mGuard

Development Updates

Suravi Regmi, Lan Wang
University of Memphis

What is mGuard

mGuard is a secure, Real-Time mHealth Data Distribution application.

Built on NDN

mGuard uses NDN, shifting communication from host-centric to data-centric networking for enhanced security.

Access control via NAC-ABE

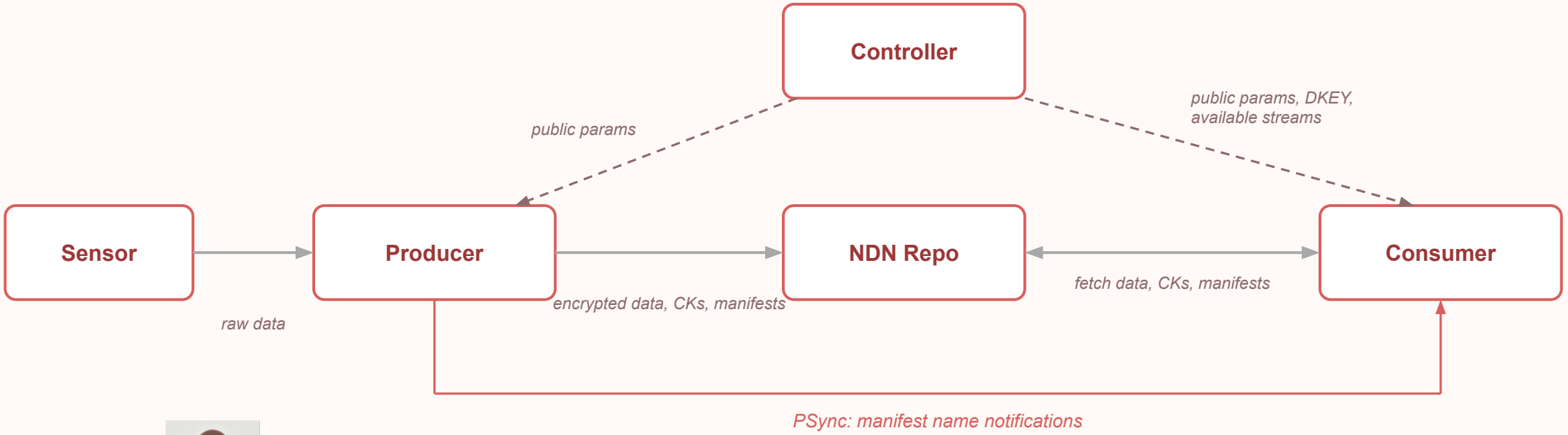
NAC-ABE enforces cryptographic access policies tied to data names and attributes for secure data decryption.

Driven by MD2K

MD2K cyberinfrastructure enables collection of multi-stream wearable sensor data at up to 128 Hz.

Together: real-time delivery of high-frequency sensor data with cryptographic, context-aware access control rooted in the data itself.

System overview



Meet Alice.

Her phone and wearables produce continuous health streams. She shares them with different parties under different rules.

Doctor

health data
at home only

Trainer

health data
at gym only

Researcher

all data
after April 7

Updates in mGuard

Library changes that enable higher data rates with proper trust enforcement, plus testbed experiments that validate the resulting pipeline.

- **NAC-ABE library**

  Content key redesign with sharing and caching, plus trust schema validation backed by a stream-scoped certificate hierarchy.

- **Naming scheme**

  Realigned with the new CK structure and trust schema; extended for KP-ABE attribute encoding and segmentation.

- **ndn-python-repo**

  New ingest protocol with explicit insertion acknowledgement, replacing the old multi-round-trip protocol that gave producers no feedback.

- **Data aggregation**

  Same-attribute sensor points combined into a single data packet to reduce packet count and insertion load.

Experiments run on Mini-NDN and the NDN testbed.

NAC-ABE library updates

Key-policy ABE

Original NAC-ABE supported only CP-ABE, which would have required re-encrypting published data whenever a new user was authorized. We added KP-ABE so policies are encoded in user decryption keys, and new users can be added without re-encrypting existing data.

