Data and communication protection in reconfigurable embedded systems

Jérémie Crenne, Pascal Cotret, Guy Gogniat, Russel Tessier, Jean-Philippe Diguet

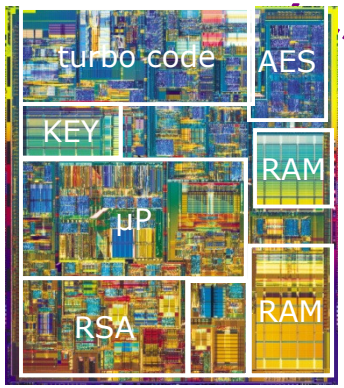
Outline

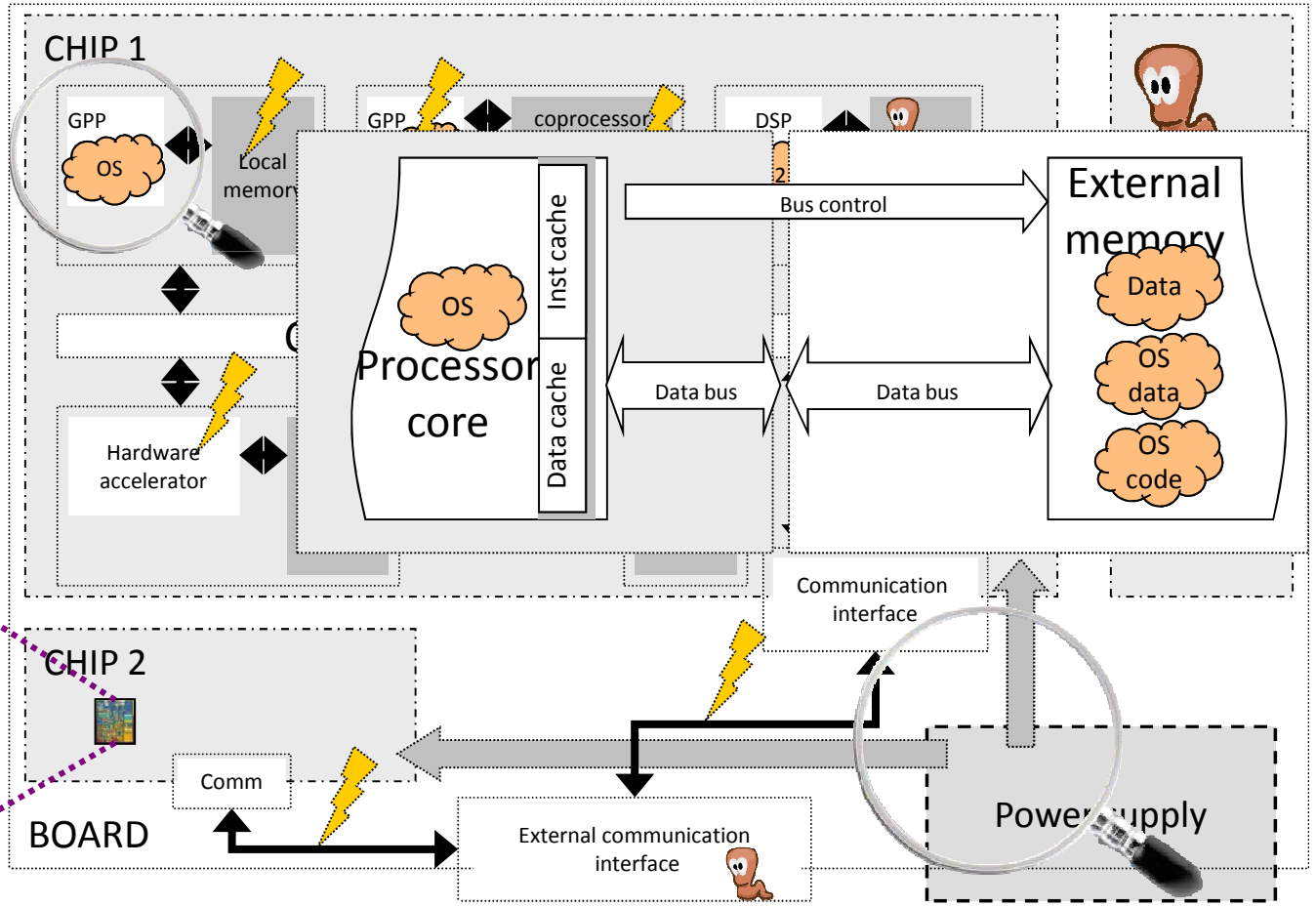
1) Global picture
2) Memory protection
3) Boot protection
4) Experimental setup & results
5) Communication protection: first ideas
6) Conclusion

Embedded Systems & potential attacks

- Example of embedded system architecture

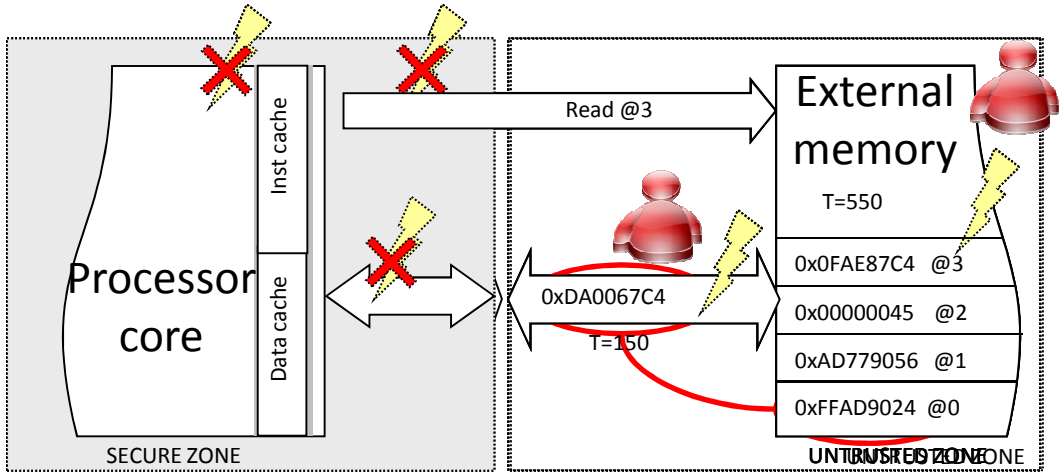
- **Threats**
  - Virus/Worms
  - Reverse engineering
  - Fault injection
  - Memory modification
  - Bus modification
  - Side channel
  - Bus probing





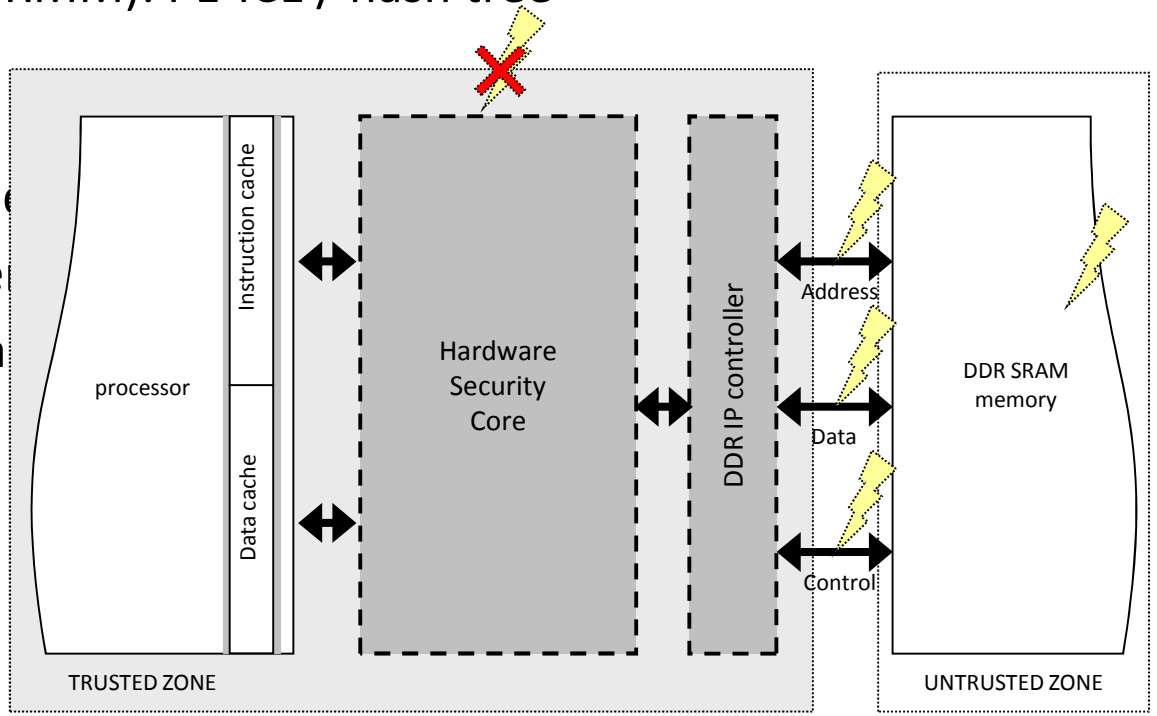
The challenge of memory protection & Threat Model

- External bus access leads to
  - Code extraction\modification
  - Private data extraction\modification
- Threat model
  - A secure zone
  - Any possible modification and observation on the address and data buses
- Targeted attacks
  - Spoofing
  - Relocation
  - Replay



State of the art

- Existing solutions relying on the same threat model
  - AEGIS (MIT): One-time-pad / Cached hash tree (OS controlled)
  - XOM (Stanford): One-time-pad / MD5 (OS controlled)
  - PE-ICE (LIRMM): AES / tag comparison
  - TEC-Tree (Princeton\LIRMM): PE-ICE / hash tree
- Issues
  - High memory overhea
  - Software execution pe
  - Area overhead (severa



