TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik, Institut für Technische Informatik, Professur für VLSI-Entwurfssysteme, Diagnostik und Architektur

Virtualized Reconfigurable Resources and Their Secured Provision in an Untrusted Cloud Environment

Verteidigung der Diplomarbeit

Paul R. Genßler – paul.genssler@tu-dresden.de

Dresden, 29. November 2017

1. Motivation und Literatur

2. Design

3. Implementierung

4. Ergebnisse

5. Zusammenfassung und Ausblick

Virtualized Reconfigurable Resources

- Flexibel konfigurierbare Hardware
- Beschleunigung von Anwendungen
- FPGAs in der Cloud
  - Microsoft Catapult, Amazon EC2 F1 Instances
  - Virtualisierung nötig

Chen u. a. Virtualisierung mit OpenStack[1]

Asiatici u. a. Virtualisierter Speicher als HW/SW Schnittstelle[2]

Knodel u.a. Flexible Partitionierung mit Service Modellen[3]

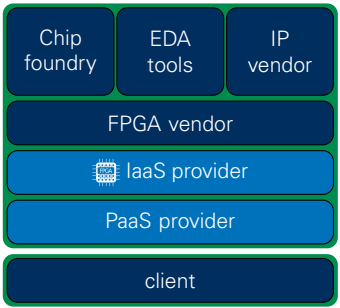
[1] Chen u. a., 2014: "Enabling FPGAs in the cloud"

[2] Asiatici u. a., 2016: "Designing a virtual runtime for FPGA accelerators in the cloud"

[3] Knodel u.a., 2017: "Virtualizing Reconfigurable Hardware to Provide Scalability in Cloud Architectures"

Secured Provision

- 58 %: Verlust der Herrschaft über die Daten[4]
- Kein Schutz des geistigen Eigentums
- Vielschichtige Umgebung
  → Viele Einfallstore
- Sichere Übertragung
  – TLS
  – Devic u. a. 2010[5]



[4] Heidkamp u. a., 2016: *Cloud-Monitor 2016*

[5] Devic u. a., 2010: "Secure protocol implementation for remote bitstream update preventing replay attacks on FPGA"

Cloud Environment

Cloudanbieter (IaaS, PaaS Provider)

- Ressourcenverwalter
- Flexibilität, Skalierbarkeit, schnelle Bereitstellung
- Geringer CPU Overhead

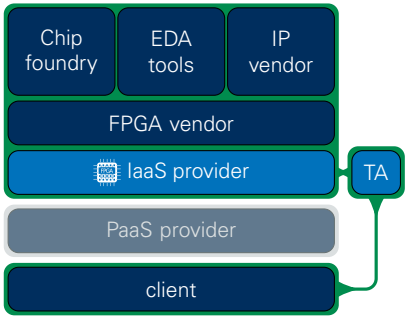
Mittelständisches Unternehmen

- Auslagerung der Datenverarbeitung in die Cloud
- Hoher Durchsatz, viel Speicher
- Sensible Daten → keine Public Cloud
- Sicherer, authentifizierbarer Endpunkt

Sicherheitskonzepte

- Trusted Platform Model (TPM)
- Homomorphe Verschlüsselung

- FPGA Sicherheitskonzept vorhanden
- Einbeziehung einer Trusted Authority
  - Kepa u. a. 2008[6]
  - Eguro u. a. 2012[7]



[6] Kepa u. a., 2008: "Serecon: A secure dynamic partial reconfiguration controller"
[7] Eguro u. a., 2012: "FPGAs for trusted cloud computing"

Ziele der Arbeit

- Literaturstudium relevanter kryptographischer Verfahren und bisheriger Arbeit zur Authentifizierung eines Bitstreams
- Konzeptionierung eines Host/FPGA-Hypervisors zur Ver- und Entschlüsselung partieller Bitstreams
- Prototypische Umsetzung des Konzepts auf Basis des RC2F
- Auswertung der Ergebnisse

2 Design

Gefahrenmodell

Level 5 Außerhalb des Rechenzentrums

Level 4 Virtueller Zugang zum Host

Level 3 Physischer Zugriff auf die Platine

Level 2 Benachbarte rekonfigurierbare Region

Level 1 Physischer Zugang direkt zum Chip

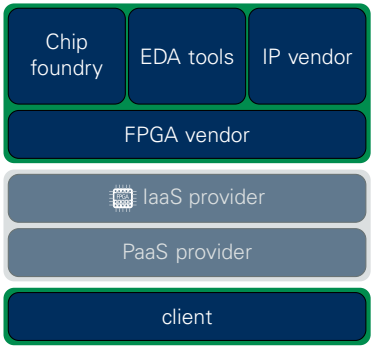
Level 0 Zugriff vor und während der Fertigung

