UNIVERSITY OF CALIFORNIA

Los Angeles

Private and Resilient Mobile Edge Networks

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

Joshua G Joy

2018

ABSTRACT OF THE DISSERTATION

Private and Resilient Mobile Edge Networks

by

Joshua G Joy
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2018
Professor Mario Gerla, Chair

The oncoming data explosion of the Internet of Things, and in particular the Internet of Vehicles, has led to placing computation near the data, as opposed to backhauling large amounts of data to the Internet cloud. This type of edge networking provides efficiency gains; however, mobile edge networking must also be resilient in disruptive networks as well as provide security and privacy protection.

In this thesis, we address both challenges, namely support efficient and robust communications at the network edge, even in disruptive connectivity conditions; and in the same conditions, maintain decentralized, private, confidential exchanges. First, we introduce the concept of Information Centric Networks (ICNs), an increasingly popular strategy for wireless edge scenarios and describe how caching and Network Coding can improve performance in disruptive wireless connectivity environments. Secondly, we introduce a privacy mechanism for mobile data that improves the privacy strength while preserving utility. That is, we perform query expansion to reduce the information leakage due to an individual's participation.

The dissertation of Joshua G Joy is approved.

Gregory J. Pottie

Mani B. Srivastava

Songwu Lu

Mario Gerla, Committee Chair

University of California, Los Angeles

2018

To my wife and parents...

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Foremost, I would like to express my sincere gratitude to my advisor, Mario Gerla for his continuous support and encouragement throughout my PhD.

Thanks to the UCLA Computer Science Department for providing me with TA opportunities. In addition, to the DARPA CBMEN program which helped with my first year of funding.

I am highly thankful to Mark-Oliver Stehr and the entire research group at SRI International Ashish Gehani and Minyoung Kim for hosting me for a summer and making it an amazing experience. Thanks to my fellow interns Sam Wood and Hasnain Lakhani. Also thanks to Victor Perez and Dennis Lu for their help with running experiments.

I would also like to thank Ken Oguchi and Onur Altintas for hosting me for a summer at Toyota Infotechnology Center, USA in Mountain View. Also thanks to Falko Dressler and Takamasa Higuchi.

Thanks to UCLA Dining, Martin Verde, Keshav Tadimeti, and Tyler Lindberg for their contributions to making CrowdZen a reality at UCLA. Thanks to Minh Le for his help with implementing LocationSafe in the GPSD daemon.

Thanks to everyone in the UCLA Network Research Lab during my time there for providing me with support, both in research and in morale - Uichin Lee, Daeki Cho, Youngtae Noh, Fabio Angius, Yu-Ting Yu, Ruolin Fan, Pengyuan Du, Vince Rabsatt, Tuan Le, You Lu, Seunghyun Yoo, Noor Abani, Jorge Mena, Qi Zhao, Giovanni Pau and the visitors Ciarán Mc Goldrick of Trinity College Dublin, and Torsten Braun of the University of Bern.

VITA

2012 M.S. (Computer Science), UCLA. Los Angeles, CA.

2012 Researcher, SRI International. Menlo Park, CA.

2017 Researcher, Toyota Infotechnology Center, USA. Mountain View, CA.

PUBLICATIONS

Joy, Josh. "Scalable Privacy—Attacks on Local Privacy and Countermeasures." Vancouver, BC: USENIX Association, 2017.

Joy, Joshua, Vince Rabsatt, and Mario Gerla. " Internet of Vehicles: Enabling Safe, Secure, and Private Vehicular Crowdsourcing ." *Internet Technology Letters, December 2017*

Joy, Joshua, Ciaran McGoldrick, and Mario Gerla. "Mobile Privacy-Preserving Crowdsourced Data Collection in the Smart City." *Proceedings of the Scientific Challenges in Data and Event-driven Smart City Service and Applications, SDESS@DEBS 2016, Irvine, CA, USA, June 24, 2016.*

Joshua Joy and Mario Gerla. "Internet of Vehicles and Autonomous Connected Car - Privacy and Security Issues." In *26th International Conference on Computer Communication and Networks, ICCCN 2017, Vancouver, BC, Canada, July 31 - Aug. 3, 2017*, pp. 1–9. IEEE, 2017.

Joy, Joshua, Eric Chung, Zengwen Yuan, Jiayao Li, Leqi Zou, Mario Gerla, " Discover-Friends: secure social network communication in mobile ad hoc networks ." *Wireless Communications and Mobile Computing, May 2016*

Joy, Joshua and Mario Gerla. "Internet of Vehicles and Autonomous Connected Car - Privacy and Security Issues." *26th International Conference on Computer Communication and Networks, ICCCN 2017, Vancouver, BC, Canada, July 31 - Aug. 3, 2017*. IEEE, 2017, 1–9.

Joy, Joshua, Minh Le, and Mario Gerla. "LocationSafe: granular location privacy for IoT devices." *Proceedings of the Eighth Wireless of the Students, by the Students, and for the Students Workshop, S3@MobiCom 2016, New York City, NY, USA, October 7, 2016*. Ed. Shubham Jain and Yasaman Ghasem Pour ACM, 2016, 39–41.

Joy, Joshua, Yu-Ting Yu, Mario Gerla, Ashish Gehani, Hasnain Lakhani, and Minyoung Kim. "Energy efficient, context-aware cache coding for mobile information-centric networks." *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems, DEBS '16, Irvine, CA, USA, June 20 - 24, 2016*. Ed. Avigdor Gal, Matthias Weidlich, Vana Kalogeraki, and Nalini Venkasubramanian ACM, 2016, 270–280.

Joy, Joshua, Yu-Ting Yu, Mario Gerla, Samuel B. Wood, James Mathewson, and Mark-Oliver Stehr. "Network coding for content-based intermittently connected emergency networks." *The 19th Annual International Conference on Mobile Computing and Networking, MobiCom'13, Miami, FL, USA, September 30 - October 04, 2013*. Ed. Sumi Helal, Ranveer Chandra, and Robin Kravets ACM, 2013, 123–126.

Joy, Joshua, Yu-Ting Yu, Victor Perez, Dennis Lu, and Mario Gerla. "A new approach to coding in content-based MANETs." *International Conference on Computing, Networking and Communications, ICNC 2014, Honolulu, HI, Feb 3-6, 2014*. IEEE, 2014, 173–177.

CHAPTER 1

Introduction

Self-driving vehicles are poised to form the most relevant realization of mobile ad hoc networks. Vehicle ad hoc networks (VANETs) will collaborate, communicate, and compute at the network edge without a centralized or fixed infrastructure coordination.

Traditionally, the vehicle has been the extension of the manual ambulatory system, docile to the drivers' commands. Recent advances in communications, controls and embedded systems have changed this model, paving the way to the Intelligent Vehicle Grid. The car is now a formidable sensor platform, absorbing information from the environment, from other cars (and from the driver) and feeding it to other cars and infrastructure to assist in safe navigation, pollution control and traffic management. The next step in this evolution is just around the corner: the Internet of Autonomous Vehicles.

Like other important instantiations of the Internet of Things (e.g., the smart building, etc), the Internet of Vehicles will not only upload data to the Internet with V2I. It will also use V2V communications, storage, intelligence, and learning capabilities to anticipate the customers' intentions and learn from other peers. V2I and V2V are essential to the autonomous vehicle, but carry the risk of the loss and disruption of packets, in addition to attacks.

The urban fleet of vehicles is evolving from a collection of sensor platforms that provide information to drivers and upload filtered sensor data (e.g., GPS location, road conditions, etc.) to Internet Servers; to a network of autonomous vehicles that exchange their sensor inputs among each other in order to optimize several different utility functions. One such function, and probably the most important for autonomous vehicles, is prompt delivery of the passengers to destination with maximum safety and comfort and minimum impact on the

environment. We are witnessing today in the vehicle fleet the same evolution that occurred ten years ago in the sensor domain from Sensor Web (i.e., sensors are accessible from the Internet to get their data) to Internet of Things (the computers with embedded sensors are networked with each other and make intelligent use of the sensors).

In the intelligent home, the IOT formed by the myriad of sensors and actuators that cover the house internally and externally, can manage all the utilities in the most economical way, with maximum comfort to residents and virtually no human intervention. Similarly, in the modern energy grid, the IOT consisting of all components large and small can manage power loads in a safe and efficient manner, with the operators now playing the role of observers.

In the vehicular grid, the Internet of Vehicles (IOV) is more complex than the smart home and smart energy grid IOTs. In fact there are many different "Things" in the IOV. Namely:

- External sensors (GPS, cameras, lidars, etc.)
- Internal automotive sensors and actuators (brakes, steering wheel, accelerator, etc.)
- Internal cockpit sensors (driver's state of health, alertness, tone of voice, health sensors like the Ford heart monitor seat, etc.)
- The driver's messages (e.g., tweets, Facebook) are also measurable sensor outputs that characterize the state of the system and of the driver.
- Vehicle's beacons, alarm reports on the vehicle state; say, position, key internal parameters, possible dangers, etc.

This complex picture (of sensors and stakeholders) tells us that IOVs are different from other IOTs. What sets them apart from other IOTs are the following properties/characteristics:

1. **High Mobility**: IoVs must manage the high mobility of vehicles and its impact on wireless communication

2. **Safety critical Applications**: this implies low latency and high reliability requirements

3. **Vehicle-to-Vehicle Communication**: short-range communication and limitations in wireless environments pose many challenges

4. **Privacy**: driver behavior and vehicular sensor data must be privately crowdsourced

In the vehicular network, like in all the other IOTs, when the human control is removed, the autonomous vehicles must efficiently cooperate to maintain smooth traffic flow in roads and highways. Visionaries predict that the self-driving vehicles will behave much better than human drivers, handling more traffic with lower delays, less pollution and better driver and passenger comfort. However, the complexity of the distributed control of hundreds of thousands of cars cannot be taken lightly. If a natural catastrophe suddenly happens, say an earthquake, the vehicles must be able to coordinate the evacuation of critical areas in a rapid and orderly manner. This requires the ability to efficiently communicate with each other and also to discover where the needed resources are (e.g., ambulances, police vehicles, information about escape routes, images about damage that must be avoided, etc.). Moreover, the communications must be secure, to prevent malicious attacks that in the case of autonomous vehicles could be literally deadly since there is no standby control and split second chance of intervention by the driver (who meantime may be surfing the web).

All of these functions, from efficient communications to distributed processing over various entities, will be provided by an emerging compute, communications and storage platform specifically designed for vehicles—the *Vehicular Cloud*. The Vehicular Cloud is justified by several observed trends:

1. Vehicles are becoming powerful sensor platforms (e.g., GPS, video cameras, pollution, radars)

2. Spectrum is becoming scarce => Internet upload of all the sensor outputs is expensive and infeasible

3. Cooperative data processing by vehicles rather than uploading to the Internet (e.g., pedestrians crossing, shock wave mitigation, platoon coordination)

To support the above functions, the mobile Vehicle Cloud provides several basic services, from routing to content search, through standard, open interfaces that are shared by all auto manufacturers. These functions are dependent upon *resilient* communications to maintain the fabric of the Vehicle Cloud. In addition, large scale *private* data collection is required in order to streamline the efficiency of the Vehicle Cloud components.

1.1 Our Results

In this work, we put forth new approaches for resilient mobile communications in disruptive networks by leveraging coded caches. We explore the trade-offs of coding versus no coding and introduce a mechanism that is resilient to pollution attacks by malicious intermediary caches. We then show how to reduce energy consumption by selectively enabling coding when required.

Next, we introduce a mobile architecture to achieve confidentiality and enforce role-based access control in disruptive networks even if infrastructure is not accessible. We leverage the cryptographic primitive attribute-based encryption to cryptographically enforce the access policy in the content itself.

Then, we show how to privately crowdsource information. We create a privacy module that privately releases data to applications querying GPS data via the GPSD daemon which runs on the majority of GPS enabled devices. We then show an order of magnitude scalability enhancement to the function secret sharing cryptographic primitive. Function secret sharing enables a group of data owners to privately upload data to a set of database servers that provides non-attributable writes as long as at least one database operator does not collude.

Finally, we introduce a privacy mechanism that improves the privacy strength while preserving utility. That is, we perform query expansion to reduce the information leakage due to an individual's participation.

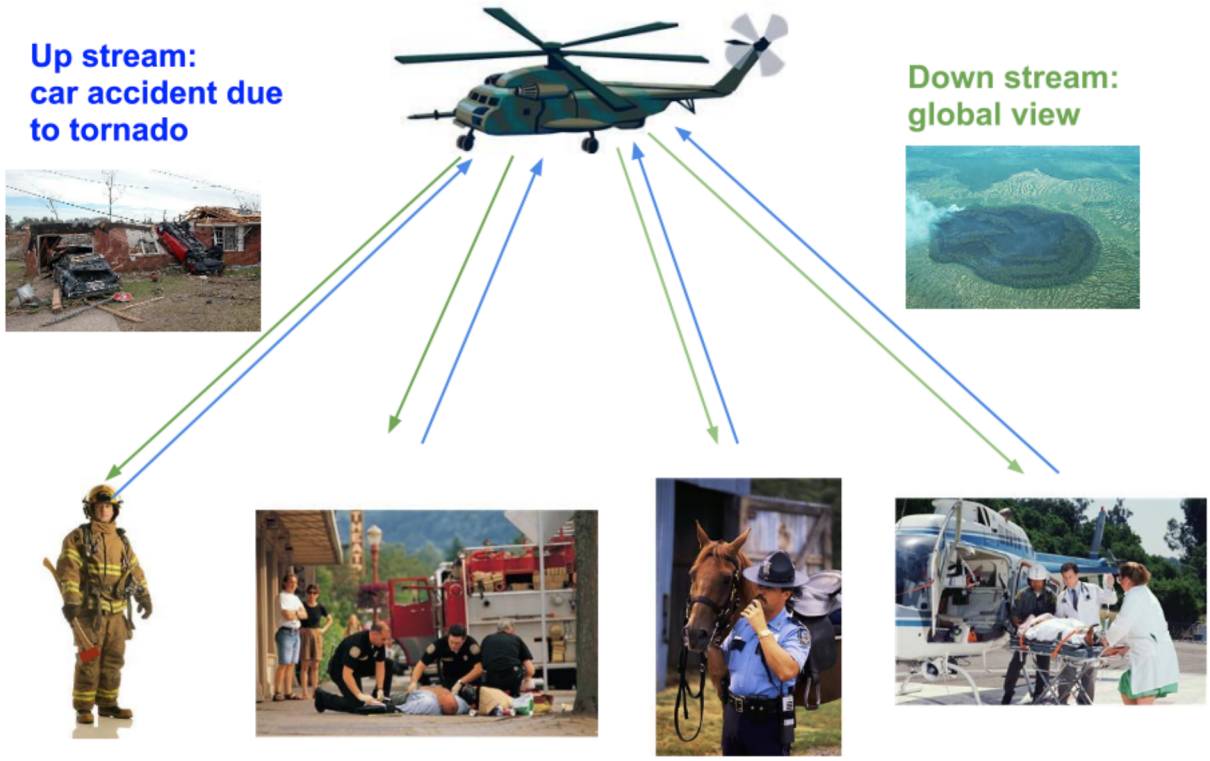

Figure 1.1: : First responders in an emergency situation share location and images with each other. Overhead, a helicopter circles transmitting the global view down to the first responders. Simulataneously, the first responders upload to the helicopter their local view of the situation. Due to the intermittent connectivity, only partial images are often received, cached and shared. The image file completion may require repeated contacts between ground teams with each other (exchanging the partial caches) and with the helicopter [NES, fir].

1.2 Brief Overview

Resilient Communication. Both tactile networks and first responders utilizing edge networks rely on situation awareness updates to arrive in a timely matter, even when the fixed infrastructure is unavailable. . The technical advancements of the commercial mobile phones make them capable of supporting such requirements under very disruptive network conditions.

Tactical and emergency response scenarios require efficient, robust, and secure network communications to quickly deliver data for situational awareness applications. The dynamic and resource limited constraints in these networks require that nodes opportunistically communicate with limited global knowledge, and make efficient use of the scarce available bandwidth. However, despite the remarkable increase in sensing, storage, processing, and communication capabilities in mobile devices, efficient dissemination and storage of content on the volatile network edge remains a challenge.

In this dissertation, we utilize coded blocks to increase the utilization of the scarce bandwidth and limited resources by providing receivers with content error recovery enabling them to fetch files in disruptive connectivity whereby *without* coded blocks the receivers would not be able to download the file. Thus, situational awareness in tactical and emergency response scenarios is increased by the use of coded blocks.

Private Data Collection. An open vehicular data testbed composed of detailed driver behaviors and real-world statistics would provide a greater understanding and help mitigate the number of self-driving vehicle accidents due to driver behavior misunderstandings. Ideally, such data should be gathered and aggregated into an open vehicular data testbed where manufacturers and regulators are able to test and validate vehicle safety standards. However, such a vehicle data testbed composed of detailed statistics should respect driver's privacy expectations.

In this dissertation, we design a privacy module to protect the release of location data that works across the majority of GPS enabled devices, we show an order of magnitude scalability enhancement to privately upload data over the default implementation, and we introduce a privacy mechanism that improves the privacy strength while preserving utility. We demonstrate the practicality of our approach by deploying our privacy system called CrowdZen to the UCLA campus.

1.3 Organization

We first introduce Cache Coding and show it's resiliency in disruptive networks as well as pollution protection in Chapter 3. We then show how to recognize the energy trade-offs of coding versus no coding by introduction Context Aware Cache Coding. In Chapter 6 we describe our approach for decentralized confidentiality and role-based access control without depending on infrastructure to enforce the policies in mobile edge networks using Attribute-Based Encryption. Finally, we describe our scalable privacy mechanism and implementation of LocationSafe in the GPS service daemon as well as the deployment of CrowdZen to the UCLA campus in Chapter 9.

CHAPTER 2

Preliminaries I: Mobile Information Centric Networks and Network Coding

2.1 Mobile Information Centric Networks

In information centric networks (ICN), the fundamental network primitive transitions from host-based addressing to content-based addressing (see examples [JST09d, ndn, Hag]). Each transmission unit (content block) is uniquely identified. Content is fetched by names, which allows intermediate nodes to act as either full or partial caches.

In dynamic, intermittent networks bandwidth is scarce. However, storage is becoming increasingly cheaper. The trend of increasingly cheap storage suggests to embrace the delay tolerant network philosophy of compensating for intermittent connectivity with, intermediate node caches. Store-and-forward content dissemination enables a requestor the ability to fetch content pieces from multiple sources. This allows retrieval of content from nearby caches without reaching all the way back to the original content source, which produces improved throughput especially in severely disruptive scenarios. In case of popular files, this allows requestors to download from multiple caches even when the origin is unreachable.

Mobile Ad Hoc Information Centric Networks (which we refer to as MANET ICN) have intermittent connectivity. This introduces the following challenges during file transfer and dissemination:

- *End-to-end connectivity not guaranteed*: A continuous path from source to destination may sometimes not occur. Nodes must therefore maintain coded caches of partial transfers and wait for contact with other nodes to continue transmission.

- *Partial caches*: Various caches contain different pieces of a file. A receiver does not know which caches contain which pieces of the file in advance and must ask each individual cache. It is important to parameterize the coding parameters to maximize throughput and minimize energy consumption.

- *Energy efficiency*: The tactical networks we are addressing here are largely comprised of handheld devices with limited battery life. Improving the resilience of the network requires an explicit tradeoff between energy consumption and transmission reliability. One of the key players in this tradeoff is network coding, as it improves reliability at the cost of extra energy.

We now examine how network coding can be used to address these issues in MANET ICN.

In intermittent networks, the following challenges affect file dissemination:

- *Last coupon problem*: Teams may form and split frequently, thus a file must be transmitted (and can be retrieved from caches) in a piecemeal fashion. Thus, pieces are received out of order. This makes it difficult for the requestor to reliably reconstruct a file.

- *Lack of end to end connectivity*: Hop by hop transmissions are required, with nodes acting as partial caches. Requestors must wait for the next contact opportunity to resume transmission.

- *Partial caches*: Various nodes contain different parts of a file.

- *Busy caches*: A requestor may find out that a cache which has the required pieces is busy serving other requestors. This causes the requestor to either wait for the next transmission opportunity or must locate another cache.

Content coding can help achieve reliable dissemination even when network partitions and severe disruptions occur and can address each of the above challenges. In particular:

- *Dispenses With Last Coupon Problem*: By using content coding, the last coupon problem is eliminated since with high probability each coded block received is innovative (i.e., helpful) and can be used to reconstruct the file. Thus, the throughput will be higher with content coding compared to not coding.

- *Overcoming Intermittent Connectivity*: Since transmissions are connectionless and hop-by-hop, we cache blocks at intermediate nodes. A requestor can then ask nearby caches for network coded blocks. The neighbors pull coded blocks from their cache and either transmit as they are or mix them and transmit new coded blocks.

- *Leverage Partial Caches*: Intermediate nodes cache partial files with innovative blocks. Since each block is helpful, nodes can make efficient use of limited connectivity by transmitting arbitrary innovative blocks.

- *Parallel Cache Download*: When a requestor finds a nearby cache busy to answer requests, it can ask other nearby caches for blocks since each network coded block is as helpful as any other.

Given the limited contact duration in MANET scenarios and the broadcast nature of wireless systems, the relays are most likely to obtain only partial files. The pieces (blocks) can be different from relay to relay, but some are replicated in several relays. When requesting a data object from multiple relays, the pieces are likely to be duplicate and arrive out of order. There will be gaps and missing pieces, which will make reliable reconstruction of the file difficult for a receiver. This is called the coupon collector problem (or last coupon problem). Network coding [HMK06, FBW06] has been used in MANET [LPY06] for data dissemination to overcome the last coupon problem.

We now briefly describe network coding. Then, we describe Cache Coding [JYP14], our form of network coding that provides full protection from pollution attacks.

2.2 Network Coding

Network coding [HMK06] has been used in mobile ad hoc networks [LPY06] for data dissemination to overcome the problem of intermittent connectivity. One major advantage of this approach is that the ordering of partially reassembled files is no longer needed for file transfer consistency. The requestor can retrieve arbitrary coded blocks from any node, and reassemble the original file from a sufficiently large number of linearly independent coded blocks without worrying about sorting them.

CodeTorrent and CodeCast has studied network coding in MANETs whereby coded blocks are broadcasted and mixed at intermediate caches [LPY06, PGL06]. By exploiting partial caches, unrestricted coding is able to greatly decrease the delay required to deliver files.

2.2.1 Network Coding Overview

The algorithm of network coding is as follows [HMK06, FBW06]. A source node publishes a file F. In order to disseminate the file in pieces using network coding, the source node first transforms F into a set of m vectors (e.g., chunks) $\mathbf{v_1}, ..., \mathbf{v_m}$ in an n-dimensional vector space over a finite field $\mathbb{GF}(2^8)$. These vectors are linearly combined by drawing, from the finite field $\mathbb{GF}(2^8)$, an encoding coefficient $\mathbf{e_i}$ to linearly combine with the vector to create m coded blocks $\mathbf{b_1}, ..., \mathbf{b_m}$. The set of these coefficients then forms the encoding vector $\mathbf{e}$ with $[\mathbf{e_1}, ..., \mathbf{e_n}]$. To reconstruct the file, a node must receive enough linearly independent coded blocks to be able to perform matrix inversion. First, we take the transpose of the received vectors such that: $\mathbf{E^T} = [\mathbf{e^T_1}, ..., \mathbf{e^T_n}]$, $\mathbf{B^T} = [\mathbf{b^T_1}, ..., \mathbf{b^T_n}]$, and $\mathbf{V^T} = [\mathbf{v^T_1}, ..., \mathbf{v^T_n}]$. Then we take $\mathbf{E^{-1}B}$ which will reconstruct all the original blocks in the file.

Note that the major advantage of network coding out-of-order blocks are no longer an issue. The requestor can retrieve coded blocks from any node, and reassemble the original file as long as it obtains a sufficient number of linearly independent blocks. Each independently generated coded blocks is equally innovative and useful to relays. The bandwidth saved by network coding due to control overhead and redundant data blocks at relays can be huge in

a disruptive MANET ICN.

However, network coding is easily attacked. The primary advantage of Cache Coding [JYP14] over network coding is that Cache Coding protects against pollution attacks while still achieving high throughput for file delivery. In Cache Coding, only sources and intermediate nodes have fully reconstructed the file, and are able to encode and propagate the coded blocks into the network.

2.2.2 Pollution Attacks

Traditional Random Linear Network Coding (RNLC) in MANETs performs random network coding packet mixing at intermediate nodes. The benefits of RLNC are reliable dissemination of files despite mobility, random interference, and losses. However, the downside is that pollution attacks become possible. A pollution attack occurs when a malicious (or faulty) node mixes invalid linear combinations of blocks into the network. These polluted blocks then get mixed with valid linear combinations and go undetected by honest intermediate nodes. The attack is detected only when the receiver is unable to reconstruct the original file, e.g. the reconstructed file hash does not match the original file hash. At this point, the entire file must be retransmitted from the source (if still available). To protect from pollution, homomorphic signatures (which are preserved through linear combinations) can be used. This provides non-repudiation and the ability to track and find malicious nodes. The drawback of homomorphic signatures is the Homomorphic NC processing cost, two order of magnitude higher than the Conventional NC mixing cost - a prohibitive proposition in heterogeneous MANETs that include smart phones [LGK11]. While there exist less costly alternatives for preventing pollution attacks, these solutions place limitation on topologies, require loose clock synchronization on the order of 100ms, limit the hop count, require large field sizes, or demand that new public keys be generated per generation. Obviously, these requirements are not feasible in dynamic CB-MANETs.

This leaves us with two NC alternatives. One option is to perform source only coding, whereby only the publisher performs network coding and signs all the blocks. Since only

the source codes and signs, non-repudiation is provided whereby the integrity of the blocks and the linear combination used to generate the blocks can be verified. Thus, receivers can identify pollution attacks and blacklist the malicious source. Another approach is to allow certified intermediate nodes that have fully reassembled the file to perform coding. To provide non-repudiation, the certified intermediate node also signs the regenerated blocks in addition to the originator. In both these cases (source coding and full cache coding), non-repudiation is provided and thus downstream nodes are protected from untraceable pollution attacks.

Homomorphic cryptography is computationally expensive and on the order of magnitude 2 times more expensive than unrestricted coding [LGK11]. This makes homomorphic cryptography infeasible for mobile devices such as smartphones.

More practical approaches for wireless networks have been proposed which utilize checksums [DCN09]. However, these approaches require the receiver to establish loose time synchronization with the sender. Additionally, attacker identification requires joint cooperation between the receiver and source. Both of these constraints are difficult if not impossible to achieve in CB-MANETs and DTN type environments.

Oh and Gerla showed that it is sufficient in a MANET for a small fraction of nodes to use homomorphic signatures with unrestricted network coding, while the other nodes simply forward [OG10]. This is useful in heterogeneous radio scenarios with powerful laptops and light smart phones internetworked in the battlefield. Untrusted nodes are only able to forward blocks; thus, signatures are preserved and pollution attacks are prevented. Only trusted nodes are able to code and append a secure "digest" so that downstream nodes can verify the digest and discard polluted blocks.

2.2.3 Network coding in ICNs

In content-based mobile ad hoc networks (CB-MANETs), random linear network coding (NC) can be used to reliably disseminate large files under intermittent connectivity. Conventional NC involves random unrestricted coding at intermediate nodes. This however is

vulnerable to pollution attacks. To avoid attacks, a brute force approach is to restrict the mixing at the source. However, source restricted NC generally reduces the robustness of the code in the face of errors, losses and mobility induced intermittence. CB-MANETs introduce a new option. Caching is common in CB MANETs and a fully reassembled cached file can be viewed as a new source. Thus, NC packets can be mixed at all sources (including the originator and the intermediate caches) yet still providing protection from pollution. The hypothesis we wish to test in this paper is whether in CB-MANETs with sufficient caches of a file, the performance (in terms of robustness) of the restricted coding equals that of unrestricted coding.

In this paper, we examine and compare unrestricted coding to full cache coding, source only coding, and no coding. As expected, we find that full cache coding remains competitive with unrestricted coding while maintaining full protection against pollution attacks.

Montpetit et al. [MWT12] have identified network coding within content-centric networking (CCN) as a strategy with tremendous potential. However, their work only presented an architecture, rather than reporting on an empirical analysis. Further, we have introduced the idea of adaptively enabling network coding, explained the need for context-aware network coding to preserve energy resources on resource constrained devices, and implemented our ideas on Android devices. We have also emulated the system, the network-wide effects, and reported our findings.

Wu et al. [WLX13] have implemented and evaluated the benefits that network coding provides for cache hit rates in content centric networks. Interestingly, the evaluation used unrestricted coding applied to real traces from PPTV (peer-to-peer video streaming). However, their work does not address selectively enabling network coding, which may not always be required in a wired peer-to-peer system. Moreover, it does not address pollution attack protection in unrestricted coding. Our solution detects when to enable network coding, a characteristic that is of critical importance in mobile environments with limited power.

2.2.4 Context-aware network coding

Previous work on network coding based on context has mostly been associated with forwarding in disrupted networks. It has focused on adjusting the degree of redundancy in the encoding. MORE [CJK07] is the earliest work that proposes adaptive network coding for such environments. MORE relays opportunistically form multiple paths through which packets are re-encoded and forwarded. CodeMP [CTY12] further studies adaptive network coding based on measured loss rates that affect TCP sessions. While this family of adaptive network coding approaches adjust the use of multiple paths and degree of redundancy using link loss rate estimates, they focus on connected networks with unstable channels. In contrast, we apply network coding in delay-tolerant networking scenarios. We have also conducted our experiments with more realistic channel models.

Existing work on using network coding in delay-tolerant networks (DTNs [MF09]) focuses on reducing the number of transmissions needed for epidemic routing or probabilistic forwarding. Lin et al. [LLL08] studied the tradeoff between performance and resource consumption in DTNs. They propose spreading more coded packets than are needed to reduce the number of transmissions required. Chuah et al. [CYR12] proposed CANCO, which only spreads coded packets among some of the nodes encountered. Delivery predictability and friendliness are used as metrics to decide which nodes to use.

Our work differs from previous work in several respects. First, our goal is to maximize the delivery rate while reducing the energy consumption by selectively enabling and disabling network coding. Prior work is unable to adapt to improved network conditions. In such cases, always-on network coding imposes a cost, and depletes precious power on resource constrained devices. Second, the context we use is the condition of the network, rather than the history of encounters or social relationships with other nodes. Third, we employ network coding by broadcasting blocks over UDP connections to neighbors, rather than reducing transmissions by forwarding to only a subset of nodes. More importantly, we provide an algorithm and evaluate an implementation that automatically switches between fragmenting and network coding the content, based on the runtime context. Of equal significance, we

have performed emulations of real-world scenarios and implemented the system on Android devices. In contrast, earlier work on context-aware and adaptive network coding is limited to theoretical analysis or simulations of limited scenarios. It is worth noting that CACC can be used with previously proposed coding-aware routing protocols, such as [LLL08] and [CYR12].

CHAPTER 3

Resilient Mobile Edge Networking

With Cache Coding [JYP14], the content originator encodes the file being transferred. Intermediate nodes can re-encode only if they have cached the file. More precisely, each file consists of file fragments. The originator transforms the file into coded blocks, which are linear combinations of fragments of the file. These coded blocks are then propagated in the network. Note that in this scheme, a node can code only if it has possession of the entire file in its cache. Such nodes are either the originator or an intermediate node that happens to have cached the whole file. Thus, both originators and intermediate caches perform Cache Coding.

