Louisiana State University

LSU Digital Commons

LSU Doctoral Dissertations | Graduate School

March 2019

Distributed Wireless Algorithms for RFID Systems: Grouping Proofs and Cardinality Estimation

Vanya D. Cherneva

Follow this and additional works at: https://digitalcommons.lsu.edu/gradschool_dissertations

Part of the Other Computer Engineering Commons

Recommended Citation

Cherneva, Vanya D., "Distributed Wireless Algorithms for RFID Systems: Grouping Proofs and Cardinality Estimation" (2019). *LSU Doctoral Dissertations*. 4873.
https://digitalcommons.lsu.edu/gradschool_dissertations/4873

This Dissertation is brought to you for free and open access by the Graduate School at LSU Digital Commons. It has been accepted for inclusion in LSU Doctoral Dissertations by an authorized graduate school editor of LSU Digital Commons. For more information, please contactgradetd@lsu.edu.

DISTRIBUTED WIRELESS ALGORITHMS FOR RFID SYSTEMS: GROUPING PROOFS AND CARDINALITY ESTIMATION

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Division of Electrical and Computer Engineering

by
Vanya D Cherneva
BSE, Bulgaria's National Military Academy, 2002
MSE, Bulgaria's National Military Academy, 2003
MSE, Louisiana State University, 2013
May 2019

ACKNOWLEDGEMENTS

I would like to express my never ending gratitude to my advisors, Professor Ramachandran Vaidyanathan and Professor Jerry Trahan for their guidance, and constant motivation towards the completion of this dissertation. Their technical advice and suggestions helped me to overcome hurdles and kept me enthusiastic and made this work a great learning experience. In addition, I would like to thank my committee members Professor Suresh Rai, Professor Shuangqing Wei, and Professor Sunyoung Park for taking time out of their busy schedule and agreeing to be a part of my committee and for their valuable feedback.

I would like to thank the faculty members of the Division of Electrical and Computer Engineering, whose classes I had the privilege of attending while completing my studies at Louisiana State University.

TABLE OF CONTENTS

CHAPTER

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The breadth and depth of the use of Radio Frequency Identification (RFID) are becoming more substantial. RFID is a technology useful for identifying unique items through radio waves. We design algorithms on RFID-based systems for the Grouping Proof and Cardinality Estimation problems.

A grouping-proof protocol is evidence that a reader simultaneously scanned the RFID tags in a group. In many practical scenarios, grouping-proofs greatly expand the potential of RFID-based systems such as supply chain applications, simultaneous scanning of multiple forms of IDs in banks or airports, and government paperwork. The design of RFID grouping-proofs that provide optimal security, privacy, and efficiency is largely an open area, with challenging problems including robust privacy mechanisms, addressing completeness and incompleteness (missing tags), and allowing dynamic groups definitions. In this work we present three variations of grouping-proof protocols that implement our mechanisms to overcome these challenges.

Cardinality estimation is for the reader to determine the number of tags in its communication range. Speed and accuracy are important goals. Many practical applications need an accurate and anonymous estimation of the number of tagged objects. Examples include intelligent transportation and stadium management. We provide an optimal estimation algorithm template for cardinality estimation that works for a $\{0, 1, e\}$ channel, which extends to most estimators and ,possibly, a high resolution $\{0, 1, ..., k-1, e\}$ channel.

CHAPTER 1
INTRODUCTION

In this time of ubiquitous computing and the evolution of the Internet of Things (IoT), the deployment and development of Radio Frequency Identification (RFID) is becoming more extensive. RFID is a method of identifying unique items using radio waves. An RFID system typically consists of three different types of entities: tags, readers and a verifier. The tags are embedded in, or attached to, objects to be identified. The most common tags are passive tags. Passive tags use radio frequency (RF) energy induced by electromagnetic waves emitted by the reader. A typical communication sequence is: the reader emits a continuous RF wave, a tag in the RF field of the reader receives energy from the field and responds [17, 40, 52, 67, 86, 90, 111, 125]. RFID is an open wireless communication technology, inexpensive and very energy efficient, particularly for passive RFID tags. Important features of RFID technology are that it allows to write and read data contactless and without a line of sight.

RFIDs are important to modern society. This dissertation studies algorithms on RFID-based systems: Grouping Proof and Cardinality Estimation.

Grouping Proof: A grouping-proof protocol is evidence that a reader simultaneously scanned the RFID tags in a group [16, 53, 86, 87, 93, 98, 99, 106].

There are many practical scenarios where grouping-proofs can greatly expand the potential of RFID-based systems, such as drugs to be shipped, or dispensed, with information leaflets (safety regulation); inventory of equipment in before and out after a surgery; simultaneous scanning of multiple forms of IDs in banks or airports; government paperwork; evidence in court etc. We are proposing grouping-proof protocols that improve on security, privacy, and efficiency over existing schemes. For example, the protocol must resist replay attacks, where an adversary who eavesdrops on the communications between reader and tag and obtains exchanged messages should not be able to obtain any tag/reader secrets by resending earlier messages. RFID tags are critically constrained in memory, communication, and computation. We want a protocol to be compatible with the technical capabilities of an RFID. We also want a protocol to be scalable. Scalability problems in RFID grouping-proofs, such as collision from responses of multiple tags after a reader request or due to an increasing number of tags in the system, will gradually degrade its performance. Researchers have often overlooked scalability when designing protocols. We work through these protocol ideas in designing and proposing our own approaches.

The design of RFID grouping-proofs that provide optimal security, privacy, and efficiency is largely an open area, with challenging problems including robust privacy mechanisms, addressing completeness and incompleteness (missing tags), and allowing dynamic groups definitions.

Cardinality Estimation: In cardinality estimation, the reader determines the number of tags in its communication range. Speed and accuracy are crucial [4, 20, 34, 42, 48, 53, 68, 97, 121, 129, 133]. It has many practical applications. For example, in warehouse management (with retailers like Walmart), where thousands of products (such as mobile phones, iPods, tablets, and other peripherals) are stored and tracked in a small space. Accurately estimating the number of tagged objects for recurrent inventory reports, instead of laborious and unreliable manual counting, is important. In some privacy sensitive scenarios, such as counting the number of visitors to an event where attendees have RFID tickets/cards/bands, the exposure of identification information on tags, such as credit card information, can put privacy at risk. Therefore, a scheme that can use the non-identifiable information from tags to compute the cardinality is highly desired. In this part of the study we consider a slotted (synchronous) communication channel (through which tags communicate with the reader) that accepts binary inputs from set $\{0, 1\}$ and outputs a symbol $\{0, 1, e\}$, which indicates at a given slot there are zero, one, or more than one writes by tags to the channel.

We provide an algorithm for tag population that works for a $\{0, 1, e\}$ channel. This first part of our work is an extension of the work of Hasan *et al.* [48] who proposed the GERT (Gaussian Estimator of RFID Tags) algorithm for estimating tag population in $\{0, e\}$ (a channel that outputs 0 when no tags write and an e if it writes) and applied the $f_e - f_0$ estimator. Following the same analytical approach, we extend the work to a $\{0, 1, e\}$ channel. We too used the $f_e - f_0$ estimator; however, it has different meaning compared to that in the $\{0, e\}$ channel. In the second part of our work on tag population estimation, we developed a framework for using simulated data to generate an optimal estimation algorithm. As a product of this work we also develop methods for initial estimates, extending a well known method due to Willard [119] to more accurate Willard, Willard+, and Willard$*$ algorithms.

1.1 RFID and Its Importance

The applications of RFID-based systems are many and varied in numerous industries, including ones in high-risk environments, health and safety. The fundamental capability of RFID systems is to track tagged items which avoids the need of manual workforce as the data is generated automatically. It wirelessly identifies people and objects by sensing the surroundings. More and more businesses are using this technology including in the next generation of business intelligence [2, 5, 30, 31, 32, 41, 43, 49, 52, 54, 61, 66, 83, 86, 92, 96, 111, 113, 114, 116, 117, 123, 126].

The breadth of applications includes the healthcare industry [2, 30, 31, 43, 113], where RFIDs are used to track inventory, identify patients, manage personnel, monitor the user's health and activate remote assistance.

Food supply chain (FSC) - farm to fork [32, 83, 92, 116, 123] uses tags in food logistics that aim to enable new types of efficient and responsive networks with flexible tracking, tracing, and decision support based on information.

The retail industry itself employs smart shopping carts [49, 61] that use low energy Bluetooth to track and localize items and display promotions and sales for the customer.

The data generated can also be used for pricing, product placement, and reducing waiting times in checkout (Amazon Go).

The transportation sector uses tags for traffic management [5, 41, 96, 114]. Examples are an advanced plate recognition system for car parking, traffic congestion control, vehicle fusion positioning, tracking items on conveyor belt, and traffic light control.

The increasing popularity and ubiquity of RFID technology, however, comes with serious concerns for security and privacy. Some example security concerns are the following. Impersonating the reader or a tag, an attacker may be able to obtain a reader's and a tag's secret values. If an attacker blocks the exchanged message(s) or when messages are lost during transmission (system crash or communication error), then the tags can be desynchronized from a reader. Privacy concerns include the following. An attacker may be able to break the anonymity of the reader and tags in the grouping-proof (called "information leakage"). If an attacker observes that a tag's responses are static or linked, then it can track the tag's location over time (called "malicious traceability").

Other factors affecting the efficiency and the performance of an RFID system are [36, 37, 103, 104, 105, 106, 107, 108]: an estimate for tag population, number of tags read by the reader in case of collision, the read range of the tag around the reader's surroundings, an authentication mechanism of the entities involved in the communication, a searching mechanism for tags, etc. Continued improvement in circuit design aims to improve protocols, read range, reliability, etc. RFID costs decreased in the past decade which leads to escalating of their applications.

The remainder of this work is organized as follows. Chapter 2 presents background and prior work in the literature. Background contains general technical details for the components of a common RFID system. Prior work in the literature includes discussion on prior work on grouping-proofs and prior work on cardinality estimation. Chapter 3 describes the overview of RFID grouping-proof protocols. It provides comprehensive grouping-proof characteristics - definition, model, design requirements and how it all applies while designing a protocol. We have formally defined an adversary model and RFID privacy model, under which we evaluate our presented grouping-proofs protocols in the following chapters. Chapter 4 describes the Serial-Dependency Grouping Proof Protocol (SDGPP). Chapter 5 describes the Parallel-Dependency Grouping Proof Protocol (PDGPP). Chapter 6 describes the Dynamic Grouping Proof Protocol (DGPP). Chapter 7 describes the cardinality estimation approaches for tag population. We then conclude with a summary of results in this dissertation and propose possible future work.

CHAPTER 2
BACKGROUND AND PRIOR WORK

2.1 Background

Wireless channels connect objects and enable communication among those objects and also with the Internet. Radio Frequency Identification (RFID) is one of the resource-constrained technologies (such as IEEE 802.15.4, Bluetooth, Bluetooth Low Energy, IEEE 802.11 Low Power, PLC, NFC) that permit data transfer with low energy consumption [111]. Advances in RFID technology allow information to be stored (tags can hold data) and then be read contactless and without a line of sight. In an RFID system tags are classified as active, semi-passive, or passive [40]. We intentionally omit the technical description of active and semi-passive tags. Both types of tags are too expensive to place on low-cost items, and our work does not consider them.

A Passive tag: This type of tags has no internal power source. It harvests energy from the nearby readers, then that energy energizes the chip to send an answer back to the reader's request. Thus, its computation and communication capabilities are very limited. For the EPC Gen2 protocol, a global standard in ISO UHF [1], the communication distance is at most 10m, data rate is 0-60 Kb/s, the frequency range is 860-960MHz, and up to 3000 gates are available for security implementation [40, 86, 87, 125]. Also, this standard provides a tag with up to four inventory flags, which allow four readers in parallel to communicate with a single tag at any given time.

Common RFID protocols include EPC Gen1 and EPC Gen2 [2]. Our work considers the EPC Gen2 features. EPC Gen1 has no global standard. EPC Gen2 has a requirement of a minimum 96-bits for EPC identifier. EPC Gen2 also provides an optional unlimited user tag memory. The additional memory could store time stamps from transactions, codes, and

[1] ISO 18000-6C Ultra High Frequency

[2] EPC - Electronic Product Code

sensing data. [33, 86].

RFID tags do not have clocks. However, the activity time of a tag during a single session is limited. In EPC Gen2 when tags are energized by a reader and engaged in a communication, they are capable of receiving and acting on reader commands within a period not exceeding the maximum settling-time interval in the protocol. Both reader and tags need to meet all timing requirements according the technical documentation for the protocol.

The Reader: This is a relatively powerful device. That provides power to the tags in order to communicate with them. For EPC Gen2, the maximum theoretical reading speed is 1000 tags/second. Experimental results [38, 106] show a minimum tag read rate of 150 tags/second to a maximum read rate of 450/second. Both simulations were conducted in virtual environments and hence some slight performance variation should be anticipated in real-world implementations. In writing operations, readers can write to five tags per second. A reader may have up to 1000m range [21, 40, 86, 87, 125].

A reader manages tag populations using three operations - select, inventory, and access, described briefly below [40].

- Select - A reader selects a tag population for subsequent inventory authentication. Select comprises the Select and Challenge commands.
- Inventory - The process by which a reader identifies tags. Inventory comprises multiple commands.
- Access - The process by which a reader transacts with (reads, writes, authenticates, or otherwise engages with) an individual tag. Access comprises multiple commands.

Each of the reader operations refesr to multiple commands on a tag side - ready, arbitrate, reply, acknowledge, open, secured, killed [40].

The verifier: This is a powerful back-end server. That acts as a trusted entity that maintains a database, containing the information needed to identify tags.

2.1.1 Other Definitions Related to RFID Systems

We have added a list for some additional technical definitions that may be helpful throughout the content of the research presented [40, 86].

Electronic Product Code (EPC) - Complements the bar codes. The EPC has digits to identify the manufacturer, product category and individual item.

Backscatter modulation - A method of communication between passive tags and readers; in this process a tag responds to a reader signal or field by modulating and reradiating the response signal at the same carrier frequency. The reflected signal is modulated to transmit data.

Reader-Talks-First - A reader initiates communication with tags in its read field. The reader sends energy to the tags but the tags sit idle until the reader requests them to respond. The reader is able to find tags by specifications designated in the EPC protocol.

Singulation - When an RFID reader identifies a tag with specific identity from a number of tags in its field. There are different methods of singulation, but the most common is "tree walking" (for example, asking all tags with an identity that starts with 0 or 1 to respond; if more than one responds, then the reader may ask for all tags with an identity that starts with 01 to respond, and so on).

2.1.2 The Communication Channel

Tags communicate with the reader through a wireless channel, signaling their presence by writing to the channel. In this level of abstraction a tag is said to write (a symbol $'1'$) or not write (a symbol $'0'$) on the channel. Thus the channel accepts symbols from $\{0, 1\}$. The channel produces one output symbol which is typically read by the reader. Communication through the channel is synchronized or "slotted". That is, all tags and reader access the channel in lock step. During a slot, a subset S of tags can write to the channel. This is indicated as tags of S writing a $'1'$ to the channel and the tags of $\overline{S}$ writing a $'0'$ to the channel.

The $\{0, e\}$ channel is usually represented in the literature to as the $\{0, 1\}$ channel. Our tag estimation scheme as well as others, uses a frame (collection of slots) to examine the population. Typically, tags write a $'1'$ or a $'0'$ into each slot of the frame and the channel outputs a symbol from $\{0, 1, ..., k-1, e\}$ for each slot, indicating the number of tags writing to the slot. For any symbol $s \in \{0, 1, ..., k-1, e\}$, we denote by f_s the number of slots containing symbol $'s'$. We will use f_s for tag population estimation. In our work we are also introducing the concept of high resolution channel. A channel of *resolution* $k \geq 1$ is one that accepts input symbols from $\{0, 1\}$ and outputs a symbol from $\{0, 1, ..., k-1, e\}$

2.2 Prior Work on Grouping-Proof

Juels [55] introduced the problem of giving evidence that two tags have been simultaneously scanned, called a *yoking proof.* A *grouping proof* generalizes this to a larger number of tags. Bolotnyy and Robins [11] introduced the idea of anonymous grouping-proofs. Early work on anonymous grouping-proof protocols includes Burmester *et al.* [15], Peris-Lopez *et al.* [85], Chien and Liu [27], and others [39, 65, 73, 88, 110]. Grouping-proof protocols from the past few years reveal a variety of approaches to address different goals, such as achieving high efficiency (scalability) and improving on privacy and security [1, 16, 26, 51, 79, 93, 94, 98, 99, 101, 106, 134, 136]. Below we review some recent and interesting papers in grouping-proof research, also used for comparison to our own work.

Zhang *et al.* [134] introduced a parallel grouping-proof protocol with an anti-collision algorithm based on adaptive pruning query tree (A4PQT) to identify the response message of tags. The protocol involves updating state variables, which allows the entity involved in the grouping-proof protocol to have values that stay run specific.

The protocol begins with a reader's request to get pre-authorized from the verifier. Then the reader executes two rounds in parallel then one round with serial communication, sending a message to each individual tag in the group to update its pseudonym and secret values for its state. The protocol does not have integrity checks and does not verify that tags updated at the end of the protocol, which make it vulnerable to impersonation, desynchronization, denial of proof, replay, and message integrity attacks.

Rostampour *et al.* [93] introduced a one-round grouping-proof protocol. The reader broadcasts the request message to all tags once and the tags respond to the reader. The protocol lacks dependency among tag communications, which requires a second round in a grouping-proof protocol. The protocol is suitable for passive tags, as it requires only PRNG and XOR operations. The protocol does not provide an acceptable security level and is vulnerable to replay, message integrity, and denial of proof attacks.

The protocol provides a formal security analysis using BAN (Burrows, Abadi, Needham) logic for analyzing its robustness. A comparison table weighs the performance in terms of various aspects such as scalability, gate equivalent, computation cost of a tag, exchanged messages between the reader and the tags, and the storage of each tag. However, we find some comparisons not accurately evaluated in the security comparison table. Also, the protocol compares to older grouping proof protocols rather than current ones.

Shi *et al.* [98] introduced a parallel grouping-proof protocol based on a DHCP (Dynamic Host Configuration Protocol) mechanism. The protocol involves multiple readers and multiple tag groups. During the grouping proof, the verifier chooses one reader and one tag group by means of a DHCP mechanism. DHCP mechanism implies selecting entities in a protocol according to their arrival time after a query is sent.

The protocol is designed for two modes: active mode and passive mode. Under active mode, the verifier knows the tag group for which it wants to search, as opposed to under the passive mode where the verifier does not know the tag group for which it wants to search. The protocol includes four phases: authorize a reader, choose a tag group, generate grouping-proof evidence, and verify the grouping-proof evidence. The protocol is not suitable for passive tags, because the tags compute hash functions. The protocol is vulnerable to denial of proof and message integrity attacks.

Hsi *et al.* [51] introduced a protocol for scalable grouping-proofs. Scalability problems in a grouping-proof protocol arise due to:

- messages relayed from one tag to another - when tag numbers increase, system performance gradually reduces;

- collision - after a reader queries tags, multiple tags respond simultaneously;

- exhaustive search - for protecting the privacy of tags, protocols adopt anonymity and make the verifier exhaustively search a tag database for the actual identity of a tag.

Hsi *et al.*'s protocol uses a run-specific pseudo-identity of tags in the group to protect each tag's privacy and improve on system performance. Tags order their responses based on the location of their pseudo-identity in a randomly permuted sequence of those identities. This also helps avoid tracking internal members of the group.

Burmester and Munilla [16] introduced a well designed protocol under very clear and strong grouping-proof design criteria. The reader uses an "erasure code" to enable identifying missing tag identities from a generated grouping proof if the proof is incomplete.

Abughazalah *et al.* [1] introduced a one-round grouping-proof protocol with no dependency among tag computations. The reader broadcasts a message to all tags, then the tags prepare the responses and reply to the reader. The protocol is not suitable for passive tags, because the tags compute hash functions. The protocol does not provide an acceptable security level and is vulnerable to impersonation, desynchronization, denial of proof, replay, and message integrity attacks.

Many of the current approaches to grouping proofs do not comply with the EPC standard for passive RFID tags and assume complex encryption schemes and cryptographic techniques. Zhou *et al.* [136] introduced a protocol that uses elliptic-curve cryptography (ECC). The practical implementation of ECC is still an open research problem [100], as it is a costly technique in term of gates to implement security features in passive EPC tags. The paper provides improvements in key distribution.

Yuan and Liu [128] introduced a protocol within the universally composable (UC) framework. Universal composability [18, 19] specifies a particular approach to security formalization, which guarantees that a UC-based security proof for a protocol remains valid if it is composed with other protocols (modularity) or if it is executed in arbitrary concurrent settings. RFID systems can be components of a more elaborate application, and as a result protocols are secure under composability with arbitrary applications. The main attribute of the UC framework is that UC security of a composite system can be obtained from the UC security of its components without the need of further integrated evaluation of its robustness [86].

In Yuan and Liu's protocol, messages exchanged during a session have no time limitation and the protocol implements no updating mechanism. The protocol is very well designed under the assumption of UC framework. However, it has several issues related to inefficiency.

2.2.1 RFID Privacy Models

Coisel and Martin [28] introduced an overview of existing RFID privacy models. The paper reviews eight RFID privacy models and analyzes their advantages and disadvantages, examining them across a selection of RFID authentication protocols. The paper concludes the following: as protocols ensure different privacy levels, no model is able to accurately distinguish them. They group the models according to features (for example - tag corruption ability). A classification like this helps to determine the most suitable model to be applied for a privacy analysis of an RFID protocol. However, some combinations of features may not match any model which makes it difficult to distinguish protocols that evidently ensure different privacy levels. The paper provides an overall comparison view of the eight models' relations and privacy properties. It shows that no model globally outclasses the other ones.

The paper stated that the most comprehensive RFID privacy model published so far is the Vaudenay model [115]. Vaudenay introduced an important point in his model, where privacy is considered as being ensured as long as an adversary cannot detect that a given tag is simulated without the knowledge of its secret. After then, researchers have proposed many models using different approaches, introducing new features, or pointing out weaknesses in the existing models. Unfortunately, even though the Vaudenay model is shown to be the most comprehensive RFID privacy model, the model is unable to distinguish some currently proposed protocols that have different privacy levels and so cannot provide a full privacy assessment.

Avoine *et al.* [8] introduced an RFID untraceability model. The model aims to assist while designing proofs or describing attacks. It is a modular model where adversary actions (oracles), capabilities (selectors and restrictions), and goals (experiment) follow a straight forward approach. The model design enhances the ability to formalize new adversarial assumptions and future evolutions of the technology. The paper's untraceability model provides a comprehensive privacy assessment and evaluation of protocols. We have used the model to assess our protocols (see Chapters 4 - 6). Chapter 3 presents details about the model.

2.3 Prior Work on Cardinality Estimation

Hasan *et al.* [48] introduced an estimator for the $\{0, e\}$ (or $\{0, 1\}$) channel. This estimator is $f_e - f_0$ (or $f_1 - f_0$ in this setting). Their GERT (Gaussian Estimator of RFID Tags) algorithmic model uses slotted Aloha. The method makes the estimator distribution in a frame Gaussian by using an adequately large frame size. Also, it analytically bounds the approximation error (assuming a Gaussian distribution) in terms of the error requirement of the tag estimate problem. We use the method of Hasan *et al.* to analyze the $f_e - f_0$ estimator for the $\{0, 1, e\}$ channel.

Liu *et al.*'s [68] SEM (simultaneous estimation for multi-category RFID systems) exploits the Manchester-coding mechanism, where it decodes the combined signals (simultaneously obtaining the reply status of tags from each category). An output is multiple bit vectors, decoded from just one physical slotted frame. To ensure the predefined accuracy, SEM calculates the variance of the estimate in one round, as well as the variance of the average estimate in multiple rounds.

Shahzad *et al.*'s [97] ART (Average Run-Based Tag) uses the average run length of nonempty slots in a $\{0, e\}$ channel. The first trial in ART is to obtain a rough estimate. The quality of this rough estimate is low since, since ART uses only a single trial. In each following trial, each tag participates independently with a certain probability of writing (balls-and-bins trials). ART then observes which slots in each trial are nonempty and it calculates the average run length of nonempty slots (the average length of sequences of consecutive nonempty slots) and uses such information to generate a final estimate (the more nonempty slots the larger the average run length is).

Zhang *et al.* [129] introduced joint cardinality estimation among tags. The paper's motivation comes from practical scenarios with a need to monitor tagged objects of many different types. The paper extends the traditional RFID estimation problem to a problem for estimating the category-level information over multiple tag sets at different locations and/or different times.

Yu *et al.* [53] introduced a method for tag estimation and counting for static and dynamic groups. The paper defined the static case in which tag population remains constant during the estimation process, as opposed to the dynamic case where tag population may vary during the estimation process. The paper proposes a generic framework of stable and accurate tag population estimation schemes based on Kalman filtering.

Gong *et al.* [72] showed that ART [97] and SRC [133] work only for certain distributions of the number of tags (such as uniform or normal). This does not work in practice. The Gong *et al.* work [72] introduces a new estimator called RPC (rigorous practical cardinality) and shows that it works well. However, it requires passive tags to use hashed IDs which impose an overhead on the system.

Zhou *et al.* [133] demonstrated that the key design aspect for any RFID counting function to achieve near-optimal performance is a conceptual separation of a protocol into two phases. The first phase uses a small overhead to obtain a rough estimate, and the second phase uses the rough estimate to achieve an accuracy target.

Based on this they establish lower bounds of:

- [Phase 1:] $\Omega(\log\log T)$ slots to obtain an initial estimate with constant relative error; this is similar to the Willard Algorithm approach [119];
- [Phase 2:] $\Omega(1/\alpha^2 \log(1/\alpha))$ slots to obtain a final estimate with α error probability.

They also compared several estimators including EZB, (enhanced) FNEB [47], LOF [47], ZOE [131], A3 [45], etc. They also presented their own approach SRC (Simple RFID Counting Protocol), that applied the two-phase method.

Wu *et al.* [121] introduced a method for tag estimation using a capture-aware Bayesian estimate algorithm in which tag distance from the reader is used to refine the tag estimate.

Some other important estimation functions are listed below.

- LoF - uses the length of continues non-empty slots [89]
- FNEB - uses indices of the first non-empty slot for multiple rounds [47]
- EZB - number of empty slots in the frame for estimation [60]

CHAPTER 3
OVERVIEW OF GROUPING PROOF PROTOCOL CONCEPTS

3.1 Introduction

A grouping-proof is evidence that a reader simultaneously scanned the RFID tags in a group [1, 15, 16, 26, 51, 55, 79, 101, 106, 109, 136]. All communication is within RFID wireless channels. A verifier must be able to use the proof to document the presence of objects in the group. The unique aspects of RFID applications, efficiency, security, privacy, and scalability are key considerations when designing a grouping-proof protocol. In recent research for RFID grouping-proofs, many protocols make assumptions that are not compatible with the technical capabilities for RFID. RFID tags are critically constrained in memory, communication, and computation. Recall from Chapter 2, Section 2 under the EPC Class 1 Generation 2 (C1G2) standard, passive tags can accommodate fewer than 3K gates to implement security features [40]. Some grouping-proof protocols, however, use hashing or encryption. Hash functions (for example, [16]) require 8K to 10K gates. Elliptic-curve cryptography (ECC), while less costly than alternatives such as AES and RSA, still requires 8K to 15K equivalent gates [100, 101].

In general, grouping proofs fall into two categories - serial and parallel grouping proofs.

- In serial grouping-proofs, the reader sends messages to the tags one tag at a time. The reader sends a message to a tag, the tag computes, the tag responds, then the reader moves on to the next tag. The reader constructs dependency among tags in a proof by having each tag incorporate a message sent by previous tag(s) into its own computation. The reader is the one forwarding all the messages from one tag to another.

- In parallel grouping-proofs, the reader broadcasts messages to all tags. All tags that receive the message compute in parallel, but communicate back to the reader one at a time.

Other categorizations are also common among RFID grouping-proof research.

- Offline and online grouping-proofs - In an RFID system, the verifier can be online or offline with respect to its communication to the reader. In the offline mode, the reader does not stay continuously connected to the verifier, and the reader will send its proof to the verifier when they next connect, which may be some time after it builds the proof. In contrast, in the online mode, the reader has a continuous connectivity with the verifier. In online mode, the reader does not have to construct a proof, as it can send information from each tag to the verifier as the information arrives. Offline mode is considered to be the challenging case.

- Static and dynamic groups - Static grouping-proofs establish only the presence of predefined groups of tags [86]. Dynamic grouping-proofs can handle any subsets of a larger group of authorized tags. One application of the dynamic grouping proofs is when the subgroups form a partition of a large group [63]. The demand to address such a need comes from the unreliable radio interference, as when the number of the tags of a group is larger, the probability of the interrogation failure grows. Partitioning into smaller subgroups is an advantage for the protocol in two ways: the interrogation process can abort early if there is an error, and in a second iteration the reader will need to interrogate only the subgroups for which no proof was generated.

- Reading order-dependent and reading order-independent - In a reading order-dependent protocol, the reader must interrogate tags in predetermined order. Reading order-independent protocols can operate regardless of the order of interrogating tags or receiving their responses.

Parallel grouping-proofs are primarily reading order-independent, though, there are some protocols [51, 63] that are reading order-dependent. Serial grouping-proofs are primarily reading order-dependent but can be reading order-independent. An example reading order-dependent serial grouping-proof [65] shows the following disadvantages - inefficiency, higher interrogation failure rates, and inability to provide acceptable privacy.

RFID communication is a sequential process and interrogation simultaneity can only be captured by an “exposure-time” window: events are considered as happening simultaneously if they take place within this window defined by the verifier [16, 87]. A grouping proof should prove simultaneity, a mechanism to do it is to guarantee causality. To guarantee causality, a proof must assure that a message input to a tag should be derived from computations that can only be carried out by other tag(s) participating in the proof.

3.2 Background Concepts of a Grouping Proof Protocol

Chapter 2 gives background on RFID systems. Recall that the protocol involves three types of entities: tags, a reader, and a verifier. Our work considers passive tags operating under the EPC Gen2 version 2 specification [21, 40]. The reader manages tag groups using three basic operations: select, inventory and access. In access, an authorized reader can put a tag into an open state such that the tag will not respond to any other reader when a protocol is executed. Though a tag does not have a clock, it can be programmed to respond to the reader for a bounded time period.

In each of our protocols the reader and tags use a pseudo-random number generator (PRNG). The verifier also uses a PRNG to verify the reader and tag computations (messages). Mandal *et al.* [74] and Martin *et al.* [76] present PRNGs suitable for EPC Gen2 version 2.

We assume that the wireless communication channel between a reader and a tag is not secure, while the channel between the reader and the verifier is secure. Entities do not trust readers and tags, but the verifier can authenticate readers, and readers and tags can authenticate each other using shared information loaded during initialization.

3.2.1 General Design Requirements for Grouping-Proof Protocols

Design requirements for our grouping-proof protocols include the following.

1. Establish "simultaneous" presence of tags in a group, meaning that the reader must communicate with all tags within a limited time window [88].

2. Construct a proof even if some tags are missing such that the proof identifies the tags that are present and omits the missing tags.

3. Detect tags that do not belong to the group and omit them from the grouping proof.

4. Keep reader and tags synchronized across multiple runs [73].

5. Prevent a second reader from interfering with a grouping proof once it has started [65].

6. Verify the integrity of messages exchanged between reader and a tag and between reader and verifier to prevent the generation of invalid proofs [106].

7. Ensure tag privacy - prevent an adversary from using information communicated during a run to track a tag in the future or identify past presence of a tag using information from previous runs of the protocol.

3.2.2 Grouping-Proof Problem Definition

The verifier securely loads information onto readers and tags during an initialization phase, but it is not connected to either reader or tags during the construction of the grouping-proof by a reader. (Hence, the reader constructs the grouping-proof offline.) The trigger for a reader to begin a grouping-proof protocol can be a pre-loaded time (such as in a warehouse application) or a user can start the proof process (such as in a supply-chain application). The reader is to determine which tags of a group of tags are simultaneously present within its range, and the reader is to construct a grouping-proof that it sends to the verifier. The verifier receives the grouping-proof from a reader at some time after the reader constructs this proof, and the verifier checks this proof for validity.

3.2.3 Adversary Model

We assume that the adversary has the ability to completely control the communication channel between the reader and tags. That is, the adversary can eavesdrop, modify, block (delete), delay, and replay any messages during transmission. The adversary can send its own messages to the reader (spoofing a tag) and to tags (spoofing the reader). The adversary can attempt to spoil a grouping proof in various ways, including generating proofs that an absent tag is present or vice versa and extracting information to allow it to track specific tags. At times, we will assume that the adversary has the ability to corrupt some tags, that is, extract the secrets and other information stored in a tag.

3.2.4 Definitions of the Attacks

Attacks on Privacy

Tracking attack: An adversary is able to trace and/or identify a tag in future communication.

Anonymity attack: After analyzing the transmitted messages between reader and tags, an adversary is able to differentiate specific tags' secrets or the reader's secrets.

