

PHILIPS

Physical Unclonable Functions, Helper Data Algorithms and Their Applications

Jorge Guajardo¹

Joint work with:

Christoph Boesch³, Sandeep S. Kumar¹, Roel Maes⁴,
Ahmad-Reza Sadeghi³, Geert.-Jan Schrijen², Jamshid Shokrollahi³,
and Pim Tuyls²

¹ Philips Research Europe, Eindhoven, The Netherlands

² Business Line Intrinsic-ID, Eindhoven, The Netherlands

³ Ruhr-Universitaet Bochum, Bochum, German

⁴ COSIC – KU Leuven, Leuven, Belgium

Contents

Relevance

Physical Unclonable Functions

Helper Data Algorithms (on FPGAs)

Applications

Conclusions

Intellectual Property Theft

- Annual value of trade in fake goods: \$650 Billion
 - Spare parts
 - Clothing
 - Perfumes
 - Medicines
 - Audio & video
 - Software
 - Electronic Designs

10% of all High Tech Products sold are Counterfeit!

- IC designs
- Electronic circuitry
- Configuration data of programmable devices

Real world example in the news

The New York Times
nytimes.com

PRINTER-FRIENDLY FORMAT
SPONSORED BY FROM THE FIGHT

May 9, 2008

F.B.I. Says the Military Had Bogus Computer Gear

By [JOHN MARKOFF](#)

SAN FRANCISCO Counterfeit products are a routine threat for the electronics industry. However, the more sinister specter of an electronic Trojan horse, lurking in the circuitry of a computer or a network router and allowing attackers clandestine access or control, was raised again recently by the [F.B.I.](#) and the Pentagon.

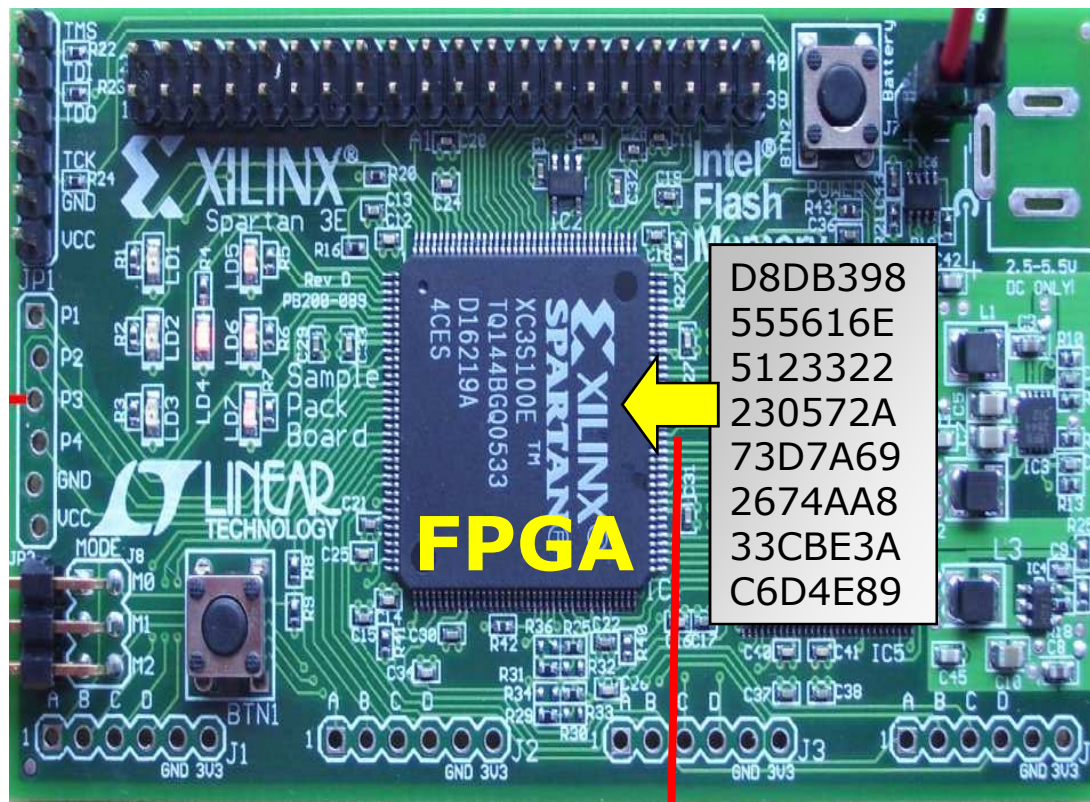
The new law enforcement and national security concerns were prompted by Operation [Cisco Raider](#), which has led to 15 criminal cases involving counterfeit products bought in part by military agencies, military contractors and electric power companies in the United States. Over the [two-year operation](#), 36 search warrants have been executed, resulting in the discovery of 3,500 counterfeit Cisco network components with an estimated retail value of more than \$3.5 million, the F.B.I. said in a statement.

