA-128
The 8-bit S-Box for QARMA-128
The 8-bit S-Box for QARMA-128
The 8-bit S-Box for QARMA-128
The 8-bit S-Box for QARMA-128
The 8-bit S-Box for QARMA-128
The 8-bit S-Box for QARMA-128
The Design of QARMA

Cryptanalysis
Considered attacks (designing block ciphers is horrible, horrible)

- Linear and differential cryptanalysis (MILP models, following Beierle)
- —, under related tweak model (MILP models, following Beierle)
- Reflection Attacks
- Generic attacks on Even-Mansour schemes (new: follows from structure)
- Slide attacks (new: follows from round heterogeneity)
- Meet-in-the-middle attacks (following MIDORI/MANTIS)
- Invariant subspace attacks (new: heuristic arguments)
- Algebraic cryptanalysis (new: equations and variables counting, degree growth)
- Impossible diff. & zero corr. linear cryptanalysis (new: following Sun et al. EC ’16)
- Higher order differential cryptanalysis (boomerang, integral) (following MIDORI/MANTIS)
- And more...
Algebraic Attacks

Using [BCC11], we have following lower bounds for the degree of QARMA as a Boolean function:

<table>
<thead>
<tr>
<th>Rounds</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>QARMA-64</td>
<td>3</td>
<td>9</td>
<td>27</td>
<td>51</td>
<td>59</td>
<td>62</td>
<td>63</td>
<td>63</td>
</tr>
<tr>
<td>QARMA-128</td>
<td>3</td>
<td>9</td>
<td>27</td>
<td>81</td>
<td>112</td>
<td>122</td>
<td>126</td>
<td>127</td>
</tr>
</tbody>
</table>

This means that with \( 2r + 2 \) rounds \( (r = 7, 11) \) the degree has room to grow and get maximal.

Also, the size of the MQ system is:

<table>
<thead>
<tr>
<th>Equations</th>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>QARMA_{7}-64</td>
<td>5376</td>
</tr>
<tr>
<td>MANTIS_{7}</td>
<td>5376</td>
</tr>
<tr>
<td>PRINCE</td>
<td>4032</td>
</tr>
<tr>
<td>QARMA_{11}-128</td>
<td>16128</td>
</tr>
<tr>
<td>AES-128</td>
<td>6400</td>
</tr>
<tr>
<td>AES-256</td>
<td>8960</td>
</tr>
</tbody>
</table>

So we do not worry.
Invariant Subspace Attacks

These are subtle attacks and focus of very recent research.

Suppose there is a vector space $\mathcal{U}$, two vectors $u, v$ s.t. $F_i(u + \mathcal{U}) \subseteq v + \mathcal{V}$ (hence $=$) and $c_i \in z + \mathcal{U}$ $\forall i$.

If $k \in (u + v + z) + \mathcal{V}$ and $P \in u + \mathcal{U}$, then $C \in v + \mathcal{U}$. The attacker checks whether $k \in (u + v + z) + \mathcal{U}$ by testing some $(P, C)$ pairs, then search $k$ in $(u + v + z) + \mathcal{V}$ – sometimes much faster than brute force.

If $\mathcal{U}$ very small, these weak keys are rare.

Observe that in our case we have $\mathcal{U} = \mathcal{U}_T$, where $\mathcal{U}_T$ depends on $t$ and the condition is $c_i \oplus T_i \in z + \mathcal{U}$ – This makes attacks more difficult as well, esp. if the cipher is used with the “tweak masking” of our minimal encryption mode. However, it is very difficult to verify whether this type of attacks cannot happen, but we have a strong heuristic argument.
Invariant Subspace Attacks

These are subtle attacks and focus of very recent research.

\[
\begin{array}{cccccccc}
 & k & & c_0 & & & & & k \\
 & & & \oplus & & & & \oplus & \\
 & & & F_0 & & & & \oplus & \\
 & & & & & & & k & \\
 & & & & & & \oplus & & \\
 & & & & & & F_1 & & \\
 & & & & k & & & \oplus & \\
 & & & & & & F_2 & & \\
 & & & & k & & & \oplus & \\
 & & & & & & F_3 & & \\
 & & & & k & & & \oplus & \\
 & & & & & & F_4 & & \\
 & & & & k & & & \oplus & \\
 & & & & & & C & & \\
\end{array}
\]

Suppose there is a vector space \( \mathcal{U} \), two vectors \( u, v \) s.t. \( F_i(u + \mathcal{U}) \subseteq v + \mathcal{V} \) (hence \( = \)) and \( c_i \in z + \mathcal{U} \ \forall i \).

If \( k \in (u + v + z) + \mathcal{V} \) and \( P \in u + \mathcal{U} \), then \( C \in v + \mathcal{U} \). The attacker checks whether \( k \in (u + v + z) + \mathcal{U} \) by testing some \((P, C)\) pairs, then search \( k \) in \((u + v + z) + \mathcal{V}\) – sometimes much faster than brute force.

If \( \mathcal{U} \) very small, these weak keys are rare.