Forward security attack: After intercepting or eavesdropping on a tag's communication during a protocol run and corrupting the tag to obtain its secrets, an adversary can identify that tag in previous runs of the protocol.

Message integrity attack: An adversary can modify messages to a reader/tag, and the reader/tag cannot detect it.

Attacks on Security

Impersonation:

- Impersonate tag to reader - An adversary captures tags' secrets or other tags' data that allow it to impersonate the tag to the reader.

- Impersonate reader to tag - An adversary sends a message to a tag that the tag accepts as if was from a legitimate reader, then either a proof fails or as a result the tag's privacy is compromised.

- Impersonate reader to verifier - An adversary captures the reader's secrets that allow it to impersonate the reader to the verifier.

Concurrency attack: While a grouping proof protocol is executing, additional readers execute the grouping proof protocol on the same group of tags, spoiling the current or future grouping proofs.

Desynchronization: An adversary causes tags to assume a state from which they can no longer participate in the protocol properly.

Denial of proof: When an adversary attempts to force a proof to fail on reader-end and/or tag-end in various ways.

Replay attack: After intercepting a tag's (or reader's) communication, an adversary replays that message intending to extract further information from the reader (or tag) or desynchronize the reader and tag.

In our work we are mainly concerned with security issues at the protocol and application layer. We are not concerned with physical or link layer issues, such as the coupling design and the power-up.

3.3 Formal Untraceability Analysis

Establishing proof of security and privacy in a clear and precise model is a major concern in the area of RFID protocols. The RFID research community has developed many such models. See Coisel and Martin [28] for a survey in privacy models. We choose to evaluate our protocols with the untraceability model presented by Avoine *et al.* [8].

This model captures different privacy levels of the protocols. Compared to other models, the model of Avoine *et al.* structures its attributes differently to enhance its capabilities and fairness to analyze the untraceability level of RFID protocols.

This untraceability model provides us with an easily applicable model that meaningfully evaluates the privacy of our protocols. The model uses adversary actions (oracles), capabilities (selectors and restrictions), and goals (experiment) and gives the flexibility to modify adversarial assumptions if needed.

Below we describe the Avoine *et al.* model. Their paper [8] provides more details.

3.3.1 Description of the Avoine *et al.* Untraceability Model

The RFID System in the Model

An RFID system $\mathcal{S}$ defines three building blocks (more details in [8]):

- The architecture comprises three kinds of entities: tags, readers, and a verifier (database). System architectures fall into two categories based on the numbers of readers: single-entity (SE) and multiple-entity (ME). In our untraceability analysis, we consider only SE (RFID system architecture with a single autonomous reader).

- Initialization procedures include generating system secrets, registering and initializing tags, and preauthorizing a reader $\mathcal{R}$ to execute the protocol on tags $\mathcal{T}$.

- Protocol Prot is a sequence of steps to achieve a well-defined objective. Prot involves one or more entities, where each entity executes its own algorithm to reach the protocol objective. An algorithm is the set of steps and transitions (an internal or external event required by an entity to move to the next step). When an entity executes an algorithm that exchanges data with other involved entities, call the record of exchanged data as a transcript. A transcript can include, for example, the reception/emission time of a message, message's issuer and/or recipient, etc. Each algorithm execution may further modify the entity's internal state.

The Untraceability Experiment

In the untraceability experiment UNT, the adversary $\mathcal{A}$ can interact with the system $\mathcal{S}$ within some limitations, which define different adversary classes. UNT also includes an honest entity, challenger $\mathcal{C}$, and challenge tags.

The notion of a challenge tag is a tag that is a target of an attack performed by $\mathcal{A}$ [8, 28]. $\mathcal{A}$ can choose challenge tags. $\mathcal{A}$ does not interfere with the challenger $\mathcal{C}$ (honest entity) during the experiment with the challenge tags.

In UNT experiment, at some point $\mathcal{A}$ selects two challenge tags, i and j. $\mathcal{A}$ has access to all the transcripts of communications to and from all tags, and it has the identities of which messages are to/from which tags. Challenger $\mathcal{C}$ relabels the two challenge tags as $\widetilde{i}$ and $\widetilde{j}$ with respect to a random bit b such that if $b = 0$, then $\widetilde{i} = i$ and $\widetilde{j} = j$ (that is, the labels remain the same), but if $b = 1$, then $\widetilde{i} = j$ and $\widetilde{j} = i$ (that is, $\mathcal{C}$ swaps the labels). $\mathcal{C}$ executes the protocol c_1 times on Tag $\widetilde{i}$ and c_2 times on Tag $\widetilde{j}$ without interference from $\mathcal{A}$, then provides the transcripts of these executions to $\mathcal{A}$. Depending on the untraceability class, $\mathcal{A}$ may or may not be able to corrupt some tags. $\mathcal{A}$ can run further executions of the protocol, interacting with the tags. Finally, $\mathcal{A}$ must determine the value of b. If the probability that $\mathcal{A}$ can correctly determine b is not very different from $\frac{1}{2}$, then the protocol is untraceable in that class. Below we quote the following two definitions from Avoine *et al.* [8].

Definition 1 *(P-**Universal untraceability**). An RFID system $\mathcal{S}$ is said P-Universal untraceable, denoted* P-Universal-UNT*, if:*

$\forall (c_1, c_2) \in \mathbb{N}^2$, $\forall \mathcal{A}_p \in \mathcal{P}, \forall b \in \{0, 1\}$:

$$\left| Pr\left(Exp^{UNT}_{\mathcal{S},\mathcal{A}_p}(\lambda, b, c_1, c_2) \longrightarrow 1 \right) - 1/2 \right| \leq \epsilon(\lambda),$$

where ϵ is a negligible function in the security parameter λ.

Definition 2 *(P-**Existential untraceability**). An RFID system $\mathcal{S}$ is said P-Existential untraceable, denoted* P-Existential-UNT*, if:*

$\exists (c_1, c_2) \in \mathbb{N}^2$, $\forall \mathcal{A}_p \in \mathcal{P}, \forall b \in \{0, 1\}$:

$$\left| Pr\left(Exp^{UNT}_{\mathcal{S},\mathcal{A}_p}(\lambda, b, c_1, c_2) \longrightarrow 1 \right) - 1/2 \right| \leq \epsilon(\lambda),$$

where ϵ is a negligible function in the security parameter λ.

Adversary Classes

The collection of attributes (actions, capabilities, goals) available to the adversary gives rise to the concept of adversary class. The Avoine *et al.* [8] model formalizes the actions that can be carried out on an RFID system with *oracles*. An *oracle* allows an adversary $\mathcal{A}$ to interact with the system $\mathcal{S}$ and collect data. Example oracles include the abilities to interact with the tags and reader by eavesdropping, blocking messages, tampering with messages, spoofing reader or tag, and corrupting a tag to obtain all of its secrets

Two typical adversary classes are $\mathsf{CLASSIC}$ and STRONG.

For the class $\mathsf{CLASSIC}$, $\mathcal{A}$ cannot corrupt any tags. The adversary $\mathcal{A}$ can interact with the tags and reader by eavesdropping, blocking messages, tampering with messages, and spoofing reader or tag.

For the class STRONG, $\mathcal{A}$ can corrupt any tags, including the challenge tags, and can interact with the tags and reader by eavesdropping, blocking messages, tampering with messages, spoofing reader or tag.

There are three classes defined between those two in the model - $\mathsf{FORWARD}$, $\mathsf{BACKWARD}$, and SIDE.

- $\mathsf{FORWARD}$ - Tags i and j cannot be corrupted, but $\widetilde{i}$, $\widetilde{j}$, and non-challenge tags can be corrupted.

- $\mathsf{BACKWARD}$ - Tags $\widetilde{i}$ and $\widetilde{j}$ cannot be corrupted, but i, j, and non-challenge tags can be corrupted.

- SIDE - Both $\mathsf{FORWARD}$ and $\mathsf{BACKWARD}$ restrictions must be respected, meaning that challenge tags cannot be corrupted but non-challenge tags can.

Universal and Existential Untraceabilities

The notions of $\mathsf{Universal\text{-}UNT}$ and $\mathsf{Existential\text{-}UNT}$ depend on restrictions about the choices of c_1, c_2 (parameters of the experiment). The restrictions are defined as follows.

- Universal-UNT - $\mathcal{C}$ executes the protocol and (c_1, c_2) can be any number of executions of the protocol, including (0, 0).

- Existential-UNT - $\mathcal{C}$ executes the protocol c_1 times on Tag $\widetilde{i}$ and c_2 times on Tag $\widetilde{j}$ for some specified numbers (c_1, c_2)

In Sections 4.4.2 and 5.4.2, we apply the untraceability model to provide a formal comprehensive evaluation of our protocols within the adversary classes defined above considering universal and existential untraceability.

CHAPTER 4
SERIAL - DEPENDENCY GROUPING PROOF PROTOCOL

4.1 Introduction

Motivated by the SDZ Protocol, we develop the Serial-Dependency Grouping Proof Protocol (SDGPP) that reduces the amount of communication and computation, addresses security gaps, removes the outside trusted timestamp server, and prevents a compromised tag from spoiling the entire proof. Our light version of the SDZ serial-dependency protocol preserves the security and privacy properties of the original protocol. Also, we use the EPC specified settling-time intervals for tags to limit the time duration of the protocol. We remove the dependency on a specific reader to make the protocol suitable for applications like a supply chain with different readers at different locations for the same group.

We analyzed untraceability of SDGPP in Section 4.4.2. Later, in Chapter 5 (Section 5.5) we will compare the security and performance of SDGPP to that of other protocols, including our PDGPP.

4.2 Motivation

Below is a review of the Sundaresan *et al.* [106] (SDZ) grouping-proof protocol in detail, because the paper's protocol motivated our work.

The Sundaresan *et al.* [106] (SDZ) grouping-proof protocol uses secrets shared by reader and tags and other group- and tag-specific secrets held by tags to perform mutual authentication as well as verify the integrity of the exchanged messages. The SDZ protocol builds around a "zero-knowledge property" that enables a tag to convince the verifier of its presence without revealing its identity or any tracking information to the reader or the adversary. The essence of this technique is as follows. The tag selects one bit of a secret shared with

Part of this chapter was previously published as Vanya Cherneva and Jerry Trahan, (2018)"Serial-Dependency Grouping-Proof Protocol for RFID EPC C1G2 Tags," 2018 IEEE Green Energy and Smart Systems Conference (IGESSC) the verifier. The tag generates a random number r_i, then, depending on the value of the bit of the shared secret, transmits either the square ($r_i^2 \mod p$) or the pseudosquare ($w \cdot r_i^2 \mod p$). The verifier uses the prime factors of modulus p to distinguish square from pseudosquare, while it is computationally infeasible for the reader or adversary to do so without these factors. The protocol assumes a trusted timestamp-server that provides an encrypted timestamp to the reader before each communication with a tag. Unlike the offline verifier, this timestamp-server must remain connected to the reader. During the initialization phase, the verifier loads information on reader and tags. The reader receives values including reader-specific RH, group-specific GH, and tag-specific values for u protocol runs for each tag and each group. Each tag receives values including RH, GH, and tag-specific $TH(i)$. During the proof generation phase, the reader sends information to a tag, receives a response from that tag, then moves on to the next tag in turn. The information sent to Tag i includes $K(i-1)$ calculated by the preceding tag, Tag $i-1$. Folding together a fresh random number, secrets shared with Tag i, and information loaded by the verifier (that itself folds in secrets shared with Tag i), the reader sends eight items of data to the tag. Tag i extracts the random number and then extracts GH and $TH(i)$. If they match stored values of GH and $TH(i)$, then the tag has authenticated the reader and confirmed the integrity of the data sent by the reader. If they do not match, then the tag tries again using stored values from its previous run of the protocol. (This permits a tag one run out of synchronization with the reader to regain synchronization. The update scheme prevents a tag from getting more than one run out of synchronization with the reader in some, but not all, circumstances.) The tag uses the zero-knowledge technique to compute a value $K(i)$ to send to the reader then builds a value $L(i)$ including this $K(i)$ and a stored hash RH of the reader ID. Receiving $K(i)$ and $L(i)$, the reader extracts RH to authenticate the tag and ensure the integrity of the message. During the proof construction phase, the reader collects RH, GH, run number, the random number and encrypted timestamp used for each tag, and the $K(i)$ and $L(i)$ values received from tags. It then encrypts this collection using a secret shared with the verifier.

The SDZ protocol has many strong points. These include use of a hashed group ID to permit the reader to authenticate a tag without divulging the tag's ID, the use of random numbers to keep communications fresh across runs and prevent an eavesdropper from tracking, the use of the zero-knowledge technique, and checking the integrity of each communication. The SDZ protocol, however, also has weak points. (i) The protocol assumes the presence of a trusted timestamp-server always connected to the reader. (ii) A tag can participate only with a specific reader. (iii) An adversary can desynchronize a tag by blocking its messages in consecutive runs. (iv) Checking extracted vs. stored values for all of RH, GH, and $TH(i)$ is redundant. (v) Reader and tags exchange many data items. (vi) A tag does not check the integrity of the K value sent to it by the reader. We have proposed a protocol that addresses these points.

4.3 Description of SDGPP

The protocol has three phases: initialization, grouping-proof collection, and proof-verification. Table 4.1 describes notation used in the protocols.

4.3.1 Initialization Phase

In the initialization phase, the verifier pre-computes information for u protocol runs and stores this information in the reader and also stores one set of values in each tag.

Reader: The verifier initializes the reader with the following values: independent of group and run - $\{RH, s_R, K_{rv}\}$, where K_{rv} is a secret key shared between verifier and the reader; group-specific - $\{GH, nextrun(1 \dots m)\}$, where each $nextrun(i) = 1$ and m is the number of tags in the group; group- and run-specific - $\{J, \mu(1 \dots m), s_{RT}(1 \dots m)\}$. The reader starts with $seed_R$ initiated by verifier (can be once or specific to a group or specific to a run on a group or specific to a tag in a run on a group).

Tags: The verifier initializes each Tag i with the following values: independent of group and run - $\{TH(i)\}$; group-specific - $\{GH, s_G, s_T(i), p, w, c_V\}$; group- and run-specific

- $\{s_{VT}(i), s_{VT}^{-1}(i),\ s_{RT}(i), s_{RT}^{-1}(i)\}$, where $s_{VT}^{-1}(i), s_{RT}^{-1}(i)$ are initialized to 0. Tag i starts a run with $seed_T(i)$ for its PRNG, initiated by verifier. Tags store no values specific to an individual reader (as opposed to the SDZ protocol). This modification allows the protocol to be processed from any reader pre-authorized from the verifier. Our protocol uses the timing requirement built in EPC GS1 version 2 [40]. Tags execute any command (in any state) within a certain default time controlled by the c_V preset timing variable. Tags must respond in this time window to be part of the grouping proof.

Table 4.1. Notation for SDGPP

Notations	Description
s_{V1}, s_{V2}, s_{V3}	Unique secrets known only to the verifier
$GID(k), GH$	Group ID $GH(k) = h(GID(k), s_{V1})$, where h is a hash function
$TID(i), TH(i)$	Unique Tag ID $TH(i) = h(TID(i), s_{V2})$
$RID(j), RH$	Unique reader ID $RH(j) = h(RID(j), s_{V3})$
r_V	Run-specific PRN generated by the verifier
J	$J = GH \oplus PRNG(s_G \oplus r_V)$, where $PRNG$ is a pseudorandom number generator
$\mu(i)$	$\mu(i) = PRNG(TH(i) \oplus s_{VT}(i)) \oplus (x\|\|y\|\|z\|\|r_V)$
$seed_R$	seed for the reader PRNG function
$r_R(i)$	PRN generated by the reader for Tag i
s_R	Secret key unique to each reader in the system
s_G	Shared group secret among group tags, $s_G(k)$
$s_{VT}(i), s_{VT}^{-1}(i)$	$s_{VT}(i)$ ($s_{VT}^{-1}(i)$) is a secret shared between verifier and Tag i for current (previous) run
$s_{RT}(i), s_{RT}^{-1}(i)$	$s_{RT}(i)$ ($s_{RT}^{-1}(i)$) is a secret shared between reader and Tag i for current (previous) run
$s_T(i)$, $s_T(i, x)$	Secret key unique to Tag i; $s_T(i, x)$ is 128-bit string and x is index of 128-bit string in $s_T(i)$
$seed_T(i)$	seed for Tag i PRNG function
$r_T(i, index)$	PRN generated by Tag i
x, y, z	Indices used in zero-knowledge proof
p, w	Values for zero-knowledge proof
c_V	Time constant to limit activity in a protocol run

4.3.2 Grouping-Proof Collection Phase

Figure 4.1 is a view of the messages exchanged among reader and tags in SDGPP. During the grouping-proof collection phase, the verifier does not connect to the reader or tags. When

Reader	
$\longrightarrow J_r(1), \mu_r(1), \delta(1)$ to Tag 1	
	Tag 1
	$\longleftarrow K(1), L(1)$ to Reader
Reader	
$\longrightarrow J_r(2), K_{last}, \mu_r(2), \delta(2)$ to Tag 2	
	Tag 2
	$\longleftarrow K(2), L(2)$ to Reader
...	...
...	...
...	...
$\longrightarrow J_r(i), K_{last}, \mu_r(i), \delta(i)$ to Tag i	
	Tag i
	$\longleftarrow K(i), L(i)$ to Reader
...	...
...	...
	Tag m
	$\longleftarrow K(m), L(m)$ to Reader

Figure 4.1. SDGPP messages

in a run, the reader places all tags in a group in open state. This starts the time window controlled by c_V.

Reader - Round 1
use J, $\mu(1)$, $s_{RT}(1)$ from $nextrun(1)$
$r_R(1) \leftarrow PRNG(seed_R)$
$seed_R \leftarrow r_R(1) \oplus s_R$
$J_r(1) \leftarrow J \oplus r_R(1)$
$\mu_r(1) \leftarrow \mu(1) \oplus r_R(1)$
$\delta(1) \leftarrow s_{RT}(1) \oplus r_R(1)$
Send to Tag 1: $J_r(1)$, $\mu_r(1), \delta(1)$

Figure 4.2. Round 1

Round 1 - Reader

The reader uses a run value $nextrun(1)$ on which the reader expects Tag 1 to be. The reader selects values for J, $\mu(1)$, and $s_{RT}(1)$ for Tag 1 according to this run value. The reader generates a fresh pseudo-random number $r_R(1)$ and XORs this with precomputed values to produce $J_r(1), \mu_r(1), \delta(1)$. The randomness from $r_R(1)$ keeps messages fresh across runs and avoids sending values in the clear. The reader sends three values to Tag 1 compared to seven in the SDZ protocol.

```
**Tag 1 - Round 1**
$synch \leftarrow 1$
$r_R(1) \leftarrow \delta(1) \oplus s_{RT}(1)$
 ; Use extracted $r_R(1)$ to remove randomness
$J \leftarrow J_r(1) \oplus r_R(1)$
$\mu(1) \leftarrow \mu_r(1) \oplus r_R(1)$
$x||y||z||r_V \leftarrow PRNG(TH(1) \oplus s_{VT}(1)) \oplus \mu(1)$
 ; Tag 1 uses stored $TH(1), s_{VT}(1)$ to obtain $r_V$
$GH \leftarrow J \oplus PRNG(s_G \oplus r_V)$
 ; $r_V$ obtained from $\mu(1)$ used to extract $GH$ from $J$
if (extracted $GH \neq$ stored $GH$)
 ; if equal, then Group ID verified
 ; and message integrity of $J$ and $\mu(1)$ verified
 ; else reader and Tag 1 runs not synchronized
 ; so use $s_{RT}^{-1}(1), s_{VT}^{-1}(1)$ and try again
        $synch \leftarrow 0$
        $r_R(1) \leftarrow \delta(1) \oplus s_{RT}^{-1}(1)$
        $J \leftarrow J_r(1) \oplus r_R(1)$
        $\mu(1) \leftarrow \mu_r(1) \oplus r_R(1)$
        $x||y||z||r_V \leftarrow PRNG(TH(1) \oplus s_{VT}^{-1}(1)) \oplus \mu(1)$
        $GH \leftarrow J \oplus PRNG(s_G \oplus r_V)$
        if (extracted $GH \neq$ stored $GH$)
                abort
$S(1) \leftarrow s_T(1, x) \oplus PRNG(TH(1) \oplus s_T(1, y))$
            $\oplus PRNG(s_T(1, z) \oplus r_V)$
$K(1) \leftarrow null$
for $index \in \{x, y, z\}$
        $r_T(1, index) \leftarrow PRNG(seed_T(1))$
        $seed_T(1) \leftarrow r_T(1, index)$
        if $S(1)[index] = 0$
            $Z \leftarrow (r_T(1, index))^2 \bmod p$
        else ; that is, $S(1)[index] = 1$
            $Z \leftarrow w \cdot (r_T(1, index))^2 \bmod p$
        $K(1) \leftarrow K(1)||Z$
$L(1) \leftarrow GH \oplus PRNG((J_r(1)||K(1)) \oplus r_R(1))$
$seed_T(1) \leftarrow PRNG(seed_T(1))$
if $synch = 1$
        $s_{RT}^{-1}(1) \leftarrow s_{RT}(1)$
        $s_{RT}(1) \leftarrow PRNG(s_{RT}(1))$
        $s_{VT}^{-1}(1) \leftarrow s_{VT}(1)$
        $s_{VT}(1) \leftarrow PRNG(s_{VT}(1))$
Send to reader: $K(1), L(1)$
```

Figure 4.3. Round 1

Round 1 - Tag 1

Tag 1 authenticates the reader, verifies the integrity of the incoming messages, and computes its response using zero-knowledge techniques. Tag 1 performs the following steps. It authenticates that the reader is authorized by the verifier and confirms the integrity of messages received from the reader. If the verification fails, Tag 1 tries again using a different set of values from the previous run because the tag may be at most one run ahead of (out of synchronization with) the reader, so this allows it to resynchronize. Tag 1 has a *synch* flag that marks whether it uses values from its current or previous run. Tag 1 next computes values that will establish its presence to the verifier. Our protocol uses the zero knowledge property from SDZ described in Section 4.2. Tag 1 sends $K(1)$ and $L(1)$ to the reader. This is two values compared to three in the SDZ protocol.

Round i - Reader

In a round i, where $2 \leq i \leq m$, if the reader does not receive a response from Tag $i-1$ within a defined time window, it moves on to Tag i, leaving Tag $i-1$ marked as not valid.

```
Reader - Round $i$ $(2 \leq i \leq m)$
if reader received $K(i-1)$, $L(i-1)$ from Tag $i-1$
    $GH \leftarrow L(i-1) \oplus PRNG((J_r(i-1)||K(i-1)) \oplus r_R(i-1))$
    if (extracted $GH$ = stored $GH$))
        $nextrun(i-1) = nextrun(i-1) + 1$
        $K_{last} \leftarrow K(i-1)$
  ; $K_{last}$ is the last good $K$ value received from a tag
        mark Tag $i-1$ as valid
    else
        discard $K(i-1)$, $L(i-1)$
  ; if reader did not receive $K(1)$, $L(1)$ from Tag 1
  ; or if $K(1)$, $L(1)$ did not extract proper $GH$,
  ; then reader treats next tag as "Tag 1"
use $J$, $\mu(i)$, $s_{RT}(i)$ from $nextrun(i)$
$r_R(i) \leftarrow PRNG(seed_R)$
$seed_R \leftarrow r_R(i) \oplus s_R$
$J_r(i) \leftarrow J \oplus K_{last} \oplus r_R(i)$
$\mu_r(i) \leftarrow \mu(i) \oplus r_R(i)$
$\delta(i) \leftarrow s_{RT}(i) \oplus r_R(i)$
Send to Tag $i$: $J_r(i), K_{last}, \mu_r(i), \delta(i)$
```

Figure 4.4. Round i – Reader

```
Tag $i$ - Round $i$ $(2 \leq i \leq m)$
$synch \leftarrow 1$
$r_R(i) \leftarrow \delta(i) \oplus s_{RT}(i)$
        ; Use extracted $r_R(i)$ to remove randomness
$J \leftarrow J_r(i) \oplus K_{last} \oplus r_R(i)$
$\mu(i) \leftarrow \mu_r(i) \oplus r_R(i)$
$x||y||z||r_V \leftarrow PRNG(TH(i) \oplus s_{VT}(i)) \oplus \mu(i)$
 ; Tag $i$ uses stored values $TH(i), s_{VT}(i)$ to obtain $r_V$
$GH \leftarrow J \oplus PRNG(s_G \oplus r_V)$
 ; $r_V$ obtained from $\mu(i)$ used to extract $GH$ from $J$
if (extracted $GH \neq$ stored $GH$)
 ; if equal, then Group ID verified
 ; and message integrity of $J$, $K_{last}$, and $\mu(i)$ verified
 ; else reader and Tag $i$ runs not synchronized,
 ; so use $s_{RT}^{-1}(i), s_{VT}^{-1}(i)$ and try again
        $synch \leftarrow 0$
        $r_R(i) \leftarrow \delta(i) \oplus s_{RT}^{-1}(i)$
        $J \leftarrow J_r(i) \oplus K_{last} \oplus r_R(i)$
        $\mu(i) \leftarrow \mu_r(i) \oplus r_R(i)$
        $x||y||z||r_V \leftarrow PRNG(TH(i) \oplus s_{VT}^{-1}(i)) \oplus \mu(i)$
        $GH \leftarrow J \oplus PRNG(s_G \oplus r_V)$
        if (extracted $GH \neq$ stored $GH$)
                abort
$S(i) \leftarrow s_T(i,x) \oplus PRNG(TH(i) \oplus s_T(i,y))$
            $\oplus PRNG(K_{last} \oplus s_T(i,z) \oplus r_V)$
$K(i) \leftarrow null$
for $index \in \{x,y,z\}$
        $r_T(i,index) \leftarrow PRNG(seed_T(i))$
        $seed_T(i) \leftarrow r_T(i,index)$
        if $S(i)[index] = 0$
            $Z \leftarrow (r_T(i,index))^2 \bmod p$
        else ; that is, $S(i)[index] = 1$
            $Z \leftarrow w \cdot (r_T(i,index))^2 \bmod p$
        $K(i) \leftarrow K(i)||Z$
$L(i) \leftarrow GH \oplus PRNG((J_r(i)||K(i)) \oplus r_R(i))$
$seed_T(i) \leftarrow PRNG(seed_T(i))$
if $synch = 1$
        $s_{RT}^{-1}(i) \leftarrow s_{RT}(i)$
        $s_{RT}(i) \leftarrow PRNG(s_{RT}(i))$
        $s_{VT}^{-1}(i) \leftarrow s_{VT}(i)$
        $s_{VT}(i) \leftarrow PRNG(s_{VT}(i))$
Send to reader: $K(i), L(i)$
```

Figure 4.5. Round i – Tag i

If the reader does receive a response from Tag $i-1$, it extracts the hashed Group ID GH. If the extracted GH matches the stored GH, this authenticates Tag $i-1$ and also confirms the integrity of the messages $K(i-1)$ and $L(i-1)$. Otherwise, the response is not valid and the reader discards $K(i-1)$, $L(i-1)$. For the case of Tag 1, if the reader did not receive $K(1)$, $L(1)$ or extract the proper GH, then it treats the next tag as "Tag 1". The reader maintains a value K_{last}, which is the last valid K value received from a tag. If the response from Tag $i-1$ was valid, then the reader updates K_{last}. The reader acts for Tag i much as it did for Tag 1. Differences involve K_{last}: the reader XORs it into $J_r(i)$ to allow an integrity check and sends it directly sent to Tag i. The reader sends four values to Tag i versus eight in the SDZ protocol.

Round i - Tag i

Tag i's actions are much like those of Tag 1 with differences involving K_{last}.

4.3.3 Proof - Verification

The reader compiles all partial proofs from valid tags' messages to construct the proof $P = \{RH, GH, run, (i, nextrun(i), K(i), L(i), r_R(i))\}$ with information from all valid tags i. Then the reader encrypts proof P as P' using the secret key K_{rv} shared with the verifier. The reader sends P' to the verifier either immediately or at a later time with more proofs.

Verification Phase

The verifier decrypts P' using the shared secret key K_{rv}. Using RH, GH, and run, the verifier identifies which reader sent the proof, the tag group, and the run of the protocol for that group. For each Tag i, the verifier computes $S(i)$. Using $K(i)$, p, and w, the verifier distinguishes between squares and pseudosquares and decodes the tag- and run-specific bits. If the decoded bits match in $S(i)$, the verifier confirms that Tag i has participated in the proof.

4.4 Security Analysis

4.4.1 Security and Privacy Properties

We now establish that our protocol resists the attacks that the SDZ protocol [106] targets. See Sundaresan *et al.* [106] for a comparison to other protocols. For definitions of these attacks, see Section 3.2.4 and [15, 73, 87, 100, 106, 122, 127].

Privacy: The reader and tags never send Tag ID and group ID information in the clear but always as part of a value computed with XOR or PRNG. The reader XORs values sent to a tag with a new random value each run, even if $nextrun(i)$ does not change. The $K(i)$ and $L(i)$ values sent by a tag change each run because they build on fresh random numbers. So an adversary cannot track the same tag from the information exchanged from one run to the next. Attacks and other considerations relevant to privacy include tracking attack, anonymity attack, and forward security. Section 4.4.2 gives a more thorough analysis of untraceability and forward security.

Impersonation attacks: Impersonation attacks include three variants: (i) impersonate tag to reader; (ii) impersonate reader to tag; and (iii) impersonate reader to verifier. (Attacks under this category include forgery, illegitimate tag, subset replay, and multi-proof-session attacks.) (i) Without GH and $r_R(i)$, an adversary cannot construct an $L(i)$ value that a reader will accept when the reader checks the match between extracted and stored GH. (ii) Without $s_{RT}(i)$, $s_{VT}(i)$, $TH(i)$, and s_G, an adversary cannot construct values that Tag i will accept when it checks the match between extracted and stored GH. (iii) The reader sends proof P in encrypted form P' using a key shared with the verifier, so without this key, the adversary cannot send a P' that the verifier will decode into valid information. Furthermore, besides this encryption, an adversary cannot insert into a proof P information from a missing tag because the chaining of K values would not be consistent. Observe that an adversary cannot extract GH, $r_R(i)$, $s_{RT}(i)$, etc. from transmitted values.

Concurrency attack: This attack is when multiple readers simultaneously try to generate grouping proofs for the same tags. From the RFID specification [40], once a tag is in open state to one reader, it does not respond to others. So, after a reader places a tag in open state at the start of a run, the tag will complete its participation in this grouping proof before responding to another reader.

Desynchronization: For each Tag i, the reader tracks $nextrun(i)$, which is the run at which it expects the tag to be during the next proof. Each tag maintains $s_{RT}(i)$ and $s_{VT}(i)$ for the current run and $s_{RT}^{-1}(i)$ and $s_{VT}^{-1}(i)$ from its previous run. With selective updates of these, a tag can be at most one run ahead of the reader even if the adversary blocks messages. In that circumstance, the tag uses $s_{RT}^{-1}(i)$ and $s_{VT}^{-1}(i)$ to successfully compute its outputs.

Denial of proof: An adversary can attempt to force a proof to fail in various ways, and the protocol counters each of these. It can block or tamper with messages to desynchronize reader and tags (see above for resolution), it can block or tamper with messages to break the chain across tags (the use of K_{last} from the last valid tag resolves this), and it can impersonate illegitimate tags (see above for resolution).

Replay attack: If an adversary replayed a message from a Tag i to the reader, the reader would find it not valid because it will not incorporate the current $J_r(i-1)$ and $r_R(i-1)$ values. If an adversary replayed a message from the reader to Tag i from more than one run previously, then Tag i would abort. But for a replay attack from the immediately previous run for Tag i, the tag would not detect the replay attack, as it would see this as an instance of being one run ahead of the reader. Tag i would reply with fresh values built on fresh pseudorandom numbers (see above under privacy), and it would not update anything, so this would not cause any future harm to our protocol. (The same applies to the SDZ protocol.)

Message Integrity attack: A tag extracts GH from $\delta(i)$, $\mu_r(i)$, K_{last}, and $J_r(i)$. A reader extracts GH from $K(i)$ and $L(i)$. In both cases, a match between extracted and stored GH

confirms the integrity of received messages.

4.4.2 Untraceability Analysis of SDGPP

In this section, we classify SDGPP in the untraceability model of Avoine *et al.* [8]. We will establish that SDGPP is SIDE-$\mathsf{Universal}$-UNT and $\mathsf{FORWARD}$-$\mathsf{Existential}$-UNT but is not $\mathsf{FORWARD}$-$\mathsf{Universal}$-UNT.

Recall from Section 3.3.1 the following. $\mathcal{A}$ selects two challenge tags, i and j. Challenger $\mathcal{C}$ relabels the two challenge tags as $\widetilde{i}$ and $\widetilde{j}$ with respect to a random bit b such that if $b = 0$, then $\widetilde{i} = i$ and $\widetilde{j} = j$ (that is, the labels remain the same), but if $b = 1$, then $\widetilde{i} = j$ and $\widetilde{j} = i$ (that is, $\mathcal{C}$ swaps the labels). $\mathcal{A}$ must determine the value of b. If the probability that $\mathcal{A}$ can correctly determine b is not very different from $\frac{1}{2}$, then the protocol is untraceable in that class.

