Introduction to Virtio Crypto Device

arei.gonglei@huawei.com
xin.zeng@intel.com

Agenda

- Overview of virtio crypto device
- Virtio crypto device spec
- Introduction to software implementation
- WIP and future plans

Cryptography in cloud

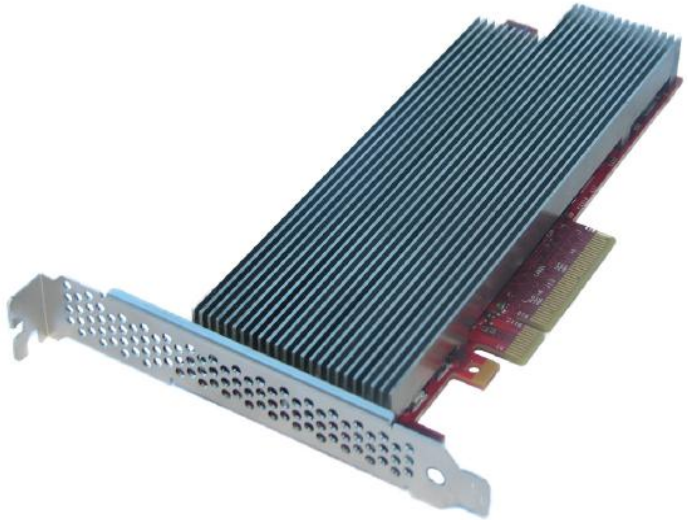
- Used widely
  - Wireless, telecom, data center, enterprise systems
- Compute-intensive tasks
- Hardware accelerators support virtualization are offered with high performance, but
  - Limited VF/PF number for VMs
  - Different VF drivers needed

Why Virtio-crypto?

- Friendly Cloud Characteristic
  - Hardware cryptography device agnostic
  - Live migration friendly
  - Unified device interface and frontend driver as well
- Good scalability
- Low cost in software

What’s virtio-crypto device

- A virtual cryptography device under virtio device framework
- Provides a set of unified operation interfaces for different cryptography services
- Contributions from Huawei, Intel, IBM, RedHat, SUSE, ARM, etc… in community



Spec overview (in RFC)

Device type	• Virtio Crypto Device
Device ID	• 0x1054
Device specific feature bits	• Multiplexing mode support for symmetric service • Indirect descriptors support
Device specific configuration	• Supported maximum queues • Detailed crypto algorithms mask bits • Misc fields such as maximum key length supported
Virtqueue design	• 1 control queue for session/control request • 1 or multi data queues for service request
Defined cryptography services	• Symmetric • Asymmetric

Virtqueue design

- One control queue
  - Session management for symmetric service
  - Facilitate control operations for device
- One or more data queues
  - Transport channel for crypto service requests

Request of control queue

- Consists of two parts
  - General header: virtio_crypto_ctrl_header
  - Service specific fields
    - Fixed size service-specific fields in session mode
    - Variable size in multiplexing mode

Request of data queue

- Consists of two parts
  - General header: virtio_crypto_op_header
  - Service specific fields
    - Fixed size service-specific fields in session mode
    - Variable size in multiplexing mode

Device specific configuration

```
struct virtio_crypto_config {
    le32 status;
    le32 max_dataqueues;
    le32 crypto_services;
    /* Detailed algorithms mask */
    le32 cipher_algo_l;
    le32 cipher_algo_h;
    le32 hash_algo;
    le32 mac_algo_l;
    le32 mac_algo_h;
    le32 aead_algo;
    /* Maximum length of cipher key in bytes */
    le32 max_cipher_key_len;
    /* Maximum length of authenticated key in bytes */
    le32 max_auth_key_len;
    le32 reserved;
    le64 max_size;
};
```

- **status** is used to show whether the device is ready to work or not
- **max_dataqueues** is the maximum number of data virtqueues exposed by the device.
- **crypto_services** crypto service offered
- **cipher_algo_l** CIPHER algorithms bits 0-31
- **cipher_algo_h** CIPHER algorithms bits 32-63

  … …
- **max_cipher_key_len** is the maximum length of cipher key supported by the device
- **max_auth_key_len** is the maximum length of authenticated key supported by the device
- **max_size** is the maximum size of each crypto request's content supported by the device

