Building Security Protocols Against Powerful Adversaries

THÈSE N^{O} 7079 (2016)

PRÉSENTÉE LE 7 OCTOBRE 2016

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE D'ARCHITECTURE DES RÉSEAUX
PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Iris SAFAKA

acceptée sur proposition du jury:

Dr O. Lévêque, président du jury
Prof. A. Argyraki, Prof. C. Fragouli, directrices de thèse
Prof. P. Papadimitratos, rapporteur
Prof. S. Diggavi, rapporteur
Prof. B. Ford, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Suisse
2016

What matters most is how
well you walk through fire.

— Charles Bukowski

To my family,

Afroditi & Thanasis, Alexandros, and Lorenzo

Acknowledgments

First and foremost, I would like to express my gratitude and respect to my advisor Prof. Christina Fragouli. I am grateful to Christina for giving me the opportunity to work on exciting research topics, for her exceptional technical guidance and for her ever-positive attitude, motivating approach and support that have been of significant importance during my PhD. I truly believe Christina is an example of a brilliant advisor and of the kind of professors academia nowadays needs. She has helped me to develop both professionally and personally and I feel honored to have had her as my advisor.

Second, I would like to thank my co-advisor Prof. Katerina Argyraki for her guidance and support during my PhD. While collaborating with Katerina, I had the unique opportunity to acquire valuable knowledge by observing her conducting outstanding research while also delivering top quality academic teaching. She has inspired me in various ways and she has motivated me to always aim for excellence. Her genuine advise, help and support were fundamental toward completion of this thesis and I am truly grateful to her.

I would also like to thank the members of my thesis committee, Dr. Olivier Lévêque, Prof. Suhas Diggavi, Prof. Panos Papadimitratos and Prof. Bryan Ford, for accepting to evaluate my work and for providing useful feedback.

I am very grateful to our secretary Françoise Behn and our system administrator Damir Laurenzi, the "invisible" support team of my thesis – and of many others' as well. From the first day I arrived in Lausanne, these people ensured that I felt welcomed and that there was always someone to whom I could turn to for support. Their helpful attitude and effectiveness in providing solutions have created a functional lab environment that allowed us to focus undistracted on our research.

In EPFL I had the opportunity to meet, collaborate and befriend with exceptional people. First, I would like to thank my colleagues and friends in ARNI, Emre Atsan, Ayan Sengupta, László Czap and Siddhartha Brahma, and in NAL, Mihai Dobrescu and Pavlos Nikolopoulos, for their friendship and support during the good and the bad moments of the PhD. Second, a big thanks to my friends and fellow PhD students Christina Vlachou and Sofia Karygianni for being a second family to me, and to Dorina, Manos, Yiannis, Tassos, Matt, Iraklis and Vassilis for sharing unforgettable moments through these years. Finally, a special thank to Marina and George, whose support was crucial during my first months in Lausanne, and also to my friends Katerina, Yiannis, Antonis, Thomas, Athina and Sissy for always being there for me.

Acknowledgments

Finally, I would like to thank my parents, Afroditi and Thanasis, and my brother Alexandros. Their unconditional love and support, not only during my PhD but in every step of my life so far, has always been my reference point and has given me the strength and motivation to overcome difficulties. Last but not least, a heartfelt thanks to Lorenzo – I cannot indeed thank him enough for the love, care, support and motivation he has given me from the very beginning.

Lausanne, 20 June 2016 Iris Safaka

Abstract

As our sensitive data is increasingly carried over the Internet and stored remotely, security in communications becomes a fundamental requirement. Yet, today's security practices are designed around assumptions the validity of which is being challenged. In this thesis we design new security mechanisms for certain scenarios where traditional security assumptions do not hold.

First, we design secret-agreement protocols for wireless networks, where the security of the secrets does not depend on assumptions about the computational limitations of adversaries. Our protocols leverage intrinsic characteristics of the wireless to enable nodes to agree on common pairwise secrets that are secure against computationally unconstrained adversaries. Through testbed and simulation experimentation, we show that it is feasible in practice to create thousands of secret bits per second.

Second, we propose a traffic anonymization scheme for wireless networks. Our protocol aims in providing anonymity in a fashion similar to Tor – yet being resilient to computationally unbounded adversaries – by exploiting the security properties of our secret-agreement. Our analysis and simulation results indicate that our scheme can offer a level of anonymity comparable to the level of anonymity that Tor does.

Third, we design a lightweight data encryption protocol for protecting against computationally powerful adversaries in wireless sensor networks. Our protocol aims in increasing the inherent weak security that network coding naturally offers, at a low extra overhead. Our extensive simulation results demonstrate the additional security benefits of our approach.

Finally, we present a steganographic mechanism for secret message exchange over untrustworthy messaging service providers. Our scheme masks secret messages into innocuous texts, aiming in hiding the fact that secret message exchange is taking place. Our results indicate that our schemes succeeds in communicating hidden information at non-negligible rates.

Key words: security, secret key generation, anonymizing networks, linguistic steganography

Riassunto

Poiché una sempre maggiore quantità di dati sensibili viene inviata via Internet ed immagazzinata nella rete, la sicurezza delle comunicazioni diventa un tema sempre più importante. Allo stesso tempo la validità di alcune delle ipotesi, sulle quali le pratiche di sicurezza in uso sono state pensate, è messa in discussione. In questa tesi proponiamo nuovi meccanismi di sicurezza, il cui funzionamento è garantito anche se alcune di queste ipotesi non sono valide.