Cache Coding, like network coding, improves file delivery in real-world scenarios with severely disrupted networks. The benefits are summarized here:

- *Overcoming intermittent connectivity*: Since end-to-end connectivity is not guaranteed, blocks are cached at intermediate nodes. A requestor can ask nearby nodes for network coded blocks. The neighbors pull coded blocks from their cache and, depending on context, either transmit them as they are or mix them and transmit new coded blocks.

- *Exploiting partial caches*: Nodes cache partial files that are encoded in the form of linearly independent "innovative" blocks. Since each coded block is as useful for decoding as any another, a requestor need not contact a particular cache to find missing blocks. Any node that has blocks for that file will do.

- *Leveraging alternative peer caches*: When a requestor discovers that the nearest cache is offline or is too busy to answer requests, it can ask other nearby coded caches for blocks since each network coded block is as useful as any other for decoding the file.

- *Efficient battery usage*: It has previously been shown that network coding achieves the minimum energy per bit required for reliable dissemination of files in environments with intermittent connectivity [WCK05, LMH04]. However, network coding leads to shorter battery lifetimes due to the processing overhead [KR08, FWL06]. Our contribution, Context-Aware Cache Coding specifically addresses this problem. It detects when Cache Coding is required based on the context of the transfer. It then forwards packets as they come in, without extra code processing overhead, thus saving precious energy. This allows nodes to limit the use of Cache Coding (and corresponding energy consumption) to the situations when it is needed. At all other times, Cache Coding is automatically disabled.

3.1 Cache Coding Basics

Cache Coding provides good diversity, which is necessary for network coding efficiency, without requiring unrestricted network coding at all intermediate nodes. If we recode at intermediate nodes (without Cache Coding), we must protect the data from pollution attacks, e.g., by using homomorphic codes. With Cache Coding, a node signs the cache before mixing packets. When the generation is decoded, if it fails the signature, it is discarded. Because signed blocks cannot be repudiated, malicious intermediate nodes are easily detected.

The contribution of this dissertation is two-fold. First, we demonstrate that Cache Coding must be used to ensure reliable file delivery in situations where intermittent connectivity losses and/or severe network disruptions occur. Secondly, we introduce an energy efficient, Context-Aware Cache Coding scheme (CACC) that adapts to network conditions and deployed applications. Using simple metrics, such as link loss rate and file size, CACC identifies the context in which Cache Coding is needed and then enforces it. Both emulation and real-world deployment on Android-based smartphones show that CACC

3.2 Full Cache Coding

In CB-MANETs, files are opportunistically cached at mobile nodes to favor future file requests. Files can be downloaded in parallel from multiple caches to make downloads reliable and fast. Network coding across parallel caches further improves the throughput. However, in intermittent connectivity, caches may often be partial. Thus, these caches cannot be signed since the signature implies that the intermediate node has received the full file, has verified the signature and has replaced in each block the originator signature with its own. Note that an intermediate cache can reconstruct the file from contributions from different caches which is the traditional unrestricted coding. One may then state that the full cache strategy is like the unrestricted strategy in the following manner. As soon as an intermediate node decides to mix, it must fully reassemble the file and verify integrity before it reissues newly mixed packets. As we shall see, this intermediate full cache mixing can improve performance considerably as compared to source only coding.

The above implies that the full cache strategy must be network coding aware. The question is whether a node should fully cache and decode/recode before forwarding (and signing) or should just forward the blocks as it receives them, no signature required. There is a trade off between reassembly delay (-) and improved orthogonality (ie. linear independence) of the packets (+). We will show that in some cases we can achieve higher throughput if we wait for the full cache.

3.2.1 Pollution Protection With Full Caches

There are two types of pollution attacks to consider. The first is whereby a malicious node mixes and corrupts the coefficients such that downstream nodes are never able to successfully decode. The second attack occurs when blocks are polluted in such a way that downstream nodes are able to decode; yet, the reconstructed file's signature does not match the original file signature.

By pollution protection with full caches only, we mean the following. The file is first signed by the source, thus providing authentication, integrity, and non-repudiation. The

signature can be saved in the header of the payload. Once an intermediate cache receives the full file, it verifies the source signature. Once the source signature has been validated, only then does the intermediate cache now assume responsibility for the integrity and non-repudiation. The file is random linearly network coded, and each block is signed by the cache owner (the intermediate node). Signing each block provides non-repudiation. By non-repudiation, we mean the following. If coded blocks from cache A cannot be decoded in spite of the collection of a full rank set, cache A is black-listed and avoided in the future (or inspected for faulty software). Now, with source signatures when the file is published and with intermediate node signatures after full cache reconstruction, the system is fully protected from pollution attacks.

Thus, to recover from either form of pollution attack, a receiver takes the following actions. Suppose a node receives from N caches and cannot decode. The receiver then requests blocks from a single cache at a time. The receiver must try to decode data from one cache at a time in order to isolate the faulty cache. The cache that provides an un-decodable stream or faulty signature is the polluter and must be investigated.

3.3 System Description

The CB-MANET system used to evaluate the different NC pollution protection strategies subsumes both the family of peer-to-peer content dissemination network (e.g. Haggle [SHC06, SSH07, NGR]) as well as the family of Content-Based Networks in which all blocks are uniquely identifiable (e.g. NDN [Zha10, JST12, JST09a, Zha11]). To improve the delivery of content, we segment a large file into blocks. The transmissions are performed in the unit of data blocks and all blocks are named as filename/blockID. In the case of network coding, the blocks of a file are encoded as coded blocks and the block IDs are randomly generated.

3.3.1 CB-MANET System Operation

The basic operation of our system is as follows. Three messages are periodically broadcast at each node: interest, request, and cache summary. All three messages are represented in the form of bloom filters. The interest represents a collection of file names a node itself wants, and may be opportunistically disseminated over multiple hops when the bandwidth is sufficient. Relays have full control on when and which interests should be propagated. The request represents a collection of file the node is willing to receive at the time the request is sent. Note that the interest is separate from the request so that the node has the right to decide what contents to request based on the volume of interests it receive, the network condition, and its local content prioritization policy. Requests are broadcast only one hop to retrieve available contents from neighbors. To assist the prioritization and compactness of requests, nodes also periodically broadcast their cache summaries (in chunks or files, depending on the completeness of the data at this node.) The cache summaries are leveraged by neighbor nodes to decide which files/chunks to send. Cache summaries are useful in terms of reducing bandwidth waste, as nodes may blindly push redundant file/chunk based on a request for a large file. Additionally, the nodes also update neighborsćache summaries based on the control messages and data communication they hear.

Data transmissions may be triggered when a new request comes in or when a new data is received. When a new request comes in, the node examines the request with the data it currently has and the requesters' cache summary, and initiates data transmissions for the data that matches the requesterś request. A three-way handshake procedure is assciated with each data block transmission to eliminate redundant transmission in the broadcast network. For each data block, the sending node first sends a Request-To-Send-Block (RTSB), which carries the block name, to the target node. Upon receiving an RTSB, the target node replies a RTSB-Reply, which may accept or reject the block. If the block is rejected, a reject code is carried in RTSB-Reply to indicate one of the three reasons to reject: (1) The block is already received (2) The file is already received (3) The block is being sent by other neighbors. The data is then transmitted if accepted. Once the target node receives the data, it acknowledges

by an ACK. Note that all neighbors of the target node may update their cache summaries based on the broadcast RTSB-Reply and ACK.

When a data block is received, the receiver may propagate the data back to its original requester(s) by checking all requests it received from neighbors. If matches are found, the receiver (now becomes the sending node) starts another three-way handshake to deliver the block. In this way, the file or blocks are delivered back to the original requestors via the trail of breadcrumbs in a multi-hop environment.

3.4 Energy Efficient and Context-Aware

While Cache Coding has the advantages as previously described, it requires extra bandwidth to carry the coefficients and handle the computational resources dedicated to the encoding/decoding processes. The tradeoffs of performance gain and overhead for disruptive MANET ICNs must be managed by our proposed energy-efficient CACC.

Our proposed contribution CACC aims to adaptively switch between Cache Coding, low-overhead fragmentation, and atomic transmission of data objects. Depending on the context, CACC automatically turns Cache Coding on and off for a given data object. The objective is to eliminate unnecessary bandwidth consumption due to increased header size and processing overhead when Cache Coding is unlikely to improve end-to-end performance, but be ready to trigger Cache Coding instantly in emergencies by using context indicators.

3.4.1 Context Indicators

The nature of Cache Coding methods requires fragmenting the data object being transmitted to utilize the benefit of transmitting random coded blocks, either when frequent retransmissions are required from one source or when multiple content sources may be leveraged. However, in the case when a data object is small, fragmenting that object may be less efficient than simply requesting an atomic re-transmission due to the fact that identification of the fragments requires additional overhead. Moreover, Cache Coding incurs even higher

overhead due to the required coefficients in each coded block. Therefore, Cache Coding should remain disabled in the case when the file size is relatively small compared to the basic fragment unit.

The major advantage of Cache Coding is its resilience to intermittent links due to high mobility. This advantage comes from the fact that Cache Coding is randomized and all coded blocks contain equal entropy. When the link breaks down often, it is difficult to predict which fragmented blocks get lost without proper feedback. With Cache Coding, however, all sources/caches can send innovative blocks (i.e., each one of them is linearly independent with the rest of the received fragments) without waiting for feedbacks, which leads to a higher chance of finishing full file transmission given a reasonably short period of time over one hop. Another important factor to consider is the coefficient overhead of Cache Coding. The coefficient overhead increases the transmission unit size and may worsen the performance. Therefore, it is expected that Cache Coding should be disabled when the link is not lossy.

Based on the above observations, we propose two context indicators.

- **Application-related context indicator**: The *data object size* needs to be considered when deciding whether the data object is more suitable for Cache Coding (i.e., the overhead can be compensated) vs. atomic (re-)transmission.
- **Network condition-related context indicator**: When the connectivity is intermittent and the contact time is short, the *link loss rate* can reflect the current situation (i.e., network condition and node mobility) to determine whether Cache Coding can speed up the data delivery from multiple sources/caches.

3.4.2 CACC

Our adaptive Cache Coding algorithm takes the following steps. By default, Cache Coding is disabled for new data objects and destination nodes.

1. CACC monitors the link loss rate for all known one-hop neighbors. The loss rate

estimation is simply a weighted moving average over a fixed interval of time. The link loss rate is estimated based on the number of successful receptions of periodic beacons at the link layer. The beacon interval $\triangle t$ is parameterized and can be adjusted to vary sensitivity. For downstream nodes, if the downstream link quality is good, fragmentation can be used for subsequent hops after receiving Cache Coded blocks and reconstructing content. We use the following equations to compute the loss rate estimate:

$$\alpha = 1 - e^{\frac{\triangle t}{-W}} \tag{3.1}$$

$$l_n(t) = \alpha \cdot b_i + (1 - \alpha) \cdot l_n(t - 1) \tag{3.2}$$

where W is defined as the length of the interval of time to measure the loss estimate over, $\triangle t$ is defined as the duration between beacons, and b_i is 1 if no beacon was received (indicating a loss) and 0 if a beacon was received (indicating no loss). $l_n(0)$ is defined to be 1, indicating that we perform cache coding until the loss rate settles down to a point where we can safely stop using it.

2. At the sender side (i.e., the sender can be a cache or a data source), Cache Coding is enabled for data objects that satisfy the following two conditions:

$$l_n(t) > l_{th} \tag{3.3}$$

$$s(d) > s_{th} \tag{3.4}$$

where l_{th} and s_{th} are the threshold values of link loss rate and data object size, respectively. $l_n(t)$ is the average link loss rate at time t for the 1-hop neighbor node n. $s(d)$ is the size of data object d.

Note that even relay nodes that are not data sources nor caches (i.e., did not issue an interest for this file) must apply the CACC procedure. More precisely, if the link loss criterion determines that Cache Coding should be used, the Relay must start accumulating blocks of the file in question until it has cached the entire file. At that point it switches to Cache Coding. It is intuitive to understand why arbitrary relays

(not interested in the file) must cache code. In fact, a relay node may be attacked by an adversarial jammer and drop all the packets. Recovery is possible only if the Relay nodes on the cut-set under attack enter the Cache Code mode, thus providing enough diversity to overcome the attack.

3. At the receiver side (i.e., the receiver may be a relay node or the intended receiver), if a Cache Coded block is received from any of its neighbors, the previously received un-encoded fragments are converted to encoded blocks.

The conversion from an un-encoded fragment to an encoded block allows CACC to recycle already received fragments rather than waiting sufficient numbers of innovative blocks are generated from a source and disseminated to a receiver. In CACC, the i-th fragment of a file can be seen as a special case of coded block in which the coefficients used to encode this block is a unit vector in which the i-th element is 1 and all other elements are 0. Therefore, the fragment can be converted to a coded block by simply encoding it with such a unit vector. For example, to convert the second fragment f_2 of a 4-fragment file, we compute the converted coded block by $[0, 1, 0, 0][\mathbf{0}, \boldsymbol{f}_2, \mathbf{0}, \mathbf{0}]$. In this way, all received fragments and encoded blocks can be used for decoding the original data object so that all successful transmissions are utilized.

CHAPTER 4

Evaluation I: Mobile Information Centric Networks and Network Coding

Unrestricted network coding, in which intermediate nodes perform unrestricted mixing (as opposed to mixing of fragments from the cached file), offers more diversity than Cache Coding. The restriction to Cache Coding thus accepts a greater cost in performance. However, as mentioned earlier, unrestricted coding is vulnerable to devastating pollution attacks launched by malicious intermediate nodes, and can corrupt all downstream blocks. Conventional solutions use homomorphic signatures to overcome these pollution attacks. Unfortunately, homomorphic codes are typically 100 times more processor-intensive than regular networking codes, and have a 100-fold increase in expenditure [LGK11]. Cache Coding provides full protection against pollution attacks while avoiding the extra processing overhead of homomorphic coding [JYP14]. This comes at the cost of a modest loss in code diversity and thus throughput efficiency, which is offset by enormous processing and energy savings.

The remainder of this section discusses our system and the alternatives to unrestricted coding. We then perform a throughput comparison of the three options: unrestricted coding, full cache only coding, and finally source only coding.

4.1 Implementation

We implemented and evaluated our context-aware cache coding (CACC) scheme in an ICN architecture called Information-CEntric Mobile Ad hoc Networking (ICEMAN) which was developed for the DARPA Content-Based Mobile Edge Networking (CBMEN) [dar13, WMJ15, WMJ13, JYG13]. ICEMAN is a modular, open-source, content-based mobile net-

working framework with a simple but expressive attribute-based mechanism for describing content and expressing interest, and in-network resolution to offer content dissemination by matching content to interest. Further details can be found in [enc15].

The ICEMAN architecture [WMJ15, WMJ13] is event-driven, modular, and layer-less, which provides flexibility and scalability. Central in the architecture is the kernel. It implements an event queue, over which managers that implement the functional logic communicate. Managers are responsible for specific tasks such as managing communication interfaces, encapsulating a set of protocols, and forwarding content.

A total of 45K lines of C++ code were added or modified to ICEMAN to implement ICEMAN (among which 10K lines of code were added for cache coding) on Linux and Android systems. ICEMAN has been tested and run on up to 30 real deployed Android phones. ICEMAN's cache coding feature is based on CodeTorrent (available at `https://github.com/uclanrl/codetorrent`), which has been further extended to support context-awareness as proposed by CACC.

In ICEMAN, files are opportunistically cached at mobile nodes to favor future file requests. Files can be downloaded in parallel from multiple caches to make downloads reliable and fast. Cache Coding across parallel caches further improves the throughput. However, in intermittent connectivity, caches may often be partial. Thus, these caches cannot be signed since the signature implies that the intermediate node has received the full file, has verified the signature and has replaced in each block the originator signature with its own.

The traditional unrestricted network coding in which all relays may mix (i.e., re-encode) any available chunks even if the cached data object is partial, exposes security concerns in an ICN due to the possibility of pollution attack. Therefore, we apply the *cache coding*, which allows only the caches who have the full data objects to reencode, generate, and sign new coded blocks. Caches holding only partial data objects are only allowed to forward the coded blocks they have received as is. In this way, attackers can be identified and blacklisted by looking at the signature.

In ICEMAN, interests are composed of key-value pairs to describe the content the re-

questor is searching for. Suppose node n broadcasts an interest and gets metadata notification from the source node s as a match to its interest. Node n then starts downloading the blocks from node s. During this process, node n learns the unique content identifier (Cid) of the file carried in each block, which can be used to optimize the content flow of blocks.

For this purpose, we introduce the concept of *precise* interests when the requestor learns, from its neighbors, Cid for which it is searching for. Using Cid, node n can check the Bloom filters received from other neighbors and determine if there are other caches for file f. Knowing that other neighbors have a copy of f, n can send them *precise* interests, which is essentially Cid of f and can start pulling the blocks from multiple caches in parallel. This multiple cache downloading via precise interests improves the performance of Cache Coding. In this case, Bloom filters (cache summaries that disseminated side by side with interests to provide a view of the files available, both complete and incomplete files) can serve as routing tables to indicate the direction to the cache.

CACC switches between *cache coding* and low-overhead fragmentation by generating an *event* in ICEMAN architecture. The switching occurs at a sender for specific target nodes; that is, the decision is based on each pairwise link. The link loss rates are calculated as a moving average of the percentage of lost beacon packets over given period of time. To measure the packet loss, an event is sent from the Ethernet connectivity module of ICEMAN to notify the loss estimation module upon successfully receiving a beacon. The loss estimation module then decides to generate another event to CACC that in turn triggers switching between Cache Coding and fragmentation. A node switches to Cache Coding for data transmissions targeting nodes associated with the high loss links, and switches back to fragmentation on other links upon receiving this event. The implementation of CACC parameterizes l_{th} and s_{th}, via user-defined configuration files.

4.2 Model

Consider the corridor model that used in Oh and Gerla's 2009 paper [OGT09]. The model depict two possible multipath configurations in a MANET, with perfectly disjoint and highly

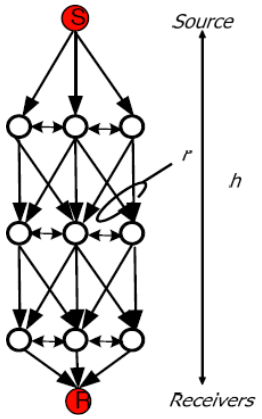

Figure 4.1: 1-3-3-1 corridor model [OGT09]. There is a single node which broadcasts at the top. A single receiver subscribes to all files.

interfering paths respectively. The reality will be "in the middle", so we will study both cases and argue about average behavior.

In our case, the origin of the file is the node S. Node R has issued an interest for the file. The interest has traced several paths (as shown in Fig (a) and (b)). The file is splits into blocks which are broadcasted on the mesh (or braid) created by the interests. The broadcast mode precludes MAC layer ACKs so, no loss detection and retransmissions.

4.2.1 No Coding

First we consider the non coded file transmission. Each packet is triplicated by broadcast. Thus, 3 copies travel along the braid. With some probability a packet will be lost and the file cannot be properly received. Note that the probability of packet loss is much higher in (a) than in (b), because of greater (b) redundancy. However, (b) is slower than (a). More precisely, (b) is three times slower than (a), so in principle we could improve redundancy in (a) by transmitting each block 3 times. In general, with jamming and mobility, this non coding strategy becomes unacceptable.

4.2.2 Unrestricted Coding

Consider first the braid (a). If packets are transmitted one at the time as soon as the get in, no mixing is possible and the performance is the same as for the non coded file. In order to benefit from mixing we must accumulate at least 2 blocks and mix them to generate new mixed blocks. In this case, if a block is lost on a strand of the braid, the next coded block will allow recovery. The more blocks we accumulate in a queue, the more losses we can recover. Naturally, when the blocks are merged at the end, the triple redundancy of the three strands also comes to help. Next, consider the braid (b). In this case, because of the interconnection between the paths, two or three blocks are accumulated in each queue at each stage. These blocks can be mixed and will allow the recovery of the lost blocks. In summary, to exploit NC, we must ACCUMULATE AND MIX at intermediate nodes, at the expense of a few block delays. If blocks are not mixed, the performance is the same as for no coding

4.2.3 Full Cache Only Coding

Suppose the blocks are coded at the source and sent out in the network, ie they are broadcast on the braid. If there is no intermediate node mixing, the performance is the same as for uncoded file, i.e. potentially very bad. To improve the performance, the top three nodes in the corridor will assemble the file, while transmitting to the nodes below.

Once the top nodes have assembled the files, the mix them again and they transmit as many blocks as necessary to compensate for the lost blocks. In this particular case, the same outcome is achieved by transmitting additional blocks from the source. To guarantee that no blocks are repeated, each block is mixed (from the blocks in the cache) at the time it is transmitted. Loss recovery is very good. In fact as good as in the intermediate mixing case. Suppose ALL the blocks of the first broadcast are lost. Once the three caches start emptying their blocks, just 1/3 of blocks from each cache will suffice to recover the entire file. This is an interesting observation. If instead of building the three caches, we just relied on the source to transmit more coded blocks, we would take three times more to recover.

4.2.4 Pre-Existing Caches

Suppose now that there are several preexisting caches and that the interest query reaches multiple caches. If the caches are full caches, we can assume that they will be remixed as they are transmitted, so diversity is guaranteed and the recovery from loss very good. If the caches are partial caches and remixing is allowed, the mixing can be helpful if the caches are overlapped in the vector space. If the caches are disjoint subsets, as would be the case of a UAV spewing out the full coded file and 3 soldiers getting each a different 1/3 of the file, the mixing would not help much when they join and try to assemble the full file. Mixing will help if there is significant overlap in the files, as it will increase diversity. Anyway, as we saw earlier, if the soldier coded files already have built in diversity, further mixing at intermediate nodes will not help much. In summary, even with partial caches, the intermediate node mixing in general does not buy much.

4.2.5 When does intermediate mixing help?

Intuition suggests that intermediate node mixing helps if there is a sudden upsurge of losses and jamming that requires added redundancy. Suppose that the file is transmitted by the source in encoded fashion. Everything is fine for a few wireless hops and no copies are made nor mixing occurs at intermediate nodes. Until at some depth of the network intense jamming occurs that causes, say 90With the caching strategy, one solution is actually possible if mixing upon full cache is allowed, namely reassembling the entire file at the upstream nodes and releasing freshly mixed blocks thereafter.

4.2.6 Hypothesis

We have the following hypothesis: (1) intermediate full cache re-encoding provides better performance than source only network coding and (2) intermediate mixing is after all not very critical if diverse caches (full or not full) can be tapped in the network.

Our explanation is as follows:

- *Explanation of hypothesis 1* (i.e., full cache re-encoding is better than source only coding). Suppose a receiver receives packets from two streams. If the streams have been independently coded by the upstream caches, the probability of getting a full rank set is higher than if the neighbors provide two identical sets.

- *Explanation of hypothesis 2* (i.e., full cache re-encoding provides comparable performance to unrestricted coding). In our scenarios, the main advantage of unrestricted coding is that even partial caches and, in the limited, nodes with two packets can mix thus creating more diversity. Also, unrestricted coding incurs less latency since it need not reassemble the entire file before mixing. However, unrestricted coding must accumulate and mix. Even in the extreme conditions when all else (but intermediate mixing) fails, the ability to create a full cache at node upstream of the critical section would save the day.

4.3 Evaluation: Cache Coding

We evaluated the throughput of unrestricted coding, full-cache coding, source-only coding, and finally no coding. We examine both a static topology using the corridor model [OGT09] with varying levels of packet loss and a mobile model using random waypoint. For simplicity, we evaluate a single generation, though our results are generalizable to multiple generations as our technique is not bound to generations.

4.3.1 Evaluation Description

Our test scenarios utilize multiple publishers disseminating using broadcast. The network is intermittent due to interference and induced packet loss, and in the dynamic case mobility. Many caches are partial; however, a receiver can download from multiple caches in parallel. Due to using broadcast there are no retransmissions. Redundancy is provided by multiple paths. After a time, the decoder discards the file that cannot be decoded. In both scenarios we perform our evaluation using Qualnet 6.1. The radio range is about 70 meters.

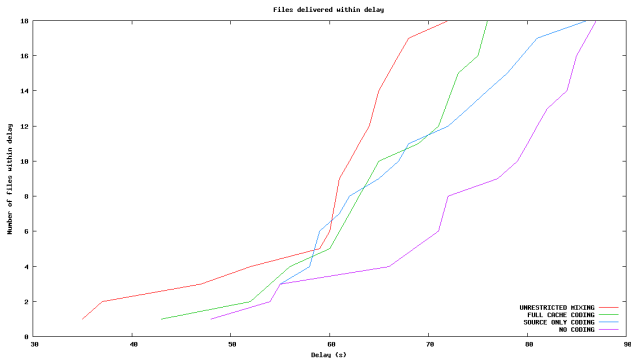

Figure 4.2: Corridor model with 30% packet loss. Single publisher and single downstream receiver with partial and full intermediate caches.

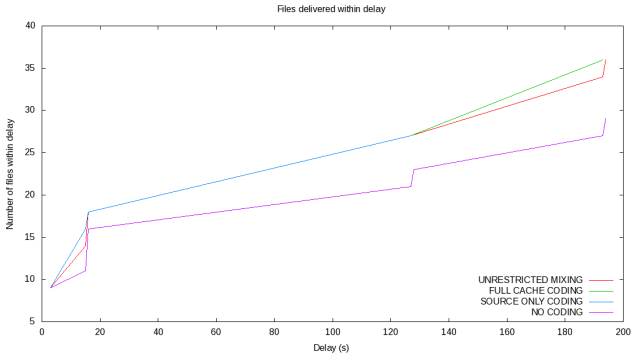

Figure 4.3: 10 node mobility with 3 publishers and 7 receivers.

4.3.2 Results

In our static scenario as seen in Figure 4.2, we observe that as expected, full cache coding competes with unrestricted coding. This is due to the ability for the intermediate nodes to be able to quickly reconstruct the original file in parallel from the multiple publishers. The intermediate nodes act as seeds to propagate the file to the remaining receivers. Source only coding delivers files quicker than no coding. However, due to the bandwidth and processing overhead of network coding due to the encoding vector, source only coding is not able to as efficiently utilize the parallel caches as well as full cache coding.

During mobility as seen in Figure 4.3, full cache coding performs as well as unrestricted mixing. As we observed in our hypothesis, unrestricted mixing gains its power when it is able to accumulate and mix new innovative blocks. During mobility, unrestricted mixing is not

able to accumulate enough new innovative blocks, where the partial cache outperforms simply forwarding blocks. In the case of the full cache, unrestricted coding generates innovative blocks; however, so does full cache coding. This leads to the equivalent throughput.

4.4 Evaluation: Context Aware Cache Coding

It should be noted that in an ICN architecture the decision to cache or not to cache at intermediate nodes can also be made context-aware. If an intermediate node is already a receiver, i.e., a requester, then the cache is available for free. However, if the intermediate note did not issue an interest for this file, then caching is an extra cost. If the link loss from the intermediate node to destination is high (e.g., due to enemy jamming), then caching is appropriate since this gives us more diversity and eventually more throughput. This way context awareness is used to optimize the trade off between energy (i.e., processing cost) and performance in the case of Cache Coding, initial coding, or no coding. However, in our evaluation, we only focus on using context awareness to determine when to enable Cache Coding.

4.4.1 Micro Benchmark

We first evaluate the performance overhead of Cache Coding versus fragmentation in a simple 2-node scenario. We run a 6 minute scenario in which two nodes publish 60 objects each, in order to measure the overhead of NetCode versus Fragmentation under ideal conditions on a resource constrained device.

In the Micro Benchmark, there is good connectivity between the two Android phones, and Frag performs better. In Table 4.1, NetCode imposes extra overhead due to the expensive coding operations, while Frag allows data to be transmitted quickly and with lower power usage. This motivates the development of CACC, which switches intelligently between them.

	Files Delivered	Normalized Energy Utility
Frag	30	1.0
NetCode	20	0.33

Table 4.1: Micro Benchmark: In static scenarios with good connectivity, NetCode has uncessary overhead.

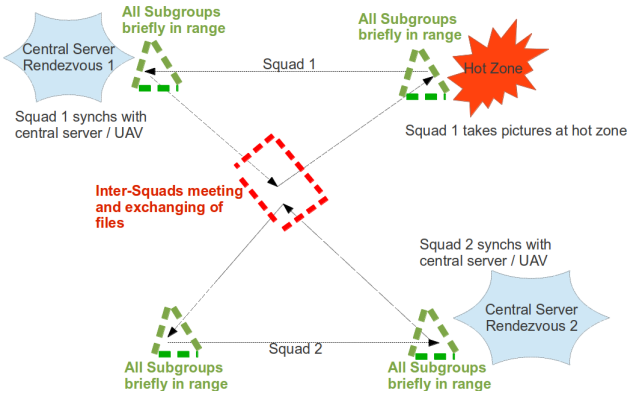

Figure 4.4: Search patrol scenario with 30 nodes: 2 squads (composed of 3 sub-squads) walk in a triangle pattern between 3 rendezvous points.

4.4.2 Scenarios

Our evaluation tests CACC across two scenarios: a search patrol scenario in Section 4.4.2.1, and a data mule scenario in Section 4.4.2.2; to illustrate the behaviour of CACC in dynamic settings.

4.4.2.1 Search Patrol Scenario

Consider the search patrol model in Figure 4.4, where a search patrol visits rendezvous points and shares reconnaissance and updates information. In this search patrol scenario, there are 2 squads with 14 members each. Each squad is composed of 2 subgroups. Intra-squad communication occurs periodically at each rendezvous point (i.e., central servers, hot spot, and red box in Figure 4.4).

Both the central servers publish two 512KB files each. Each scout in each subgroup takes a 1MB picture at the hot zone and performs intra-subgroup sharing. This results in a total of 60 files published and shared to a total of 30 subscribers. Intra-squad sharing occurs only at each rendezvous point. Inter-squad file sharing occurs in the middle ground as illustrated as the red box in Figure 4.4. Due to intermittent connectivity, each node only has partial caches. The partial caches consist of the files from the central servers, and pictures taken at the hot zone. If there are partial files, intra-squad sharing continues as each squad walks to next rendezvous point. This scenario continues for 720 seconds.

4.4.2.2 Data Mule Scenario

In the data mule model, two large squads (4x4 and 3x4 members each) are connected by a few data mules which ferry files. Nodes in a squad are 30 meters apart. Intra-squad communication is conducted by multi-hop connections while inter-squad communication is facilitated by three data mules. In this scenario, every node in each squads publishes two 1MB files, one every five minutes. The three data mules publish five 512K files, one every minute. All nodes subscribe to all content. The data mule speed is 1.4 m/s. Each data mule is out of range of other data mules and communicates with each squad for 60 seconds. The duration of this scenario is 600 seconds. This model allows us to evaluate the effects of large, internally connected groups that are interconnected with each other by a low-bandwidth connection. We expect to see the coupon collector problem taking effect and slowing down the file reconstruction between the two groups.

4.4.3 Setup

In order to evaluate the performance on larger mobile scenarios, our test framework emulates mobile network scenarios on a Linux server. The network emulation is done using EMANE 802.11 [ema]. EMANE includes a path loss model and interfaces with the Common Open Resource Emulator, CORE 4.3 [ADH08]. We use EMANE and CORE together to run realistic emulations of mobile performance on a Linux server using the same codebase that runs

on Android devices with real testbed experiments. EMANE models the network topology, and CORE is responsible for the mobility model and moving nodes around in accordance with scenario requirements. We use CORE for compartmentalization so that multiple virtual ICEMAN nodes can exist on the same physical machine; through the use of resource and network isolation.

For Android testing, we ran tests on Nexus S smartphones running Gingerbread. In order to emulate the resource constrained nature of an Android device, we use the cpulimit [cpu] tool to limit the CPU time allocated to ICEMAN processes.