Content key sharing and caching

One CK now covers all data objects with identical attributes within the policy's time granularity, instead of one CK per object. Producer and consumer both cache CKs to avoid repeated encryption and decryption.

Trust hierarchy and schema validation

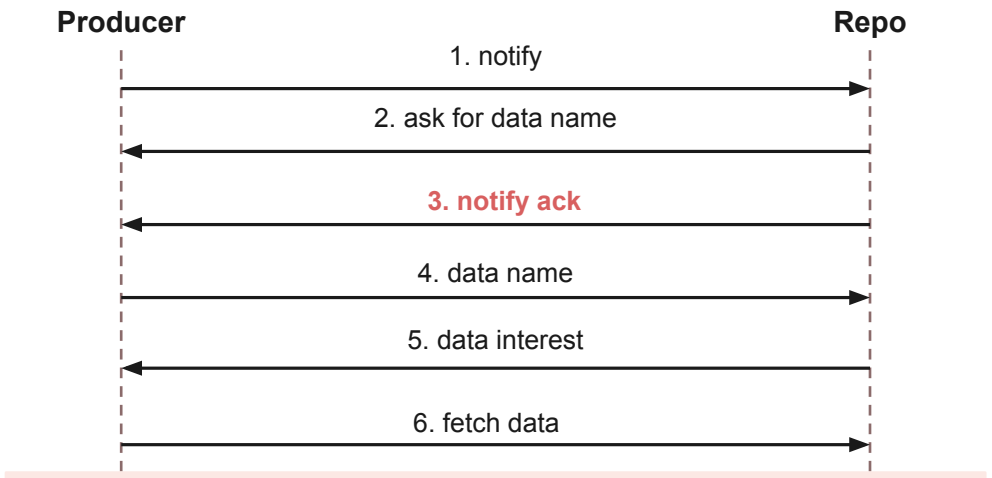
Each data stream has its own certificate, signed by the Producer certificate. A validator interface checks every fetched packet against a trust schema before delivery. Compromise stays contained to a single stream.

Naming scheme realigned with the NAC-ABE redesign and extended for KP-ABE, key versioning, and segmentation.

ndn-python-repo: insertion acknowledgement

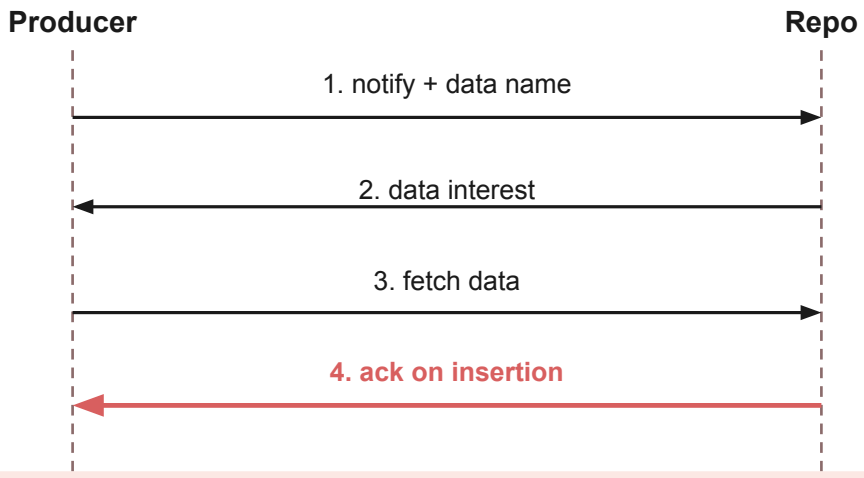
The old protocol gave no insertion confirmation, so under load CKs and manifests dropped silently. A single missing CK or manifest could render a large block of data undecryptable on the consumer side.

Old protocol



no acknowledgement back to producer

New ingest protocol



producer retries on failure

Impact Insertion failures drop to zero. The shorter exchange removes the old name-request round trip, so insertion is faster overall.

Data aggregation

Sensor points are small (80–120 B), but one NDN packet per point creates high overhead from encryption, manifests, and repo insertions.