Real world example in the news

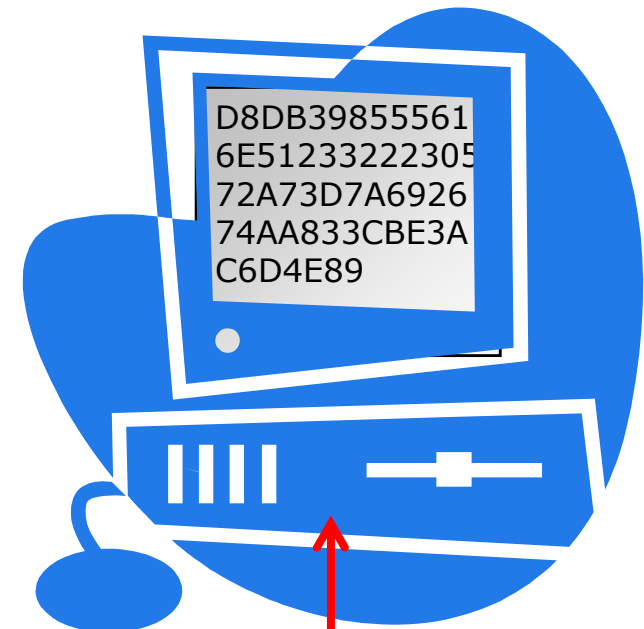


Source: slide from FBI presentation

Our Focus: FPGA Design Cloning

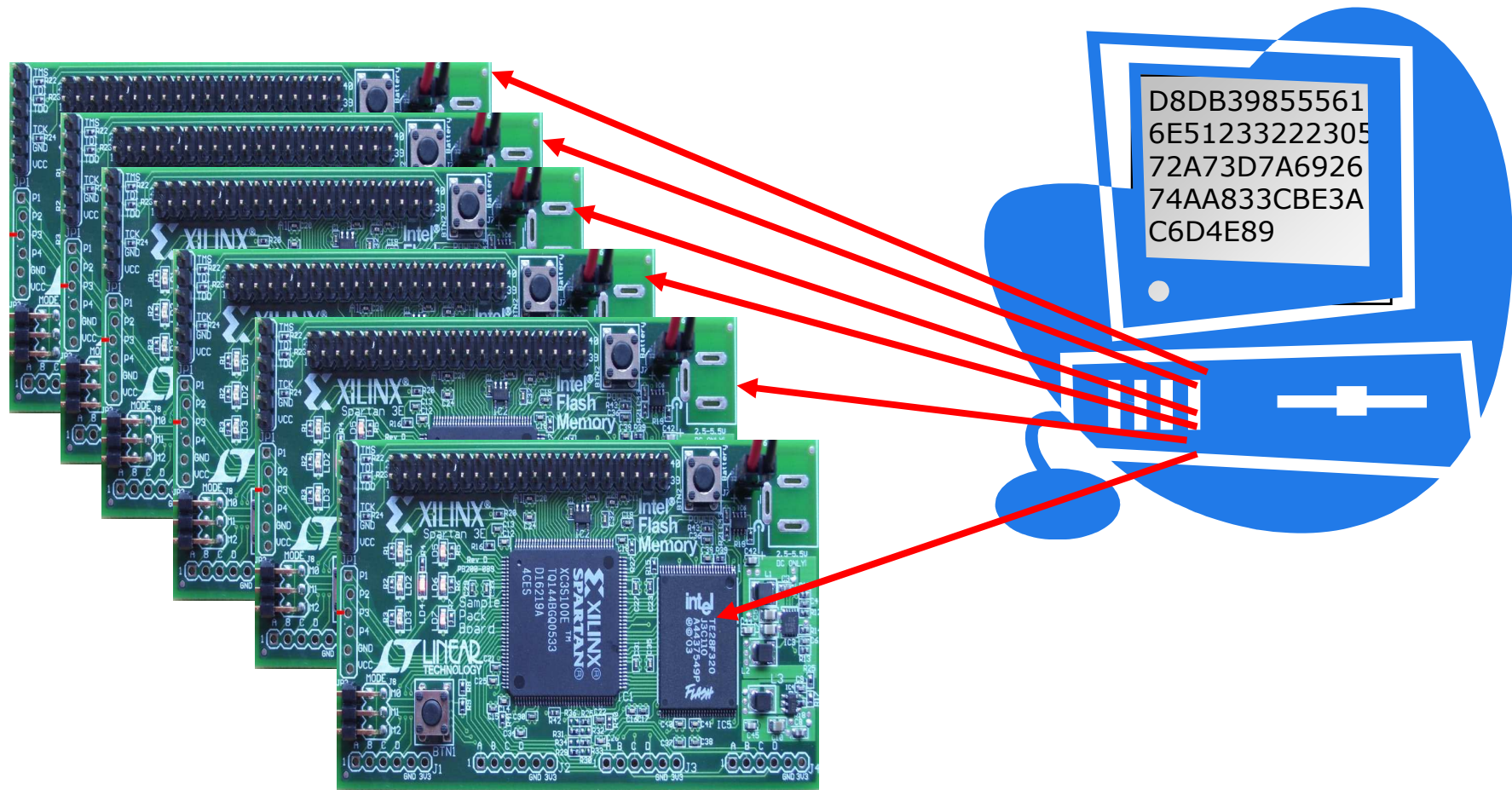


D8DB398
555616E
5123322
230572A
73D7A69
2674AA8
33CBE3A
C6D4E89



D8DB39855561
6E51233222305
72A73D7A6926
74AA833CBE3A
C6D4E89

Problem: FPGA Design Cloning



Contents

Relevance

Physical Unclonable Functions

Helper Data Algorithms (on FPGAs)

Applications

Conclusions

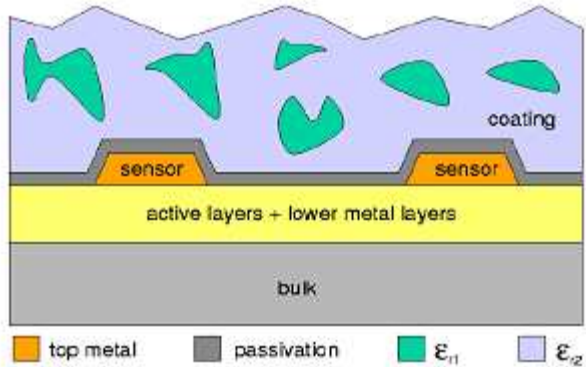
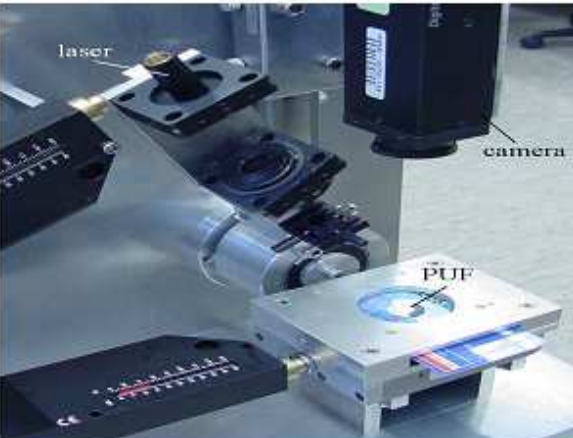