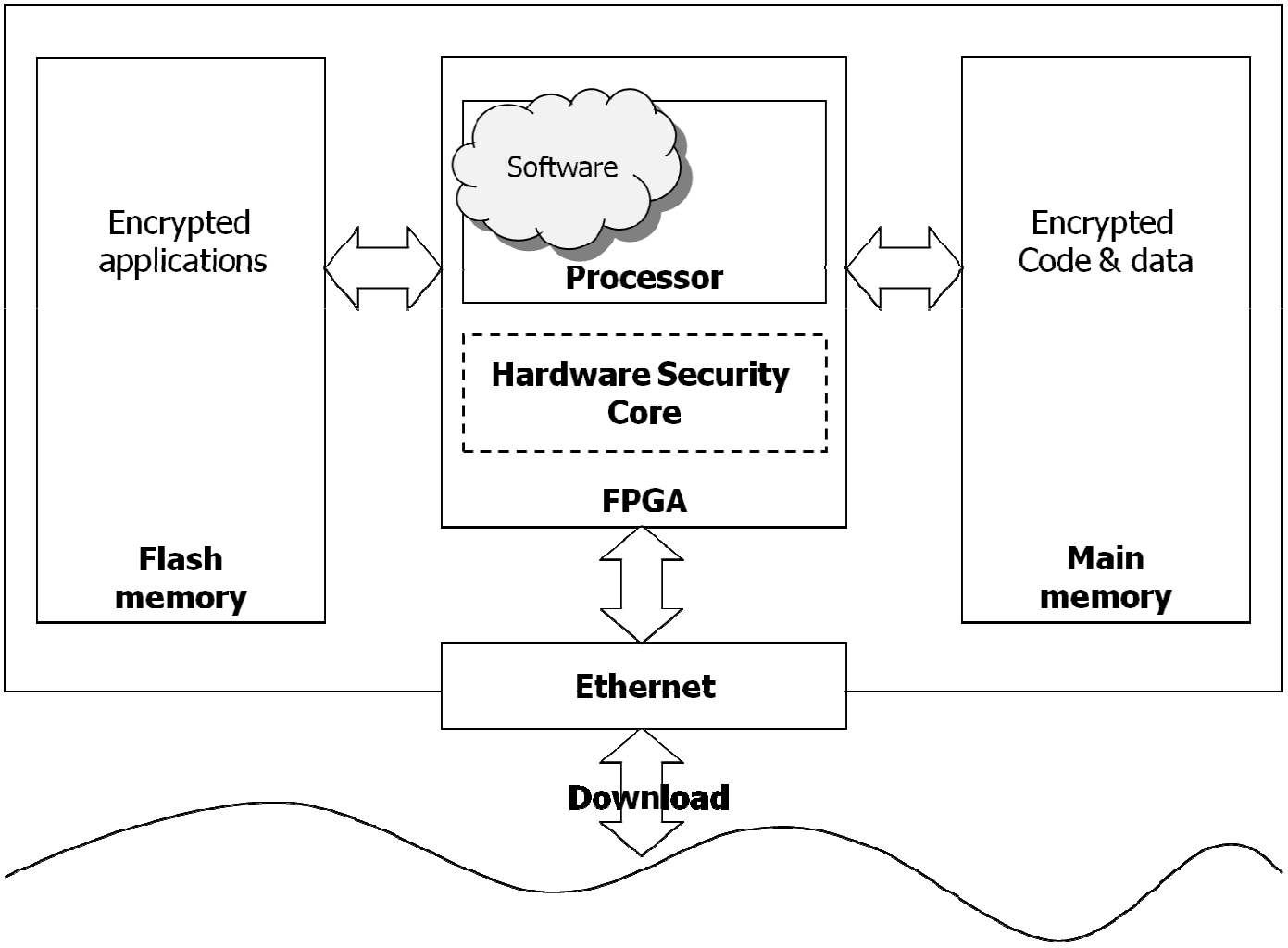
Contributions

- Solution fitting with embedded systems resources
  - Logic size
  - Memory footprint (including security data)
  - Performance
- Flexible solution for the software designer
  - Flexible architecture
  - Flexible security policy
- End to end solution, from boot to steady-state system operation

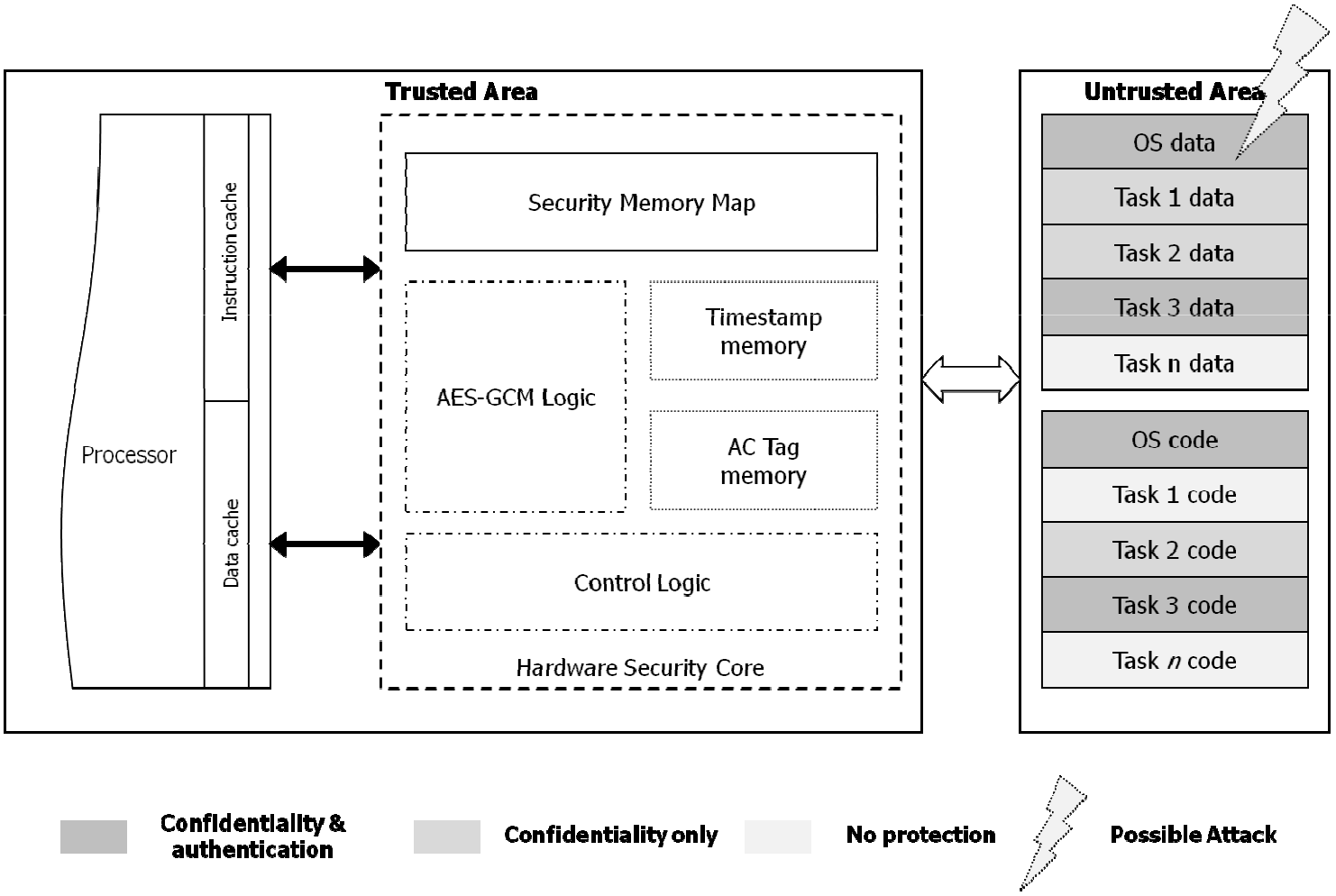
Outline

1) Global picture
2) **Memory protection**
3) Boot protection
4) Experimental setup & results
5) Communication protection: first ideas
6) Conclusion

