ETH zürich

B. ROTHENBERGER, K. TARANOV, A. PERRIG, T. HOEFLER

ReDMArk: Bypassing RDMA Security Mechanisms

Usenix Security 2021

SPCL

ETH zürich
Network Security Group

Collaborators for this work

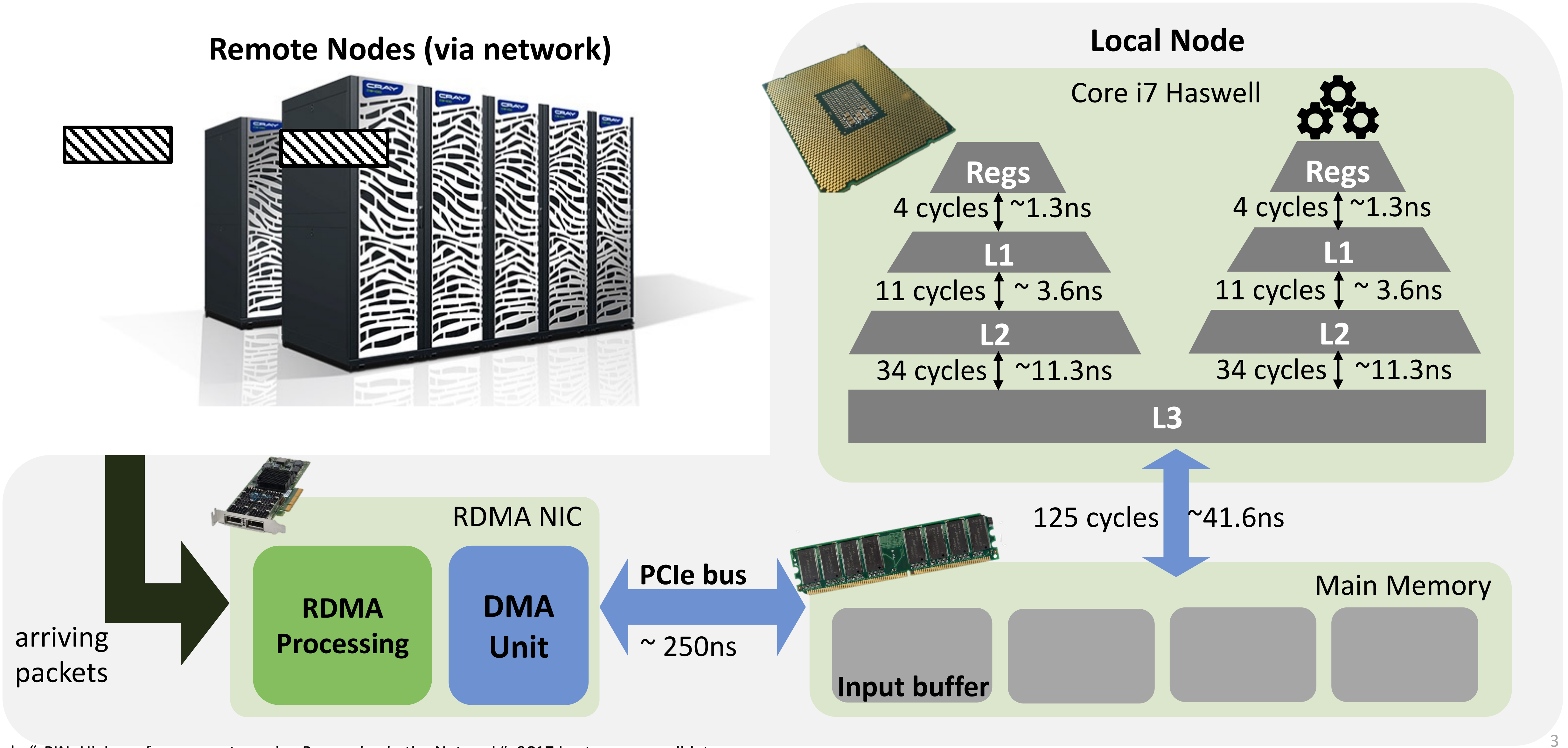
Benjamin Rothenberger
PhD Student
Netsec