Physical Unclonable Function

- **PUF = Physical Unclonable Function:** Derive strings from a complex physical system that is inherently unclonable
 - Easy to evaluate (by probing the physical system)
 - Inherently tamper evident
 - Manufacturer not-reproducible
 - Noisy

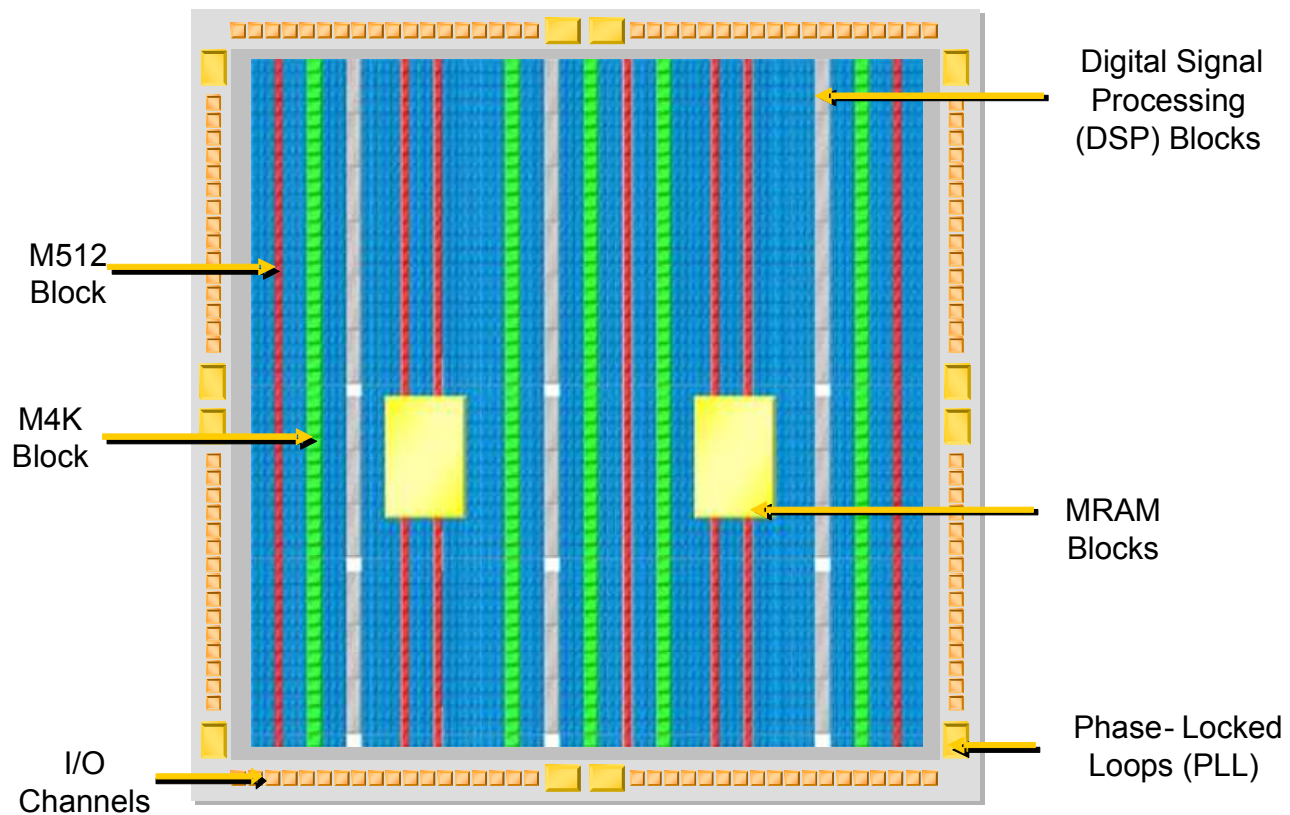
- **Unclonable**
 - Hard to make a physical clone
 - Hard to make a mathematical model that simulates the behavior of the physical structure

- **Practicality Requirements**
 - Easy to challenge the source
 - Cheap and easy integrable on an IC
 - Excellent mechanical and chemical properties