Nella prima parte proponiamo un protocollo di generazione di chiavi per reti senza fili, la cui sicurezza non dipende dalla tradizionale ipotesi che vuole la capacità di calcolo degli avversari limitata. Il protocollo proposto sfrutta le caratteristiche intrinseche della comunicazione senza fili per permettere ad ogni coppia di nodi della rete di accordarsi su delle chiavi che sono sicure da un avversario con capacità di calcolo illimitate. Grazie ad un banco di prova e ad esperimenti simulati mostriamo che con questo protocollo è possibile creare migliaia di bit segreti per secondo.

Nella seconda parte proponiamo un schema che permette di comunicare in forma anonima in reti senza fili. Il nostro protocollo mira ad offrire un'anonimia simile a quella offerta dal protocollo Tor – ma, a differenza di quest'ultimo, è in grado di resistere ad attacchi d'un avversario con capacità di calcolo illimitate – grazie all'uso del protocollo di generazione di chiavi proposto nella prima parte. La nostra analisi e i risultati delle nostre simulazioni indicano che questo schema offre un livello d'anonimia simile a quello raggiunto da Tor.

Nella terza parte progettiamo un protocollo di criptazione per proteggere le comunicazioni nelle reti di sensori senza fili da avversari con capacità di calcolo illimitate. Il nostro protocollo mira a migliorare la sicurezza che naturalmente la codifica di rete garantisce, usando le scarse risorse disponibili su sensori a basso consumo energetico. I risultati delle nostre simulazioni mostrano che il nostro protocollo porta ad un miglioramento della sicurezza.

Per finire presentiamo un meccanismo di steganografia che permette di scambiare messaggi attraverso un fornitore di servizi di messaggistica di cui non si ha completa fiducia. Il nostro schema nasconde i messaggi segreti in testo dal contenuto apparentemente innocuo, al fine di nascondere il fatto che il messaggio segreto è stato inviato. I nostri esperimenti mostrano che lo schema riesce a comunicare l'informazione nascosta a velocità di trasmissione significative.

Acknowledgments

Key words : sicurezza, generazione di chiavi, reti anonime, steganografia linguistica

Contents

List of Figures

List of Tables

1 Introduction

Security is a fundamental requirement in communications and the development of new security practices is an ever on-going process: with constant technological progress, new security challenges and threats continuously arise. As we proceed in the era of pervasive computing and connectivity, and a significant fraction of our sensitive data is carried over the Internet and stored remotely, security becomes an indispensable necessity. Yet, the remarkable technological advances that revolutionize the availability of computing power and connectivity come to challenge the validity of traditional assumptions in the foundations of today's security practices.

Current cryptographic approaches to security are designed around computational-hardness assumptions: security breach cannot be reached in useful time since the adversaries do not posses the essential computational power. Recent technological advances in quantum computing [22], indicate, though, that computations that have been traditionally considered as "intractable" could become feasible in the very near future. As the National Institute of Standards and Technology (NIST) stresses out in a recent report [75], "within the next twenty or so years sufficiently large quantum computers will be built to break essentially all public key schemes currently in use", enabling adversaries to "seriously compromise the confidentiality and integrity of digital communications on the Internet ". There exists, therefore, an evident urgency to explore alternative approaches to security that do not rely anymore on computational-hardness assumptions.

Another traditional assumption being challenged is the "trustworthiness" assumption. We normally *trust* the various entities of a dedicated communication infrastructure we use, to respect the privacy of our communications and data. Nevertheless, trusted network entities, such as our ISP or our e-mail provider, may betray our trust for various reasons, including (but not limited to) maximizing their profits, e.g., by selling our data to other companies, or being legally forced to do so by third parties, e.g., by powerful governments or corporations. The latter threat is not placed far in the future but is indeed already present. For example, according to Google's Transparency Report [44] (as of 2016) the company "regularly receives requests from governments and courts around the word to hand over user data". Only in the

year 2014, Google received 62 thousand such requests, with the amount escalating through the years (24 thousand requests in 2010), out of which the company complied with 63%, resulting into handing over data for approximately 100 thousand user accounts. This is a clear indication that we should gradually start considering these threats that were harmless to ignore so far.

In this thesis we design new security mechanisms for protecting against adversaries in some certain where traditional assumptions do not hold. Our goal is to provide practical schemes that can be viewed both as alternatives and complements to existing practices, without requiring significant changes in the existing infrastructure. The contributions of this thesis are in three different fields of communications security: information-theoretic security, anonymizing networks and steganography.

Information-theoretic, or unconditional, security is a prominent technique for encountering computationally powerful adversaries, since its effectiveness does not rely on their computational limitations but rather on the fact that they simply do not posses enough *information* to breach security. The information theory community has long ago demonstrated the theoretical feasibility of exploiting intrinsic characteristics of the wireless channel for building information theoretic secrets [98, 67]. Despite the large number of theoretical contributions in this field, there have been few advances toward developing concrete secret-agreement protocols that effectively operate on actual wireless networks with large number of nodes. In this thesis we aim toward this direction and we design practical secret-agreement protocols for establishing simultaneously pairwise secrets among all nodes in arbitrary wireless networks, and a data encryption protocol for enhancing the inherent weak information-theoretic security that network coding offers in wireless sensor networks.

Anonymized networking is a recent approach to secure communications in the Internet and its goal is to enable users to hide the trace of their communications: by observing the traffic at a part of the network, it should be infeasible to link a message to its originator and its final destination. This approach aims in addressing the problem where the assumption of privacy of communications is not guaranteed due to luck of trust in the communication infrastructure. There exist various examples of such networks with Tor [38] being the most widely used among them. In the core of the Tor system basic cryptographic primitives are used, e.g., the Diffie-Hellman key-agreement, RSA and AES encryption, making, therefore, Tor vulnerable to computationally powerful adversaries. In this thesis we design a traffic anynomization protocol for wireless networks that builds on the capabilities of our secret-agreement protocols to offer anonymity in a fashion similar to Tor, yet being robust against computationally powerful adversaries.