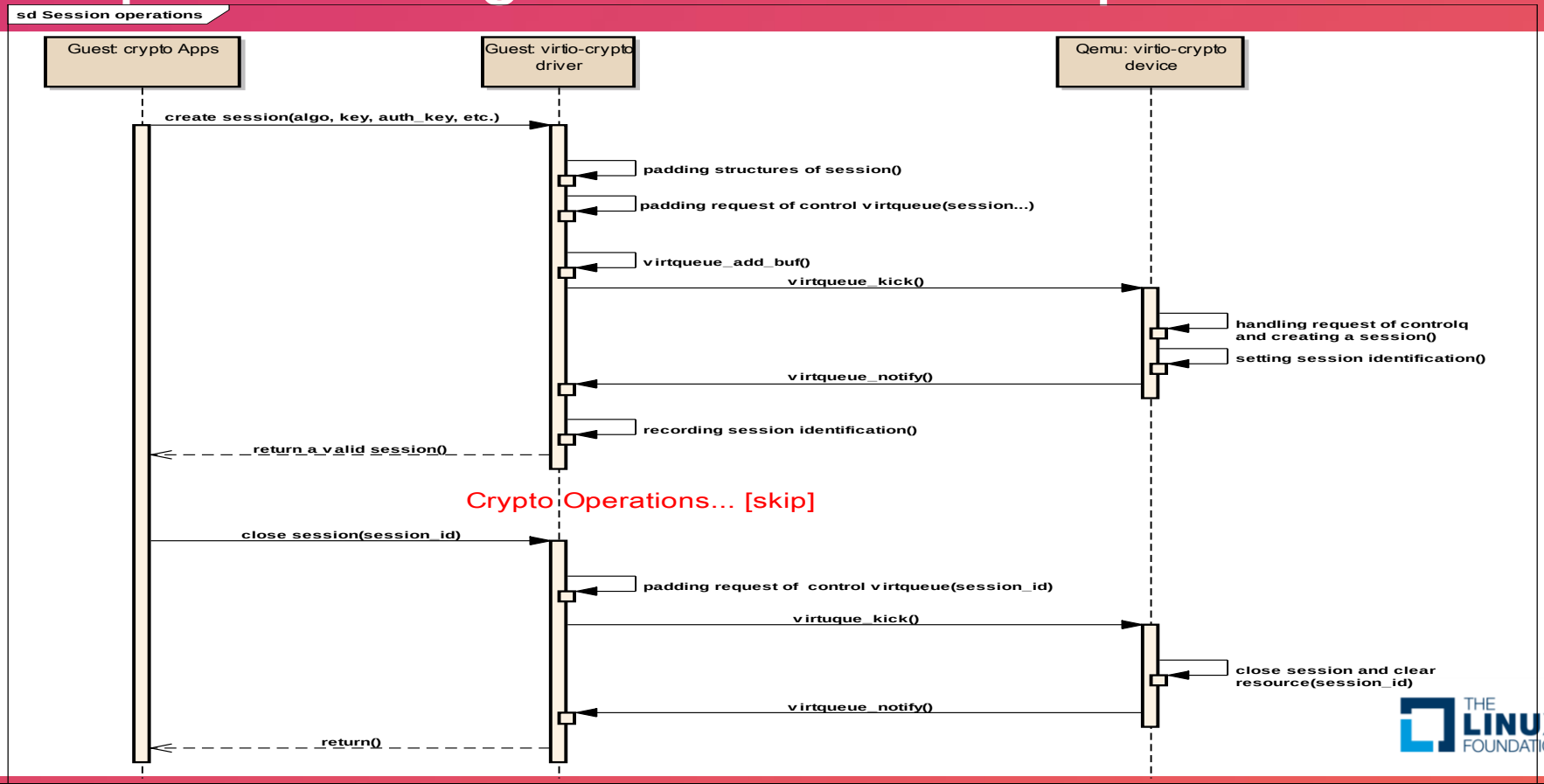
Symmetric crypto service

- Working modes
  - Session mode
    - Efficient for those numerous requests with same context
  - Multiplexing mode
    - To support stateless mode as well as session mode
    - Stateless mode is proposed to reduce cost of session creation for those one-shot requests
    - Controlled by feature bits
- Defined services & operations
  - Cipher
    - Encryption operation/Decryption operation
  - HASH
  - MAC
  - AEAD
    - Encryption operation/Decryption operation