To capture the energy requirement, we use a normalized energy utility measurement. We define this as measuring the total number of files delivered and the total CPU time needed. We then normalize both file delivery and CPU overhead to fragmentation (no coding). Finally, we divide the normalized number of files delivered by the normalized CPU overhead. It has been shown previously that coding induces noticeable computation overhead on mobile devices [SL10, SL09]. Thus, we capture the CPU overhead as an distinguishing energy indicator when comparing coding versus no coding.

4.4.4 Parameters

Our test framework runs publisher and subscriber applications on each ICEMAN node that communicates with the ICEMAN process on each virtual node according to the requirements set by the scenario. Each publisher and subscriber application keeps logs of the data objects that are sent and received. These logs are then analyzed, along with emulation output from EMANE and CORE, to study the bandwidth, latency, and delivery rates from the experiment.

We configured our system to use Cache Coding for files larger than 32KB. The block size for Cache Coding operations was also set to 32KB. The window parameter W for the loss estimate calculation was set to 30 seconds. Data Objects were disseminated using UDP broadcast. The UDP broadcast mode precludes MAC layer ACKs so there was no loss detection and retransmission. Through out our experiments, we use *ieee802.11abg* link

	Files Delivered	Normalized Energy Utility
Frag	158	1.0
CACC(0.2)	513	1.08
CACC(0.4)	489	0.92
CACC(0.6)	363	0.87
CACC(0.8)	382	0.71
NetCode	396	0.71

Table 4.2: Search Patrol Scenario: CACC is able to utilize both NetCode and Frag for improved delivery rates and reduced power consumption.

provided by EMANE with an omnidirectional antenna gain of -5 dbi and a system noise factor of 4 db in freespace loss model.

4.4.5 Results

To measure the effectiveness of dissemination, we measure both the delivery rate, in terms of the number of data objects delivered, as well as the normalized energy utility as defined in Section 4.4.3.

We compare across three different schemes, low overhead fragmentation (Frag); cache coding without any context aware switching (NetCode), and network coding with context aware switching (CACC). We use the notation CACC(l_{th}) to indicate the usage of a CACC with the given value of l_{th}.

We experiment across various values of l_{th}, using values of 20%, 40%, 60%, and 80%. Conceptually, Frag is equivalent to using CACC with an l_{th} value greater than 100%; as l_{th} can never exceed 1, network coding will never be enabled. NetCode is equivalent to using an l_{th} value of 0% as network coding will then immediately be enabled.

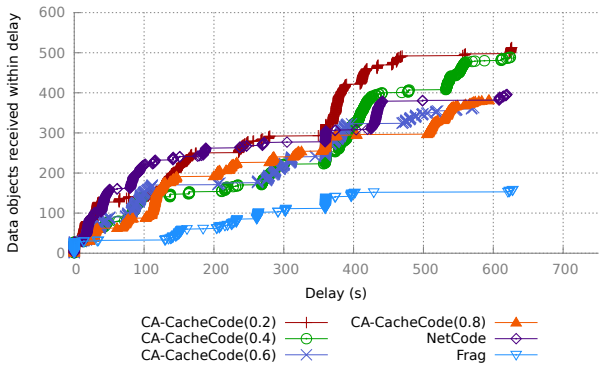

Figure 4.5: Search Patrol Scenario: CACC delivers more data objects than NetCode using less power.

4.4.5.1 Search Patrol Scenario

In this scenario, the subsquads move as groups and the channel loss rates between intra-squad links are low, while the average loss rates of inter-squad links are relatively high. Additionally, the rendezvous time is short so that data objects are delivered in parts. Therefore, NetCode has a clear advantage for transmissions between nodes from different squads. However, since the number of nodes in the mobile group is small and the number of data objects to be exchanged within the mobile group is few, the interference caused by channel contention is relatively low. Therefore, NetCode is not required for intra-subsquad communications and would just introduce unnecessary overhead. Frag gets the fewest data objects through, since it gets hit with the coupon collector problem.

CACC combines the best of both worlds, it is able to utilize low-overhead Fragmentation for intra-subsquad transmissions and use NetCode when needed for inter-subsquad transmissions. From the results we can see that CACC uses less power than NetCode; and delivers many more data objects in the case of CACC(0.2) and CACC(0.4).

Figure 4.5 and Table 4.2 illustrate this point clearly. Note that the delivery rate is a function of the mobility, and jumps when the subsquads interact. Initially, NetCode performs the best and has the lowest latency, at the expense of power consumption; NetCode always uses expensive network coding operations while CACC tries to save power and use low-

	Files Delivered	Normalized Energy Utility
Frag	114	1.0
CACC(0.2)	256	1.29
CACC(0.4)	202	1.21
CACC(0.6)	287	1.45
CACC(0.8)	262	1.31
NetCode	248	1.25

Table 4.3: Data Mule Scenario: CACC delivers more data than NetCode while using similar amounts of power.

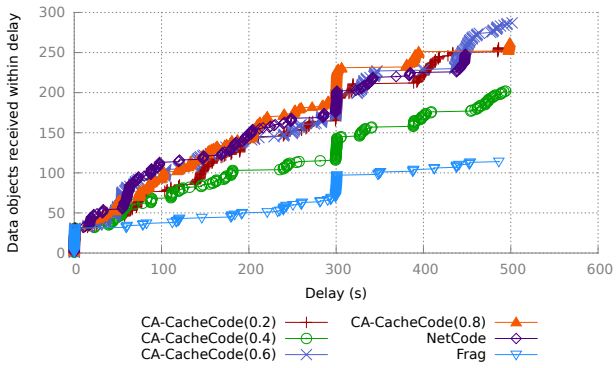

Figure 4.6: Data Mule Scenario: CACC delivers more data than NetCode and Frag.

overhead Fragmentation. Over time, when enough blocks get through and the squads meet again, CACC is able to reconstruct data objects and improve delivery rates.

4.4.5.2 Data Mule Scenario

The Data Mule scenario illustrates the benefits of using CACC for delivery rates versus simply using NetCode or Frag. Intra-squad sharing is reliable, but inter-squad sharing depends on how efficiently blocks can be transmitted by the data mules to the other squad. In this scenario, the delivery rates are a function of the mobility. We expect that all intra-squad subscriptions will complete successfully, while the delivery rate for inter-subsquad

subscriptions will depend on the mules. If the mules have a shorter stay, the spikes in data object delivery will be closer together. However, the spikes would be shorter as the mules would not be able to receive/send enough data objects when they are in contact with the squads. A stay duration of 60 seconds, as in our scenario, is a reasonable middle ground.

In Table 4.3, we can see CACC outperforms both NetCode and Frag. CACC(0.6) delivers more data objects when compared to plain NetCode. Other cases (except CACC(0.4)) perform like NetCode in terms of delivery rate. CACC is able to utilize the network context to learn that blocks should be sent to the mules. It thus pays the cost of network coding for those nodes, but avoids expensive network coding operations when sharing data over reliable intra-squad links. We note that the power consumption is similar to NetCode, but more data is delivered.

Figure 4.6 illustrates this point. Frag takes much longer to deliver data objects intra-squad initially, due to the coupon collector problem. Very few data objects are delivered through the mules, again due to the same problem. NetCode, since it does not suffer from the coupon collector problem, is able to deliver more data objects, and deliver them quickly. Initially, the performance of CACC is somewhere in between NetCode and Frag. This is due to the fact that at startup a lot of data is being exchanged, and CACC needs to detect the network state. CACC detects the initial lossy channel and switches to network coding. Later on, past the 100 second mark, when enough blocks and fragments come in through the mules, intra-squad sharing can proceed through low-overhead fragmentation and thus delivery rates are improved.

4.4.5.3 Search Patrol Scenario on Android

Results for the Search Patrol scenario on Android are presented in Table 4.4 and Figure 4.7. We can see that NetCode initially takes longer to deliver data objects, due to the time required to encode and decode blocks on a slower CPU. In this time, CACC, using a mix of fragmentation and network coding, is able to deliver data objects with lower latency. Note that Frag is unable to deliver many data objects as it is hit hard by the coupon collector

	Files Delivered	Normalized Energy Utility
Frag	60	1.0
CACC(0.2)	330	7.93
CACC(0.4)	353	7.06
CACC(0.6)	234	3.58
CACC(0.8)	178	2.73
NetCode	252	5.81

Table 4.4: Android Search Patrol Scenario: CACC(0.2) delivers the most data while using the least power.

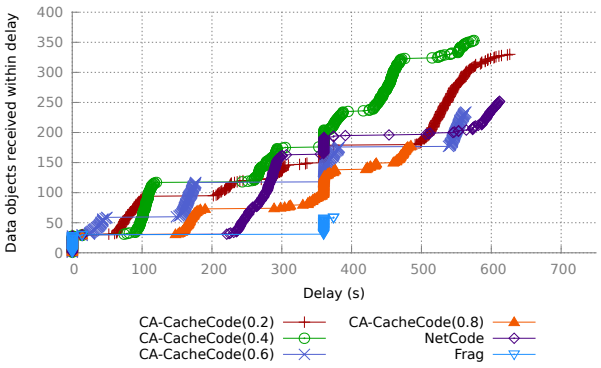

Figure 4.7: Android Search Patrol Scenario: CACC delivers more data than NetCode and Frag.

problem.

In terms of power usage, we see that CACC(0.2) uses the least power, while other CacheCode settings use more power. In addition, Frag uses more power per data object than NetCode in this scenario, due to the fact that delivery rates are low. This illustrates that the choice to use low-overhead transmissions must also consider the fact that data delivery may be adversely affected. We can also observe that CACC is adaptive and able to both save power and increase delivery rates.

CHAPTER 5

Preliminaries II: Decentralized Content Confidentiality and Authorization

Today people carry various consumer electronic devices such as digital cameras, smartphones, and laptops. These Internet-enabled smart devices are both the *consumers* of published content and the *producers* of user generated content. Content creation has become very easy, and anyone can post content using Web 2.0 tools, e.g., YouTube, Flickr, Twitter, etc. As a result, personal content is exploding—content is shared and scattered in multiple places ranging from personal devices to cloud storage. A recent report estimated that by 2015, terabytes of data will be in a person's pocket, and petabytes of data in a person's home [Sto07]. Under this circumstance, it is very important to have a system that seamlessly enables networking of personal content such that users can manage personal content scattered over multiple devices (including cloud storage) and selectively share content with friends.

The first step toward this goal is to introduce single persistent naming over personal content scattered over multiple devices. Since the current generation of personal devices keeps individual namespace in each device, content is tied to a device. As the amount of content increases, users tend to lose track of what files are where, and content management becomes difficult. A unified view with persistent naming will allow users to make location independent (or content centric) queries where there is no need to specify which device has the requested content. For instance, Alice can access her favorite songs via name: "*Alice/Music/My Favorites*." Similarly, she can share the collection to Bob, by simply telling the name.

This content centric approach is widely acknowledged as a key feature of the future

Internet as in content centric networking (CCN) that replaces conventional host-to-host conversations with name-based communications and provides secure binding between name and data to thwart security attacks [JST09b]. Name-based routing of CCN enables content retrieval over a fully distributed network without specifying where the content is located as any nodes that have the requested data locally answer the request [JST09b]. While CCN was originally designed for large-scale content dissemination (or even replacing the existing IP network), its key principles (i.e., name based routing and secure binding) are also applicable to realize secure personal content networking. However, due to scalability and performance reasons CCN transfers data and also caches data on *untrusted nodes* that forward/store data properly and yet do not necessarily keep data confidential. Additionally, CCN lacks an essential component of personal content networking, namely secure content management such as content updates and access control over untrusted nodes.

While secure content management is an active area of research in the field of distributed file systems, existing work mostly focused on host-centric, trusted file systems—a trusted file server handles user authentication and access control authorization and then provides data confidentiality by securing the communication channel (e.g., SFS [MKK99]). When dealing with untrusted storage, the files must be encrypted to assure data confidentiality as in the Cryptographic File System (CFS) [Bla93] and Plutus [KRS03]. However, such a cryptographic storage system cannot generally provide fine-grained, expressive access control; e.g., in Plutus, a file can be only encrypted using a single key. If a user wants to share the file with more than two groups, it is not clear which key should be used for encryption. A simple solution is to use a common key for file encryption and to encrypt this key using each user's public key as in SiRiUS [GSM03]. This approach is still limited in that the metadata size linearly scales with the number of users, and supporting more expressive access control is difficult as it only uses a single file encryption key. Moreover, existing cryptographic systems do not support *secure binding* between name and data, and as a result, the channel must be secured to prevent a man-in-the-middle-attack.

5.1 Content Centric Networking Review

PCN's underlying content retrieval is based on Content Centric Networking (CCN) [ccn12, JST09c]. CCN is a specific design and implementation of Information Centric Networking architecture, and was one of the precursors of today's information centric networking architectures.

In this section, we review the core components of CCN, namely (1) naming, (2) content reachability, (3) content retrieval, and (4) content centric security.

Naming: CCN names a file with a user friendly, structured, location-independent name, and each file is divided into segments. Consider: "*/ucla.edu/music/abc.mp*3/*v*3/*s*0." Here, "*ucla.edu*" is a globally routable name (called prefix), "*/music/abc.mp3*" is a local name in "*ucla.edu*," "*v3*" is a version name (represented using a timestamp), and "*s0*" denotes the segment number.

Content reachability: As a future Internet architecture, CCN operates on top of an underlying physical network topology that could include end-user devices. Like BGP of the current Internet, a prefix owner announces the prefix to the entire network. For instance, Alice from "*ucla.edu*" announces her files say "*/ucla.edu/Alice/*" from her laptop. Each node in the network broadcasts the incoming prefix to its neighboring nodes. Whenever a node receives the prefix, it sets up a backward pointer to the sender in its Forwarding Information Base (FIB) for that prefix. As a result, any node can reach the node that announces the prefix by following the backward pointer in the FIB.

Content retrieval: Content retrieval is pull-based as in HTTP (i.e., get and response). A user sends a request (via an Interest packet), and any nodes that have the requested content in its local storage (or cache) can respond. For a given prefix, the Interest packet is forwarded along the reverse path toward each data source by following the backward pointer in the FIB. Whenever a node receives an Interest packet, the breadcrumb information (i.e., a backward pointer to the previous forwarder) is stored in the Pending Interest Table (PIT). The corresponding data packet will then be delivered by following the reverse path in the PIT.

If content is replicated in multiple nodes, the same prefix will be announced by these nodes. This allows that any nodes in the network reach these nodes. When forwarding an Interest packet, CCN uses the longest prefix matching algorithm; i.e., in the FIB, a node finds the longest prefix entry that has the largest number of leading letters matching those of the content name in the interest packet. For instance, Alice's desktop has "*/Alice/my music/pepper/*," and Alice's laptop has "*/Alice/my music/*." When Bob accesses "*/Alice/my music/pepper/abc.mp3*," it matches the prefix entry of "*/Alice/my music/pepper/*" and the Interest packet will be delivered toward Alice's desktop.

Note that the CCN forwarding engine has a content cache (or content store), which is similar to the packet buffer of the IP counterpart. An incoming data packet is stored into the cache such that later requests of the same data packet can be served locally. In addition, a CCN node has a local repository that is a non-volatile content store in a disk. When a user publishes content, it is stored in the local repository, and the prefix of content is advertised to the network.

Content centric security: CCN supports the secure binding between name and content (called content-based security) where protection and trust travel with the content itself [JST09b]. To this end, CCN uses asymmetric cryptography: all content is authenticated with digital signatures, and private content is protected with encryption. Each data packet contains a signature by owner P, $Sign_P(N, C)$ which is over the name (N) and content (C). This content-based security is critical since content can be cached in untrusted intermediate nodes. For key management, CCN can use a traditional certificate-based public key infrastructure (PKI) or the arbitrary graphs used by the PGP Web of Trust.

5.2 Content Confidentiality and Role-Based Access Control

Distributed file systems: Research on distributed file systems for a mobile environment has been mainly directed to extending existing client-server based file systems to cope with node mobility and network disruption [RHR04, SGZ02, PSY04]. The common technique is to use optimistic file replication and eventual consistency. BlueFS [NF04] extends a

client/server based file system by focusing on power management to save energy of mobile devices. EnsemBlue [PF06] builds upon BlueFS to provide a consistent view of all files scattered over multiple devices with heterogeneous device capabilities. Ficus [RHR04] uses a peer-to-peer (P2P) model for optimistic replication where all replicas are equal and can propagate updates to all other replicas. Bayou [TTP95] is also based on a P2P model; it uses anti-entropy for consistently management and supports a database language for data retrieval.

There are several systems designed for a multi-device environment. Unmanaged Internet Architecture (UIA) provides zero-configuration connectivity among mobile devices through personal names [FSL06]. Unlike existing work, UIA assumes that each device has its own persistent namespace, and a user has to keep track of all the files scattered over multiple devices. Instead, Eyo [SLP09] offers a device transparency model in which users view and manage their entire data collection of all the devices by periodically flooding meta-data everywhere. PersonalRAID [SGZ02] supports optimistic replication at a volume level, and a mobile storage device is used to synchronize the volume in a delay tolerant fashion. Footloose [PSY04] supports application specific optimistic replication with eventual consistency (e.g., address book), and yet it uses a persistent, flat namespace (called ObjectID).

Wide area P2P storage systems: Wide area P2P storage can be classified based on the overlay structure; (1) a structured system (e.g., PAST, CFS, Ivy) forms a structured overlay network using a distributed hash table (DHT); (2) a structureless scheme (e.g., Gnutella and eDonkey) forms a structureless overlay network where the overlay links are arbitrarily established. Unlike unstructured P2P networks, DHTs provide a better performance for searching items over a large number of distributed nodes, and they have been widely adopted to implement wide area P2P storage. Most P2P storage systems assume wired Internet scenarios and support strong consistency, which is less suitable for personal content networking.

Decentralized access control: The following concepts are closely related, namely user authentication, access control authorization, and data confidentiality. Existing access control systems could be classified based on types of authentication methods. When AUTH_SYS

(UNIX's default) and Kerberos are used, systems mostly provide UNIX-style ACL (e.g., NFS, AFS, xFS). When public-key cryptography is used, systems typically support either UNIX-style ACL (e.g., SFS [MKK99]) or certificate authorization (e.g., DisCFS [MPI03]). These systems basically assume that *file servers are trustworthy*, but the network is not secure; thus, data confidentiality is guaranteed by securing the channel (e.g., SSL). If the servers are not trustworthy, we can either rely on some other semi-trusted servers as in Cobalt [VMF08] or use cryptographic encryption to preserve data confidentiality as in Cryptographic File System (CFS) [Bla93], Plutus [KRS03], and SiRiUS [GSM03]).

In the latter approach (called a cryptographic file system), authentication is typically done using public key cryptography where a user's public key is used as an ID, and digital certificate is used for authentication. CFS uses a single key for encryption (coarse grained, e.g., directory/volume) and is dependent on the underlying file system for write authorization [Bla93]. Later variants used a lockbox to protect the keys (with more fine-grained access control) and introduced several mechanisms for verifying the write operations without depending on the underlying file system [KRS03, GSM03]. In particular, SiRiUS [GSM03] permits that a file can be shared by multiple individuals or groups using a common file encryption key which is encrypted again using each user/group's public key.

Given that attribute-based encryption (ABE) is designed to provide a fine-grained, expressive access control, several existing work used ABE for *read-only* content sharing over untrusted storage [YRL08, BBS09, YWR10]. In particular, Yu et al. [YWR10] used key-policy ABE (KP-ABE) to provide privacy-aware content sharing over untrusted cloud storage and Proxy Re-Encryption (PRE) to delegate the task of re-encryption on the cloud servers. While PCN is considered as a cryptographic file system, unlike existing systems, PCN provides fine-grained, expressive access control using CP-ABE in a fully distributed environment with untrusted nodes and allows file owners to set up expressive *read and write* access policies based on attributes (e.g., college friends, family members). Further, none of the aforementioned systems do provide *secure binding* between name and data, and thus, the channel must be secured to prevent a man-in-the-middle-attack.

CHAPTER 6

Decentralized Content Confidentiality and Authorization

Securely sharing/managing personal content is considered to be a challenging task in a multi-device environment. In this dissertation, we design and implement a new platform called personal content networking (PCN). Our work is inspired by content centric networking (CCN) in that we aim at enabling seamless access of personal content without specifying its location. The unique challenge of PCN is to support secure file operations such as replication, updates, and access control over distributed untrusted nodes. The main contribution of this paper is the design and implementation of a secure content management mechanism that supports secure replication/updates and fine-grained content centric access control. Further, we demonstrate its feasibility with prototype implementation on the basis of CCNx.

We propose the personal content networking (PCN) platform that provides a secure content management mechanism over CCN, enabling secure replication/updates and fine-grained content centric access control. We extend CCN to build a basic framework for distributed content management with replication and updates. We then propose and implement a *secure content centric access control* mechanism using a recently proposed cryptography tool called attribute-based encryption (ABE) that permits secure sharing of content within a group over untrusted nodes [BSW07]. ABE supports fine-grained, expressive access policies called attribute based access control (ABAC). An owner can define a set of attributes (e.g., college friends, CS219 team, and family members) and issues a secret key for the assigned attributes to an individual. A file is encrypted based on the access policy over the attributes using an owner's public key. For a given encrypted file and access policy, any user can decrypt the file as long as he has the secret key with the attributes that satisfy the given policy.

While ABE was designed and has been used for selective *read-only* content sharing over untrusted storage [YRL08, YWR10], to the best of our knowledge, our work is the first attempt to build a fully distributed personal storage system that supports ABE-based fine-grained access control with *read-write* operations over untrusted devices and *secure-binding* between name and data. The main contribution is twofold. We design the PCN platform by significantly extending CCN to realize a secure content management mechanism that supports secure replication/updates and fine-grained content centric access control. Further, we build a PCN prototype by integrating the whole system using FUSE, a user level file system and demonstrate that a user can seamlessly access/manage content using PCN.

The rest of this section is organized as follows. We present the design goals of PCN. We then provide an overview of PCN's basic framework and secure content management methods. Next, we present the prototype implementation of PCN.

6.1 Secure Personal Content Networking

Recent advances of technology in consumer electronics have promoted a lifestyle where people live with convenience and ease by accessing any kind of information at their finger tips. These devices also allow people to generate/share a sheer amount of personal content such as photos, videos, and documents. However, personal content is now exploding, and personal content sharing/management is considered to be a challenging task particularly when users need to deal with personal content scattered over multiple devices. To mitigate this problem, we aim at enabling seamless access of personal content without specifying its location via information centric networking (ICN) over personal content. In this paper, we design a platform called personal content networking (PCN) that uses a single persistent, hierarchical naming space for personal content, allows users to securely initialize their devices and establish trust with other users, enables efficient content management over multiple devices (e.g., updates, removal, replication), and supports content centric access control via attribute-based encryption (ABE) for selective sharing where access control is not tied into hosts and yet fine-grained attribute based access control is permitted. We demonstrate its

feasibility with prototype implementation on the basis of CCNx.

Personal content is exploding. The storage demand appears to be infinite. A recent report estimated that by 2015, terabytes of data will be in a person's pocket, and petabytes of data in a person's home [Sto07]. Under this circumstance, it is very important to have a system that seamlessly enables networking of personal content such that users can efficiently manage personal content scattered over multiple devices and selectively share content with friends.

Our work is motivated by the recent proposal of future Internet, namely information centric networks (ICN), a new Internet architecture that replaces conventional host-to-host conversations with named data oriented communications based on an URL-like persistent namespace [JST09b]. The key features of ICN are that (1) it supports single persistent, hierarchical naming space and provides secure binding between name and data which can effectively thwart most security attacks; (2) it supports name based content access in which users can request content without specifying where the content is located, and any nodes that have the requested data can answer the request as nodes can cache data locally.

While building upon CCNx, PCN addresses the unique issues that are essential to personal content networking scenarios. First, the system should provide an intuitive trust management mechanism for trustworthy content sharing among users (e.g., introducing other users into a personal network). Second, the system should support distributed content management over multiple devices such as replica management and content updates/consistency management. Third, the system must enable *content centric access control* for selective content sharing among friends. Simple public key based end-to-end encryption is not sufficient for access control, because this approach complicates content naming and nullifies the benefit of content caching in the intermediate nodes. Finally, the system should provide efficient content routing using an overlay network that is on the basis of social relationship.

6.2 PCN System Design

While building upon CCN, our system design aims to support the following features that are essential to realize personal content networking: (1) PCN uses a single hierarchical, persistent naming scheme of $N = P : L$ (where P is the name of a user, and L is the label representing the location of data in the hierarchy) and manages the trust using SPKI/SDSI [CEE01]; (2) for secure identity introduction in mobile environments, PCN uses a secure introduction protocol that can effectively thwart the man-in-the-middle-attack; (3) given that one of the key functions of personal content networking is to share content among friends, PCN builds an overlay network based on social relationship; (4) PCN supports attribute based access control (ABAC) with attribute based encryption (ABE) to enable selective content sharing among users [BSW07]; and (5) PCN users can effectively manage content in their personal devices and support content updates and automatic synchronization over CCN.

6.3 Motivating Scenario and Design Goals

We use the following example to motivate the needs of personal content networking. Bob has a number of smart devices: Internet TV, desktop, iPhone, EyeFi-enabled digital camera, Internet fridge, and network attached storage (NAS). He has other devices at school (e.g., desktop, laptop) and also maintains a few cloud servers (e.g., Amazon EC2). His personal content is currently scattered over these places, and Bob had a hard time tracking all these files. For instance, his friend Alice asks him to send the lecture material of the course that they took last year. He only remembers that it is located at some document directory, but he forgot where he put it. He searches through machines one by one: laptop, servers, desktop, and NAS, and finds that it is stored in his NAS at home. After locating the lecture material, Bob feels a bit frustrated because the size is over 1GB (even after compression) and cannot send it via email. He calls Alice saying he will give the files to Alice using his USB stick.

This example clearly illustrates the needs of seamless networking of personal content such that users can manage personal content scattered over multiple devices and selectively share

content with friends. The design goals of PCN can be summarized as follows:

- *Single persistent, hierarchical namespace*: Single persistent, hierarchical naming of personal content will guide users to have a unified view of their personal content scattered over multiple devices. Hierarchical namespace is critical because people typically prefer managing their personal content hierarchically [JPG05, HS09]—it is reported that hierarchical naming significantly lessens the cognitive overhead of locating files [Lan88]. We expect that single namespace with location-independent content access could greatly reduce the cognitive burden in a multi-device environment.

- *Social networking*: Users often want to share content with their friends. PCN should leverage the social networking aspect by establishing/managing trust relationship among friends and by building an overlay network for content sharing.

- *Fine-grained access control*: PCN must provide fine-grained, expressive access control to enable secure content management over distributed untrusted servers that store/transfer data properly and yet do not necessarily keep data confidential.

- *Disrupted operations*: Since devices can go offline at any time, users should be able to replicate files, and files must be automatically synchronized whenever they become online again as in existing distributed file systems [SKK90, RHR04, PSY04].

- *Security guarantee*: Since PCN deals with distributed untrusted devices connected over the Internet, it must be resilient to well-known security attacks such as a denial of service attack, and a false data injection attack.

In the above scenario, PCN will allow Bob to easily locate the material (say, "*/Bob/My Doc/CS101*") and to pass this link to his friend Alice. Bob does not need to examine each device, but simply need to browse his namespace at any machine. Alice can download the content using that link. In the following, we review CCN, a building block of PCN (Section 3) and show how PCN's basic framework can be built over CCN (Section 4).

6.4 Basic PCN Framework with CCNx

We build a basic framework of PCN by extending CCNx. PCN is based on a web of trust with SPKI/SDSI [CEE01] and a UIA-style secure introduction process for secure key exchanges [FSL06]. PCN uses a single hierarchical, persistent naming scheme of $P : L$ where P is the identifier of a user, and L is the label representing the location of data in the hierarchy. PCN builds an overlay network based on social relationship as one of the key functions of personal content networking is to share content among friends.

6.4.1 Naming

The current generation of personal devices use rigid and weak naming of the form "hostname:path." The key problem is that content is tied to a host, making personal content management non-trivial, particularly when a user interacts with a number of devices (e.g., laptop, desktop, smartphone) including cloud-based storage services (e.g., Dropbox). A user has to track what files are located in each of these devices/services and to decide how to migrate/replicate/update content.

In PCN, we define a single persistent, hierarchical namespace for each person. It is known that global namespaces are politically and technically difficult to implement (e.g., X.509, PEM). Thus, we use local, decentralized namespaces of SPKI/SDSI [CEE01]. Each person has a public-private key pair to verify the identity of the sender (sign/verify) and to ensure privacy (encrypt/decrypt). Relationship among users in personal content networking is considered to be flat, and it is sufficient to use the public key as identity. Nonetheless there are cases where hierarchical naming is useful; e.g., a group of users has a set of sub-groups. In SPKI/SDSI, a user can define a local namespace as a sequence of length two consisting of her key K followed by a single identifier (that is distinct within the local namespace). For instance, Alice with key K_a makes her own name as "K_a Alice." A study group with key K_g can name its sub-groups as "K_g sub1" and "K_g sub2." If a sub-group has multiple smaller groups inside, that group can name those groups similarly; e.g., sub1's two internal groups (ssg1 and ssg2) can be named as "K_g sub1 ssg1" and "K_g sub1 ssg2." Note that a local

name is globally unique because the name contains a public key of the user. Moreover, each user can make signed statements of these *local names*, which allows anyone to certify a key via a web of trust [CEE01].

Given this, a name of data is of the form $N = P : L$ where P is the name of a user (or its cryptographic hash), and L is the label representing the location of data in the hierarchy; e.g., Alice's music can be denoted as "*/K_a Alice/music/*." PCN's naming can be used in CCN with minimal modification as CCN uses hierarchical naming (e.g., "*/ucla.edu/test.txt*"). As in CCN, each device will advertise the content reachability information by broadcasting the name prefix of the content that is stored in the device. For instance, Bob's laptop will advertise "*K_b Bob/my doc/*," and his iPad will advertise "*/K_b Bob/my music/beatles/*." For the sake of brevity, hereafter we will use *abbreviated names* without a public key, e.g., "*/Bob/my music/*."

6.4.2 Trust management

Each PCN user has a private-public key pair which is used to define a user's name. When a new device is purchased, this information must be securely installed to initialize a PCN service. Moreover, for content sharing with others, a user must establish trust relationship by securely exchanging the public keys (e.g., how does Bob make sure that a key belongs to Alice?)—nonetheless trust relationship does not necessarily guarantee data confidentiality. For both problems (i.e., device initialization and trust establishment) secure key distribution is the key issue. Users can use USB sticks or can use local/wide area networks for key exchanges. The latter is less secure than the former, because it is vulnerable to the man-in-the-middle-attack—an attacker can eavesdrop the channel and make independent connections with the victims and relays messages between them, making them believe that they are talking directly to each other over a private connection, when in fact the entire conversation is controlled by the attacker.

A simple method of avoiding the attack is to use another secure channel. Alice can show (or read) her public key to Bob (e.g., via physical presence, SMS, email, voice communica-

tions), and ask Bob to check whether his key matches with the key that she received. Given that checking a large number is laborious and erroneous, Ellison et al. [DE02] proposed an approach where the keys are represented in color bars so that users can easily verify the key. In UIA, multiple choice questions are used to further reduce the burden of users [FSL06]. Alice sends her multiple choice question to Bob, and Bob sends his question to Alice. After solving each other's question, they exchange the hashed values of their answers (and both keys), hoping that the attacker cannot solve the questions and thus fails to control the conversation.