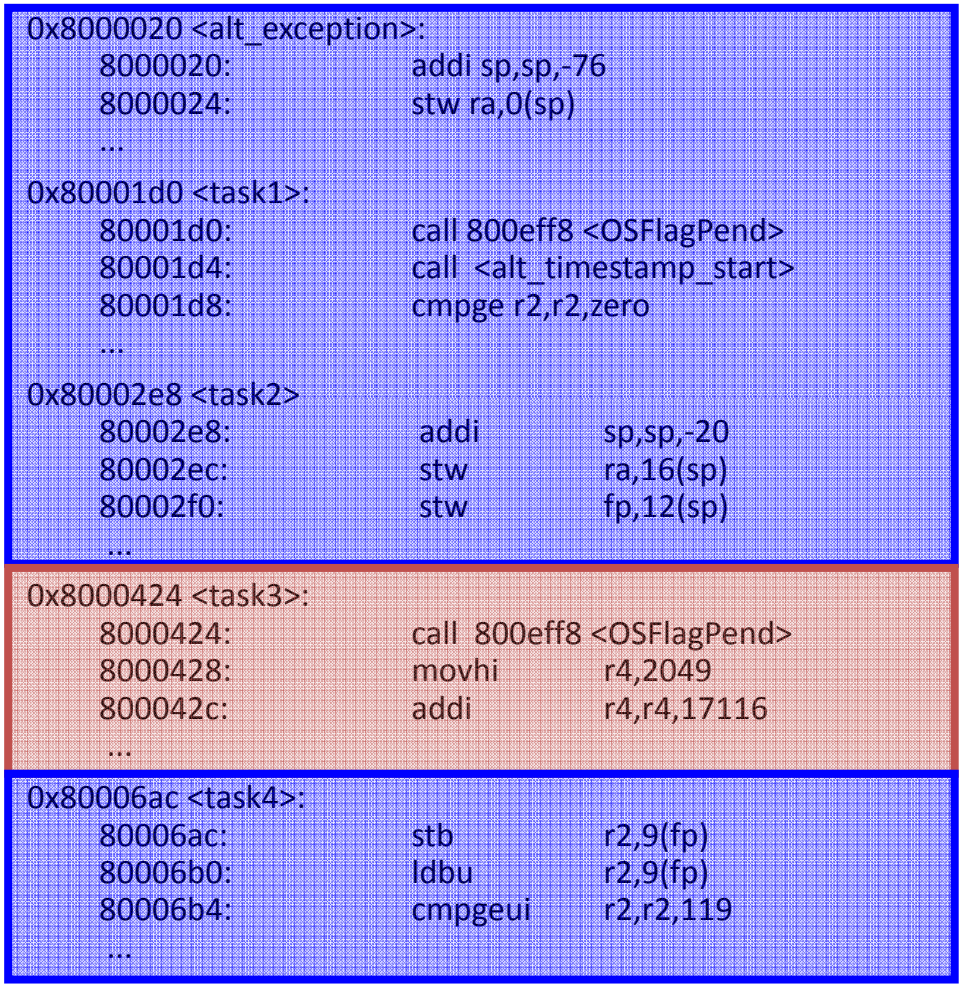
Model of the system setup



Main memory security system overview



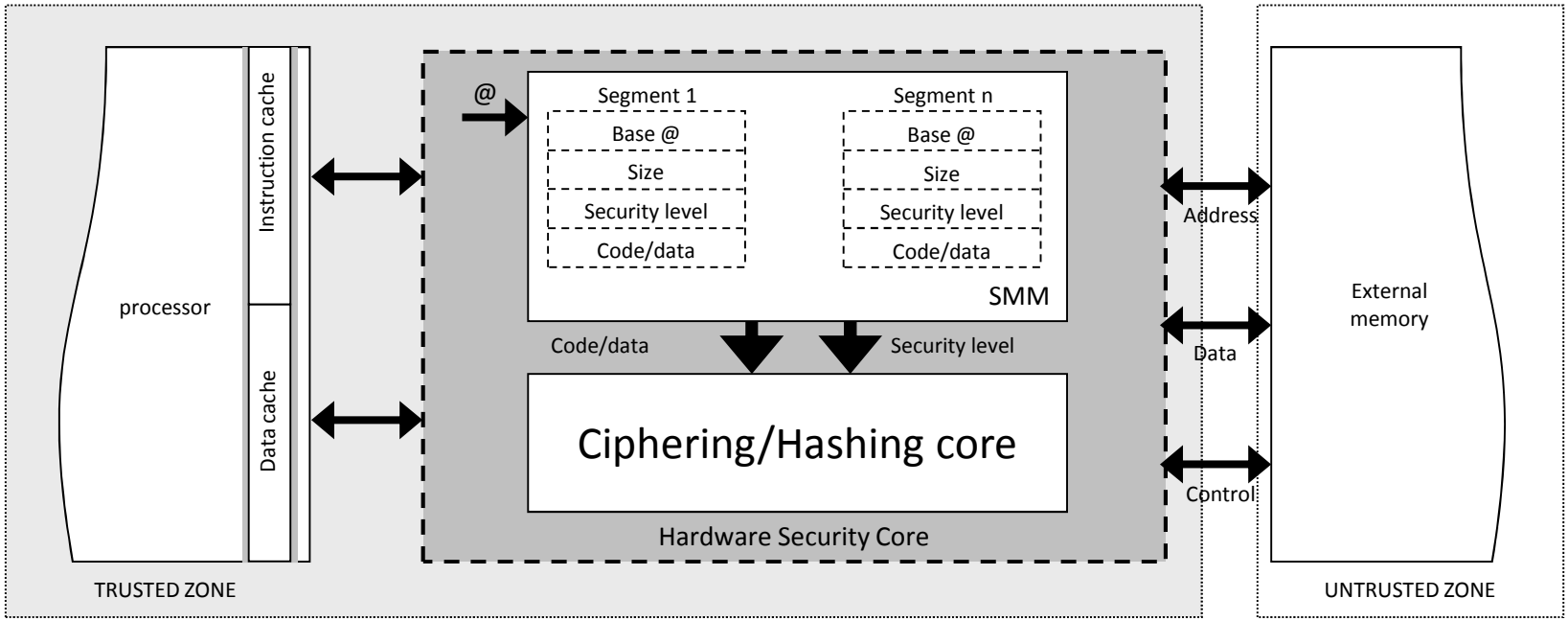
Security Memory Map construction



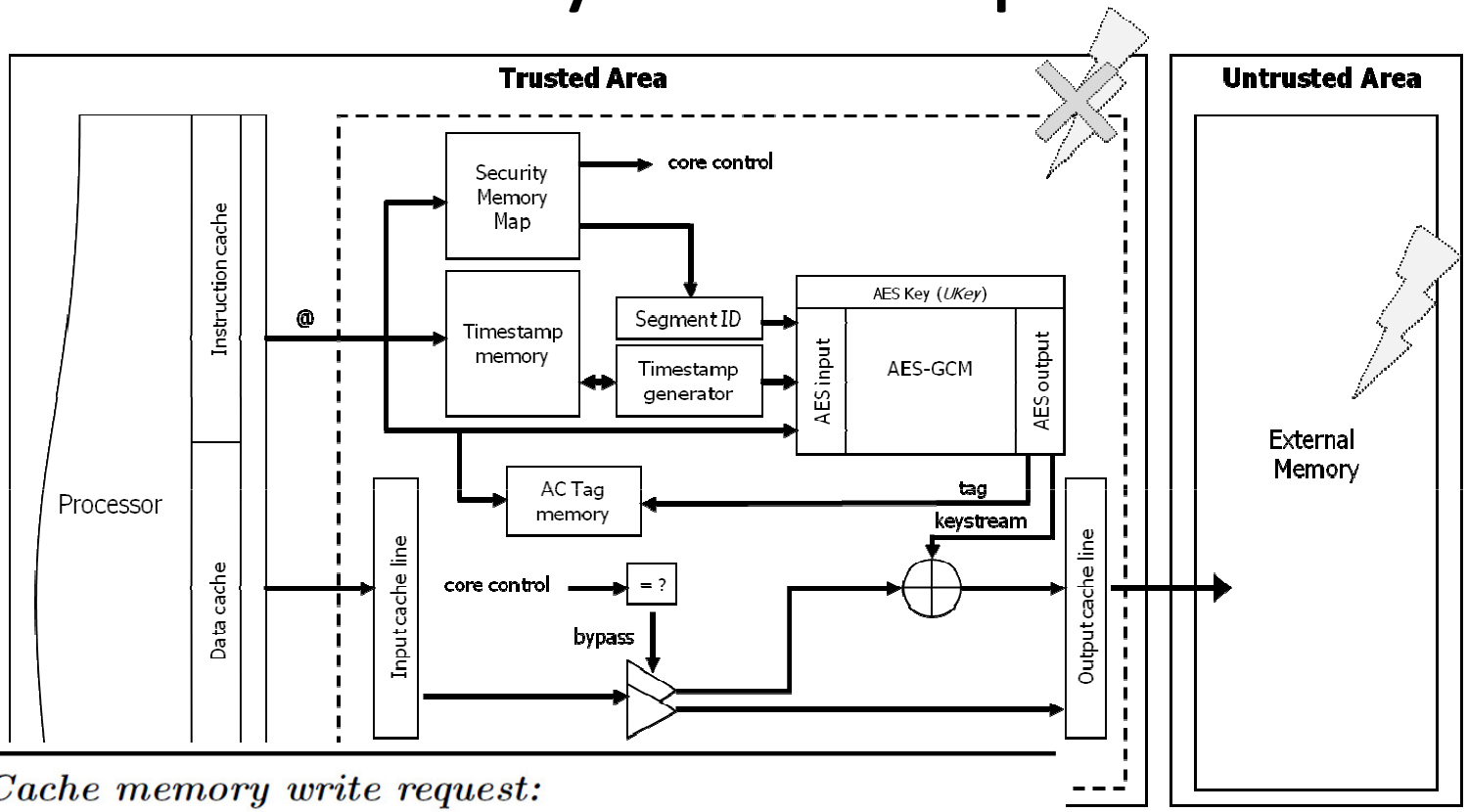
- Segment 0:
  - Base @: 0x8000020
  - Size: 1028 bytes
  - Confidentiality & integrity
  - Code
- Segment 1:
  - Base @: 0x8000424
  - Size: 680 bytes
  - Confidentiality only
  - Code
- Segment 2:
  - Base @: 0x80006ac
  - Size: 2048 bytes
  - Confidentiality & integrity
  - Code

Secure architecture with SMM

- Security Memory Mapping
  - Fully done in hardware, no OS modification



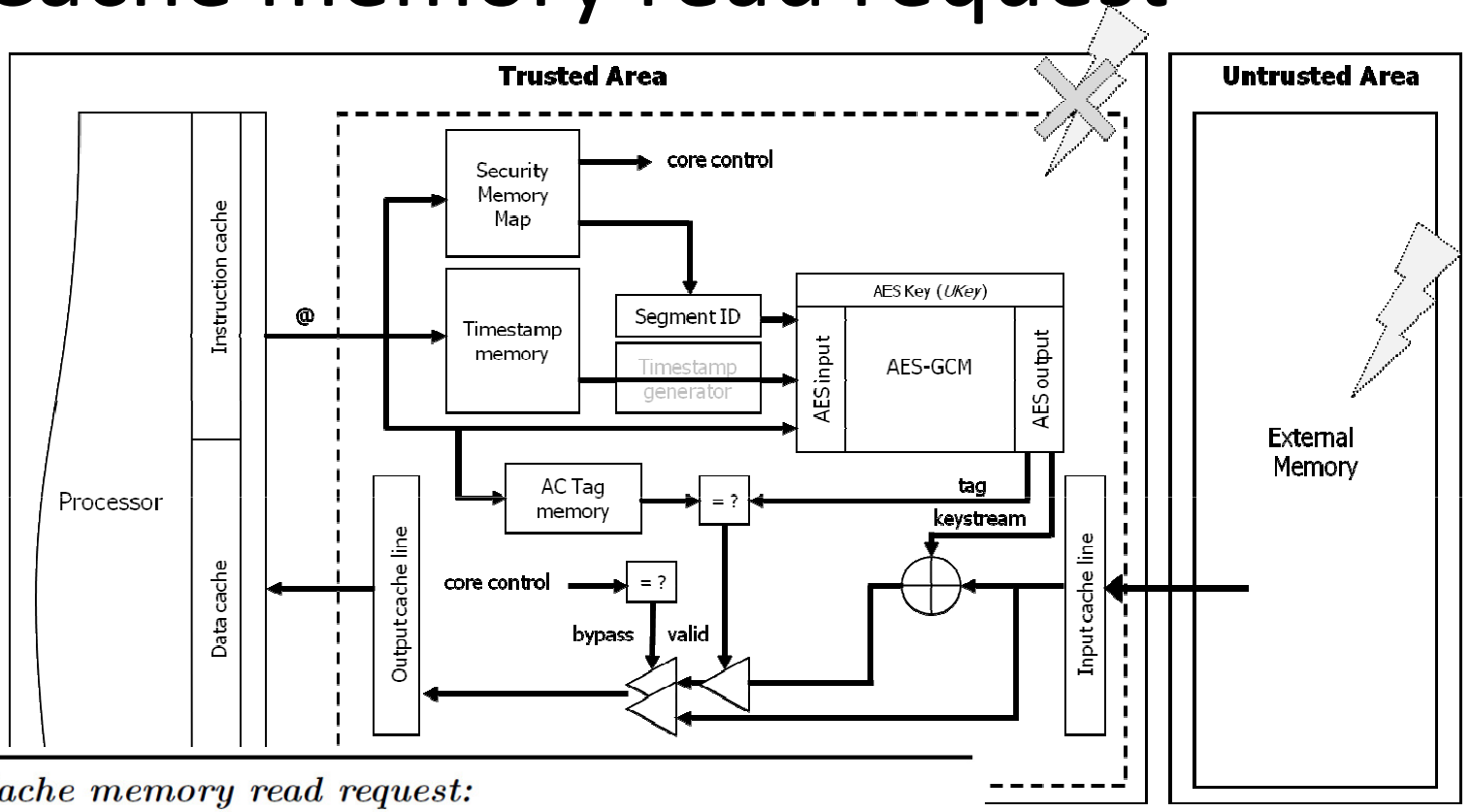
Cache memory write request



Algorithm 1 - ***Cache memory write request:***

1 – $Timestamp\ incrementation: TS = TS + 2$
2 – $\{Keystream, Tag\} = AES_{GCM}\{SegID, @, TS\}$
3 – $Ciphertext = Plaintext \oplus Keystream$
4 – $Ciphertext \Rightarrow external\ memory$
5 – $Timestamp\ storage: TS \Rightarrow TS\ memory$
6 – $Authentication\ Tag\ storage: Tag \Rightarrow Tag\ memory$