2 Design

Anforderungsanalyse

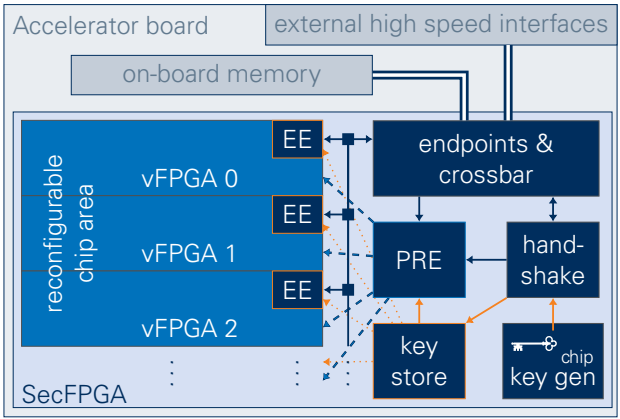
- Virtualisierung in vFPGAs
- Sicherer Transfer von Konfigurationen
- Kein Zugriff durch Cloud Provider
- Keine Trusted Authority nötig

SecFPGA



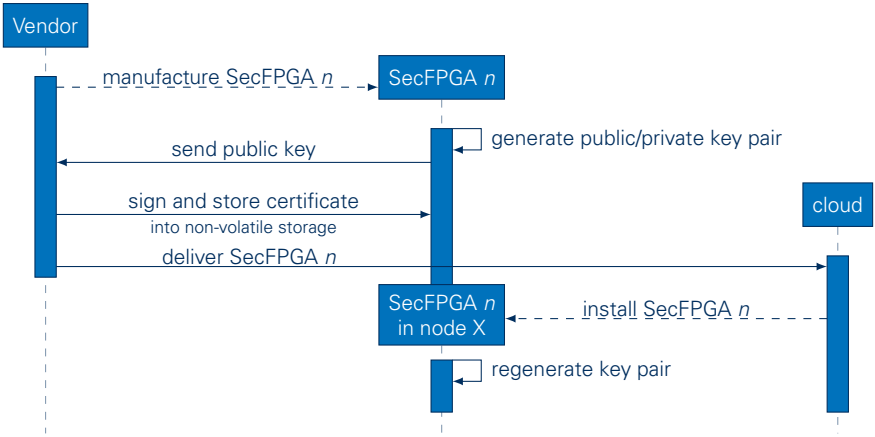
2 Design

SecFPGA Aufbau



2 Design

Erste Bereitstellung



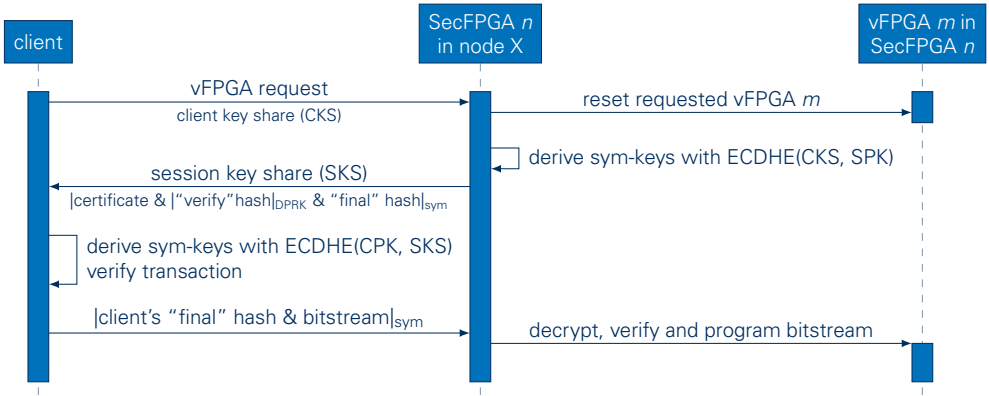
2 Design

TLS Protokoll

- Standard für sichere Kommunikation im Internet
- SSL 1.0 - 3.0, TLS 1.0 - 1.2
- TLS 1.3 in Entwurfsphase
- Viele Verschiedene Algorithmen (Cipher Suites)