However, this approach is vulnerable to the man-in-the-middle-attack since a malicious user can perform a dictionary attack. The attacker knows both keys and the multiple choice questions. He can easily find the answer by computing a hashed value for each answer choice and comparing this value with the received answer. Like UIA, we use multiple choice questions, and yet we solve the man-in-the-middle-attack by using Ellison's approach [Ell96] that is based on Pedersen's interlock protocol [Ped91]. Given that the secret (answer of a multiple choice question) is a, one chooses a random value u and then computes $x = g^a h^u \mod p$ where p is prime, g and h are generators of the group $\mod p$. Alice and Bob generate their own numbers and exchange these values: i.e., Alice: $x_A = g^{a_A} h^{u_A} \mod p$, and Bob: $x_B = g^{a_B} h^{u_B} \mod p$. The attacker cannot infer value u and must use a random value to finish the transaction, thus effectively thwarting the dictionary attack.

6.4.3 CCN overlay construction

Trust management among friends can be used to form a social network. We use this social relationship to create an overlay network for content centric networking (CCN). Whenever identity introduction happens, corresponding personal devices also exchange IP addresses and join the overlay network. Each device maintains a peer list that contains IP addresses and port numbers of other devices. For a given user, the list includes a user's own devices and direct friends' devices. For instance, Alice's laptop has a list of all her devices and a list of Bob's devices. These devices periodically check the availability of neighboring devices to

maintain the overlay network.

A device may be behind a NAT, and it cannot actively participate in the overlay network. In this case, the device can be connected through a relay node that is not behind a NAT and is stable enough (say 90% of time the device is up and running). NAT can potentially reduce the number of peering devices, thus lowering the connectivity among devices. We can increase the connectivity by allowing devices to exchange IP addresses of k-hop friends' devices. For instance, when $k = 2$, Alice can connect to Bob and also to Bob's friends. Further, computing resources in the cloud systems could be utilized to increase the connectivity; e.g., a personal account in Dropbox can serve as a node for personal content networking.

6.5 Secure Content Management

We first illustrate file replication and synchronization and justify the need of prefix protection for replication. Then, we present the details about content-centric access control, followed by the illustration of remote content management and the discussion of key revocation.

6.5.1 Synchronization

PCN provides "eventual consistency" in which all replicas eventually converge to the same version given enough messages exchanged among participating devices (i.e., a file with the freshest timestamp) [PSY04, RHR04]. Eventual consistency is one of the widely used consistency models in disruption-prone mobile environments.

Whenever a replicated file is updated, a new version is created (with a new timestamp). Each replica has an associated version vector that tracks its update history [RHR04, PGH98]. To alert this event, the node that makes the update will re-announce the corresponding prefix with a modification mark, which is a special type of the prefix announcement used for update notification. The prefix announcement also contains detailed information of the updated file, including its name, the current version, and the version vector. For synchronization, the local client compares the version vector of the local replica with that of the updated file. If the

updated file is strictly newer than the local one, its version vector will dominate; the local client fetches the updated file and replaces the local file in the repository. If two version vectors are not equal and neither one dominates, an update/update conflict has happened. If automatic merging fails, PCN notifies the user that a conflict has detected. The user will be presented with a revision history including authors, dates, and versioned content. It is up to the user to resolve the conflicts and mark the content as merged. Note that whenever intermediate nodes hear the modification announcements, their local caches are examined to find whether there are matching files, and the matched files (or data packets) in the caches are invalidated.

Synchronization of a replicated directory needs special care. Although the only modification operations are adding new entries or deleting/changing existing entries, a directory replica can be modified from multiple places, which causes several well-known synchronization issues such as insert/delete ambiguity, remove/update conflicts, and name conflicts [BP98, PGH98]. In PCN, we basically adopt the existing solutions used in the Ficus file system [RHR04, PGH98].

When a node re-joins the PCN network after disruption, it first checks its neighbors to find any missing prefix announcements. As we will see, a PCN user has a reserved namespace for devices, namely "*/dev*," and devices are accessible through this name; e.g., Alice's iPod is named as "*/Alice/dev/iPod*." For prefix announcement synchronization, each device stores the received prefix announcements in a designated place; e.g., Alice's iPod has "*/Alice/dev/iPod/received_prefix*." This allows the node to search for the updates of the files located in its local storage. If the node finds a prefix with a modification mark, it performs file synchronization as illustrated earlier.

Note that in PCN, nodes fetch the updated file for synchronization. If the size of a file is large and only small part of the file is updated, the bandwidth wastage will be severe. A simple solution to this problem is that a node generates/publishes a delta file (e.g., using diff) and includes the link of the delta file in the prefix announcement.

6.5.2 Prefix protection

So far we assume that any node can replicate the content and announce the named prefix. After replicating the content, however, malicious users can launch an attack by inundating the network with fake update announcements. PCN nodes could waste considerable resources to handle such updates. Given that CCN does not deal with updates, this problem is unique to PCN.

To solve this problem, we propose to restrict that a prefix announcement is signed by the prefix owner. This mechanism is a reasonable approach in that people typically want to have a full control of their namespace and the locations of files in a multi-device environment. A similar technique is used in BGP security where each prefix is signed in order to prevent prefix hijacking where an attacker has a partial or full control of the named prefix. While this problem is less serious in CCN because requests (interests) will be routed to all replicas and a denial of services cannot be achieved, it is still possible that malicious users can launch update flooding attacks.

In PCN, a prefix announcement is augmented to include signature that certifies the prefix ownership. Further, we implement the ownership delegation such that an owner certifies that a named user is allowed to announce the named prefix by issuing a prefix certificate. For instance, Alice can issue a certificate to Bob that Bob can announce the prefix "*/Alice/my doc/proj/*." Intermediate nodes can verify that the certificate is valid, and the prefix announcement is actually originating from Bob (similar to data packet validation). As long as Bob is permitted, Bob can update the local replica of Alice's file, and the update will be automatically propagated. Note that it is possible for the attackers to perform the replay attack where a CCN speaker replays a prefix which it has previously heard. This problem can be mitigated by adding an expiration timer as in S-BGP [KLS00].

6.5.3 Content centric access control

Access control in personal devices is mostly host centric. In identity based access control (IBAC) [SS94], a user first logs into the system (authentication) and then accesses files based

on the permissions in the access matrix (authorization). SPKI/SDSI supports role based access control (RBAC) where permissions in the access matrix are tied to roles [CEE01]. SPKI/SDSI is also host centric as it basically assumes trusted servers and insecure channels; i.e., an individual must first set up a secure channel (using SSL) to prevent man-in-the-middle-attack, and the server checks whether a requester's key is on the role-based ACL [BCD02]. In PCN, nodes store/transfer data properly and yet do not keep data confidential, and thus, host centric access control is not suitable.

For personal content networking, we need *content centric access control*; i.e., access control of content is self-contained and is not tied to a host. A simple solution is to encrypt the content using the receiver's public key and to define a specific name of the encrypted data meaningful to the receiver. The encrypted content can be placed in the untrusted servers as others cannot decrypt the content. If a file needs to be shared with multiple people, a common key is used for file encryption, and this key is encrypted using each user's public key [GSM03]. The encrypted keys are then included in the meta-data of an encrypted file, and the entire content (meta-data + encrypted file) is published. However, this approach has several limitations. Supporting expressive access control is difficult as it only uses a single file encryption key. If multiple keys are used, the size of metadata linearly increases with the number of users/groups. More importantly, once the content is published, the owner cannot give access to the other users—the owner must republish the original file by including additional users.

In PCN, we solve this problem by using ciphertext-policy attribute based encryption (CP-ABE) that permits secure sharing of content within a group across multiple untrusted servers [BSW07]. ABE is the key enabler for attribute based access control (ABAC) where access decision is based on the attributes associated with individuals. Each user first generates an ABE public key (PK) and an ABE master key (MK). A user can define a set of attributes (e.g., college friends, CS219 team, family members) and an access policy using Boolean formula on attributes. This allows a user to perform fine-grained, expressive access control. The user assigns a set of attributes to each user and then issues a secret key corresponding to the attribute set; i.e., *Secret Key Generation(MK, S)* where MK is the master

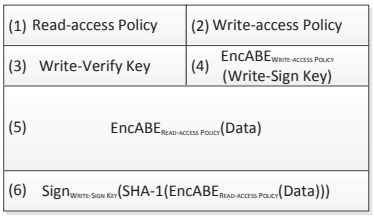

Figure 6.1: PCN's file data structure for secure content centric access control

key, and S is a set of attributes assigned to a user. A file can be encrypted using the public key, and the access policy; i.e., *Encrypt(PK, M, A)* where PK is the public key, M is a message, and A is an access policy. Here, any user can encrypt the file using the public key. Further, any user who has a secret key with attributes that satisfy the policy can decrypt the content; i.e., *Decrypt(PK,CT, SK)* where PK is the public key, CT is the cipher-text, and SK is the secret key. In ABE, the metadata only contains the access policy information whose size scales with the number of attributes (not with the number of users). Even though the content is published, the owner can still issue attribute keys to the other users without republishing the content.

Suppose Alice would like to selectively share her music collection, "*/Alice/my music/rock.*" This content is digitally signed using Alice's CCN publisher key and is published under that namespace. For access control, "*Red hot chili peppers*" is encrypted with the attribute "*college friends.*" "*Incubus*" is encrypted with "*college friends*" and "*CS219 team*" attributes because Alice discussed Incubus with her teammates and wants to share the songs with them only. Bob is Alice's college friend, and Alice issues a secret key for the attribute "*college friends*"; Cathy is Alice's CS219 team mate, and Alice issues secret keys for the attributes of "*college friends*" and "*CS219 team.*" Bob can decrypt "*Red Hot Chili Peppers,*" while Cathy can decrypt both "*Red Hot Chili Peppers*" and "*Incubus.*"

In PCN, an owner of a file can set access permissions of *read* and *write* using separate access policies (used in ABE). The resulting access modes in PCN are *read-only* and *read-write*; write-only mode is not suitable for personal content networking. Access modes can be also used to directory files in order to limit access of directory listing. As shown in Figure 6.1, PCN payload contains the following fields: (1) read-access policy, (2) write-

access policy, (3) write-verify key (public key), (4) write-sign key encrypted using ABE with write-access policy, (5) actual data encrypted using ABE with read-access policy, and (6) write signature (optional). For write access control, a file owner issues a private-public key pair that is located at fields (3) and (4). The write-sign key is only accessible to those who have write permission as we encrypt the write-sign key using ABE with the write-access policy. Whenever a file is updated, the file is encrypted using ABE with the read-access policy. This legitimate modifier then reads the write-sign key through which he generates the signature of the updated content, which is finally placed in the field (6). This update event is then notified to all the nodes that replicate the content via prefix announcement with a modification mark. The replica nodes will then fetch the updated content and verify whether it is modified by the legitimate users who satisfy the write-access policy. Note that in our prototype implementation, we use symmetric encryption to reduce the overhead of encrypting/decrypting the content; i.e., the content is encrypted using AES, and ABE is used to encrypt AES key.

So far we assume that each file has its own access policy, but setting access policy for each file is a laborious task. To mitigate this problem, PCN allows users to maintain a set of access policies in their own namespaces. For instance, Alice has two access policies that she set up for content sharing; (1) *college friends*: "*/Alice/Policy/p1*" and (2) *college friends* and *CS219 team*: "*/Alice/Policy/p2*." Associated write-sign/verify keys can be stored as well: "*/Alice/Policy/p1.key*" and "*/Alice/Policy/p2.key*." Then, these links are simply embedded into the aforementioned fields (1) to (4). As in the original CCN, when a PCN user downloads a file for the first time, associated access control files need to be fetched to access the file. Note that when a file owner updates a read/write-access policy, PCN generates a new write-sign key, and an updated file version will then be propagated through the network. Replica nodes should immediately fetch the up-to-date policy information. In PCN, nodes with up-to-date keys will simply reject any updates that are signed with old keys due to security reasons (which could happen while synchronization is in progress).

6.5.4 Key revocation

PCN mainly uses the following keys, namely a personal public-private key pair, group public-private key pairs, and ABE keys. Secure key distribution can be assured as PCN uses the secure identity exchange mechanism and relies on SPKI/SDSI style web of trust. Any intermediate nodes will be able to correctly acquire public keys which will then be used to validate the secure binding between name and data, which is one of the core concepts of CCN. While we can leverage CCN for secure key distribution, we should be able to properly handle key revocation scenarios: a public-private key pair and an ABE attribute secret key.

If a user's public-private key pair is compromised, the existing key pair can be revoked through the prefix announcement with a revocation mark that is similar to a suicide note in PGP [Riv98]. Recall that in CCN, we add two additional prefix types, namely modification (for update notification) and revocation (for revocation notification). The user will then generate a new key pair and distribute the public key via the secure identity introduction process which guarantees that the attacker cannot impersonate the victim. Note that the same procedure can be used to handle the case where a group's public-private key pair is compromised.

If an ABE attribute secret key is compromised or the owner wants to revoke a specific attribute, the owner must revoke the master key and public key because CP-ABE does not provide a mechanism for revoking an individual attribute. While CP-ABE has a single attribute revocation by adding a timer attribute for each attribute, this approach is less practical because it makes the overall system quite complicated: (1) the owner must periodically issue keys, and all the files must be re-encrypted with new attribute sets, and (2) a tamper-proof clock is required to ensure the security guarantee.

Whenever the master key and public key are revoked, the owner must re-encrypt all the files. However, this process is very expensive. To reduce the overhead, we employ a lazy revocation scheme proposed by Kallahalla et al. [KRS03]. Unlike the compromise of a public-private key pair, that of an ABE attribute secret key is less serious, as long as the revoked user (or the attacker) has only a read-only access right—the revoked user cannot

remove or update the files. In this scenario, we can generally assume that the revoked user has read and copied all the files, and it is still acceptable for the user to read unmodified or cached files. Yet, the lazy revocation ensures that the revoked users are not able to read *updated files*; i.e., updated files will be re-encrypted with the new ABE public key.

Note that it is also possible for users to immediately revoke all the keys and re-encrypt all the files. In this case, the user must undergo a series of steps: (1) re-generating a new ABE key set, (2) invalidating all the cached files via prefix announcement, (3) removing the replicas over multiple devices, and (4) re-encrypting all the files and re-distributing replicas.

6.5.5 Naming

The current generation of personal devices use rigid and weak naming of the form "host-name:path." The key problem is that content is tied to a host, making personal content management non-trivial, particularly when a user interacts with a number of devices (e.g., laptop, desktop, smartphone, ipad) and storage services (e.g., dropbox). A user has to track what files are in each of these devices/services and to decide how to migrate/replicate/update content.

Relationship among users in personal content networking is considered to be flat, and it is sufficient to use the public key as identity. Nonetheless there are cases where hierarchical naming is useful; e.g., a group of users has a set of sub-groups. In SPKI/SDSI, a user can define a local namespace as a sequence of length two consisting of her key K followed by a single identifier (that is distinct within the local namespace). For instance, Alice with key K_a makes her own name as "K_a Alice." A study group with key K_g can name its sub-groups as "K_g sub1" and "K_g sub2." If a sub-group has multiple smaller groups inside, that group can name those groups similarly; e.g., sub1's two internal groups (ssg1 and ssg2) can be named as "K_g sub1 ssg1" and "K_g sub1 ssg2." Note that a local name is globally unique because the name contains a public key of the user. Moreover, each user can make signed statements of these local names, which allows anyone to certify a key via a web of trust (see the following section).

6.5.6 Device Initialization and Trust Management

As shown above, each PCN user has a private-public key pair which defines the user's name. When a new device is purchased, this information must be securely installed to initialize a PCN service. Moreover, for content sharing with others, a user must establish trust relationship by securely exchanging the public keys (e.g., how does Bob make sure that a key belongs to Alice?). For both problems (i.e., device initialization and trust establishment) secure key distribution is the main issue. Users can use USB sticks or can use local/wide area networks for key exchanges. The latter is less secure than the former, because it is vulnerable to the man-in-the-middle-attack—an attacker eavesdrops the channel and makes independent connections with the victims and relays messages between them, making them believe that they are talking directly to each other over a private connection, when in fact the entire conversation is controlled by the attacker.

6.5.7 Content Centric Access Control

We use attribute based encryption (ABE) with the goals of securely sharing content within a group across multiple untrusted servers and caches and yet of preventing collusion [BSW07]. ABE is the key enabler for attribute based access control (ABAC) that supports fine-grained access policies. Each user first generates an ABE public key and an ABE master key. A user can define a set of attributes (e.g., college friends, CS219 team, family members) and an access policy using Boolean formula on attributes. This allows a user to perform fine grained access control: the user assigns attributes to each peer and then issues a secret key corresponding to the assigned attribute to the peer (using the master key). The user encrypts the file once based on the access policy, and any peer can decrypt the file if he has attributes that satisfy the policy.

6.5.8 Secure Replica Management

PCN allows users to replicate any files over multiple devices. Users can also browse the files scattered over multiple devices. Any files can be updated as long as a user has a right to

update the content. Further, users can manage any files located in remote machines with "device-to-device communications" over CCN.

6.5.8.1 Synchronization

If a file is updated, a new version is created (with a new timestamp). To alert this event to the rest of nodes with the content, the node that makes the update will re-announce the corresponding prefix with a modification mark, which is a special type of prefix announcement for update notification. Each node will fetch the directory entries from the nodes that replicate the named prefix. From this, nodes can discover which files have updated and can perform file level synchronization.

Consistency: It is possible that a node may not be available at the time of an update announcement. In PCN, we support "eventual consistency" in which all replicas eventually converge to the same version given enough messages exchanged among participating devices (i.e., a file with the freshest timestamp) [PSY04, SKK90, RHR04]. Eventual consistency is one of the widely used consistency models in disruption-prone mobile environments.

A node should be able to synchronize the files as long as it is connected to the overlay network. When a node re-joins the overlay network after disruption, it first checks its neighbors to find any missing prefix announcements. This allows the node to search for the updates of the files located in its local storage. If the node finds a prefix with a modification mark, it performs file synchronization as illustrated earlier.

PCN's two-phase process for synchronization is less efficient than the schemes in traditional distributed file systems. The notification via prefix re-announcement does not tell us where the update is originating from; it only hints that it is from one of multiple replicas. As a result, each node must probe all replica nodes (under the named prefix) to discover which file has updated, and then must fetch the updated file for synchronization. One simple solution that can mitigate this problem is to add an extra field in the prefix announcement that lists updated files and their version information (or a locator to the file that lists updated files with version information).

Conflict resolution: Due to disconnected operations, PCN should deal with conflicts where multiple devices modify the same file without knowing other devices' modification. If all the updates are properly applied in the order of modification time, we can simply perform an automatic merging process using the UNIX diff algorithm. However, automatic merging may fail under disconnected operations. In this case, PCN notifies the user that a conflict has detected. The user will be presented with a revision history including authors, dates, and versioned content. It is up to the user to resolve the conflicts and mark the content as merged.

6.5.8.2 Update access control

PCN basically assumes that a namespace owner can manipulate the file as he wishes (e.g., as in Microsoft Windows). We can also implement a more sophisticated access control mechanism, e.g., UNIX like access control with read-only or read-write access rights. To this end, we maintain an ACL file in each directory that details access rights on files therein (say ".acl"), which is signed by the owner. The ACL file can be replicated as a regular file and can be securely distributed over the network. Replica nodes can then use this ACL file to decide whether to permit an update. The ACL file can be encrypted to protect an owner's privacy (e.g., using ABE). It is up to the user how to set up one's own personal content networking system.

6.5.8.3 Replica management

A user may want to know what files are stored where and wish to replicate files to remote devices. Regular content browsing like UNIX command *ls* does not tell users in which device the files are located. For replica management, PCN reserves a special file located in each replicated directory, namely ".replica" that contains the information about the replicated files in the named directory and the information about the device. In PCN, we reserve a special directory for devices, namely the "/dev" directory through which a user can freely name personal devices. For instance, Alice's iPad can be named as "/Alice/dev/iPad."

Further, each device announces this device name prefix, which will enable device-to-device communications over CCN.

A user can list the files and their replication status over the remote devices by simply collecting ".replica" files. This procedure is very similar to content browsing except that we are retrieving all the ".replica" files within the named prefix. For instance, Alice can check the replication status of her music files by fetching "/Alice/my music/.replica." If the directory contains sub-directories, the user needs to recursively retrieve sub-directories to know replication status. Note that a user may not wish to reveal the replication status to other people as it may be considered personal. In this case, the user can encrypt ".replica" files using ABE and assign attributes like "personal items" to ensure that he can only know the replication status.

CHAPTER 7

Evaluation II: Decentralized Content Confidential and Authorization

We now describe the implementation and evaluation of our confidential and authorized access personal content network prototype.

7.1 Prototype Implementation

We implemented a PCN prototype in Linux and the Android platform. Since the current CCNx codebase only supports manually configured, static network topology, we implemented an overlay network client (called *ccn-overlay*) that builds and maintains an overlay network based on social relationship. Whenever network topology changes, an overlay client uses external commands to re-configure the local CCND. The prefix announcement will be disseminated through the overlay clients because the current CCNx codebase does not fully support the prefix announcement feature. Each client periodically exchanges ping messages to check whether its neighboring nodes are alive. Recall that we have three types of prefix announcements, namely regular prefix, modification, and key revocation announcements. Besides device initialization, users can establish a trust relationship using a tool called *pcn-intro*, which is based on UIA's device management UI tool [FSL06].

For ABE support, we used the CP-ABE toolkit [cpa12]. A file published in the local repository can be encrypted using *pcn-abe-enc*. This tool communicates with the local CCN repo daemon, encrypts the named file, and re-publishes the file into the repository. ABE keys will be stored in a user's local keystore (e.g., *.ccnx* at home). If a file is encrypted, the *ccn-fuse* tool automatically decrypts the file and returns the plaintext to the reader. It accesses

	MK setup	SK: 5	SK: 10	SK: 15
Laptop	166(±0.2)	531(±0.4)	913(±0.2)	1343(±1.9)
Mobile	354(±0.9)	2068(±0.5)	3981(±0.5)	5947(±0.3)

Table 7.1: CP-ABE performance of Laptop (L) and Nexus One (M) in milliseconds: master key (MK) setup and secret key (SK) generation with k number of attributes

	1KB	10KB	100KB	1MB	10MB	100MB
[D2L] CCNx retrieve	1152 (±0.4)	1225.8(±1.0)	1410(±2.5)	2102.2(±2.3)	11085.4(±16.3)	80593.8(±139.1)
Local ABE pri-key	8.2(±0.1)	10.2(±0.1)	9.8(±0.)	9.4(±0.2)	13.6(±0.1)	16.6(±0.1)
Remote ABE pub-key	358(±0.1)	343.6(±0.1)	346(±0.2)	348.6(±0.2)	346.4(±0.2)	348.4(±0.1)
AES key decrypt	37.8(±0.3)	37.7(±0.2)	37.8(±0.4)	36.8(±0.9)	37.3(±0.5)	37.1(±0.5)
Content decrypt	1.0(±0.1)	1.0(±0.1)	3.2(±0.1)	35.0(±0.2)	380.2(±1.6)	3946.7(±0.8)
[L2M] CCNx retrieve	784.8(±0.5)	973.6(±0.9)	1157.8(±0.5)	2273.2(±4.3)	10751(±18.5)	106752.6(±154.6)
Local ABE pri-key	37.6(±0.1)	39.6(±0.1)	38.6(±0.0)	37.6(±0.1)	39(±0.0)	39.4(±0.1)
Remote ABE pub-key	527(±0.5)	533(±0.2)	532.2(±0.2)	536.4(±0.2)	538.8(±0.1)	539(±0.1)
AES key decrypt	425.8(±0.4)	428.6(±0.3)	428.1(±0.8)	427.1(±1.3)	425.6(±3.3)	433.6(±6.1)
Content decrypt	2.1(±0.1)	19.9(±0.3)	120.0(±0.2)	402.0(±1.1)	3414.9(±2.3)	20713.4(±5.3)
[D2M] CCNx retrieve	706.4(±0.1)	914.2(±0.6)	1146.2(±0.3)	2096.2(±1.1)	10259.4(±26.6)	96724.2(±218.5)
Local ABE pri-key	35(±0.1)	36.2(±0.1)	37.6(±0.1)	38.6(±0.1)	39.2(±0.1)	39.6(±0.1)
Remote ABE pub-key	532.4(±0.1)	425.8(±0.1)	433(±0.1)	432.4(±0.1)	431.8(±0.1)	431.4(±0.1)
AES key decrypt	427.1(±5.3)	419.6(±7.1)	429.7(±7.3)	435.1(±10.1)	429.3(±6.1)	435.8(±11.3)
Content decrypt	2.4(±0.1)	18.1(±0.4)	128.0(±0.1)	387.1(±1.3)	3371.1(±2.5)	21001.4(±4.1)

Table 7.2: Breakdown of retrieval time (in milliseconds) of a file. D2L: Desktop to Laptop, L2M: Laptop to Nexus One, D2M: Desktop to Nexus One. Each result is the mean of 5 trials with a 95% confidence interval. Each trial was run by setting the CCN cache size to 0 (CCND_CAP=0) and restarting the CCND in between each run to reset the local cache.

the user's local keystore for decryption. We ported the CP-ABE toolkit to the Android platform via cross-compilation. Since CCNx codebase supports the Android platform, we integrated the basic PCN tools to the mobile platform.

7.2 Evaluation

We present our preliminary system evaluation answering the following questions: (1) What is the overhead of ABE? (2) What is the detailed performance of each component used in PCN? (3) Given realistic user traces, what is the overhead of PCN (e.g., routing table size, update overhead)?

To provide secure personal sharing, we have designed our implementation to incur minimal overhead to the existing CCNx codebase. While a complete evaluation of the CCNx method traces are outside the scope of this paper, our experience shows that the CCNx performance is improving with every release. We analyzed the performance of providing security using ABE in three major areas, namely (1) key setup and generation, (2) encrypting and storing content, and (3) retrieving and decrypting content. To model user behavior we measured the performance using a mobile device the Android Nexus One (Qualcomm Snapdragon 1GHz, 512MB of RAM), a laptop (Dell Inspirion 9400 with Intel dual core 2Ghz CPU, 2GB of RAM, and Intel WiFi Link 5300 that runs Ubuntu 10.10 with Linux 2.6.35), and a desktop (Apple iMac with Intel i5-2500s 2.7Ghz CPU and Broadcom Gigabit Ethernet that runs Ubuntu 10.10 with Linux 2.6.35). The measured device-to-device TCP performance using Iperf is given as follows (average of 10 trials with 95% confidence interval): Nexus One to Laptop over Wi-Fi: 8.21Mbps ($\pm$0.02), Laptop to Nexus One over Wi-Fi: 8.00 Mbps ($\pm$0.01), Desktop to Laptop (Laptop Wi-Fi and desktop wired): 10.34 Mbps ($\pm$0.02).

Table 7.1 shows the master key setup delay and the secret key generation delay as a function of the number of attributes. The results show that the delay almost linearly increases with the number of attributes. The master key setup is independent of the number of attributes, and that of a laptop and Nexus one is given as 166ms and 354ms, respectively.

We then measure the performance of remote content retrieval (single hop). Retrieving/decrypting involves with a number of steps: (1) FUSE open/read call (in a laptop only), (2) CCN data retrieval over a remote node, (3) ABE local repository key look-up, (4) ABE public key retrieval from a remote node, (5) CP-ABE decryption of an AES key, and (6) content decryption with AES-256. The most time consuming operations are CCN retrieval and CP-ABE decryption, and the delay linearly increases with the file size (see Table 7.2). The cost of FUSE operations took less than few milliseconds, and we did not report the delay in the table.

All of our results give an indication that the performance of generating private keys, encrypting content, and decrypting content tends to grow linearly with the size of the file. The ABE private key lookup and retrieval consumes only fractions of milliseconds, and

	Naming	DTN	Topology	Replication Unit	Update	Trust	Access Control	Secure Binding
Ficus	SP+H	Yes	P2P	File/Dir	Yes	-	ACL	-
BlueFS/EnsemBlue	SP+H	Yes	C/S	File	Yes	-	ACL	-
UIA/Eyo	DP+H	Yes	P2P	-	-	-	ACL	-
PersonalRAID	SP+H	Yes	P2P	Volume	Yes	-	-	-
Footloose	SP-F	Yes	P2P	File	Yes	-	-	-
DisCFS	SP-H	No	C/S	Volume	Yes	KeyNote	Certs	-
Bayou	SQL	Yes	P2P	Volume	Yes	PKI	Certs	-
Plutus/SiRiUS	SP-H	No	C/S	-	-	PKI	Certs/Enc-PKC	-
PAST	SP-F	No	P2P	File	Yes	PKI	Certs/Enc-PKC	-
CCN	SP-H	Yes	P2P	File	No	PKI	Certs/Enc-PKC	Yes
PCN	SP-H	Yes	P2P	File/Dir	Yes	SPKI	Certs/Enc-ABE	Yes

Table 7.3: Feature comparison: DTN (Delay Tolerant Networking), SP/DP (Single Persistent or Device Persistent), F/H (Flat/Hierarchical), PKC (Public-Key Cryptography)

retrieving the ABE public key as content from CCN takes only a couple hundred milliseconds.

7.3 Discussion

Security attacks: PCN shares the security benefits of CCN as it is a pull-based content retrieval and uses secure binding, thereby effectively thwarting a distributed denial of service attack, a request flooding attack, and a man-in-the-middle-attack [JST09b]. While PCN introduces new features like extra prefix announcements (modification and revocation) and content updates, PCN's explicit prefix protection restricts that only authorized users can replicate a named prefix. Moreover, PCN limits that replicated content can be updated by the individual with explicit write-permissions. Thus, a user can neither request content replication nor inject an update without explicit permissions from the content owner, as illegitimate requests are automatically discarded by the intermediate PCN nodes.

Energy efficiency: The PCN system includes battery powered personal devices. Battery limited devices need to constantly listen to the announcement messages which prevents them from switching to a sleep mode for power saving. Recall that whenever there are updates, PCN broadcasts the messages to the k-hop neighbors in the overlay network. One solution to this problem is to introduce a *proxy server* in an AC powered device (e.g., desktop or laptop). A mobile device can re-configure the underlying overlay network topology

such that messages always travel through the local proxy server. The local proxy buffers all the incoming announcements. Then, the mobile client periodically wakes up and pulls the aggregated announcements.

Interest-based push for synchronization: In our prototype implementation, we used the prefix announcement to notify content updates to the replica nodes. An alternative to this approach is to use *interest solicitation* as suggested in the NDN proposal [Zha10]. A node that updated the content will send an interest solicitation packet to the replica nodes who are interested in receiving the updated content. Those interested replica nodes will then send an interest packet requesting this updated content. For efficient synchronization, the interest solicitation packet includes detailed information about the updated content as we augmented the prefix announcement.

Private PCN: For security reasons, a user could have two different namespaces: one for private access and the other for shared access. The private PCN is not visible to the other users, and thus, a user can simplify the access control; e.g., just setting a single attribute for content encryption. Given that a large fraction of content is personal use only, we expect that a private PCN network can possibly lower the burden of content management.

Offline devices: If devices are offline, a user cannot browse the content stored in the devices. To help content retrieval from off-line devices, PCN can take a similar approach used in Eyo [SLP09]. Each device periodically pulls the content lists of the other devices and stores them in its local repository. Given this information, PCN nodes can tell which device has a file, and thus, a user can access the file from the off-line devices.

CHAPTER 8

Preliminaries III: Privacy

An open vehicular data testbed composed of detailed driver behaviors and real-world statistics would provide a greater understanding and help mitigate the number of self-driving vehicle accidents due to driver behavior misunderstandings. Ideally, such data should be gathered and aggregated into an open vehicular data testbed where manufacturers and regulators are able to test and validate vehicle safety standards. However, such a vehicle data testbed composed of detailed statistics should respect driver's privacy expectations.