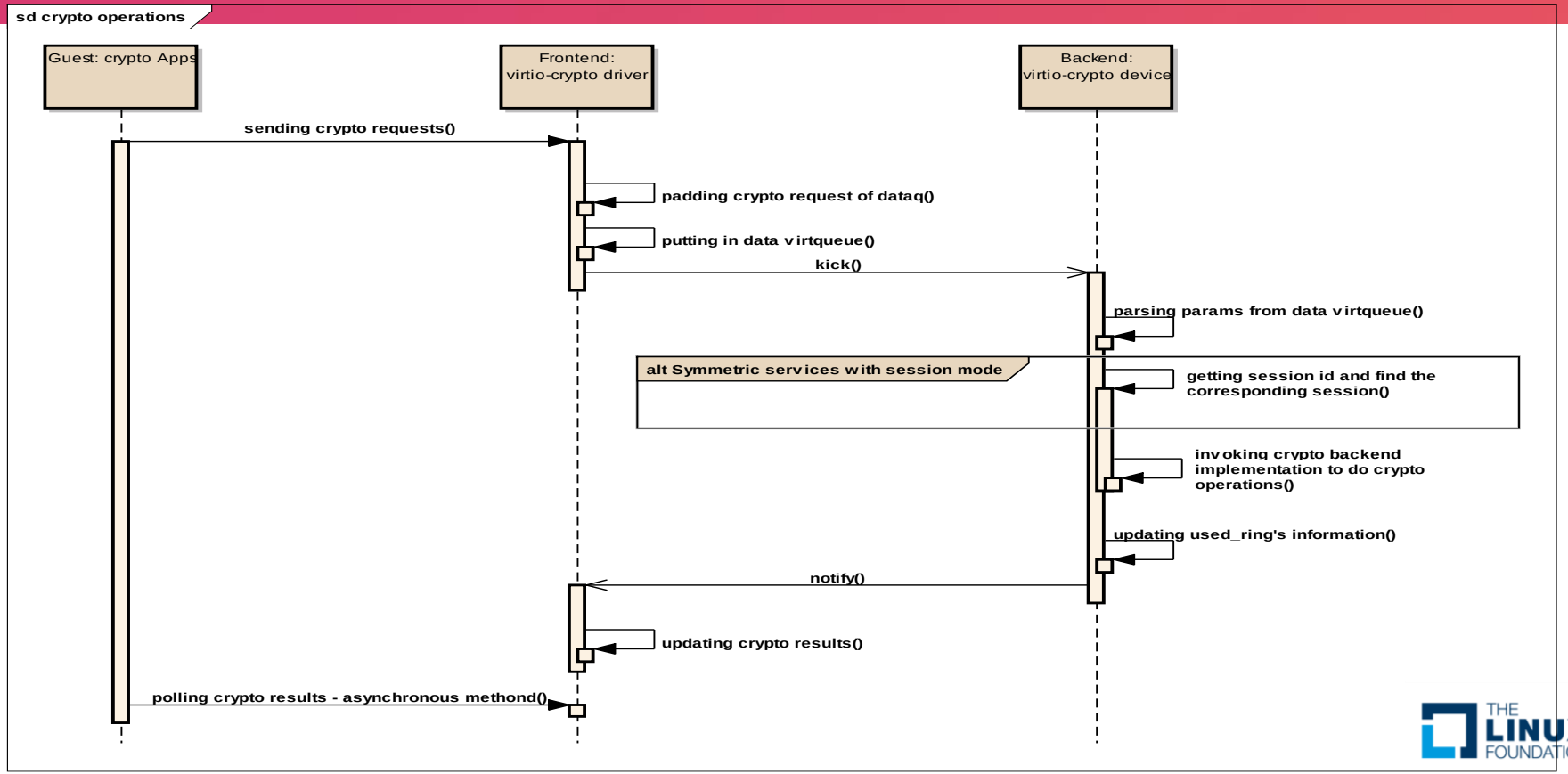
Asymmetric crypto service

- No session concept
- Requests are conveyed in data queue
- Defined service operations
  - Signature/Verification
    - RSA, DSA, ECDSA
  - Encryption/Decryption
    - RSA
  - Key Generation
    - RSA, DSA, EC
  - Key Exchange
    - DH, ECDH