Examples of PUFs

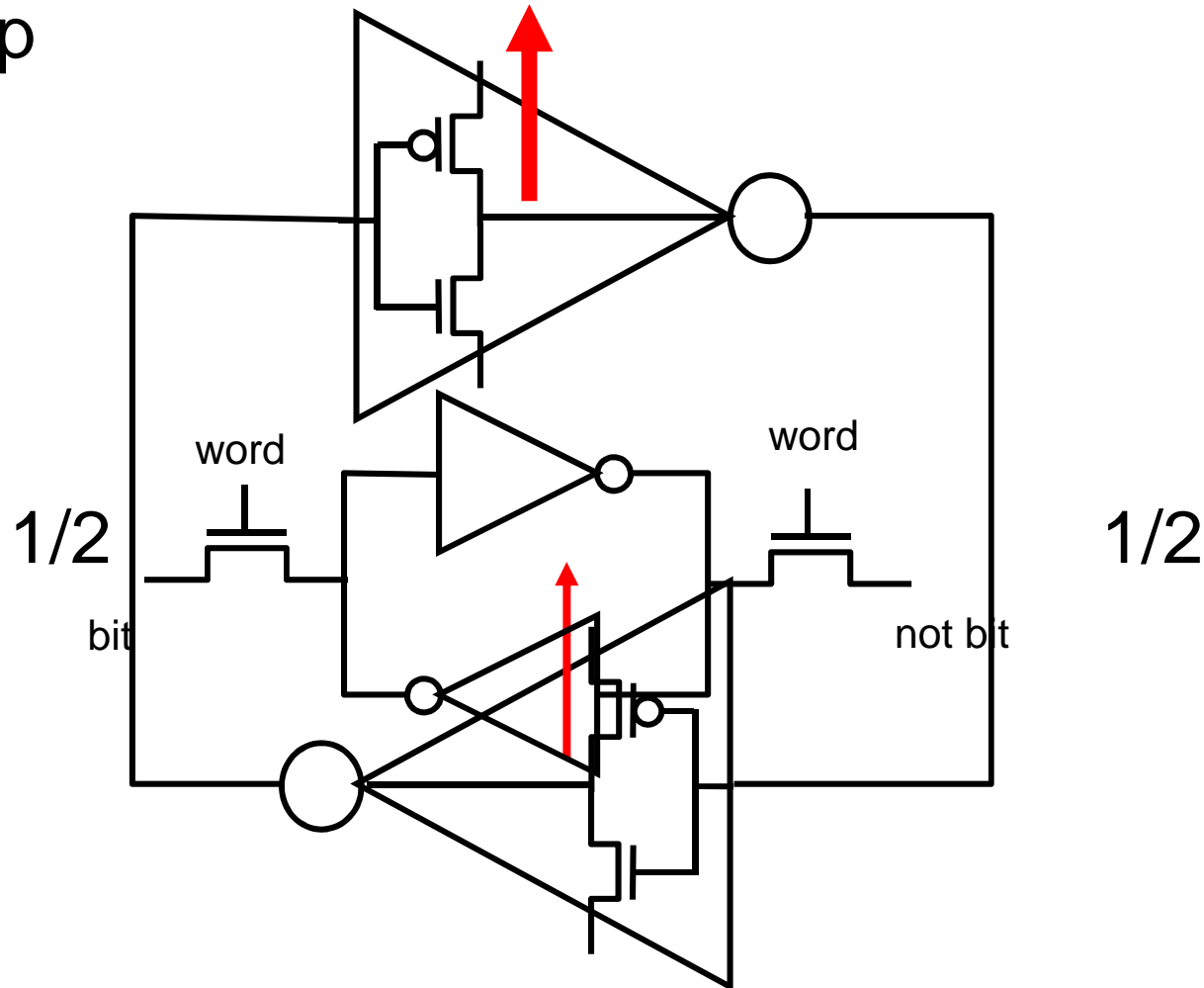
Coating PUFs	Optical PUFs	Intrinsic PUFs
<p>Measuring the capacitances of a coating with random dielectric particles</p>	<p>Disordered structures illuminated by a laser beam</p>	<p>Intrinsic device variations. Examples: memory-based and delay-based PUFs</p>
		

Modern FPGA Floorplan



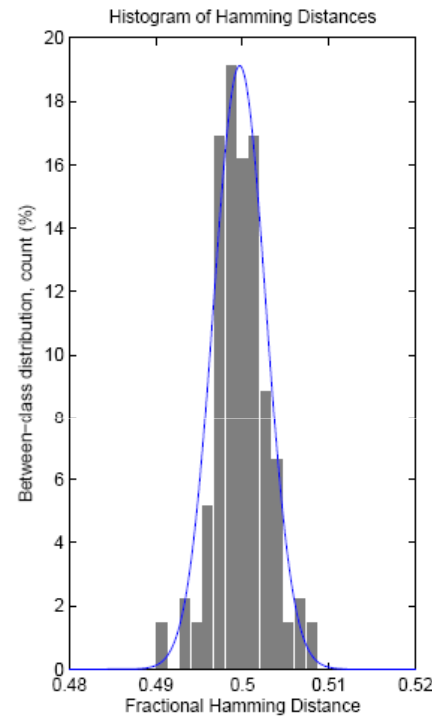
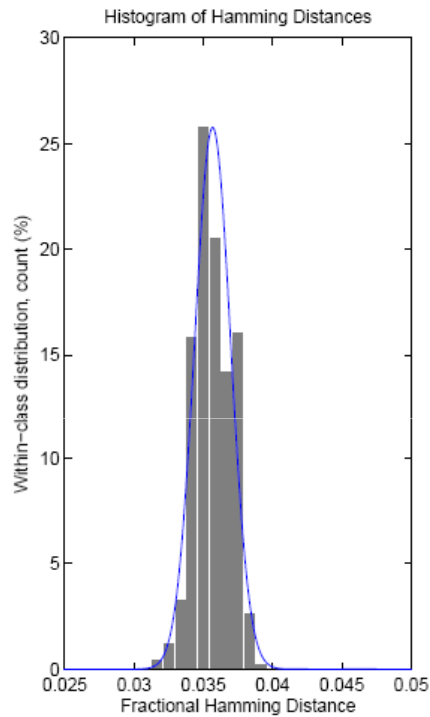
S-RAM PUF