Cache memory read request

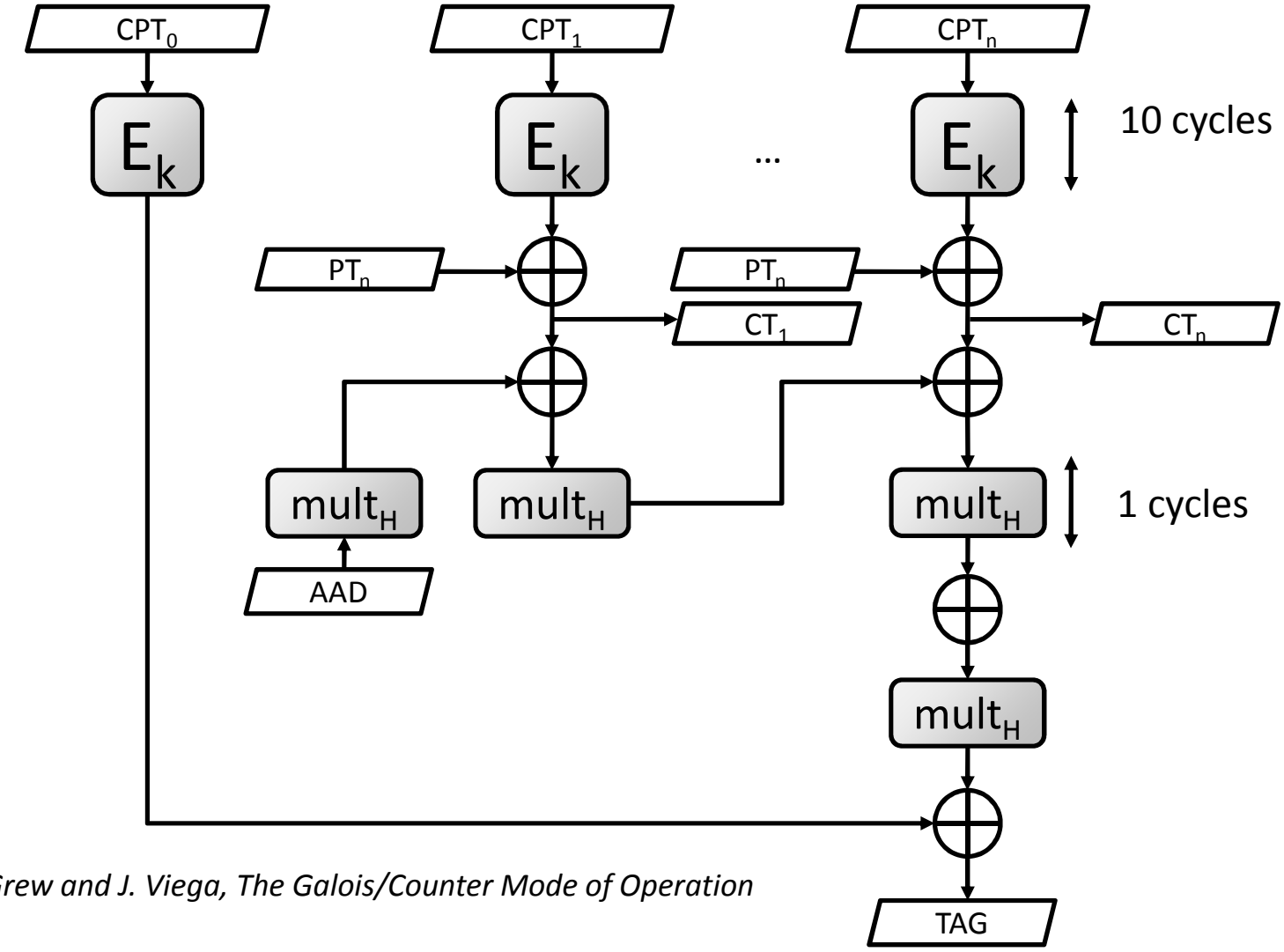


Algorithm 2 - ***Cache memory read request:***

1 – $TS\ loading: TS \Leftarrow TS\ memory$
2 – $Tag\ loading: Tag \Leftarrow Tag\ memory$
3 – $\{Keystream, Tag\} = AES_{GCM}\{SegID, @, TS\}$
4 – $Ciphertext\ loading: Ciphertext \Leftarrow external\ memory$
5 – $Plaintext = Ciphertext \oplus keystream$
6 – $Authentication\ checking: Tag \equiv Tag$
7 – $Plaintext \Rightarrow cache\ memory$

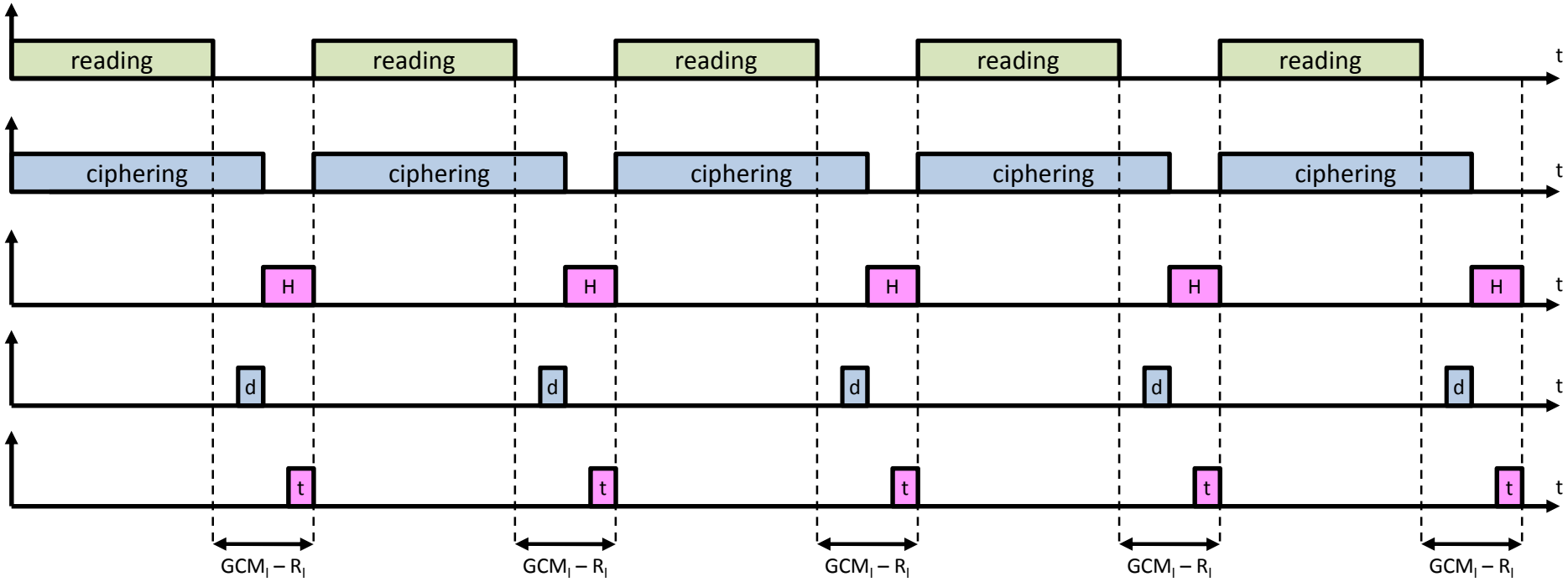
AES-GCM [8]

Overview



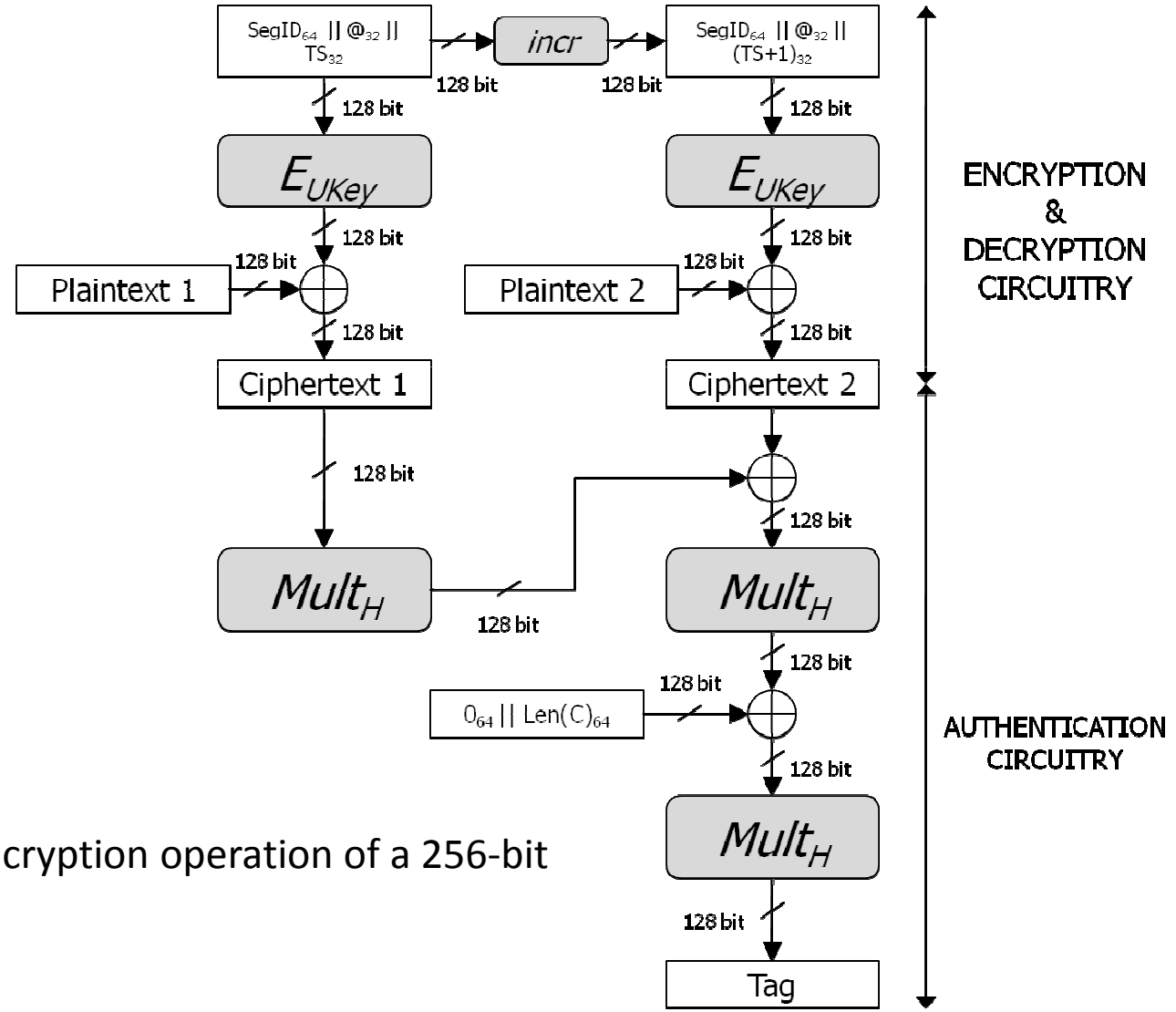
[8] *D. A. McGrew and J. Viega, The Galois/Counter Mode of Operation (GCM)*

AES-GCM



- Latency penalty : GCM$_I$ - R$_I$

AES-GCM architecture



For an authenticated encryption operation of a 256-bit plaintext cacheline.