Manifest: signed NDN packet containing data names and SHA-256 digests

- Only manifests are RSA-signed
- PSync tracks sequential manifest names
- One notification per batch instead of per object

Remaining bottleneck: repo insertion still occurs per individual packet.

What we do Aggregate multiple sensor points into a single NDN data packet.

Constraints

- All aggregated points must share the same data attributes
- All aggregated points must fall within the same CK granularity window
- Each aggregated packet capped at 8 KB to stay within NDN packet size limits

Result Fewer packets, fewer signatures, fewer repo insertions. Encryption and decryption stay sub-millisecond per packet.

Experiment setup

Goal verify the full pipeline runs end-to-end at 1 to 100 Hz, which was not achievable before these updates.

Environments

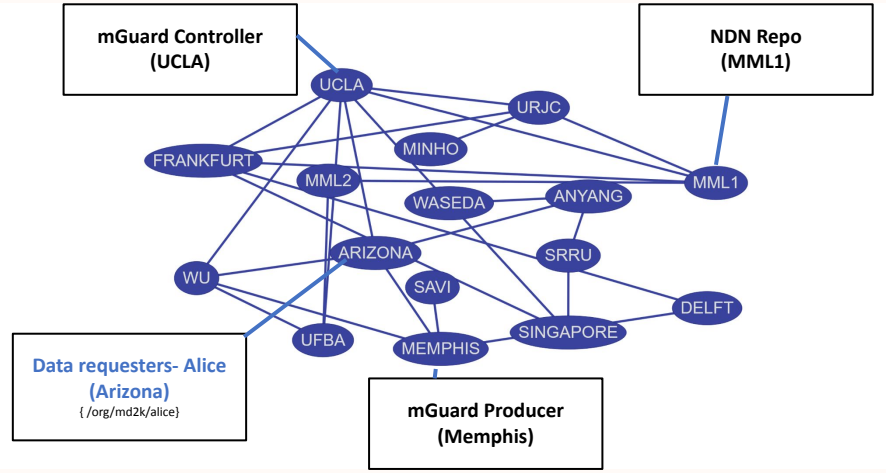
- Mini-NDN for local validation & Global NDN testbed for distributed runs across geographically separated nodes

Topology

- Producer, Controller, Repo, Consumer(s) on independent nodes with separate NFD instances
- Four-node topology for baseline, frequency, CK, and long-duration runs
- Multi-consumer topology with five consumers for scalability and policy runs