Steganography is the practice of hiding information within innocuous data, in such a way that only the intended communication parties know that a piece of data carries hidden information and are able to extract it. Steganographic techniques aim in addressing the problem where private communication is not allowed by the infrastructure, but its users still wish for this

property. Carriers for embedding hidden information include images, videos, audio files or text. The latter case is usually referred to as *linguistic* steganography and requires tools from both Natural Language Processing (NLP) and communications security fields. Existing linguistic steganographic techniques (a) enable the transfer of only a few secret bits per communication round, and (b) require off-line access to sophisticated NLP tools and large linguistic resources, being, therefore, quite impractical to use. In this thesis we design an implementable linguistic steganographic mechanism for exchanging short private messages, in the order of a few hundreds of bits, over untrustworthy messaging service providers.

Our Contributions

The contributions of this thesis are the following:

- First, we design practical secret-agreement protocols for wireless networks, that build on recent theoretical results in information-theoretic security [89, 88]. We leverage broadcast and the existence of *packet erasures* that naturally arise in wireless networks, in order to enable a set of nodes to simultaneously agree on common pairwise secrets, simply by exchanging traffic among them, both in single-hop and in arbitrary multi-hop networks. The security of the secrets we produce does not depend on computational or memory limitations of an adversary, but rather on her limited network presence, i.e., the fact that she cannot overhear every transmission in the network. We evaluate our protocols through testbed experimentation and simulations, and we show that it is feasible in practice to create thousands of secret bits per second, without assuming anything about the adversary's computational capabilities.
- Second, we propose a traffic anonymization scheme for wireless networks that builds on the capabilities of the secrets we construct using our secret-agreement protocols. Our traffic anonymization protocol aims in providing anonymity in a fashion similar to Tor [38]: Tor achieves anonymity by bouncing encrypted communications around a distributed network of relays; we similarly bounce encrypted communications among the wireless network nodes. By leveraging the capabilities of our secrets, that do not depend on cryptographic primitives, our approach is resilient to computationally unbounded adversaries. Our privacy analysis and simulation results indicate that our approach can offer a level of anonymity comparable to the level of anonymity that Tor does.
- Third, we design a lightweight data encryption protocol, suitable for wireless sensor networks, that builds on top of the operations of a data collection protocol that employs network coding. We leverage the shared information between nodes and the sink across communication rounds to enable secure data delivery under the presence of network limited, yet computationally unconstrained adversaries. Our approach can be viewed as an enhancement of the weak security that network coding inherently offers, with low additional operational complexity. We experimentally evaluate the performance of our protocol over several settings and we demonstrate that our approach yields substantial

improvements in comparison to the level of weak security that network coding naturally offers.

- Finally, we design a linguistic steganographic mechanism for enabling exchange of short secret messages over untrustworthy messaging service providers. We propose a semi-automated approach that masks a secret message into innocuous text by involving the "user-in-the-loop"; the message is initially mapped to a sequence of linguistic words that the user enhances to produce the final text. Our approach does not require off-line access to large linguistic resources and sophisticated NLP tools and it is easily implementable. We implement our design and we evaluate its performance by experimenting with human users, through the Mechanical Turk platform [15], and by applying steganalysis methods that we design. Our results indicate that (a) our approach succeeds in embedding a large number of hidden bits, without requiring an unreasonable amount of user-effort, and (b) the "user-in-the-loop" involvement helps in hiding the existence of steganography.

Thesis Outline

The rest of the thesis is organized into four chapters. Chapter 2 presents the design, theoretical analysis and experimental evaluation of our secret-agreement protocols. Chapter 3 presents the design, privacy analysis and performance evaluation of our Tor-like traffic anonymization scheme, that leverages the properties of the secrets we produce with our secret-agreement protocols. Chapter 4 presents the design, theoretical analysis and the experimental evaluation of our data encryption protocol for wireless sensor networks. Chapter 5 presents the design, the steganalysis attacks and the evaluation of our linguistic steganographic mechanism.

2 Secret-agreement Protocols

2.1 Introduction

In this chapter we consider the problem where a group of n wireless nodes that form an ad-hoc wireless network, want to create $\binom{n}{2}$ pairwise secrets, such that a passive eavesdropper Eve, who is located in an unknown position in the network, learns very little about them. We are interested in strong information-theoretical or unconditional security, where the security of the secrets does not depend on computational limitations of Eve, but rather on the fact that Eve does not posses enough *information* to breach security. We are investigating, whether it is possible to offer strong security, as the number of nodes n and number of pairwise secrets increases, and over arbitrary wireless topologies.

In recent years, there has been significant interest in building information-theoretical security out of wireless channel properties, but the work has been limited to very specific topologies and scenaria. The majority of the work considers pairwise key generation over a single channel with a single source and receiver [98, 67] (see also [54] and references therein); the few works that have looked at multiple receivers still only consider a single source and receivers within the same broadcast domain [37, 39]. Works that look at larger networks typically do not provide strong, but weak information security guarantees [20, 95, 14], and mostly focus on single message distribution, as opposed to creating $\binom{n}{2}$ different secret keys (see also section 2.9 on related work). Moreover, in most of the proposed practical works, the secret key generation rates achieved are only a few tens of bits per second [100, 62, 80]. In contrast, we show in this chapter that can we leverage both channel and network properties, to create pairwise keys at rates that are of the order of Kb per second, for arbitrary n and wireless network topologies.