For the class SIDE-$\mathsf{Universal}$-UNT, $\mathcal{A}_{\mathsf{SIDE}}$ cannot corrupt the challenge tags, but can corrupt any other tags immediately after $\mathcal{C}$ relabels the tags or after any number of executions of the protocol. That is, (c_1, c_2) can be any values, including (0, 0).

For the class $\mathsf{FORWARD}$-$\mathsf{Existential}$-UNT, $\mathcal{A}_{\mathsf{FORWARD}}$ can corrupt the challenge tags and any other tags after $\mathcal{C}$ executes the protocol c_1 times on Tag $\widetilde{i}$ and c_2 times on Tag $\widetilde{j}$ for some specified numbers (c_1, c_2).

For the class $\mathsf{FORWARD}$-$\mathsf{Universal}$-UNT, $\mathcal{A}_{\mathsf{FORWARD}}$ can corrupt the challenge tags and any other tags immediately after $\mathcal{C}$ relabels the tags or after any number of executions of the protocol. That is, (c_1, c_2) can be any values, including (0, 0).

Theorem 4.4.1 *SDGPP is* SIDE-$\mathsf{Universal}$-UNT.

Proof. For the SIDE-$\mathsf{Universal}$-UNT class, adversary $\mathcal{A}_{\mathsf{SIDE}}$ cannot corrupt the challenge tags but can corrupt any other tags. $\mathcal{A}_{\mathsf{SIDE}}$ can also obtain transcripts of communications to and from all tags in all executions of the protocol.

For universal untraceability, (c_1, c_2) can be any values, and proving untraceability with

$(c_1, c_2) = (0, 0)$ suffices [8]. That is, when $\mathcal{A}_{\mathsf{SIDE}}$ has selected Tags i and j, it has access to all transcripts for all tags in all previous executions of SDGPP, and it has the information on which of these transcripts belong to which tags, including i and j. After the last execution of SDGPP, $\mathcal{A}_{\mathsf{SIDE}}$ can corrupt all non-challenge tags, and $\mathcal{A}_{\mathsf{SIDE}}$ receives all transcripts of communications, however, the Challenger labels the transcripts for i and j in the last execution as belonging to $\widetilde{i}$ and $\widetilde{j}$ according to a random selection by the Challenger.

To establish that $i = \widetilde{j}$, $\mathcal{A}_{\mathsf{SIDE}}$ would have to establish that Tag i in one of the previous executions is the same as Tag $\widetilde{j}$. To do this, $\mathcal{A}_{\mathsf{SIDE}}$ would have to calculate $K(i)$ or $L(i)$ from the information it has obtained from transcripts and from corrupted non-challenge tags using information that matches Tag $\widetilde{j}$. From the communication transcripts, $\mathcal{A}_{\mathsf{SIDE}}$ holds that Tag i received inputs $J_r(i)$, K_{last} (for i), $\mu_r(i)$, $\delta(i)$ and produced outputs $K(i)$, $L(i)$, while Tag $\widetilde{j}$ received inputs $J_r(\widetilde{j})$, K_{last} (for $\widetilde{j}$), $\mu_r(\widetilde{j})$, $\delta(\widetilde{j})$ and produced outputs $K(\widetilde{j})$, $L(\widetilde{j})$. For each non-challenge Tag q, $\mathcal{A}_{\mathsf{SIDE}}$ can extract all its stored information (that is, $TH(q)$, GH, s_G, $s_T(q)$, p, w, c_V, $s_{VT}(q)$, $s_{VT}^{-1}(q)$, $s_{RT}(q)$, $s_{RT}^{-1}(q)$, and $seed_T(q)$). Also, for Tag q, $\mathcal{A}_{\mathsf{SIDE}}$ can calculate all values in the corresponding run of SDGPP for Tag q up through $S(q)$ (that is, $r_R(q)$; J for q; $\mu(q)$; x, y, z, and r_V for q; and $S(q)$).

Even if $i = \widetilde{j}$ and even if the run used $s_{RT}^{-1}(\widetilde{j}) = s_{RT}(i)$, $K(\widetilde{j})$ will be different from $K(i)$. This is because Tag $\widetilde{j}$ generated fresh pseudorandom $r_T(\widetilde{j}, index)$ values to build $K(\widetilde{j})$. Correspondingly, $L(\widetilde{j})$ will be different from $L(j)$ again due to different pseudorandom values $K(\widetilde{j})$ and $r_R(\widetilde{j})$. Secrets and calculations from non-challenge tags do not change this.

Because $\mathcal{A}_{\mathsf{SIDE}}$ does not have $s_{RT}(i)$, it cannot extract $r_R(i)$. Also, $\mathcal{A}_{\mathsf{SIDE}}$ cannot calculate $r_R(i)$ from $r_R(i-1)$ because of how the reader changes the seed used in the pseudorandom generation of $r_R(i)$. So, $\mathcal{A}_{\mathsf{SIDE}}$ cannot extract x, y, z, and r_V and so cannot calculate $S(i)$ and so cannot calculate $K(i)$. Also, even with GH obtained by corruption of other tags, $J_r(i)$ obtained from the transcript, and $K(i)$ obtained from the transcript, $\mathcal{A}_{\mathsf{SIDE}}$ cannot calculate $L(i)$ without $r_R(i)$. Therefore, $\mathcal{A}_{\mathsf{SIDE}}$ cannot match Tag $\widetilde{j}$ with Tag i any better than it can match Tag $\widetilde{i}$. □

Theorem 4.4.2 *SDGPP is* FORWARD-Existential-UNT*.*

Proof. For the FORWARD-Existential-UNT class, adversary $\mathcal{A}_{\mathsf{FORWARD}}$ can corrupt all tags, including the challenge tags, after the Challenger executes the protocol without interference from $\mathcal{A}_{\mathsf{FORWARD}}$ for a specific number of times. In this proof, we will use $(c_1, c_2) = (2, 2)$, meaning that the Challenger will execute SDGPP for two full executions before $\mathcal{A}_{\mathsf{FORWARD}}$ corrupts any tags. $\mathcal{A}_{\mathsf{FORWARD}}$ can also obtain transcripts of communications to and from all tags in all executions of the protocol.

To establish that $i = \widetilde{j}$, $\mathcal{A}_{\mathsf{FORWARD}}$ would have to establish that Tag i in one of the previous executions is the same as Tag $\widetilde{j}$ in the last execution. To do this, $\mathcal{A}_{\mathsf{FORWARD}}$ would have to calculate $K(i)$ or $L(i)$ from the information it has obtained from transcripts and from corrupted tags. The proof of Theorem 4.4.1 describes the relevant information from transcripts and from corruption, but in the current proof, the corrupted tags also include challenge tags $\widetilde{i}$ and $\widetilde{j}$.

In the selected execution of Tag i, let $s_{RT}(i) = \beta$ and $s_{RT}^{-1}(i) = \alpha$ at the end of the run. Whether Tag i was synchronized or desynchronized with the reader at the start of the run, it would have used α as the s_{RT} value during that run. We will examine cases in which $i = \widetilde{j}$ and the two executions of SDGPP by the Challenger immediately follow the selected execution of Tag i. If other executions are between these, then the conclusions still hold.

Consider the case where Tag i is synchronized with the reader at the end of its run, which means that Tag $\widetilde{j}$ is synchronized at the start of the first execution by the Challenger. This will be the case if $\mathcal{A}_{\mathsf{FORWARD}}$ allows the message from Tag i to reach the reader intact. The first execution by the Challenger on Tag $\widetilde{j}$ will use s_{RT} value β. At the end of that execution, $s_{RT}(\widetilde{j}) = \gamma$ and $s_{RT}^{-1}(\widetilde{j}) = \beta$. Tag $\widetilde{j}$ no longer holds the value α.

Consider next the case where Tag i is desynchronized with the reader at the end of its run, which means that Tag $\widetilde{j}$ is desynchronized at the start of the first execution by the Challenger. This will be the case if $\mathcal{A}_{\mathsf{FORWARD}}$ blocks the message from Tag i from reaching the reader or tampers with the message. The first execution by the Challenger on Tag $\widetilde{j}$ will use s_{RT} value α. At the end of that execution, $s_{RT}(\widetilde{j}) = \beta$ and $s_{RT}^{-1}(\widetilde{j}) = \alpha$ because the tag does not update these in a desynchronized execution. Tag $\widetilde{j}$ is now synchronized with the reader. The second execution by the Challenger on Tag $\widetilde{j}$ will use s_{RT} value β. At the end of that execution, $s_{RT}(\widetilde{j}) = \gamma$ and $s_{RT}^{-1}(\widetilde{j}) = \beta$. Tag $\widetilde{j}$ no longer holds the value α.

In both cases, by the arguments above in the proof of Theorem 4.4.1, because the corrupted Tag $\widetilde{j}$ does not hold s_{RT} value α used by Tag i, $\mathcal{A}_{\mathsf{FORWARD}}$ cannot determine $r_R(i)$, so it cannot calculate either $K(i)$ or $L(i)$. Therefore, $\mathcal{A}_{\mathsf{FORWARD}}$ cannot match Tag $\widetilde{j}$ with Tag i any better than it can match Tag $\widetilde{i}$. □

Theorem 4.4.3 *SDGPP is not* FORWARD-Universal-UNT.

Proof. For the FORWARD-Universal-UNT class, adversary $\mathcal{A}_{\mathsf{FORWARD}}$ can corrupt all tags, including the challenge tags, after the Challenger executes the protocol without interference from $\mathcal{A}_{\mathsf{FORWARD}}$ for any adversary-specified number of times (including zero). In this proof, we will use $(c_1, c_2) = (1, 1)$, meaning that the Challenger will execute SDGPP for one full execution before $\mathcal{A}_{\mathsf{FORWARD}}$ corrupts any tags. $\mathcal{A}_{\mathsf{FORWARD}}$ can also obtain transcripts of communications to and from all tags in all executions of the protocol.

Suppose that $i = \widetilde{j}$. To establish that $i = \widetilde{j}$, $\mathcal{A}_{\mathsf{FORWARD}}$ will have to establish that Tag i in one of the previous executions is the same as Tag $\widetilde{j}$ in the last execution. To do this, $\mathcal{A}_{\mathsf{FORWARD}}$ will calculate $L(i)$ from the information it has obtained from transcripts and from corrupted tags. The proof of Theorem 4.4.1 describes the relevant information from transcripts and from corruption, but in the current proof, the corrupted tags also include challenge tags $\widetilde{i}$ and $\widetilde{j}$.

During the selected execution with Tags i and j, $\mathcal{A}_{\mathsf{FORWARD}}$ blocks messages from Tags i and j to the reader (or tampers with these messages), so that Tags i and j are desynchronized with the reader at the end of the execution. The Challenger then immediately executes SDGPP once, then $\mathcal{A}_{\mathsf{FORWARD}}$ corrupts the challenge Tags $\widetilde{i}$ and $\widetilde{j}$.

In the selected execution of Tag i, let $s_{RT}(i) = \beta$ and $s_{RT}^{-1}(i) = \alpha$ at the end of the run. Whether Tag i was synchronized or desynchronized with the reader at the start of the run, it would have used α as the s_{RT} value during that run. Because the execution by the challenger $\mathcal{C}$ begins with Tags $\widetilde{i}$ and $\widetilde{j}$ desynchronized with the reader, neither Tag $\widetilde{i}$ nor Tag $\widetilde{j}$ updates its s_{RT} values. So, $s_{RT}^{-1}(i) = s_{RT}^{-1}(\widetilde{j}) = \alpha$. Using $s_{RT}^{-1}(\widetilde{j}) = \alpha$ and $\delta(1)$ from the transcript of Tag i in the selected run, $\mathcal{A}_{\mathsf{FORWARD}}$ extracts $r_R(i)$ used by Tag i. Then with GH (via corruption), $J_r(i)$ (via transcript), $K(i)$ (via transcript), and $r_R(i)$ (just calculated), $\mathcal{A}_{\mathsf{FORWARD}}$ calculates $L(i)$ and matches it with the $L(i)$ value in the transcript of messages sent by Tag i. A match confirms that $i = \widetilde{j}$. This establishes that SDGPP is not FORWARD-Universal-UNT. □

CHAPTER 5
PARALLEL - DEPENDENCY GROUPING PROOF PROTOCOL

5.1 Introduction

In parallel grouping-proofs, the reader broadcasts communications to all tags. All tags that receive the message compute in parallel then communicate back to the reader one at a time. Because of use in large-scale applications, the number of tagged items read may be significantly high. Therefore, scalability tailored to privacy and security is an important challenge that affects the RFID system quality when designing a protocol.

5.2 Motivation

This chapter presents the Parallel-Dependency Grouping Proof Protocol (PDGPP) motivated by SDGPP. We adapt the serial-dependency protocol idea to reduce protocol execution time by overlapping tag computation times.

Many grouping-proof protocols exist, but these often do not satisfy all aspects of privacy, security, and efficiency. PDGPP has parallel dependency among tags, extends the security and privacy properties, and improves on efficiency, scalability, and resilience over existing schemes. Out proposed protocol withstands a range of attacks on security and privacy, but it does trade a weakened untraceability for improved scalability with respect to SDGPP. It is lightweight and scalable for large systems while remaining in compliance with the EPC Class 1 Gen 2 (C1G2) standard.

5.3 Description of PDGPP

PDGPP has two rounds. In the first round, the reader broadcasts a message to all tags and receives their replies. Then, in the second round, the reader uses these replies to build a message that it broadcasts to all tags (for parallel dependency) and receives their replies that incorporate this message. The reader builds a proof from the tags' replies, encrypts it, and sends it to the verifier. Figure 5.1 is a view of the messages exchanged among reader and tags in PDGPP.

Reader
$\longrightarrow J_r(1), \eta_r, \delta(1)$ to all tags

Tag 1
$\longleftarrow K(1,1), L(1,1), ta_r(1,1)$ to Reader
...
Tag i
$\longleftarrow K(i,1), L(i,1), ta_r(i,1)$ to Reader
...
Tag m
$\longleftarrow K(m,1), L(m,1), ta_r(m,1)$ to Reader

Reader
$\longrightarrow J_r(2), K^*, \delta(2)$ to all tags

Tag 1
$\longleftarrow Q(1,2), L(1,2), ta_r(1,2)$ to Reader
...
Tag m
$\longleftarrow Q(m,2), L(m,2), ta_r(m,2)$ to Reader

Figure 5.1. PDGPP messages

We now describe in detail the three phases of the protocol: initialization, grouping-proof collection, and proof verification. Table 5.1 describes notation used in the protocol.

5.3.1 Initialization Phase

In the initialization phase, the verifier pre-computes information for u protocol runs and stores this information in the reader and also stores one set of values in each tag (see Table 5.2).

Reader: Because the reader broadcasts its messages to all tags, the verifier loads it with values that are not specific to any individual tag. The verifier initializes the reader with the following values (see Table 5.2) - independent of group and run: $\{RH, K_{RV}\}$; group-specific: $\{GH, s_{RT}\}$; group- and run-specific $\{run, J, \eta\}$. The reader starts with $seed_R$ initiated by verifier (can be once or specific to a group or specific to a run on a group).

Table 5.1. Notation for PDGPP

Notations	Description
s_{V1}, s_{V2}, s_{V3}	unique secrets known only to the verifier
$GID(k), GH$	group ID $GH(k) = h(GID(k), s_{V1})$, where h is a hash function
$TID(i), TH(i)$	unique Tag ID $TH(i) = h(TID(i), s_{V2})$
$RID(j), RH$	unique reader ID $RH(j) = h(RID(j), s_{V3})$
K_{RV}	secret key shared between reader and the verifier
$ta(i)$	unique tag alias identifier
r_V	run-specific PRN generated by the verifier
s_G	shared group secret among group tags, $s_G(k)$
s_{VT}	a secret shared between verifier and tags
s_{RT}	a secret shared between reader and tags
x, y, z	indices used in zero-knowledge proof
p, w	values for zero-knowledge proof
J	$J = GH \oplus PRNG(s_G \oplus r_V)$, where $PRNG$ is a pseudorandom number generator
η	$\eta = PRNG(s_{VT}) \oplus (x\|\|y\|\|z\|\|r_V)$
$seed_R$	seed for the reader PRNG function
$r_R(h)$	PRN generated by the reader for a round h
$s_T(i)$, $s_T(i,x)$	secret key unique to Tag i; $s_T(i,x)$ is a 128-bit string with index x in $s_T(i)$
$seed_T(i)$	seed for Tag i PRNG function
$r_T(i, index)$	PRN generated by Tag i
c_V	time constant to limit activity in a protocol run

Table 5.2. Initial values stored in the Reader - PDGPP

group	run	GH	J	η	s_{RT}
1	1	$GH[1]$	$J[1,1]$	$\eta[1,1]$	$s_{RT}[1]$
1	2	$GH[1]$	$J[1,2]$	$\eta[1,2]$	$s_{RT}[1]$
...	...	...	...	...	...
1	u	$GH[1]$	$J[1,u]$	$\eta[1,u]$	$s_{RT}[1]$
2	1	$GH[2]$	$J[2,1]$	$\eta[2,1]$	$s_{RT}[2]$
2	2	$GH[2]$	$J[2,2]$	$\eta[2,2]$	$s_{RT}[2]$
...	...	...	...	...	...
j	1	$GH[j]$	$J[j,1]$	$\eta[j,1]$	$s_{RT}[j]$
...	...	...	...	...	...
j	u	$GH[j]$	$J[j,u]$	$\eta[j,u]$	$s_{RT}[j]$

Tags: The verifier initializes each Tag i (that is, the tag with index i in its group) with the following values - independent of group and run: $\{TH(i)\}$; group-specific: $\{GH, s_G, s_T(i), s_{VT}, s_{RT}, p, w, c_V, ta(i)\}$. Tag i starts a run with $seed_T(i)$ for its PRNG, initiated by verifier. Tag i does not update any stored values so that tags can stay synchronized.

Tag i has $ta(i)$, which has the following structure. Let values such as GH and s_{RT} be d bits long (say 128 bits), and let tag index i be c bits long (log of group size). The verifier chooses a random bit position k in $ta(i)$ where it places index i. It places value k in the low $\log d$ bits of $ta(i)$ and puts random values in the remaining bits of $ta(i)$. Observe that given $ta(i)$, a reader can extract bit position k then index i.

5.3.2 Grouping-Proof Collection Phase

During the grouping-proof collection phase, the verifier does not connect to the reader or tags. When in a run, the reader places all tags in a group in open state. This starts the time window controlled by c_V.

Round 1 - Reader

Figure 5.2 shows the reader actions in Round 1. The reader generates a fresh pseudo-random number $r_R(1)$ and XORs this with precomputed values to produce $J_r(1), \eta_r, \delta(1)$. Here, as in SDGPP (Section 4.3), the randomness introduced with $r_R(i)$ keeps messages fresh across runs and avoids sending values in the clear. The reader sends three values to all tags in the group - $J_r(1), \eta_r, \delta(1)$.

Round 1 - Tag i

Tag i authenticates the reader, verifies the integrity of the incoming messages, and computes its response using zero-knowledge techniques. Tag i extracts $r_R(1)$ then r_V (and x, y, z to use in the zero-knowledge part) then GH. If the extracted GH matches the stored GH, then Tag i authenticates that the reader is authorized by the verifier and confirms the integrity of messages received from the reader. If the match fails, then Tag i aborts. Tag i next computes values that will establish its presence to the verifier. PDGPP uses the

```
Reader - Round 1 - parallel
use $J$, $\eta$ from current run
$r_R(1) \leftarrow PRNG(seed_R)$
$seed_R \leftarrow r_R(1)$
$J_r(1) \leftarrow J \oplus r_R(1)$
$\eta_r \leftarrow \eta \oplus r_R(1)$
$\delta(1) \leftarrow s_{RT} \oplus r_R(1)$
Send to all tags: $J_r(1), \eta_r, \delta(1)$
```

```
Tag $i$ - Round 1 - parallel
$r_R(1) \leftarrow \delta(1) \oplus s_{RT}$
        ; Use extracted $r_R(1)$ to remove randomness
$J \leftarrow J_r(1) \oplus r_R(1)$
$\eta \leftarrow \eta_r \oplus r_R(1)$
$x||y||z||r_V \leftarrow PRNG(s_{VT}) \oplus \eta$
        ; Tag $i$ uses stored value $s_{VT}$ to obtain $r_V$
$GH \leftarrow J \oplus PRNG(s_G \oplus r_V)$
        ; $r_V$ obtained from $\eta$ is used to extract $GH$ from $J$
if (extracted $GH \neq$ stored $GH$)
        abort
$S(i,1) \leftarrow s_T(i,x) \oplus PRNG(TH(i) \oplus s_T(i,y)) \oplus PRNG(s_T(i,z) \oplus r_V)$
$K(i,1) \leftarrow null$
for $index \in \{x,y,z\}$
        $r_T(i,index) \leftarrow PRNG(seed_T(i))$
        $seed_T(i) \leftarrow r_T(i,index)$
        if $S(i,1)[index] = 0$
                $Z \leftarrow (r_T(i,index))^2 \bmod p$
        else ; that is, $S(i,1)[index] = 1$
                $Z \leftarrow w \cdot (r_T(i,index))^2 \bmod p$
        $K(i,1) \leftarrow K(i,1)||Z$
$L(i,1) \leftarrow GH \oplus r_R(1) \oplus PRNG(J \oplus K(i,1) \oplus ta(i))$
$ta_r(i,1) \leftarrow ta(i) \oplus r_R(1)$
Send to Reader: $K(i,1), L(i,1), ta_r(i,1)$
```

Figure 5.2. Round 1: PDGPP

zero-knowledge property (as in SDZ and SDGPP) described in Sections 4.2 and 4.3. Using x, y, z, Tag i selects subsets of the tag's secret $s_T(i)$ along with $TH(i)$ to compute $S(i,1)$. Depending on bits of $S(i,1)$, Tag i concatenates random squares and random pseudosquares to form $K(i,1)$. This proves to the verifier that Tag i has the right secret $s_T(i)$, without revealing it to an attacker.

Tag i then computes $L(i,1)$ and $ta_r(i,1)$ to enable the reader to authenticate the tag, verify the integrity of received messages, and identify index i. Tag i sends $K(i,1)$, $L(i,1)$,

```
Reader - Round 2 - parallel
$K^* \leftarrow 0$
initialize all $valid(i,1)$, $valid(i,2)$ to 0
for each $K(i,1), L(i,1), ta_r(i,1)$ message received
        $ta(i) \leftarrow ta_r(i,1) \oplus r_R(1)$
        $GH \leftarrow L(i,1) \oplus r_R(1) \oplus PRNG(J \oplus K(i,1) \oplus ta(i))$
        if (extracted $GH$ = stored $GH$)
                $K^* \leftarrow K^* \oplus K(i,1)$
                extract $i$ from $ta(i)$
                $valid(i,1) \leftarrow 1$
                        ; Tag $i$'s response is valid
$r_R(2) \leftarrow PRNG(seed_R)$
$seed_R \leftarrow r_R(2)$
$J_r(2) \leftarrow J \oplus K^* \oplus r_R(2)$
$\delta(2) \leftarrow s_{RT} \oplus r_R(2)$
Send to all tags: $J_r(2), K^*, \delta(2)$
```

```
Tag $i$ - Round 2 - parallel
    ; tags in OPEN state so do not need to re-authenticate reader
$r_R(2) \leftarrow \delta(2) \oplus s_{RT}$
    ; Use extracted $r_R(2)$ to remove randomness
$J \leftarrow J_r(2) \oplus K^* \oplus r_R(2)$
    ; Tag $i$ uses same $x, y, z, r_V$ as in Round 1
$GH \leftarrow J \oplus PRNG(s_G \oplus r_V)$
if (extracted $GH \neq$ stored $GH$)
    abort
$Q(i,2) \leftarrow PRNG(K^* \oplus r_V \oplus i)$
$L(i,2) \leftarrow GH \oplus r_R(2) \oplus PRNG(J \oplus Q(i,2) \oplus ta(i))$
$ta_r(i,2) \leftarrow ta(i) \oplus r_R(2)$
Send to Reader: $Q(i,2), L(i,2), ta_r(i,2)$
```

Figure 5.3. Round 2: PDGPP

and $ta_r(i,1)$ to the reader.

Round 2 - Reader

The reader receives a response from Tag i and extracts hashed Group ID GH (Figure 5.3). If the extracted GH matches the stored GH, this authenticates Tag i and also confirms the integrity of the messages from Tag i - $K(i,1)$, $L(i,1)$, and $ta_r(i,1)$. Otherwise, the response is not valid and the reader discards $K(i,1)$, $L(i,1)$, and $ta_r(i,1)$. The reader flags all valid responses using $valid(i,1)$ (initialized to 0 for Round 1). The reader repeats this for all tags' responses building K^* from valid $K(i,1)$ values. It generates a fresh random number $r_R(2)$ for its Round 2 message to the tags. It builds $J_r(2)$ from J, K^*, and $r_R(2)$ to allow an integrity check. The reader sends three values to all tags in the group - $J_r(2), K^*, \delta(2)$.

Round 2 - Tag i

Recall that tags are left from Round 1 in open state, so tags do not need to re-authenticate the reader. In Round 2, Tag i extracts $r_R(2)$ from $\delta(2)$, then extracts J, then GH (Figure 5.3). If the extracted GH matches the stored GH, then this confirms the integrity of messages received from the reader. Using K^* and r_V extracted in Round 1, Tag i will compute its second message to the verifier: $Q(i,2) = PRNG(K^* \oplus r_V \oplus i)$. This introduces parallel-dependency into the protocol as K^* includes computation by all tags. Each Tag i in parallel uses K^* in its computation of $Q(i,2)$.

Then Tag i computes $L(i,2)$ and $ta_r(i,2)$ to enable the reader to authenticate the tag, verify the integrity of the received messages, and identify index i. Tag i sends $Q(i,2)$, $L(i,2)$, and $ta_r(i,2)$ to the reader.

5.3.3 Proof - Verification

The reader compiles information to construct the proof -
$P = \{RH, GH, run, r_R(1), r_R(2), (i, valid(i,2), K(i,1), Q(i,2))\}$ including messages from all tags i with valid responses in Round 1 (Figure 5.4). Note that a tag could have sent a valid response in Round 1 but not in Round 2 so $valid(i,2)$ indicates this, while if a tag did not send a valid response in Round 1, the proof does not include the tag regardless of any Round 2 response. Then the reader encrypts proof P using the secret key K_{RV} shared with the verifier to form P'. The reader sends P' to the verifier either immediately or at a later time with more proofs.

The verifier decrypts P' using the shared secret key K_{RV}. Using RH, GH, and run, the verifier identifies which reader sent the proof, the tag group, and the run of the protocol for that group. For each Tag i, the verifier computes $S(i,1)$. Using $K(i,1)$, p, and w, the verifier distinguishes between squares and pseudosquares and decodes the xth, yth, and zth bits. If the decoded bits match in $S(i,1)$, the verifier confirms that Tag i has participated in the

```
Reader - Proof construction - parallel
for each $Q(i,2), L(i,2)$, and $ta_r(i,2)$ message received
        $ta(i) \leftarrow ta_r(i,2) \oplus r_R(2)$
        $GH \leftarrow L(i,2) \oplus r_R(2) \oplus PRNG(J \oplus Q(i,2) \oplus ta(i))$
        if (extracted $GH$ = stored $GH$)
                extract $i$ from $ta(i)$
                if $valid(i,1) = 1$
                      $valid(i,2) \leftarrow 1$
                      ; Tag $i$'s response is valid
Proof construction
Reader puts together proof $P = \{RH, GH, run, r_R(1), r_R(2), (i, valid(i,2), K(i,1),
Q(i,2))\}$, including all tags $i$ for which $valid(i,1) = 1$. The reader encrypts $P$ as $P'$ using
$K_{RV}$ then sends $P'$ to the verifier.
```

Figure 5.4. Reader - to Verifier: PDGPP

proof. The verifier computes K^*, then along with r_V and i, the verifier computes $Q(i,2)$ and compares against the received $Q(i,2)$ to confirm the dependence among tag computations.

5.4 Security Analysis

5.4.1 Security and Privacy Properties

We now compare our protocol to other recent grouping proof protocols in terms of attacks resisted (Table 5.5.1). We observe that recently proposed protocols are strong on anonymity and protection from tracking attacks. Both of these properties are very crucial to have a private system. We now establish that our protocol resists a range of attacks on security and privacy. For definitions of these attacks, see Section 3.2.4 and [15, 73, 87, 100, 106, 122, 127].

Privacy: The reader and tags never send Tag ID (T to R communication) and group ID (T to R and R to T) information in the clear but always as part of a value computed with XOR or PRNG. So an adversary cannot track the same tag from the information exchanged from one run to the next. Attacks and other considerations relevant to privacy include tracking attack, anonymity attack, and untraceability. Section 5.4.2 gives a ... through analysis of untraceability. *Impersonation attacks:* Impersonation attacks include three variants: (i) impersonate tag to reader; (ii) impersonate reader to tag; and (iii) impersonate reader to verifier. (Attacks under this category include forgery, illegitimate tag, subset replay, and multi-proof-session attacks.) (i) Without GH and s_{RT}, an adversary cannot construct an $L(i)$ value that a reader will accept when the reader checks the match between extracted and stored GH. (ii) Without s_{RT}, s_{VT}, and s_G, an adversary cannot construct values that Tag i will accept when it checks the match between extracted and stored GH. (iii) The reader sends proof P in encrypted form P' using a key shared with the verifier, so without this key, the adversary cannot send a P' that the verifier will decode into valid information. Furthermore, besides this encryption, an adversary cannot insert into a proof P information from a missing tag i because the Q values derived from K^* would not incorporate $K(i,1)$.

Concurrency attack: This attack is when multiple readers simultaneously try to generate grouping proofs for the same tags. From the RFID specification [40], once a tag is in open state to one reader, it does not respond to others. So, after a reader places a tag in open state at the start of a run, the tag will complete its participation in this grouping proof before responding to any second reader.

Desynchronization: Because the tags do not update any stored secret, no synchronization concerns exist.

Replay attack: If an adversary replayed a message from a Tag i to the reader, the reader would find it not valid because the $L(i,1)(L(i,2))$ does not depend on the current $r_R(1)(r_R(2))$. If an adversary replayed a message from the reader to Tag i from a previous run, the tag would not detect the replay attack. Tag i would reply with fresh values built on fresh pseudorandom numbers (see above under privacy), and since it would not update anything, this would not cause any future harm to our protocol.

Message integrity attack: A tag extracts GH from $\delta(g)$, η_r, K^*, and $J_r(g)$. A reader extracts GH from $K(i,1)$ or $Q(i,2)$, $L(i,g)$, and $ta_r(i,g)$. In both cases, a match between extracted and stored GH confirms the integrity of received messages.

Denial of proof: An adversary can attempt to force a proof to fail in various ways, and the protocol counters each of these. An adversary can block or tamper with messages to desynchronize reader and tags, it can block or tamper with messages to break the dependency chain across tags, and it can impersonate tags. See above for how the protocol prevents each of these from spoiling a proof.

5.4.2 Untraceability Analysis of PDGPP

In this section, we classify PDGPP in the untraceability model of Avoine *et al.* [8] and establish that PDGPP is $\mathsf{CLASSIC\text{-}Universal\text{-}UNT}$ but is not $\mathsf{SIDE\text{-}Existential\text{-}UNT}$.

Recall from Section 3.3.1 the following. $\mathcal{A}$ selects two challenge tags, i and j. Challenger $\mathcal{C}$ relabels the two challenge tags as $\widetilde{i}$ and $\widetilde{j}$ with respect to a random bit b such that if $b = 0$, then $\widetilde{i} = i$ and $\widetilde{j} = j$ (that is, the labels remain the same), but if $b = 1$, then $\widetilde{i} = j$ and $\widetilde{j} = i$ (that is, $\mathcal{C}$ swaps the labels). $\mathcal{A}$ must determine the value of b. If the probability that $\mathcal{A}$ can correctly determine b is not very different from $\frac{1}{2}$, then the protocol is untraceable in that class.

The verifier initializes tags in PDGPP with values:

- independent of group and run, unique for each tag - $\{TH(i)\}$
- tag- and group-specific - $\{ta(i), s_T(i), seed_T(i)\}$
- group-specific, which are shared among all tags in a group - $\{GH, s_G, s_{VT}, s_{RT}, p, w\}$

Tag i does update $seed_T(i)$, but does not update other stored values so that tags can stay synchronized in the protocol. In the untraceability UNT experiment, the adversary $\mathcal{A}$ can interact with the system S within some limitations, which define the adversary classes (see Section 3.3).

For the class $\mathsf{CLASSIC\text{-}Universal\text{-}UNT}$ and each of the other classes, adversary $\mathcal{A}_{\mathsf{CLASSIC}}$ can interact with the tags and reader by eavesdropping, blocking messages, tampering with messages, spoofing reader or tag, but cannot corrupt any tags.

For the class $\mathsf{SIDE\text{-}Existential\text{-}UNT}$, $\mathcal{A}_{\mathsf{SIDE}}$ cannot corrupt the challenge tags, but can corrupt any other tags after Challenger $\mathcal{C}$ executes the protocol c_1 times on Tag $\widetilde{i}$ and c_2 times on Tag $\widetilde{j}$ for some specified numbers (c_1, c_2).