Power-up

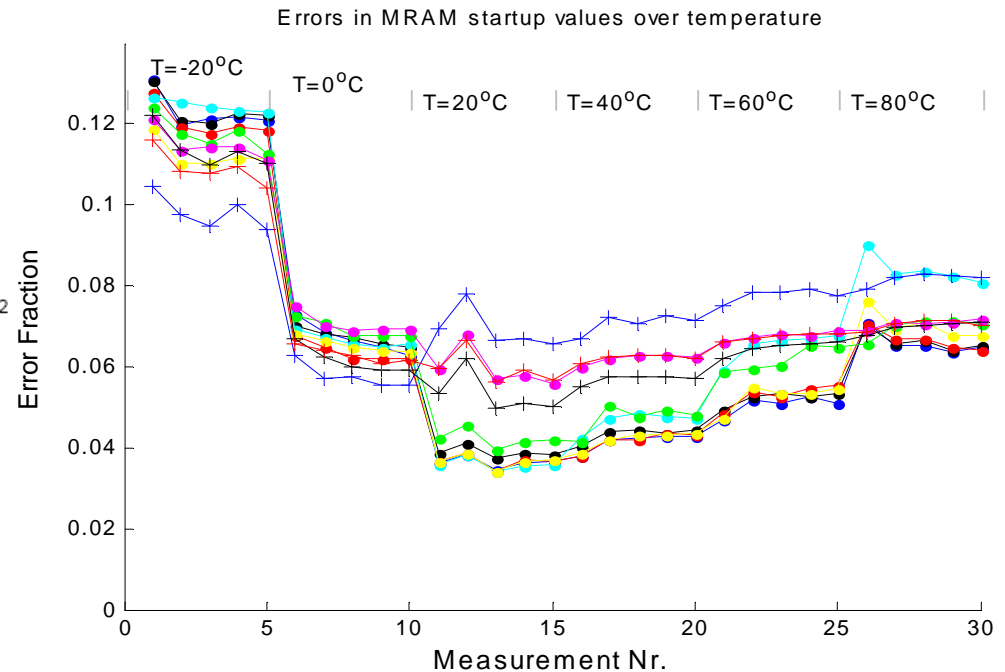


J. Guajardo, S. S. Kumar, G.-J Schrijen, P. Tuyls, FPGA Intrinsic PUFs and Their Use for IP Protection, CHES 2007, also see Daniel Holomb, Wayne Burleson, Kevin Fu, RFID Sec 2007

SRAM-Based PUF Performance



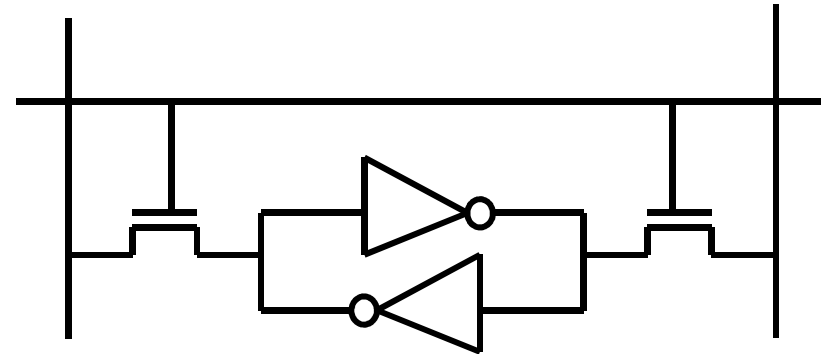
Entropy: 0.95 bits for every SRAM cell



One Catch

Problem:

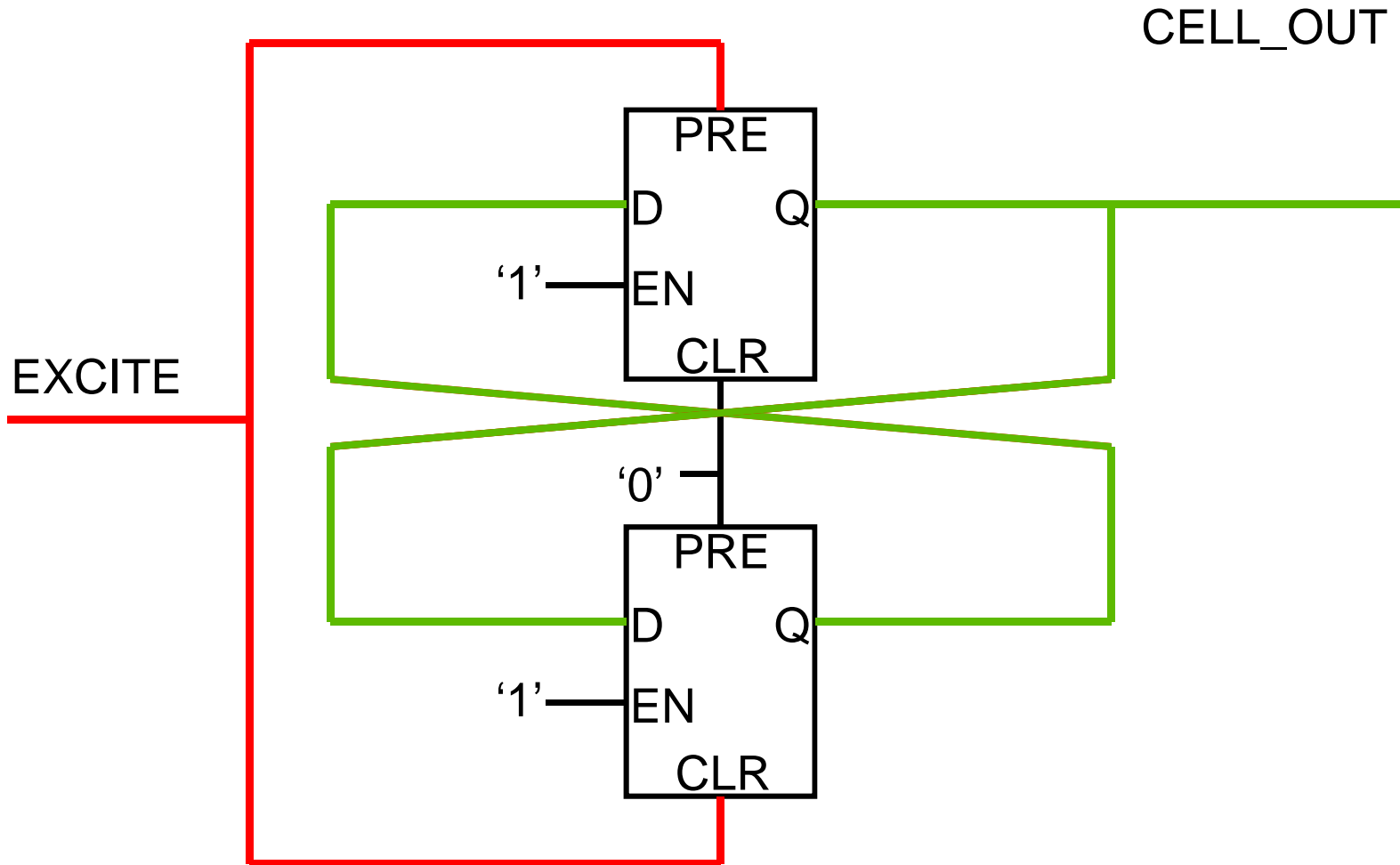
- Not all FPGAs contain uninitialized SRAM memory



Solution:

- Try to simulate SRAM cell with intrinsic FPGA components
- FPGAs have lots of latches!