First, we present a basic secret-agreement protocol, which enables n nodes connected to the same broadcast domain to create pairwise secrets that Eve knows very little about. Our protocol leverages the broadcast nature of the wireless to create pairwise secrets between *all* pair of nodes simultaneously, has polynomial time complexity and is readily implementable in simple wireless devices. We analyze our protocol in two ways: (i) Under standard information-theory assumptions (independent erasure channels between nodes and known erasure probabilities),

we formally show that: (1) Our basic protocol is information-theoretically secure, i.e., it leaks no information to Eve about the secrets. (2) It achieves a secret-generation rate that is optimal for $n = 2$ nodes and scales well with the number of nodes n. (ii) Through experimental evaluation, and estimation of the network parameters, as we discuss later.

Second, we consider secret-agreement over arbitrary, multi-hop networks. This is important, firstly, from a practical point of view: even when networks have a small number of nodes, as connectivity is impaired from distance, interference and other impediments (e.g., metal obstructions), it is challenging to consistently maintain a single-hop connected network. Secondly, multi-hop networks are also interesting from a technical point of view since they provide two new opportunities for secrecy that we could leverage: interference and multi-path propagation. Interference between concurrent transmissions (such as caused by the hidden terminal problem) may interfere with Eve's reception but not with the reception of other legitimate nodes; distinct packet propagation through multiple paths can ensure that Eve, located in an unknown but fixed position in our network, does not have access to all of them, and again misses packets that legitimate nodes receive. Finally, secret-agreement over arbitrary multi-hop networks can enable applications similar to the one we describe in chapter 3.

We design a secret-agreement protocol for multi-hop networks, that builds on our basic protocol, but also comprises new design features that realize the benefits multi-hop offers for secrecy. This includes a customized packet dissemination protocol that balances two conflicting goals: spreading the packets as efficiently and as widely as possible among the legitimate nodes, while ensuring that a significant fraction of packets will not be overheard by Eve, who could be located in any place within the network. Our protocol is completely decentralized, does not differentiate between nodes and is readily implementable in simple wireless devices.

Third, we experimentally evaluate the performance of our protocols and we provide evidence that it is feasible in practice to create pairwise secrets at rates of thousands of bits per second in realistic setups. In the experimental setup, we assume no knowledge of channel parameters, and no knowledge of Eve's location or collected information – we estimate the quantities we need online. For the single-hop case, we use a small wireless testbed and for the multi-hop case we simulate different network configurations, consisting of up to 500 nodes and located up to 5-hops apart. We show that we can achieve secret generation rates in the magnitude of Kbps, independently from the adversary's computational capabilities.

The work presented in this chapter has been presented in [86] and [84].

2.2 Setup

System Model

We consider a set of n wireless nodes, $T_1, \ldots, T_n$, that form an ad-hoc network. We will refer to these nodes as *legitimate terminals*, or simply *terminals*. Sometimes we will refer to terminals T_1, T_2, T_3, and T_4 respectively as Alice, Bob, Calvin, and David.

We capture the network structure using two parameters:

- **Number of hops** describes the maximum distance (in hops) between any two terminals. More formally, in a k-hop network, for any two terminals T_i / T_j in the network there exists a k-hop path $T_i, r_1, \ldots, r_{k-1}, T_j$ with $k-1$ intermediate terminals such that every terminal is the neighbor of its preceding terminal along the path; moreover, there exists at least one pair of terminals for which there is no path with $k-1$ hops.
- **Network density** expresses the expected number of terminals per unit network area; it affects the expected number of neighbors that a terminal has. We define the *unit area* as an 1-hop network, i.e., a network where all terminals are within the same broadcast domain.

The terminals communicate with each other in three ways:

- When we say that terminal T_i *transmits* a packet, we mean that it broadcasts the packet once, within its broadcast domain.
- When we say that terminal T_i *reliably broadcasts* a packet, we mean that it ensures that all other terminals $T_{j\neq i}$ in the whole network receive it, e.g., through ACKs and re-transmissions.
- When we say that terminal T_i *unicasts* a packet to terminal T_j, we mean that the packet is intended only for terminal T_j and it might get re-transmitted up to a certain number of times.

Each terminal in the network has a unique id, that is revealed to all other terminals, and it can generate and transmit *random* packets. A random packet has a payload of L symbols over a finite field $\mathbb{F}_q$ and thus has a size of $L \log q$ bits. The payload of a random packet is drawn from the uniform distribution. Each packet has a *unique identifier*, that consists of the generator's unique id together with a sequence number.

We assume that in our network *packet erasures* occur. A terminal experiences a packet erasure, or simply misses a packet (knows nothing about its content), if the packet gets transmitted in the network but cannot get received by the terminal's radio receiver. Packet erasures may occur in our network due to different effects (and/or combinations of these) that inherently

arise in wireless networks: channel noise and fading (low reception SNR), collisions because of concurrent transmissions, a packet was transmitted outside a terminal's reception region etc. Independently of the causing effect(s), whenever a terminal's radio receiver was not able to lock on a packet's transmitted physical signal, we account this event as a packet erasure.

Our goal is to design protocols, that exploit packet erasures, in order to enable each pair of terminals in the network, T_i / T_j, to create a secret $\mathcal{S}_{ij}$, that is secure from an adversary as we model in the following.

Adversary Model

We assume that in our network there exists a passive[1] adversary Eve, a *non*-legitimate node located at an unknown position, who eavesdrops every transmission in her reception region. Eve does not make any transmission herself, but uses the eavesdropped information at her disposal to compromise the security of the secrets created by the legitimate terminals.