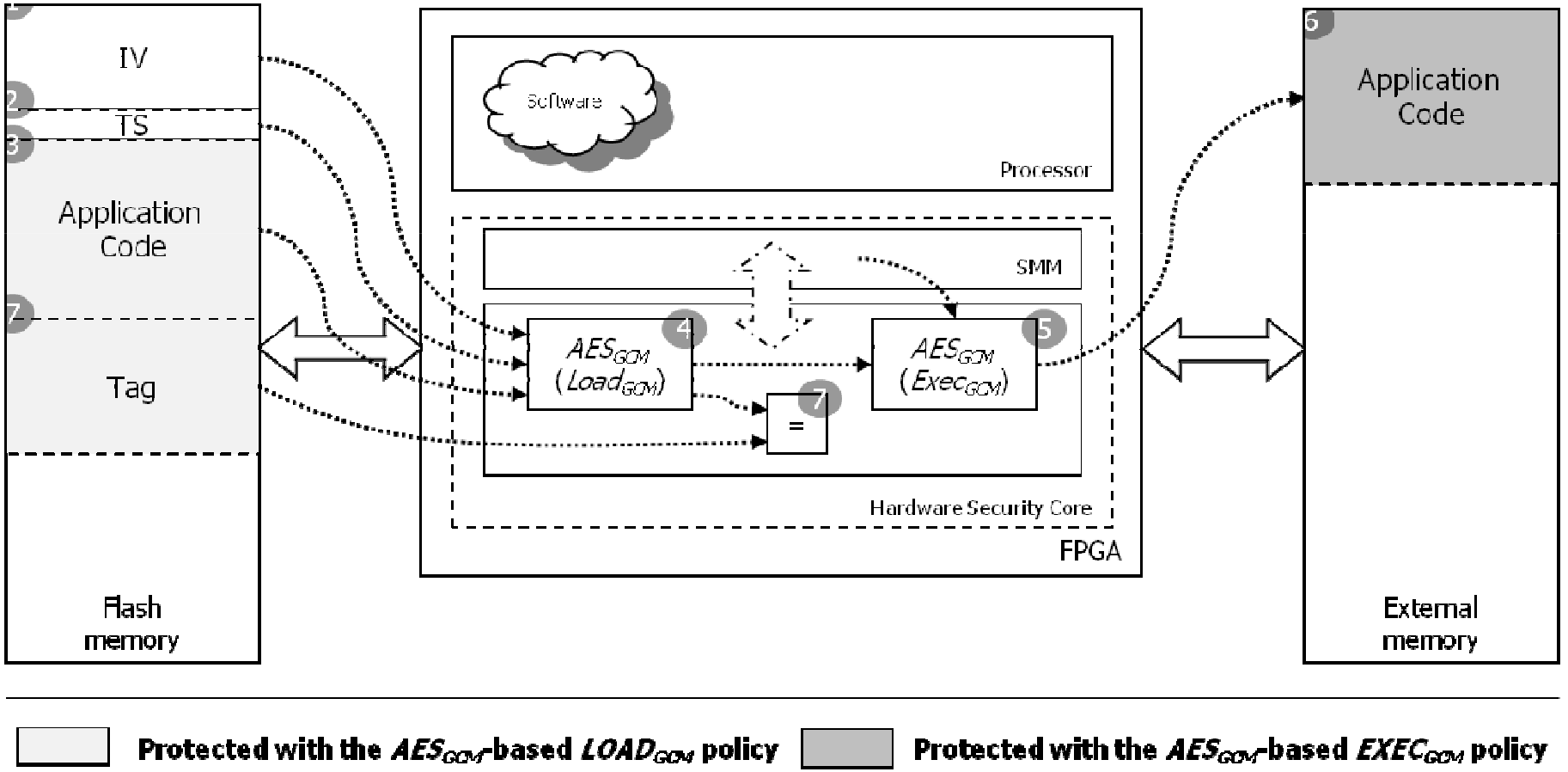
Outline

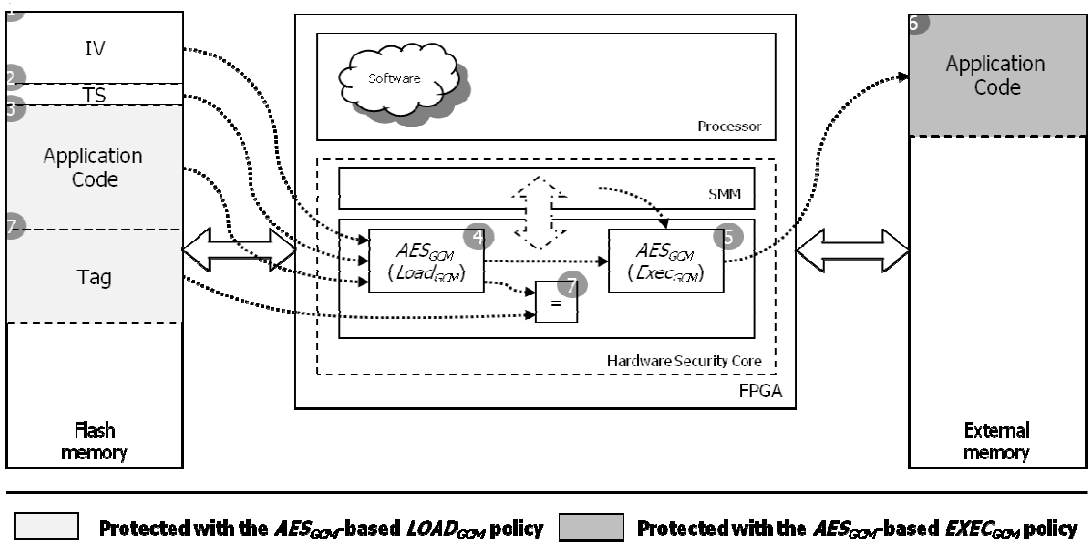
1) Global picture
2) Memory protection
3) **Boot protection**
4) Experimental setup & results
5) Communication protection: first ideas
6) Conclusion

Secure application Loading

- In our secure system, two distinct scenarios are considered
- Application code loading
  - In this scenario, the SMM is already loaded in hardware so that only application instruction loading to main memory is needed
  - This scenario may occurs if the SMM is included in an FPGA bitstream
- Application code and SMM loading
  - The SMM information must be loaded into a memory-based table adjacent to the microprocessor and application code must be loaded to main memory. SMM loading takes place first, followed by application code loading

Secure Application Code Loading from Flash Memory





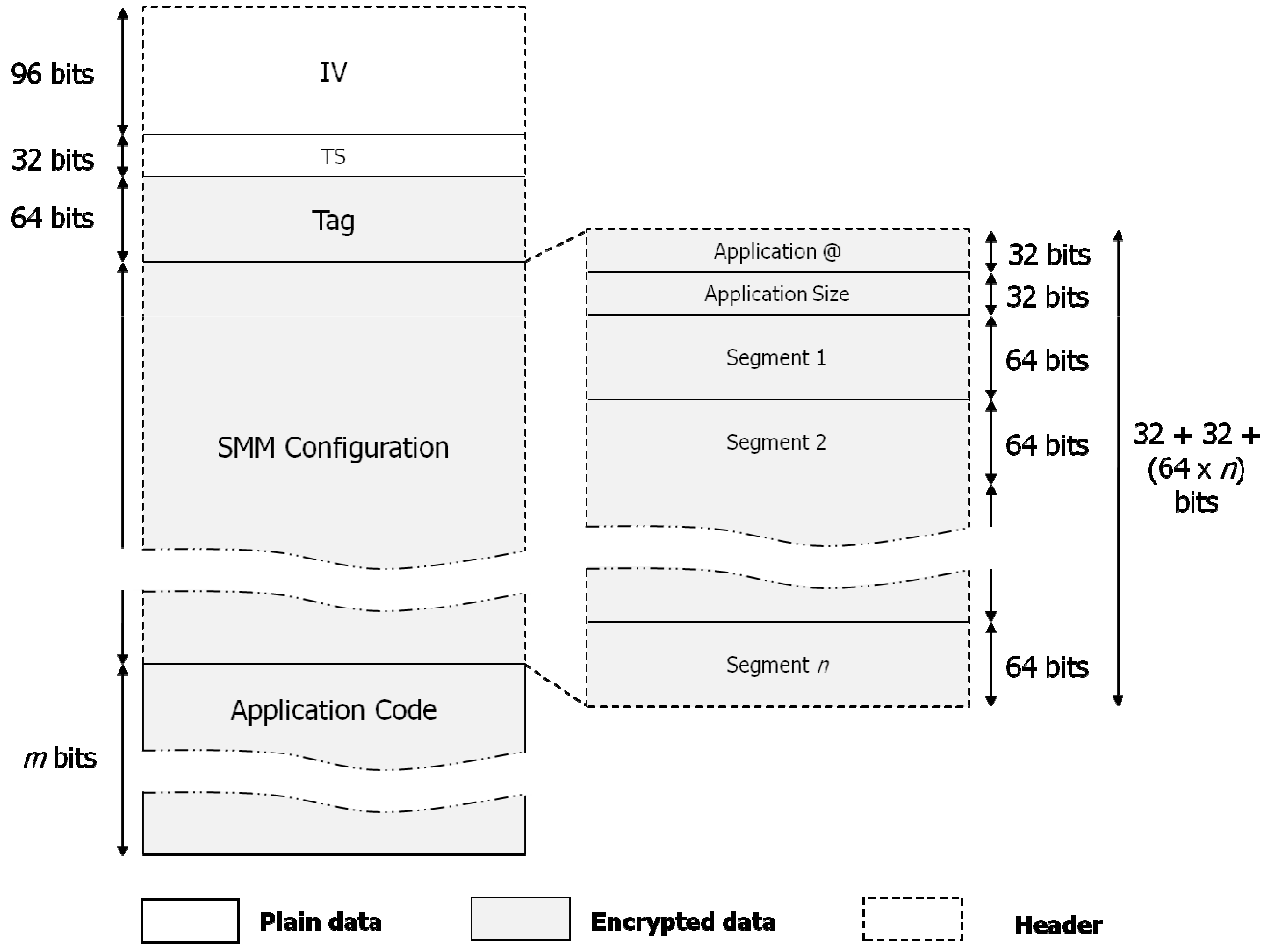
In more details…

Unique for each application

Algorithm 3 - ***Application loading***

1 – *The IV is copied to the* AES_{GCM} *running the* $Load_{GCM}$ *policy*
2 – *The TS is copied to the* AES_{GCM} *running the* $Load_{GCM}$ *policy*
Pipelined loop (for all application code)
3 – *The encrypted application code is copied to the* AES_{GCM} *running the* $Load_{GCM}$ *policy*
4 – *The encrypted application code is decrypted with the* $Load_{GCM}$ *policy*
5 – *The decrypted data is encrypted with the* AES_{GCM} *running the* $Exec_{GCM}$ *policy*
6 – *The encrypted data is copied in main memory*
End Loop
7 – *The application tag is compared with the one generated by the* AES_{GCM} *running the* $Load_{GCM}$ *policy. If both tags match, the application is securely loaded and can be safely decrypted for secure execution*