Konstantin Taranov
PhD Student
SPCL

Adrian Perrig
Professor
Netsec

Torsten Hoefler
Professor
SPCL

Data Processing in Modern RDMA Networks



TH et al.: "sPIN: High-performance streaming Processing in the Network", SC17 best paper candidate

RDMA is a Trending Topic in HPC and Cloud Systems

DSLR'18
DrTM+H'18
NAM-DB'17
FaRM'14
HERD'14
CoRM'21
Wukong'16
RAMCloud'15
Octopus'17
RDMP-KV'20
Catfish'19
FileMR'14
FaSST'16
XSTORE'20
ccNUMA'18
RDMA
HydraDB'15
Derecho'19
Hermes'20
Grappa'15
A1'20
DrTM+R'16
Crail'19
SparkRDMA'14
C-Hint'14
ScaleRPC'19
Dare'15
Storm'19
DaRPC'22
DrTM'15
TH-DPMS'16
Hyperloop'18
APUS'17

designed for performance – lower latency, higher bandwidth, lower CPU utilization etc.

Vulnerabilities? **Exploits?** **Mitigations?**

ReDMArk Overview

4 adversary models | **10 vulnerabilities** | **8 mitigations**

Implemented 6 attacks

	T1	T2	T3	T4	V1 weak rkey	V2 static init.	V3 shared key gen.	V4 weak mem. rand.	V5 lin. inc. QPN	V6 fixed starting PSN	V7 lim. attack detect.	V8 no enc./auth.	V9 single PD	V10 ODP enabled	M1 rand. QPN	M2 rand. rkey	M3 HW counters	M4 mem. win. type 2	M5 multiple PDs	M6 enc./auth. in IB	M7 resource const.	M8 in-network filt.
A1	◑	●	●	○	○	○	○	○	◑	◑	○	●	○	○	✢	✗	✗	✗	✗	✓	✗	✓
A2	◑	●	●	○	○	○	○	○	◑	◑	○	●	○	○	✢	✗	✗	✗	✗	✓	✗	✓
A3	●	◑	●	○	●	◑	◑	●	○	○	●	◑	●	◑	✗	✓	✗	✓	✓	✗	✗	✗
A4	●	○	○	○	○	○	○	○	●	○	○	○	○	○	✗	✗	✗	✗	✗	✗	✢	✗
A5	●	◑	●	○	○	○	○	○	○	○	●	○	○	○	✗	✗	✢	✗	✗	✗	✢	✗
A6	○	○	○	●	○	○	○	○	○	○	●	○	○	◑	✗	✗	✢	✗	✗	✗	✗	✗

● enables attack ◑ facilitates attack ○ does not affect attack
✓ mitigates attack ✢ increases attack complexity ✗ does not mitigate attack