We assume that Eve has access to the same physical layer (radio technology, number of antennas etc.) as the legitimate terminals – experiencing, therefore, packet erasures as they do. However, we also assume that Eve has infinite memory as well as unbounded computational capabilities at her disposal; this would follow the model of an adversary that does not want to reveal her presence by using specialized equipment, yet has offline access to unbounded resources to breach security. Moreover, we assume that Eve has perfect knowledge of the protocols used, of the network topology and of the terminals' identities. To be conservative, we also assume that Eve receives correctly *all* reliably broadcasted and unicasted (as defined in section 2.2) packets. Eve, using her knowledge, can optimally position herself inside the network, and keep her position secret. However, she has limited network presence; in the following we assume that she is situated in a single position, yet this assumption can be relaxed in the case where Eve is in multiple positions, as we discussed in [85].

Apart from the existence of Eve, we additionally assume that every legitimate terminal T_i in the network may act as "honest but curious" towards the other terminals: T_i runs the secret-agreement protocols honestly but may as well try to eavesdrop on other terminals' communications. Note that, this additional assumption, is not related to the behavior of Eve (she remains a passive adversary that reveals nothing about her knowledge) and does not weaken our adversary model. In contrast, with this assumption we aim for protection in a stricter context than if we only considered the behavior of Eve: our ultimate goal is to create $\binom{n}{2}$ different secrets between all pairs of nodes, in such a way so that every pairwise secret is secure from Eve but also from *any* other honest but curious node in the network.

[1] We discuss the case where Eve is an active adversary in section 2.8.

Possible Use-Case Scenarios

A possible use-case would correspond to the scenario where a group of n political dissidents rendezvous in a public place (potentially under visual surveillance) and use their cell phones in ad-hoc mode to secretly communicate; or the scenario where a group of n friends connect to the same social network and use their cell phones in ad-hoc mode to exchange private content. It should be infeasible for an eavesdropper who listens in on the same broadcast domain to record what she overhears, process the recording, and reconstruct their communications. Moreover, it should be infeasible for an eavesdropper to record what she overhears, extract from the dissidents/friends a set of passwords or keys, combine them with the recording, and reconstruct their communications. The dissidents/friends can periodically use our protocol to create pairwise secrets and use these secrets to continuously refresh the keys with which they encrypt/authenticate their communications.

Theoretical Network Conditions

We define the theoretical network conditions as follows:

1. When terminal T_i transmits a packet, terminal T_j (Eve):
   - misses the entire packet, with probability δ_{ij} (δ_{iE})
   - receives the entire packet correctly, otherwise.

   δ_{ij} (δ_{iE}) is the *erasure probability* of the $T_i - T_j$ (T_i – Eve) channel.
2. The $T_i - T_j$ channel is independent from any $T_i - T_{l\neq j}$ channel[2] and the T_i – Eve channel, for all i, j, l.
3. The erasure probability δ_{iE} of the T_i – Eve channel is known, for all i.

Performance Metrics

We use the following metrics to evaluate the performance of our secret-agreement protocols:

- *Efficiency* captures the cost of the protocol, i.e., the amount of traffic it produces in order to generate pairwise secrets of a given size. The efficiency achieved by two terminals T_i and T_j that create a secret $\mathcal{S}_{ij}$, of length $|\mathcal{S}_{ij}|$ bits, is defined as:

$$E_{ij} = \frac{|\mathcal{S}_{ij}|}{total\ \ transmitted\ \ bits}.$$

  The denominator is the total number of bits transmitted by the protocol until $\mathcal{S}_{ij}$ is created. In the case of multi-hop setups, this number includes re-transmissions of random packets from terminals other than the generator terminals.

[2] Assuming independent channels is not necessary for any of our results, but simplifies our proofs.

- *Secrecy rate* measures how many secret bits per second are created between a pair of terminals; the secrecy rate is a function of the efficiency and the transmission rate.

- *Reliability* captures the quality of the created secrets, i.e., the extent to which they are unknown to Eve. The reliability of a secret $\mathcal{S}$ is defined as:

$$R_{\mathcal{S}} = \frac{H(\mathcal{S}|\mathcal{X}_E)}{H(\mathcal{S})},$$

  where $\mathcal{X}_E$ is the information obtained by Eve via eavesdropping on the terminals' communications, $H(\mathcal{S})$ is Eve's entropy (her uncertainty about $\mathcal{S}$ before she eavesdrops), and $H(\mathcal{S}|\mathcal{X}_E)$ is Eve's conditional entropy (her uncertainty about $\mathcal{S}$ after she eavesdrops). $R_{\mathcal{S}} = 1$ implies information-theoretical secrecy: $I(\mathcal{S};E) = 0$; in other words, Eve does not learn anything about $\mathcal{S}$ by observing the protocol and the produced traffic. $R_{\mathcal{S}} < 1$ means that Eve can correctly guess the value of one bit of generated secret with probability higher than 0.5, e.g., $R_{\mathcal{S}} = 0.8$ means that this probability is $2^{-0.8} = 0.57$.

If the terminals had knowledge of the exact information observed by Eve via eavesdropping, they would always be able to construct the longest possible secrets of reliability 1 (information-theoretically secure secrets). In practice, the terminals do not have access to this knowledge; the best they could do is to compute an estimate $\hat{\mathcal{X}}_E$ of $\mathcal{X}_E$. Under well-defined network models this estimation can become arbitrarily good, enabling the terminals to create secrets, using our protocols, of reliability 1 (we show this for the case of the erasure channel model in section 2.5 and Appendices A.1, A.2). In real-world wireless networks, where theoretical conditions do not hold, the terminals need to *heuristically* compute this estimation (we elaborate on this in section 2.6). Needless to mention, in case they underestimate $\hat{\mathcal{X}}_E$, the constructed secrets will have reliability less than 1.