Suppose that most drivers in the future will allow their medical providers to monitor their vital signs via LTE while they are driving. The driver does not mind sharing their identification (with authentication) and accurate medical sensor data with their own provider. However, unless he/she expects to be rescued by the Medical Provider Paramedics in case of collapse (say heart attack or diabetic crises), they would like to protect their location privacy.

The provider uses the data for various purposes. One purpose could be to study the ability of patient to drive safely with certain physical conditions. This monitoring would possibly be required if an independent study showed that brain cancer patients of certain stage,say, are unsafe drivers. The provider wants to monitor driving behavior of its own patients to verify the claim and possibly get disclaimers from unsafe patients so it does not get sued. The collection of such personal information should be uploaded privately and the data should be privatized yet maintain utility.

Next, suppose that academic analysts want to access this data for their research correlating accidents, safe driving and driver medical conditions. A single provider will not release the data, albeit anonymized, for fear to be blamed that its patients are the unsafest or the

physically least fit drivers.

To overcome this problem, the providers get together to generate a *privatized decentralized* database from which it is impossible to tell who is the driver and which provider the driver belongs to. This shows the need to use a decentralized solution. In fact, a centralized solution would not work for the following reasons: the centralized analyst would not be able to authenticate the drivers without learning their IDs, violating anonymity (in our decentralized scheme, providers guarantee authenticity). Furthermore, in the centralized scheme the analyst would probably be able to infer location, in spite of privatization attempts, by colluding with LTE provider and then track the driver (while with decentralized provider indirection the driver is protected).

In the remainder of this section we provide background information on differential privacy, which is the *gold* standard for privacy mechanism. We then provide an overview of the GPS daemon which runs on the majority of GPS enabled devices providing location updates. Finally, we discuss how data owners are able to create non-attributable writes to a distributed database without the database operator learning which data owner wrote a particular value.

8.1 Differential Privacy

Roughly speaking, differential privacy says that the ability of an adversary to inflict harm should be essentially independent of whether any individual opts in to or out of, the dataset [Dwo11]. Thus, a data owner may safely utilize differential privacy techniques when sharing their personal data, as it enables them control insight into their personal information.

A popular technique which satisfies differential privacy is the randomized response mechanism, originally proposed in the 1960s [War65, FT86]. Randomized response has been shown to be optimal in the local privacy model [] and is used by many companies today (e.g., Apple, Google [EPK14]) due to its simplicity while satisfying the differential privacy guarantee.

However, a drawback to the randomized response mechanism is poor scalability, i.e., the

error in the estimate quickly increases with the population size due to the underlying truthful distribution distortion. We elaborate further in Section 9.3.2.

Some protocols which leverage the randomized response mechanism have made assumptions that the majority of the underlying truthful population truthfully responds "Yes" (e.g., the percentage is greater than 2/3 or 3/4) in order to preserve accuracy. However, it's not clear what privacy guarantees can be provided this way since any adversary is able to successfully guess with greater than 50% probability the value of any data owner in such a population. For example, suppose our query is how many home owners reside within 15 blocks from the beach, yet we ask only those home owners within 20 blocks from the beach. We would have a pretty good guess of where the "yes" respondents reside.

The Laplace mechanism was introduced as a way to add privacy noise independent of the database size [DMN06] by drawing privacy noise from the Laplace distribution. The Laplace mechanism is calibrated to the max difference between any two rows in the database. That is, the noise is sufficient to protect the max leakage that any particular data owner induces. For example, first a service aggregates all the data owners truthful responses. Then, the service draws from the Laplace distribution by calibrating the variance according to the desired privacy strength. Drawing from other distributions such as Gaussian also satisfies differential privacy, though the Laplace mechanism is preferred as it's mathematically cleaner [DR14].

However, there is a drawback to the Laplace mechanism in graph datasets such as social networks [GLP11, GHL12] or vehicle commuting patterns. Even if a particular data owner does *not* participate, their friends that do participate leak information that can be used to deprivatize the targeted data owner (e.g., shadow profiles). For example, it is possible to learn political beliefs or sexual orientation even if a particular individual does not participate and maintain an active profile in an online social network. An adversary simply needs to analyze the similarity metrics amongst the social circles that a data owner participates in to understand politics beliefs or sexual orientation [SGS14, HHH12, Gay09, KSG13, JM09].

Furthermore, if the graph structures of a social network are eventually anonymized and released, an adversary simply needs to participate and influence the graph structure (e.g., by

joining a social network) to learn and influence the actual social graph before it's privatized and released. Thus, there needs to be a mechanism which also perturbs the underlying *structure* of the data itself and preserves accuracy as the underlying distribution structure becomes distorted.

Sampling whereby responses are randomly discarded reduces the the graph dependencies leaked by a targeted individuals connections. Severing connections reduces the social circle size and makes it challenging for the adversary to make similarity inferences from reduced social circles alone. Thus, it has been shown that the strength of privacy mechanisms is increased by applying sampling and reducing the privacy leakage [NRS07, KLN08, GHL12].

To guarantee data owner privacy upon the release of data, various mechanisms have been proposed [MGK06, Swe02, LLV07, Dwo06, DMN06]. Differential privacy has emerged as the strongest of these privacy mechanisms [Dwo06, DMN06]. The core idea of differential privacy is to provide strong bounds and guarantees on the privacy leakage when multiple aggregate analytics are run despite the presence or absence of a single data owner from the dataset. This privacy mechanism is provided by adding differentially private noise to the aggregrate answer. As opposed to the originally proposed differentially private mechanism which first collects data in a centralize database and then privatizes the release of the data, LocationSafe immediately privatizes the data at the data source (sensor) in real-time.

Randomized response operates in the distributed model (local privacy) and is used by many companies today (e.g., Apple, Google [EPK14]) due to its simplicity while satisfying the differential privacy guarantee. However, a drawback to the randomized response mechanism is poor scalability, i.e., the error in the estimate quickly increases with the population size due to the underlying truthful distribution distortion. These protocols, such as Rappor [EPK14], require an inordinate amount of samples, yet still lack strong utility.

Some protocols which leverage the randomized response mechanism have made assumptions that the majority of the underlying truthful population truthfully responds "Yes" (e.g., the percentage is greater than 2/3 or 3/4) in order to preserve accuracy. However, it's not clear what privacy guarantees can be provided this way since any adversary is able to suc-

cessfully guess with greater than 50% probability the value of any data owner in such a population. For example, suppose our query is how many home owners reside within 15 blocks from the beach, yet we ask only those home owners within 20 blocks from the beach. We would have a pretty good guess of where the "yes" respondents reside.

Zero-knowledge privacy [GLP11] is a cryptographically influenced privacy definition that is strictly stronger than differential privacy. Crowd-blending privacy [GHL12] is weaker than differential privacy; however, with a pre-sampling step, satisfies both differential privacy and zero-knowledge privacy. However, these mechanisms are suited for the centralized system model and rely on aggressive sampling, which significantly degrades the accuracy estimations.

Differential privacy [DMN06] has been proposed as a privacy definition such that anything that can be learned if a particular data owner is added to the database could have also been learned before the data owner was added. A data owner is thus "safe" to participate as statistical inferences amongst the aggregate are learned yet specific information regarding the individual is not learned.

Distributional privacy [BLR13] is a privacy mechanism which says that the released aggregate information only reveals the underlying ground truth distribution and nothing more. Each data owner is protected by the randomness of the other randomly selected data owners rather than by adding explicit privacy noise to the output. The indistinguishability from the underlying distribution protects individual data owners and is strictly stronger than differential privacy. However, it is computationally inefficient though can work over a large class of queries known as Vapnik-Chervonenkis (VC) dimension.

Sampling. Sampling whereby a centralized aggregator randomly discards responses has been previously formulated as a mechanism to amplify privacy [CM06, NRS07, KLN08, GHL12]. The intuition is that when sampling approximates the original aggregate information, an attacker is unable to distinguish when sampling is performed and which data owners are sampled. These privacy mechanisms range from sampling without a sanitization mechanism, sampling to amplify a differentially private mechanism, sampling that tolerates a

bias, and even sampling a weaker privacy notion such as k-anonymity to amplify the privacy guarantees.

However, sampling alone has several issues. First, data owners are not protected by plausible deniability as data owners do not respond "No". Second, the estimation of the underlying truthful "Yes" responses quickly degrades as we increase the population that truthfully responds "No".

Multi-party Computation. Multi-party computation (MPC) is a secure computation model whereby parties jointly compute a function of the data such that each party only learns the aggregate output and nothing more. However, MPC mechanisms that release the *exact* answer have no strong privacy guarantees against active privacy attacks, particularly when the data is publicly published. A participant that does not perturb their responses and provides their *exact* answer is easily attacked by an adversary that knows the values of $n-1$ participants. For example, an adversary first runs a counting query that includes all n data owners and then runs a second counting query over $n-1$ data owners (the targeted data owner is the excluded row). Subtracting the two results *reveals* the value of the targeted data owner. In contrast, the differential privacy model assumes a strong adversary that knows the $n-1$ data owner values. In this paper we combine MPC and differential privacy by introducing a sampling-based privacy mechanism that maintains constant error and show a performance optimization for a new cryptographic primitive named Function Secret Sharing [BGI15].

8.1.1 Randomized Response Privacy Guarantee

8.1.1.1 Privacy Guarantee of Randomized Response

The randomized response mechanism achieves ϵ-differential privacy, where:

$$\epsilon = \max\left(\ln\left(\frac{\Pr[\text{Resp='Yes' }|\text{'Yes'}]}{\Pr[\text{Resp='Yes' }|\text{'No'}]}\right), \ln\left(\frac{\Pr[\text{Resp='Yes' }|\text{'No'}]}{\Pr[\text{Resp='Yes' }|\text{'Yes'}]}\right)\right)$$

More specifically, the randomized response mechanism [FT86] achieves ϵ-differential pri-

vacy, where:

$$\epsilon = \ln\left(\frac{\pi_1 + (1-\pi_1) \times \pi_2}{(1-\pi_1) \times \pi_2}\right) \tag{8.1}$$

That is, if a data owner has the sensitive attribute A, then the randomized answer will be "Yes" with the probability of '$\pi_1 + (1-\pi_1) \times \pi_2$'. Else, if a data owner does not have the sensitive attribute, then the randomized answer will become "Yes" with the probability of '$(1-\pi_1) \times \pi_2$'.

8.2 GPSD

GPSD is a daemon that network enables the GPS sensor on the majority of mobile embedded systems including Android, iOS, Windows Mobile, UAVs, and driverless cars [GPS]. On smartphones the network access is limited to localhost applications only (as opposed to remote applications). GPSD enables unfettered access to location data and does not enable or provide any privacy guarantees. LocationSafe provides a privacy module that provides uniform private access across all platforms.

Mobile device permission systems has received attention in the past. Human interaction studies which seek to enhance reader comprehension have been proposed and evaluated [FHE12, FEW12]. Such systems lack strong and enforacable privacy guarantees. Static analysis tools have been proposed [YYZ13]. Though such systems serve only to notify the data owner of privacy breaches and are unable to enforce any privacy runtime guarantees. However, these solutions modify the underlying OS thus making them specific to a single OS or device [ZZJ11, XPr]. Furthermore, these solutions are unable to balance the privacy and utility tradeoff, ultimately resulting a binary approach to privacy.

8.3 Non-Attributable Writes

Consider a data owner that would like to write into a distributed database without any of the database operators learning which row we wrote into. Knowledge of the row and thus data uploaded by the owner would allow the operator to track the owner over subsequent

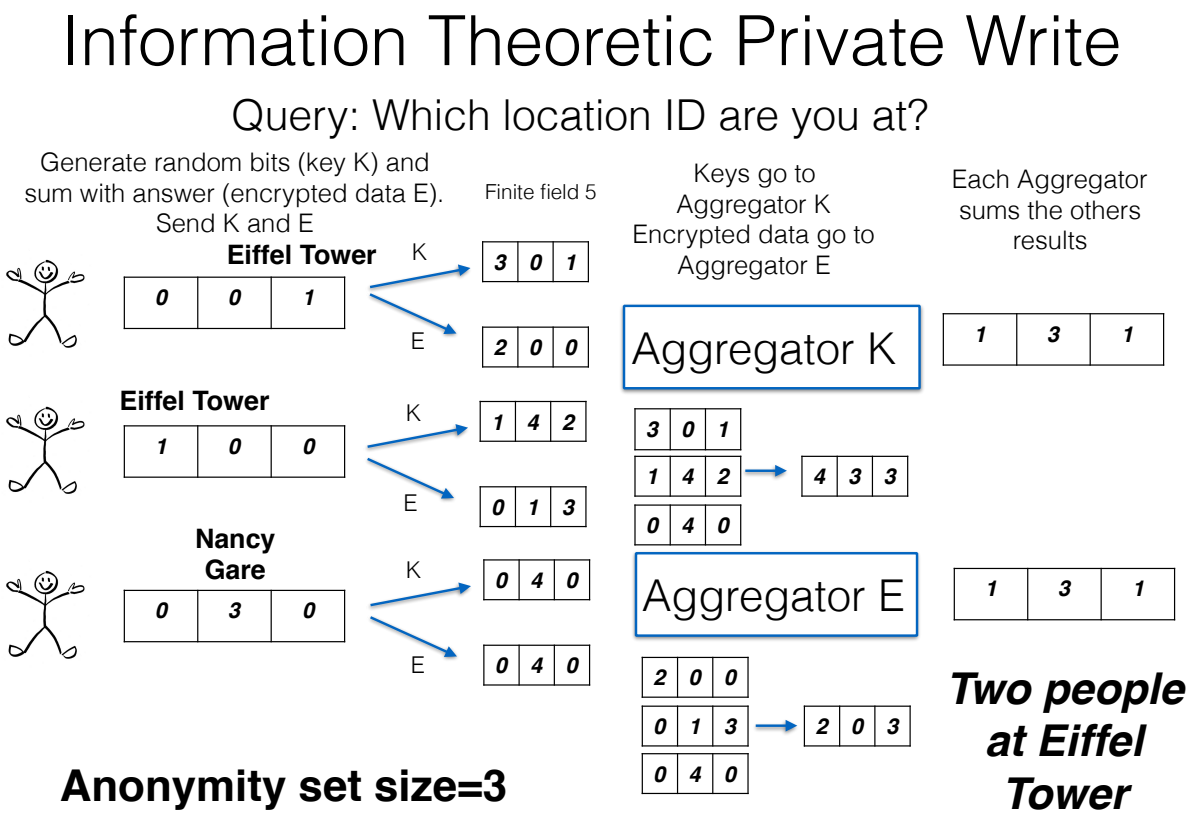

Figure 8.1: **Each data owner uniformly at random selects a slot to write their location ID. The aggregators are unable to determine which data owner wrote to a particular slot, as long as there is one honest aggregator who does not collude. The aggregate count of each location ID is computed as the final step.**

epochs even if the data is privatized.

We assume a distributed database setting such that, so long as one database operator remains honest and does not collude with other database operators, it is not possible to learn which database row was written into (as long as there are at least two data owners participating). The data owner should continuously re-select a new database row at random every epoch.

(Two Party) Let us first consider two operators and two databases. The protocol proceeds as follows.

Assume the database is represented by n rows. Each data owner uploads a message of size m bits, in a randomly chosen row. Without loss of generality, for our example we describe now we assume m is one bit. Thus the database is a bitstring. Extending to a message size of more than one bit would only require a larger finite field (instead of finite field size 2 we could choose a prime number larger than the desired message size in bits).

Each data owner begins with a bitstring of length n (the size of the database). The data owner uniformly at random selects an index of the bitstring and sets its message value (assume it is to 1). Every other index is set to 0.

Next, the data owner creates a key by generating a random bitstring of length n.

The data owner then XORs the randomly generated bitstring key with the bitstring containing the message (that has only one index set) to produce the encrypted bitstring.

The data owner then transmits the encrypted bitstring to one database operator and the key bitstring to the other database operator. The data owner may randomly decide which database operator to send the encrypted and key bitstrings.

The same process repeats for each data owner. That is, a second data owner repeats the process of uniformly at random selecting an index of the bitstring to set to 1, generating a key bitstring, and encrypting the bitstring.

As the database operators receive each bitstring (either encrypted or key bitstring), each database operator cumulatively XORs the bitstrings.

Finally, at the end of an agreed epoch, the database operators share the cumulatively XORed bitstrings with each other. By doing so, they are able to reconstruct a database consisting of each data owners message at their specified indexes. The privacy guarantee is that the database operators are unable to determine which data owner wrote to which index, as long as there are at least two participating data owners and there is at least one database operator that does not collude with any other. (There is also an assumption that the data owners do not write to the same index, though collisions can be probabilistically avoided by

sizing the database large enough to minimize the likelihood of collisions).

(Key Size) The issue is that the key size is the length of the database n. Suppose the number of data owners is on the order of millions and the database row size is several hundred bits. The bitstring size will be of the order of several hundred MBs, which is prohibitively expensive for mobile devices and edge networks continually uploading every few minutes.

We could compress each of the key bitstrings by using a pseudorandom generator (PRG) for the Z-1 keys. However, we somehow must compress the bitstring containing the message (that has only one index set). Unfortunately, by definition of a PRG, it is computationally difficult to generate a PRG seed that expands to the desired bitstring. We must utilize a more sophisticated approached to enable cryptographic compression of the bitstring. We utilize a new primitive named Function Secret Sharing (FSS) [BGI15] that achieves a square root size reduction in the key size. Our contribution to the function secret sharing primitive is an order of magnitude scalability enhancement.

Private Data Upload. Wang et al. [WYG17] employed and extended the function secret sharing primitive to enable efficient private information retrieval operations that protect the data owner's queries from being learned by the database operators. They proposed an optimization by using the Matyas-Meyer-Oseas one-way compression function as an alternative to the heavy AES operations for the *two party* case. Wang et al. achieves a 2.5x speedup by utilizing one-way compression functions. Our K Privacy demonstrates an *order of magnitude* improvement by a square root reduction of the.number of AES initializations for the *multi-party* function secret sharing protocol.

8.3.1 Share Verification

We now describe the MPC protocol [BGI16] run amongst the aggregator parties to verify all data owner shares. The protocol does not violate data owner privacy and is extremely

efficient as it does not utilize any publick-key primitives and relies solely on finite field operations.

We first describe the MPC protocol in detail and then provide an example.

MPC Protocol Let p represent the number of parties participating in the protocol.

Let n represent the unit vector length (e.g., length of the bit string or number of database slots).

Let m represent the number of bits of the message M. Let $M \in \mathbb{F}_Z$ where Z is a relatively large prime number.

Given $\mathbb{F}_Z$ a finite field of characteristic Z where Z is a relatively large prime, let $\mathbf{R}$ be a blinding (randomization) matrix where the the values in the first row are chosen uniformly at random over $0, ..., Z-1$.

This is a particular randomization matrix such that elements of each row is raised to the power of the first row, where the power is equivalent to the row number. There will be a total of p rows, one for each party. That is,

$$\mathbf{R} = \begin{bmatrix} r_1 & r_2 & ... & r_n \\ r_1^2 & r_2^2 & ... & r_n^2 \\ ... & ... & ... & ... \\ r_1^p & r_2^p & ... & r_n^p \end{bmatrix} \tag{8.2}$$

We wish to secretly share a unit vector and verify that the shares correctly sum to the unit vector.

For example,

$$\hat{\mathbf{u}} = \begin{bmatrix} 0 \\ \mathrm{M} \\ ... \\ 0 \end{bmatrix} \tag{8.3}$$

The value can be m bits taking on a value from the finite field of character of characteristic p where p is a relatively large prime.

To share $\hat{\mathbf{u}}$, the user can randomly generate a total p vectors $\mathbf{V_i}$

$$\mathbf{V_i} = \begin{bmatrix} v_{i,1} \\ v_{i,2} \\ \dots \\ v_{i,n} \end{bmatrix} \tag{8.4}$$

such that

$$\sum_{i=1}^{p} \mathbf{V_i} = \hat{\mathbf{u}} \tag{8.5}$$

We then blind these values such that

$$\sum_{i=1}^{p} \mathbf{R} \cdot \mathbf{V_i} = \mathbf{R} \cdot \hat{\mathbf{u}} \tag{8.6}$$

Let's describe an example where $n = 2$ and $p = 3$.

We know that sum of the vectors should equal the unit vector.

$$\begin{bmatrix} v_{1,1} \\ v_{1,2} \end{bmatrix} + \begin{bmatrix} v_{2,1} \\ v_{2,2} \end{bmatrix} + \begin{bmatrix} v_{3,1} \\ v_{3,2} \end{bmatrix} = \hat{\mathbf{u}} \tag{8.7}$$

We now apply the randomization (blinding) matrix.

$$\begin{bmatrix} r_1 & r_2 \\ r_1^2 & r_2^2 \\ r_1^3 & r_2^3 \end{bmatrix} \cdot \begin{bmatrix} v_{1,1} \\ v_{1,2} \end{bmatrix} + \begin{bmatrix} r_1 & r_2 \\ r_1^2 & r_2^2 \\ r_1^3 & r_2^3 \end{bmatrix} \cdot \begin{bmatrix} v_{2,1} \\ v_{2,2} \end{bmatrix} + \begin{bmatrix} r_1 & r_2 \\ r_1^2 & r_2^2 \\ r_1^3 & r_2^3 \end{bmatrix} \cdot \begin{bmatrix} v_{3,1} \\ v_{3,2} \end{bmatrix} = \mathbf{R} \cdot \hat{\mathbf{u}} \tag{8.8}$$

$$\begin{bmatrix} r_1 \cdot v_{1,1} + r_2 \cdot v_{1,2} \\ r_1^2 \cdot v_{1,1} + r_2^2 \cdot v_{1,2} \\ r_1^3 \cdot v_{1,1} + r_2^3 \cdot v_{1,2} \end{bmatrix} + \begin{bmatrix} r_1 \cdot v_{2,1} + r_2 \cdot v_{2,2} \\ r_1^2 \cdot v_{2,1} + r_2^2 \cdot v_{2,2} \\ r_1^3 \cdot v_{2,1} + r_2^3 \cdot v_{2,2} \end{bmatrix} + \begin{bmatrix} r_1 \cdot v_{3,1} + r_2 \cdot v_{3,2} \\ r_1^2 \cdot v_{3,1} + r_2^2 \cdot v_{3,2} \\ r_1^3 \cdot v_{3,1} + r_2^3 \cdot v_{3,2} \end{bmatrix} = \mathbf{R} \cdot \hat{\mathbf{u}} \tag{8.9}$$

$$\begin{bmatrix} r_1 \ (v_{1,1} + v_{2,1} + v_{3,1}) + r_2 \ (v_{1,2} + v_{2,2} + v_{3,2}) \\ r_1^2 \ (v_{1,1} + v_{2,1} + v_{3,1}) + r_2^2 \ (v_{1,2} + v_{2,2} + v_{3,2}) \\ r_1^3 \ (v_{1,1} + v_{2,1} + v_{3,1}) + r_2^3 \ (v_{1,2} + v_{2,2} + v_{3,2}) \end{bmatrix} = \mathbf{R} \cdot \hat{\mathbf{u}} \tag{8.10}$$

Since the summation of the elements of a unit vector should sum to zero, we can denote the value as follows

$$\begin{bmatrix} a + b \\ a^2 + b^2 \\ a^3 + b^3 \end{bmatrix} = \mathbf{R} \cdot \hat{\mathbf{u}} \tag{8.11}$$

From Equation 8.7 that the sum of the vectors is the unit vector. Thus, we then know that if the shares are properly formed that a and b should represent either all zeros or the blinded message. Thus, $(a+b)^2 - (a^2+b^2) = 0$ and $(a+b)^3 - (a^3+b^3) = 0$.

If a and b are both zero then the terms fall out.

In the case of only a or b being the blinded message the terms fall out.

If both a and b are non-zero then the difference will be a non-zero value. These shares are invalid and should be discarded.

8.3.1.1 Alternate Algorithms

There are two alternate algorithms for the "square" algorithm described above, which were also presented in [BGI16]. The same process is used, but the structure of the blidning matrix is different, as well as the final check of $\mathbf{R} \cdot \hat{\mathbf{u}}$. The first algorithm is the "product" algorithm where

$$\mathbf{R} = \begin{bmatrix} r_{1,1} & r_{2,1} & \dots & r_{n,1} \\ r_{1,2} & r_{2,2} & \dots & r_{n,2} \\ \dots & \dots & \dots & \dots \\ r_{1,p} & r_{2,p} & \dots & r_{n,p} \end{bmatrix} \tag{8.12}$$

such that

$$\forall i \prod_{j=1}^{p-1} r_{i,j} = r_{i,p} \tag{8.13}$$

Then, we can apply the bliding matrix to our vectors $\mathbf{V_i}$, to achieve the final result:

$$\begin{bmatrix} a_1 + b_1 \\ a_2 + b_2 \\ a_3 + b_3 \end{bmatrix} = \mathbf{R} \cdot \hat{\mathbf{u}} \tag{8.14}$$

where

$$\prod_{i=1}^{p-1}(a_i + b_i) = a_p + b_p \tag{8.15}$$

An Alternative scheme is the "inverse" algorithm, which has a blinding matrix of

$$\mathbf{R} = \begin{bmatrix} r_{1,1} & r_{2,1} & \dots & r_{n,1} \\ r_{1,2} & r_{2,2} & \dots & r_{n,2} \\ \dots & \dots & \dots & \dots \\ r_{1,p} & r_{2,p} & \dots & r_{n,p} \end{bmatrix} \tag{8.16}$$

such that

$$\forall i \ \prod_{j=1}^{p} r_{i,j} = 1 \tag{8.17}$$

Then,

$$\begin{bmatrix} a_1 + b_1 \\ a_2 + b_2 \\ a_3 + b_3 \end{bmatrix} = \mathbf{R} \cdot \hat{\mathbf{u}} \tag{8.18}$$

where

$$\prod_{i=1}^{p}(a_i + b_i) = 1 \tag{8.19}$$

8.3.1.2 Share Verification Analysis

Here we analyze the protocol to ensure that data owners' responses are correctly formed unit vectors where all indexes are zero except for only one index.

Correctness The protocol outputs whether the final answer is a unit vector (i.e., all the indexes are zero except for one location). If the vector is all zeroes then the sum will be

zero. If the answer is a unit vector then the blinded message terms fall out leaving zero. If the vector is not a unit vector, the sum will be non-zero and we can discard this share.

Privacy All parties only view their own input and the final output. The blinding mechanism effectively masks the data owners true value.

Fairness All parties which participate will all view the same final answer as the shares sum to the same value.

CHAPTER 9

Scalable Privacy

9.1 LocationSafe

Today, mobile data owners lack consent and control over the release and utilization of their location data. Third party applications continuously process and access location data without data owners granular control and without knowledge of how location data is being used. The proliferation of IoT devices will lead to larger scale abuses of trust.

In this dissertation we present the first design and implementation of a privacy module built into the GPSD daemon. The GPSD daemon is a low-level GPS interface that runs on GPS enabled devices. The integration of the privacy module ensures that data owners have granular control over the release of their GPS location. We describe the design of our privacy module and then evaluate the performance of private GPS release and demonstrate that strong privacy guarantees can be built into the GPSD daemon itself with minimal to no overhead.

Today data owners' personal mobile devices are constantly being tracked and monitored by third party applications without data owners granular consent and control. Data owners' trust is being continuously violated [Fac].

Data owners have a desire to occasionally share their location data, though desire granular control and approved consent. Third party analysts seek to track data owners continuously. Unfortunately today this tension has resulted in disproportionate control being in favor of the third party analysts.

Recent research has tried to improve user behavior in recognizing permission issues [FHE12], user-defined runtime constraints [NKZ10], or tools to help developers identify least-

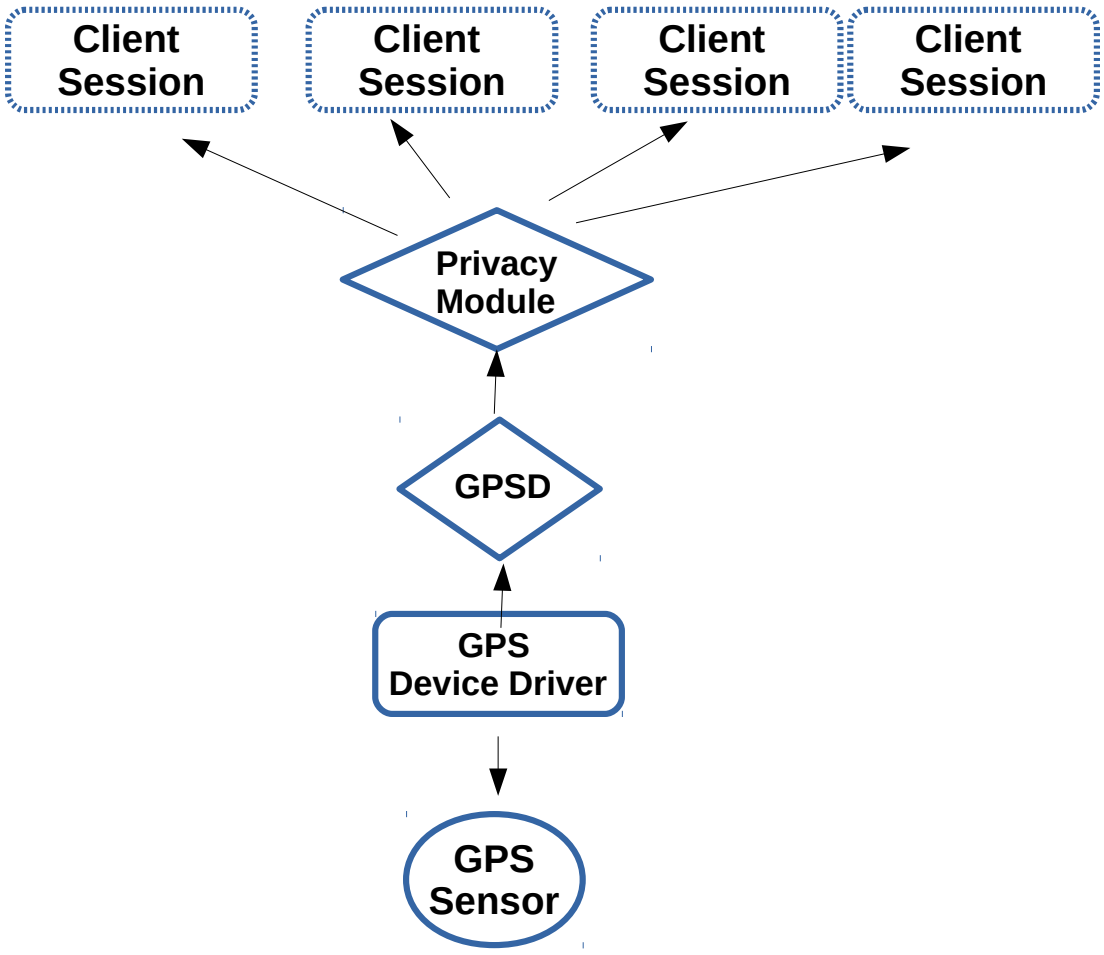

Figure 9.1: Privatization occurs before data is released to the client application.

privilege [VCC11].

Additionally, permission managers (e.g., Android and iOS) offer binary permissions to disable or enable location services. However, while this allows data owners to disable location services for applications that do not require location (e.g., Flashlight application) [Fla], fine grained granularity is still missing. An Android modification called CynagonMod has a module called XPrivacy [XPr]. XPrivacy enables data owners to configure random or a static location, empty cell ID, blocks geofences from being set, prevents sending NMEA data

to application, prevents cell tower updates from being sent to an application, prevents aGPS, returns empty Wi-Fi scans, and disables activity recognition. Ultimately, this provides the data owner control at the application layer.

User applications requesting data of users is a binary permission, either I share my data or I don't. However, sensitive data such as location needs finer control on how accurate and how often the location information is released. Users should be able to control the granularity of their personal data that is released. Users require freedom and control over their own personal data.