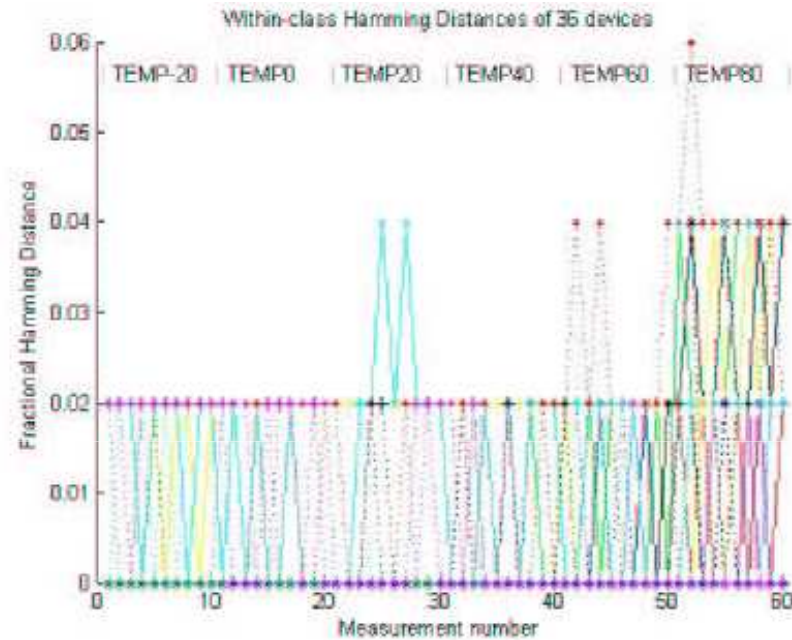
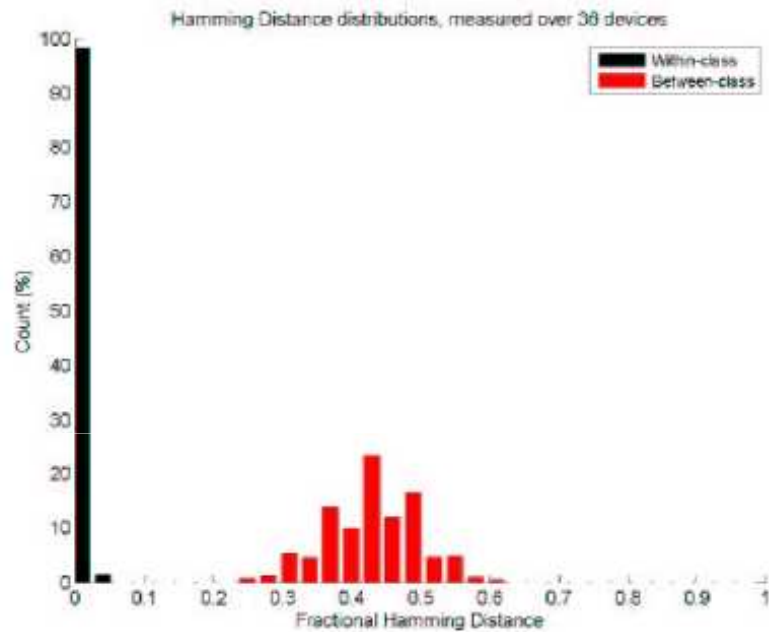
Butterfly-PUF-cell



S. S. Kumar, J. Guajardo, Roel Maes, G.-J Schrijen,
 P. Tuyls, FPGA Intrinsic PUFs and Their Use for IP.
 Protection, HOST 2008

Butterfly-PUF-cell

Butterfly-PUF Performance



Entropy: 0.78 bits for every Butterfly-PUF cell

- => 50 bit identifier for 130 slices
- => 128 fully random key for 1500 slices

Contents

Relevance

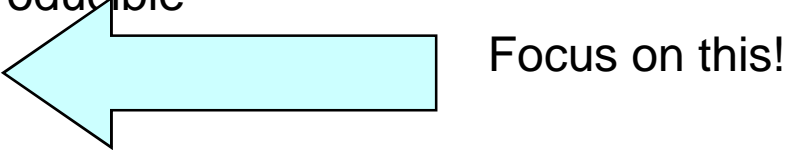
Physical Unclonable Functions

Helper Data Algorithms (on FPGAs)

Applications

Conclusions

Remember this?

- **PUF = Physical Unclonable Function:** Derive strings from a complex physical system that is inherently unclonable
 - Easy to evaluate (by probing the physical system)
 - Inherently tamper evident
 - Manufacturer not-reproducible
 - Noisy
 - **Unclonable**
 - Hard to make a physical clone
 - Hard to make a mathematical model that simulates the behavior of the physical structure
 - **Practicality Requirements**
 - Easy to challenge the source
 - Cheap and easy integrable on an IC
 - Excellent mechanical and chemical properties
- 