Adversary Model

(T1) An attacker with a normal end host

- can connect to RDMA services
- issue messages over these connections

(T2) An attacker with a compromised end host

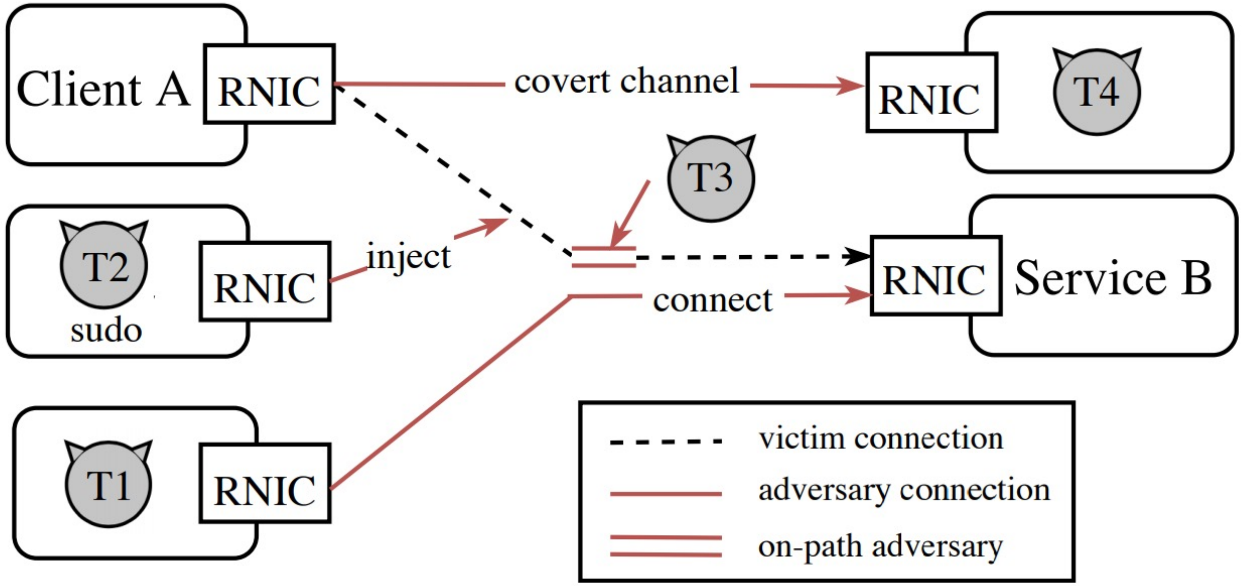
- fabricate and inject messages

(T3) An in-network attacker (e.g., malicious switch)

- on-path
- eavesdrop, modify

(T4) A malware-based attack

- use RDMA for data exfiltration (e.g., as covert channel)



RDMA Write Packet Format and Packet Processing

Routing Header (RH)	Base Transport Header (BTH)	RDMA Header	Payload	Checksums
InfiniBand: IB Routing Header RoCEv1: Ethernet + IB RH RoCEv2: Ethernet + UDP/IP	Queue Pair Number (QPN) Packet Sequence Number (PSN) Type of RDMA request	Target virtual address Memory key (*rkey*) Data Length		Two integrity CRC checksums

RDMA Write Packet Format and Packet Processing

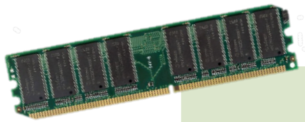
Routing Header (RH)	Base Transport Header (BTH)	RDMA Header	Payload	Checksums