Theorem 5.4.1 *PDGPP is* $\mathsf{CLASSIC\text{-}Universal\text{-}UNT}$.

Proof. For universal untraceability, (c_1, c_2) can be any values, and proving untraceability with $(c_1, c_2) = (0, 0)$ suffices [8]. That is, when $\mathcal{A}_{\mathsf{CLASSIC}}$ has selected Tags i and j, it has access to all transcripts for all tags in all previous executions of PDGPP, and it has the information on which of these transcripts belong to which tags, including i and j. After the last execution of PDGPP, $\mathcal{A}_{\mathsf{CLASSIC}}$ receives all transcripts of communications, however, the Challenger labels the transcripts for i and j in the last execution as belonging to $\widetilde{i}$ and $\widetilde{j}$ according to a random selection by the Challenger.

Adversary $\mathcal{A}_{\mathsf{CLASSIC}}$ is not allowed to corrupt any tag. Therefore, the only set of information that $\mathcal{A}_{\mathsf{CLASSIC}}$ can collect for its attack are transcripts (the messages exchanged during protocol executions), nothing that it can block or tamper with messages and spoof reader or tags.

To establish that $i = \widetilde{j}$, $\mathcal{A}_{\mathsf{CLASSIC}}$ would have to establish that Tag i in one of the previous executions is the same as Tag $\widetilde{j}$. To do this, $\mathcal{A}_{\mathsf{CLASSIC}}$ would have to calculate a message that identifies Tag i (that is, $K(i,1), L(i,1), Q(i,2), L(i,2), ta_r(i,1)$, or $ta_r(i,2)$) from the information it has obtained from transcripts using information that matches Tag $\widetilde{j}$.

From the communication transcripts, $\mathcal{A}_{\mathsf{CLASSIC}}$ holds that Tag i:
received inputs in round 1 - $J_r(1)$, η_r, $\delta(1)$ - and round 2 - $J_r(2)$, K^*, $\delta(2)$ - with all values specific to this execution, and produced outputs in round 1 - $K(i,1), L(i,1), ta_r(i,1)$ - and round 2 - $Q(i,2), L(i,2), ta_r(i,2)$, while Tag $\widetilde{j}$: received inputs in round 1 - $J_r(1)$, η_r, $\delta(1)$ - and round 2 - $J_r(2)$, K^*, $\delta(2)$ (for $\widetilde{j}$) - with all values specific to this execution, and produced outputs in round 1 - $K(\widetilde{j},1), L(\widetilde{j},1), ta_r(\widetilde{j},1)$ - and round 2 - $Q(\widetilde{j},2), L(\widetilde{j},2), ta_r(\widetilde{j},2)$.

Even if $i = \widetilde{j}$, $K(\widetilde{j},1)$ will be different from $K(i,1)$, and $Q(\widetilde{j},2)$ will be different from

$Q(i,2)$. This is because Tag $\widetilde{j}$ generated fresh pseudorandom $r_T(\widetilde{j}, index)$ values to build $K(\widetilde{j},1)$ and these form K^* that Tag $(\widetilde{j})$ uses for $Q(\widetilde{j},2)$. Correspondingly, $L(\widetilde{j},1), L(\widetilde{j},2), ta(\widetilde{j},1), ta(\widetilde{j},2)$ will be different from $L(i,1), L(i,2), ta_r(i,1), ta_r(i,2)$ again due to different pseudorandom values $K(\widetilde{j},1), Q(\widetilde{j},2), r_R(1), r_R(2)$ respectively, used in round 1 and 2.

Because $\mathcal{A}_{\mathsf{CLASSIC}}$ does not have s_{RT}, it cannot extract $r_R(1)$ or $r_R(2)$. Also, $\mathcal{A}_{\mathsf{CLASSIC}}$ cannot calculate $r_R(2)$ from $r_R(1)$ in the last transcript or any previous transcript because of how the reader changes $seed_R$. So, $\mathcal{A}_{\mathsf{CLASSIC}}$ cannot extract x, y, z, and r_V and so cannot calculate $S(i,1)$ and so cannot calculate $K(i,1)$. $\mathcal{A}_{\mathsf{CLASSIC}}$ cannot calculate $L(i,1), L(i,2), ta_r(i,1), ta_r(i,2)$ without $r_R(1)$ and $r_R(2)$. Therefore, $\mathcal{A}_{\mathsf{CLASSIC}}$ cannot match Tag $\widetilde{j}$ with Tag i any better than it can match Tag $\widetilde{i}$.

In PDGPP the reader and tags never send tag secrets and group secrets in the clear but always as part of a value computed with XOR or PRNG. So $\mathcal{A}_{\mathsf{CLASSIC}}$ cannot track the same tag from the messages exchanged from one run to the next.

□

Theorem 5.4.2 *PDGPP is not* SIDE-Existential-UNT*.*

Proof. For the Side-Existential-UNT class, adversary $\mathcal{A}_{\mathsf{SIDE}}$ cannot corrupt the challenge tags but can corrupt any other tags. It can also obtain transcripts of communications to and from all tags in all executions of the protocol. To proof that PDGPP is not SIDE-Existential-UNT, we will prove that challenge tags are traceable for any (c_1, c_2) so they are traceable for $(c_1, c_2) = (0, 0)$ and for any number of execution of PDGPP by the challenger.

This protocol is not SIDE-Existential-UNT because of the static data initialized in all of the tags in a group. $\mathcal{A}_{\mathsf{SIDE}}$ can collect $GH, s_{GT}, s_{VT}, s_{RT}$ after any non-challenge tag corruption.

From the communication transcripts, $\mathcal{A}_{\mathsf{SIDE}}$ holds that Tag i received $\delta(1)$ as input in round 1 (specific to this execution) and produced $ta_r(i,1)$ as output in round 1, while Tag $\widetilde{j}$ received $\delta(1)$ as input in round 1 (specific to this execution) and produced $ta_r(i,1)$ as output in round 1.

To establish that $i = \widetilde{i}$ or $i = \widetilde{j}$, $\mathcal{A}_{\mathsf{SIDE}}$ would have to establish that Tag i in one of the previous executions is the same as Tag $\widetilde{i}$ or $\widetilde{j}$ in the last execution. Using s_{RT} from the corruption of non-challenge tags, $\mathcal{A}_{\mathsf{SIDE}}$ can calculate for Tag i in the previous execution $r_R(1) = \delta(1) \oplus s_{RT}$, then $ta(i) = ta_r(i, 1) \oplus r_R(1)$. For the execution with challenge tags $\widetilde{i}$ and $\widetilde{j}$, s_{RT} remains the same, so $\mathcal{A}_{\mathsf{SIDE}}$ calculates the following: $r_R(1) = \delta(1) \oplus s_{RT}$, $ta(\widetilde{i}) = ta_r(\widetilde{i}, 1) \oplus r_R(1)$, $ta(\widetilde{j}) = ta_r(\widetilde{j}, 1) \oplus r_R(1)$. Because i is either $\widetilde{i}$ or $\widetilde{j}$, then either $ta(\widetilde{i})$ or $ta(\widetilde{j})$ will equal $ta(i)$.

□

5.5 Grouping Proof Protocols Comparison

5.5.1 Comparison of Security and Privacy Properties

Table 5.5.1 compares the security properties of SDGPP and PDGPP to recent grouping proof protocols. We observe that recently proposed protocols are strong on anonymity and protection from tracking attacks. Both of these properties are crucial to have a private system. Also, protocols AbMM'16, Yuan'16, Zhou'18 are vulnerable to some impersonation attacks because they fail to add an updating mechanism or a time counter mechanism to ensure the freshness of its communication values for any of the communication sessions.

Table 5.3. Comparison of privacy and security properties of grouping-proof protocols.

	Features										
Protocol	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11
SDZ'15 [106]	√	√	√	√	√	√	√	X	√	√	√
AbMM'16 [1]	√	√	√	X	X	√	√	X	X	X	X
BuM'16 [16]	√	√	√	√	√	√	√	√	√	√	√
Yuan'16 [128]	√	√	√	√	√	√	√	√	X	√	√
Hsi'15 [51]	√	√	√	√	√	√	√	√	X	√	X
Zhou'18 [136]	√	√	√	√	√	√	X	√	X	√	X
ZhQW'18 [134]	X	√	√	X	X	√	√	X	X	X	X
ShZW'17 [98]	X	√	√	√	X	√	√	√	√	√	X
RoBH'18 [93]	√	√	√	√	√	√	√	√	√	√	X
SDGPP [25]	√	√	√	√	√	√	√	√	√	√	√
PDGPP	√	√	√	√	√	√	√	√	√	√	√

A1: Privacy/tracking **A2:** Privacy/anonymity **A3:** Privacy/untraceability
A4: Impersonation (T - R) **A5:** Impersonation (R - T) **A6:** Impersonation (R - V)
A7: Concurrency **A8:** Desynchronization **A9:** DoP
A10: Replay **A11:** Message integrity

5.5.2 Performance Comparison

Table 5.4 shows clearly that many of the protocols use a type of hash function, MAC, or elliptical cryptography in tags. These operations require too many gates to be EPC compliant. We also show that SDGPP and PDGPP proposed are more efficient and scalable than others. We reduced significantly the communication among entities, while still maintaining and enhancing security and privacy.

Table 5.4. Comparison of communication overhead.

	Features						
Protocol	C1	C2	C3	C4	C5	C6	C7
SDZ'15 [106]	√	PRNG	-	8	1	$6m$	$11m$
AbMM'16 [1]	X	Hash	3	1	3	m	$7m$
BuM'16 [16]	X	Hash	8	1	3	$m+2$	$11m$
Yuan'16 [128]	√	PRNG	-	8	-	$5m+2$	$11m$
Hsi'15 [51]	X	MAC, Hash	4	1	1	m	$10m$
Zhou'18 [136]	X	ECC	4	1	3	m	$15m$
ZhQW'18 [134]	√	PRNG	-	4	-	$m+5$	$6m$
ShZW'17 [98]	X	Hash	4	2	4	2	$4m+4$
RoBH'18 [93]	√	PRNG	-	6	-	$2m+3$	$3m+2$
SDGPP [25]	√	PRNG	-	9	1	$2m$	$6m$
PDGPP	√	PRNG	-	9	1	$2m+2$	$6m+6$

C1: EPC Compliance
C2: Type of Crypto Function(s) in tags
C3: Number of Crypto Operation(s) in Tag
C4: Number of PRNG Operations in Tag
C5: Number of Crypto Operation(s) in Reader
C6: Number of PRNG Operations in Reader
C7: Items communicated in a message

CHAPTER 6
DYNAMIC GROUPING PROOF PROTOCOL

6.1 Introduction

In some applications, a verifier may need to identify the presence of tags in a subgroup of a larger group. It may create these subgroups dynamically. So a tag can carry information about the large group, but not about any subgroup to which it will belong. A dynamic grouping-proof generates proofs of presence of tags in each of these subgroups.

In many large-scale applications, the number of tagged items interrogated may be significantly high. Due to unreliability of the radio interface, when a group has a large number of tags, the probability that one or more tags will fail this interrogation is high. An approach to overcome this problem is to partition the large group into subgroups, each with an adequate number of tags and each subgroup interrogated separately.

Challenges for creating a dynamic grouping-proof include collision, scalability, limiting the computational load on the entities involved, and guaranteeing privacy and security. We designed the Dynamic Grouping Proof Protocol (DGPP) to overcome those challenges.

6.2 Motivation

Many RFID grouping-proofs focus on the static scenario where the tag group is constant and tag participants remain the same during the grouping-proof. However, many practical RFID applications, such as logistic control, are dynamic and require partitioning a large group to subgroups, tag subgroup variations, tags to belong to multiple subgroups, or tags (the tagged object) to enter and/or leave the reader's covered area frequently. A fundamental research question is how to design an efficient and secure protocol in this dynamic setting. To tackle all those challenges we design the DGPP, implementing solutions to collision, scalability, limiting the computational load, and guaranteeing privacy and security.

DGPP provides collision resolution by informing tags in a subgroup a specific order in which to respond. The approach is adopted to introduce randomness in a tag response across executions and avoid collisions [51, 63].

DGPP provides scalability by partitioning subgroups within the large group, removing group- and reader-dependency, limiting the computation and communication load of the tags (remain compatible under the EPC standard) which allows the protocol to remain light and suitable for the scope of large-scale applications.

DGPP allows missing tags. If one or more tags are missing or for some reason leave the reader range while generating a proof (faulty tags, tampered tags, tags removed for genuine reasons), than the reader collects proof from the available tags. We introduce the *desired significance* parameter to allow the reader to treat an incomplete subgroup as satisfactory at a verifier-defined level. For subgroups that do not satisfy the desired significance, the reader will run a new grouping-proof iteration, while it will skip already satisfied subgroups in the new iteration.

DGPP provides privacy and security under precise design requirements for grouping-proof protocols (see Section 3.2.1).

Because we assume large-scale RFID applications for dynamic grouping-proofs, we choose DGPP to have a parallel-dependency structure. The reader broadcasts communications to all tags. All tags that receive the message compute in parallel then communicate back to the reader one at a time in the specified order.

6.3 Description of DGPP

This section presents the Dynamic Grouping Proof Protocol (DGPP). The reader executes the protocol. The protocol executes up to u iterations for subgroups $g[1], ..., g[z]$. It builds an array of proofs from each subgroup. Then the reader encrypts this array of subgroup proofs and sends it to the verifier.

For subgroup $g[q]$ in iteration n, DGPP runs subroutine $\textsc{Subgroup.Check}(q, n)$. An overview of the subroutine $\textsc{Subgroup.Check}(q, n)$ in the protocol is as follows. The subroutine has two rounds. The first round has two parts. In Part 1, the reader broadcasts a message to all tags in its range then waits for the tags to finish a computation. In Part 2, the reader sends a list of transformed tag IDs that selects the tags that will participate in the grouping proof and specifies the order of the tag responses. The reader receives their replies. In the second round, the reader uses these replies to build a message that it broadcasts to all tags (for parallel dependency) and receives their replies that incorporate this message.

We now describe in detail the three phases of the protocol: initialization, grouping-proof collection, and proof verification. Table 6.1 describes notation used in the protocol.

Table 6.1. Notation for DGPP

Notation	Description
G	Group of tags
$g[q]$	subgroup (subset) of G generated by the verifier
d_q	number of tags in a subgroup $g[q]$
$\gamma[q]$	designated significance for subgroup $g[q]$
s_V	unique secret known only to the verifier
$TID(i)$	unique tag identifier $TID(i) = h(tid(i), s_V)$ (T-part;V-part)
$a(i)$	unique tag alias
$\langle LPID(j)[q](n) \rangle$	ordering list constructed by the Verifier for subgroup[q] in iteration n; j - is rank of an element in the list
$t[q](n)$	pseudorandom number such that the sequence of these has distinct increments useful for the entities involved in the protocol
$PID(i)$	pseudo-identity of a tag $PID(i) \leftarrow PRNG(a(i) \oplus t[q](n))$
K_{VR}	secret key shared between the verifier and reader
s_{RT}	secret share by reader and tags in G
$t_T(i)$	last authenticated $t[q](n)$ value received by Tag i
$seed_R$	a seed for the reader PRNG function initialized from the verifier
$r_R(k)$	PRN generated by the reader for round k
c_{VT}	time constant to limit tag activity in the protocol
c_{VR}	verifier and reader-specific time constant to limit activity in the protocol

6.3.1 Initialization Phase

In the initialization phase, the verifier pre-computes information for u protocol iterations and stores this information in the reader (see Table 6.2). The verifier also initializes each tag in group G with specific values.

Reader: The verifier initializes the reader with the following values (see Table 6.2): $\{t[1...z](1...u), \langle LPID(j)[1...z](1...u)\rangle, seed_R[1...z](1...u), \gamma[1...z], K_{RV}, s_{RT}, c_{VR}\}$. Variable t is a long pseudorandom value with distinct increments useful for the entities involved in the protocol. It gives each tag a way of comparing with an incrementing variable, so no previous t can be replayed. All tags will follow a specific response sequence (ordering list $\langle LPID\rangle$) within a predefined time interval to send back self-response to the reader. $\langle LPID(j)\rangle$ is a random permutation of tags pseudo-identity in a subgroup, which is constructed by the verifier and is subgroup- and run-specific. A tag's pseudo-identity is calculated as: $PID(i) = PRNG(a(i) \oplus t[q](n))$. $\gamma[k]$ is the *designated significance* parameter (tunable parameter). It is the count of the number of tags that must be present in order to consider the subgroup satisfied.

Table 6.2. Initial values stored in the Reader - DGPP

sub-group	*iteration*	designated significance	$\langle PID(i,j)\rangle$	t	seed for RNG
$g[1]$	(1)	$\gamma[1]$	$\langle LPID(j)[1](1)\rangle$	$t[1](1)$	$seed_R[1](1)$
$g[2]$	(1)	$\gamma[2]$	$\langle LPID(j)[2](1)\rangle$	$t[2](1)$	$seed_R[2](1)$
...	...	...	...	...	...
$g[z]$	(1)	$\gamma[z]$	$\langle LPID(j)[z](1)\rangle$	$t[z](1)$	$seed_R[z](1)$
$g[1]$	(2)	$\gamma[1]$	$\langle LPID(j)[1](2)\rangle$	$t[1](2)$	$seed_R[1](2)$
$g[2]$	(2)	$\gamma[2]$	$\langle LPID(j)[2](2)\rangle$	$t[2](2)$	$seed_R[2](2)$
...	...	...	...	...	...
$g[z]$	(2)	$\gamma[z]$	$\langle LPID(j)[z](2)\rangle$	$t[z](2)$	$seed_R[z](2)$
...	...	...	...	...	...
$g[1]$	(u)	$\gamma[1]$	$\langle LPID(j)[1](u)\rangle$	$t[1](u)$	$seed_R[1](u)$
$g[2]$	(u)	$\gamma[2]$	$\langle LPID(j)[2](u)\rangle$	$t[2](u)$	$seed_R[2](u)$
...	...	...	...	...	...
$g[z]$	(u)	$\gamma[z]$	$\langle LPID(j)[z](u)\rangle$	$t[z](u)$	$seed_R[z](u)$

Tags: The verifier initializes each Tag i (the tag with index i in G) with the following values: $\{TID(i), a(i), t_T(i), s_{RT}, c_{VT}\}$, where $TID(i), a(i)$ are tag-specific, unique for each tag; $t_T(i)$ holds the last value of $t[q](n)$ received, initialized to 0; and s_{RT} is a secret shared by reader and tags in G.

Note that time interval is limited with c_{VT} for Tag i to complete the protocol.

6.3.2 Grouping-Proof Collection Phase

The reader will start constructing proofs for all subgroups $g[1...z]$ included in the verifier scanning request for $iteration(1)$, using fresh random values for each subgroup and iteration. If any of the subgrouping-proofs in $iteration(1)$ fails to reach $\gamma[k]$ tags, then the reader will try again to generate a proof for those subgroups in $iteration(2)$, and so on. During the grouping-proof collection phase, the verifier does not connect to the reader or tags. When in an iteration, the reader places all tags in a subgroup in open state. This starts the time window controlled by c_{VT}.

6.3.2.1 The Protocol

```
The protocol
$P_{RV} \leftarrow \varnothing$      ; initialized array of $z$ entries for storing subgroup proofs
$stored.count \leftarrow 0$      ; initialized array of $z$ entries to store $valid.count$ for subgroup proofs

$Unsat\_Groups\ (1) \leftarrow$ queue of $1...z$
$n \leftarrow 1$      ; iteration index
while $n \leq u$ and $Unsat\_Groups\ (n) \neq \varnothing$
    $Unsat\_Groups\ (n+1) \leftarrow \varnothing$
    while $Unsat\_Groups\ (n) \neq \varnothing$
        $q \leftarrow \textsc{Dequeue}(Unsat\_Groups\ (n))$
        $(valid.count,\ P[q](n)) \leftarrow \textsc{Subgroup.Check}(q, n)$
        if $valid.count \geq \gamma(q)$
            $P_{RV}[q] \leftarrow P[q](n)$
            $stored.count[q] \leftarrow valid.count$
        else
            $\textsc{Enqueue}\ (Unsat\_Groups\ (n+1), q)$
            if $valid.count > stored.count[q]$
                $P_{RV}[q] \leftarrow P[q](n)$
                $stored.count[q] \leftarrow valid.count$
    end while
    $n \leftarrow n+1$
end while
$P'_{RV} \leftarrow E(K_{RV}, P_{RV})$      ; $P[q](n) = (q, n, (j, valid(2,j), PID(i)), M1^*, M2^*, r_R(1), r_R(2))$
includes all information that the verifier needs to rebuild and validate the proof

send to Verifier $P'_{RV}$
```

Figure 6.1. Protocol DGPP

The reader will start the protocol (see Figure 6.1) by initializing two arrays: P_{RV} to store subgroup proofs for each subgroup and *stored.count* to stored *valid.count* for each stored subgroup proof. The protocol will keep a queue of unsatisfied subgroups, to indicate the subgroups that need to be executed in the next iteration.

The flow of the protocol is as follows. In iteration n, the reader starts with a queue $\mathit{Unsat\_Groups}(n)$ of unsatisfied subgroups, that is, subgroups for which the reader has not yet found enough tags in earlier iterations to reach the designated significance. For each such subgroup $g[q]$, the reader will run $\textsc{Subgroup.Check}(q, n)$ to check for the presence of tags in $g[q]$. This returns a count of the tags found and a grouping proof for $g[q]$. If the count reaches the designated significance, then the reader stores the proof and count. If not and the count is more than previous iterations, then the reader stores the proof and count. Otherwise, the reader discards the returned data. For the counts of $g[q]$ that do not reach the designated significance, the reader adds q to $\mathit{Unsat\_Groups}(n + 1)$.

6.3.2.2 Subroutine $\textsc{Subgroup.Check}(q, n)$

Figure 6.2 is a view of the messages exchanged among reader and tags in $\textsc{Subgroup.Check}(q, n)$. This is a high level view of the protocol.

Reader - to $\mathit{subgroup}[q](n)$	**Tag** i in $\mathit{subgroup}[q](n)$
round 1	
$\longrightarrow m1_R, m2_R, \langle LPID(j)[1](1) \oplus r_R(1)\rangle$	
	round 1
	$\longleftarrow m1(i), M1(i)$ to Reader
round 2	
$\longrightarrow m3_R, m4_R, M1^*_r$	
	round 2
	$\longleftarrow m2(i), M2(i)$ to Reader

Figure 6.2. $\textsc{Subgroup.Check}(q, n)$ messages

Reader - Round 1, Part 1

```
for $j = 1$ to $d_q$     ; initialize array for subgroup $g[q]$ with number of tags $d_q$
    $valid(1, j) \leftarrow 0$
    $valid(2, j) \leftarrow 0$
$r_R(1) \leftarrow PRNG(seed_R)$
$seed_R \leftarrow r_R(1)$     ; reader updates seed for round 2
$m1_R \leftarrow s_{RT} \oplus r_R(1)$     ; to carry $r_R(1)$ to tags
$m2_R \leftarrow t[q](n) \oplus r_R(1)$     ; to carry $t[q](n)$ to tags
send to tags $m1_R, m2_R$
```

Tag i - Round 1, Part 1

```
$r_R(1) \leftarrow m1_R \oplus s_{RT}$     ; extract $r_R(1)$, using shared $s_{RT}$
$t[q](n) \leftarrow m2_R \oplus r_R(1)$     ; use extracted $r_R(1)$ to remove randomness
if $t[q](n) > t_T(i)$
  ; tag computes its new pseudo-identity
    $PID(i) \leftarrow PRNG(a(i) \oplus t[q](n))$
    $PID_r(i) \leftarrow PID(i) \oplus r_R(1)$
else abort
```

Figure 6.3. Round 1, Part 1: subroutine SUBGROUP.CHECK(q, n) - DGPP

Round 1, Part 1 - Reader

Figure 6.3 shows the reader actions in Round 1. The reader generates a fresh pseudo-random number $r_R(1)$ using $seed_R$ (subgroup- and iteration-specific) and XORs this with the current counter value and the shared secret s_{RT}. Here the randomness introduced with $r_R(1)$ keeps messages fresh across runs and avoids sending values in the clear. The reader sends two values to all tags in its range - $m1_R, m2_R$. Then the reader will delay its Round 1, Part 2 messages long enough for tags to complete their Round 1, Part 1 computation.

Round 1, Part 1 - Tag i

All tags in the reader range will do as follows (see Figure 6.3): extract $r_R(1)$ using shared s_{RT}, then use extracted $r_R(1)$ to remove randomness and get the current value of $t[q](n)$. If $t[q](n) > t_T(i)$, Tag i will compute its pseudo-identity for this run with a PRNG function using its unique tag alias $a(i)$ XORed with the current $t[q](n)$ and will further randomize this with $r_R(1)$. The additional randomization is necessary to assure privacy and anonymity and to allow Tag i to send distinct $PID(i)_r$ messages if it belongs to multiple subgroups within an execution of DGPP. Then Tag i will listen to Round 1, Part 2 messages from the reader.

```
Reader - Round 1, Part 2
delay
    ;long enough for tags to complete their Round 1, part 1 computation
$\langle LPID_r(j)\rangle \leftarrow \langle LPID(j)[q](n) \oplus r_R(1)\rangle$
send to tags $\langle LPID_r(j)\rangle$
```

```
Tag $i$ - Round 1, Part 2
$rank(i) \leftarrow 0$
$count(i) \leftarrow 0$
while tag receives $LPID_r(j)$ values from reader
    $count(i)$ ++      ; count number of $PID$ messages from reader
    if $PID_r(i) = LPID_r(j)$
        $rank(i) \leftarrow count(i)$
if $rank(i) = 0$
    abort
; Tag $i$ counts the items in the reader transmission. If an item matches Tag $i$'s $PID_r(i)$,
then Tag $i$ stores the $count$ as it $rank$.
; This match establishes that Tag $i$ belongs to the subgroup, authenticates the reader, sets
$rank$ for responding to the reader, and verifies the integrity of $t[q](n), r_R(1)$.

$t_T(i) \leftarrow t[q](n)$      ; update $t_T$ with the current $t[q](n)$ value
$M1(i) \leftarrow PRNG(TID(i) \oplus t[q](n)) \oplus r_R(1)$      ; a tag's secret designated for the Verifier
$m1(i) \leftarrow PID(i) \oplus M1(i) \oplus s_{RT}$      ; to carry $PID(i)$ and integrity of $M1(i)$ to reader
$count(i) \leftarrow 0$      ; reset $count$
send to reader in slot $rank(i)$: $m1(i), M1(i)$
```

Figure 6.4. Round 1, Part 2: subroutine SUBGROUP.CHECK(q, n) - DGPP

Round 1, Part 2 - Reader

After the delay, the reader (see Figure 6.4) will use the ordering list, initialized by the verifier for $g[q]$ in iteration (n). The reader additionally randomizes each $PID(i)$ from the list with its fresh pseudo-random number $r_R(1)$ and sends this to the tags in the list order. The reader sends d_q (number of tags in $g[q]$) number of messages to transmit the list $\langle LPID_r(j)\rangle$.

Round 1, Part 2 - Tag i

While Tag i receives $LPID_R(j)$ messages from the reader, it counts the items received (see Figure 6.4). If an item matches Tag i's $PID_r(i)$, the n Tag i stores the *count* value as its *rank*. This match establishes that Tag i belongs to the subgroup, authenticates the reader, sets its *rank* for responding to the reader, and verifies the integrity of $t[q](n), r_R(1)$. Tag i will use $t[q](n), r_R(1)$ as a part of its computation designated to the verifier. Tag i

will replace its $t_T(i)$ with the authenticated $t[q](n)$ in its tag memory. This value is the only value that changes in tag's memory when a tag is interrogated. Value $t_T(i)$ prevents replay attacks and impersonation by a malicious reader.

Tag i next computes values that will establish its presence to the verifier. Tag i will compute $M1(i)$ with a PRNG function including its unique $TID(i)$ and the current $t_T(i)$ (which is $t[q](n)$) value and will further randomize this with $r_R(1)$. Tag i then computes $m1(i)$ to enable the reader to authenticate the tag in its slot and verify the integrity of received messages. Tag i sends in slot $rank(i)$ two messages $M1(i), m1(i)$ to the reader.

```
Reader - Round 2
$M1^* \leftarrow 0$
for $j = 1$ to $d_q$
    if $m1(i), M1(i)$ message received in slot $j$
        $PID(i) \leftarrow m1(i) \oplus M1(i) \oplus s_{RT}$
    ; Reader computes the $PID(i)$ received in slot $j$
        if $(PID(i)$ computed $= LPID(j))$
        ; message integrity of $m1(i), M1(i)$ is verified
        ; Tag $i$ is authenticated
                $M1^* \leftarrow M1^* \oplus M1(i)$
                $valid(1, j) \leftarrow 1$      ; Tag $i$'s response is valid
$r_R(2) \leftarrow PRNG(seed_R)$
    ; reader does not update it seed here
    ; reader has a fresh seed set from the verifier for the next execution of the subroutine
$M1^*_r \leftarrow M1^* \oplus r_R(1)$
$m3_R \leftarrow s_{RT} \oplus r_R(2)$
$m4_R \leftarrow t[q](n) \oplus M1^*_r \oplus r_R(2)$
send to tags $m3_R, m4_R, M1^*_r$
```

Figure 6.5. Round 2: subroutine SUBGROUP.CHECK(q, n) - DGPP

Round 2 - Reader

The reader receives responses from tags in slots ordered by $\langle LPID_r(j)\rangle$ (see Figure 6.5). The reader processes information "slot-by-slot".

The reader computes $PID(i)$ received in slot j, using the received $M1(i), m1(i)$, and shared secret s_{RT}. If the computed $PID(i)$ matches the stored $LPID(j)$, then the reader has authenticated the tag and verified the integrity of received messages. Otherwise, the response is not valid and the reader discards $M1(i), m1(i)$. The reader flags all valid responses using *valid*$(1, j)$ (initialized to 0 in Round 1, Part 1). The reader repeats this for all tags' responses, building $M1^*$ from valid $M1(i)$ values. The reader generates a fresh random number $r_R(2)$ for its Round 2 message to the tags. It builds $m3_R, m4_R$, and $r_R(2)$ to allow an integrity check. The reader broadcasts three values to all tags in the group $m3_R, m4_R, M1^*_r$.

```
Tag $i$ - Round 2
$r_R(2) \leftarrow m3_R \oplus s_{RT}$
$t[q](n) \leftarrow m4_R \oplus r_R(2) \oplus M1^*_r$
if ($t[q](n)$ extracted $\neq t[q](n)$ stored )
    abort
      ; message integrity of $r_R(2), M1^*_r$ is verified if match
$M1^* \leftarrow M1^*_r \oplus r_R(1)$    ; tag extracts subgroup dependency message
$M2(i) \leftarrow PRNG(M1^* \oplus t[q](n)) \oplus r_R(2)$    ; a tag's secret designated for the Verifier
$m2(i) \leftarrow PID(i) \oplus M2(i) \oplus s_{RT}$
send to reader in slot $rank(i)$: $m2(i), M2(i)$
$rank(i) \leftarrow 0$    ; reset $rank$

; after a run, abort, or timeout, Tag $i$ discards all temporary stored variables and any computational results
```

Figure 6.6. Round 2: subroutine SUBGROUP.CHECK(q, n) - DGPP

Round 2 - Tag i

Recall that tags are left from Round 1 in open state, so tags do not need to re-authenticate the reader (Figure 6.6). In Round 2, Tag i extracts $r_R(2)$ from $m3_R$, then extracts $t[q](n)$ using the received $m3_R, M1^*_r$, and extracted $r_R(2)$. If the extracted $t[q](n)$ matches the one stored in Tag i, then this confirms the integrity of messages received from the reader. Tag i further removes the randomness from $M1^*_r$ using $r_R(1)$ and obtains the subgroup dependency message $M1^*$. Tag i will compute its second message to the verifier. This introduces parallel-dependency into the protocol as $M1^*$ includes computation by all tags. Each Tag i uses $M1^*$ in its computation of $M2(i)$. Tag i then computes $m2(i)$ to enable the reader to authenticate it in its slot and verify the integrity of received messages. Tag i sends in slot $rank(i)$ two messages $M2(i), m2(i)$ to the reader. Then Tag i resets its $rank(i)$ to 0.

```
**Reader - Proof construction in subroutine** SUBGROUP.CHECK$(q, n)$
$M2^* \leftarrow 0$
$valid.count \leftarrow 0$
for $j = 1$ to $d_q$
   if $m2(i), M2(i)$ message received in slot $j$ and $valid(1, j) = 1$
      $PID(i) \leftarrow m2(i) \oplus M2(i) \oplus s_{RT}$
      ; Reader computes the $PID(i)$ received in slot $j$
   if ($PID(i)$ computed = $LPID(j)$)
      ; message integrity of $m2(i), M2(i)$ is verified
      ; Tag $i$ is authenticated
            $M2^* \leftarrow M2^* \oplus M2(i)$
            $valid(2, j) \leftarrow 1$      ; Tag $i$'s response is valid
            $valid.count++$
Tag $i$ could have sent a valid response in Round 1 but not in Round 2 so $valid(2, j)$
indicates this, while if a tag did not send a valid response in Round 1, the proof does not
include the tag regardless of any Round 2 response.

subroutine SUBGROUP.CHECK $[q](n)$
$P[q](n) = (q, n, (j, valid(2, j), PID(i)), M1^*, M2^*, r_R(1), r_R(2))$      ; includes slot $j$ for
which $valid(1, j) = 1$
return ($valid.count$, $P[q](n)$)
```

Figure 6.7. Reader: subroutineSUBGROUP.CHECK(q, n) $[q](n)$ - DGPP

Proof construction in subroutine SUBGROUP.CHECK(q, n)

The reader handles information sent by tags in Round 2 (see Figure 6.7) in similar way as it does information sent by tags in Round 1. The reader receives responses from tags in $\langle LPID_r(j)\rangle$ order. The reader ignores messages in slots without a valid Round 1 message. The reader computes $PID(i)$ received in slot j using the received $M2(i), m2(i)$, and shared secret s_{RT}. If the computed $PID(i)$ matches the stored $LPID(j)$, then the reader has authenticated the tag and verified the integrity of received messages. Otherwise, the response is not valid and the reader discards $M2(i), m2(i)$. The reader flags all valid responses using $valid(2, j)$. Note that $valid(2, j) = 1$ indicates that Tag i has sent a valid response in Round 1 and in Round 2. The reader repeats this for all tags' responses building $M2^*$ from valid $M2(i)$ values. Both $M1^*, M2^*$ constructed will be included in the constructed proof. The reader construct the proof: $P[q](n) = (q, n, (j, valid(2, j), PID(i)), M1^*, M2^*, r_R(1), r_R(2))$, including slots j in which valid tags participated in Round 1. The subroutine SUBGROUP.CHECK(q, n)

returns (*valid.count*, $P[q](n)$) to the protocol.