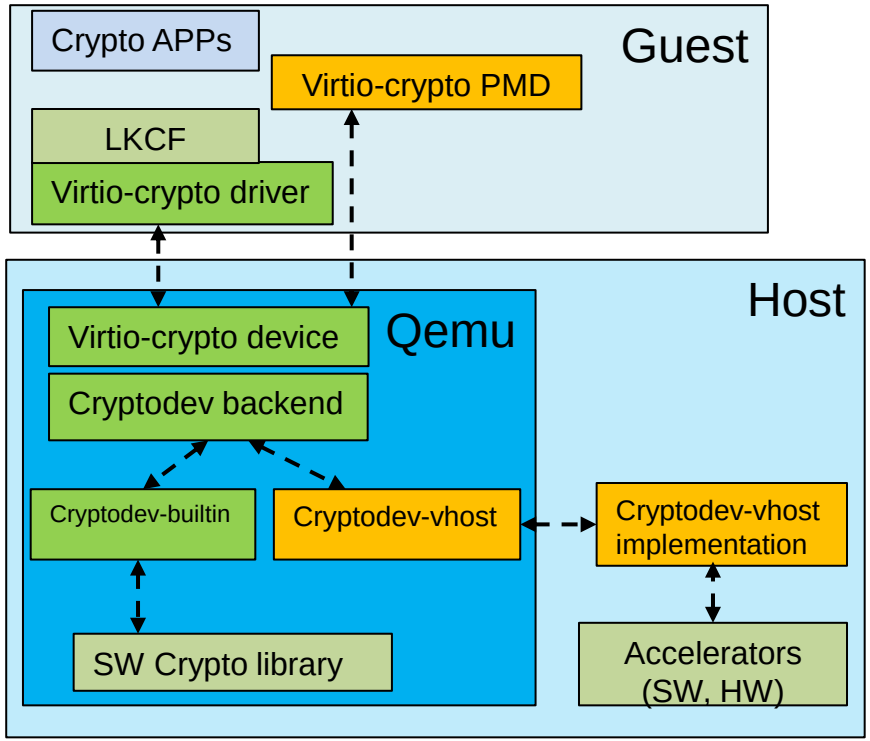
Sequence diagram – Session operations



Sequence diagram – Service operations

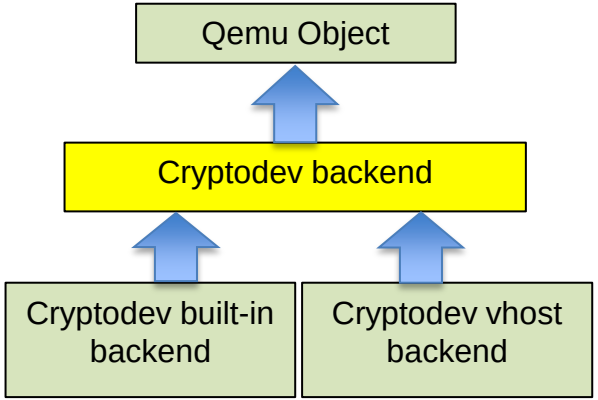


Software implementation diagram



- **In guest**
  - virtio-crypto user space pmd driver
  - LKCF based kernel space driver
- **In host**
  - virtio-crypto device inside QEMU
  - Cryptodev backend object inside QEMU which could be:
    - A cryptodev builtin backend
    - A cryptodev vhost backend
  - A vhost server implementation(vhost-user or vhost-kernel)

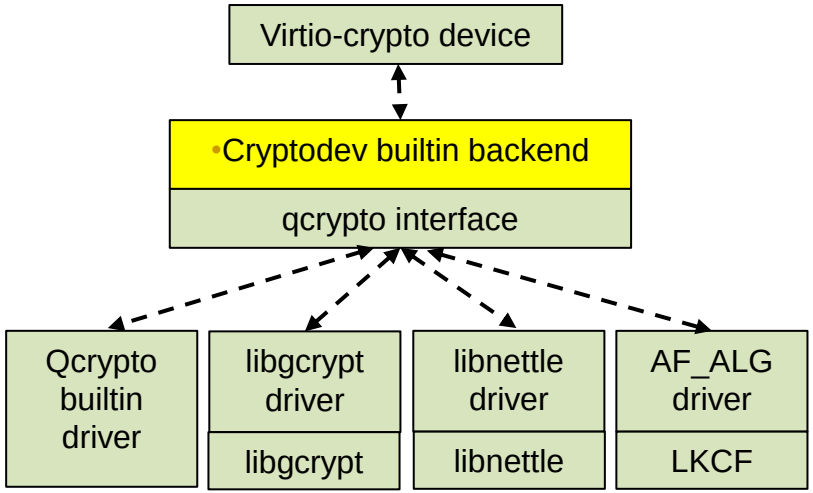
Virtio Cryptodev backend in host



- **An user creatable object in QEMU**
  - Commands: -object/object-add/object_add
  - Example: #./qemu-system-x86_64 -object cryptodev-backend,id=cy0
- **Easily to be realized with different child objects**
- **Key code:**

```
static const TypeInfo cryptodev_backend_info = {
    .name = TYPE_CRYPTODEV_BACKEND,
    .parent = TYPE_OBJECT,
    .instance_size = sizeof(CryptoDevBackend),
    .instance_init = cryptodev_backend_instance_init,
    .instance_finalize = cryptodev_backend_finalize,
    .class_size = sizeof(CryptoDevBackendClass),
    .class_init = cryptodev_backend_class_init,
    .interfaces = (InterfaceInfo[]) {
        { TYPE_USER_CREATABLE },
        { }
    }
```