2 Design

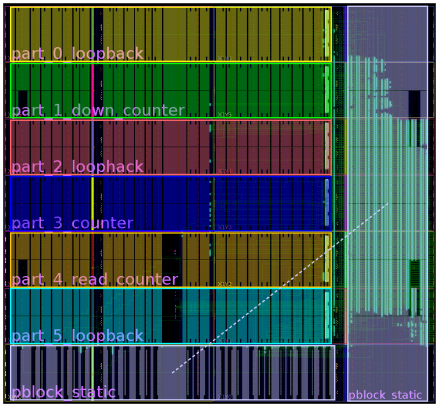
SecFPGA-Hypervisor Handshake

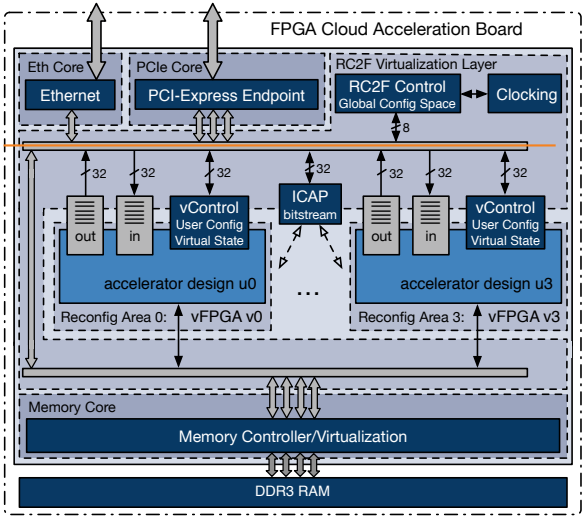


2 Design

Partielle Rekonfiguration

- 6 nutzbare Regionen
- Beeinflussung eines anderen Clients
- Format der Konfigurationsdatei basiert auf Frames
- 5 Adressbereiche
- Analyse in Hardware
- Filter für unerlaubte Bereiche

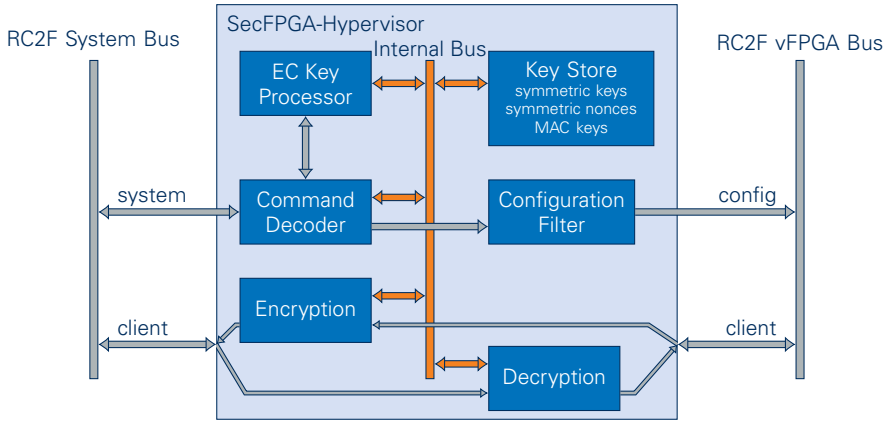




Knodel u. a. 2016 "RC3E: Reconfigurable Accelerators in Data Centres and their Provision by Adapted Service Models"

3 Implementierung

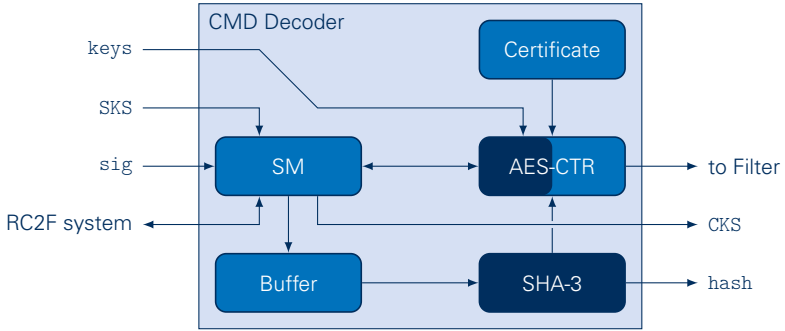
Integration des SecFPGA-Hypervisors



3 Implementierung

CMD Decoder - Blockschaltbild

- Sichere Übertragung der vFPGA Bitstreams
- Enge Kopplung mit dem EC Key Processor



EC Key Processor - Blockschaltbild

- Elliptische Kurven Kryptographie: 233-bit Binärkurve `sept233r1`
- Enge Kopplung mit dem CMD Decoder