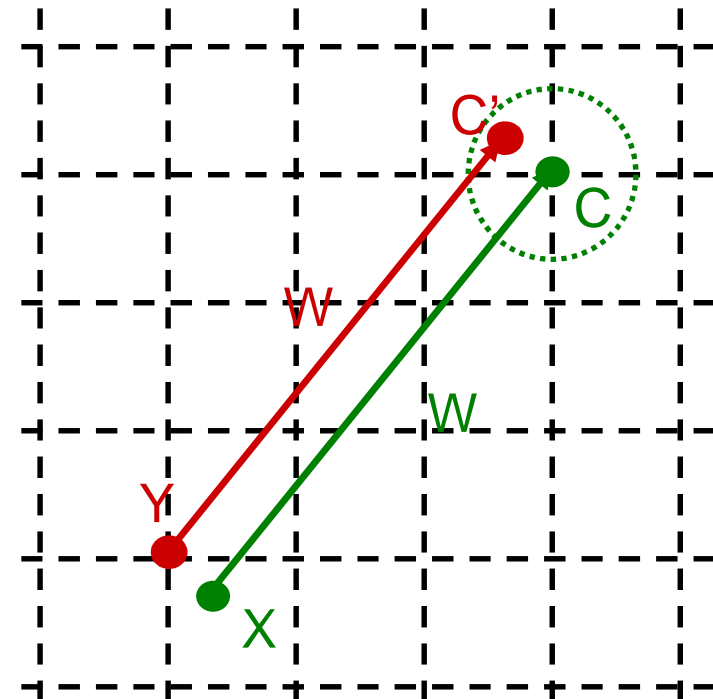
Helper Data Algorithm

- Grid points represent ECC Code words
- Enrollment**
- Random codeword $C(S)$ is chosen
- Response X is measured
- Helper data W (difference between X and C) is stored in NV-memory

Assumption: Response X uniformly random

Key Reconstruction

- Y is noisy response
- $Y+W=C'$
- $S'=DEC(C')$



Juels et al. CCS'99, Linnarz & Tuyls AVBPA 2003,
 Dodis et al. Eurocrypt 2004

Helper Data Algorithm Implementation on FPGAs

- Experimentally observe that errors occurring in PUF responses are random.
- We model errors as random and independent (BSC). So for strings of size n , the probability than more than t errors occur can be computed as

$$P_{t \text{ errors}} = \binom{n}{t} p_b^t (1 - p_b)^{n-t} \Rightarrow P_{Total} = \sum_{i=t+1}^n \binom{n}{i} p_b^i (1 - p_b)^{n-i}$$

- Set $P_{Total} = 10^{-6}$
- First try:
 - Choose error correcting code good at performing random error correction (as opposed to burst error correction)
 - Consider, repetition, Reed-Muller, Golay and BCH codes

C. Boesch, J. Guajardo, J. Shokrollahi, A.-R. Sadeghi, P. Tuyls, Efficient Helper Data Key Extractor on FPGAs, To be presented at CHES 2008

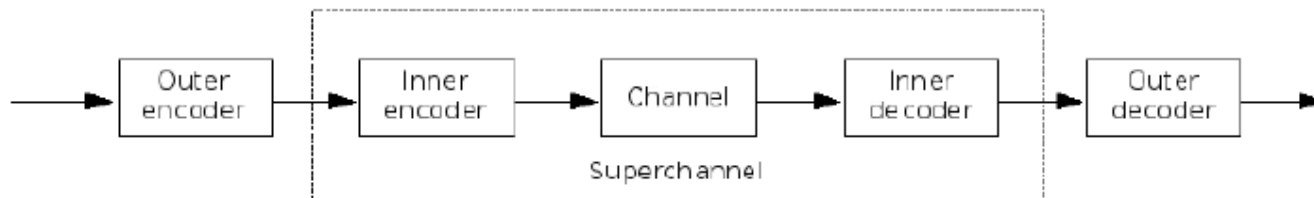
First Try:

- Assume entropy is 0.75 bits per every output bit.
- Assume $p_b=0.15$
- Wish to obtain a 128-bit uniformly distributed and random keys

Code	$[n, k, d]$	$t = \lfloor d/2 \rfloor$	P_{total}	source bits for 171 bits
Repetition	[33, 1, 33]	16	1.0010^{-6}	5643
Reed-Muller	[256, 9, 128]	63	2.0410^{-5}	4864
Reed-Muller	[512, 10, 256]	127	2.5410^{-9}	9216
Golay	[23, 12, 7]	3	0.4604	345
BCH	[511, 19, 239]	119	2.9710^{-7}	4599
BCH	[1023, 46, 439]	219	1.8510^{-8}	4092
BCH	[1020, 43, 439]*	219	1.4410^{-8}	4080

Can we do better?

- In terms of number of source bits
- In terms of potential hardware resources: BCH Coders/Encoders tend to be big.
- We do not care too much about performance (speed)
- **Idea:**
 - **Use concatenated codes (Forney '66)**



Results for Concatenated Codes

C_2 $[n_2, k_2, d_2]$	C_1 $[n_1, k_1, d_1]$	P_1	source bits for 171 bits
repetition [3, 1, 3]	<i>BCH</i> [127, 29, 43]	8.48 E-06	2286
	<i>RM</i> [64, 7, 32]	1.02 E-06	4800
	<i>BCH</i> [63, 7, 31]	8.13 E-07	4725
repetition [5, 1, 5]	<i>RM</i> [32, 6, 16]	1.49 E-06	4640 ←
	<i>BCH</i> [226, 86, 43]*	2.28 E-07	2260 ←
repetition [7, 1, 7]	G_{23} [23, 12, 7]	1.58 E-04	2415
	G_{23} [20, 9, 7]*	8.89 E-05	2660
	<i>BCH</i> [255, 171, 23]	8.00 E-05	1785
	<i>RM</i> [16, 5, 8]	3.47 E-05	3920
	<i>BCH</i> [113, 57, 19]*	1.34 E-06	2373
repetition [9, 1, 9]	<i>BCH</i> [121, 86, 11]	6.84 E-05	2178
	G_{23} [23, 12, 7]	8.00 E-06	3105 ←
	<i>RM</i> [16, 5, 8]	1.70 E-06	5040
repetition [11, 1, 11]	G_{24} [24, 13, 7]	5.41 E-07	3696
	G_{23} [23, 12, 7]	4.52 E-07	3795