Observe that in our case we have \( \mathcal{U} = \mathcal{U}_T \), where \( \mathcal{U}_T \) depends on \( t \) and the condition is \( c_i \oplus T_i \in z + \mathcal{U} \) – This makes attacks more difficult as well, esp. if the cipher is used with the “tweak masking” of our minimal encryption mode. However, it is very difficult to verify whether this type of attacks cannot happen, but we have a strong heuristic argument.
Invariant Subspace Attacks

These are subtle attacks and focus of very recent research.

Suppose there is a vector space $\mathcal{U}$, two vectors $u, v$ s.t. $F_i(u + \mathcal{U}) \subseteq v + \mathcal{V}$ (hence $\subseteq$) and $c_i \in z + \mathcal{U}$ $\forall i$.

If $k \in (u + v + z) + \mathcal{V}$ and $P \in u + \mathcal{U}$, then $C \in v + \mathcal{U}$. The attacker checks whether $k \in (u + v + z) + \mathcal{U}$ by testing some $(P, C)$ pairs, then search $k$ in $(u + v + z) + \mathcal{V}$ – sometimes much faster than brute force.

If $\mathcal{U}$ very small, these weak keys are rare.

Observe that in our case we have $\mathcal{U} = \mathcal{U}_T$, where $\mathcal{U}_T$ depends on $t$ and the condition is $c_i \oplus T_i \in z + \mathcal{U}$ – This makes attacks more difficult as well, esp. if the cipher is used with the “tweak masking” of our minimal encryption mode. However, it is very difficult to verify whether this type of attacks cannot happen, but we have a strong heuristic argument.
Invariant Subspace Attacks

These are subtle attacks and focus of very recent research.

Suppose there is a vector space $\mathcal{U}$, two vectors $u, v$ s.t. $F_i(u + \mathcal{U}) \subseteq v + V$ (hence $=$) and $c_i \in z + \mathcal{U}$ $\forall i$.

If $k \in (u + v + z) + V$ and $P \in u + \mathcal{U}$, then $C \in v + \mathcal{U}$. The attacker checks whether $k \in (u + v + z) + \mathcal{U}$ by testing some $(P, C)$ pairs, then search $k$ in $(u + v + z) + V$ – sometimes much faster than brute force.

If $\mathcal{U}$ very small, these weak keys are rare.

Observe that in our case we have $\mathcal{U} = \mathcal{U}_T$, where $\mathcal{U}_T$ depends on $t$ and the condition is $c_i \oplus T_i \in z + \mathcal{U}$ – This makes attacks more difficult as well, esp. if the cipher is used with the “tweak masking” of our minimal encryption mode. However, it is very difficult to verify whether this type of attacks cannot happen, but we have a strong heuristic argument.
Invariant Subspace Attacks

Observe that if $M = Q$ and involutory, then any invariant subspace must be invariant under $\tau$, $M$ and $S$ and contain the differences of round constants (plus tweaks) – because of particular variety of rounds.

So construct a $\mathcal{V}$ by taking the various $(c_i + T_i) + (c_j + T_j)$ and $\alpha$, repeatedly applying $\tau$ and $M$.

Compute dimension of $\mathcal{V}$ - range and average for millions of random tweaks.

<table>
<thead>
<tr>
<th></th>
<th>$r$</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>QARMA-64</td>
<td>Average</td>
<td>54.41</td>
<td>60.32</td>
<td>62.08</td>
<td>63.02</td>
<td>63.51</td>
</tr>
<tr>
<td></td>
<td>Range</td>
<td>[41..58]</td>
<td>[48..62]</td>
<td>[52..63]</td>
<td>[55..64]</td>
<td>[58..64]</td>
</tr>
<tr>
<td>MANTIS</td>
<td>Average</td>
<td>39.59</td>
<td>46.92</td>
<td>52.93</td>
<td>55.37</td>
<td>57.31</td>
</tr>
<tr>
<td></td>
<td>Range</td>
<td>[32..41]</td>
<td>[38..48]</td>
<td>[46..54]</td>
<td>[50..56]</td>
<td>[51..58]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$r$</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>QARMA-128</td>
<td>Average</td>
<td>122.17</td>
<td>123.61</td>
<td>124.72</td>
<td>125.69</td>
<td>126.51</td>
<td>127.15</td>
</tr>
<tr>
<td></td>
<td>Range</td>
<td>[115..123]</td>
<td>[117..124]</td>
<td>[120..125]</td>
<td>[121..126]</td>
<td>[122..127]</td>
<td>[124..128]</td>
</tr>
<tr>
<td>QARMA-128 with MIDORI matrix</td>
<td>Average</td>
<td>82.43</td>
<td>92.17</td>
<td>99.61</td>
<td>104.17</td>
<td>107.17</td>
<td>109.58</td>
</tr>
<tr>
<td></td>
<td>Range</td>
<td>[73..84]</td>
<td>[80..94]</td>
<td>[84..104]</td>
<td>[94..106]</td>
<td>[98..108]</td>
<td>[101..110]</td>
</tr>
</tbody>
</table>

If we add also the choice of S-box we always get the full space or codimension 1 (rare).
Current Cryptanalysis Status (no need to read the slide during the presentation!)