Application Code and SMM Loading from Flash Memory

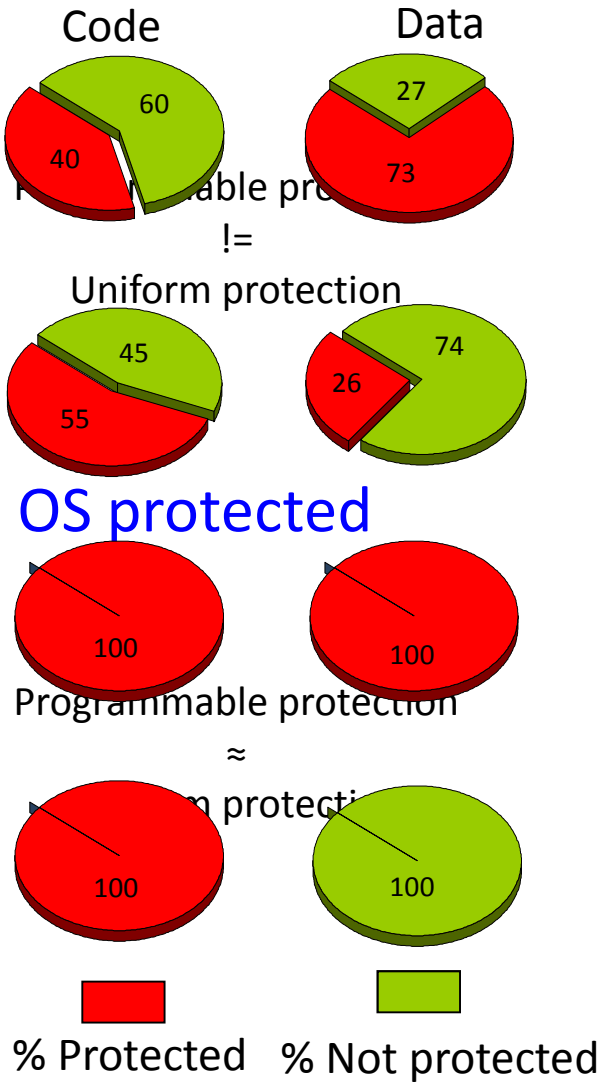


Outline

1) Global picture
2) Memory protection
3) Boot protection
4) **Experimental setup & results**
5) Communication protection: first ideas
6) Conclusion

Applications security policy

- Image processing:
  - Only algorithm core code & data protected (CI)
- Video-On-Demand:
  - MPEG decoder code must not be stolen (CO)
  - Image must not be stolen (CO)
  - AES sensitive data must be protected (CI)
- Communication:
  - Processed data must not be stolen (CO)
  - Code must not be attacked (CI)
- Hash:
  - Code must not be stolen (CO)
  - Processed data can be stolen



Application memory protection details by protection level

	Conf. and Auth.						Conf. only						No Protection					
	Code			Data			Code			Data			Code			Data		
App	KB	T	S	KB	T	S	KB	T	S	KB	T	S	KB	T	S	KB	T	S
Img	25	2	5	33	2	3	7	2	1	10	2	1	48	1	1	16	1	1
VOD	26	5	3	113	6	4	58	1	1	0	0	0	68	1	1	318	1	1
Com	71	6	1	28	0	2	0	0	0	40	6	1	0	0	0	0	0	0
Halg	0	0	0	0	0	0	92	5	1	0	0	0	0	0	0	55	5	1

KB denotes the size in KB, T the number of tasks and S the number of segments

Area Overhead of Security

- All four applications were implemented on a Spartan-6 (SP605- XC6SLX45T), 128 MB of external DDR3 memory and 32 MB of flash memory

	Uniform protection				Programmable protection			
Arch.	μB + HSC		HSC		μB + HSC		HSC	
	LUTs	FFs	LUTs	FFs	LUTs	FFs	LUTs	FFs
Img. 512	7095	3769	3485	1122	7237	3796	3627	1149
	16.3%	6.9%	8.0%	2.1%	16.6%	7.0%	8.3%	2.1%
Img. 2k	6820	3660	3485	1122	6962	3687	3627	1149
	15.6%	6.7%	8.0%	2.1%	15.9%	6.8%	8.3%	2.1%
VOD 512	7229	3796	3619	1149	7209	3792	3599	1145
	16.6%	7.0%	8.3%	2.1%	16.5%	6.9%	8.2%	2.1%
VOD 2k	6954	3687	3619	1149	6934	3683	3599	1145
	15.9%	6.8%	8.3%	2.1%	15.9%	6.7%	8.2%	2.1%
Com. 512	7080	3768	3470	1121	7120	3776	3510	1129
	16.2%	6.9%	7.9%	2.1%	16.3%	6.9%	8.0%	2.1%
Com. 2k	6805	3659	3470	1121	6845	3667	3510	1129
	15.6%	6.7%	7.9%	2.1%	15.7%	6.7%	8.0%	2.1%
Hash 512	6186	3598	2576	951	6153	3596	2543	949
	14.2%	6.6%	5.9%	1.7%	14.1%	6.6%	5.8%	1.7%
Hash 2k	5911	3489	2576	951	5878	3487	2543	949
	13.5%	6.4%	5.9%	1.7%	13.5%	6.4%	5.8%	1.7%

Detailed breakdown of hardware security core (HSC) logic resource usage

Uniform protection										
App.	Total		AES_{GCM}		AC Tag Storage		SMM		Ctrl.	
	LUTs	FFs	LUTs	FFs	LUTs	FFs	LUTs	FFs	LUTs	FFs
Img.	3485	1122	2065	798	473	154	23	3	924	167
	8.0%	2.1%	4.7%	1.5%	1.1%	0.3%	0.1%	0.0%	2.1%	0.3%
VOD	3619	1149	2065	798	604	177	29	3	921	171
	8.3%	2.1%	4.7%	1.5%	1.4%	0.3%	0.1%	0.0%	2.1%	0.3%
Com.	3470	1121	2065	798	470	153	20	3	915	167
	7.9%	2.1%	4.7%	1.5%	1.1%	0.3%	0.0%	0.0%	2.1%	0.3%
Hash	2576	951	2065	798	0	0	21	3	490	150
	5.9%	1.7%	4.7%	1.5%	0.0%	0.0%	0.0%	0.0%	1.1%	0.3%
Programmable protection										
App.	Total		AES_{GCM} Core		AC Tag Storage		SMM		Ctrl.	
	LUTs	FFs	LUTs	FFs	LUTs	FFs	LUTs	FFs	LUTs	FFs
Img.	3627	1149	2065	798	473	153	214	31	875	167
	8.3%	2.1%	4.7%	1.5%	1.1%	0.3%	0.5%	0.1%	2.0%	0.3%
VOD	3599	1145	2065	798	473	153	161	27	900	167
	8.2%	2.1%	4.7%	1.5%	1.1%	0.3%	0.4%	0.0%	2.1%	0.3%
Com.	3510	1129	2065	798	475	154	61	10	909	167
	8.0%	2.1%	4.7%	1.5%	1.1%	0.3%	0.1%	0.0%	2.1%	0.3%
Hash	2543	949	2065	798	0	0	12	1	466	150
	5.8%	1.7%	4.7%	1.5%	0.0%	0.0%	0.0%	0.0%	1.1%	0.3%

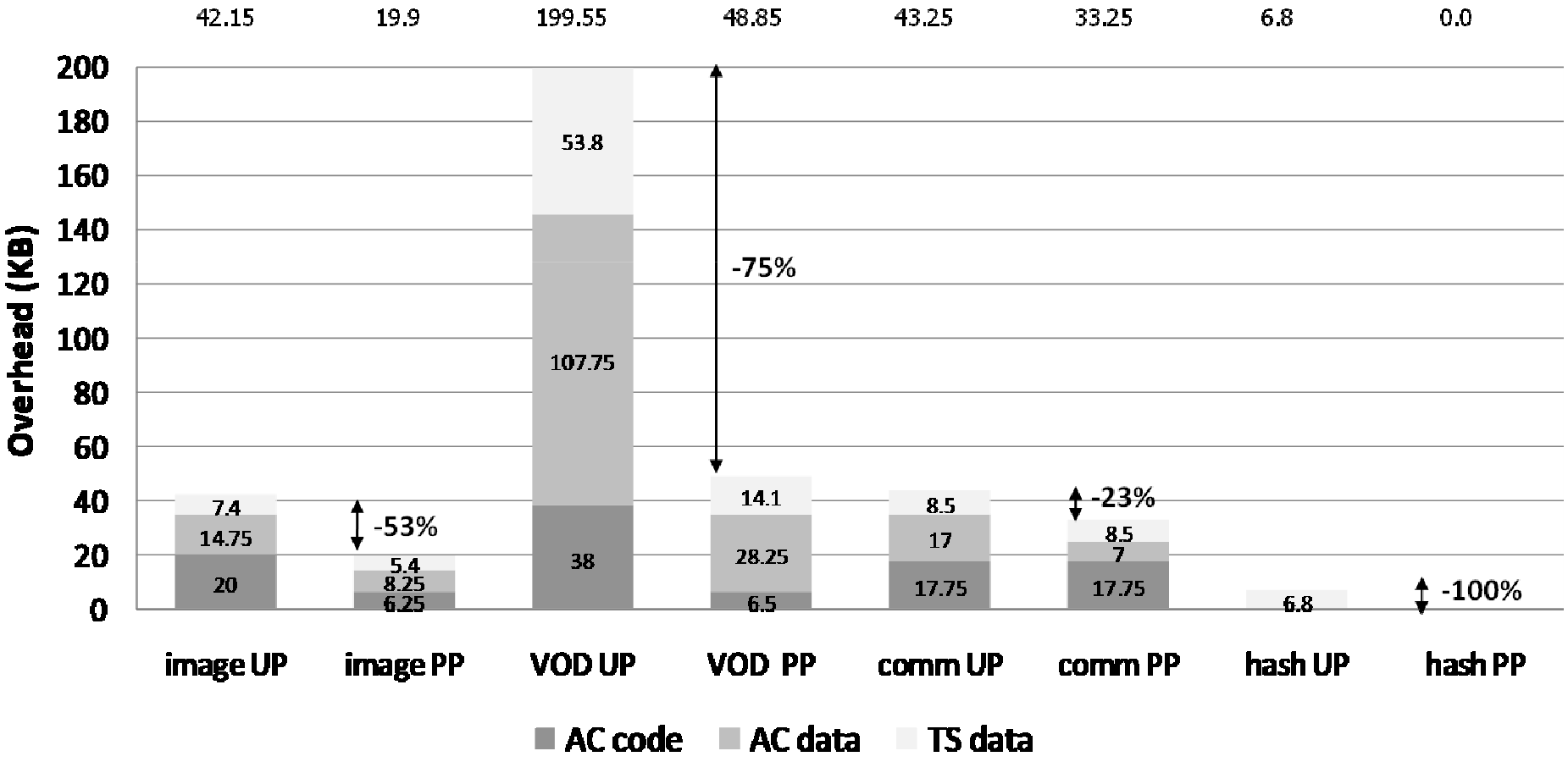
Performance Cost of Security

	No protection	Uniform protection		Programmable protection	
Arch.	Time (ms)	Time (ms)	Overhead	Time (ms)	Overhead
Img. 512	150.5	188.0	-24.9%	173.4	-15.2%
Img. 2k	131.3	156.9	-19.5%	146.9	-11.9%
VOD 512	13691.5	16806.4	-22.8%	15619.8	-14.1%
VOD 2k	11940.3	13751.2	-15.2%	13453.5	-12.7%
Com. 512	69.1	84.1	-21.6%	78.7	-14.0%
Com. 2k	60.2	66.7	-10.8%	65.4	-8.6%
Hash 512	8.6	10.2	-18.9%	9.9	-15.1%
Hash 2k	7.5	8.7	-15.9%	8.6	-14.4%

The extra latency caused by our security approach for the prototype implementation is 7 cycles for a 256-bit cacheline read and 13 cycles for a cacheline write.