However, these approaches discards several important facts: 1) these privacy mechanisms protect at the application layer only and the underlying operating system still has access to all system location APIs 2) granular privacy permission solutions (e.g., XPrivacy) are only for rooted Android phones 3) there is no compromise between third party analyzers and data owners. The expected proliferation of IoT devices will further exacerbate these privacy issues.

In this dissertation, we present the first (to our knowledge) implementation of a privacy module to GPSD. Figure 9.1 shows an overview of the flow of queries and responses and demonstrates that the privatization occurs before releasing the data back to the application. The privacy module ensures that all GPS data is released according to the data owner's consent and choice. We demonstrate that appropriate methodologies can be placed which provides strong location privacy guarantees, yet enable analyzers access to privatized location data.

1. A privacy module that integrates into the GPSD software (runs on every GPS enabled device)

2. A granular privacy interface and control to manage location privacy settings (e.g., location coarseness and release frequency)

3. A performant privacy module with minimal overhead

We first describe the architecture and flow of GPSD, we then describe our privatization

algorithms, then we describe our integration with GPSD, and finally we evaluate our scheme.

9.1.1 System Goals

There should be well defined and enforced constraints regarding third party application's (apps) access to location data. The data owner should be able to specify the constraints such as how accurate location information should be disclosed and how frequent the location data should be disclosed.

Apps only have access to the privatized data and are unable to directly access GPSD daemon and data. All location data released must be approved by the data owner.

The system should support applications that need real-time access to location data. The privacy policy defines how frequently the application is allowed to receive updates (express in epochs), how accurate the location data may be, and geographical regions as to where the application is allowed to receive location data from.

We use a social network messaging application as an example. The application may want to know which city an individual is in, though pinpoint location information within meter accuracy is not required. The data owner is allowed to define both the radius (e.g., city) that is allowed to be returned as well as the frequency (e.g., say at most every hour).

Ultimately the data owner has final say over how location data and the tradeoff between privacy and utility. The utility has benefits for third party analysts interesting in learning aggregate behavior.

9.1.2 Performance Goals

The system should scale gracefully as the number of applications connecting to the GPSD daemon increases. Location data will be released within the defined epochs.

9.1.3 Threat Model

Mobile devices (e.g., smartphones, tablets, wearables) are under the data owner's control. Kernel and underlying OS is vetted and verified (signatures and trusted sources). Focus is not on low level system threats. We assume that the operating system itself is not malicious and provides a mechanism to provide a privacy policy settings manager accessible to the data owner. Secure micro kernels such as seL4 address these issues and are out of scope for this dissertation. Applications do not have a system exploit (e.g., rootkit) to circumvent the system.

Applications may try to request data more frequently than the defined epoch. LocationSafe will deny such aggressive requests and ensure that data is only released within the defined epoch.

Applications may act as sybils and send false application IDs in order to confuse the GPSD daemon. LocationSafe will treat sybil applications accordingly using data owner defined defaults. Thus, sybil applications may either be receive location data using default privacy configurations or not at all.

9.1.4 Privacy Goals

Data owners should be able to limit how frequently an application access location data. Data owners should also be able to define fine-grained access to location data. Applications for which the data owner feels the application does not meter level accuracy, the data owner should be allowed to define a radius from which the location value can be returned from. Additionally, for scenarios where fine-grained location is required, the data owner can define a grid system from which potential locations can be returned from.

GPS sensor data is only accessible via GPSD.

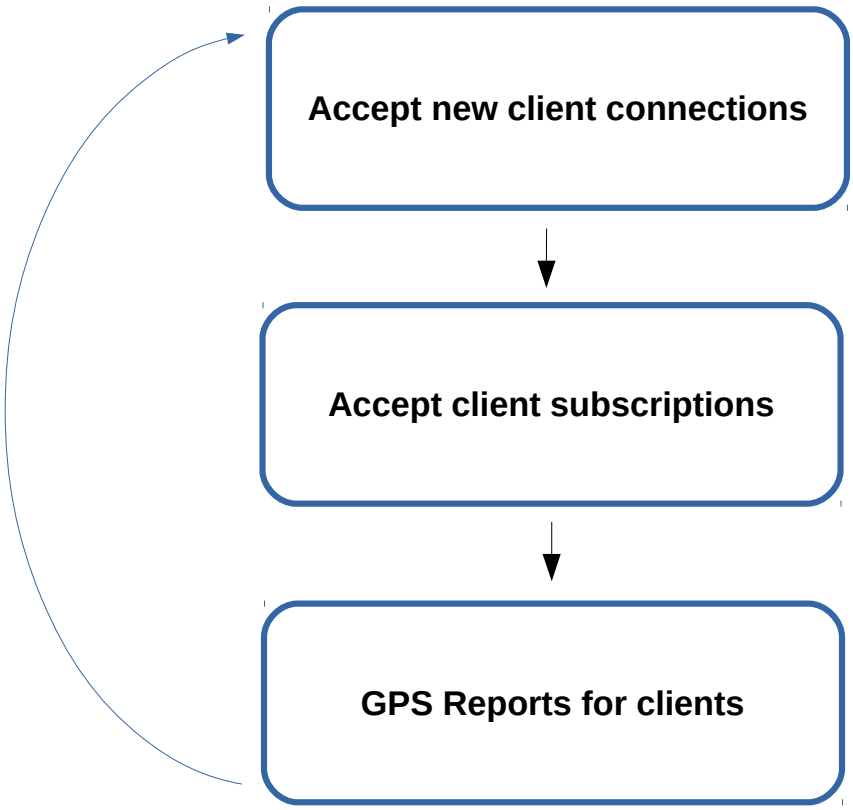

Figure 9.2: GPSD event loop. Privatization occurs when reporting GPS data to the client.

9.1.5 Architecture

Figure 9.2 depicts the main components GPSD event loop: accepting new client connections, accepting new client subscriptions, and GPS reporting to all subscribed clients. Each client connecting passes in an application identifier which is mapped to a privacy configuration managed by the system. The privacy configuration contains the epoch (how often data is released in milliseconds), differential private ϵ, privatization radius in meters, and the

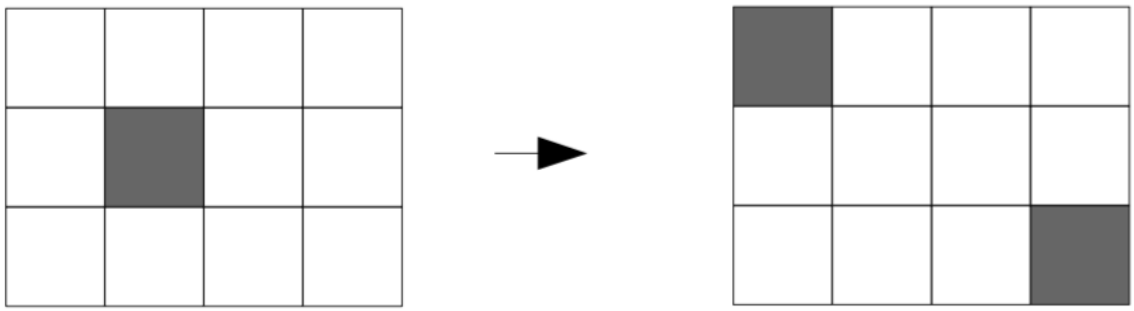

Figure 9.3: In the grid privatization a single location may randomize to one or many locations. In the example above two locations are returned. However, in the aggregate the analyst is able to estimate the underlying population value without violating individual privacy.

randomized response coin flips. Clients are allowed to pass in a recommended set of privacy parameters, though these are checked against user settings and are not allowed to exceed the privacy threshold defined by the user. In such cases user settings are adhered to.

9.1.6 Privatization

LocationSafe currently supports two modes of privatization: radius privacy and grid privacy via differential privacy.

The first mechanism is via radius privacy whereby the data owner can specify a radius cover wherein a random point within the defined range is chosen. This approach favors strong privacy at the expense of utility. That is a larger radius grants more privacy though limits the location accuracy.

The second privacy mechanism represents the location space as a grid. The grid can be sized according to the data owner's specification. The current location is placed within the grid. Then leveraging the randomized response method one or many grid locations are

returned as seen in Figure 9.3.

Randomized response [War65] was originally created by social scientists as a mechanism to perform a population study over sensitive attributes (such as drug use or certain ethical behaviors). Randomized response allows data owners to locally randomize their truthful answer to analyts' sensitive queries and respond only with the privatized (locally randomized) answer. We utilize randomized response as our privacy mechanism as randomized response satisfies the differential privacy guarantee for individual data owners, it provides the optimal sample complexity for local differential privacy mechanisms [DWJ13], and it easily suitable for the location grid type answers we provide.

9.1.6.1 Mechanism Description

We will now describe how each data owner privatizes their response utilizing the randomized response mechanism. Suppose each data owner has two independently biased coins. Let the first coin flip heads with probability p, and the second coin flip heads with probability q. Without loss of generality, in this dissertation, heads is represented as "yes" (i.e., 1), and tails is represented as "no" (i.e., 0).

Each data owner flips the first coin. If it comes up heads, the data owner responds truthfully; otherwise, the data owner flips the second coin and reports the result of this second coin flip.

Suppose there are N data owners participating in the population study. Let $\hat{Y}$ represent the total aggregate of "yes" randomized answers. The estimated population with the sensitive attribute Y_A can be computed as:

$$Y_A = \frac{\hat{Y} - (1-p) \times q \times N}{p} \tag{9.1}$$

The intuition behind randomized response is that it provides "plausible deniability", i.e., any truthful answer can produce a response either "yes" or "no", and data owners retain strong deniability for any answers they respond. If the first coin always comes up heads, there is high utility yet no privacy. Conversely, if the first coin is always tails, there is low

utility though strong privacy. It has been shown that by carefully controlling the bias of the two coin flips, one can strike a balance between utility and privacy (Table 4 in [FT86] and Table I in [JRG16]).

9.1.6.2 Multiple Sensitive Attributes

While randomized response is an intuitive privacy mechanism for a single location, naturally the question becomes how does one deal with multiple locations, i.e., a grid representation? A host of "polychotomous" mechanisms have been studied and surveyed in the literature [FT86] using multiple randomizing mechanisms or maximum likelihood estimators [Tam81]. However, it turns out that simply repeating an application of [FT86] for each grid location turns out to be an "optimal" [Tam81] approach.

Thus, LocationSafe repeats the randomized response mechanism for each grid location. For example, if a traffic analyst wishes to understand the traffic flow of a few key locations, the traffic analyst issues a query that is a Boolean bit-vector asking each data owner to indicate the location they are at. Then, each data owner performs randomized response for each location and replies with a Boolean bit-vector. The traffic analyst then aggregates and sums the bit-vectors to calculate the number of vehicles at each location.

9.2 Non-Attributable Writes

We now describe our order of magnitude scalability enhancement for the Function Secret Sharing (FSS) [BGI15] primitiv over the default implementation. Function secret sharing privacy properties hold as long as there is, among multiple (decentralized) database operators, at least an honest one that does not collude.

9.2.1 Optimization

Function Secret Sharing (FSS) relies on symmetric cryptography. Thus, we utilize AES in counter mode for the pseudorandom generator.

Our main observation is that the default FSS evaluation algorithm repeatedly evaluates the same seeds multiple times. The difference for each evaluation is the randomized bits *position* for the seed expansion. Thus, we can simply evaluate each seed *once* and iterate over the expanded seed randomized bits.

Our optimization is for the share evaluation as shown in Algorithm 2 and Algorithm 3. The share generation in Algorithm 1 is the same as the default implementation [BGI15].

The default implementation share evaluation performs 2^n PRG seed initializations. However, the full PRG evaluation is the same for each value of $\gamma\prime$. Thus, we need to perform ν PRG seed initializations instead of repeating the same PRG evaluation. We do *one* evaluate per δ, which means that we can reuse the same evaluated output and just take different partitions for varying γ. The default FSS version evaluates δ times the same seeds in order to extract the differing γ sections.

Algorithm 1 $\text{Gen}^{p_i}(1^\lambda, x, y)$

1: Let $G : \{0,1\}^\lambda \longrightarrow \{0,1\}^{m\mu}$ be a PRG

2: Let $\mu \leftarrow \lceil 2^{n/2} \times 2^{p-1/2} \rceil$. Let $\nu \leftarrow \lceil 2^n/\mu \rceil$

3: Use the higher and lower bits of the input x as a pair $x = (\gamma\prime, \delta\prime)$, $\gamma\prime \in \lceil \nu \rceil$ $\delta\prime \in \lceil \mu \rceil$

4: Choose ν arrays $A_1, ..., A_\nu$ s.t. $A_\gamma \in_R O_p$ and $A_{\gamma\prime} \in_R E_p$ for all $\gamma\prime \neq \gamma$

5: Choose 2^{p-1} random strings $cw_1, ..., cw_{2^{p-1}} \in 0, 1^{m\mu}$ s.t. $\bigoplus_{j=1}^{2^{p-1}} (cw_j \oplus G(s_{\gamma,j})) = e_\delta \cdot b$

6: Set $\sigma_{i,\gamma\prime} \leftarrow (s_{\gamma\prime,1} \cdot A_{\gamma\prime}[i,1]) \parallel ... \parallel (s_{\gamma\prime,2^{p-1}} \cdot A_{\gamma\prime}[i, 2^{p-1}])$ for all $1 \leq i \leq p$, $1 \leq \gamma\prime \leq \nu$.

7: Set $\sigma_i = \sigma_{i,1} \parallel ... \parallel ...\sigma_{i,\nu}$ for $1 \leq i \leq$ p

8: Let $k_i = (\sigma_i \parallel cw_1 \parallel ... \parallel cw_{2^{p-1}})$ for $1 \leq i \leq$ p

9: Return $(k_i, ..., k_p)$

Algorithm 2 EvaluateShare$^{p_i}(k_i)$

Let $\mu \leftarrow \lceil 2^{n/2} \times 2^{p-1/2} \rceil$. Let $\nu \leftarrow \lceil 2^n/\mu \rceil$

Use the higher and lower bits of the input x as a pair $x = (\gamma\prime, \delta\prime)$, $\gamma\prime \in \lceil \nu \rceil$ $\delta\prime \in \lceil \mu \rceil$

for $j = 1, ..., \nu$ **do**

    $y_j \leftarrow \text{Eval}(j, k_i)$

    Let $result_j \leftarrow (y_j[1] \parallel ... \parallel y_j[\mu])$

end for

Return $(result_1, ..., result_\nu)$

Algorithm 3 Eval$^{p_i}(\nu\prime, k_i)$

Let $G : \{0,1\}^\lambda \longrightarrow \{0,1\}^{m\mu}$ be a PRG

Parse k_i as $k_i = (\sigma_i, cw_i, ..., cw_{2^{p-1}})$

Parse σ_i as $\sigma_i = s_{1,1} \parallel ... \parallel s_{1,2^{p-1}} \parallel ... \parallel s_{\nu,2^{p-1}}$

Let $y_i \leftarrow \bigoplus_{1 \le j \le 2^{p-1}} (cw_j \oplus G(s_{\gamma\prime,j}))$ where $s_{\gamma\prime,j} \neq 0$

Return y_i

9.3 Mechanism

9.3.1 Discretization

We illustrate the scheme using location coordinate data, although the discretization scheme can be employed for all real valued data. Suppose a data owner currently on London Bridge participates in the protocol. First, the location is discretized to a location identifier (ID) as seen in Figure 9.4. For example, using a 16 bit identifier provides 65,536 possible locations, which covers a 64 x 64 mile square with 0.25 mile sections for a total of 4,096 square miles. For comparison Paris is 41 square miles, London is 607 square miles, New York City is 305 square miles [Lis17]. In Figure 9.4, London Bridge corresponds to location ID 8.

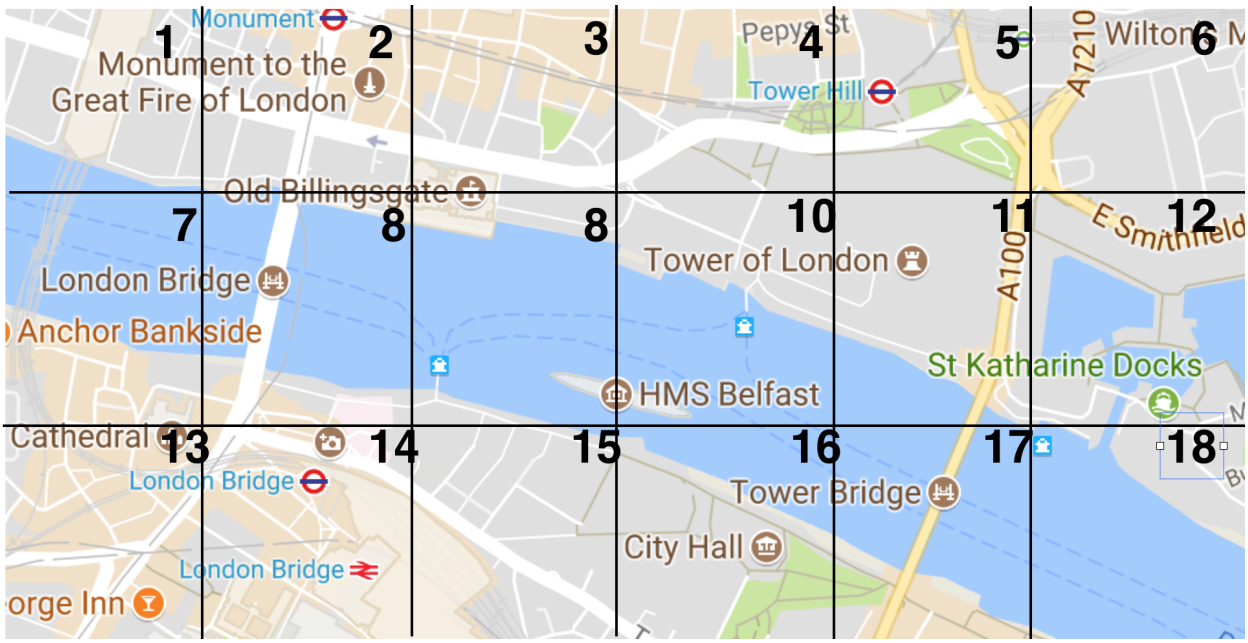

Figure 9.4: **Location Discretization.** Each location (latitude,longitude) is discretized to a location identifier which corresponds to a 0.25 square mile block. London Bridge corresponds to location ID 8.

9.3.2 Sampling Error

We now introduce our K Privacy mechanism that improves in privacy strength via query expansion yet preserving utility. We first examine how the Randomized Response [War65, FT86] mechanism grows in error as the No_{pop} increases. We formally describe the Randomized Response mechanism and then describe how the sampling error increases with the population.

(Randomized Response) We use two independent and biased coins. Let π_1 and π_2 refer to the heads probabilities of the first and second biased coin toss respectively. The coin toss parameters are published publicly while the number of data owners is private and needs to

be estimated.

$$Privatized\ Value_{Yes} = \begin{cases} 1 & \text{with probability} \\ & \pi_1 + (1-\pi_1) \times \pi_2 \\ 0 & \text{otherwise} \end{cases} \tag{9.2}$$

That is, the Yes_{pop} subpopulation responds "Yes" with probability $\pi_1 + (1-\pi_1) \times \pi_2$. Otherwise they respond "No".

$$Privatized\ Value_{No} = \begin{cases} 1 & \text{with probability} \\ & (1-\pi_1) \times \pi_2 \\ 0 & \text{otherwise} \end{cases} \tag{9.3}$$

That is, the No_{pop} subpopulation responds "Yes" with probability $(1-\pi_1)\times\pi_2$. Otherwise they respond "No".

(Expected Value) We now formulate the expected value in order to carry out the estimation of the underlying population. The expected value of those that respond '1' (i.e., privatized "Yes") is the sum of the binomial distribution of each subpopulation.

$$E[1] = \pi_1 \times Yes_{pop} + (1 - \pi_1) \times \pi_2 \times (Yes_{pop} + No_{pop}) \tag{9.4}$$

(Estimator) We solve for Yes_{pop} by the following. Let the aggregated privatized count E[1] defined in Equation 9.4 be denoted as $Private\ Sum$.

$$Yes_{pop} = \frac{Private\ Sum - (1-\pi_1) \times \pi_2 \times (Yes_{pop} + No_{pop})}{\pi_1} \tag{9.5}$$

That is, we first subtract from Private Sum of the "privacy noise". We then divide by the first flip π_1 which is the sampling parameter which determines how frequently a data owner truthfully responds "Yes" from the Yes_{pop} subpopulation.

(Sampling Error) Suppose published parameters of the coin tosses are configured independently with $\pi_1 = 0.85$, $\pi_2 = 0.3$ and 100 data owners. We estimate the underlying "Yes" truthful population using Equation 9.5 by aggregating the privatized responses from all data owners, subtracting the expected value of $(1 - 0.85) \times 0.3 \times 100$ and dividing by 0.8 [1].

However, a drawback to the randomized response mechanism is that the estimation error quickly increases with the population size due to the underlying truthful distribution distortion. For example, say we are interested in how many vehicles are at a popular stretch of the highway. Say we configure $\pi_1 = 0.85$ and $\pi_2 = 0.3$. We query 10,000 vehicles asking for their current location and only 100 vehicles are at the particular area we are interested in (i.e., 1% of the population truthfully responds "Yes"). The standard deviation due to the privacy noise will be 21 [2] which is slightly tolerable. However, a query over one million vehicles (now only 0.01% of the population truthfully responds "Yes") will incur a standard deviation of 212. The estimate of the ground truth (100) will incur a *large* absolute error when the aggregated privatized responses are two or even three standard deviations (i.e., 95% or 99% of the time) away from the expected value, as the mechanism subtracts *only* the expected value of the noise.

We desire better calibration over the privacy mechanism and a mechanism which maintains constant error as the No_{pop} population scales up. We introduce the K Privacy mechanism in the next section. Though first, we examine how to ensure that data owners are able to privately upload their responses.

9.3.3 K Privacy Mechanism

We now describe our K Privacy mechanism that achieves constant error even where the population which does not truthfully respond "Yes" (No_{pop}) increases. We illustrate the

[1] For instance with a 60% truthful population, the answer to the first toss is $0.6 \times 0.85 = 0.51$ and the answer to the second toss is $(1 - 0.85) \times 0.3 = 0.045$

[2] $(\sqrt{(1 - 0.85) \times 0.3 \times 10,000})$

scheme using location coordinate data, although the scheme can be employed for all real valued data.

Illustration. To illustrate and demonstrate the K Privacy mechanism, we employ the following example. Suppose we are interested in the distribution of vehicles across London. We query every vehicle in London asking for their location coordinates. Each data owner responds to a binary version of the binary query such as "Are you at the London Bridge?". The mechanism has two rounds and proceeds as follows.

Suppose a particular data owner is at London Bridge. First, the location is discretized to a location identifier (ID) as described in Section 9.3.1. In this case the location ID is 8 as shown in Figure 9.4.

Next, the data owner tosses a multi-sided die. One side samples whether the data owner should respond truthfully for their location ID. The remaining sides selects a location ID for the data owner to respond.

Suppose in the first round the data owner is sampled and selected. The data owner should respond "Yes" (they are at London Bridge).

In the second round the sampled data owner participates to maintain the crowd size though writes a nil value.

A privatized sum is computed by aggregating the "Yes" counts in each round.

Finally, estimation occurs by subtracting the privatized sum in round one from round two and dividing by the sampling parameter.

The following three privacy observations are made. First, a majority of the population provides privacy noise by randomly responding either "Yes" or "No" regardless of their truthful response. Second, plausible deniability is provided as each data owner probabilistically responds opposite of their truthful response. Finally, *every* data owner acts as a potential candidate for the truthful population. Our assumption is that every data owner is active in both rounds and only the aggregate counts are released.

9.3.3.1 Binary Value

We now formally introduce the binary value K Privacy mechanism whereby a data owner responds either "No" or "Yes", either 0 or 1 respectively.

(Round One) In the first round each data owner tosses a three sided die with probabilities π_s, π_{Yes}, and π_{No}. Let π_s be the probability that a data owner truthfully responds. Otherwise, regardless of their truthful response let π_{Yes} be the probability that a data owner randomly responds "Yes" and π_{No} be the probability that a data owner randomly responds "No".

$$\textit{Round One}_{Yes} = \begin{cases} \mathbf{1} & \textbf{with probability } \pi_s \\ 1 & \text{with probability } \pi_{Yes} \\ 0 & \text{with probability } \pi_{No} \end{cases} \tag{9.6}$$

$$\textit{Round One}_{No} = \begin{cases} 1 & \text{with probability } \pi_{Yes} \\ \mathbf{0} & \textbf{with probability } \pi_s \\ 0 & \text{with probability } \pi_{No} \end{cases} \tag{9.7}$$

At this point, privacy noise has been added and thus the underlying truthful distribution is becoming distorted as the number of non-truthful data owners participate. The distortion makes it difficult to estimate the the underlying truthful distribution as we have one equation and two variables (number of truthful and non-truthful data owners).

Thus, we execute a second round while fixing the two variables enabling us to solve for the truthful population estimate.

(Round Two) In the second round only the data owner which was selected and sampled with probability π_s writes a nil value, though participates to maintain the crowd size. The

remaining data owners stay with the responses from round one.

$$Round\ Two = \begin{cases} \varnothing & \textbf{with probability } \pi_s \\ 1 & \text{with probability } \pi_{Yes} \\ 0 & \text{with probability } \pi_{No} \end{cases} \tag{9.8}$$

Now combining the second round with the first round we obtain accurate estimations as we see below.

(Expected Values) We now formulate the expected values as follows. The subscript refers to the round number. That is, 1_1 refers to output 1 and round 1. The first round of expected values are:

$$\begin{aligned} E[1_1] &= \pi_{Yes} \times TOTAL_{pop} + \pi_s \times Yes_{pop} \\ E[0_1] &= \pi_{No} \times TOTAL_{pop} + \pi_s \times No_{pop} \end{aligned} \tag{9.9}$$

That is, both the Yes_{pop} and No_{pop} contribute both "Yes" and "No" responses while a small subpopulation responds truthfully.

The second round of expected values are:

$$\begin{aligned} E[1_2] &= \pi_{Yes} \times TOTAL_{pop} \\ E[0_2] &= \pi_{No} \times TOTAL_{pop} \end{aligned} \tag{9.10}$$

That is, the small subpopulation from round 1 samples out and does not participate. The remaining data owners randomly respond "Yes" or "No" while remaining at their round one responses.

(Estimator) We solve for the Yes_{pop} population by subtracting round one by round two as follows. Let $Private\ Sum_{\text{“}Yes\text{”},1}$ refer to the aggregated privatized counts for output space "Yes" and round 1.

$$Yes_{pop} = \frac{Private\ Sum_{\text{“}Yes\text{”},1} - Private\ Sum_{\text{“}Yes\text{”},2}}{\pi_s} \tag{9.11}$$

That is, we subtract the privatized sum of output space “Yes” round 1 from the output space “Yes” of round 2. The result is the sampled “Yes” aggregate. We then obtain the estimation by dividing by the sampling parameter.

The sampling error affects only the Yes_{pop} as seen in Equation 9.9. Thus we are able to scale the No_{pop} yet retain constant error. Plausible deniability is provided as each data owner may respond to either output space based on the coin toss parameters.

9.3.3.2 Differential Privacy Guarantee

K Privacy satisfies differential privacy as we show in this section. We first examine the binary value mechanism and then the multiple value mechanism.

(Binary Value) The differential privacy leakage is measured as the maximum ratio of the binary output space given the underlying truthful answer is “Yes” and “No” respectively.

In round one, the output space “Yes” is slightly more likely as the truthful response is sampled in addition to being responded randomly. In round two, there is no privacy leakage as both output space “Yes” and “No” are both equally likely and indistinguishable given the truthful answer is either “Yes” or “No” respectively.

Thus, we are interested in the privacy leakage of output 1 round 1 (1_1) as follows:

$$\max\left(\ln\left(\frac{\Pr[Response = Yes|\text{“}Yes\text{”}]}{\Pr[Response = Yes|\text{“}No\text{”}]}\right), \ln\left(\frac{\Pr[Response = No|\text{“}No\text{”}]}{\Pr[Response = No|\text{“}Yes\text{”}]}\right)\right) \tag{9.12}$$

$$\epsilon_{DP} = \max\left(\ln\left(\frac{\Pr[1_1|\text{“}Yes\text{”}]}{\Pr[1_1|\text{“}No\text{”}]}\right), \ln\left(\frac{\Pr[1_1|\text{“}No\text{”}]}{\Pr[1_1|\text{“}Yes\text{”}]}\right)\right) \tag{9.13}$$

$$\frac{\Pr[1_1|\text{“}Yes\text{”}]}{\Pr[1_1|\text{“}No\text{”}]} = \frac{\pi_{V'} + \pi_s}{\pi_{V'}} \tag{9.14}$$

$$\frac{\Pr[1_1|\text{“}No\text{”}]}{\Pr[1_1|\text{“}Yes\text{”}]} = \frac{\pi_{V'}}{\pi_{V'} + \pi_s} \tag{9.15}$$

$$\epsilon_{DP} = \max\left(\ln\left(\frac{\pi_{V'} + \pi_s}{\pi_{V'}}\right), \ln\left(\frac{\pi_{V'}}{\pi_{V'} + \pi_s}\right)\right) \tag{9.16}$$

CHAPTER 10

Evaluation II: Scalable Privacy

10.1 LocationSafe

To evaluate the overhead of the addition of the privacy module to GPSD, we run 25 and 64 clients connecting to GPSD with varying epochs of 5,10,15 seconds as seen in Table 10.1. The evaluation was run on a laptop running Archlinux release 2016.06.01 kernel 4.5.4 with two i5 physical cores (four logical) and 12gb ram. GPSD by default has a limit of 64 clients so we stay within this bound.

The results show that minimal overhead is incurred by the privacy module and that clients are able to reasonably receive location updates within the allotted epoch. Even as more clients connect the performance guarantees do not degrade.

		Epoch (seconds)		
		5	10	15
# Clients	25	7	12	16
	64	6	11	14

Table 10.1: Scaling performance of clients receiving a response in specified epoch. Values are averaged across ten iterations.

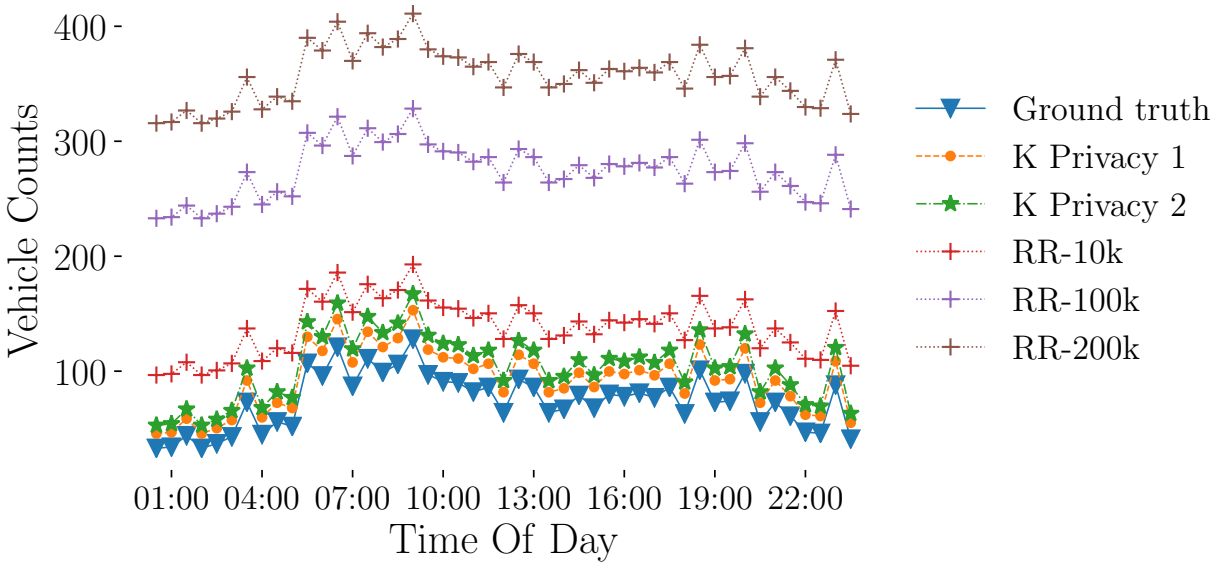

Figure 10.1: **(Vehicle counts)** K Privacy Each vehicle reports it's current location.