- Rui Zong and Xiaoyang Dong: *Meet-in-the-Middle Attack on QARMA Block Cipher*
  10 rounds of QARMA-64 and QARMA-128, with external whitening removed (but keeping the internal two additions) faster than brute force, with very high memory requirements

- Rui Zong, Xiaoyang Dong and Xiaoyun Wang: *MILP-Aided Related-Tweak/Key Impossible Differential Attack and Its applications to QARMA, Joltik-BC*
  11 rounds of QARMA using $2^{61}$ chosen plaintexts and in time $2^{64.4}$

- Dong Yang, Wen-feng Qi, and Hua-jin Chen: *Impossible Differential Attack on QARMA Family of Block Ciphers*
  10 rounds of QARMA-64 (QARMA-128) in time $2^{119.3}$ (resp. $2^{237.3}$) encryptions, $2^{61}$ (resp. $2^{122}$) chosen plaintexts and $2^{72.17}$ (resp. $2^{144.17}$) text blocks of space

- Ongoing efforts by Maria Eichlseder
  Attacks on MANTIS$_5$ and MANTIS$_6$, so far they do not extend to QARMA (minimum recommended for non-PAC application QARMA$_7$)

- Unbalanced 4 forward + 8 backward rounds (to be verified, confidential)
**Implementation (64-bit Block Version – 7nm FinFet)**

<table>
<thead>
<tr>
<th>Cipher</th>
<th>Minimum Area</th>
<th>Minimum Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Delay (ns)</td>
<td>Area (GE)</td>
</tr>
<tr>
<td>QARMA7-64-σ1</td>
<td>6.23</td>
<td>18362</td>
</tr>
<tr>
<td>MANTIS7</td>
<td>5.85</td>
<td>15831</td>
</tr>
<tr>
<td>PRINCE</td>
<td>4.07</td>
<td>8702</td>
</tr>
<tr>
<td>Mult. in $\mathbb{F}_{2^{64}}$</td>
<td>1.05</td>
<td>13083</td>
</tr>
</tbody>
</table>
Act 2, Scene 4

Memory Protection
Trouble in memory encryption land: the example of XEX

Problem: first block latency is double than that of a single block cipher instance!
Trouble in mem enc land: XEX with Polynomial Authentication
Trouble in mem enc land: XEX with Encrypted Checksum (OCB)
A TBC saves the day again

If latency of the TBC is less than twice the “classical” block cipher – profit!
A TBC saves the day again – Usage
A TBC saves the day again – Usage Details
# Implementation (128-bit Block Version – 7nm FinFet)

<table>
<thead>
<tr>
<th>Targeting</th>
<th>Minimum Area</th>
<th>Minimum Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Delay</td>
<td>Area</td>
</tr>
<tr>
<td>Cipher</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QARMA_{11}-128-σ₁</td>
<td>8.88</td>
<td>53872</td>
</tr>
<tr>
<td>AES-128, pipelined*</td>
<td>15.67</td>
<td>71164</td>
</tr>
<tr>
<td>AES-256, pipelined*</td>
<td>21.99</td>
<td>101128</td>
</tr>
<tr>
<td>Mult. in ( \mathbb{F}_{2^{128}} )</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

* Note: The latency of one full AES round is 1.58 ns

Compare \( 2 \times \text{AES} \) plus one GFMULT to \( 1 \times \text{QARMA-128} \)
Epilogue
When to roll your own crypto

- As a rule of thumb, never
- Really, never
- But sometimes you want to introduce one or more features for which there is no suitable standard / widely studied primitive around that fits your constraints – or somebody else has it but has patented it, and you do not want to license it – always do your cost analysis
- Be sure you can design something that fits your constraints for one of the “weaker” applications while still overdesigning aiming at the applications that require higher security
- Deploy it for the lower security applications – which can tolerate some cryptanalysis
- If it survives a few years, start thinking at the other applications, possibly tweak
- An ARM motto is “we, not I” – if you want to dare, pool with other experts …
- Project QARMAGEDDON!
Sources and Acknowledgements

This lesson contains recycled material:

- Mark Rutland’s ARM beamer style
- The definition of security that Alex W. Dent taught me while I was at Qualcomm
- Can Acar, Robert Turner, and Arvind Krishnaswamy for the CFI instructions
- Some slides from my talk at the 2014 Lightweight Crypto Day in Haifa
- Some slides from my talk at the 2015 Lightweight Crypto Day in Haifa
- Some slides from my talk at the 2016 Lightweight Crypto Day in Haifa
- Part of my presentation on QARMA given at FSE 2017 in Tokyo
- A good chunk of Mark Rutland’s talk “ARMv8.3 Pointer Authentication” given at the Linux Security Summit September 14, 2017
- … and QARMA is based on MANTIS, which in turn is joint work with a lot of great folks, especially those in Bochum
Questions?