The cacheline write overhead is primarily due to the 10-cycle 128-bit AES operation. The read overhead is reduced due to an overlap in AES operation and bus read operations.

Memory Cost of Security



Secure loading time

- For application code transferred from flash to the output of the crypto core

	No protection	Uniform protection				Programmable protection			
App.	Time (ms)	Time (ms)				Time (ms)			
		Load	Exec	Total	Overhead	Load	Exec	Total	Overhead
Img.	19.84	20.36	1.51	21.87	10.21%	20.36	1.05	21.41	7.92%
VOD	37.32	38.30	2.87	41.17	10.32%	38.30	2.41	40.71	9.07%
Com.	17.46	17.92	1.34	19.26	10.33%	17.92	1.34	19.26	10.33%
Hash	22.48	22.91	1.73	24.64	9.62%	22.91	1.73	24.64	9.62%

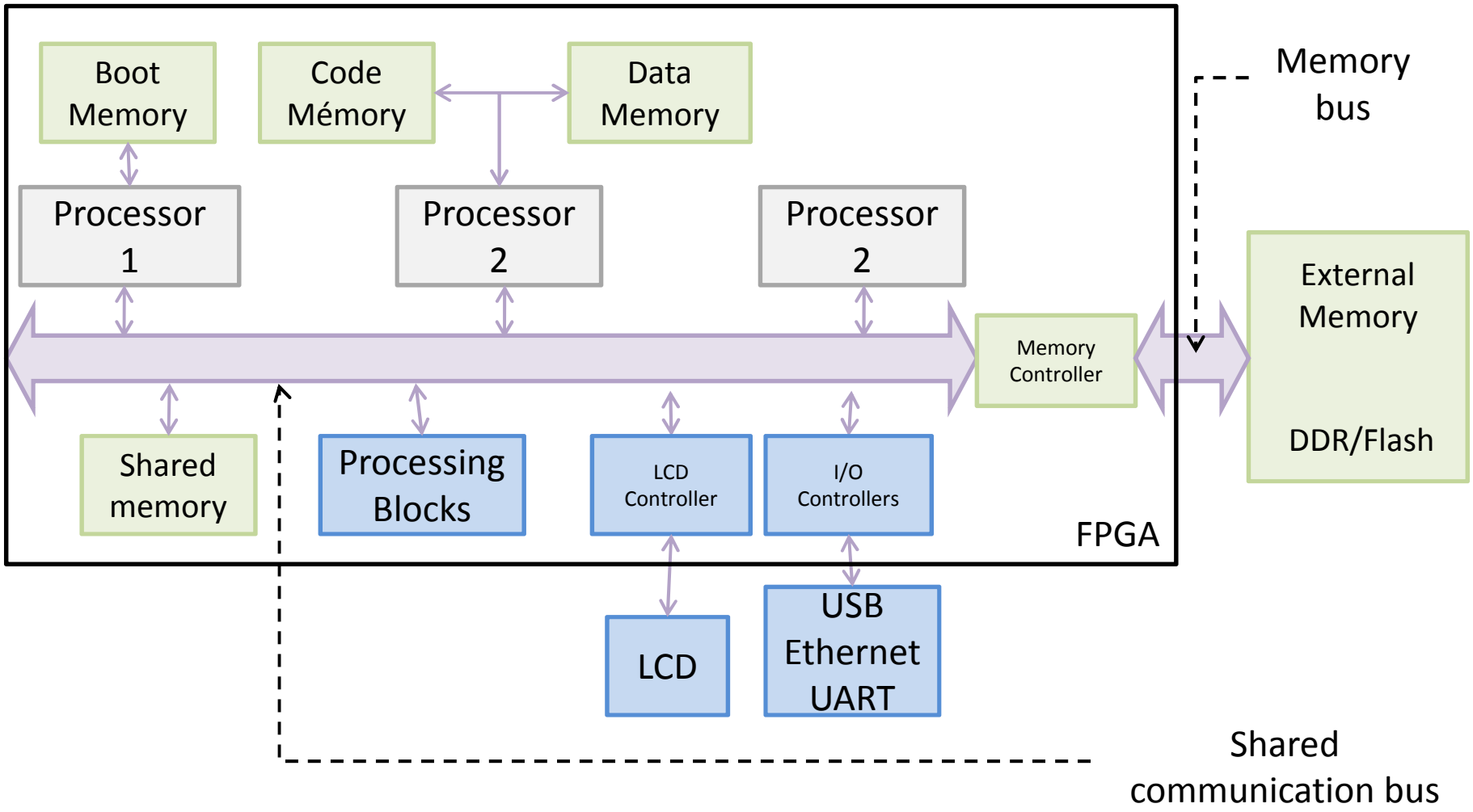
Load indicates the amount of time to load the application from flash and decrypt/authenticate it using the GCM policy.
Exec indicates the amount of time needed to reencrypt and generate authentication tags for the application using the GCM policy.

Outline

1) Global picture
2) Memory protection
3) Boot protection
4) Experimental setup & results
5) **Communication protection: first ideas**
6) Conclusion

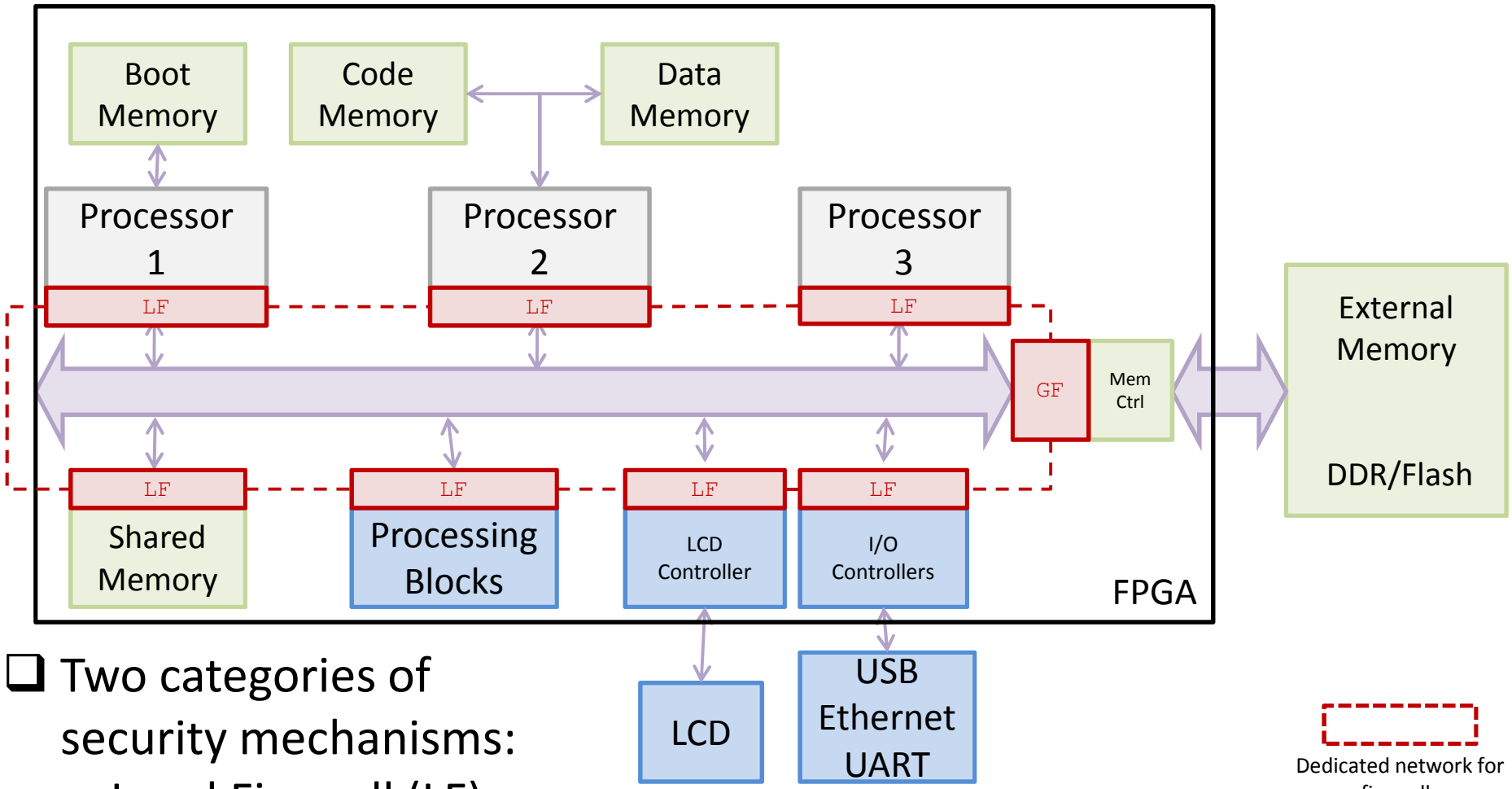
A Multi-Processors Architecture with Secured Communications (1/3)

Inter-communication model



A Multi-Processors Architecture with Secured Communications (2/3)

Application of security enhancements



- Two categories of security mechanisms:
  - Local Firewall (LF).
  - Global Firewall (GF).

A Multi-Processors Architecture with Secured Communications (3/3)

What do we mean by a security policy ?

