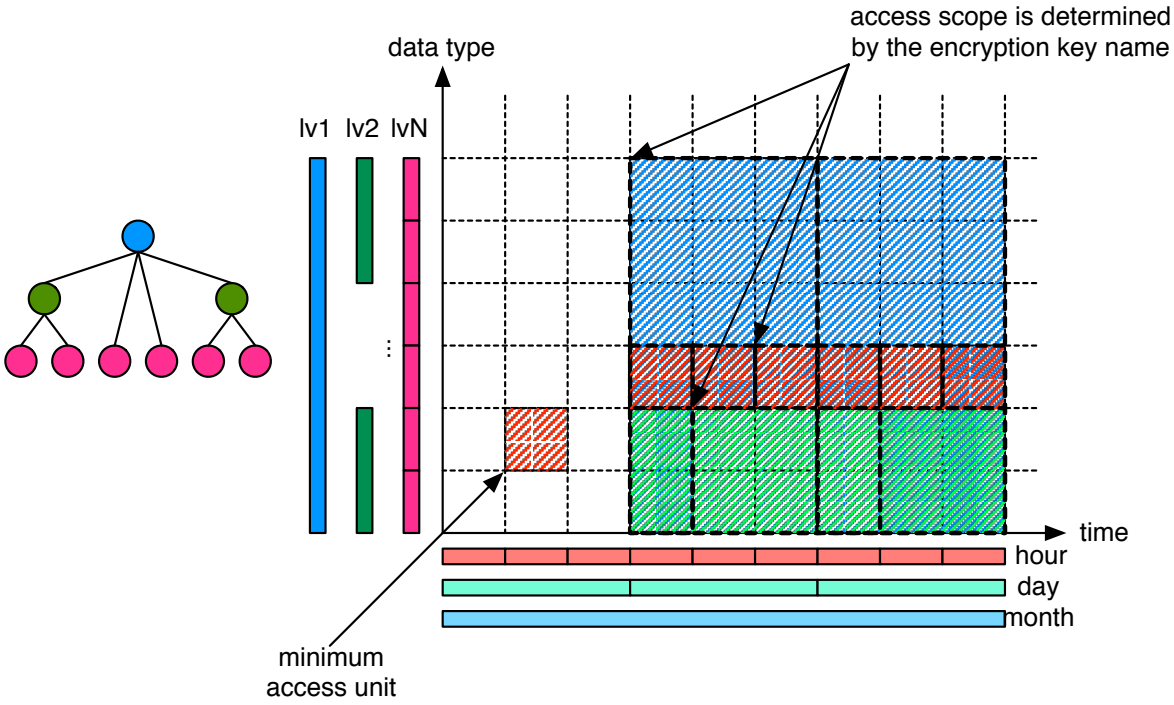
Group-based Encryption API

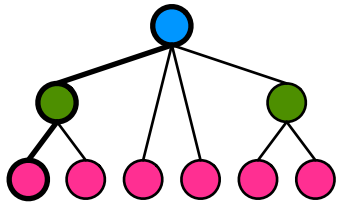
Name-based Access Control

- Encryption key (E-KEY) name defines the access scope
  - /<data_type>/E-KEY/[start-ts]/[end-ts]/
  - data type: determine a primary group
  - a primary group needs to create E-KEYs continuously
    - intervals (end-ts – start-ts) do not have to be the same, but have to be multiple minimum intervals



Data Producer

- Create content encryption key (symmetric key) for each minimum access unit
  - encrypt all data packets that fall into the access unit
  - /<data_type>/C-KEY/[timeslot]
- For a particular content encryption key, it may be covered by multiple E-KEYs
  - different levels of data type
  - predictable
    - have to belong to parent nodes back to root



- Data producer has a E-KEY manager
  - resolve **all** the E-KEYs that covers current data
    - data type is a parent name space
      - walk back alone the naming tree, check if E-KEY exists (using selectors)
    - [start-ts, end-ts) covers current timestamp
    - has valid signature
  - encrypt the content encryption key with all the covering E-KEYs
  - /<data_type>/C-KEY/[timeslot]/[group_data_type]