InfiniBand: IB Routing Header
RoCEv1: Ethernet + IB RH
RoCEv2: Ethernet + UDP/IP

Queue Pair Number (QPN)
Packet Sequence Number (PSN)
Type of RDMA request

Target virtual address
Memory key (*rkey*)
Data Length

Two integrity CRC checksums



RDMA NIC

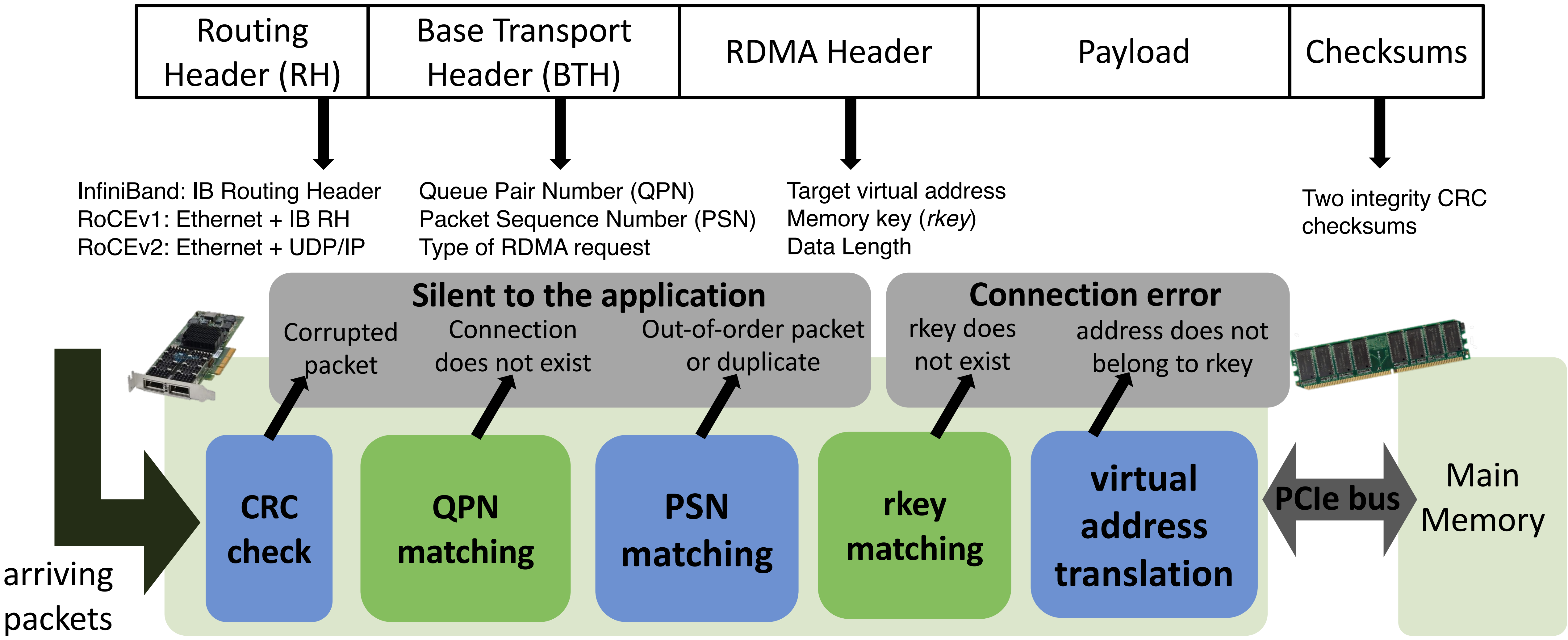
arriving packets

CRC check → QPN matching → PSN matching → rkey matching → virtual address translation

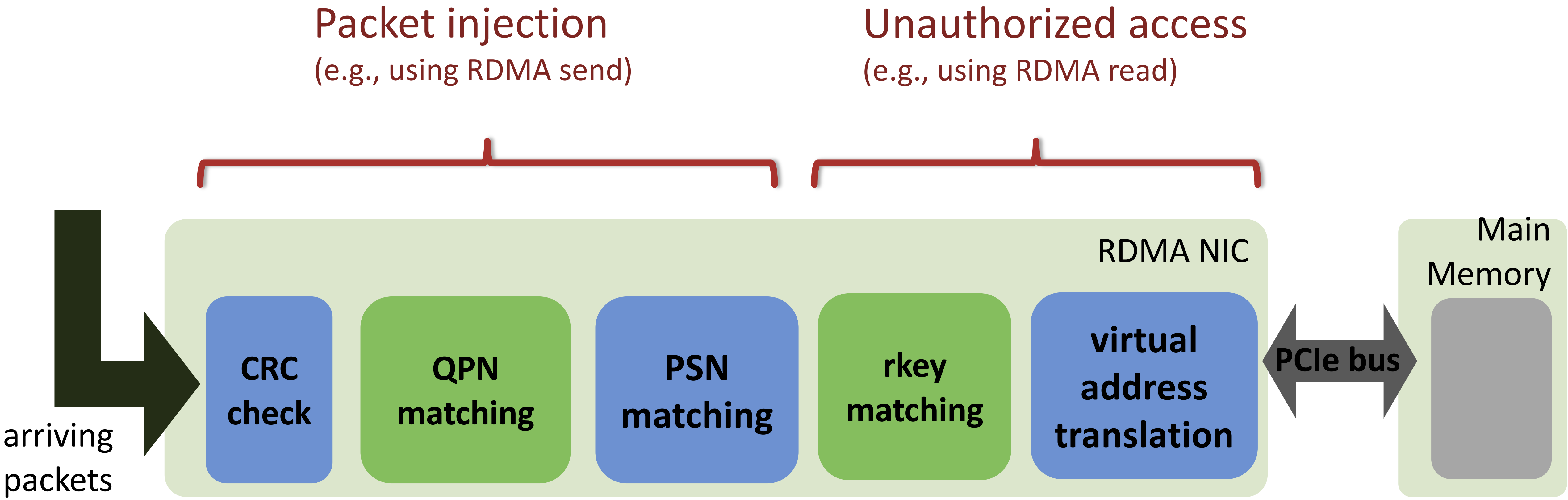
PCIe bus

Main Memory

RDMA Write Packet Format and Packet Processing



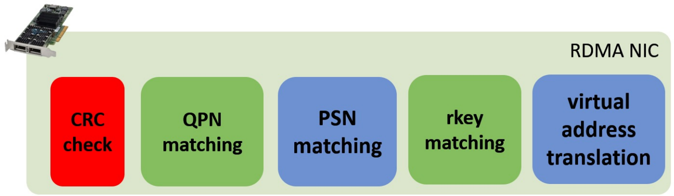
Bypassing RDMA Processing Checks



Towards Packet Injection -- CRC Check

- **Observations**
  - Neither encryption nor authentication are used in today's RDMA protocols