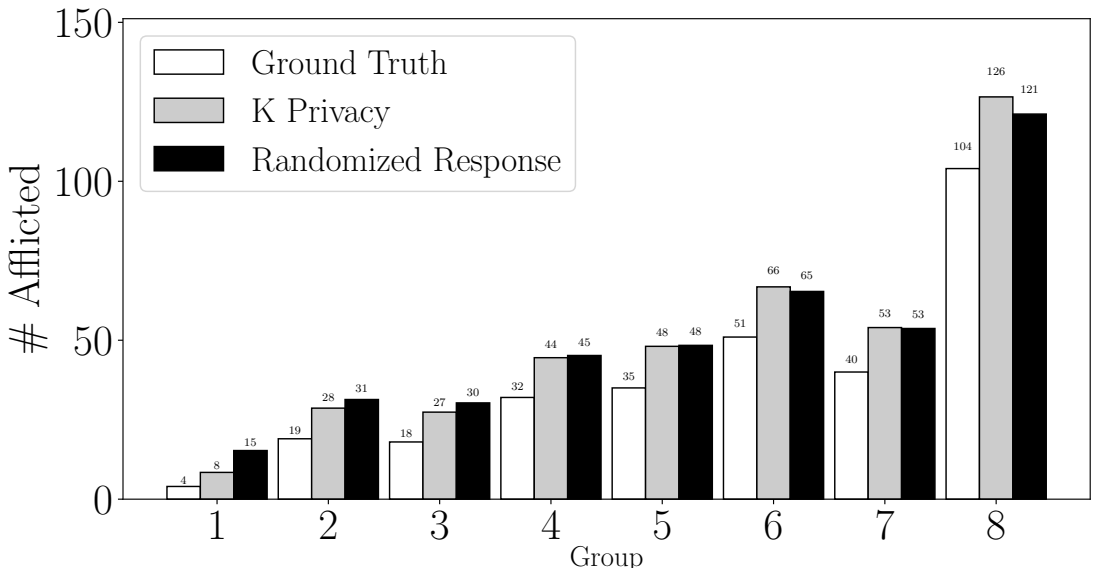

Figure 10.2: **(Heart Chest Pain)** Number of individuals out of 303 with specific types of heart related chest pain.

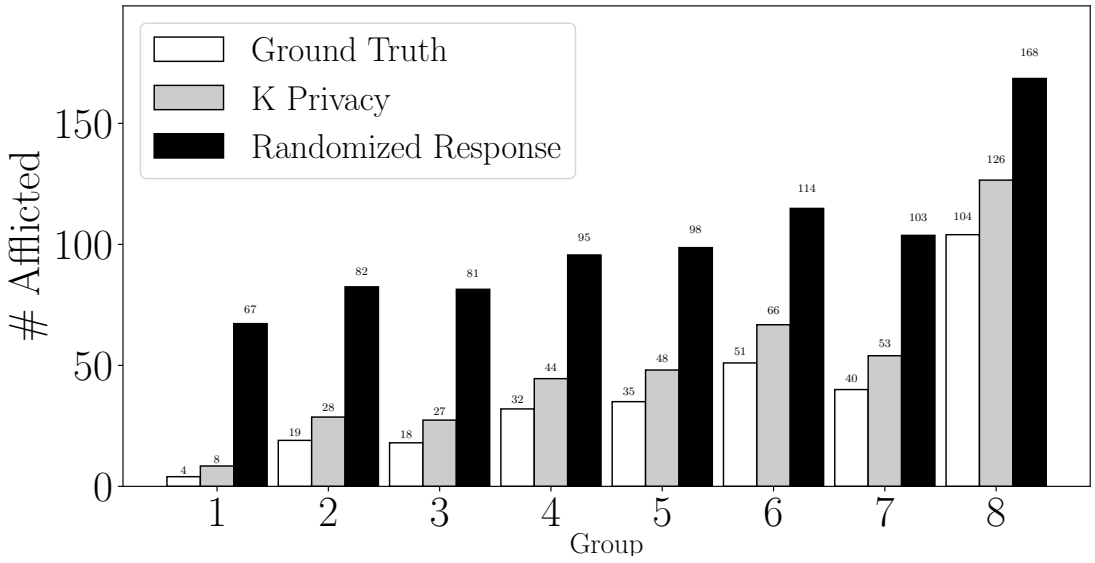

Figure 10.3: **(Heart Chest Pain)** Number of individuals out of 10,000 with specific types of heart related chest pain.

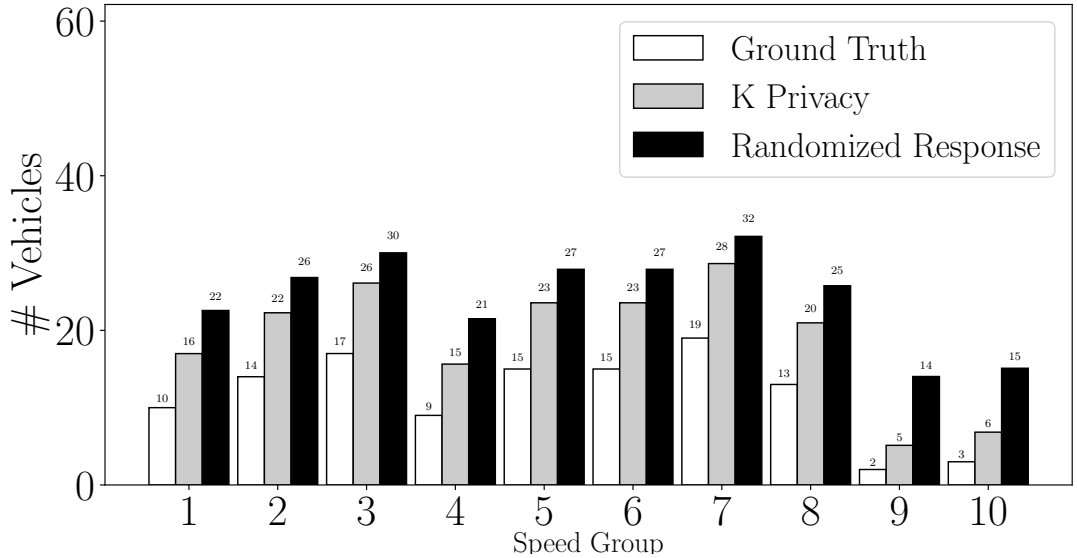

Figure 10.4: **(Vehicle Speed Distribution)** Lane 1 speed distribution.

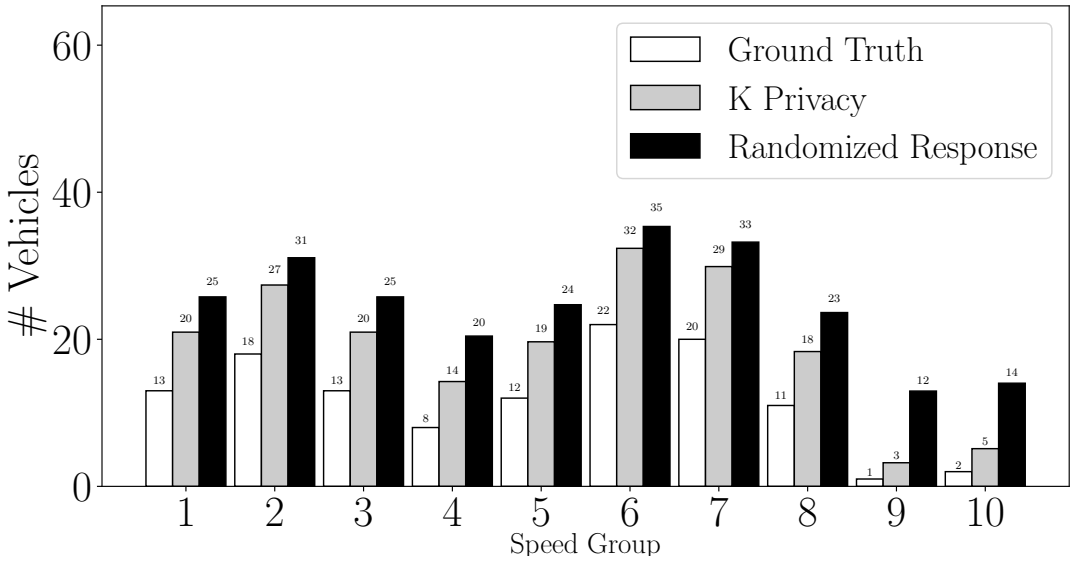

Figure 10.5: **(Vehicle Speed Distribution)** Lane 2 speed distribution.

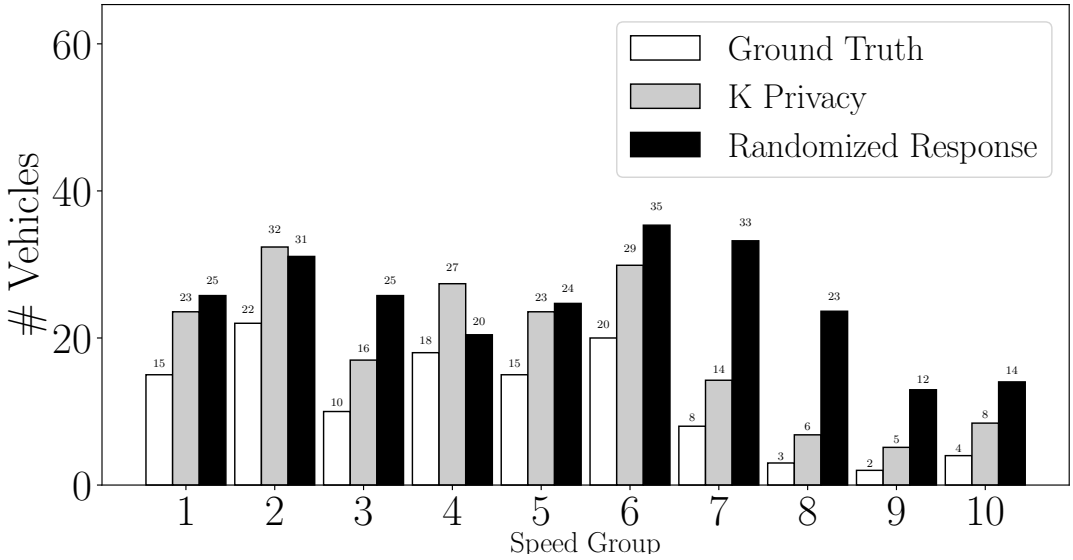

Figure 10.6: **(Vehicle Speed Distribution)** Lane 3 speed distribution.

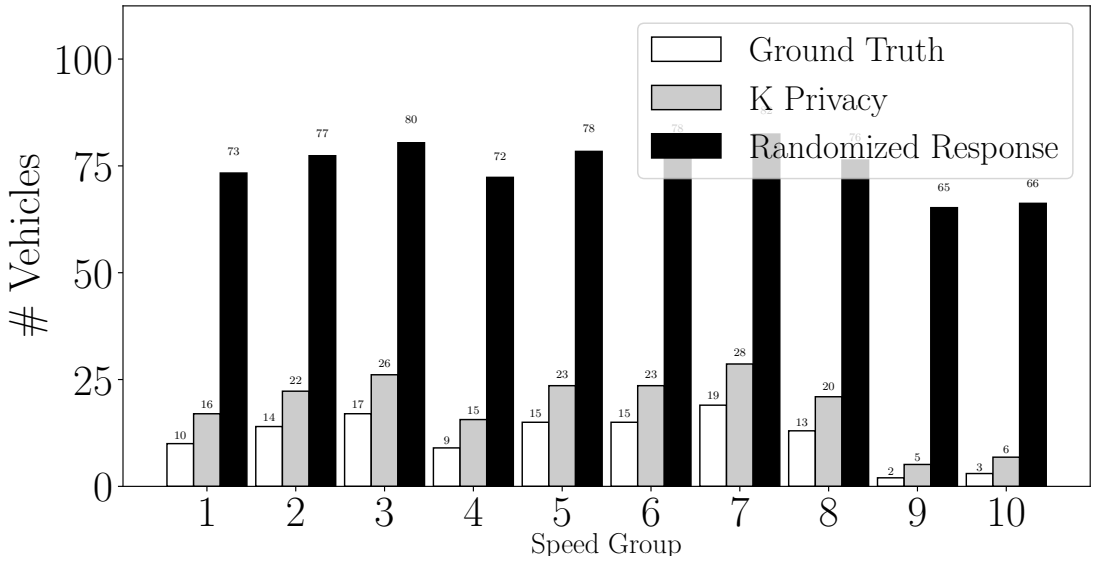

Figure 10.7: **(Vehicle Speed Distribution)** Lane 1 speed distribution over 10,000 vehicles (only 354 are currently amongst the queried 3 lanes).

10.2 K Privacy Mechanism

10.2.1 Accuracy

(PeMS Data) We evaluate the Haystack mechanism over a real dataset rather than with arbitrary distributions. We utilize the California Transportation Dataset from magnetic pavement sensors[Cal17a] collected in LA\Ventura California freeways [Cal17b]. There are a total of 3,865 stations and 999,359 vehicles total. We assign virtual identities to each vehicle. Each vehicle announces the station it is currently at. We select a single popular highway station. Every vehicle at the station reports "Yes" while every other vehicle in the population truthfully reports "No". We evaluate over a 24 hour time period. K Privacy 1 has a sampling parameter of 45% and K Privacy 2 has a sampling parameter of 25%. The randomized response mechanism has $\pi_1 = 0.8$ and $\pi_2 = 0.2$.

Figure 10.1 compares the K Privacy mechanism with the Randomized Response mech-

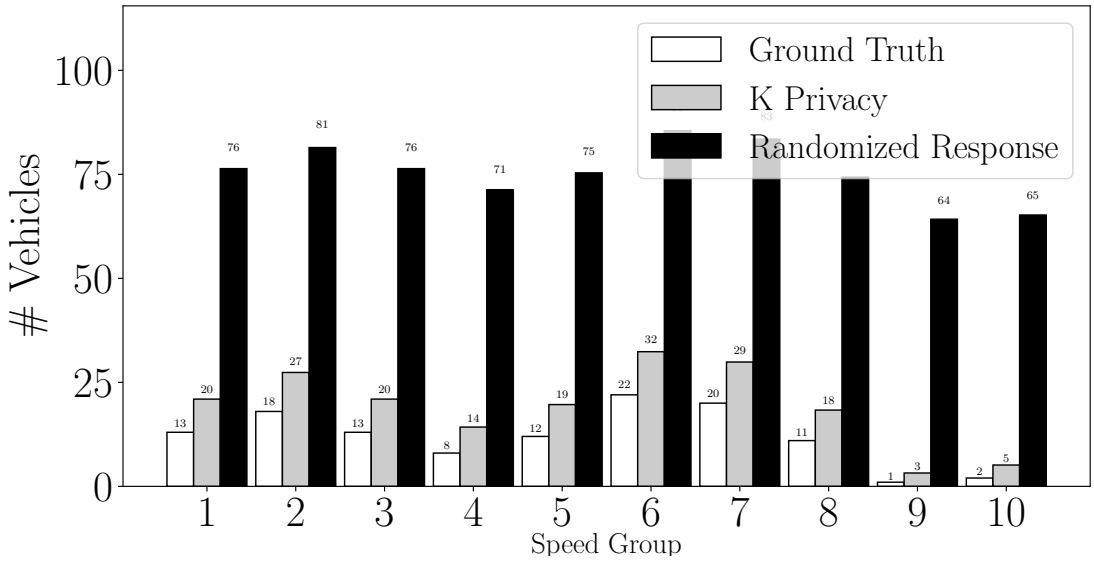

Figure 10.8: **(Vehicle Speed Distribution)** Lane 2 speed distribution over 10,000 vehicles (only 354 are currently amongst the queried 3 lanes).

anism. K Privacy is able to maintain constant error even at 1 million vehicles, while the Randomized Response quickly incurs error. Upper bounds are shown with a 95% confidence interval.

We next examine the vehicle speed distribution across the freeways at evening rush hour. Figures 10.4, 10.5, 10.6 are with the population at the specific stretch of the freeway. Figures 10.7, 10.8, 10.9 expand the query population to 10,000 vehicles (9,646) are not at the particular freeway stretch being monitored. The figures show the speed distribution whereby there are 10 groups for the following speeds "$1-10$" is group 1, "$11-20$" is group 2, etc. Upper bounds are shown with a 95% confidence interval.

(Heart Data) We next evaluate over medical data. We utilize the UCI open data repository [Lic13] for heart related data. Figure 10.2 and Figure 10.3 show the number of afflicted data owners with a particular type of chest pain. The four types of chest pain are typical angina, atypical angina, non-anginal pain, and asymptomatic. Each group corresponds to a

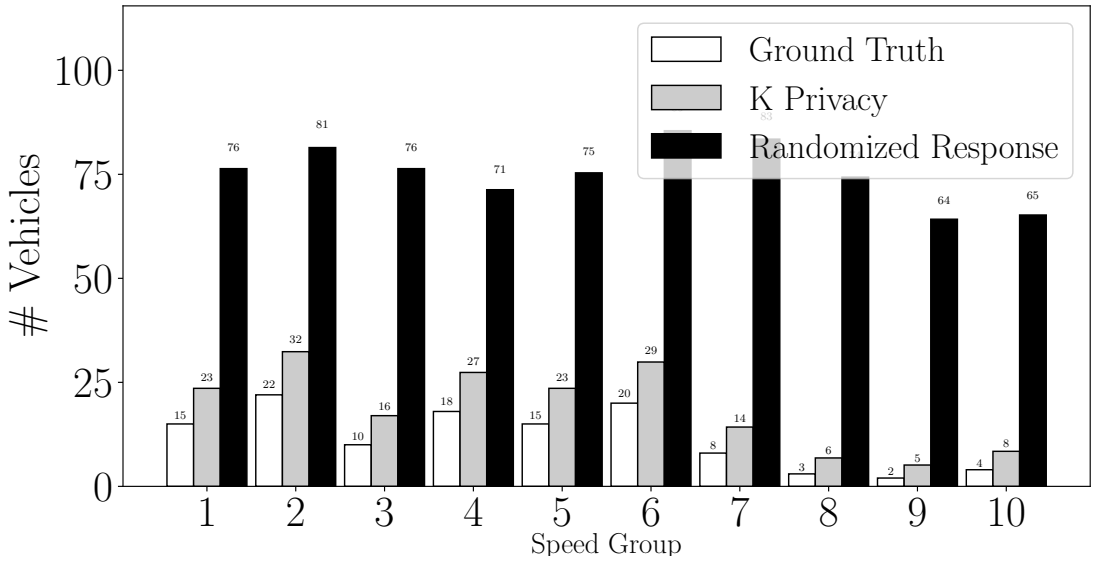

Figure 10.9: **(Vehicle Speed Distribution)** Lane 3 speed distribution over 10,000 vehicles (only 354 are currently amongst the queried 3 lanes).

particular chest pain and gender for a total of eight groups. Figure 10.3 scales the population to 10,000 whereby 303 are the original dataset and the remaining 9,697 data owners provide chaff. The K Privacy mechanism maintains constant error and the randomized response quickly incurs error. Upper bounds are shown with a 95% confidence interval.

Figure 10.13 evaluates the privacy leakage comparing K Privacy and the Randomized Response mechanism. K Privacy uses the equation defined in 9.3.3.2 to measure the privacy leakage. The Randomized Response mechanism privacy leakage is defined in the Appendix 8.1.1.

The coin toss parameters used in Figure 10.13 has Randomized Response $flip1 = 0.8$ and $flip2 = 0.2$. K Privacy has a sampling parameter of 0.45. We could increase the value of the randomized response $flip2$ though the absolute error would grow even larger than show in Figure 10.1.

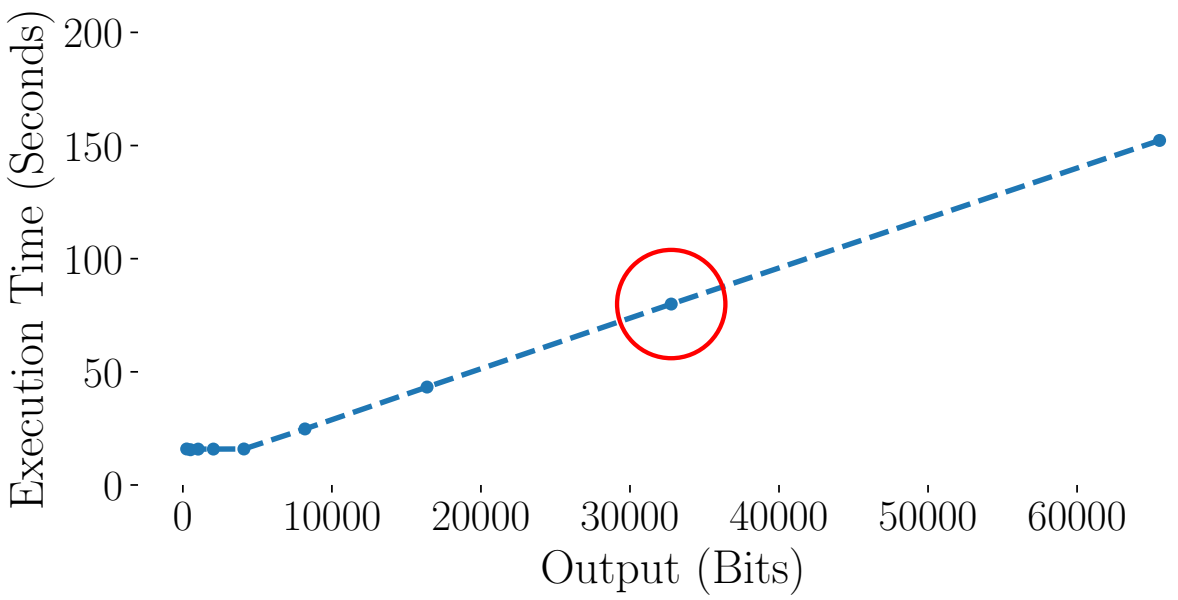

Figure 10.10: **(FSS Microbenchmark)** FSS Microbenchmark. The red circle indicates the default parameter for the number of AES initializations.

10.3 Non-Attributable Writes

FSS Optimization We evaluate the FSS optimizations on Amazon EC2 with c4.2xlarge instances which costs 0.398 per Hour. Figure 10.10 shows a microbenchmark of adjusting the the length of the output. Figure 10.11 shows the effect of the FSS optimization on the time to generate shares. The optimization is a trade-off that incurs a cost to the share generation to a speedup of the evaluation of the share. Figure 10.12 shows the effect of applying the FSS optimization for the evaluation of the shares as described in Section 9.2.1.

Running the 3 party protocol for a cluster of size 3 yields 25.77 writes per second. We would need 83 such clusters at a cost of less then $2 dollars a minute.

Share Verification. We now discuss the evaluation of our implementation of the FSS share verification [BGI16]. We microbenchmark the share blinding operations as they are the most expensive operation performed by the data owner. The three algorithms for creat-

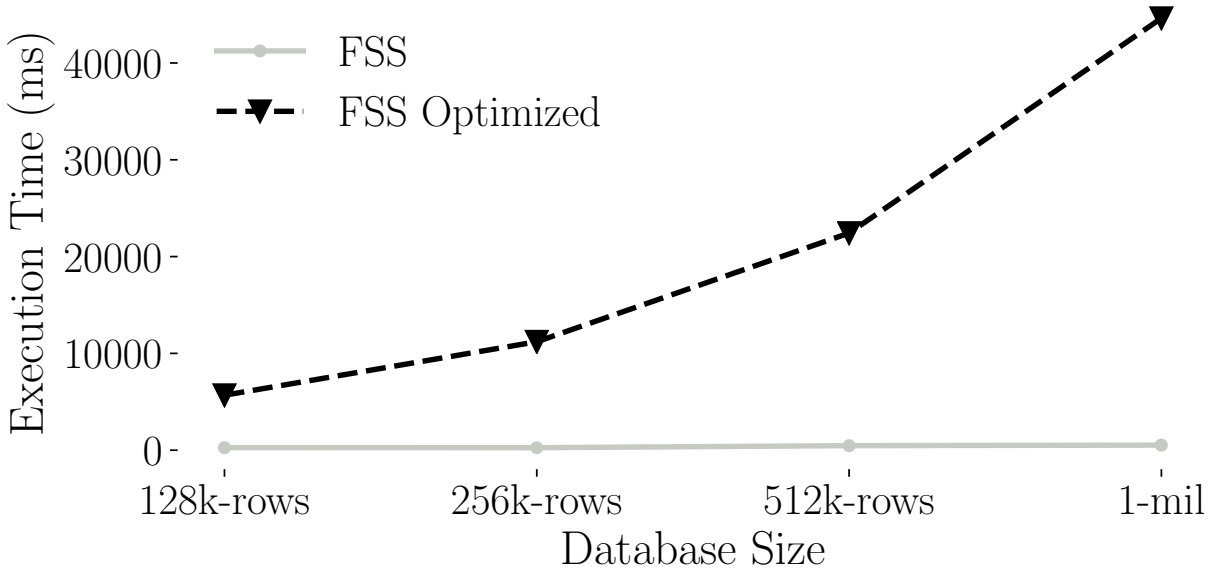

Figure 10.11: **FSS Share Generation.** Comparison of FSS versus FSS optimized parameters. Bigger is worse.

ing the blinding structure are "square", "product", and "inverse" (see Section 8.3.1 for more details). Figure 10.15 shows the scalability of the blinding operations. "Product" is slightly faster than "square" as "product" must only do$(p-1)$ multiplications, while "square" does $(p-1)$ exponent operations. "Inverse" is the slowest as it performs $(p-1)$ multiplications and then a finite field inverse, where p is the number of parties. The MPC verification performed by the aggregation servers is on the order of a couple hundred milliseconds and is extremely efficient.

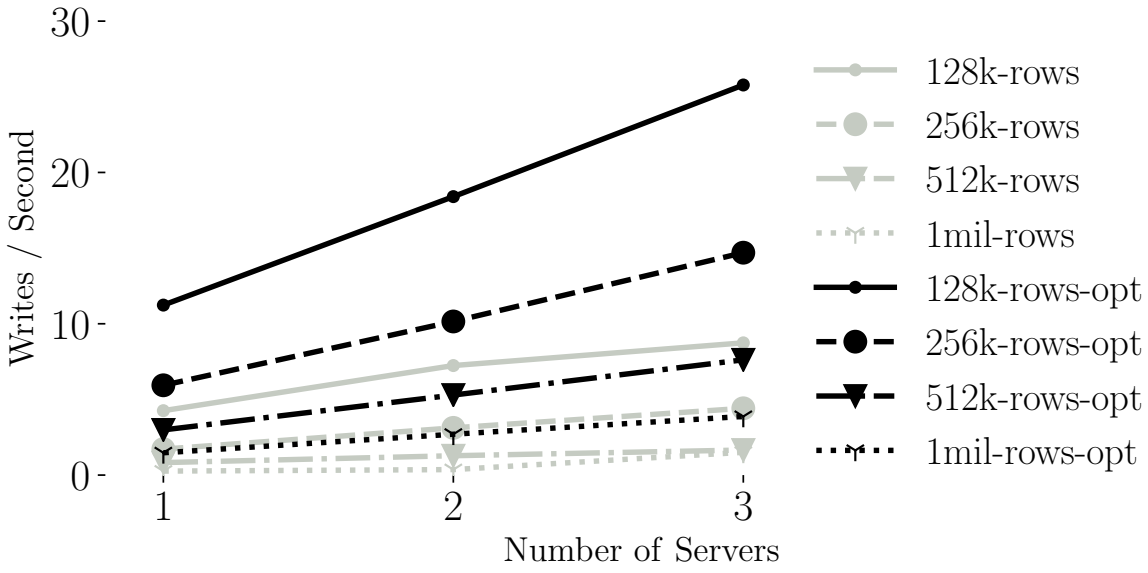

Figure 10.12: **FSS Scaling Optimization.** Comparison of FSS versus FSS optimized parameters. Bigger is better.

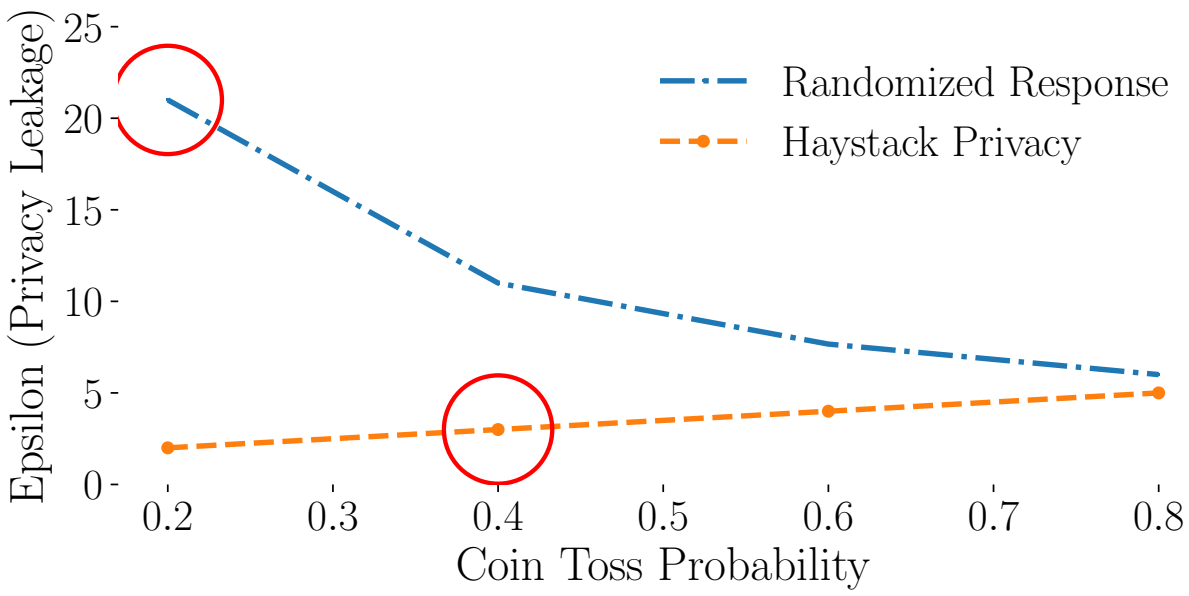

Figure 10.13: **(Privacy Leakage)** Haystack compared with Randomized Response privacy leakage as the coin toss probability increases. Higher epsilon means more information is leaked.