In section 2.7, we experimentally evaluate the performance of our secret-agreement protocols by measuring (i) the *ideal* efficiency/secrecy rate; this is the efficiency/rate achieved by an *oracle-assisted* protocol, that is, a protocol that works like ours, with the only difference that it does not estimate how much information Eve obtains through eavesdropping – that knowledge is directly provided to the legitimate terminals by the oracle, (ii) the *effective* efficiency/secrecy rate that is achieved by our protocols, where secrets are constructed based on estimations of Eve's knowledge. The reliability is a metric that allows us to capture how close does our protocol behave to the oracle-assisted one. Ideally, we would like our protocols to achieve high efficiency/secrecy rate, along with reliability scores as close as possible to 1.

2.3 Basic Secret-agreement Protocol

In this section, we describe the core of our secret-agreement protocol, that enables terminals T_i and T_j, which are connected in the same broadcast domain, i.e., they form a single-hop network, to create a secret $\mathcal{S}_{ij}$. Assuming the theoretical network conditions (as defined in

Symbol	Meaning
n	Number of terminals
T_i	Terminal i
$\mathcal{S}_{ij}$	Secret between terminals T_i and T_j
δ_{ij}	Erasure probability of $T_i - T_j$ channel
δ_{iE}	Erasure probability of T_i – Eve channel
N	Number of x-packets transmitted by each terminal (initial phase, step 1)
M_{ij}	Number of shared y-packets constructed by T_i and T_j (privacy amplification phase, steps 1 – 3)

Table 2.1 – Commonly used symbols in our secret-agreement protocols.

section 2.2), $\mathcal{S}_{ij}$ is perfectly secret from any terminal $T_{k\neq i,j}$ and an adversary Eve (we show this in section 2.5.1). In Table 2.1 we explain the meaning of commonly used symbols throughout this section.

Main Idea

Suppose Alice and Bob exchange three random packets, x_1, x_2 and x_3. Suppose Eve misses (knows nothing about the contents of) two of the packets shared by Alice and Bob, x_1 and x_2. If an oracle told Alice and Bob that Eve misses two of their shared packets (but not which two), they could create a perfect shared secret (one that Eve knows nothing about), by using two linear combinations of their shared packets, e.g., $\langle x_1 + x_2, x_2 + x_3 \rangle$[3] (where + denotes addition over a finite field, e.g., bit wise XOR over the binary field).

Building on this idea, our protocol consists of two phases: In the *initial* phase, the terminals exchange traffic to ensure that each terminal pair shares some number of random packets (as Alice and Bob share x_1, x_2, and x_3 in the example above). This happens over n rounds, with a different terminal transmitting in each round. In the *privacy amplification* phase, each terminal pair creates a secret out of the information they shared in the initial phase. For this, they "compress" their shared information enough to ensure that any other terminal or Eve know nothing about the secret (as Alice and Bob "compress" x_1, x_2, and x_3 into $x_1 + x_2$, $x_2 + x_3$ in the example above). To do this compression correctly, the terminals need to know how much of their traffic exchange was overheard by Eve (but not which particular bits).

A naive approach would be to have each terminal pair create their secret separately, which would not scale well with the number of terminals. Instead, our protocol creates the pairwise secrets simultaneously, by harnessing the broadcast nature of wireless networks.

[3] This secret is perfect, because Eve's probability of guessing its value is equal to the probability of guessing the values of the two packets she misses.

2.3.1 Algorithm

Each terminal T_i maintains $n-1$ queues $\mathcal{Q}_{ij}$, $j \neq i$. In the beginning, these are empty.

Initial Phase

In round $r = 1 \ldots n$:

1. Terminal T_r generates and transmits N random packets (we will call them *x-packets*).
2. Each terminal $T_{i \neq r}$ reliably broadcasts the identifiers of the x-packets it received.
3. Each terminal T_i adds to queue $\mathcal{Q}_{ij}$ the identifiers and contents of the x-packets it shares with terminal $T_{j \neq i}$.

At this point, $\mathcal{Q}_{ij}$ contains all the packets shared by terminals T_i and T_j.

Privacy Amplification Phase

For $i = 1 \ldots n-1$:

1. Terminal T_i constructs M_{ij} linear combinations of the packets in the queue $\mathcal{Q}_{ij}$, for all $j > i$ (we will call them *y-packets*). It determines the number of y-packets M_{ij} and constructs the y-packets as described in section 2.3.2.
2. Terminal T_i reliably broadcasts the coefficients it used to construct the y-packets.
3. Each terminal $T_{j>i}$ uses the broadcasted coefficients and the contents of its queue $\mathcal{Q}_{ji}$ to reconstruct the M_{ij} y-packets.

At this point, terminals T_i and $T_{j>i}$ share M_{ij} y-packets. Their secret $\mathcal{S}_{ij}$ is the concatenation of these y-packets.

An Example Agreement

Suppose we have $n = 3$ terminals, Alice, Bob, and Calvin, and a passive adversary, Eve. All the channels between terminals or any terminal and Eve have erasure probability $\delta = 0.5$.

In the initial phase, the terminals create shared information by exchanging packets. In the first round, Alice transmits $N = 8$ x-packets, $a_1, a_2, \ldots a_8$, of which Bob, Calvin, and Eve receive (not the same) half. Similarly, in the second and third rounds, Bob transmits $b_1, b_2, \ldots b_8$, and Calvin transmits $c_1, c_2, \ldots c_8$. Alice, Bob, and Calvin know which x-packets are received by one another (thanks to Step 2 of the initial phase), but not which x-packets are received by Eve.