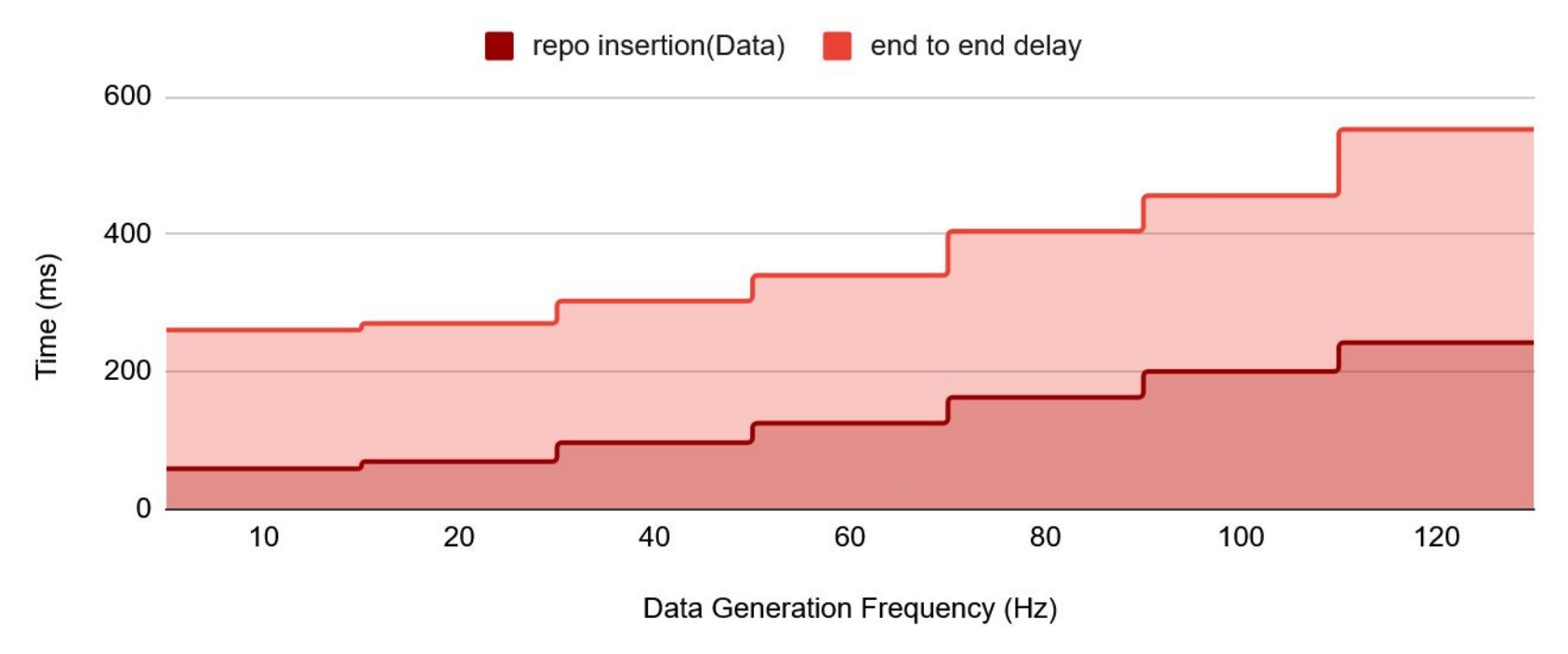
Workload

- Cerebral Cortex random data generator
- with a 1 second - 2 minute batch window, 1 to 120 Hz
- Varied: CK granularity, packet size, number of consumers, duration up to 8 hours



The repo insertion problem

5-minute runs, 1-second batch, 10 to 120 Hz. One packet per data row (no aggregation, old repo protocol). RTT ≈ 20 ms.



As data frequency increases from 10 Hz to 120 Hz:

- Packet count: 3,000 to 36,000
- Repo insertion time 59 ms to 242 ms
- End-to-end delay grows from 260 ms to 552 ms

Mini-NDN run;

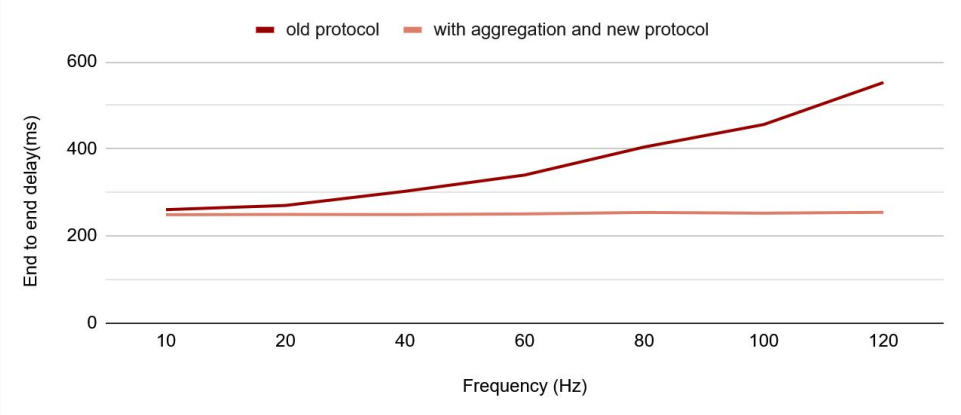
Aggregation + new repo protocol

5-minute runs, 1-second batch, 10 to 120 Hz. RTT ≈ 20 ms.