  - CRC checksums are used for packet integrity checks
    but have known seeds and polynomials
    and can easily be computed by an adversary

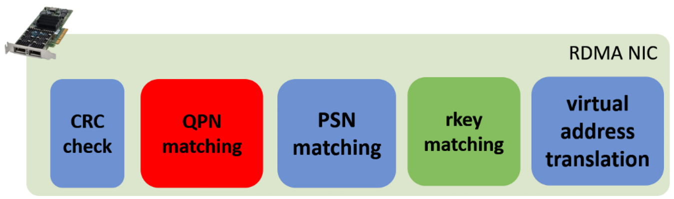


Towards Packet Injection -- QPN Matching

- **Observations**
  - Queue pair numbers are 24 bits (< 17M possible QPNs!)
  - In practice: they are allocated sequentially!
    → predicting preceding or subsequent QPNs is trivial

- **Device analysis**

Model	Driver	Arch.	QPNs
Broadcom NetXtreme-E BCM57414	bnxt_re	RoCEv2	sequential
Broadcom Stingray PS225 BCM58802	bnxt_re	RoCEv2	sequential
Mellanox ConnectX-3 MT27500	mlx_4	IB/RoCEv1	sequential
Mellanox ConnectX-4 MT27700	mlx_5	IB/RoCEv2	sequential
Mellanox ConnectX-5 MT27800	mlx_5	IB/RoCEv2	sequential
softRoCE	rxe	RoCEv2	sequential

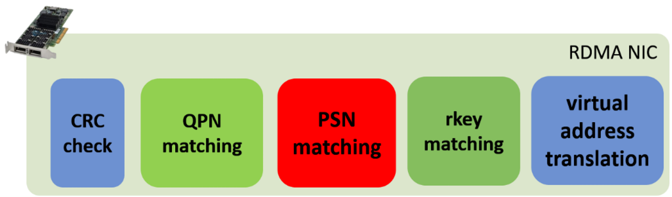


Towards Packet Injection – PSN matching

- **Observations**
  - Packet Sequence Number (PSN) is also 24 bits
  - PSN can be selected by the entity that creates an RDMA connection

- **Connection establishment via IB verbs versus RDMA connection manager**
  - RDMA connection manager assigns a random PSN to the connection
  - Establishing a connection using InfiniBand verbs leaves the option to the developer
  - Most analysed open-source RDMA systems tend to use IB verbs with a static PSN (simplicity?)

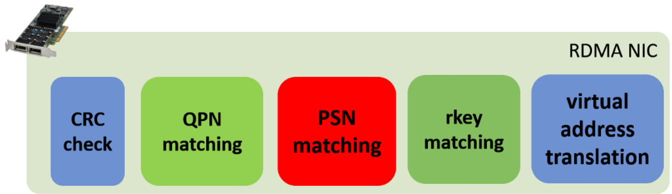
System	Connection
Infiniswap [11]	Manager
Octopus [21]	Native
HERD [12]	Native
RamCloud [25]	Native
Dare [28]	Native
Crail [30, 31]	Manager



Towards Packet Injection

- **Approach**
  - Bypassing the first three checks allows us to inject RDMA send packets (no RDMA header)
  - Our PoC injection tool can inject up to 1.6 Mpps (Mellanox ConnectX-5)
    → takes roughly 11s to enumerate the full 24 bit PSN

- **Observations**
  - Injecting RDMA packets with invalid PSN does not break the connection
  - Duplicate packets are dropped (and acknowledged)
    → "silent" packet replacement is possible!
  - Injecting 2^24 packets makes PSN counter wrap and can hide the attack from the application



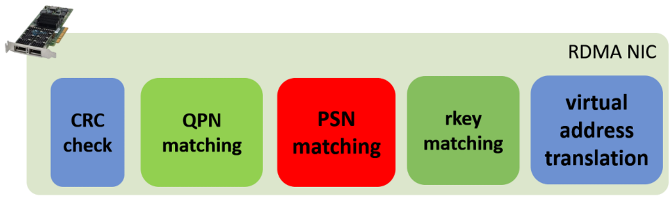
Misuse Packet Injection for Denial-of-Service