6.3.2.3 Performance remarks

DGPP use only PRNG and XOR operations, which are within the capabilities of EPC Gen2 tags. In a run of $\textsc{Subgroup.Check}(q, n)$ a tag performs 3 PRNG operations, the reader performs 2 PRNG operations, and reader and tag communicate a total of $5m + 4$ items.

6.4 Security Analysis

6.4.1 Untraceability Analysis of DGPP

In this section, we classify DGPP in the untraceability model of Avoine *et al.* [8]. We will establish that DGPP is SIDE-Universal-UNT but not FORWARD-Existential-UNT.

Recall from Section 3.3.1 the following. $\mathcal{A}$ selects two challenge tags, i and j. Challenger $\mathcal{C}$ relabels the two challenge tags as $\widetilde{i}$ and $\widetilde{j}$ with respect to a random bit b such that if $b = 0$, then $\widetilde{i} = i$ and $\widetilde{j} = j$ (that is, the labels remain the same), but if $b = 1$, then $\widetilde{i} = j$ and $\widetilde{j} = i$ (that is, $\mathcal{C}$ swaps the labels). $\mathcal{A}$ must determine the value of b. If the probability that $\mathcal{A}$ can correctly determine b is not very different from $\frac{1}{2}$, then the protocol is untraceable in that class.

For the class SIDE-Universal-UNT, $\mathcal{A}_{\mathsf{SIDE}}$ cannot corrupt the challenge tags, but can corrupt any other tags immediately after $\mathcal{C}$ relabels the tags or after any number of executions of the protocol. That is, (c_1, c_2) can be any values, including (0, 0).

For the class FORWARD-Existential-UNT, $\mathcal{A}_{\mathsf{FORWARD}}$ can corrupt the challenge tags and any other tags after $\mathcal{C}$ executes the protocol c_1 times on Tag $\widetilde{i}$ and c_2 times on Tag $\widetilde{j}$ for some specified numbers (c_1, c_2).

Theorem 6.4.1 *DGPP is* SIDE-Universal-UNT*.*

Proof. For the SIDE-Universal-UNT class, adversary $\mathcal{A}_{\mathsf{SIDE}}$ cannot corrupt the challenge tags but can corrupt any other tags. $\mathcal{A}_{\mathsf{SIDE}}$ can also obtain transcripts of communications to and from all tags in all executions of the protocol.

For universal untraceability, (c_1, c_2) can be any values, and proving untraceability with $(c_1, c_2) = (0, 0)$ suffices [8]. That is, when $\mathcal{A}_{\mathsf{SIDE}}$ has selected Tags i and j, it has access to all transcripts for all tags in all previous executions of DGPP, and it has the information on which of these transcripts belong to which tags, including i and j. After the last execution of DGPP, $\mathcal{A}_{\mathsf{SIDE}}$ can corrupt all non-challenge tags, and $\mathcal{A}_{\mathsf{SIDE}}$ receives all transcripts of communications, however, the Challenger labels the transcripts for i and j in the last execution as belonging to $\widetilde{i}$ and $\widetilde{j}$ according to a random selection by the Challenger.

To establish that $i = \widetilde{j}$, $\mathcal{A}_{\mathsf{SIDE}}$ would have to establish that Tag i in one of the previous executions is the same as Tag $\widetilde{j}$. To do this, $\mathcal{A}_{\mathsf{SIDE}}$ would have to calculate $m1(i), M1(i), m2(i)$, or $M2(i)$ from the information it has obtained from transcripts and from corrupted non-challenge tags using information that matches Tag $\widetilde{j}$. From the communication transcripts, $\mathcal{A}_{\mathsf{SIDE}}$ holds that Tag i received inputs $m1_R, m2_R, \langle LPID_r(j)\rangle, m3_R, m4_R, M1^*_r$ (specific to this execution in an iteration n) and produced outputs $m1(i), M1(i), m2(i), M2(i)$, while Tag $\widetilde{j}$ received inputs $m1_R, m2_R, \langle LPID_r(j)\rangle, m3_R, m4_R, M1^*_r$ (specific to this execution in an iteration n') and produced outputs $m1(\widetilde{j}), M1(\widetilde{j}), m2(\widetilde{j}), M2(\widetilde{j})$. For each non-challenge Tag q, $\mathcal{A}_{\mathsf{SIDE}}$ can extract all its stored information (that is, $TID(q), a(q), t_T[q], s_{RT}$). Also, for Tag q, $\mathcal{A}_{\mathsf{SIDE}}$ can calculate all values in both rounds of Subgroup.Check.

Suppose $\widetilde{i}$ and $\widetilde{j}$ are in iteration n' and i is in iteration n. With s_{RT} extracted from a non-challenge tag, $\mathcal{A}_{\mathsf{SIDE}}$ can extract in n': $r_R(1), r_R(2)$, and $t[q](n')$ and in n: $r_R(1), r_R(2)$, and $t[q](n)$. These all will be different because of how the reader changes $r_R(1), r_R(2)$, and $t[q](n)$ in each run of Subgroup.Check. $\mathcal{A}_{\mathsf{SIDE}}$ can also extract $PID(\widetilde{i}), PID(\widetilde{j})$, and $PID(i)$ but cannot reverse the PRNG computation in PID to reveal any of $a(\widetilde{i}), a(\widetilde{j})$, or $a(i)$. Because either $\widetilde{j} = i$ or $\widetilde{i} = i$, then either $a(\widetilde{j}) = a(i)$ or $a(\widetilde{i}) = a(i)$. Without $a(\widetilde{i}), a(\widetilde{j})$, and $a(i)$,

$\mathcal{A}_{\mathsf{SIDE}}$ cannot tell which of $PID(\widetilde{i})$ or $PID(\widetilde{j})$ uses the same a value as $PID(i)$. So $\mathcal{A}_{\mathsf{SIDE}}$ cannot match $m1(\widetilde{i})$ or $m1(\widetilde{j})$ to $m1(i)$.

$\mathcal{A}_{\mathsf{SIDE}}$ also cannot calculate $M1(\widetilde{i}), M1(\widetilde{j})$, or $M1(i)$. In iteration n' $\mathcal{A}_{\mathsf{SIDE}}$ can extract $r_R(1)$ from $m1_R$ and in iteration n can extract $r_R(1)$ from $m1_R$. Also, $\mathcal{A}_{\mathsf{SIDE}}$ cannot reverse the PRNG computation in $M1$ to reveal any of $TID(\widetilde{i}), TID(\widetilde{j})$, or $TID(i)$. Because either $\widetilde{j} = i$ or $\widetilde{i} = i$, then either $TID(\widetilde{j}) = TID(i)$ or $TID(\widetilde{i}) = TID(i)$. Without $TID(\widetilde{i}), TID(\widetilde{j})$, and TID value, $\mathcal{A}_{\mathsf{SIDE}}$ cannot tell which of $M1(\widetilde{i})$ or $M1(\widetilde{j})$ uses the same $TID(i)$ as $M1(i)$. So $\mathcal{A}_{\mathsf{SIDE}}$ cannot match $M1(\widetilde{i})$ or $M1(\widetilde{j})$ to $M1(i)$.

$\mathcal{A}_{\mathsf{SIDE}}$ cannot match $m2(\widetilde{i})$ or $m2(\widetilde{j})$ to $m2(i)$ because of the same argument as for $m1$. In iteration n', $M2(\widetilde{i}) = M2(\widetilde{j})$, so $\mathcal{A}_{\mathsf{SIDE}}$ cannot match $M2(\widetilde{i})$ or $M2(\widetilde{j})$ to $M2(i)$ because it cannot tell them apart.

Therefore, $\mathcal{A}_{\mathsf{SIDE}}$ cannot match Tag $\widetilde{j}$ with Tag i any better than it can match Tag $\widetilde{i}$.

□

Theorem 6.4.2 *DGPP is not* FORWARD-Existential-UNT.

Proof. For the FORWARD-Existential-UNT class, adversary $\mathcal{A}_{\mathsf{FORWARD}}$ can corrupt the challenge tags and any other tags after a specified number of executions of the protocol by the Challenger $\mathcal{C}$. It can also obtain transcripts of communications to and from all tags in all executions of the protocol.

To prove that DGPP is not FORWARD-Existential-UNT, we will prove that challenge tags are traceable for any pair (c_1, c_2). The Challenger $\mathcal{C}$ executes the DGPP protocol for multiple executions without interference from $\mathcal{A}_{\mathsf{FORWARD}}$. In this proof, we will first use $(c_1, c_2) = (0, 0)$, then later we will show that even after Challenger $\mathcal{C}$ executes DGPP for any number of executions, tags will still be traceable. $\mathcal{A}_{\mathsf{FORWARD}}$ can corrupt any tags. $\mathcal{A}_{\mathsf{FORWARD}}$ can also obtain transcripts of communications to and from all tags in all executions of the protocol.

The key idea is that $TID(i)$ does not change.

To establish that $i = \widetilde{i}$ or $i = \widetilde{j}$, $\mathcal{A}_{\mathsf{FORWARD}}$ would have to establish that Tag i in one of the previous executions is the same as Tag $\widetilde{i}$ or $\widetilde{j}$ in the last execution.

After corrupting the challenge Tags $\widetilde{i}$ and $\widetilde{j}$, $\mathcal{A}_{\mathsf{FORWARD}}$ can obtain the following secrets, for $\widetilde{i}$: $a(\widetilde{i}), TID(\widetilde{i}), t[q](n')$ and for $\widetilde{j}$: $a(\widetilde{j}), TID(\widetilde{j}), t[q](n')$. $\mathcal{A}_{\mathsf{FORWARD}}$ recomputes the potential answers of these tags as follows, for $\widetilde{i}$:$M1(\widetilde{i}) \leftarrow PRNG(TID(\widetilde{i}) \oplus t[q](n')) \oplus r_R(1)$ and for $\widetilde{j}$: $M1(\widetilde{j}) \leftarrow PRNG(TID(\widetilde{j}) \oplus t[q](n')) \oplus r_R(1)$.

Using extracted s_{RT} (via corruption), $\mathcal{A}_{\mathsf{FORWARD}}$ can extract $r_R(1)$ and $t[q](n)$ in iteration n. Using $r_R(1)$ and $t[q](n)$ from iteration n and $TID(\widetilde{i})$ and $TID(\widetilde{j})$ $\mathcal{A}_{\mathsf{FORWARD}}$ calculates $PRNG(TID(\widetilde{i}) \oplus t[q](n)) \oplus r_R(1)$ and $PRNG(TID(\widetilde{j}) \oplus t[q](n)) \oplus r_R(1)$. One of these will equal $M1(i)$, identifying which of the Tag $\widetilde{i}$ or $\widetilde{j}$ is the same as Tag i.

Even if Challenger $\mathcal{C}$ then runs DGPP for any number of executions without interference from $\mathcal{A}_{\mathsf{FORWARD}}$, tags will be traceable because $TID(i)$ (and other secrets) does not change. This establishes that DGPP is not FORWARD-Existential-UNT.

□

CHAPTER 7
CARDINALITY ESTIMATION FOR TAG POPULATION

7.1 Introduction

In radio frequency identification (RFID) technology, a fundamental task is tag cardinality estimation. The reader tries to determined the tag population size speedily and accurately. Example applications for cardinality estimation include large-scale dense deployment for intelligent transportation, stadium management, conference, and warehouse systems with thousands or even millions of objects. A wide variety of tag estimation algorithms has been proposed in the literature [4, 47, 48, 60, 68, 72, 89, 131, 133].

We first formally discuss the problem (Section 7.2). Then we elaborate on the working environment and the idea of "estimators" (Section 7.3). We also introduce an algorithm template for a methodology for cardinality estimation and adopt this template for our approach (Section 7.6). In Section 7.4 we introduce new methods to obtain an initial "Phase 1" estimate of the cardinality. In Section 7.5 we present an analysis of the $(f_e - f_0)$ estimator on the $\{0, 1, e\}$ channel. This result is an extension of the GERT approach of Hasan *et al.* [48] for the $\{0, e\}$ channel. The result shows that both approaches, ours and GERT, exceed the requirements of the problem (at potentially added cost). We propose an approach to use simulation data (Section 7.6) to design an algorithm that better balances the cost and accuracy constraints of the problem.

7.2 Problem Statement

Suppose there are T tags in the vicinity of the reader. The aim of cardinality estimation of the tag population, henceforth called *cardinality estimation*, is to determine an estimate of the numbers of tags in the vicinity of the reader while meeting desired accuracy requirements.

Formally, for any real error probability $0 < \alpha < 1$ and real relative error $0 < \beta < 1$, the solution to cardinality estimation must satisfy:

$$\mathbf{Pr}\left[\frac{|T_{est} - T|}{T} \leq \beta\right] \geq \alpha \tag{7.1}$$

where T_{est} is the estimated value of the tag population size, T is the actual value of tag population size , and $\mathbf{Pr}(\cdot)$ is the probability of an event.

For example, if the exact number of tags is T =1000, the specified relative error $\beta = 3\%$ and target error probability $\alpha = 95\%$ then the output estimate T_{est} of an (α, β) accuracy requirement should be between 970 and 1030 at least 95% of the time.

7.3 Problem Environment

Consider a set of tags that are assumed to be anonymous (no IDs transmitted) and that possess minimal computational ability. In this work we require a tag, given an integer $f \geq 1$, to only be able to generate a random bit with probability $\frac{1}{f}$ of being a 1 and probability $(1 - \frac{1}{f})$ of being a 0. The tags communicate with the reader through a channel of resolution k (see Section 2.1.2). We will primarily consider $k \leq 2$ ($\{0, e\}$ and $\{0, 1, e\}$ channels, although some of the proposed ideas extend to large values of k).

The tags operate in a slotted (synchronous) environment in which a typical sequence of actions performed by tags includes (a) reading an integer $f \geq 1$ from the channel and (b) for each of the next f slots writing to the channel with probability $\frac{1}{f}$. This sequence of f slots is called a frame of size f. The remaining portions of the algorithms are executed on the, relatively more powerful, reader.

As a result of T tags writing independently on f slots of a frame, the channel returns a sequence of f output symbols to the reader. Formally, let $\{t_i \ : \ 1 \leq i \leq T\}$ be the set of tags. Let tag t_i write symbol $s_{i,j}$ on a slot, j $(1 \leq j \leq f)$ where $s_{i,j} \in \{0, 1\}$. Let the output symbol on slot j be $\hat{s}_j \in \{0, 1, ..., k-1, e\}$. Assume the input symbols to be numerical values $\{0, 1\}$ and output symbols $\{0, 1, ..., k-1, e\}$ $(e \geq 2)$ to also be numerical values, we have the following with $\sum$ denoting arithmetic addition:

$$\hat{s}_j = \begin{cases} \sum_{i=1}^{T} s_{i,j}, & \text{if } \sum_{i=1}^{T} s_{i,j} < k \\ e, & \text{otherwise} \end{cases}$$

For any output symbol $\hat{s} \in \{0, 1, ..., k-1, e\}$, define count $f_{\widehat{s}}$ to be

$$f_{\widehat{s}} = \sum_{j=1}^{f} 1_{\hat{s}}(\hat{s}_j)$$

where $1_a(v) = 1$ iff $b = a$. That is, f_0 is the number of $'0'$ output symbols in the f-slot frame, f_1 is the number of $'1'$ output symbols in the f-slot frame and so on.

An *estimator* is a function of the output symbol $\hat{s}_j$ (where $1 \leq j \leq f$) that is used to estimate the tag population. Typically estimators use $f_{\widehat{s}}$ in various ways for example, f_0, f_1, f_e, $f_e - f_0$ introduced earlier. Estimators include the average of 0 (EZB [60]), or the average of collisions e (UPE [59]), the average run length of non-zero slots (ART [131]), the length of continuous non-zero slots (LoF [89]), and the indices of the first non-zero slot for multiple rounds (FNEB [47]).

Figure 7.1 presents a template for an algorithm for cardinality estimation. For a set of T tags, let Phase 1 and Phase 2 require τ_1 and τ_2 time (slots). Observe that if the number of bits needed to represent the quantities f, n are constants relative to the tag size T, then the number of slots needed for Phase 2 (including any slots used to synchronization or separate iterations) is $a_1 + a_2 n + nf$, where a_1 and a_2 are constants. In this work we will assume that $a_1 = a_2 = 0$ since n and the constant overheads are generally small compared to f. Thus, the time for Phase 2 is assumed to be nf.

The above algorithm template can be further generalized to accommodate a different frame size at each iteration and a persistence probability to further temper the probability of a tag writing in a slot (with persistence probability $\widetilde{p}$, tags write a $'1'$ with probability $\frac{\widetilde{p}}{f}$; we have assumed $\widetilde{p} = 1$). Using persistence probability is like using a frame size of $\frac{f}{\widetilde{p}}$ with $\lceil n\widetilde{p} \rceil$ iterations. The algorithmic template could also be extended to allow any method of

```
Algorithm Cardinality Estimation
Phase 1: Get an initial estimate $T_0$.
Phase 2:
The reader uses $T_0$ to determine a frame size $f$ and the number of iterations $n$.
The reader broadcasts $f, n$ to all tags.

  for each slot $j$ (where $1 \leq j \leq f$) do
      Each tag independently writes symbol '1' (respectively, '0') to the channel with
          probability $\frac{1}{f}$ (respectively, $1 - \frac{1}{f}$).
      The reader reads the channel output symbol $\widehat{s_j}$ and computes an estimator value
          $E_j$.
  end
end
The reader computes the average estimator value $\overline{E} = \frac{1}{n}\sum_{i=1}^{n} E_j$ over all $n$ iterations. Then
uses an inverse operation on $\overline{E}$ to determine the final estimate $T_{est}$
```

Figure 7.1. Algorithm Cardinality Estimation

using $(\widehat{s_j})$ to determine E. These extensions accommodate most approaches in the literature. For Phase 1 we will use the well known algorithm of Willard [119] for leader election on a $\{0, 1, e\}$ channel (henceforth called WILLARD). We modify this algorithm to produce an estimate T_0 of the tag population. Our modification runs in $O(\log \log T_{\max})$ time, where $T_{\max}$ is the maximum number of tags that the system can expect. (In our simulation we have assume $T_{\max} = 2^{16} - 1$.) Further details of the Willard algorithm appear in Section 7.4. We also extend WILLARD algorithm to provide more accurate outputs. These extensions are called WILLARD+, WILLARD$*$, WILLARD+$*$. Section 7.4.2 describes these extensions.

We now briefly explain the approach used in Phase 2 to count the channel output symbols to obtain estimate T_{est}. Consider, for example that the estimator used is the number of zeroes f_0 in a frame. Figure 7.2 shows the average value of f_0 plotted as a function of $\frac{T}{f}$.

Consider any estimator E, for example consider f_0 estimator on Figure 7.2. Suppose f_0 was 0.7. Then it correspond to a $\frac{T}{f}$ value of 0.5. That is with an $f_0 = 0.7$ value one could expect that $\frac{T}{f} = 0.5$, However, there is a variance in the observed value of f_0. For this

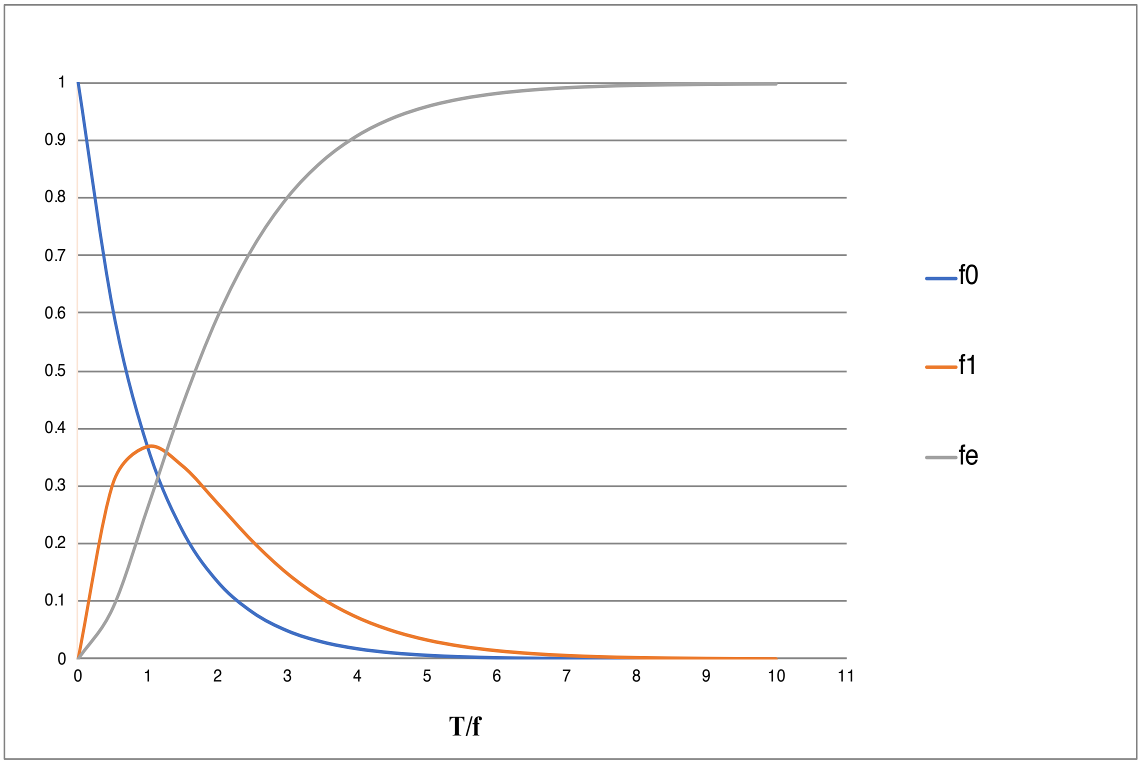


Figure 7.2. General nature of the quantities $\{f_0, f_1, f_e\}$

illustration suppose the observed value $f_0 = 0.7$ can vary between 0.6 and 0.8. Then the value of T can vary approximately between 0.4 and 0.6. This results is an average error of $\frac{|T_{est}-T|}{T}$ of $\frac{|0.4-0.5|}{0.5}$ or $\frac{|0.5-0.6|}{0.5} \leq 20\%$. On the other hand suppose $f_0 = 0.2$ with $T = 1.7$ and could vary between 0.3 and 0.1. Then $\frac{T}{f}$ can vary approximately between 1.1 and 2.3. Here the error is $\frac{|1.1-1.7|}{1.7}$ or $\frac{|2.3-1.7|}{1.7} \leq 35.3\%$. This is a higher slope of the average value of the estimator. Suppose that for the $f_0 = 0.7$ case the values the $\frac{T}{f}$ could vary between 1 and 0.3. Then the error could have been approximately $\frac{|0-0.5|}{0.5}$ or $\frac{|1.5-0.5|}{0.5} \leq 200\%$. Therefore, the variance of the estimators is also important. In addition, to high slope and low variance the average value of the estimators should be monotonic to facilitate inversion.

Observe that this plot decreases monotonically. Therefore the function $\psi(\frac{T}{f}) = f_0$ that gives the average value of f_0 for each value of $\frac{T}{f}$, has an inverse $\psi^{-1}(f_0) = \frac{T}{f}$. For a given value of f_0 and with the known value f, T can now be estimated. The n iterations serve to reduce the variance in the observed value f_0. Not all estimators are equally effective (particularly in certain range of values of $\frac{T}{f}$ [59]) and some are not monotonic (for example f_1, see Figure 7.2). In our work we will use the estimator $f_e - f_0$, which showed promise in initial investigations (see Section 7.3.1).

Table 7.1 summarizes the main notation used across this chapter.

Table 7.1. Notation Cardinality Estimation.

f	frame size
f_0	number of slots in a frame with no write
f_1	number of slots in a frame with one write
f_e	number of slots in a frame with more than one write
T_{est}	an estimate of a tag population
T_0	tag population size estimate in Phase 1
T	the actual tag population
$\widetilde{p}$	persistence probability
$\frac{T\widetilde{p}}{f}$ or Tp	system load ratio
E	an estimator

7.3.1 Empirical Study of the Behavior of Estimators

The quantities $\{f_0, f_1, f_e\}$ are critical to this work. Figure 7.2, and after shows the general nature of these quantities. In our algorithm in Figure 7.1, and after 7.1 the reader uses a function from its reading f_0, f_1, f_e to determine an estimate of T. Several standard estimations appear in the literature that include f_0, f_1, f_e.

We study many combinations of f_0, f_1, f_e such as: $\frac{f_0}{f}$, $\frac{f_1}{f}$, $\frac{f_e}{f}$, $\frac{f_0}{f_1}$, $\frac{f_0}{f_e}$, $\frac{f_1}{f_0}$, $\frac{f_1}{f_e}$, $\frac{f_e}{f_0}$, $\frac{f_e}{f_1}$, $f_0 + f_1$, $f_0 + f_e$, $f_0 + f_1$, $f_1 + f_e$, $f_0 - f_1$, $f_e - f_1$, $f_0 - f_e$, $\frac{f_0 f_e}{f_1}$, $\frac{f_1 f_e}{f_0}$, $\frac{f_0 f_1}{f_e}$, $\frac{f_0}{f_1 f_e}$, $\frac{f_1}{f_0 f_e}$, $\frac{f_e}{f_1 f_0}$, $\sqrt{\frac{f_0 f_e}{f_1}}$, $\sqrt{\frac{f_1 f_e}{f_0}}$, $\sqrt{\frac{f_0 f_1}{f_e}}$, $\frac{f_1}{\sqrt{f_0 f_e}}$, $\frac{f_0}{\sqrt{f_1 f_e}}$, $\frac{f_e}{\sqrt{f_0 f_1}}$, as potential estimators to be used.

For each estimator we plotted its average value and variance across a large number of trials and tag populations (see Figures 7.3 and 7.4). Notice that all estimators have a small slope and low variance for large values of $\frac{T}{f}$. That is, one need a large frame size f for

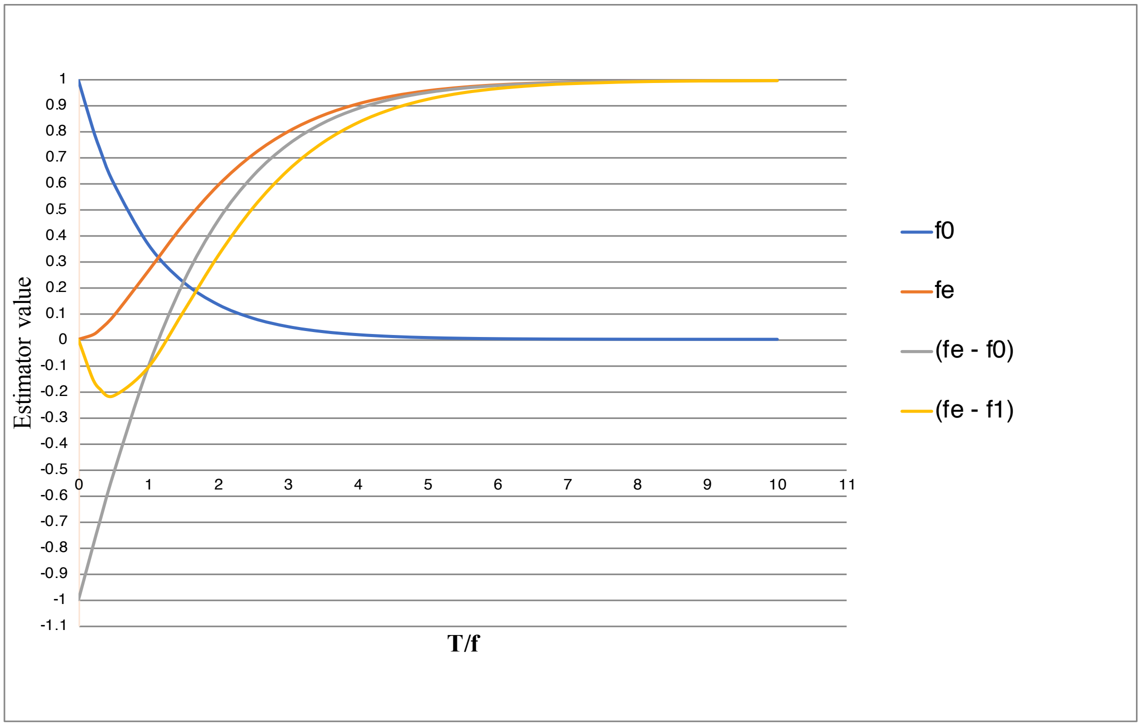


Figure 7.3. Average value for potential estimators

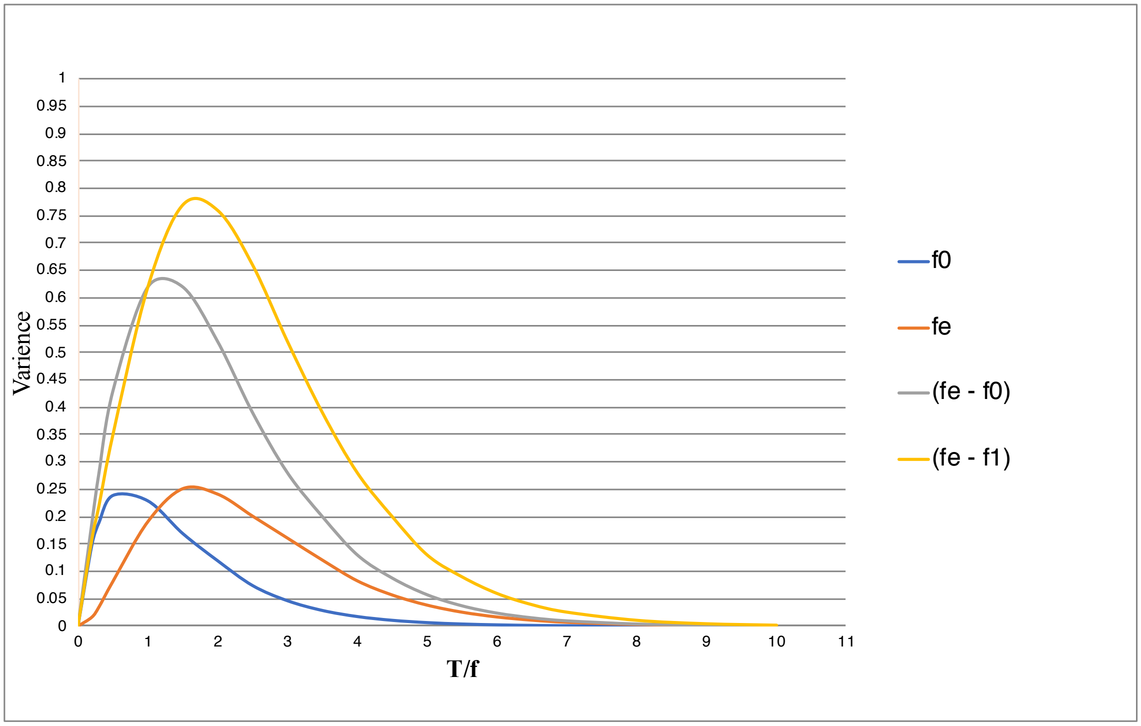


Figure 7.4. Variance for potential estimators

some level of accuracy. For small values of $\frac{T}{f}$ the $f_e - f_0$ estimator has the large sloop with a somewhat larger variance. Estimators investigating f_1 are non-monotonic and were not consider functions.