Vorberechnungen

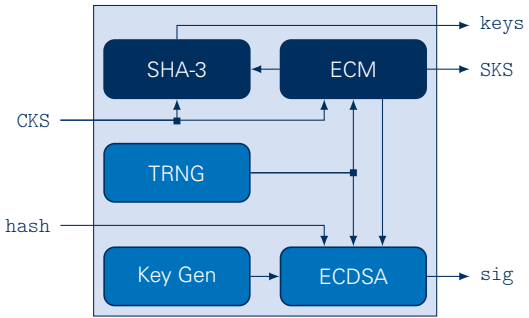
$$x_1 = ECM(k)$$

$$k^{-1} = k^{2^{233}-2} \mod x^{233} + x^{74} + 1$$

Handshake Berechnungen

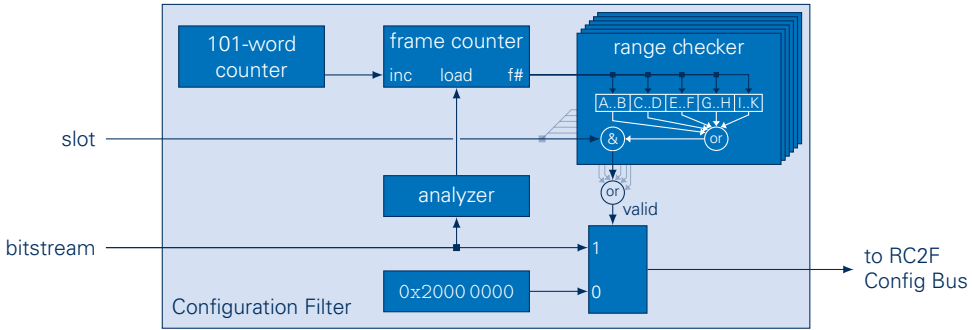
$$r = x_1 \mod n$$

$$s = k^{-1}(\mathrm{m} + r * \text{private key}) \mod n$$



Configuration Filter

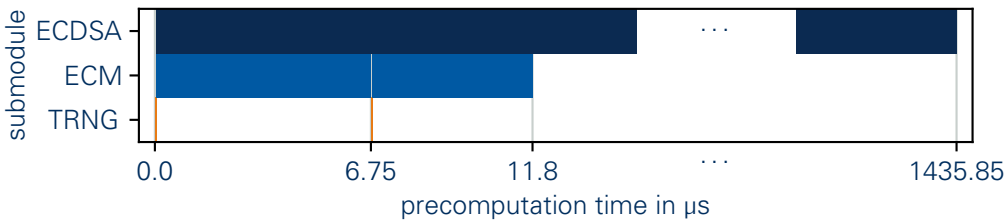
- Schutz des SecFPGAs und anderer vFPGAs
- Filter basierend auf Frameadressen



Vorberechnungen

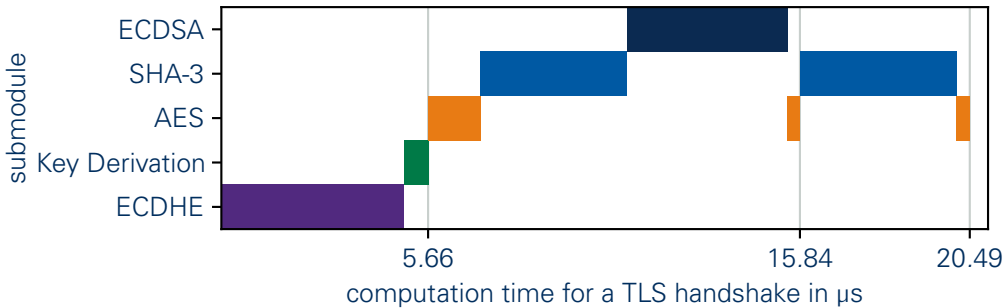
$$k^{-1} = k^{2^{233}-2} \mod x^{233} + x^{74} + 1$$