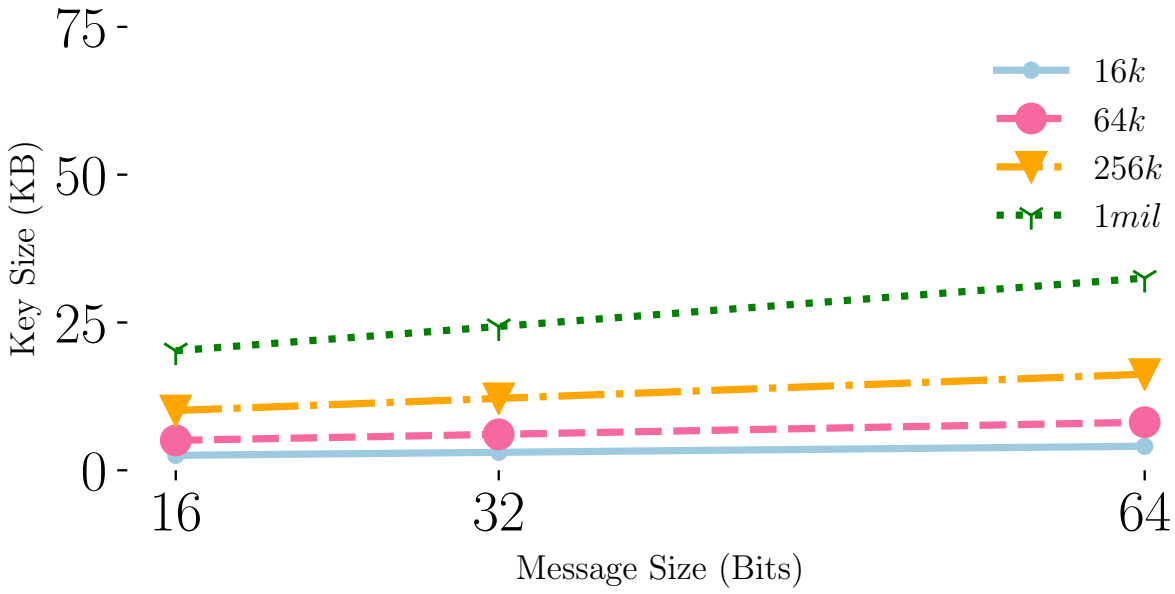

Figure 10.14: **FSS Keysize.**

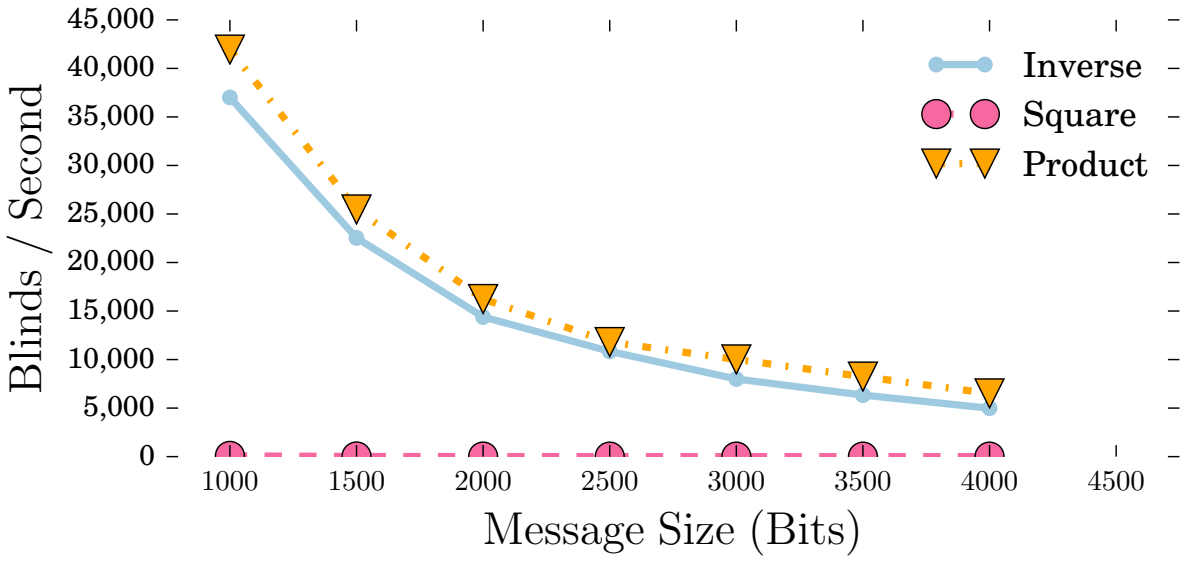

Figure 10.15: **MPC Verification Benchmark.**

CHAPTER 11

Conclusion and Future Work

In this dissertation we examined resiliency and privacy concerns in the Internet of Vehicles (IOV). The Internet of Vehicles, composed of self-driving vehicles, either isolated or in platoons, is poised to become the most prominent realization of mobile ad-hoc networks. Mobile vehicle clouds must effectively manage the wireless medium as well as protect data from privacy attacks.

To address the communication resiliency concerns, we introduced the concept of Cache Coding that is resilient to pollution attacks in Chapter 3. By leveraging coded caches mobile nodes are able to increase their file delivery throughput despite disruptive wireless communication. However, there is an energy cost associated with Cache Coding due to the increased computation complexity, as compared to no coding. Thus, we introduced an energy efficient solution to reduce energy consumption for energy constrained devices (e.g., smartphones).

We then presented our mobile architecture to achieve decentralized content confidentiality and authorization without depending on centralized always available infrastructure in Chapter 6. Role-based access control is cryptographically enforced by Attribute-Based Encryption in the content alone without requiring mobile nodes to reach back to infrastructure to verify access.

Finally, we presented a privacy mechanism that improves the privacy strength while preserving utility in Chapter 9. We implemented this privacy technique as a privacy module for the GPS daemon that runs in the majority of GPS enabled devices. We also have deployed this privacy system in the UCLA campus under the name "CrowdZen". The system is in daily use by thousands of students. In addition, we provided an order of magnitude improvement

over the default implementation for the cryptographic primitive Function Secret Sharing (FSS). FSS enables data owners to perform non-attributable writes to a distributed set of databases without the database operator learning which data owner wrote a particular value, as long as there is at least one honest database operator.

For future work, we are continuing to enhance our scalable privacy mechanism. There is still more to explore regarding the underlying definition of privacy and how to enhance privacy guarantees and strength.

In addition, the emergence of the Internet of Vehicles demands more efficient utilization of the communication channels. There will be more opportunities to leverage Cache Coding techniques to facilitate platoon style communications for large data dissemination.

[ADH08] J. Ahrenholz, C. Danilov, T.R. Henderson, and J.H. Kim. "CORE: A real-time network emulator." In *Military Communications Conference, 2008. MILCOM 2008. IEEE*, pp. 1–7, 2008.

[BBS09] Randy Baden, Adam Bender, Neil Spring, Bobby Bhattacharjee, and Daniel Starins. "Persona: An Online Social Network with User-Defined Privacy." In *SIGCOMM'09*, Barcelona, Spain, Aug. 2009.

[BCD02] Matthew Burnside, Dwaine Clarke, Srinivas Devadas, and Ronald Rivest. "Distributed SPKI/SDSI-based Security for Networks of Devices." In *Technical report, MIT Laboratory for Computer Science*, 2002.

[BGI15] Elette Boyle, Niv Gilboa, and Yuval Ishai. "Function Secret Sharing." In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pp. 337–367. Springer, 2015.

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. "Function Secret Sharing: Improvements and Extensions." In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pp. 1292–1303. ACM, 2016.

[Bla93] M. Blaze. "A Cryptographic File System for Unix." In *CCS*, Fairfax, VA, Nov. 1993.

[BLR13] Avrim Blum, Katrina Ligett, and Aaron Roth. "A learning theory approach to noninteractive database privacy." *J. ACM*, **60**(2):12:1–12:25, 2013.

[BP98] S. Balasubramaniam and Benjamin C. Pierce. "What is a File Synchronizer?" In *MobiCom*, 1998.

[BSW07] J. Bethencourt, A. Sahai, and B. Waters. "Ciphertext-Policy Attribute-Based Encryption." In *SP'07*, Oakland, CA, Apr. 2007.

[Cal17a] "California Department of Transportation." http://pems.dot.ca.gov/, 2017.

[Cal17b] "Google's Waze announces government data exchange program with 10 initial partners." http://www.dot.ca.gov/cwwp/InformationPageForward.do, 2017.

[ccn12] "CCNx Codebase http://ccnx.org.", 2012.

[CEE01] Dwaine Clarke, Jean-Emile Elien, Carl Ellison, Matt Fredette, Alexander Morcos, and Ronald L. Rivest. "Certificate Chain Discovery in SPKI/SDSI." *Journal of Computer Security*, **9**(4):285–322, 2001.

[CJK07] Szymon Chachulski, Michael Jennings, Sachin Katti, and Dina Katabi. "Trading structure for randomness in wireless opportunistic routing." In Jun Murai and Kenjiro Cho, editors, *Proceedings of the ACM SIGCOMM 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Kyoto, Japan, August 27-31, 2007*, pp. 169–180. ACM, 2007.

[CM06] Kamalika Chaudhuri and Nina Mishra. "When Random Sampling Preserves Privacy." In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*, pp. 198–213. Springer, 2006.

[cpa12] "CP-ABE Implementation http://acsc.cs.utexas.edu/cpabe/.", 2012.

[cpu] "cpulimit." https://github.com/opsengine/cpulimit.

[CTY12] Chien-Chia Chen, Guruprasad Tahasildar, Yu-Ting Yu, Joon-Sang Park, Mario Gerla, and MY Sanadidi. "CodeMP: Network coded multipath to support TCP in disruptive MANETs." In *Mobile Adhoc and Sensor Systems (MASS), 2012 IEEE 9th International Conference on*, pp. 209–217. IEEE, 2012.

[CYR12] M. Chuah, P. Yang, S. Roy, and Bo Sheng. "Performance Evaluation of Dissemination Schemes for Coded Packets in Heterogeneous Sparse Ad Hoc Networks." *Ad Hoc And Sensor Wireless Networks*, **15**(2-4):151–181, 2012.

[dar13] "CONTENT-BASED MOBILE EDGE NETWORKING." http://www.darpa.mil/Our_Work/STO/Programs/Content-Based_Mobile_Edge_Networking_(CBMEN).aspx, 2013.

[DCN09] Jing Dong, Reza Curtmola, and Cristina Nita-Rotaru. "Practical defenses against pollution attacks in intra-flow network coding for wireless mesh networks." In *Proceedings of the second ACM conference on Wireless network security*, pp. 111–122. ACM, 2009.

[DE02] Steve Dohrmann and Carl M. Ellison. "Public-key Support for Collaborative Groups." In *Annual PKI Research Workshop*, Hanover, NH, Apr. 2002.

[DMN06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. "Calibrating Noise to Sensitivity in Private Data Analysis." In *TCC*, 2006.

[DR14] Cynthia Dwork and Aaron Roth. "The Algorithmic Foundations of Differential Privacy." *Foundations and Trends in Theoretical Computer Science*, **9**(3-4):211–407, 2014.

[DWJ13] John C. Duchi, Martin J. Wainwright, and Michael I. Jordan. "Local Privacy and Minimax Bounds: Sharp Rates for Probability Estimation." In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference*

on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States., pp. 1529–1537, 2013.

[Dwo06] Cynthia Dwork. "Differential Privacy." In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pp. 1–12. Springer, 2006.

[Dwo11] Cynthia Dwork. "A Firm Foundation for Private Data Analysis.", 2011.

[Ell96] Carl M. Ellison. "Establishing Identity Without Certification Authorities." In *USENIX'96*, San Diego, CA, Jan. 1996.

[ema] "EMANE." http://cs.itd.nrl.navy.mil/work/emane/.

[enc15] "ENCODERS (Edge Networking with Content-Oriented Declarative Enhanced Routing and Storage)." http://encoders.csl.sri.com/, 2015.

[EPK14] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. "RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response." In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pp. 1054–1067. ACM, 2014.

[Fac] "Facebook changes story, now says phone location not used to recommend friends." http://fusion.net/story/319712/facebook-now-says-phone-location-not-used-to-recommend-friends/.

[FBW06] Christina Fragouli, Jean-Yves Le Boudec, and Jörg Widmer. "Network coding: an instant primer." *Computer Communication Review*, **36**(1):63–68, 2006.

[FEW12] Adrienne Porter Felt, Serge Egelman, and David Wagner. "I've got 99 problems, but vibration ain't one: a survey of smartphone users' concerns." In Ting Yu, William Enck, and Xuxian Jiang, editors, *SPSM'12, Proceedings of the Workshop on Security and Privacy in Smartphones and Mobile Devices, Co-located with CCS 2012, October 19, 2012, Raleigh, NC, USA*, pp. 33–44. ACM, 2012.

[FHE12] Adrienne Porter Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. "Android permissions: user attention, comprehension, and behavior." In Lorrie Faith Cranor, editor, *Symposium On Usable Privacy and Security, SOUPS '12, Washington, DC, USA - July 11 - 13, 2012*, p. 3. ACM, 2012.

[fir] http://www.fire.uni-freiburg.de/other_rep/research/rus/rus_re_1bor_16.jpg.

[Fla] "Android flashlight app tracks users via GPS, FTC says hold on." http://www.techrepublic.com/blog/it-security/why-does-an-android-flashlight-app-need-gps-permission/.

[FSL06] Bryan Ford, Jacob Strauss, Chris Lesniewski-Laas, Sean Rhea, Frans Kaashoek, and Robert Morris. "Persistent Personal Names for Globally Connected Mobile Devices." In *OSDI'06*, Seattle, WA, Nov. 2006.

[FT86] James Alan Fox and Paul E Tracy. *Randomized response: a method for sensitive surveys*. Beverly Hills California Sage Publications, 1986.

[FWL06] C. Fragouli, J. Widmer, and J.-Y. Le Boudec. "A Network Coding Approach to Energy Efficient Broadcasting: From Theory to Practice." In *INFOCOM 2006*, pp. 1–11, April 2006.

[Gay09] Matthew Moore. "Gay men 'can be identified by their Facebook friends'." http://www.telegraph.co.uk/technology/facebook/6213590/Gay-men-can-be-identified-by-their-Facebook-friends.html, 2009.

[GHL12] Johannes Gehrke, Michael Hay, Edward Lui, and Rafael Pass. "Crowd-Blending Privacy." In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pp. 479–496. Springer, 2012.

[GLP11] Johannes Gehrke, Edward Lui, and Rafael Pass. "Towards Privacy for Social Networks: A Zero-Knowledge Based Definition of Privacy." In Yuval Ishai, editor, *Theory of Cryptography - 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings*, volume 6597 of *Lecture Notes in Computer Science*, pp. 432–449. Springer, 2011.

[GPS] "gpsd - a GPS service daemon." http://www.catb.org/gpsd/.

[GSM03] Eu-Jin Goh, Hovav Shacham, Nagendra Modadugu, and Dan Boneh. "SiRiUS: Securing Remote Untrusted Storage." In *NDSS'03*, 2003.

[Hag] Haggle: An innovative Paradigm for Autonomic Opportunistic Communication Research project. http://www.haggleproject.org/.

[HHH12] Emoke-Agnes Horvat, Michael Hanselmann, Fred A. Hamprecht, and Katharina A. Zweig. "One Plus One Makes Three (for Social Networks)." *PLOS ONE*, **7**(4):1–8, 04 2012.

[HMK06] Tracey Ho, Muriel Médard, Ralf Koetter, David R Karger, Michelle Effros, Jun Shi, and Ben Leong. "A random linear network coding approach to multicast." *Information Theory, IEEE Transactions on*, **52**(10):4413–4430, 2006.

[HS09] Sarah Henderson and Ananth Srinivasan. "An Empirical Analysis of Personal Digital Document Structures." In *HCII'09*, San Diego, CA, July 2009.

[JM09] Carter Jernigan and Behram Mistree. "Gaydar: Facebook friendships expose sexual orientation." *First Monday*, **14**(10), 2009.

[JPG05] William Jones, Ammy Jiranida Phuwanartnurak, Rajdeep Gill, and Harry Bruce. "Don't Take My Folders Away! Organizing Personal Information to Get Things Done." In *CHI'05*, Portland, OR, Apr. 2005.

[JRG16] Joshua Joy, Sayali Rajwade, and Mario Gerla. "Participation cost estimation: Private versus non-private study." In *2016 Mediterranean Ad Hoc Networking Workshop, Med-Hoc-Net 2016, Vilanova i la Geltru, Spain, June 20-22, 2016*, pp. 1–5. IEEE, 2016.

[JST09a] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca Braynard. "Networking named content." In Jörg Liebeherr, Giorgio Ventre, Ernst W. Biersack, and Srinivasan Keshav, editors, *Proceedings of the 2009 ACM Conference on Emerging Networking Experiments and Technology, CoNEXT 2009, Rome, Italy, December 1-4, 2009*, pp. 1–12. ACM, 2009.

[JST09b] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. "Networking Named Content." In *CoNEXT'09*, Rome, Italy, Dec. 2009.

[JST09c] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. "Networking named content." In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, CoNEXT '09, pp. 1–12, New York, NY, USA, 2009. ACM.

[JST09d] Van Joacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. "Networking named content." In *CoNEXT '09: Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, pp. 1–12. ACM, 2009.

[JST12] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nick Briggs, and Rebecca Braynard. "Networking named content." *Commun. ACM*, **55**(1):117–124, 2012.

[JYG13] Joshua Joy, Yu-Ting Yu, Mario Gerla, Samuel Wood, James Mathewson, and Mark-Oliver Stehr. "Network coding for content-based intermittently connected emergency networks." In Sumi Helal, Ranveer Chandra, and Robin Kravets, editors, *The 19th Annual International Conference on Mobile Computing and Networking, MobiCom'13, Miami, FL, USA, September 30 - October 04, 2013*, pp. 123–126. ACM, 2013.

[JYP14] Joshua Joy, Yu-Ting Yu, Victor Perez, Dennis Lu, and Mario Gerla. "A New Approach to Coding in Content-Based MANETs." *JCM*, **9**(8):588–596, 2014.

[KLN08] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. "What Can We Learn Privately?" In *49th Annual*

IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA, pp. 531–540. IEEE Computer Society, 2008.

[KLS00] Stephen Kent, Charles Lynn, and Karen Seo. "Secure Border Gateway Protocol (S-BGP)." *IEEE JSAC*, **18**(4):582–592, 2000.

[KR08] A. Keshavarz-Haddadt and R. Riedi. "Bounds on the Benefit of Network Coding: Throughput and Energy Saving in Wireless Networks." In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pp. –, April 2008.

[KRS03] Mahesh Kallahalla, Erik Riedel, Ram Swaminathan, Qian Wang, and Kevin Fu. "Plutus: Scalable Secure File Sharing on Untrusted Storage." In *FAST'03*, San Francisco, CA, Mar. 2003.

[KSG13] M. Kosinski, D. Stillwell, and T. Graepel. "Private traits and attributes are predictable from digital records of human behavior." *Proceedings of the National Academy of Sciences*, **110**(15):5802–5805, Apr 2013.

[Lan88] M. W. Lansdale. "The Psychology of Personal Information Management." *Applied Ergonomics*, **19**(1):55–66, 1988.

[LGK11] Seung-Hoon Lee, M. Gerla, H. Krawczyk, Kang-Won Lee, and E.A. Quaglia. "Performance Evaluation of Secure Network Coding Using Homomorphic Signature." In *Network Coding (NetCod), 2011 International Symposium on*, pp. 1–6, 2011.

[Lic13] M. Lichman. "UCI Machine Learning Repository.", 2013.

[Lis17] Wikipedia. "List of United States cities by area ." https://en.wikipedia.org/wiki/List_of_United_States_cities_by_area, 2017.

[LLL08] Yunfeng Lin, Baochun Li, and Ben Liang. "Efficient Network Coded Data Transmissions in Disruption Tolerant Networks." In *INFOCOM 2008. 27th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 13-18 April 2008, Phoenix, AZ, USA*, pp. 1508–1516. IEEE, 2008.

[LLV07] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. "t-Closeness: Privacy Beyond k-Anonymity and l-Diversity." In *ICDE*, 2007.

[LMH04] Desmond S. Lun, Muriel Médard, Tracey Ho, and Ralf Koetter. "Network Coding with a Cost Criterion." In *in Proc. 2004 International Symposium on Information Theory and its Applications (ISITA 2004*, pp. 1232–1237, 2004.

[LPY06] Uichin Lee, Joon-Sang Park, Joseph Yeh, Giovanni Pau, and Mario Gerla. "Code torrent: content distribution using network coding in VANET." In *Proceedings of the 1st international workshop on Decentralized resource sharing in mobile computing and networking*, MobiShare '06, pp. 1–5, New York, NY, USA, 2006. ACM.

[MF09] Alex McMahon and Stephen Farrell. "Delay- and Disruption-Tolerant Networking." *IEEE Internet Computing*, **13**(6):82–87, 2009.

[MGK06] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. "l-Diversity: Privacy Beyond k-Anonymity." In Ling Liu, Andreas Reuter, Kyu-Young Whang, and Jianjun Zhang, editors, *Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, 3-8 April 2006, Atlanta, GA, USA*, p. 24. IEEE Computer Society, 2006.

[MKK99] David Mazieres, Michael Kaminsky, M. Frans Kaashoek, and Emmett Witchel. "Separating Key Management from File System Security." In *SOSP'99*, Charleston, SC, Jan. 1999.

[MPI03] Stefan Miltchev, Vassilis Prevelakis, Sotiris Ioannidis, John Ioannidis, Angelos D. Keromytis, and Jonathan M. Smith. "Secure and Flexible Global File Sharing." In *USENIX*, 2003.

[MWT12] Marie-Jose Montpetit, Cedric Westphal, and Dirk Trossen. "Network coding meets information-centric networking: an architectural case for information dispersion through native network coding." In *Proceedings of the 1st ACM workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications*, NoM '12, pp. 31–36. ACM, 2012.

[ndn] "Named-Data Networking (NDN)." http://named-data.net.

[NES] NESEC. "Tornado Car." http://www.nesec.org/images/haz_tornado_car.jpg.

[NF04] Edmund B. Nightingale and Jason Flinn. "Energy-Efficiency and Storage Flexibility in the Blue File System." In *OSDI'04*, San Francisco, CA, Dec. 2004.

[NGR] E. Nordström, P. Gunningberg, and C. Rohner. "Haggle: Relevance-aware content sharing for mobile, devices using search.".

[NKZ10] Mohammad Nauman, Sohail Khan, and Xinwen Zhang. "Apex: extending Android permission model and enforcement with user-defined runtime constraints." In Dengguo Feng, David A. Basin, and Peng Liu, editors, *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2010, Beijing, China, April 13-16, 2010*, pp. 328–332. ACM, 2010.

[NRS07] Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. "Smooth sensitivity and sampling in private data analysis." In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pp. 75–84. ACM, 2007.

[OG10] S.Y. Oh and M. Gerla. "Protecting network coded packets in coalition networks." In *Wireless On-demand Network Systems and Services (WONS), 2010 Seventh International Conference on*, pp. 168–175, 2010.

[OGT09] Soon Y. Oh, Mario Gerla, and Abhishek Tiwari. "Robust MANET routing using adaptive path redundancy and coding." In *Proceedings of the First international conference on COMmunication Systems And NETworks*, COMSNETS'09, pp. 224–233, Piscataway, NJ, USA, 2009. IEEE Press.

[Ped91] Torben Pryds Pedersen. "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing." In *CRYPTO'91*, Santa Barbara, CA, Aug. 1991.

[PF06] Daniel Peek and Jason Flinn. "EnsemBlue: Integrating Distributed Storage and Consumer Electronics." In *OSDI*, 2006.

[PGH98] T. W. Page, R. G. Guy, J. S. Heidemann, D. Ratner, P. Reiher, A. Goel, G. H. Kuenning, and G. J. Popek. "Perspectives on Optimistically Replicated Peer-to-Peer Filing." *SPE*, **28**(2):155–180, 1998.

[PGL06] Joon-Sang Park, M. Gerla, D.S. Lun, Y. Yi, and M. Medard. "Codecast: a network-coding-based ad hoc multicast protocol." *Wireless Communications, IEEE*, **13**(5):76–81, 2006.

[PSY04] Justin Mazzola Paluska, David Saff, Tom Yeh, and Kathryn Chen. "Footloose: A Case for Physical Eventual Consistency and Selective Conflict Resolution." In *WMCSA'04*, Lake District National Park, UK, Dec. 2004.

[RHR04] Peter Reiher, John S. Heidemann, David Ratner, Gregory Skinner, and Gerald J. Popek. "Resolving File Conflicts in the Ficus File System." In *USENIX'94*, Boston, MA, June 2004.

[Riv98] Ronald Rivest. "Can We Eliminate Certificate Revocation Lists?" In *In Financial Cryptography*, 1998.

[SGS14] Emre Sarigöl, David Garcia, and Frank Schweitzer. "Online privacy as a collective phenomenon." In Alessandra Sala, Ashish Goel, and Krishna P. Gummadi, editors, *Proceedings of the second ACM conference on Online social networks, COSN 2014, Dublin, Ireland, October 1-2, 2014*, pp. 95–106. ACM, 2014.

[SGZ02] Sumeet Sobti, Nitin Garg, Chi Zhang, Xiang Yu, Arvind Krishnamurthy, and Randolph Y. Wang. "PersonalRAID: Mobile Storage for Distributed and Disconnected Computers." In *FAST'02*, Monterey, CA, Mar. 2002.

[SHC06] J. Scott, P. Hui, J. Crowcroft, and C. Diot. "Haggle: A Networking Architecture Designed Around Mobile Users." In *WONS*, 2006.

[SKK90] M. Satyanarayanan, J. J. Kistler, P. Kumar, M. E. Okasaki, E. H. Siegel, and D. C. Steere. "Coda: A Highly Available File System for a Distributed Workstation Environment." *IEEE Transaction on Computer*, **39**(4):447–459, 1990.

[SL09] Hassan Shojania and Baochun Li. "Random network coding on the iPhone: fact or fiction?" In Wei Tsang Ooi and Dongyan Xu, editors, *Network and Operating System Support for Digital Audio and Video, 19th International Workshop,*

NOSSDAV 2009, Williamsburg, VA, USA. June 3-5, 2009, Proceedings, pp. 37–42. ACM, 2009.

[SL10] Hassan Shojania and Baochun Li. "Tenor: making coding practical from servers to smartphones." In Alberto Del Bimbo, Shih-Fu Chang, and Arnold W. M. Smeulders, editors, *Proceedings of the 18th International Conference on Multimedia 2010, Firenze, Italy, October 25-29, 2010*, pp. 45–54. ACM, 2010.

[SLP09] Jacob Strauss, Chris Lesniewski-Laas, Justin Mazzola Paluska, Bryan Ford, Robert Morris, and Frans Kaashoek. "Device Transparency: A New Model for Mobile Storage." In *HotStorage'09*, Big Sky, MT, Oct. 2009.

[SS94] Ravi S. Sandhu and Pierangela Samarati. "Access Control: Principle and Practice." *IEEE Communications Magazine*, **9**(32):40–49, 1994.

[SSH07] Jing Su, James Scott, Pan Hui, Jon Crowcroft, Eyal De Lara, Christophe Diot, Ashvin Goel, Meng How Lim, and Eben Upton. "Haggle: seamless networking for mobile applications." In *Proceedings of the 9th international conference on Ubiquitous computing*, UbiComp '07, pp. 391–408, Berlin, Heidelberg, 2007. Springer-Verlag.

[Sto07] "Personal Content and Home Network Storage, the Perfect Storm, Tom Coughlin.", 2007.

[Swe02] Latanya Sweeney. "k-Anonymity: A Model for Protecting Privacy." *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, **10**(5):557–570, 2002.

[Tam81] Ajit C. Tamhane. "Randomized Response Techniques for Multiple Sensitive Attributes." *Journal of the American Statistical Association*, **76**(376):916–923, 1981.

[TTP95] Douglas Terry, Marvin Theimer, Karin Petersen, Alan Demers, Mike Spreitzer, and Carl Hauser. "Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System." In *SOSP'95*, 1995.

[VCC11] T. Vidas, N. Christin, and L. Cranor. "Curbing Android permission creep." In *Proceedings of the Web 2.0 Security and Privacy 2011 workshop (W2SP 2011)*, Oakland, CA, May 2011.

[VMF08] Kaushik Veeraraghavan, Andrew Myrick, and Jason Flinn. "Cobalt: Separating Content Distribution from Authorization in Distributed File Systems." In *FAST'08*, 2008.

[War65] Stanley L Warner. "Randomized response: A survey technique for eliminating evasive answer bias." *Journal of the American Statistical Association*, **60**(309):63–69, 1965.

[WCK05] Y. Wu, P.A. Chou, and Sun-Yuan Kung. "Minimum-energy multicast in mobile ad hoc networks using network coding." *Communications, IEEE Transactions on*, **53**(11):1906–1918, Nov 2005.

[WLX13] Qinghua Wu, Zhenyu Li, and Gaogang Xie. "CodingCache: multipath-aware CCN cache with network coding." In *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*, ICN '13, pp. 41–42. ACM, 2013.

[WMJ13] Samuel Wood, James Mathewson, Joshua Joy, Mark-Oliver Stehr, Minyoung Kim, Ashish Gehani, Mario Gerla, Hamid Sadjadpour, and J.J. Garcia-Luna-Aceves. "ICEMAN: A System for Efficient, Robust and Secure Situational Awareness at the Network Edge." http://www.csl.sri.com/users/stehr/CBMEN/iceman-eval-2013.pdf, 2013.

[WMJ15] Samuel Wood, James Mathewson, Joshua Joy, Mark-Oliver Stehr, Minyoung Kim, Ashish Gehani, Mario Gerla, Hamid R. Sadjadpour, and J. J. Garcia-Luna-Aceves. "ICEMAN: A Practical Architecture for Situational Awareness at the Network Edge." In Narciso Martí-Oliet, Peter Csaba Ölveczky, and Carolyn L. Talcott, editors, *Logic, Rewriting, and Concurrency - Essays dedicated to José Meseguer on the Occasion of His 65th Birthday*, volume 9200 of *Lecture Notes in Computer Science*, pp. 617–631. Springer, 2015.

[WYG17] Frank Wang, Catherine Yun, Shafi Goldwasser, Vinod Vaikuntanathan, and Matei Zaharia. "Splinter: Practical Private Queries on Public Data." In Aditya Akella and Jon Howell, editors, *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017*, pp. 299–313. USENIX Association, 2017.

[XPr] "XPrivacy." https://github.com/M66B/XPrivacy.

[YRL08] Shucheng Yu, Kui Ren, and Wenjing Lou. "Attribute-based Content Distribution with Hidden Policy." In *NPSec'08*, Orlando, FL, Oct. 2008.

[YWR10] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. "Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing." In *INFOCOM'10*, 2010.

[YYZ13] Zhemin Yang, Min Yang, Yuan Zhang, Guofei Gu, Peng Ning, and Xiaoyang Sean Wang. "AppIntent: analyzing sensitive data transmission in android for privacy leakage detection." In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pp. 1043–1054. ACM, 2013.

[Zha10] Lixia Zhang et al. "Named Data Networking (NDN) Project." Technical report, PARC Technical Report NDN-0001, 2010.

[Zha11] Lixia Zhang. "Evolving internet to the future via named data networking." In Kanchana Kanchanasut, editor, *AINTEC '11, Asian Internet Engineering Conference, Bangkok, Thailand, November 09 - 11, 2011, Proceedings*, p. 72. ACM, 2011.

[ZZJ11] Yajin Zhou, Xinwen Zhang, Xuxian Jiang, and Vincent W. Freeh. "Taming Information-Stealing Smartphone Applications (on Android)." In Jonathan M. McCune, Boris Balacheff, Adrian Perrig, Ahmad-Reza Sadeghi, M. Angela Sasse, and Yolanta Beres, editors, *Trust and Trustworthy Computing - 4th International Conference, TRUST 2011, Pittsburgh, PA, USA, June 22-24, 2011. Proceedings*, volume 6740 of *Lecture Notes in Computer Science*, pp. 93–107. Springer, 2011.