Data Producer API

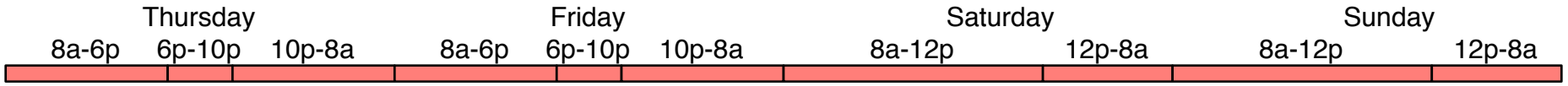
- Handling:
  - data encryption
  - data signing
  - key storage
- Not handling (leave to app)
  - data storage
  - data.key publishing
- Provides
  - create content encryption key
  - collect E-KEYs
  - prepare content encryption key encrypted with E-KEYs
  - prepare content encrypted with content encryption key
  - sign data
- Ideal usage:

```
DataProducer producer("/a/b/c");

auto encryptedContentKeys =
  producer.createContentKey("20150806T120000", onCreated);
// publish encryptedContentKeys

producer.produce(data, "20150806T120000",
                 content, contentLen, onProduced);
// publish data
```

```
class DataProducer
{
public:
  /// @brief Create a producer using @p namespace
  DataProducer(const Name& namespace);

  /**
   * @brief Create a content encryption key for the time slot
   *        @p timeslot
   *
   * This method creates a content encryption key, collects all
   * the covering E-KEYs, and encrypts the content key using the
   * collected E-KEYs.
   *
   * When processing is finished, @p callback will be invoked,
   * which accepts two arguments: first one is the content key
   * encrypted using the producer's public key. The second a set
   * of the content key encrypted using each E-KEY.
   */
  void
  createContentKey(Time timeslot,
                   const KeyCallback& callback);

  /**
   * @brief Fill @p data with @p content encrypted using the key
   *        for @p timeslot
   *
   * This method may segment content when necessary. When
   * processing is done, the @p callback will be invoked, which
   * accepts a set of produced data.
   */
  void
  produce(Data& data, Time timeslot,
          const uint8_t* content, size_t contentLen,
          const DataCallback& callback);
};
```

Primary Group Manager

- Name consistent with data namespace
  - in most cases, parent namespaces
- Create group E-KEYs
  - free to determine the interval of an E-KEY
  - /<data_type>/E-KEY/[start-ts]/[end-ts]



- Grant group members the access to D-KEYs
  - private part of E-KEY encrypted using member's public key
  - /<data_type>/D-KEY/[start-ts]/[end-ts]/[member_name]
- Grant partial D-KEYs for finer granularity control
  - give D-KEYs only for weekends
  - give D-KEYs only for off-hours in work day
- Add member: encrypt D-KEY using member's public key
- Remove member: stop encrypting D-KEY using the member's key

Group Manager API

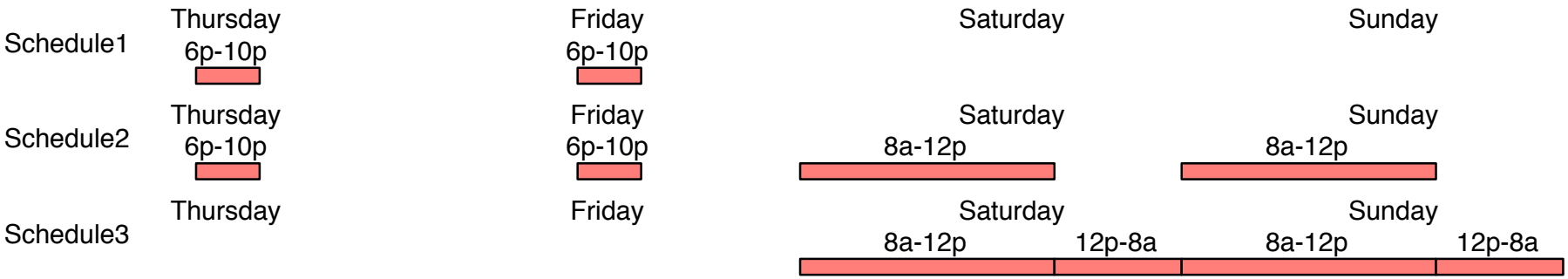
- Handling:
  - member management
  - grant partial access
  - key storage
- Not handling (leave to app)
  - key publishing
- Group manager calculates the granularity of group keys according to submitted schedule
- Ideal usage:

```
GroupManager manager("/a/b/c");

manager.addMember(userCert1, schedule1);
manager.addMember(userCert2, schedule2);

auto groupKey =
  manager.getGroupKey("20150806T120000");
// publish groupKey
```

```
class GroupManager
{
public:
  /// @brief Create group manager using @p namespace
  GroupManager(const Name& namespace);

  /**
   * @brief Create a group key for interval which
   *        @p timeslot falls into
   *
   * This method creates a group key if it does not
   * exist, and encrypts the key using public key of
   * all eligible members
   *
   * @returns The group key (the first one is the
   *          public key, and the rest are encrypted
   *          private key.
   */
  std::list<Data>
  getGroupKey(Time timeslot);

  /// @brief Add @p memCert with @p schedule
  void
  addMember(const Data& memCert,
            const Schedule& schedule);

  /// @brief Remove @p member from the group
  void
  removeMember(const Name& member);
};
```

Schedule

- For partial access
  - a set of accessible timeslots
- Define a schedule
  - starting time
  - ending time
  - repeating pattern
    - per week, per day, customizable
- Members with the same schedule forms a secondary group



Post-Fact Key Distribution

- Grant a new member the access to data certain time ago
- Create missing E-KEYs/D-KEYs if the member's schedule does not exist
  - encrypt the content key using the newly created E-KEYs
  - encrypt all D-KEYs belonging to the schedule using the member key
    - group manager already knows the content encryption key
- Publish encrypted D-KEYs