$$x_1 = ECM(k)$$



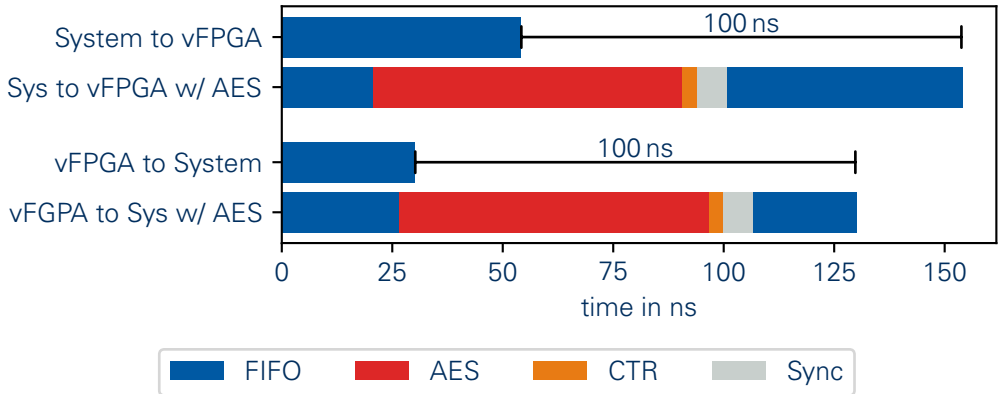
4 Ergebnisse

TLS Handshake



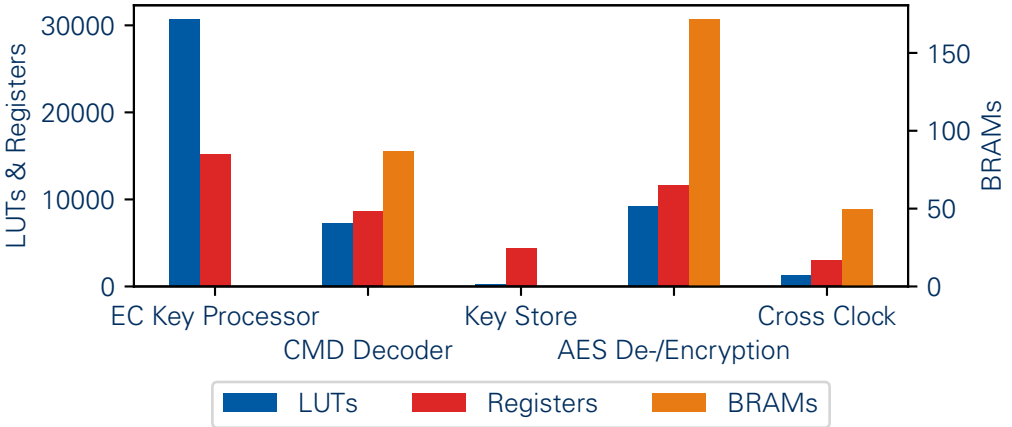
4 Ergebnisse

Latenzzeit



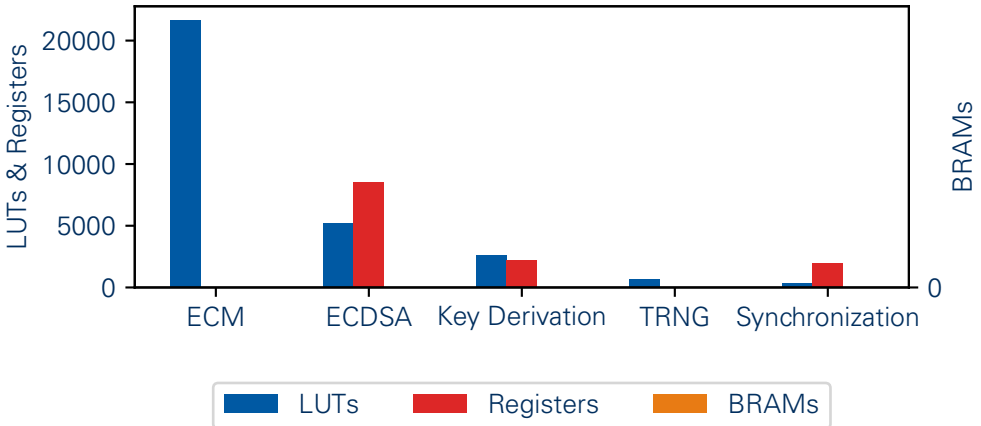
4 Ergebnisse

Ressourcen - SecFPGA-Hypervisor



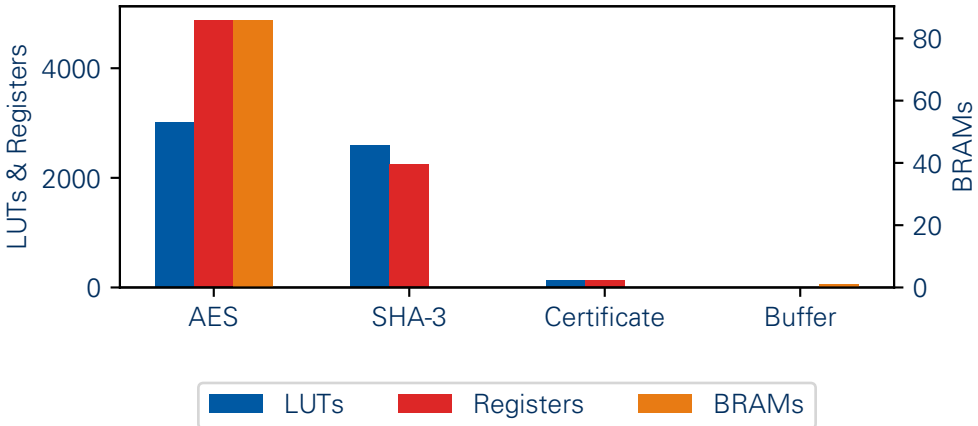
4 Ergebnisse

Ressourcen - EC Key Processor



4 Ergebnisse

Ressource - CMD Decoder



Herausforderung der sicheren FPGA-Virtualisierung

- Strikte Trennung von statischer SecFPGA-Hypervisor Infrastruktur und vFPGAs
- Eingebettete Sicherheits-Features
- Separate Taktnetze für jeden vFPGA
- vFPGA Konfigurationen ohne geteilte Adressbereiche

4 Ergebnisse

Sicherheitsanalyse

Level 5 Außerhalb des Rechenzentrums ✓

Level 4 Virtueller Zugang zum Host ✓

Level 3 Physischer Zugriff auf die Platine ✓

Level 2 Benachbarte rekonfigurierbare Region ✓

Level 1 Physischer Zugang direkt zum Chip ✓

Level 0 Zugriff vor und während der Fertigung ?