Freq Vs Packet Count

Freq	Packets (Before Aggregation)	Packets (After Aggregation)
10 Hz	3,000	**301**
60 Hz	18,000	**301**
120 Hz	**36,000**	**601**

End-to-end delay vs frequency, with and without aggregation



Takeaway Both end-to-end delay and repo insertion go from climbing-with-frequency to flat. Aggregation isolates and removes the bottleneck.

each packet can have up to 60 data points in this experiment data stream(gyroscope). Since each batch is 1 second we have 2 data packets per 1 second batch for frequency 60hz onwards till 120

CK overhead: sharing and caching

1 hz, 1-hour run, three CK granularities. Old: per-object CK generation. New: CK sharing across same-attribute objects within the granularity window.

Granularity	Old: CK objects per hour	New: CK objects per hour	Reduction
1 sec	3,600	3,600	—
1 min	3,600	60	60×
1 hour	3,600	1	3,600×

Per-CK cost

16 ms

CK encryption

24 ms

CK decryption (data path)

Sharing

One CK now covers all data objects with identical attributes within the granularity window, instead of one CK per object.

Caching

Producers cache generated CKs, consumers cache retrieved CKs. Each cache hit avoids one encryption (16 ms) or decryption (24 ms).

Takeaway Sharing cuts how many CKs exist; caching cuts how often the expensive operations run.

Open items and next steps

In progress

- Full testbed validation of the new repo protocol
- Mini-NDN runs already show clear improvement over earlier baselines
- Long-duration stability runs with CK sharing and data aggregation enabled

Known limitations

- OpenABE is no longer actively maintained, a long-term sustainability risk for the KP-ABE backend
- DKEYs grow large for users authorized across many streams
- Planned: move from per-stream attributes to prefix-based attribute definitions

Next

- Continued long-running and multi-consumer evaluation on the testbed
- Automated security bootstrapping and key management with revocation
- Direct sensor-to-repository data transfer as mGuard moves toward broader deployment

References

- Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., claffy, k., Crowley, P., Papadopoulos, C., Wang, L., Zhang, B. Named Data Networking. ACM SIGCOMM CCR, 2014.
- Afanasyev, A., Refaei, T., Wang, L., Zhang, L. A Brief Introduction to Named Data Networking. IEEE MILCOM, 2018.
- Zhang, Z., Yu, Y., Zhang, H., Newberry, E., Mastorakis, S., Li, Y., Afanasyev, A., Zhang, L. An overview of security support in Named Data Networking. IEEE Communications Magazine, 2018.
- Zhang, Z., Yu, Y., Ramani, S.K., Afanasyev, A., Zhang, L. NAC: Automating Access Control via Named Data. IEEE MILCOM, 2018.
- Goyal, V., Pandey, O., Sahai, A., Waters, B. Attribute-based encryption for fine-grained access control of encrypted data. ACM CCS, 2006.
- Bethencourt, J., Sahai, A., Waters, B. Ciphertext-policy attribute-based encryption. IEEE S&P, 2007.
- NAC-ABE Library. https://github.com/UCLA-IRL/NAC-ABE
- Zhang, M., Lehman, V., Wang, L. Scalable Name-based Data Synchronization for Named Data Networking. IEEE INFOCOM, 2017.
- Moll, P., Patil, V., Wang, L., Zhang, L. SoK: The Evolution of Distributed Dataset Synchronization Solutions in NDN. ACM ICN, 2022.
- Yu, Y., Afanasyev, A., Clark, D., Jacobson, V., Zhang, L. Schematizing Trust in Named Data Networking. ACM ICN, 2015.
- Yu, T., Kong, Z., Ma, X., Wang, L., Zhang, L. PythonRepo: Persistent In-Network Storage for Named Data Networking. IEEE ICNC, 2024.
- Zhang, Z., Yu, Y., Afanasyev, A., Zhang, L. NDN Certificate Management Protocol (NDNCERT). NDN Technical Report NDN-0054, 2017.
- Mini-NDN: A Mininet-based NDN emulator. https://github.com/named-data/mini-ndn
- MD2K: Center of Excellence for Mobile Sensor Data-to-Knowledge. http://md2k.org
- Dulal, S., Ali, N., Thieme, A.R., Yu, T., Liu, S., Regmi, S., Zhang, L., Wang, L. Building a Secure mHealth Data Sharing Infrastructure over NDN. ACM ICN, 2022.

Thank you.

Q&A