7.4 Phase 1 Algorithms

The algorithms for Phase 1 that we use in this work are all based on the work of Willard [119]. We will discuss the adaptation of Willard's algorithm to this work. Then we will discuss extensions for higher accuracy. We will also illustrate the performance of these algorithms.

7.4.1 The Algorithm Willard

The leader election of Willard [119] searches for a leader from a set of T entities (tags in our case). It proceeds in two stages. The first stage determines (for the most part) a method to estimate the value of T and that is what we use for this work. Consequently we do not describe Stage 2 of Willard's algorithm. What we describe below is the adaptation used for this work. We will call the adaptation Willard.

The algorithm Willard proceeds as follows. First, observe that if T tags write to the channel with probability $\frac{1}{T_0}$, then the average numbers of writes is $\frac{T}{T_0}$.

Suppose $1 \leq T \leq 2^m$, that is, T is an m-bit number. The algorithm first checks estimate $T_0 = 2^{\frac{m}{2}}$ as a potential value of T. Every tag writes to the channel with probability $\frac{1}{T_0}$ and the reader checks the channel output. If the channel output is $'0'$ then nominally the probability $\frac{1}{T_0}$ is too small and the estimate T_0 is too large. If the channel output is e, then nominally the estimate T_0 is too small. If the channel output is $'1'$, then (on average) the estimate is about right. If the estimate is too small (resp., large), then it is increased (resp., decreased) in a binary search manner.

As an example, if $T = 1000$, $m = 16$, then the first guess $T_0 = 2^8$ nominally produces an e. So the next T_0 is to be $2^{\frac{(16+8)}{2}} = 2^{12} = 2048$. This time the channel nominally returns a 0 and so T_0 will reduce to $2^{\frac{(12+8)}{2}} = 2^{10} = 1024$. The channel output could now be $'1'$.

In general, Willard terminates either when the symbol $'1'$ is output by the channel or when the binary search has narrowed things down to a single bit position b (where $0 \leq b < m$). In the latter case the last symbol read could be a $'0'$,$'1'$, or e. Suppose Willard terminates with symbol $'s'$ and bit position b. Then the returned estimate T_0 is as follows:

$$T_0 = \begin{cases} 2^b, & \text{if } s = \text{'1'}; \\ \left\lceil \frac{2^b + 2^{b-1}}{2} \right\rceil & \text{if } s = \text{'0'}; \\ \frac{2^b + 2^{b+1}}{2} & \text{if } s = e; \end{cases}$$

When $s = '1'$ then the algorithm assumes that T_0 is close to T. When the algorithm returns a $'0'$ then the probability $\frac{1}{2^b}$ was nominally too low to elicit any writes from the tags. In other words, 2^b is too high an estimate. Since the binary search nominally indicated that $2^{b-1} < T < 2^b$ it selects T_0 to be somewhere in the middle of the range. The ceiling is to account for $b = 0$. The case $s = e$ is similar.

This approach, on an average, can be shown to produce a value roughly within a factor of 2 of the correct value. However, this is not good enough as β (error requirement) is typically much higher and a second phase is needed. It is clear that if $T_{\text{max}} = 2^m$, then the algorithm runs in $O(\log m) = O(\log \log T_{max})$ time. Figures 7.7 and 7.9 show the performance of Willard over a range of values of T. The plot is based on 100,000 runs of Willard for some values of T in the range from 2 to 10,000.

We now modify Willard along two directions that improve its accuracy at the cost of more time.

7.4.2 Extensions of Willard

Observe that when $2^b < T < 2^{b+1}$ then even when Willard operates as expected on an average, the value of T can be anywhere in a range $(2^b, 2^{b+1})$of size approximately 2^b. As b increases, this range also increases. Algorithm Willard+ addresses this issue.

Suppose WILLARD terminates with symbol $'0'$ at bit b. Then nominally $2^{b-1} \leq T < 2^b$ or $0 \leq T - 2^{b-1} < 2^{b-1}$. One could run WILLARD again on range $[0, 2^{b-1})$ assuming an offset of 2^{b-1}. If WILLARD now produced a $'1'$ symbol with bit $0 \leq b_1 \leq b-1$ then we could now assume that the estimate is close to $2^{b_1} + 2^{b-1}$. This process proceeds recursively until the estimate is down to a resolution value of 1. As an example, consider $T = 28$. Let WILLARD run close to the expected behavior average case. Then the first time of WILLARD (called by WILLARD+) may terminate with symbol $'0'$ at bit position 5. This indicates that $2^4 < T < 2^5$ or $0 < T - 2^4 < 2^4$. The second recursive call to WILLARD will now be on range $[0, 2^4)$ but with an "offset" 2^4. This could return a bit position of 3 again with symbol $'e'$ indicating that $2^3 \leq T - 2^4 \leq 2^4$ or $0 < T - 2^3 - 2^4 < 2^3$. We now called WILLARD with range $[0, 2^3)$ and an offset of $2^3 + 2^4$. This third call may produce a $'1'$ on bit position 1, so the final estimate would be $2^1 + 2^3 + 2^4 = 25$. Figure 7.5 illustrate the general behavior of WILLARD+.

Value returned by WILLARD	Last channel symbol	Nominal range	starting range for WILLARD+
i	0	$[2^{i-1}, 2^i]$	$2^{i-1} + [0, 2^{i-1}]$
	1	$[2^i, 2^i]$	$2^i + [0, 0]$
	e	$[2^i, 2^{i+1}]$	$2^i + [0, 2^i]$

Figure 7.5. WILLARD+ ranges of an estimate

The worst case behavior of WILLARD+ happens when T is very large. The first call to WILLARD on range $[0, 2^m)$ causes a next call on range $[0, 2^{m-1})$, and the third on $[0, 2^{m-2})$ and so on. We know that a call to WILLARD on range $[0, 2^m)$ runs in $O(\log m)$ time. So the time for WILLARD+ is $O\left(\sum_{i=1}^{m} i\right) = O\left(m^2\right)$. Since 2^m represents an upper bound on T, the worst case time for WILLARD+ is $O\left(\log^2 T_{max}\right)$.

Algorithm WILLARD+ repeatedly calls WILLARD to obtain a more resolved value for

T_0. However, it does not address a fundamental issue of inaccuracy with WILLARD. A bad decision made early on cannot be compensated for by additional coin tosses. For example, let $T = 28$ and let $T_{max} = 2^{16}$. At the very first iteration all 28 tags write to the channel with probability $p = \frac{1}{2^8} = \frac{1}{256}$. Nominally we should expect a $'0'$ from the channel. If a $'1'$ is produced then T_0 is estimated to be 256. If it produces an e, the algorithm will not let T_0 be smaller than 256. The probability of not obtaining a $'0'$ in this step is $1 - \left(1 - \frac{1}{256}\right)^{28} \cong 0.1$, which is not too small.

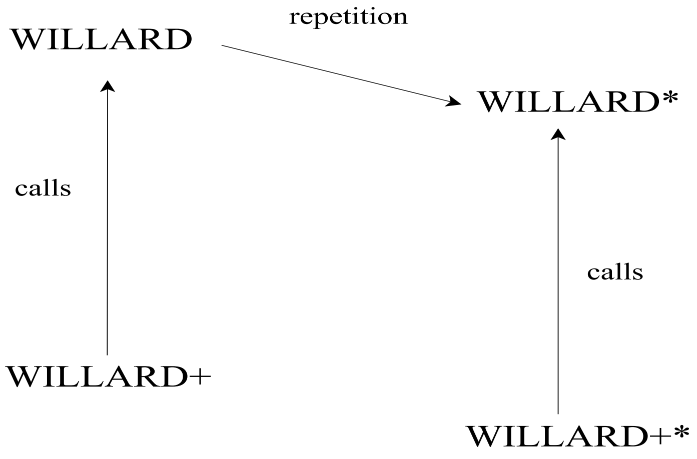


Figure 7.6. Extensions of WILLARD

Figure 7.6 shows all discussed extensions of WILLARD in Phase 1.

Algorithm WILLARD$*$ addresses this by repeating this early coin flip several times and selecting a “majority” value. Typically when $p = 2^{-x}$ and 2^x is far from T, symbols $'0'$ and e are much more probable. Repeating the test several times reinforces the probable outcome. In the previous example, the probability of not obtaining at least two $'0'$’s in three tries is about $0.1^3 + \binom{3}{2}0.1^2 = 0.031$.

When $p = 2^{-x}$ and 2^x is close to T, then the probability of obtaining $'0'$, $'1'$ or e are all quite close to each other. This is because for large T and $p \cong \frac{1}{T}$, probability of $'1'$ is $Tp(1-p)^{(T-1)} \cong (1-\frac{1}{T})^T \cong e^{-1}$. Also the probability of $'0'$ is $(1-p)^T \cong e^{-1} \cong 0.368$, so the probability of getting an e is $1 - \frac{2}{e} \cong 0.264$ (see Figure 7.2) around the 0.3 estimator value). In this case where the numbers of times $'0'$, $'1'$ or e occur are not that different, we will assume the outcome to be $'1'$.

There is no fixed method to repeat the test at each slot. However, recognizing that large values of x (where $p = 2^{-x}$) have large consequences for errors and that indiscriminate repetition can increase the running time for the algorithm, we select to repeat a trial $p = 2^{-x}$, x times. Clearly this is not the only way to implement WILLARD$*$.

Figures 7.7 - 7.9 show results of running WILLARD, WILLARD+, and WILLARD$*$ for various values of T each run a 100,000 times.

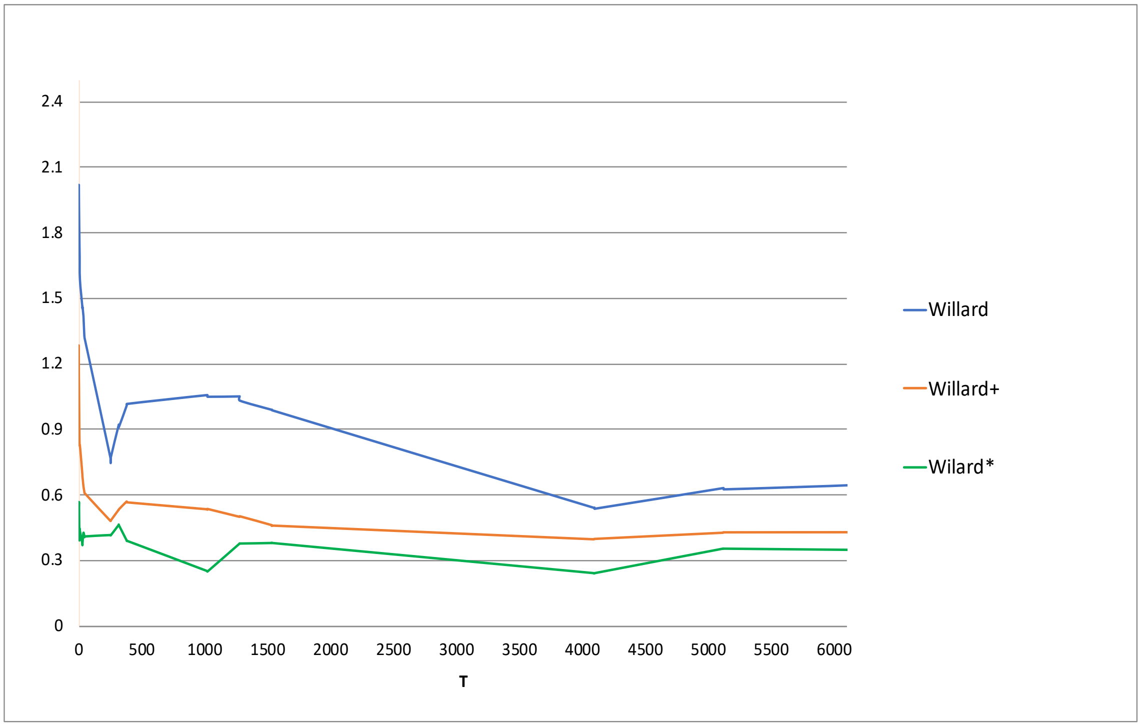


Figure 7.7. Average error WILLARD, WILLARD+, WILLARD$*$

WILLARD+$*$ has not yet been implemented, but we expect better results from it that the other two extensions of Willard. Specifically, these figures show the average error and

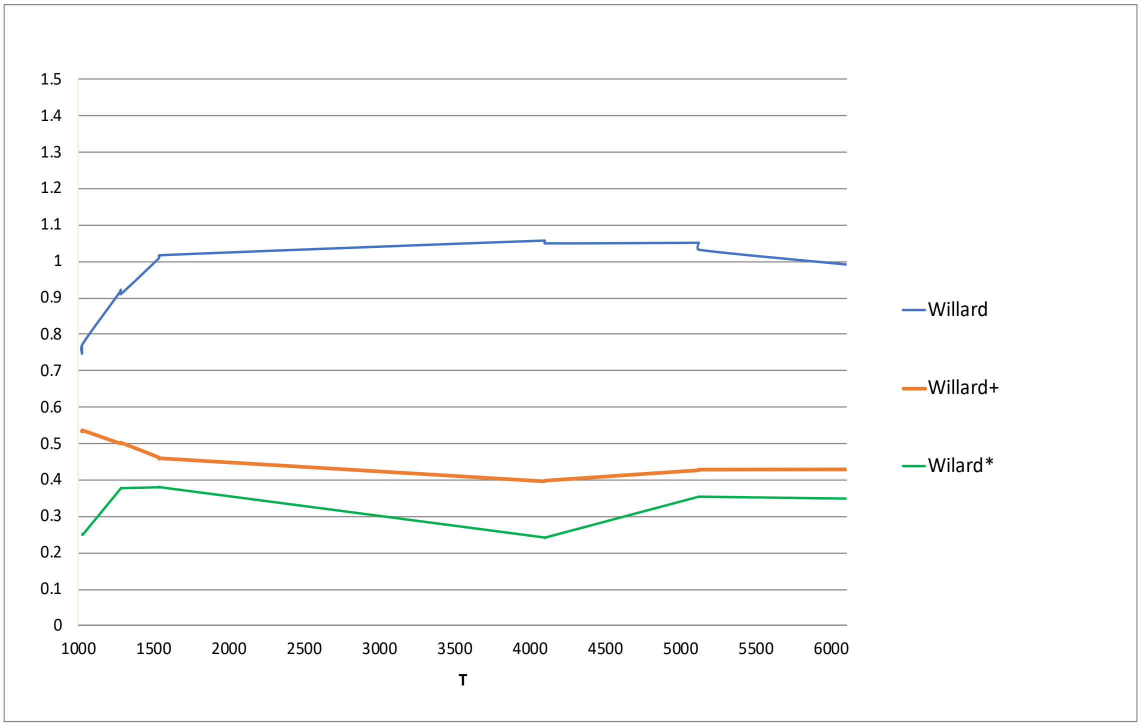


Figure 7.8. Average error WILLARD, WILLARD+, WILLARD∗ for T values in range 1000-6000

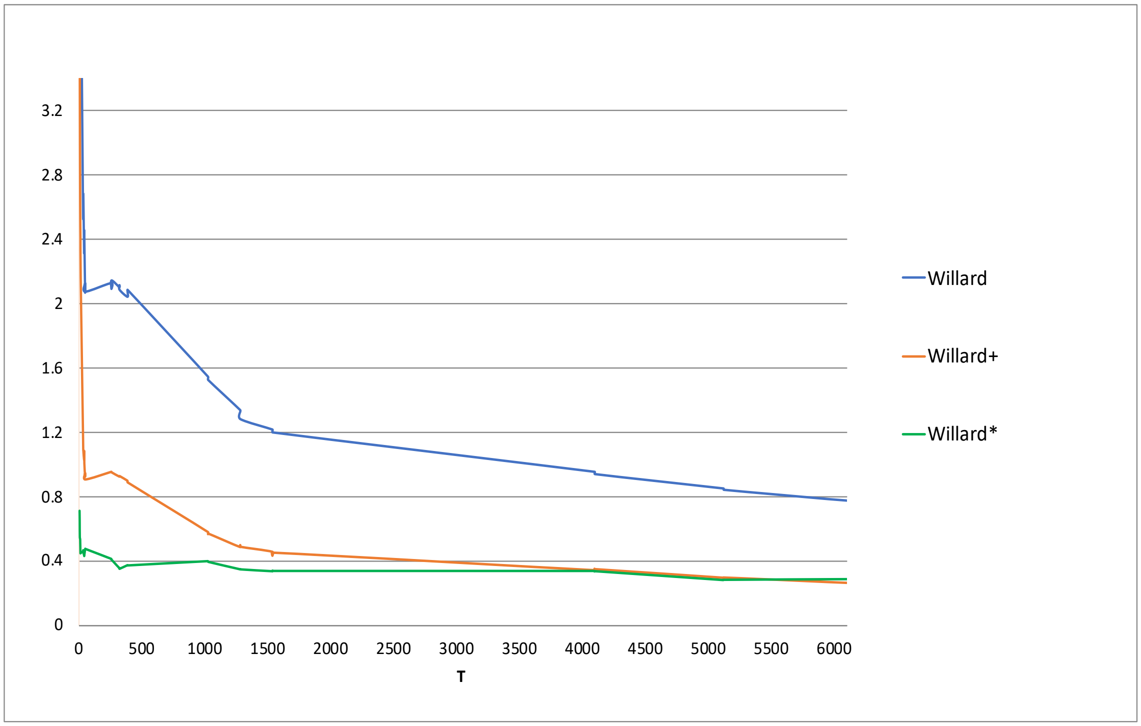


Figure 7.9. Error varinace WILLARD, WILLARD+, WILLARD∗

error variance of the three methods implemented in Phase 1. The values in Figures 7.7–7.9 we choose $1 \leq T \leq 10,000$, for $k = 2, 3, 5, 8, 10, 12$, are as follows: $2^k - 1$; 2^k; $2^k + 1$; $\left(\frac{2^k+2^{k+1}}{2}\right) - 1$; $\frac{2^k+2^{k+1}}{2}$; $\left(\frac{2^k+2^{k+1}}{2}\right) + 1$; $\left(\frac{3.2^k+2^k}{2}\right) - 1$; $\frac{3.2^k+2^k}{2}$; $\left(\frac{3.2^k+2^k}{2}\right) + 1$

It can be seen that WILLARD$*$ runs $O(\log T_{max} \log\log T_{max})$ time.

Another approach is to repeat the test proportional to the range of bit positions in the current iteration. This will make the time $O(\log^2 \log T_{max})$.

WILLARD$+$ calls WILLARD multiple times, or WILLARD$*$. So one could also design a WILLARD$+*$ algorithm that calls WILLARD$*$ in the same way as WILLARD$+$ calls WILLARD.

7.5 Analytical Results for the $f_e - f_0$ on the $\{0, 1, e\}$ Channel

Hasan *et al.* [48] introduced an estimator for the $\{0, e\}$ (often called a $\{0, 1\}$) channel. This estimator is $f_e - f_0$ (or $f_1 - f_0$ in the $\{0, 1\}$ setting). We followed the same analytical approach to extend the work to a $\{0, 1, e\}$ channel. We too use the $f_e - f_0$ estimator; however, it has different meaning compared to that in the $\{0, e\}$ channel. As a result, several of our parameters and intermediate quantities were also different, some more involved.

Notable differences in the extension are in the derivation of a key parameter κ needed to ensure a Gaussian distribution of the estimator values. As before T is the tag population and T_{est} (output of the tag estimation) is an estimate of T. For a given frame size f we define the normalized $f_e - f_0$ estimator, $C_{(e-0)}$, and its expected value, $\mu_{(e-0)}$, as follows:

$$C_{(e-0)} = \frac{(f_e - f_0)}{f} \tag{7.2}$$

$$\mu_{(e-0)} = \mathbf{Exp}[C_{(e-0)}] = \mathbf{Exp}\left[\frac{(f_e - f_0)}{f}\right] \tag{7.3}$$

Let X_{ij} represents the random variable that tag i writes on slot j; every tag writes on a slot with probability $p = \frac{\widetilde{p}}{f}$ (where $\widetilde{p}$ is the persistence probability, we assume $\widetilde{p} = 1$ (see Section 7.3) and f is the frame size). Here X_{ij} is a Bernoulli variable with probability p of success. Therefore

$$X_{ij} = \begin{cases} 1, & \text{with probability } p \\ 0, & \text{with probability } 1-p \end{cases}$$

Let Y_j be the random variable representing the number of tag writes in slot j.

$$Y_j = \sum_{i=1}^{T} X_{ij} \tag{7.4}$$

The probability distribution for Y_j's is as follows:

$$Y_j = \begin{cases} 0, & \text{with probability } p_0 = (1-p)^T \\ 1, & \text{with probability } p_1 = Tp(1-p)^{(T-1)} \\ e, & \text{with probability } p_e = 1 - (p_0 + p_1) \end{cases} \tag{7.5}$$

Here p_0 is the probability that a slot has no writes, p_1 is the probability that a slot has exactly one write, and p_e is the probability that a slot has more than one write. To help with the analysis, we will be use the following indicators:

$$Y_j^{(0)} = \begin{cases} 1, & \text{if } Y_j = 0; \\ 0, & \text{if } Y_j \neq 0; \end{cases}$$

$$Y_j^{(1)} = \begin{cases} 1, & \text{if } Y_j = 1; \\ 0, & \text{if } Y_j \neq 1; \end{cases}$$

$$Y_j^{(e)} = \begin{cases} 1, & \text{if } Y_j \geq 2 \\ 0, & \text{if } Y_j < 2; \end{cases}$$

Let

$$C_{j(e-0)} = Y_j^{(e)} - Y_j^{(0)} \tag{7.6}$$

For any symbol $s \in \{0, 1, e\}$ let the number of times s occurs in the f slot frame be $f_s =$

$\sum_{i=1}^{f} Y_j^{(s)}$. Then

$$C_{(e-0)} = \frac{f_e - f_0}{f} = \frac{1}{f}\left(\sum_{i=1}^{f} Y_j^{(e)} - \sum_{i=1}^{f} Y_j^{(0)}\right) = \frac{1}{f}\sum_{i=1}^{f}(Y_j^{(e)} - Y_j^{(0)}) = \frac{1}{f}\sum_{i=1}^{f} C_{j(e-0)} \tag{7.7}$$

The probability mass function for $C_{j(e-0)}$ is as follows:

$$C_{j(e-0)} = \begin{cases} -1, & \text{with probability } p_0 = (1-p)^T \\ 0, & \text{with probability } p_1 = Tp(1-p)^{(T-1)} \\ 1, & \text{with probability } p_e = 1 - (p_0 + p_1) \end{cases}$$

Recall that X_{ij} is Bernoulli random variable (indicating the write in a slot by a tag), and Y_j is the sum of X_{ij} (number of writes). Using Equation (7.6) we determine the mean (expected value) and variance of $C_{j(e-0)}$ and $C_{(e-0)}$ as follows:

$$\mu_{j(e-0)} = \mathbf{Exp}[C_{j(e-0)}] = -1p_0 + 0p_1 + 1p_e = p_e - p_0 = 1 - 2p_0 - p_1 \tag{7.8}$$

From Equation (7.8)

$$\begin{aligned}\mu_{(e-0)} = \mathbf{Exp}[C_{(e-0)}] = \mathbf{Exp}\left[\frac{1}{f}\sum_{i=1}^{f} C_{j(e-0)}\right] &= \frac{1}{f}\sum_{i=1}^{f}\mathbf{Exp}\left[C_{j(e-0)}\right] \\ &= \frac{1}{f}(f(p_e - p_0)) = p_e - p_0 = 1 - 2p_0 - p_1 \end{aligned}\tag{7.9}$$

$$\begin{gathered}\mathbf{Var}[C_{j(e-0)}] = \sigma^2_{j(e-0)} = \mathbf{Exp}[C^2_{j(e-0)}] - \mathbf{Exp}[C_{j(e-0)}]^2 = \\ \left((-1)^2p_0 + 0^2p_1 + 1^2p_e\right) - (p_e - p_0)^2 = \left(p_e + p_0 - (p_e - p_0)^2\right)\end{gathered}\tag{7.10}$$

$$\mathbf{Var}[C_{(e-0)}] = \sigma^2_{(e-0)} = \frac{1}{f}\left(p_e + p_0 - (p_e - p_0)^2\right) \tag{7.11}$$

Let $\ell = Tp$ (which is the average number of writes per slot). Observe that $p_0 = (1-p)^T =$

$(1-\frac{\ell}{T})^T \approx e^{-\ell}$, $p_1 = Tp(1-p)^{(T-1)} \approx \ell e^{-\ell}$, and $p_e = 1-(p_0+p_1) \approx 1-(e^{-\ell}+\ell e^{-\ell})$. The expression in the below extension of Equation (7.5) can be written as

$$\mathbf{Pr}(Y_j = Y) \cong \begin{cases} e^{-\ell}, & \text{if } Y = 0 \\ \ell e^{-\ell}, & \text{if } Y = 1 \\ 1-(e^{-\ell}+\ell e^{-\ell}), & \text{if } Y \geqq 2 \end{cases}$$

7.5.1 Gaussian Approximation of the $f_e - f_0$ Estimator

Recall that the problem requires (see Section 7.2) the following.

$$\mathbf{Pr}\left[\frac{|T_{est}-T|}{T} \leq \beta\right] \geq \alpha,$$

Recall that f_e (resp. f_0) is the number of occurrences of e (resp. 0) in the frame. To perform cardinality estimation of the tag population size while maintaining the accuracy requirements given above we need a well approximated probability distribution function for $C_{(e-0)}$. Recall from Figure 7.3, that $C_{(e-0)}$ or $f_e - f_0$ is a monotonically increasing function of T (or $\frac{T}{f}$) and is, therefore, invertible. However, given a value of $C_{(e-0)}$ there could be an error if $C_{(e-0)}$ is just inverted to get a value of T. This is because $C_{(e-0)}$ is a random variable taking a range of values. Here, as in GERT [48], we determine the conditions for the distribution of $C_{(e-0)}$ for a fixed value $\frac{T}{f}$ to be Gaussian.

The Lindeberg Feller Theorem with the special case of triangular array CLT [14] (also used by Hasan *et al.* [48]) gives a set of conditions for $C_{j(e-0)}$ to be Gaussian (or normal).

Theorem 7.5.1 (Lindeberg Feller [14]) *Let* $\{X_j\}$ *be an array of independent random variables with* $\mathbf{Exp}[X_j] = 0$ *and* $\mathbf{Exp}[X_j^2] = \sigma_j^2$, $Z = \sum_{j=0}^{f} X_j$ *and* $B^2 = \mathbf{Var}(Z) = \sum_{j=0}^{f} \sigma_j^2$, *then* Z *is a normal distribution with zero mean and* B^2 *variance if the condition below holds for every* $\epsilon > 0$.

$$\frac{1}{B^2}\sum_{j=0}^{f}\mathbf{Exp}\left[\{X_j^2 : |X_j| > \epsilon B\}\right] \longrightarrow 0 \tag{7.12}$$

We know that $\mathbf{Exp}[C_{j,(e-0)}] = \mu_{(e-0)}$. The Lindeberg Feller theorem requires the variable to have zero mean. To satisfy this requirement, we define $\widetilde{C}_{j(e-0)} = C_{j(e-0)} - \mu_{(e-0)}$. So $\mathbf{Exp}[\widetilde{C}_{j(e-0)}] = 0$ and the variance $\sigma^2_{j(e-0)}$ is the same for $C_{j(e-0)}$ and $\widetilde{C}_{j(e-0)}$.

We now have a set of independent random variables $\widetilde{C}_{j(e-0)}$, with $\mathbf{Exp}[\widetilde{C}_{j(e-0)}] = 0$, $\mathbf{Var}[\widetilde{C}_{j(e-0)}] = \sigma^2_{j(e-0)}$. Observe that,

$$\widetilde{C}_{j(e-0)} = \begin{cases} -1-\mu_{(e-0)}, & \text{with probability } p_0 \\ -\mu_{(e-0)}, & \text{with probability } p_1 \\ 1-\mu_{(e-0)}, & \text{with probability } p_e \end{cases} \tag{7.13}$$

Let

$$Z = \sum_{j=1}^{f}\widetilde{C}_{j(e-0)} = \sum_{j=1}^{f} C_{j(e-0)} - \mu_{(e-0)}$$

$$\mathbf{Var}(Z) = \sum_{j=1}^{f}\mathbf{Var}(\widetilde{C}_{j(e-0)}) = f\sigma^2_{j(e-0)}$$

According to the Lindeberg Feller Theorem, Z will be asymptotically normal with mean 0 and variance $\sigma^2_{j(e-0)}$ if the condition below holds for every $\epsilon > 0$

$$\frac{1}{f\sigma^2_{j(e-0)}}\sum_{j=0}^{f}\mathbf{Exp}\left[\left\{\widetilde{C}_{j(e-0)} : \left|\widetilde{C}_{j(e-0)}\right| > \epsilon\sqrt{f}\sigma_{j(e-0)}\right\}\right] \longrightarrow 0 \tag{7.14}$$

In the condition of membership in the set $\left\{\widetilde{C}_{j(e-0)} : \left|\widetilde{C}_{j(e-0)}\right| > \epsilon\sqrt{f}\sigma_{j(e-0)}\right\}$ from Equation (**??**), we consider three cases when $\widetilde{C}_{j(e-0)}$ is not counted in the expected value.These following conditions correspond to the three possibilities of Equation (7.13). For brevity we will use μ to indicate $\mu_{(e-0)}$ and $\lambda = \epsilon\sqrt{f}\sigma_{j(e-0)}$.

$$\begin{aligned} |-1-\mu| &\leq \lambda \\ |-\mu| &\leq \lambda \\ |1-\mu| &\leq \lambda \end{aligned} \tag{7.15}$$

So we required $\lambda \geq \max\{|-1-\mu|, |-\mu|, |1-\mu|\}$ We now consider two possibilities.

- [Case $-1 \leq \mu \leq 0$]: Here $|-1-\mu| = 1-\mu$. The first condition of Equation (7.17) becomes $-1-\mu \leq \lambda$, for the second condition of Equation (7.17) we have $|-\mu| = -\mu \leq \lambda$. The third condition becomes $|1-\mu| = 1-\mu \leq \lambda$. In summary $\lambda \geq \max\{1+\mu, 1-\mu\} = 1-\mu$. So here require $\lambda \geq 1-\mu$.

- [Case $0 \leq \mu \leq 1$]: Here $|-1-\mu| = 1+\mu$, $|-\mu| = \mu$, and $|1-\mu| = 1-\mu$. Clearly $\lambda \geq \max\{1+\mu, \mu, 1-\mu\} = 1+\mu$.

Thus $\widetilde{C}_{j(e-0)}$ will not contribute to the expected value in Equation (7.14) when $\lambda \geq 1+|-\mu|$. When this happens the simulation in Equation (7.14) becomes equal to 0.

Let

$$\kappa \leq \epsilon^2 f = \frac{\lambda^2}{\sigma^2_{j(e-0)}} = \frac{\lambda^2}{\sigma^2} \tag{7.16}$$

Then $\sqrt{\kappa}\sigma \leq \lambda$. The condition that $\lambda \geq 1-|\mu|$ can be satisfied if $\sqrt{\kappa}\sigma \geq 1-|\mu|$ or $\kappa\sigma^2 \geq (1-|\mu|)^2$. We now consider the two cases for $-1 \leq \mu \leq 0$ and $0 \leq \mu \leq 1$. Let the κ values for this two cases be κ_1 and κ_2. So if $\kappa_1 \geq \frac{(1-\mu)^2}{\sigma^2}$ and $\kappa_2 \geq \frac{(1+\mu)^2}{\sigma^2}$, and $\kappa \geq \kappa_1, \kappa_2$ then our conditions for Equation (7.13) tending to 0 be satisfied.