Zusammenfassung und Ausblick

Zusammenfassung

- Design des SecFPGAs bietet sehr hohes Sicherheitslevel
- Cloud Charakteristik durch Einsatz teilweise rekonfigurierbarer Hardware
- Nutzung etablierter Algorithmen und Protokolle
- Implementation eines Prototyps und Auswertung des Mehraufwands

Ausblick

- Flexible Bandbreiten
- Direkte Kommunikation über andere Schnittstellen
- Verifizierung einer Verbindung durch einen SecFPGA

7 Literatur

Quellen

Hsing, Homer (2013). *OpenCores - SHA3 Core*. URL: https://opencores.org/project,sha3 (besucht am 08. 09. 2017).

Hsing, Homer (2015). *OpenCores - Tiny AES*. URL: https://opencores.org/project,tiny_aes (besucht am 06. 09. 2017).

Mukhopadhyay, Debdeep u. a. (2008). *Elliptic Curve Crypto Processor for FPGA Platforms*. URL: http://cse.iitkgp.ac.in/~debdeep/osscrypto/eccpweb/index.html (besucht am 27. 08. 2017).

TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik, Institut für Technische Informatik, Professur für VLSI-Entwurfssysteme, Diagnostik und Architektur

Virtualized Reconfigurable Resources and Their Secured Provision in an Untrusted Cloud Environment

Verteidigung der Diplomarbeit

Paul R. Genßler – paul.genssler@tu-dresden.de

Dresden, 29. November 2017

Asymmetrische Verschlüsselung

- RSA, ECIES
- Privater Schlüssel: Entschlüsselung
- Öffentlicher Schlüssel: Verschlüsselung
- Basiert auf mathematischen Problemen ohne effiziente Lösung
- Schlüsselgrößen: 2048 Bit (RSA), 256 Bit (ECIES)

Digitale Signatur

- RSA, ECDSA
- Öffentlicher Schlüssel: Verifizierung
- Privater Schlüssel: Signierung
- Basiert auf gleichen Prinzipien wie asymmetrische Verschlüsselung
- Schlüsselgrößen: 2048 Bit (RSA), 256 Bit (ECDSA)

Symmetrische Verschlüsselung

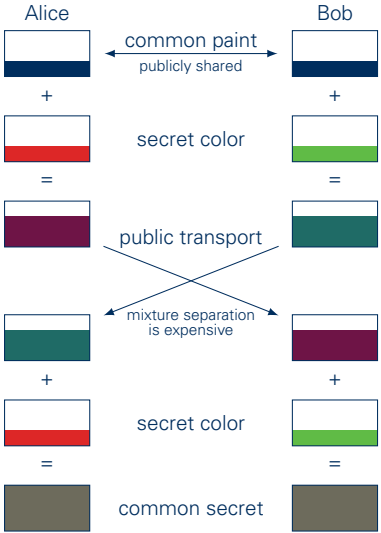
- AES, DES, KASUMI
- RC5, Chacha20
- Gleicher Schlüssel zum Ver- und Entschlüsseln
- Konfusion und Diffusion [8]
- Schlüsselgrößen: 128 - 256 Bit

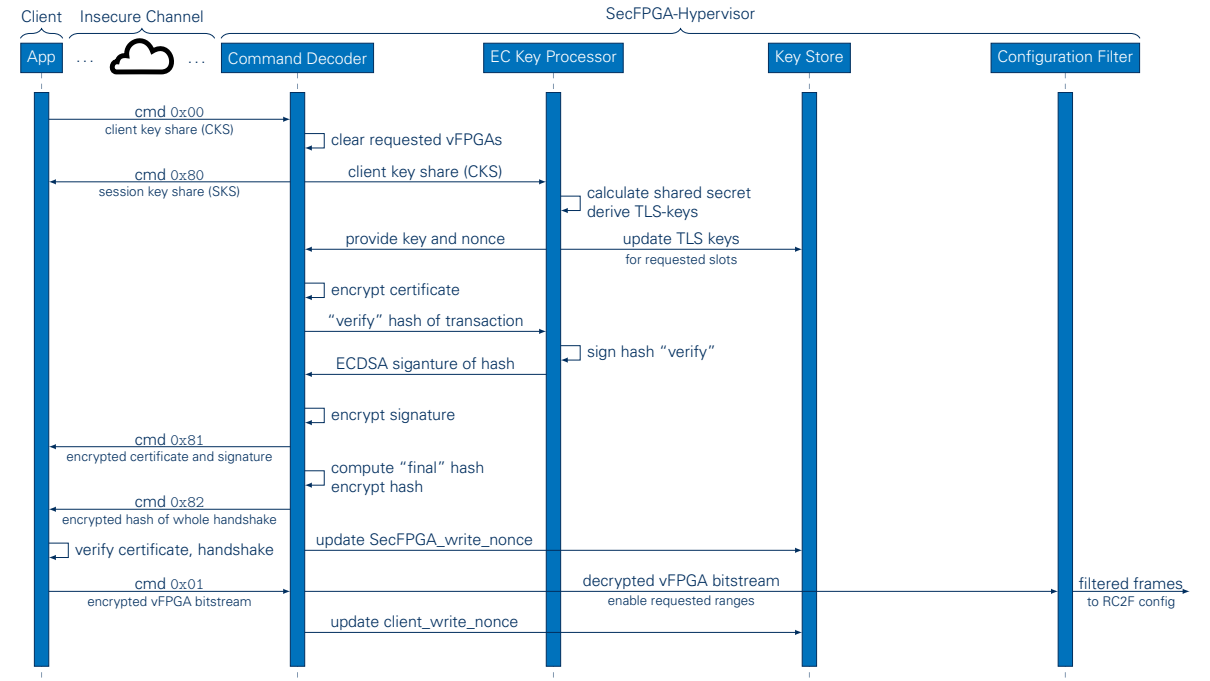
[8] Shannon, 1945: "A Mathematical Theory of Cryptography"