Hardware Cost

Code / Hash	Block Output (bits)	Slices	Latency (cycles)	Critical Path (nsec)	Performance for 128 bit key 50 MHz (sec)
Repetition [3, 1, 3]	3	41 (1%)	6	5.3	$2.1 \cdot 10^{-6}$
Repetition [5, 1, 5]	5	41 (1%)	10	5.3	$3.4 \cdot 10^{-6}$
Repetition [7, 1, 7]	7	41 (1%)	14	5.3	$4.8 \cdot 10^{-6}$
Repetition [9, 1, 9]	9	41 (1%)	18	5.3	$6.2 \cdot 10^{-6}$
Repetition [11, 1, 11]	11	41 (1%)	22	5.3	$7.5 \cdot 10^{-6}$
$\mathcal{R}(1, 4)$	16	69 (1%)	503	5.5	$3.5 \cdot 10^{-4}$
$\mathcal{R}(1, 5)$	32	90 (1%)	1743	5.6	$1.0 \cdot 10^{-3}$
$\mathcal{R}(1, 6)$	64	127 (1%)	6495	5.6	$3.2 \cdot 10^{-3}$
Golay G_{24}	24	538 (5%)	1188	6.6	$3.6 \cdot 10^{-4}$
Toeplitz Hash 16 [24]	128	319 (3%)	64	5.7	$1.0 \cdot 10^{-6}$
Toeplitz Hash 24 [24]	128	327 (3%)	96	5.7	$1.2 \cdot 10^{-6}$
Toeplitz Hash 32 [24]	128	335 (3%)	128	5.7	$1.0 \cdot 10^{-6}$
Toeplitz Hash 64 [24]	128	367 (3%)	256	5.7	$1.0 \cdot 10^{-6}$

Contents

Relevance

Physical Unclonable Functions

Helper Data Algorithms (on FPGAs)

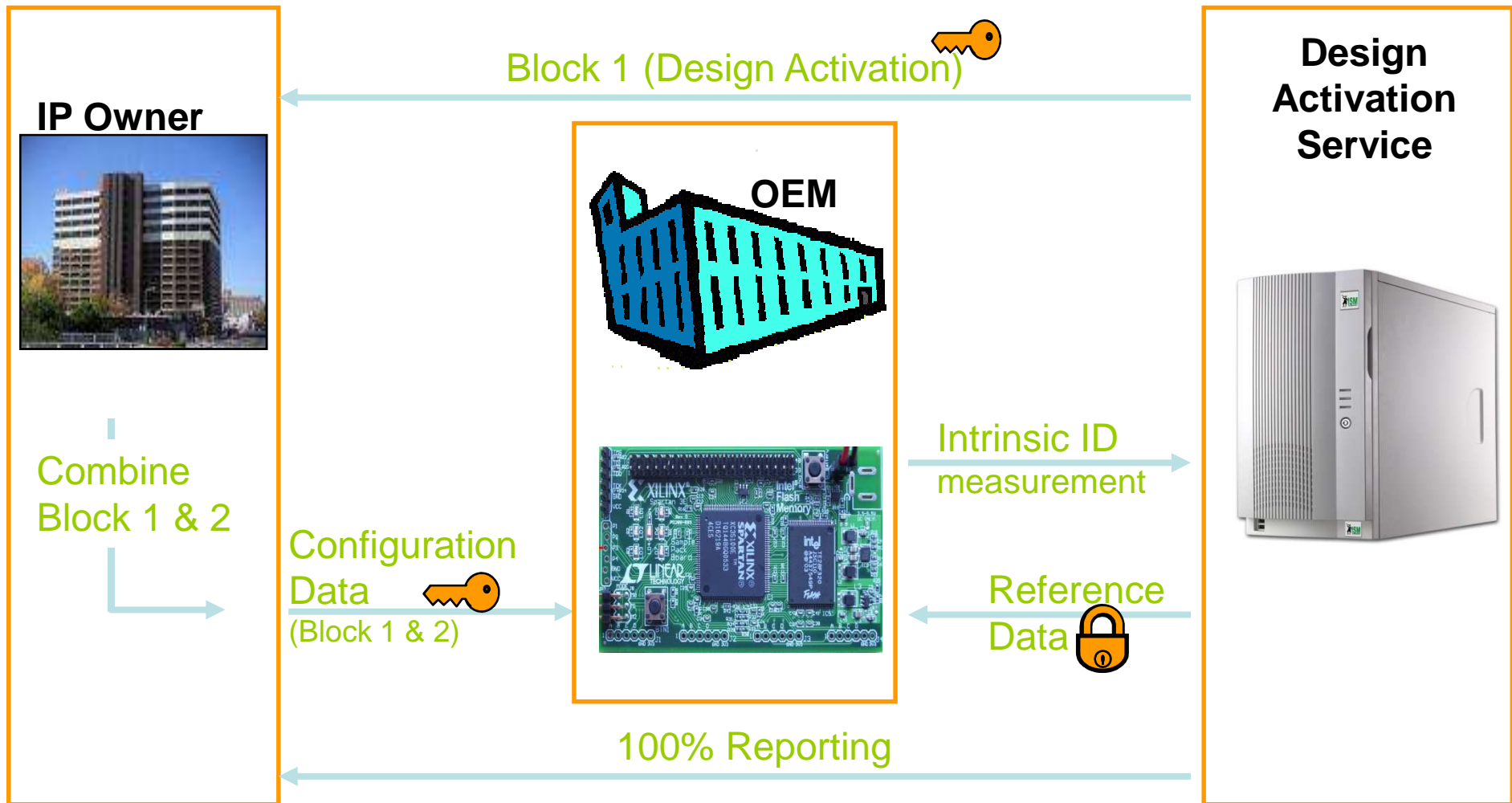
Applications

Conclusions

Secure Key Storage

1. Do **not** store a key in digital form in a device
2. Generate the key **only when needed**
(extract it from a physical source on the IC)
3. **Delete** the key

Design Activation Service



Contents

Relevance

Physical Unclonable Functions

Helper Data Algorithms (on FPGAs)

Applications

Conclusions

Conclusions

- Many PUF constructions suitable for different situations and devices.
- PUF noisy nature requires helper data algorithm. There are efficient manners to do this
- Applications to secret-key storage, IP protection, service activation, authentication, and many others.

PHILIPS