- **Approach**
  - Packets that passed the first three checks but contain protocol errors can force the QP into an error state → breaks the QP connection!
  - Our tool can inject up 1.6 Mpps

    known PSN: we can scan 1.6 M connections per second (and disconnect!)

    unknown PSN: enumerate a full PSN in ~ 11s (QPN is sequential)

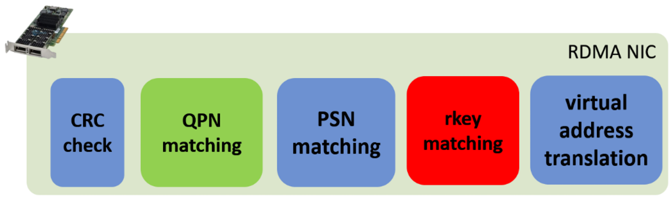
- **Observations**
  - QPN randomness is crucial to increase the attack complexity for packet injection
  - Example: victim with 1,000 RDMA connections with a random QPN, our tool is expected to break one of the connections in 48h



Towards Unauthorized Access – Guessing rkeys

- **Observations**
  - rkeys are used as 32 bit access control tokens
  - but: the rkey generation is highly predictable (less than 3 bits of entropy!)

- **Other problems**
  - Static initialization for key generation: the NIC generates the same keys after a reboot
  - Same constant protection domain for all QPs:
    allows to access memory even without impersonation using any QP connection
  - Shared key generator state: applications use the same network interface even if they use different protection domains



Towards Unauthorized Access – Guessing Addresses

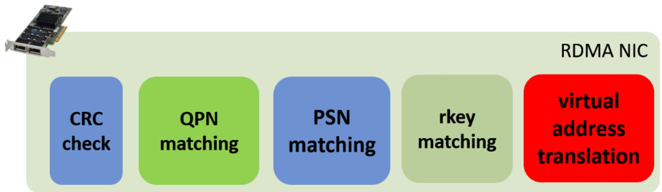
- **Virtual addresses are 64 bits**
  - Linux typically only uses 48 bits
  - Developers tend to use page-aligned memory for performance → 36 bits!

- **Consecutive allocation of memory regions**
  - Subsequent objects in memory are allocated in consecutive addresses with respect to a random address base
  - Example: InfiniSwap*

    Infiniswap is a remote swapping device for Linux

    Uses posix_memalign in a loop to allocate register buffers of 1GB

    Allows an attacker to predict the position of a newly allocated buffer



* J. Gu, Y. Lee, Y. Zhang, M. Chowdhury, and Kang G. Shin. 2017. Efficient memory disaggregation with INFINISWAP. In *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation* (*NSDI'17*).

Implementing Unauthorized Memory Access

- **Approach**
  - An attacker (M) can connect to a RDMA service to get an address and a rkey of its communication buffer
  - Assuming the buffers are allocated sequentially, the attacker can guess addresses and rkeys of other buffers on the service
  - All 6 analysed open-source RDMA systems were vulnerable to this attack
  - Attack is even simpler for an in-network attacker (eavesdrop rkey and buffer addresses)

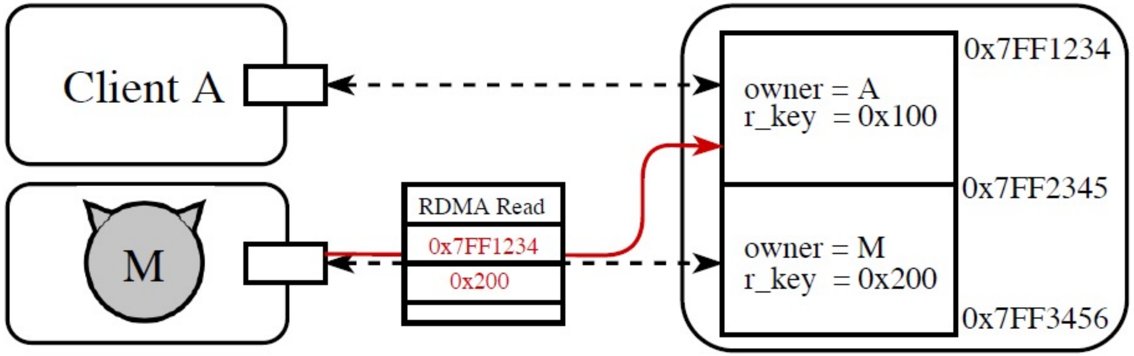
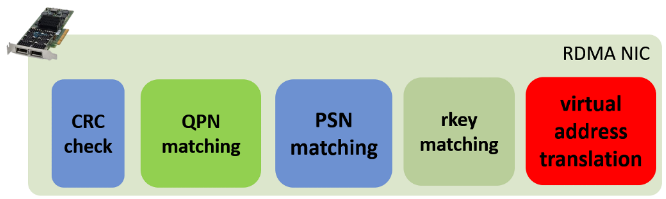