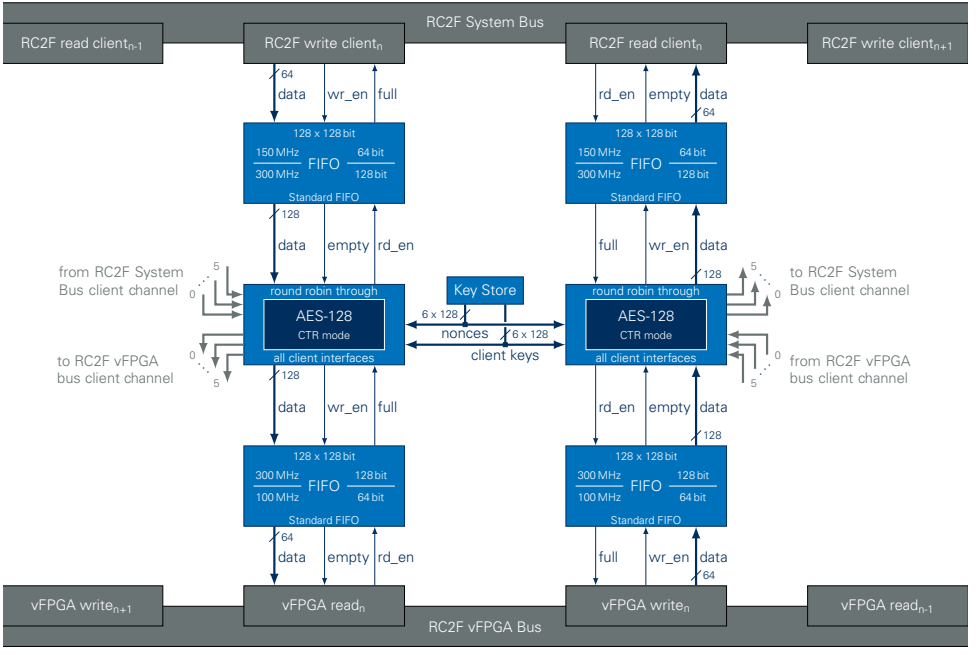
Hashfunktionen

$$h(\text{📄}) = 0x4914D65BC6140743C5B6719A79746DB1_{16}$$

- MD5, SHA1, SHA224, SHA256, SHA3-512
- Gleichverteilte Abbildung beliebig langer Daten auf Hash fester Länge
- Gleiche Eingabe erzeugt gleiche Ausgabe
- Konfusion und Diffusion
- Kollisionsresistenz
- Rekonstruktion der Eingabe praktisch unmöglich


Alice
Bob
common paint
publicly shared
+
+
secret color
=
=
public transport
mixture separation
is expensive
+
+
secret color
=
=
common secret


Client
Insecure Channel
SecFPGA-Hypervisor
App
Command Decoder
EC Key Processor
Key Store
Configuration Filter
cmd 0x00
client key share (CKS)
clear requested vFPGAs
client key share (CKS)
cmd 0x80
session key share (SKS)
calculate shared secret
derive TLS-keys
provide key and nonce
update TLS keys
for requested slots
encrypt certificate
“verify” hash of transaction
sign hash “verify”
ECDSA siganture of hash
encrypt signature
cmd 0x81
encrypted certificate and signature
compute “final” hash
encrypt hash
cmd 0x82
encrypted hash of whole handshake
update SecFPGA_write_nonce
verify certificate, handshake
cmd 0x01
encrypted vFPGA bitstream
decrypted vFPGA bitstream
enable requested ranges
filtered frames
to RC2F config
update client_write_nonce


RC2F System Bus
RC2F read client$_{n-1}$
RC2F write client$_n$
RC2F read client$_n$
RC2F write client$_{n+1}$
64
data
wr_en
full
128 x 128 bit
150 MHz
300 MHz
FIFO
64 bit
128 bit
Standard FIFO
128
data
empty
rd_en
rd_en
empty
data
64
full
wr_en
data
128
from RC2F System Bus client channel
to RC2F System Bus client channel
5
0
round robin through
AES-128
CTR mode
all client interfaces
Key Store
6 x 128
nonces
client keys
to RC2F vFPGA bus client channel
from RC2F vFPGA bus client channel
300 MHz
100 MHz
vFPGA write$_{n+1}$
vFPGA read$_n$
vFPGA write$_n$
vFPGA read$_{n-1}$
RC2F vFPGA Bus