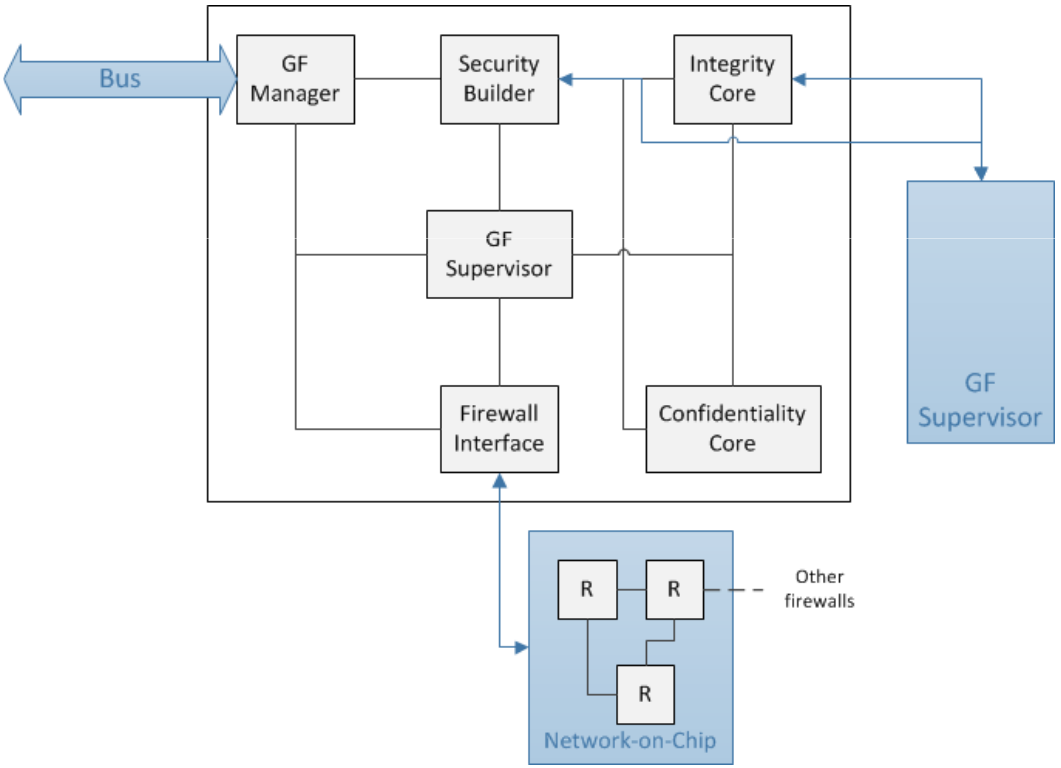
- A security policy can be defined by several parameters:
  - Read/Write access rights.
  - Allowed data characteristics (format and value).
  - Tags associated with each firewall.
  - Address spaces where security policies are applicable.

- Parameters specific to the external memory:
  - Confidentiality (use of a ciphering algorithm such as AES, RSA…).
  - Integrity (can be done in association with the ciphering : AES-CMAC, AES-GCM…).

- A security policy can be applied on discontinuous address spaces.
- A functional block of the architecture can require more than one security policy.
- Security policies are adaptative (fitting with new standards).

Firewalls features description (1/2)

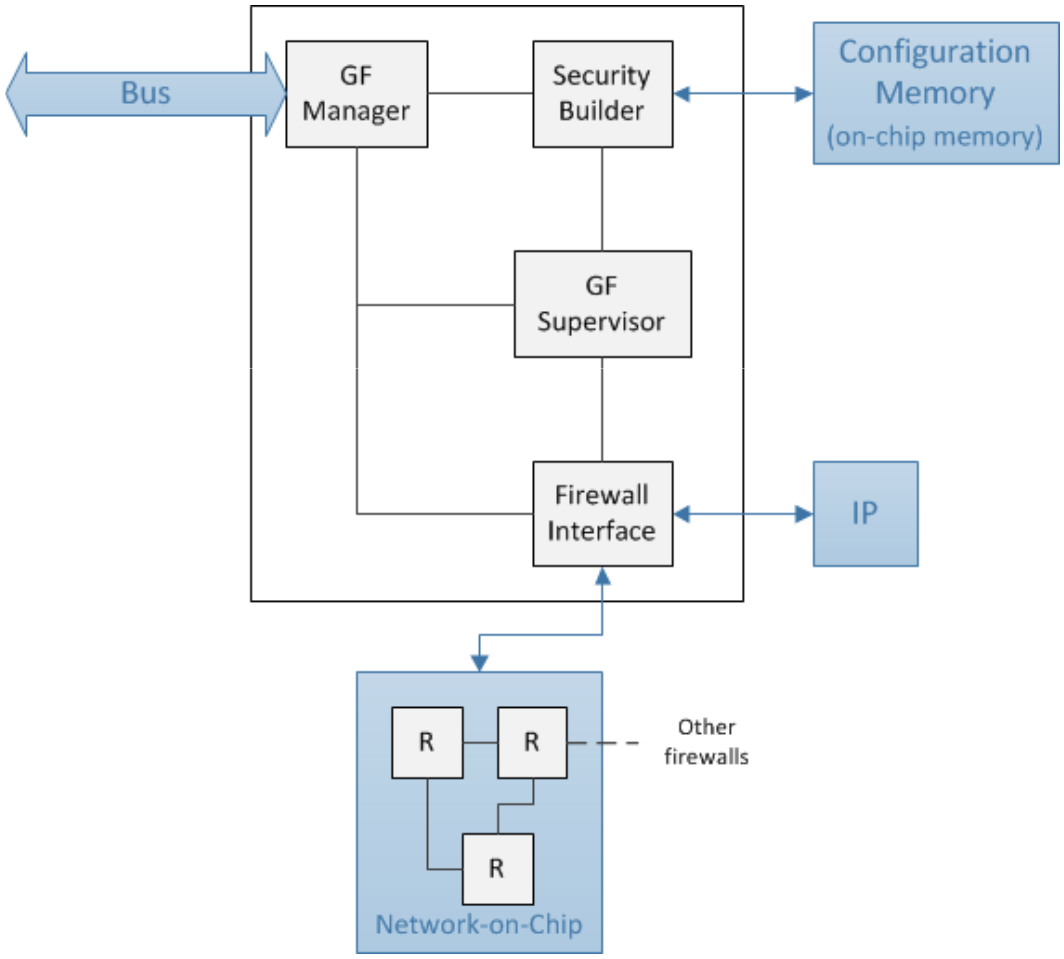
Security services provided by Global Firewall



- Confidentiality and integrity protection for unsecure external memory.
- Addresses and data checking features.
- Multi-level protection.
- Firewalls manager.

Firewalls features description (2/2)

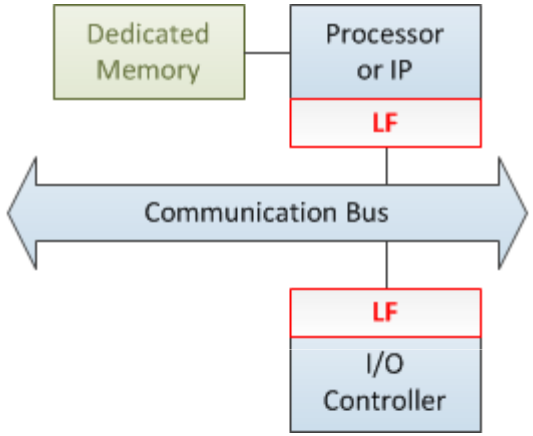
Security services provided by Local Firewall



- Addresses and data checking features.
- Multi-level protection.
- Communications with other firewalls.

Firewalls behavioural scenarios (1/2)

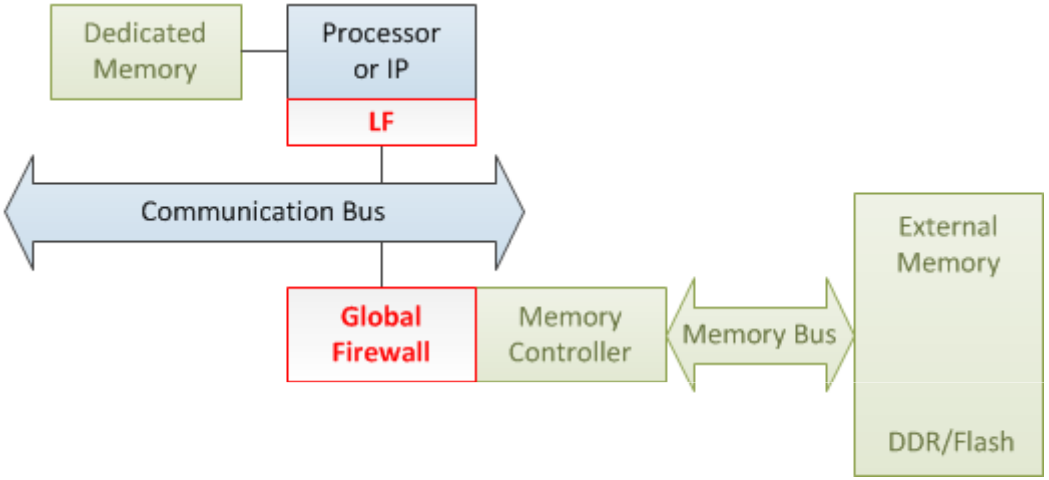
Scenario 1: communication between two Local Firewalls



Controls done the at the source	Description & comments
Existence of target IP address	Aims to cover spoofing/relocation.
Read/Write access rules	Protects from illegal accesses.
Allowed data format	Similar to buffer overflow.
Controls done at the target	**Description & comments**
Source ID	Authenticity-like feature.
Existence of the section needed	Aims to cover spoofing/relocation.
Allowed data values	An abnormal value can cause malfunctions.

Firewalls behavioural scenarios (2/2)

Scenario 2: communication between a Local Firewall and the Global Firewall



Controls done the at the source	Description & comments
Existence of DDR address	Aims to cover spoofing/relocation.
Allowed data format	Similar to buffer overflow.
Controls done at the target	**Description & comments**
Source ID	Authenticity-like feature.
Existence of the section needed	Aims to cover spoofing/relocation.
Read/Write access rules	Protect from illegal accesses.
Confidentiality / Integrity	Aims to cover replay (for read/write data).

Security flows

Connections between firewalls

- Firewalls are considered as “security sensors”:
  - A processor dedicated to security services is like a “mega-supervisor” of the system (other protection can be managed).
  - The Global Firewall is the manager of all Local Firewalls.
  - Security-related information goes through a NoC dedicated for firewall communications.

- Reconfiguration flow:
  - The processor dedicated to security services is responsible for reconfiguring security policies of the Global Firewall.
  - Global Firewall can reconfigure Local Firewall security parameters through the dedicated network.

Outline

1) Global picture
2) Memory protection
3) Boot protection
4) Experimental setup & results
5) Communication protection: first ideas
6) **Conclusion**

Conclusion

- Global approach for data and communication protection
  - Based on hardware resources, no modification of the OS / Low latency solution
- Reconfigurable security policy to set up a security on an application basis (thread level)
  - Need of monitoring
- Secure execution in a multiprocessor context (cache coherency)

Data and communication protection in reconfigurable embedded systems

Jérémie Crenne, Pascal Cotret, Guy Gogniat, Russel Tessier, Jean-Philippe Diguet