Observe that $\sigma^2 = \sigma^2_{j(e-0)} = p_e + p_0 - (p_e - p_0)^2 = 1-(p_0+p_1)+p_0-(1-(p_0-p_1)-p_0)^2$. Substituting $p_0 \cong e^{-\ell}$, $p_1 \cong \ell e^{-\ell}$, and $p_e \cong 1-(e^{-\ell}+\ell e^{-\ell})$, we have $\sigma^2 \cong e^{-\ell}(4+\ell-e^{-\ell}(2+\ell)^2)$. Next, $1-\mu = 1-(p_e-p_0) = 2p_0+p_1 \cong e^{-\ell}(2+\ell)$. So $(1-\mu)^2 \cong e^2 - \ell(2+\ell)^2$ and for the $-1 \leq \mu \leq 0$ case we have,

$$\kappa_1 \geq \frac{(1-\mu)}{\sigma^2} \cong \frac{e^{-\ell}(2+\ell)^2}{4+\ell-e^{-\ell}(2+\ell)^2} \tag{7.17}$$

For the $0 \leq \mu \leq 1$ case we have

$$\kappa_2 \geq \frac{(1+\mu)}{\sigma^2} \cong \frac{(2-e^{-\ell}(2+\ell))^2}{e^{-\ell}(4+\ell-e^{-\ell}(2+\ell)^2)} \tag{7.18}$$

With $\ell = Tp$, it can be verified for $\ell \geq 1.1462$, $\kappa_2 \geq \kappa_1$. Thus, we need

$$\kappa \triangleq \max\{\kappa_{1(\ell)}, \kappa_{2(\ell)}\} \geq \begin{cases} \dfrac{e^{-\ell}(2+\ell)^2}{4+\ell-e^{-\ell}(2+\ell)^2}, & \text{if } \ell \leq 1.1462 \\ \dfrac{(2-e^{-\ell}(2+\ell))^2}{e^{-\ell}(4+\ell-e^{-\ell}(2+\ell)^2)} & \text{if } \ell > 1.1462 \end{cases} \tag{7.19}$$

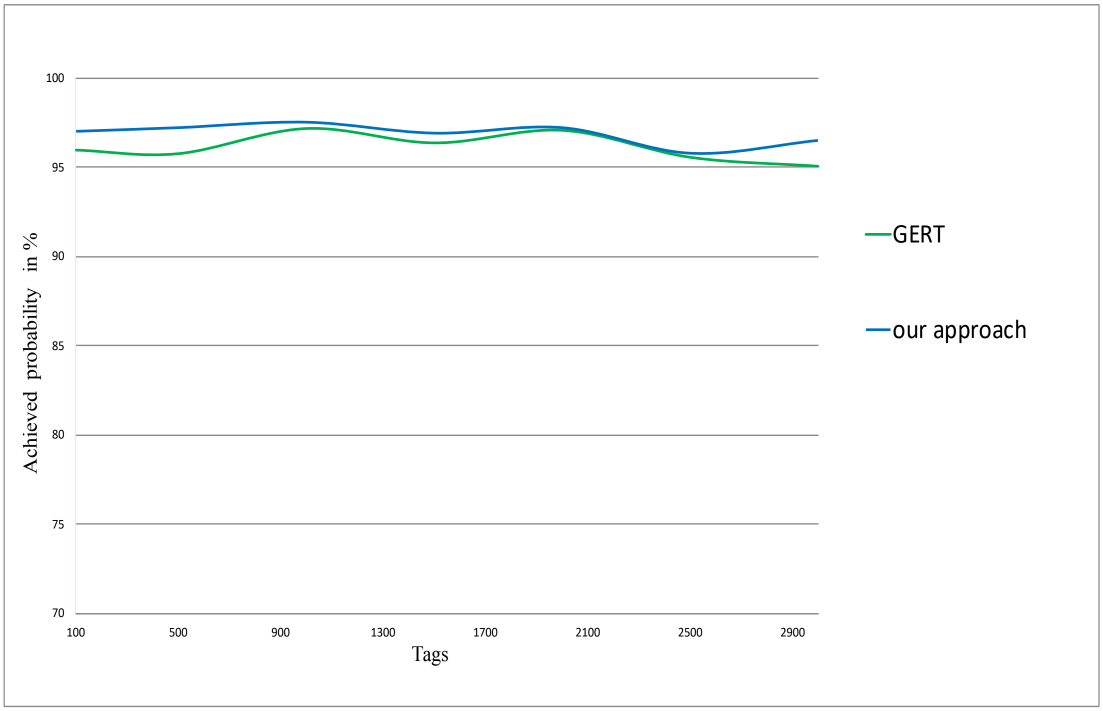


Figure 7.10. Accuracy achieved for targeted accuracy $\alpha = 0.95$, and $\beta = 0.05$

The quality of the estimation depends on ϵ. While $C_{(e-0)} \longrightarrow N(\mu_{(e-0)}, \sigma^2_{e-0})$, where $N(\mu, \sigma^2)$ represents a normal distribution with mean μ, and variance σ^2 and the frame size is large enough to satisfy Equation (7.16) and its extensions below. Let l and u be the allow lower and upper bound values of $\dfrac{C_{j(e-))} - \mu_{(e-0)}}{\sigma_{e-0}}$ that are acceptable to us. Then,

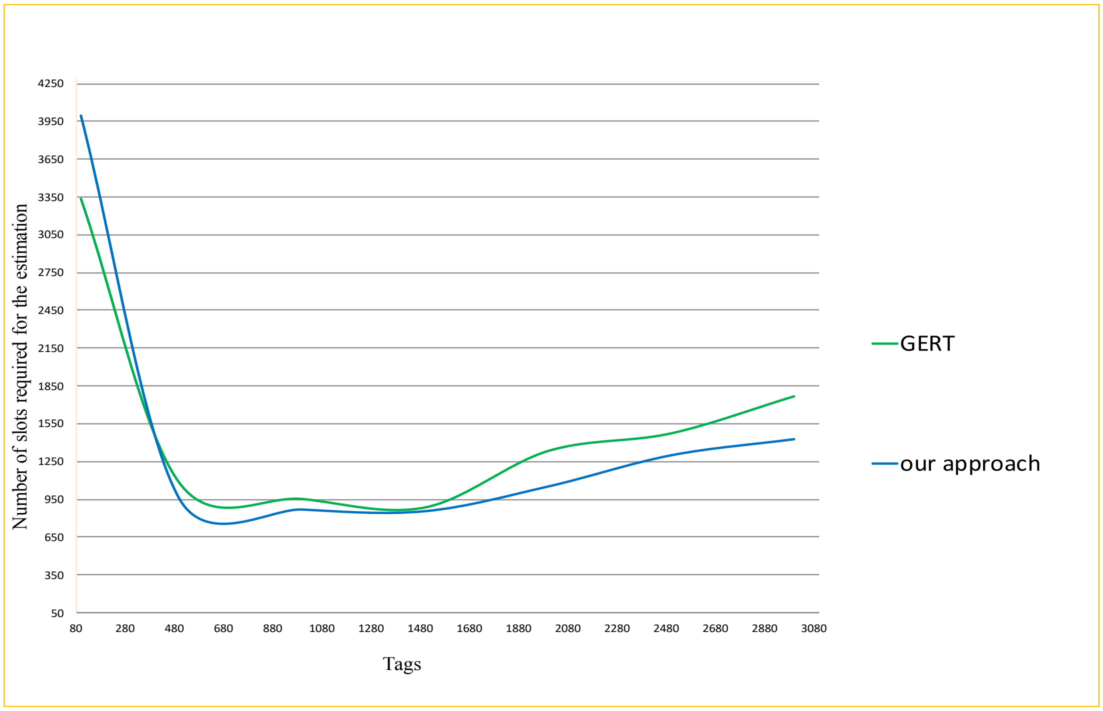


Figure 7.11. Cost incurred for $\alpha = 0.95$, and $\beta = 0.05$

$$\mathbf{Pr}\left[\ell \le \frac{C_{(e-0)} - \mu_{(e-0)}}{\sigma_{(e-0)}} \le u\right] - \mathbf{Pr}[l \le \theta \le u] \le \epsilon \tag{7.20}$$

Proceeding as in Hasan *et al.* [48] it can be shown that the maximum value of ϵ is $1-\alpha$, with $\kappa = \epsilon^2 \max f = (1-\alpha)^2 f$. Or $f = \frac{\kappa}{(1-\alpha^2)}$ given by Equation (7.19). We simulated the performance of our approach and that of a similar analysis for GERT. Figure 7.10 and 7.11 show the results for this simulation. The improvement over the $\{0, e\}$ channel is expected as the channel itself has a higher resolution. Notice from the plot that both approaches exceed the α requirements of the given problem.This also implies that both approaches have also expended a higher cost for a better than needed result. In the next section, we will use an experimental approach to close this gap.

7.6 Phase 2 - Cardinality Estimation

Several approaches have been proposed for cardinality estimation [48, 60, 72, 89, 131, 133] particularly those found in Phase 2 (see Algorithm in Figure 7.1). Many of these approaches used analytical results, for example, to approximate the distribution of estimator values. This approximation reduces the efficiency of the approach. In this section we outline a broad methodology to design an algorithm based on experimental observation. This methodology has two parts that correspond to the two phases in Figure 7.1. The goal here is to design Phase 2, in particular the determination of the frame size f and number of iteration n, given the initial estimate T_0. However, the distribution of T_0 obtained from Phase 1 is critical for the study.

Let $\boldsymbol{A}$ be any Phase 1 algorithm; $\boldsymbol{A}$ could be WILLARD, WILLARD+, WILLARD∗, WILLARD+∗ or any other algorithm (for example Nakano *et al.* [81]). The first task is to run $\boldsymbol{A}$ on different values of T (actual tag population sizes) to record the estimate generated by $\boldsymbol{A}$. To obtain statistically significant results, we need to run each value of T multiple times, say X times. Suppose for integer $1 \leq i \leq X$, algorithm $\boldsymbol{A}$ with input T produces an estimate T_0 when run for the i^{th} time. We will say that $T_0 = \boldsymbol{A}(T, i)$.

Let $\Im = \{T : T_{min} \leq T \leq T_{\max}\}$ be the set of tag population sizes on which $\boldsymbol{A}$ is run. For any value of T_0 within the accepted range, let $\mathbb{U} = (T_0)$ be a multiset of all values of T that produce estimate T_0. That is, $\mathbb{U}(T_0) = \{T : \exists i \ \boldsymbol{A}(T, i) = T_0\}$. This multiset $\mathbb{U}(T_0)$ gives the distribution of T values that appears to the Phase 2 of the cardinality estimation algorithm as T_0. We now explain how this $\mathbb{U}(T_0)$ can be used to determine f and n.

For a given estimator E, assurance probability α and relative error β, let $\boldsymbol{B}$ denote an algorithm for Phase 2 (see Figure 7.1).

Observe that when an actual cardinality estimation algorithm is run, it can observe the value of T_0, but the algorithm runs with T tags. Our simulation likewise uses T_0 as a parameter, but T only to simulate the actions of the tags. The value of T itself is not used in any other way. For a given Phase 1 estimate T_0, that is based on the value of T (actual number of tags), desired accuracy bounds α, β, frame size f, number of iterations n, and estimator E, let $\boldsymbol{B}(T_0, T, A, n, f) = T_{est}$ be the final estimate. We now proceed as follows to determined an "optimal" f and n, given the observable T_0.

Initially let $f(0) = T_0$ and let $n = 1$. We run $\boldsymbol{B}(T_0, T, A, 1, f(0))$ for each $T \in \mathbb{U}(T_0)$. Recall that $\mathbb{U}(T_0)$ is a multiset, so the same value of T may be simulated multiple times. Let $\mathbb{C} = \{B(T_0, T, 1, f(0)) : T \in \mathbb{U}(T_0)\}$; this is also a multiset. Observed that the error due to $\boldsymbol{B}(T_0, T, A, 1, f(0)) = T_{est}$ is $\dfrac{|T_{est} - T|}{T}$ which we denote by $Err(T_0, T, 1, f)$. Let $\mathbb{C}_0 = \{B(T_0, T, 1, f(0)) \in \mathbb{C} : Err(T_0, T, 1, f(0)) \leq \beta$ and let $\alpha_0 = \dfrac{\mathbb{C}_0}{\mathbb{C}}$.

If $\alpha_0 > \alpha$, then we are doing more than needed and $f(0)$ can be reduced. On the other hand, if $\alpha_0 < \alpha$, then we need to increase $f(0)$. We will select $f(1) = 2f(0)$ or $\frac{f(0)}{2}$, as needed, and repeat the process in binary search type manner until in some y iteration $\alpha_y = \alpha$ and $f(y)$ is the desired value of f. In some cases, particularly for stringent α, β requirements, it may not be possible to get a suitable f. If $f(y)$ exceeds an upper limit f_{max}, then we repeat the process with $n = 2$. The simulation can also determine the cost of each choice of f. We have conducted extensive simulations with the following possible values for the accuracy assurance parameters.

- Algorithm $\boldsymbol{A}$ could be Willard and Willard$*$

- Estimator E is assumed to be $f_e - f_0$.

- (α, β) can be $(85\%, 10\%)$ and $(95\%, 5\%)$.

- $T_{min} = 1$,$T_{max} = 10,000$

- X, the maximum number of trials for each value of $T_{min} \leq T \leq T\max$ is 100,000. This large number was used so that each of the buckets $\mathbb{U}(T_0)$ has a sufficiently large number of cases.

In the process we have extended a well known algorithm Willard, Willard$+$, Willard$*$, and Willard$+*$ that provide greater accuracy at added cost. Our work on Cardinality estimation currently focuses on the $f_e - f_0$ estimator (based on initial evidence that this estimator may outperform others). Figure 7.12 shows the cost for $\alpha = 95\%$, $\beta = 5\%$ case for GERT [48] and our adaptation of GERT for the $\{0, 1, e\}$ channel. This cost is higher than the experimental approach that we outlined in this section. Observed that even though Willard∗ is more inexpensive then Willard (see Figure 7.12) this cost is well worth its accuracy, as seen from the cost of Figure 7.12

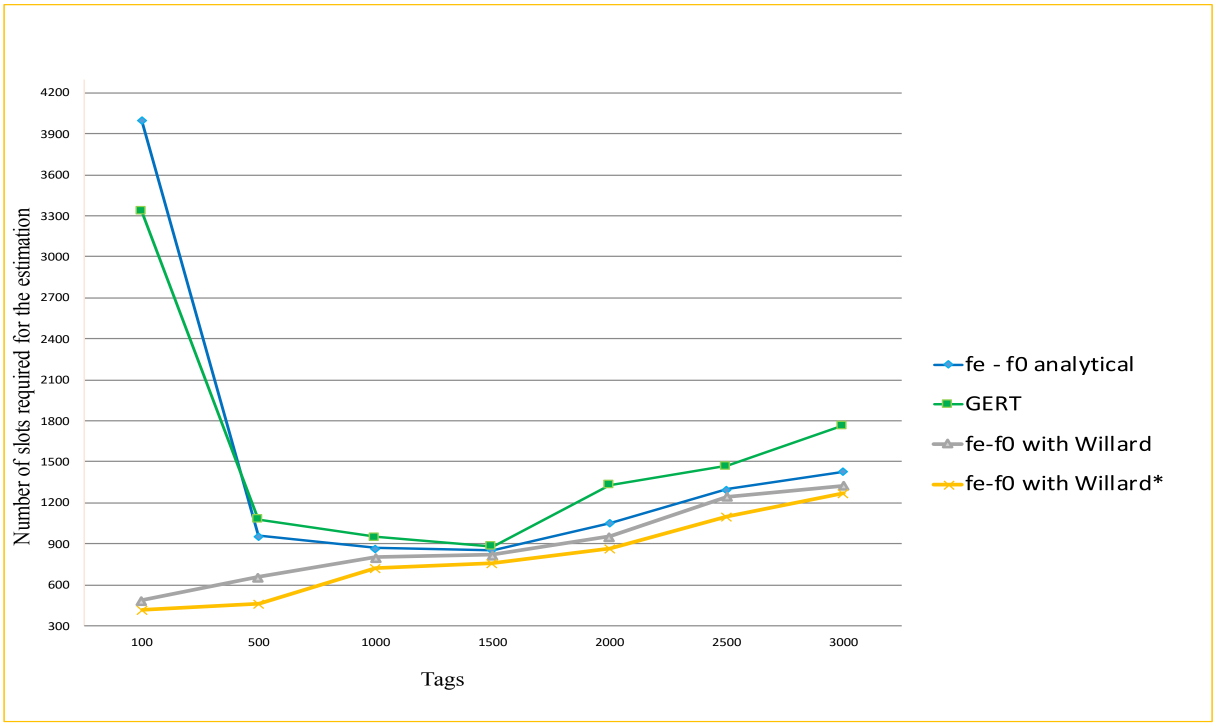


Figure 7.12. Cost achieved by GERT,$f_e - f_0$ (analytical approach), and experimental $f_e - f_0$ for $\alpha = 0.95$, and $\beta = 0.05$

The Phase 1 algorithm gets a crude estimate that guides the Phase 2 algorithm. We realized that we can use an algorithm from our own template with limited, but low-cost performance for Phase 1, which may provide more accurate algorithms. We believe that the experimental methodology lays the foundation for additional work on cardinality estimation, including the effect of Phase 1 algorithms, new estimators, and hybrid methods for different ranges of T.

CHAPTER 8
SUMMARY AND FUTURE WORK

The objective of this dissertation is a study of distributed RFID-based algorithms for Grouping Proof and Cardinality Estimation. We have made several key contributions to the fields of RFID protocols.

For Grouping Proof Protocols, security and efficiency are important goals in an RFID settings. An emphasis of this research was the balance among security, privacy, and usefulness. In the truly practical, user-driven development of RFID applications, we designed three grouping-proof protocols to minimize loss of security, and privacy. In each framework we strove to achieve efficiency tailored to the degree of security required in an RFID system.

One avenue for further study would be to research and design protocols specific to the scope of particular applications. Furthermore, quantifying the type of security specific to the scope of particular applications, we will aim to design grouping-proof protocols that achieve efficiency tailored to the degree of security required in an RFID application. Another direction would be the practical implementation of our three grouping-proof protocols, looking more into the details of the time and quality of random number generator used in our protocols, the time cost (within EPC standard) on passive tags, and the circuit cost.

For Cardinality Estimation we developed a framework for using simulated data to generate an “optimal” estimation algorithm in which we balance the cost and accuracy constraints of the problem. The findings suggest that such a method shows considerable improvement over other proposed analytical approaches that seek to model the problem (albeit inaccurately) and that often exceed the requirements of the problem (at potentially added cost).

In the process we have extended a well known algorithm Willard, for simple cardinality estimation Willard+, Willard∗, and Willard+∗ algorithms that provide greater accuracy at added cost. Our work on cardinality estimation currently focuses on the $f_e - f_0$ estimator (based on initial evidence that this estimator may outperform others). It would be interesting to extend this approach to other estimators (for example, other linear combinations of f_0, f_e).

The Phase 1 algorithm gets a crude estimate that guides the Phase 2 algorithm. Would it be beneficial to use an algorithm from our own template with limited, but low-cost performance for phase 1. This may indicate a pipeline of progressively more accurate algorithms.

How would the cost of the algorithm be affected if f was constrained to be 2^x (a power of 2)? This will considerably reduce the hardware complexity of the tag (specifically, its random number generator).

Can our cardinality estimation algorithm itself be used to simulate a higher resolution $\{0, 1, ..., k-1, e\}$ channel, where tag population up to $k-1$ can be estimated accurately?

REFERENCES

[1] S. Abughazalah, K. Markantonakis, and K. Mayes (2016), “Two Rounds RFID Grouping-Proof Protocol,” Proc. IEEE Intl. Conf. RFID, 14 pages.

[2] S. Amendola, R. Lodato, S. Manzari, C. Occhiuzzi, G. Marrocco (2014),“RFID Technology for IoT-based personal healthcare in smart spaces,” IEEE Internet Things J. 1 (2),144-152.

[3] A. Araujo and E. Gin’e (1980),“The central limit theorem for real and Banach valued random variables” Wiley New York, vol. 431.

[4] Laura Arjona, Hugo Landaluce, Asier Perallos, and Enrique Onieva (2017),“Scalable RFID Tag Estimator With Enhanced Accuracy and Low Estimation Time”, IEEE Signal Processing Letters, Vol. 24, NO. 7.

[5] Giovanni Luca Amicucci, Fabio Fiamingo, (2017), “Usage of RFID in safety applications”, IEEE, 6 pages.

[6] G. Avoine (2005),“Adversarial Model for Radio Frequency Identification,” Cryptology ePrint Archive, Report 2005/049, 14 pages. – http://eprint.iacr.org/2005/049

[7] G. Avoine, X. Carpent and J. Hernandez-Castro (2016), “Pitfalls in Ultralightweight Authentication Protocol Designs,” IEEE Trans. Mobile Computing, vol. 15, no. 9, pp. 2317-2332.

[8] G. Avoine, I. Coisel, and T. Martin (2014), “Untraceability Model for RFID,” IEEE Trans. Mobile Computing, vol. 13, no. 10, pp. 2397-2405.

[9] S. Barmpounakis, A. Kaloxylos, A. Groumas, L. Katsikas, V. Sarris, K. Dimtsa, F. Fournier, E. Antoniou, N. Alonistioti, S. Wolfert (2015),“Management and control applications in Agriculture domain via a Future Internet Business-to-Business platform”, Inf. Process. Agric. 2 (1) 51-63.

[10] B. M. Brown *et al.*, (1971), “Martingale central limit theorems” The Annals of Mathematical Statistics, vol. 42, no. 1, pp. 59?66.

[11] L. Bolotnyy and G. Robins (2006), “Generalized ’yoking-proofs’ for a group of RFID tags”, MobiQuitous, pp. 1-4.

[12] Bonuccelli, M.A., Lonetti, F., Martelli, F. (2009),“Exploiting Signal Strength Detection and Collision Cancellation for Tag Identification in RFID Systems.” IEEE Symposium on Computers and Communications, Computers and Communications p.500.

[13] Borovca-Gajic, Renata, Idreos, Stratos, Ailamaki, Anastasia, (2018),“Smooth Scan: robust access path selection without cardinality estimation,” VLDB Journal International Journal on Very Large Data Bases; Aug2018, Vol. 27 Issue 4, p521-545, 25p.

[14] B. M. Brown et al. (1971),“Martingale central limit theorems,” The Annals of Mathematical Statistics, vol. 42, no. 1, pp. 59-66.

[15] M. Burmester, B. D. Medeiros, and R. Motta (2008), “Provably secure grouping-proofs for RFID tags,” in Proc. Int. Fed. Inf. Process., pp. 176-190.

[16] M. Burmester and J. Munilla (2016) , “An Anonymous RFID Grouping-Proof with Missing Tag Identification,” Proc. 2016 IEEE Intl. Conf. on RFID (RFID), 7 pages.

[17] Caidong Gu (2018), “Fast Discrepancy Identification for RFID-Enabled IoT Networks”, IEEE Access.

[18] Canetti R. (2001) “Universally composable security: a new paradigm for cryptographic protocols”, Proceedings of 42nd IEEE Symposium on Foundations of Computer Science, Las Vegas; 136-145.

[19] Canetti R. (2007), “Obtaining universally composable security: towards the bare bones of trust.”, Proceedings of 13th International Conference on the Theory and Application of Cryptology and Information security, Kuching, Malaysia, 88-112.

[20] Li Qing Cao, Yunhe Feng, Zheng Lu , Hairong Qi, Leon Tolbert, Lipeng Wan, Zhibo Wang, Wenjun Zhou (2017),“Approximate Cardinality Estimation (ACE) in large-scale Internet of Things deployments”,Ad Hoc Networks 66, 52-63.

[21] Castiglione, A., Palmieri, F., Fiore, U., Castiglione, A., De Santis (2015), “A. Modeling energy-efficient secure communications in multi-mode wireless mobile devices.” J. Comput. Syst. Sci. , 81, 1464-1478.

[22] Chen H., Ma G., Wang Z.Wang, Q. Yu, (2018) , “MAC: Missing Tag Iceberg Queries for Multi-Category RFID Systems”, IEEE Transactions on Vehicular, 67(10):9947-9958.

[23] J. Chen, A. Miyaj, H. Sato and C. Su (2015), “Improved Lightweight Pseudo-Random Number Generators for the Low-Cost RFID Tags,” Proc. 2015 IEEE Trustcom/BigDataSE/ISPA, pp. 17-24.

[24] S. Cheng, V. Varadharajan, Y. Mu, and W. Susilo (2017), “An Efficient and Provably Secure RFID Grouping Proof Protocol,” (2017) Proc. Australasian Computer Science Week Multiconference (ACSW ’17), Article 71, 7 pages.

[25] V. Cherneva, J. Trahan (2018), “Serial-Dependency Grouping-Proof Protocol for RFID EPC C1G2 Tags”, IEEE Green Energy and Smart Systems Conference (IGESSC).

[26] S. Cheng, V. Varadharajan, Y. Mu, and W. Susilo (2017), “An Efficient and Provably Secure RFID Grouping Proof Protocol,” Proc. Australasian Computer Science Week Multiconference (ACSW ’17), Article 71, 7p.

[27] H.-Y. Chien and S.-B. Liu (2009), “Tree-based RFID yoking proof,” Intl. Conf. networks security, wireless communications and trusted computing - NSWCTC, pp. 550-553.

[28] I. Coisel and T. Martin (2013), "Untangling RFID Privacy Models," J. Computer Networks and Communications, vol. 2013, Article ID 710275, 26 pages.

[29] Jihoon Choi and Wonjun Lee (2007),"Comparative Evaluation of Probabilistic and Deterministic Tag Anti-collision Protocols for RFID Networks", IFIP International Federation for Information Processing.

[30] A. Chong, M.J. Liu, J. Luo, O. Keng-Boon (2015),"Predicting RFID adoption in healthcare supply chain from the perspectives of users", Int. J. Prod. Econ. 159 66-75.

[31] A. Chong, F. Chan (2013),"Structural equation modeling for multi-stage analysis on radio frequency identification (RFID) diffusion in the healthcare industry". Expert-Syst.Appl., 8645-8654.

[32] C. Costa, F. Antonucci, F. Pallottino, J. Aguzzi, D. Sarria, and P. Menesatti (2013),"A review on Agri -food Supply Chain Traceability by means of RFID technology", Food Bioprocess Technol., vol. 6, no. 2, pp.353-366

[33] I. A. Currie and M. K. Marina. (2008)"Experimental evaluation of read performance for RFID-based mobile sensor data gathering applications," 7th Int. Conf. Mobile Ubiquitous Multimedia-Cooperation ACM SIGMOBILE, pp. 92?95.

[34] Der-Jiunn Deng, Chun-Cheng Lin, Tzu-Hsun Huang, and Hsu-Chun Yen (2017),"On Number of Tags Estimation in RFID Systems", IEEE System journal, Vol. 11, no. 3.

[35] Der-Jiunn Deng, Hsuan-Wei Tsao (2011), "Optimal Dynamic Framed Slotted ALOHA Based Anti-collision Algorithm for RFID Systems", Wireless Pers Commun.

[36] R. Doss , W. Zhou , S. Yu (2013),"Secure RFID tag ownership transfer based on quadratic residues", IEEE Trans. Inf. Forensics Secur. 8 (2) 390-401.

[37] R. Doss , W. Zhou , S. Sundaresan , S. Yu , L. Gao (2012),"A minimum disclosure approach to authentication and privacy in RFID systems", Comput. Netw. 3401-3416.

[38] D. Dressen, (2011) "Large memory RFID system solutions," ATMEL, pp. 48-49.

[39] Duc D N, Kim J, Kim K. (2010),"Scalable grouping-proof protocol for RFID tags." Symposium on Cryptography and Information Security (SCIS), 1-6.

[40] EPC-Global (2015), "Radio-Frequency Identity Protocols, Generation-2.V2. UHF RFID." Tech. Rep.

[41] Ivan Farris, Sara Pizzi, Massimo Merenda, Antonella Molinaro, Riccardo Carotenuto, and Antonio Iera, (2017),"A Framework for Full Integration of Smart UHF RFID Tags into the Internet of Things", IEEE Network, p.66-74.

[42] Arundhoti Ferdous, Nanda Gopal Jeevarathnam and Ismail Uysal (2017), "Comparative Analysis of Tag Estimation Algorithms on RFID EPC Gen-2 Performance".

[43] Fisher, J. A., Monahan, T. (2008),"Tracking the social dimensions of RFID systems in hospitals." International Journal of Medical Informatics, 77, 176-183.

[44] L. Gong, R. Needham, and R. Yahalom (1990), "Reasoning about Belief in Cryptographic Protocols," Proc. 1990 IEEE Symp. Research in Security and Privacy, pp. 234-248.

[45] W. Gong, K. Liu, X. Miao, and H. Liu (2014),"Arbitrarily accurate approximation scheme for large-scale RFID cardinality estimation," in Proc. IEEE INFOCOM, pp. 477-485.

[46] Wei Gong, Ivan Stojmenovic, Amiya Nayak, Kebin Liu, and Haoxiang Liu (2016),"Fast and Scalable Counterfeits Estimation for Large-Scale RFID Systems", IEEE/ACM Transaction on Networking, Vol. 24, no. 2, April 2016.

[47] Hao Han, Bo Sheng, Chiu C. Tan, Qun Li, Weizhen Mao, Sanglu Lu (2010) "Counting RFID Tags Efficiently and Anonymously", IEEE INFOCOM.

[48] Hasan Md M., Wei Sh., Vaidyanathan R. (2018)"Estimation of RFID Tag Population Size by Gaussian Estimator", 2018 IEEE International Conference on Communications.

[49] S. Higginbotham (2015),"Grocery shopping might be less painful with this smart cart."

[50] Hou Y., Zheng Y.(2018), "PHY-Tree: Physical Layer Tree-Based RFID Identification", IEEE/ACM Trans. Networking, IEEE/ACM Transactions on. 26(2):711-723.

[51] C.-T. Hsi, Y.-H. Lien, J.-H. Chiu, and H. K.-C. Chang (2015), "Solving Scalability Problems on Secure RFID Grouping-Proof Protocol," Wirel. Pers. Commun., vol. 84, no. 2, pp. 1069-1088.

[52] IDTechEx, RFID Forecasts, Players and Opportunities 2017-2027 (2017) Read more at: https://www.idtechex.com/research/reports/rfid-forecasts-players-and-opportunities-2017-2027-000546.asp

[53] Jihong Yu, Lin Chen, Rongrong Zhang, Kehao Wang (2016), "From Static to Dynamic Tag Population Estimation: An Extended Kalman Filter Perspective" , IEEE Transactions on Communications 64(11):p.4706-4719

[54] R. Jedermann, M. Nicometo, I. Uysal, and W. Lang (2014),"Reducing food losses by intelligent food logistics," Philos. Trans. R. Soc. Math. Phys. Eng. Sci., vol. 372.

[55] A. Juels (2004), " 'Yoking-proofs' for RFID tags," Intl. Workshop on Pervasive Computing and Communication Security - PerSec 2004, pp. 138-143.

[56] Kalache, M.A. Fergani, L. (2014),"Performances comparison of RFID anti-collision algorithms", ICMCS 2014 International Conference, 808-813.

[57] A. Kaloxylos, J. Wolfert, T. Verwaart, C.M. Terol, C. Brewster, R. Robbemond, H. Sundmaker (2013),"The use of Future Internet Technologies in the agriculture and Food sectors: integrating the supply chain", Procedia Technol. 8 51-60.

[58] Dheeraj K. Klair, Kwan-Wu Chin, and Raad Raad (2010),“A Survey and Tutorial of RFID Anti-Collision Protocols”, IEEE Communication Surveys and Tutorials , Vol. 12, no. 3.

[59] Murali Kodialam, Thyaga Nandagopal (2006),“Fast and Reliable Estimation Schemes in RFID Systems”, MobiCom.

[60] Murali Kodialam, Thyaga Nandagopal , and Wing Cheong Lau (2007),“Anonymous Tracking using RFID tags” ,publication in the IEEE INFOCOM.

[61] R. Kumar, K. Gopalakrishna, and K. Ramesha (2013),“Intelligent Shopping Cart,” Int. J. Eng. Sci. Innov. Technol., vol. 2, no. 4, pp. 499-507.

[62] Zhonghua Li, Chunhui He, Hong-Zhou Tan (2011),“Survey of the advances in reader anti-collision algorithms for RFID systems”, (CCDC), 3771-3776 .

[63] Xuefei Leng, Yuan-Hung Lien, Konstantinos Markantonakis (2010),“ An RFID grouping proof protocol exploiting anti-collision algorithm for subgroup dividing”, International Journal of Security and Networks,vol.5.

[64] T. Li, S. Wu, S. Chen, and M. Yang (2010),“Energy efficient algorithms for the RFID estimation problem”, in Proc. IEEE INFOCOM, pp. 1-9.

[65] C.-C. Lin, Y.-C. Lai, J. Tygar, C.-K. Yang, and C.-L. Chiang (2007), “Coexistence proof using chain of timestamps for multiple RFID tags,” Proc. Adv. Web Netw. Technol. Inf. Manage., pp. 634-643.

[66] Liu, Xiulong, Cao, Jiannong, Yang, Yanni, Jiang, Shan (2018)“CPS-Based Smart Warehouse for Industry 4.0: A Survey of the Underlying Technologies.”, Computers (2073-431X);Vol. 7 Issue 1.

[67] Liu, Xiulong, Guo, Kaimin, Liu, Zijuan et al.(2018), “Fast and Accurate Missing Tag Detection for Multi-category RFID Systems”, SMARTIOT Smart Internet of Things (SmartIoT), 2018 IEEE International Conference,p135-142.

[68] Xiulong Liu, Keqiu Li, Alex X. Liu, Song Guo, Muhammad Shahzad, Ann L. Wang, and Jie Wu, (2017), ”Multi-Category RFID Estimation”, IEEE/ACM Transaction on networking , vol. 25, no. 1.

[69] Xiulong Liu, Bin Xiao, Keqiu Li, Alex X. Liu, Jie Wu, Xin Xie, and Heng Qi (2017),“RFID Estimation With Blocker Tags”, IEEE/ACM Transaction on networking , vol. 25, no. 1.

[70] Liu, X., Xie, X., Zhao, X., Wang, K. et al. (2018), “Fast Identification of Blocked RFID Tags”, IEEE Transactions on Mobile Computing, 17(9):2041-2054.

[71] Lu, Zhejun; Hu, Weidong et al. (2018)“A new Cardinalized Probability Hypothesis Density Filter with Efficient Track Continuity and Extraction”,2018 21st International Conference on Information Fusion (FUSION) Information Fusion:211-218.