Alice	Bob	Calvin	Eve
$a_1, a_2, \ldots a_8$	a_1, a_2, a_3, a_4	a_1, a_2, a_5, a_6	a_1, a_3, a_5, a_7
b_1, b_2, b_3, b_4	$b_1, b_2, \ldots b_8$	b_1, b_2, b_5, b_6	b_1, b_3, b_5, b_7
c_1, c_2, c_3, c_4	c_1, c_2, c_5, c_6	$c_1, c_2, \ldots c_8$	c_1, c_3, c_5, c_7

Table 2.2 – Information known to each node.

Phase	Alice – Bob	Alice – Calvin	Bob – Calvin
Initial	$\cancel{\bar{a}_1}, \bar{a}_2, \cancel{a_3}, a_4$ $\cancel{\bar{b}_1}, \bar{b}_2, \cancel{b_3}, b_4$ $\cancel{\bar{c}_1}, \bar{c}_2$	$\cancel{\bar{a}_1}, \bar{a}_2, \cancel{a_5}, a_6$ $\cancel{\bar{b}_1}, \bar{b}_2$ $\cancel{\bar{c}_1}, \bar{c}_2, \cancel{c_3}, c_4$	$\cancel{\bar{a}_1}, \bar{a}_2$ $\cancel{\bar{b}_1}, \bar{b}_2, \cancel{b_5}, b_6$ $\cancel{\bar{c}_1}, \bar{c}_2, \cancel{c_5}, c_6$
Privacy Amp.	$a_3 + a_4$ $a_1 + a_2 + a_3$ $b_3 + b_4$ $b_1 + b_2 + b_3$	$a_5 + a_6$ $a_1 + a_2 + a_5$ $c_3 + c_4$ $c_1 + c_2 + c_3$	$b_5 + b_6$ $b_1 + b_2 + b_5$ $c_5 + c_6$ $c_1 + c_2 + c_5$

Table 2.3 – Information shared by nodes.

Table 2.2 shows the x-packets known to each node at the end of the initial phase. Table 2.3 (top row) shows the x-packets shared by each terminal pair at the end of the initial phase (e.g., Alice and Bob share a_1, a_2, a_3, a_4 among others). To help visualize who knows which x-packets, from the x-packets shared by Alice/Bob, we mark those known to Eve[4] as "canceled out" (e.g., $\cancel{a_3}$), those known to Calvin as "barred" (e.g., $\bar{a}_2$), and those known to both Eve and Calvin as both canceled out and barred (e.g., $\cancel{\bar{a}_1}$). We do the same for the other terminal pairs.

In the privacy amplification phase, the terminals create pairwise secrets by compressing their shared information. Alice and Bob compress their 10 shared x-packets into $M_{12} = 4$ shared y-packets (linear combinations of the shared x-packets). Similarly, Alice/Calvin and Bob/Calvin compress their 10 shared x-packets into 4 shared y-packets. Table 2.3 (bottom row) shows the y-packets shared by each terminal pair. Notice that Eve cannot reconstruct any of these y-packets; she misses at least one x-packet in every linear combination constructed by the terminal pairs (e.g., Eve misses packet a_4, hence she cannot reconstruct the y-packet $a_3 + a_4$, that Alice and Bob have constructed and serves as one of their pairwise secrets). For the same reason, Calvin cannot reconstruct the y-packets constructed by Alice and Bob for their pairwise secret (e.g., Calvin misses packet b_3, he cannot, therefore, reconstruct the y-packet $b_1 + b_2 + b_3$). Similarly, Alice (Bob) cannot reconstruct the y-packets constructed by Bob (Alice) and Calvin for their pairwise secret.

This was an example to give a sense of how things work. Our protocol does not really construct so simple linear combinations (e.g., 4 random linear combinations out of 10 x-packets), as they may leak information to Eve (section 2.3.2).

[4]For sake of simplicity, we assume (only in this example) that this knowledge is provided to Alice, Bob and Calvin. In our protocol, the knowledge of Eve has to be estimated by the legitimate terminals; however, they do not need to estimate which packets Eve has overheard only *how many* (see section 2.3.2).

Key Points

The size of the secret between two terminals depends on (1) the amount of information shared by the two terminals and (2) how much of this information Eve and the other terminals have missed. In the above example, Alice and Bob share 10 x-packets. Of these, Eve misses 5, and Calvin misses 4. Hence, Alice and Bob can construct up to 5 y-packets (linear combinations of their shared x-packets) that are perfectly secret from Eve, and up to 4 y-packets that are perfectly secret from Calvin. Since we want the Alice/Bob secret to be unknown to both Eve and Calvin, Alice/Bob should create only 4 y-packets. Creating a shorter secret would be inefficient. Creating a longer secret would necessarily result in Eve or Calvin knowing something about the secret (though not necessarily the entire secret).

An important feature of the protocol is that terminals T_i and T_j create shared information during all the rounds of the initial phase, not only when one of them transmits. In the above example, at the end of the initial phase, Alice and Bob share not only x-packets transmitted by one of them, but also x-packets transmitted by Calvin (c_1, c_2). In the particular example, these packets turn out not to be useful in creating the Alice/Bob secret, because Calvin knows both of them (and we want the secret to be unknown to Calvin). However, when we have more than $n = 3$ terminals, leveraging x-packets transmitted by all terminals becomes key to the protocol's scalability with the number of terminals. For instance, imagine that there is a fourth terminal, David, which transmits x-packets d_1, d_2, received by Alice/Bob, but not Calvin or Eve. Although d_1, d_2 are known to David, now Alice/Bob can create two combinations of c_1, c_2, d_1, d_2 (e.g., $c_1 + d_1, c_2 + d_2$) and create two extra y-packets unknown to Calvin, David, and Eve.