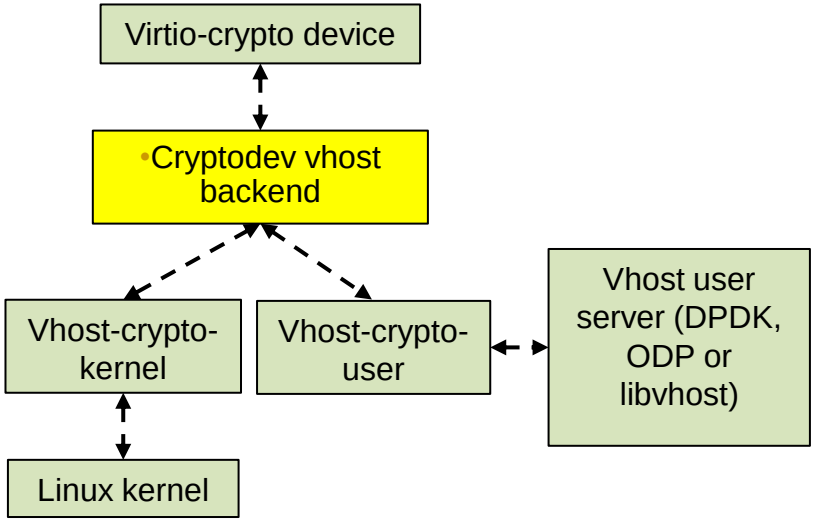
Cryptodev builtin backend



- A child of cryptodev backend
- Interfaced to QEMU crypto APIs
- Requests are consumed by underlying crypto modules
- Performance is not ideal for symmetric service
- Examples:

```
# qemu-system-x86_64 \
[...] \
        -object cryptodev-backend-builtin,id=cryptodev0 \
        -device virtio-crypto-pci,id=crypto0,cryptodev=cryptodev0 \
[...]
```

Cryptodev vhost backend



- A child of cryptodev backend
- Two kinds of implementations: vhost kernel client and vhost user client
- Vhost user server can be integrated with DPDK, ODP or libvhost
- Better performance, can be used in production environment
- Examples:

```
# qemu-system-x86_64 \
  [...] \
        -chardev socket,id=charcrypto0,path=/your/path/socket0
        -object cryptodev-vhost-user,id=cryptodev0 ,chardev=charcrypto0\
        -device virtio-crypto-pci,id=crypto0,cryptodev=cryptodev0 \
  [...]
```

WIP and Plans

Spec

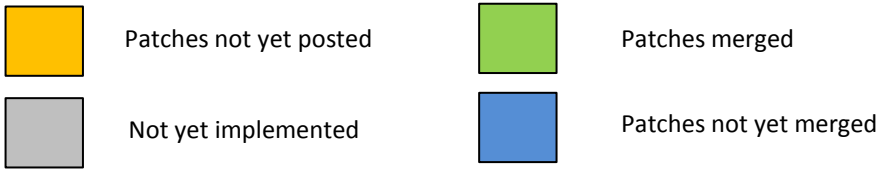
- Virtio-crypto specification for Symmetric and Asymmetric services
- More services such as KDF, PRIMITIVE.

Host

- QEMU -device virtio-crypto
- QEMU -object cryptodev-backend-builtin (symmetric)
- QEMU -object cryptodev-vhost-user
- QEMU -object cryptodev-backend-builtin (asymmetric)
- DPDK Vhost-user for virtio-crypto

Guest

- DPDK virtio-crypto-pmd
- LKCF based Virtio-crypto device driver (symmetric)
- LKCF based Virtio-crypto device driver (asymmetric)
- Support more algorithms, multi data queue, live migration etc.



Summary

- Virtio crypto device is a viable solution for cloud
- Virtio crypto device spec has been pushed to virtio community, defined services include:
  - Symmetric crypto service
  - Asymmetric crypto service
- The groundwork of implementation has been accepted
- The implementation for more service such as asym crypto service and algorithms are in progress.

Questions?

- For more information about virtio-crypto:
  - http://qemu-project.org/Features/VirtioCrypto
- For more information about DPDK:
  - http://dpdk.org/
- For more information about Intel® QAT:
  - www.intel.com/quickassist
- Welcome contributions!

KVM FORUM

THE LINUX FOUNDATION