11 Ergebnisse

TLS - Vorberechnungen

Step	Depends On	Module	Action	Start Time (µs)	Duration (µs)
1	–	TRNG	Generate 1st integer	0,00	0,05
2	1	ECDSA	Invert 1st integer	0,05	1429,10
3	1	ECM	Curve point 1st integer	0,05	6,70
4	–	TRNG	Generate 2nd integer	6,75	0,05
5	4	ECM	Curve point 2nd integer	11,80	5,00

TLS - Berechnungen

Step	Depends On	Module	Action	Start Time (µs)	Duration (µs)
1	CKS	ECDHE	complete key exchange	0,00	5,00
2	1	ECKP	key derivation	5,00	0,66
3	2	AES	encrypt certificate	5,66	1,43
4	3	SHA-3	compute “verify” hash	7,09	4,01
5	4	ECDSA	sign “verify” hash	11,10	4,40
6	5	AES	encrypt signature	15,50	0,34
7	6	SHA-3	compute “final” hash	15,84	4,28
8	7	AES	encrypt “final” hash	20,12	0,37

Latencies

Action	Source (MHz)	Destination (MHz)	Stages	Latency (ns)
AES	300	300	21	70,00
CTR mode	300	300	1	3,33
Round-Robin	300	300	2	6,67
RC2F vFPGA Bus to AES	100	300	2	26,67[a]
AES to RC2F System Bus	300	250	2	23,33[a]
RC2F System Bus to AES	250	300	2	20,67[a]
AES to RC2F vFPGA Bus	300	100	2	53,33[a]
RC2F vFPGA to System Bus[b]	100	250	2	30,00[a]
RC2F System to vFPGA Bus[b]	250	100	2	54,00[a]

[a] worst case, Xilinx Inc., 2017: *FIFO Generator v13.1* [b] without AES

11 Ergebnisse

Ressourcen - SecFPGA-Hypervisor

Submodule	LUTs		Registers		BRAMs	
EC Key Processor	30 766	10,13%	15 158	2,50%	0	0,00%
CMD Decoder	7279	2,40%	8714	1,44%	87	8,45%
Key Store	269	0,09%	4379	0,72%	0	0,00%
Configuration Filter	119	0,04%	99	0,02%	0	0,0 %
AES encryption	4612	1,52%	5820	0,96%	86	8,35%
AES decryption	4595	1,51%	5820	0,96%	86	8,35%
Cross clock FIFOs	1358	0,44%	3000	0,48%	50	4,85%
Overall	48 878	16,10%	42 891	7,06%	309	30,00%

A XC7VX485T is equipped with 303 600 LUTs, 607 200 registers and 1030 BRAMs among others.

Ressourcen - CMD Decoder

Submodule	LUTs		Registers		BRAMs	
AES	3011	41,37%	4886	56,07%	86	98,85%
Buffer	0	0,00%	0	0,00%	1	1,15%
SHA-3	2598	35,69%	2245	25,76%	0	0,00%
Certificate	128	1,76%	128	1,47%	0	0,00%
CMD Decoder	7279	100,00%	8714	100,00%	87	100,00%

Ressourcen - EC Key Processor

Submodule	LUTs		Registers		BRAMs	
ECM	21 695	70,52 %	48	0,32 %	0	0,00 %
ECDSA	5243	17,04 %	8510	56,14 %	0	0,00 %
Key derivation	2601	8,45 %	2245	14,81 %	0	0,00 %
TRNG	670	2,18 %	34	0,22 %	0	0,00 %
Synchronization	319	1,07 %	1944	12,82 %	0	0,00 %
EC Key Processor	30 766	100,00 %	15 158	100,00 %	0	0,00 %

11 Ergebnisse

Ressourcen - AES

Submodule	LUTs		Registers		BRAMs	
AES-Core	3178	69,16%	3968	68,18%	86	100,00%
Arbiter	1417	30,84%	1852	31,82%	0	0,00%
Overall	4595	100,00%	5820	100,00%	86	100,00%

11 Ergebnisse

Ressourcen - Literatur

Submodule	Reference/ Optimization	Slices		BRAMs		DSP	
		Ref.	this	Ref.	this	Ref.	this
ECM	Sasdrich u. a. 2014	1029	5765	2	0	20	0
Hash	Garcia u. a. 2014	139	715	0	0	0	0
Key derivation	*reuse*	0	714	0	0	0	0
AES encryption	Zhou u. a. 2009	4628	2334	0	86	0	0
AES decryption	Zhou u. a. 2009	4628	2052	0	86	0	0
TLS encryption[a]	*reuse*	0	1226	0	86	0	0
Savings overall			2382		256		-20

[a] De-/Encryption of handshake traffic and bitstream