[72] Jiangchuan Liu, Kebin Liu, and Yunhao Liu, (2017),“Toward More Rigorous and Practical Cardinality Estimation for Large-Scale RFID Systems”, IEEE/ACM , Transaction on networking , vol. 25, no. 3.

[73] N.-W. Lo and K.-H. Yeh (2010), “Anonymous coexistence proofs for RFID tags,” J. Inf. Sci. Eng., vol. 26, pp. 1213-1230.

[74] K. Mandal, X. Fan, and G. Gong (2016),“Design and Implementation of Warbler Family of Lightweight Pseudorandom Number Generators for Smart Devices,” ACM Trans. Embed. Comput. Syst., vol. 15, no. 1, Article 1, 28p.

[75] H. Martn, E. S. Milln, L. Entrena, J. C. H. Castro, and P. P. Lpez (2011), “AKARI-X: A Pseudorandom Number Generator for Secure Lightweight Systems,” Proc. 2011 IEEE 17th Intl. On-Line Testing Symp., pp. 228-233.

[76] H. Martn, E. S. Milln, P. Peris-Lopez, and J. E. Tapiador (2013),“Efficient ASIC Implementation and Analysis of Two EPC-C1G2 RFID Authentication Protocols,” IEEE Sensors Journal, vol. 13, no. 10, pp. 3537-3547.

[77] J. Meli-Segu, J. Garcia-Alfaro, and J. Herrera-Joancomart (2010),“Analysis and Improvement of a Pseudorandom Number Generator for EPC Gen2 Tags” Proc. Intl. Conf. Financial Cryptography and Data Security (Lect. Notes Comput. Sci. no. 6054), pp. 34-46.

[78] J. Meli-Segu, J. Garcia-Alfaro, and J. Herrera-Joancomart (2013),“J3Gen: A PRNG for Low-Cost Passive RFID,” Sensors, vol. 13, no. 3, pp. 3816-3830.

[79] D. Moriyama (2014), “A Provably Secure Offline RFID Yoking-Proof Protocol with Anonymity,” Proc. Intl. Workshop on Lightweight Cryptography for Security and Privacy (LightSec 2014), pp. 155-167.

[80] U. Mujahid, M. Najam-ul-islam, A. Sharif, T. Khan, A. Qavi, and Bilal (2014), “A Novel Lightweight Pseudorandom Number Generator for Passive RFID Systems” Proc. 17th IEEE Inta.l Multi Topic Conf., pp. 149-154.

[81] Koji Nakano, Stephan Olariu (2002),“Uniform Leader Election Protocols for Radio Networks”, IEEE Transaction on Parallel and Distributed systems , vol. 13, no. 5.

[82] Chuyen T. Nguyen, Tuyen T. Hoang, and Vu X. Phan, (2017),“A Simple Method for Anonymous Tag Cardinality Estimation in RFID Systems with False Detection”, 2017 4th NAFOSTED Conference on Information and Computer Science

[83] R. Nukala, K. Panduru, A. Shields, D. Riordan, P. Doody, J. Walsh (2016),“Internet of Things: A Review from Farm to Fork”, 27th Irish Signals and Systems Conference, pp. 1-6.

[84] C. Perera, A. Zaslavsky, P. Christen, D. Georgakopoulos (2014),“Context aware computing for the IoT: a survey”,IEEE Commun. Surv. Tutorials 16 (1) 414-454.

[85] Lopez Pedro Peris, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda (2007), “Solving the simultaneous scanning problem anonymously: Clumping proofs for RFID tags,” Proc. 3rd Workshop Security Privacy Trust Pervasive Ubiq. Comput., p. 55-60.

[86] Lopez Pedro Peris, Hernandez-Castro Julio C, Li Tieyan, (2013)“Security and Trends in Wireless Identification and Sensing Platform Tags - Advancements in RFID Security ” IGI Global (book).

[87] Lopez Pedro Peris, A. Orfila, J. C. Hernandez-Castro, and J. C. A. van der Lubbe (2011), “Flaws on RFID grouping-proofs. Guidelines for future sound protocols,” J. Netw. Comput. Appl., vol. 34, no. 3, pp. 833-845.

[88] S. Piramuthu (2006), “On existence proofs for multiple RFID tags,” Proc. IEEE Int. Conf. Pervasive Serv., Workshop Secur., Privacy Trust Pervasive Ubiq. Comput., pp. 317-320.

[89] Chen Qian, H. Ngan, Y. Liu, and L. M. Ni (2011),“Cardinality Estimation for Large-Scale RFID Systems”, EEE Transaction on Parallel and Distributed systems , vol. 22, no. 9.

[90] Y. Qiao, S. Chen, and T. Li (2013),“RFID as an Infrastructure” New York, NY, USA: Springer.

[91] von V. Raab (2011),“Assessment of novel temperature monitoring systems for improving cold chain management in meat supply chain,” University of Bonn.

[92] L. Ramundo, M. Taisch, S. Terzi (2016),“State of the Art of Technology in the Food Sector Value Chain Towards the IoT”, IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a Better Tomorrow, pp. 1-6.

[93] S. Rostampour, N. Bagheri, M. Hosseinzadeh et al. (2018) “A Scalable and Lightweight Grouping Proof Protocol for Internet of Things Applications,” Journal of Super computing, vol. 74, no. 1. pp. 71-86.

[94] S. Rostampour, N. Bagheri, M. Hosseinzadeh et al. (2016) “An authenticated encryption based grouping proof protocol for RFID systems,” Security and Communication network, vol. 9, no.18, pp.581-5590.

[95] Rotter, Pawel (2008) “A framework for assessing RFID system security and privacy risks”, IEEE Pervasive Computing, Vol. 7, Issue: 2 , p: 70-77.

[96] Saravanakumar K, Deepa K, Senthil Kumar, (2017),“Study on possible application of RFID system in different real-time environments”, 2017 International Conference on circuits Power and Computing Technologies, 7pages.

[97] Muhammad Shahzad and Alex X. Liu (2015),“Fast and Accurate Estimation of RFID Tags”, IEEE/ACM Transaction on Networking, vol. 23, no. 1.

[98] Z. Shi, X. Zhang, and Y. Wang (2017) "A Lightweight RFID Grouping-Proof Protocol Based on Parallel Mode and DHCP Mechanism," Information, vol.88, no. 3, Article no.85, 13 pages.

[99] D. Z. Sun and Yi Mu (2018) "Security of Grouping-Proof Authentication Protocol for Distributed RFID Systems," IEEE Wireless Communication Letters, vol. 7, no. 2, pp. 254-257.

[100] Y. S. Su and C. H. Wang (2015), "Design and Analysis of Unequal Missing Protection for the Grouping of RFID Tags," IEEE Trans. Communications, vol. 63, no. 11, pp. 4474-4489.

[101] D. Z. Sun and G. Q. Xu (2017), "One-Round Provably Secure Yoking-Proof for RFID Applications," Proc. 2017 IEEE Trustcom/BigDataSE/ICESS, pp. 315-322.

[102] S. Sundaresan, R. Doss, and W. Zhou (2015), "Zero Knowledge Grouping Proof Protocol for RFID EPC C1G2 Tags," IEEE Trans. Computers, vol. 64, no. 10, pp. 2994-3008.

[103] S. Sundaresan, R. Doss , W. Zhou (2012),"A secure search protocol based on quadratic residues for epc class-1 gen-2 uhf rfid tags", 23rd International Symposium on Personal Indoor and Mobile Radio Communications.

[104] S. Sundaresan, R. Doss, S. Piramuthu , W. Zhou (2014),"A robust grouping proof protocol for RFID EPC C1G2 tags," IEEE Trans. Inf. Forensics Secur.

[105] S. Sundaresan, R. Doss, W. Zhou (2012),"A Serverless Ultra-Lightweight Secure Search Protocol for EPC Class-1 Gen-2 UHF RFID Tags", International Conference on Computer and Information Science (ICCIS), pp. 580-585 .

[106] S. Sundaresan, R. Doss, W. Zhou, S. Piramuthu, (2015)"Secure ownership transfer for multi-Tag multi-owner passive RFID environment with individual-Owner-Privacy", Comp. Commun. 112-124 .

[107] S. Sundaresan, R. Doss, S. Piramuthu , W. Zhou (2015),"Secure tag search in rfid systems using mobile readers", IEEE Trans. Dependable Secure Comput.230-242 .

[108] S. Sundaresan, R. Doss, S. Piramuthu, and W. Zhou (2017), "A Secure Search Protocol for Low Cost Passive RFID Tags," Computer Networks, vol. 122, pp. 70-82.

[109] Sundaresan S, Doss R, Piramuthu S, Zhou W. (2014),"A robust grouping proof protocol for RFID EPC C1G2 tags."IEEE Transactions on Information Forensics and Security 2014; 9(6):961-975.

[110] Sun HM, Ting WC, Chang SY. (2009),"Offline simultaneous grouping proof for RFID tags."2nd International Conference on Computer Science and Its Applications. Jeju Island, Korea, 1-6.

[111] M. Thibaud, H. Chi, W. Zhou, and S. Piramuthu (2018), "Internet of Things (IoT) in high-risk Environment, Health and Safety (EHS) industries: A comprehensive review", Decision Support Systems 108, pp. 79-95.

[112] Y. Tian, G. Yang, and Y. Mu (2017), “Privacy-Preserving Yoking Proof with Key Exchange in the Three-Party Setting,” Wireless Personal Commun., vol. 94, pp. 1017-1034.

[113] C.E. Turcu, C.O. Turcu (2013),“Internet of things as key enabler for sustainable healthcare delivery”, Procedia. Soc. Behav. Sci. 73 251-256.

[114] S.F. Tzeng, W.H. Chen, F.Y. Pai,, (2008) “Evaluating the business value of RFID: evidence from five case studies”, Int. J. Prod. Econ. 112 (2) 601-613.

[115] S. Vaudenay (2007),“On Privacy Models for RFID”, Proc. Intl. Conf. Theory and Application of Cryptology and Information Security (ASIACRYPT 2007) (Lect. Notes Comput. Sci. no. 4833), pp. 68-87.

[116] C.N. Verdouw, N. Vucic, H. Sundmaeker, A. Beulens (2014),“Future internet as a driver for virtualization, connectivity and intelligence of Agri-Food supply chain networks”, Int. J. Food Syst. Dyn. 4 (4) 261-272.

[117] C.N. Verdouw, J. Wolfert, A.J. Beulens, A. Rialland (2016) ,“Virtualization of food supply chains with the internet of things”, J. Food Eng. 176 ,128-136.

[118] Harald Vogt (2002),“Efficient Object Identification with Passive RFID Tags”, International Conference on Pervasive Computing.

[119] Dan E. Willard (1884),“Log-logarithmic protocols for resolving ethernet and semaphore conflicts”, Proceedings of the sixteenth annual ACM symposium on Theory of computing, p. 512-521.

[120] W. Zhang, S. Qiz, S. Wang, L. Wu et al. (2018), “A New Scalable Lightweight Grouping Proof Protocol for RFID systems,” Wireless Pers Commun 103, pp.133-143.

[121] Haifeng Wu, Yang Wang, Yu Zeng (2018), “Capture-aware Bayesian RFID tag estimate for large-scale identification”, IEEE/CAA Journal of Automatica Sinica 5(1):119-127

[122] S. Wu, K. Chen, and Y. Zhu (2012), “A secure lightweight RFID binding proof protocol for medication errors and patient safety,” J. Med. Syst., vol. 36, pp. 2743-2749.

[123] Z. Xiaorong, F. Honghui, Z. Hongjin1, and F. Hanyu (2015),“The Design of the Internet of Things Solution for Food Supply Chain,” 5th International Conference on Education, Management, Information and Medicine.

[124] Xin Xie , Xiulong Liu , Keqiu Li , Geyong Min , Weilian Xue (2017),“Fast temporal continuous scanning in RFID systems ”,Computer Communications 106, 46-56.

[125] Lei Xie, Yafeng Yin, Athanasios V. Vasilakos, and Sanglu Lu(2014), “Managing RFID Data: Challenges, Opportunities And Solutions.” IEEE Communications Surveys and Tutorials 16.3.

[126] R. Xu, L. Yang, and S.-H. Yang (2013),“Architecture Design of Internet of Things in Logistics Management for Emergency Response,” pp. 395-402.

[127] M. H. Yang, J. N. Luo, and S. Y. Lu (2015), “A Novel Multilayered RFID Tagged Cargo Integrity Assurance Scheme,” Sensors, vol. 15, no. 10, pp. 27087-27115.

[128] B. Yuan and J. Liu (2016), “A Universally Composable Secure Grouping-Proof Protocol for RFID Tags,” Concurrency and Comput.: Practice and Exper., vol. 28, pp. 1872-1883.

[129] Zhang Youlin, Chen Shigang, Zhou Y., Fang, Yuguang (2018) “Anonymous Temporal-Spatial JointEstimation at Category Level OverMultiple Tag Sets”, IEEE Conference on Computer Communications Computer Communications (INFOCOM),846-854.

[130] Zhang Youlin, Chen Shigang, Zhou You, Odegbile Olufemi (2018), “Missing-Tag Detection with Presence of Unknown Tags”, Communication, and Networking (SECON) Sensing, Communication, and Networking (SECON).

[131] Yuanqing Zheng, Mo Li (2013),“ZOE: Fast Cardinality Estimation for Large-Scale RFID Systems”, 2013 Proceedings IEEE INFOCOM.

[132] Yao Zheng, Xiaomei Wang, Dongyu Yang, Simiao Ding, (2017),“An Efficient RFID Tag Cardinality Estimation Protocol Based on Bit Detection”, 2017 17th IEEE International Conference on Communication Technology.

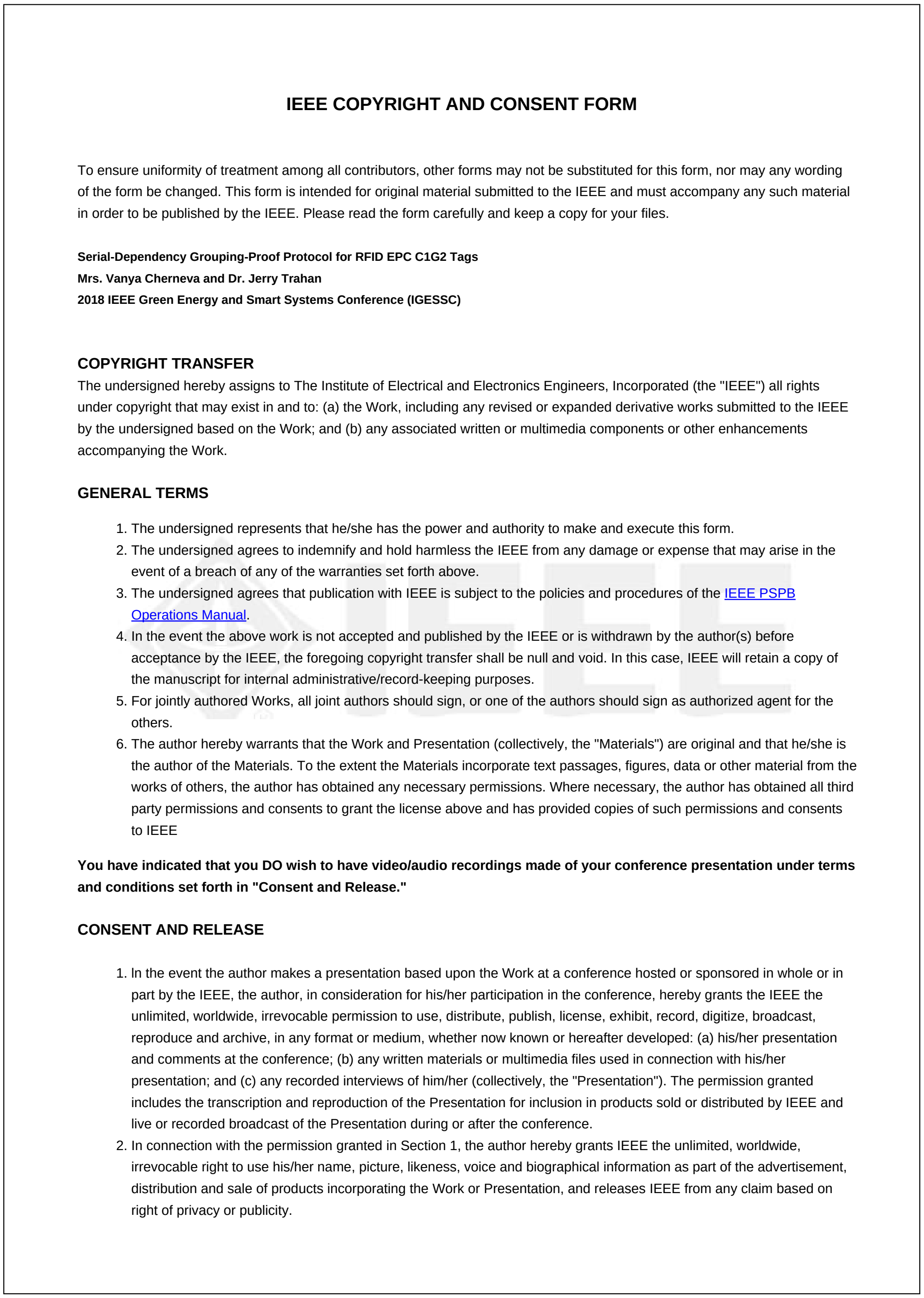
[133] Ziling Zhou, Binbin Chen, and Haifeng Yu (2016),“Understanding RFID Counting Protocols”,IEEE/ACM Transaction on Networking, vol. 24, no. 1.

[134] W. Zhang, S. Qiz, S. Wang, L. Wu et al. (2018), “A New Scalable Lightweight Grouping Proof Protocol for RFID systems,” Wireless Pers Commun 103, pp.133-143

[135] Y. Zheng and M. Li (2012),“PET: Probabilistic estimating tree for large-scale RFID estimation,”IEEE Trans. Mobile Comput., vol. 11, no. 11, pp. 1763-1774.

[136] Z. Zhou, P. Liu, Q. Liu, and G. Wang (2018), “An Anonymous Offline RFID Grouping-Proof Protocol,” Future Internet, vol. 10, no. 1, article 2, 15 pages.

APPENDIX. COPYRIGHT INFORMATION



IEEE COPYRIGHT AND CONSENT FORM

To ensure uniformity of treatment among all contributors, other forms may not be substituted for this form, nor may any wording of the form be changed. This form is intended for original material submitted to the IEEE and must accompany any such material in order to be published by the IEEE. Please read the form carefully and keep a copy for your files.

Serial-Dependency Grouping-Proof Protocol for RFID EPC C1G2 Tags
Mrs. Vanya Cherneva and Dr. Jerry Trahan
2018 IEEE Green Energy and Smart Systems Conference (IGESSC)

COPYRIGHT TRANSFER

The undersigned hereby assigns to The Institute of Electrical and Electronics Engineers, Incorporated (the "IEEE") all rights under copyright that may exist in and to: (a) the Work, including any revised or expanded derivative works submitted to the IEEE by the undersigned based on the Work; and (b) any associated written or multimedia components or other enhancements accompanying the Work.

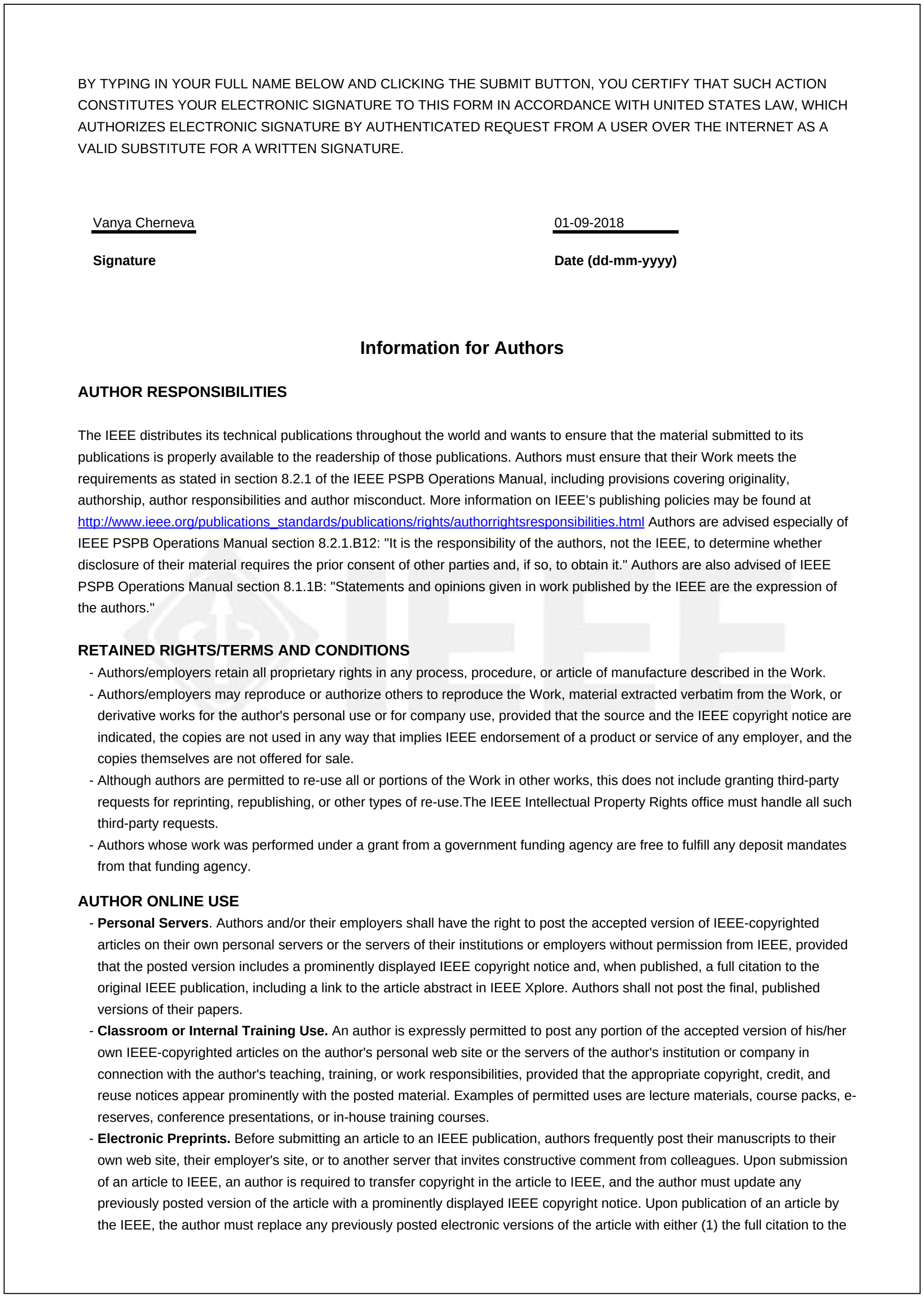
GENERAL TERMS

1. The undersigned represents that he/she has the power and authority to make and execute this form.
2. The undersigned agrees to indemnify and hold harmless the IEEE from any damage or expense that may arise in the event of a breach of any of the warranties set forth above.
3. The undersigned agrees that publication with IEEE is subject to the policies and procedures of the IEEE PSPB Operations Manual.
4. In the event the above work is not accepted and published by the IEEE or is withdrawn by the author(s) before acceptance by the IEEE, the foregoing copyright transfer shall be null and void. In this case, IEEE will retain a copy of the manuscript for internal administrative/record-keeping purposes.
5. For jointly authored Works, all joint authors should sign, or one of the authors should sign as authorized agent for the others.
6. The author hereby warrants that the Work and Presentation (collectively, the "Materials") are original and that he/she is the author of the Materials. To the extent the Materials incorporate text passages, figures, data or other material from the works of others, the author has obtained any necessary permissions. Where necessary, the author has obtained all third party permissions and consents to grant the license above and has provided copies of such permissions and consents to IEEE

You have indicated that you DO wish to have video/audio recordings made of your conference presentation under terms and conditions set forth in "Consent and Release."

CONSENT AND RELEASE

1. In the event the author makes a presentation based upon the Work at a conference hosted or sponsored in whole or in part by the IEEE, the author, in consideration for his/her participation in the conference, hereby grants the IEEE the unlimited, worldwide, irrevocable permission to use, distribute, publish, license, exhibit, record, digitize, broadcast, reproduce and archive, in any format or medium, whether now known or hereafter developed: (a) his/her presentation and comments at the conference; (b) any written materials or multimedia files used in connection with his/her presentation; and (c) any recorded interviews of him/her (collectively, the "Presentation"). The permission granted includes the transcription and reproduction of the Presentation for inclusion in products sold or distributed by IEEE and live or recorded broadcast of the Presentation during or after the conference.
2. In connection with the permission granted in Section 1, the author hereby grants IEEE the unlimited, worldwide, irrevocable right to use his/her name, picture, likeness, voice and biographical information as part of the advertisement, distribution and sale of products incorporating the Work or Presentation, and releases IEEE from any claim based on right of privacy or publicity.



BY TYPING IN YOUR FULL NAME BELOW AND CLICKING THE SUBMIT BUTTON, YOU CERTIFY THAT SUCH ACTION CONSTITUTES YOUR ELECTRONIC SIGNATURE TO THIS FORM IN ACCORDANCE WITH UNITED STATES LAW, WHICH AUTHORIZES ELECTRONIC SIGNATURE BY AUTHENTICATED REQUEST FROM A USER OVER THE INTERNET AS A VALID SUBSTITUTE FOR A WRITTEN SIGNATURE.

Vanya Cherneva	01-09-2018
Signature	**Date (dd-mm-yyyy)**

Information for Authors

AUTHOR RESPONSIBILITIES

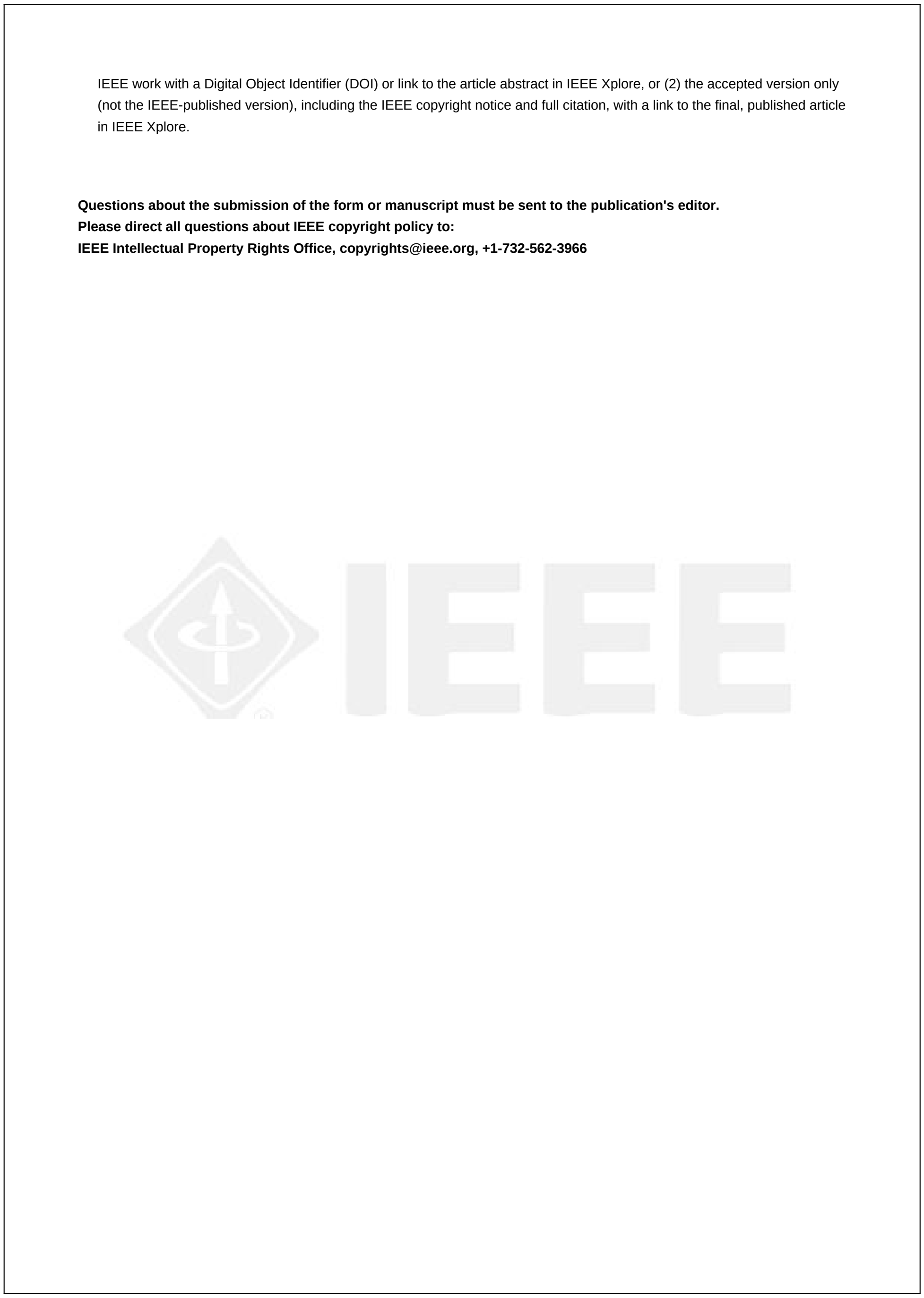
The IEEE distributes its technical publications throughout the world and wants to ensure that the material submitted to its publications is properly available to the readership of those publications. Authors must ensure that their Work meets the requirements as stated in section 8.2.1 of the IEEE PSPB Operations Manual, including provisions covering originality, authorship, author responsibilities and author misconduct. More information on IEEE's publishing policies may be found at http://www.ieee.org/publications_standards/publications/rights/authorrightsresponsibilities.html Authors are advised especially of IEEE PSPB Operations Manual section 8.2.1.B12: "It is the responsibility of the authors, not the IEEE, to determine whether disclosure of their material requires the prior consent of other parties and, if so, to obtain it." Authors are also advised of IEEE PSPB Operations Manual section 8.1.1B: "Statements and opinions given in work published by the IEEE are the expression of the authors."

RETAINED RIGHTS/TERMS AND CONDITIONS

- Authors/employers retain all proprietary rights in any process, procedure, or article of manufacture described in the Work.
- Authors/employers may reproduce or authorize others to reproduce the Work, material extracted verbatim from the Work, or derivative works for the author's personal use or for company use, provided that the source and the IEEE copyright notice are indicated, the copies are not used in any way that implies IEEE endorsement of a product or service of any employer, and the copies themselves are not offered for sale.
- Although authors are permitted to re-use all or portions of the Work in other works, this does not include granting third-party requests for reprinting, republishing, or other types of re-use.The IEEE Intellectual Property Rights office must handle all such third-party requests.
- Authors whose work was performed under a grant from a government funding agency are free to fulfill any deposit mandates from that funding agency.

AUTHOR ONLINE USE

- **Personal Servers**. Authors and/or their employers shall have the right to post the accepted version of IEEE-copyrighted articles on their own personal servers or the servers of their institutions or employers without permission from IEEE, provided that the posted version includes a prominently displayed IEEE copyright notice and, when published, a full citation to the original IEEE publication, including a link to the article abstract in IEEE Xplore. Authors shall not post the final, published versions of their papers.
- **Classroom or Internal Training Use.** An author is expressly permitted to post any portion of the accepted version of his/her own IEEE-copyrighted articles on the author's personal web site or the servers of the author's institution or company in connection with the author's teaching, training, or work responsibilities, provided that the appropriate copyright, credit, and reuse notices appear prominently with the posted material. Examples of permitted uses are lecture materials, course packs, e-reserves, conference presentations, or in-house training courses.
- **Electronic Preprints.** Before submitting an article to an IEEE publication, authors frequently post their manuscripts to their own web site, their employer's site, or to another server that invites constructive comment from colleagues. Upon submission of an article to IEEE, an author is required to transfer copyright in the article to IEEE, and the author must update any previously posted version of the article with a prominently displayed IEEE copyright notice. Upon publication of an article by the IEEE, the author must replace any previously posted electronic versions of the article with either (1) the full citation to the



IEEE work with a Digital Object Identifier (DOI) or link to the article abstract in IEEE Xplore, or (2) the accepted version only (not the IEEE-published version), including the IEEE copyright notice and full citation, with a link to the final, published article in IEEE Xplore.

Questions about the submission of the form or manuscript must be sent to the publication's editor.
Please direct all questions about IEEE copyright policy to:
IEEE Intellectual Property Rights Office, copyrights@ieee.org, +1-732-562-3966

VITA

Vanya Cherneva is Bulgarian native-born. She holds Bachelor of Engineering, and Master of Computer System and Information Technology degrees, both from Vasil Levski National Military University, Republic of Bulgaria. She started her study in Louisiana State University in August of 2012. Working toward her graduate studies in Electrical and Computer Engineering Division, with focus on Computer Engineering. In 2016 she earned her second Master's degree from Electrical and Computer Engineering Division, LSU. She is currently a doctoral candidate with Electrical and Computer Engineering Division.