Figure 4: Unauthorized memory access on the same host.

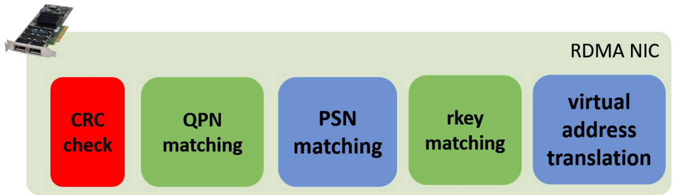


Mitigation Mechanisms

Prevent Packet Injection and Packet Alteration

- **Use a secure transport with authentication**
  - IPSec for RoCE (e.g., Mellanox Connect-X 6 DX)
  - sRDMA* for InfiniBand and RoCE

IPSec over RoCE	Routing Header (RH)	IPSec	Base Transport Header (BTH)	Payload	Checksums
sRDMA	Routing Header (RH)	Base Transport Header (BTH)	sRDMA header	Payload	Checksums

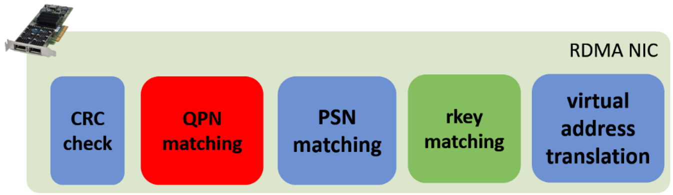


* K. Taranov, et al. sRDMA -- Efficient NIC-based Authentication and Encryption for Remote Direct Memory Access, Usenix ATC’20

QPN & PSN Randomization

- **We propose a software-based algorithm for QPN randomization in the paper**
  - Approach: RDMA allows creation of connection stubs that get a QPN assigned without actually connecting to a RDMA service

- **PSN randomization**
  - Use RDMA CM → randomly generates a PSN
  - But: RDMA CM exchanges connection parameters in plaintext
  - Solution: Native IB verbs interface with a random PSN

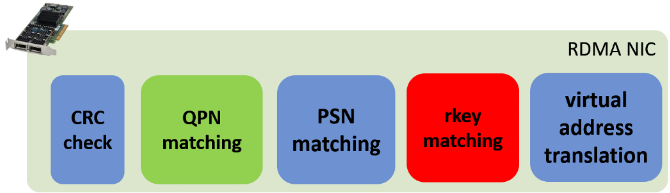


Rkey Randomization

- **We propose a software-based Rkey randomization algorithm for short-term mitigation**

- **Use multiple Protection Domains (PDs)**
  - share PDs between trusted connections
- **Use Memory Windows Type 2**
  - can be pinned to a specific QPN

- **sRDMA proposes crypto-based memory protection**



Additional Content in the Paper

- **Additional attacks**
  - QP exhaustion
  - Performance degradation
  - Using RDMA for data exfiltration

- **Mitigations**
  - Short-term and long-term mitigation mechanisms
  - Example: software-based algorithms for QPN and rkey randomization

Thank you for your attention!

- **ReDMArk provides an in-depth analysis of current RDMA security**
- **We discovered 10 vulnerabilities / design flaws**
- **We implemented 6 attacks under 4 different threat models**
- **We tested 6 open-source systems**
- **We propose 8 mitigation techniques**

ReDMArk implementation:

SPCL

ETH zürich
Network Security Group