2.3.2 Secret Construction

Terminals T_i and T_j construct the following number of y-packets in the privacy amplification phase:

$$M_{ij} = \min\{ V_E, V_1, V_2, \ldots V_n \}, \tag{2.1}$$

where:

- V_E is the expected number of x-packets that are shared by terminals T_i / T_j and missed by Eve.
- V_l is the number of x-packets shared by terminals T_i / T_j and missed by terminal T_l.

We compute V_E as $\sum_{r=1}^{n} U_{rE}$, where $U_{rE} = \delta_{rE} \cdot U_r$, and U_r is the number of x-packets transmitted by terminal T_r and received by both terminals T_i / T_j in round r of the initial phase. In short, we count, for each terminal and for Eve, how many of T_i / T_j's shared x-packets this terminal/Eve has missed (or is expected to have missed, in Eve's case), and we set M_{ij} to the smallest of these numbers.

It is straightforward to adapt this computation to the scenario where up to some number of terminals collude to learn $\mathcal{S}_{ij}$, but we do not consider this scenario here.

Terminals T_i and T_j construct the y-packets using simple constructions based on Maximum Distance Separable (MDS) codes [64], as described in Lemma 7 in the Appendix. There is no novelty in these constructions (they rely on standard properties of MDS codes). One such property is that, if Eve has t packets, then each y-packet involves at least $t+1$ packets, which ensures that Eve cannot reconstruct it.

2.4 Secret-agreement for Multi-hop Networks

In this section we describe a secret-agreement protocol for multi-hop networks, that builds on the basic protocol (section 2.3) and comprises new design features. In addition to channel noise and fading, multi-hop networks offer two more sources of packet erasures, that we aim to exploit for creating secrets: (1) interference from simultaneous transmissions, (2) existence of multiple paths between terminals. We design a protocol, consisting of a *packet dissemination* phase followed by a *feedback* phase, that essentially replaces the *initial* phase of the basic secret-agreement protocol. Before giving the protocol description, we illustrate its core design concepts.

Leveraging More Than Channel Noise

During the initial phase of the basic protocol, each terminal simply generates and transmits N x-packets during its round. For multi-hop networks, we need a more sophisticated dissemination protocol, that balances two goals: on one hand maximizing the number of random packets between every pair of terminals, and on the other hand, minimizing the number of packets that Eve overhears. For instance, having a terminal generate random packets and flooding the network with them does not work well, because Eve ends up overhearing most of these packets, and thus they cannot be exploited for secrecy. We need a protocol that efficiently "creates erasures"; a protocol that, first, exploits the intrinsic opportunities that wireless multi-hop networks offer to evoke packet erasures and, second, it does so in a way that ensures as much as possible uncorrelated packet receptions from legitimate terminals, without requiring unnecessarily many packet transmissions (that would yield a very low efficiency). We design our packet dissemination protocol leveraging the following:

1) *Channel noise and fading.* Ideally, we would like the broadcast transmissions to be subject to independent erasures across the receivers so that Eve does not receive exactly the same packets as her close neighbors. To achieve this, in the dissemination protocol we have every terminal in the network act as a source, to uncorrelate as much as possible the quality of reception from a terminal's location. Additionally, each terminal broadcasts a random packet it generates exactly once (without re-transmissions). Note that we can do this because we do not care *which* random packets terminals share, only *how many*.

2) *Interference from simultaneous terminal transmissions.* Such interference for example occurs in the IEEE 802.11 protocol due to the hidden terminal problem. For us this is not a problem but a blessing in disguise: we would like our dissemination protocol to *incur* such interference, yet still not decrease dramatically the number of successful receptions. We allow, thus, the terminals to transmit *simultaneously* x-packets (in contrast to taking turns), at a rate that does not impose restrictively high collisions in a unit network area.

3) *Multiple paths.* If there are two paths between Alice and Bob in the network, and Eve overhears only one of them, then if Alice sends packet x_1 on one path and x_2 on the other, Eve will receive only one of the two packets. In general, if Alice and Bob are connected with ν paths, while Eve can overhear at most $z < \nu$ of these (any z), it is optimal for the key generation rate if Alice sends to Bob a *different* packet through each path: Alice and Bob will share ν packets and Eve will learn only z of them [24]. To achieve this, we need a dissemination protocol that sends *each packet through a single path.*

2.4.1 Algorithm

Additional parameters and notation

Each terminal T_i can generate x-packets but also forward the x-packets generated by any other terminal in the network. In the *unique identifier* of each generated x-packet, a field ttl is appended describing the maximum number of times this packet can be transmitted in the network. Whenever a terminal transmits an x-packet (either generated locally or received by another terminal) is referred to as the *sender* of this packet. Each terminal transmits at rate $\frac{1}{\lambda}$, where λ is the number of its neighbors.

Packet Dissemination Phase

Each terminal T_i maintains $n-1$ queues $\mathcal{Q}_{ij}$, $j \neq i$, that are empty in the beginning, and it records all overheard traffic. The packet dissemination is performed as follows:

1. Each terminal T_i generates and transmits N x-packets; it waits a random time between transmissions so that on average it transmits at rate $\frac{1}{\lambda}$.
2. Upon reception of an x-packet p, the receiver checks if this is first time it received this packet; if yes, the receiver unicasts an acknowledgment to the sender, otherwise it does not acknowledge.
3. The sender of a packet p selects a forwarder: Let $\mathcal{R}_p$ denote the set of terminals that acknowledged p. The sender chooses a terminal uniformly at random from $\mathcal{R}_p$, and unicasts a control message to inform the node it is the selected forwarder. If $\mathcal{R}_p = \emptyset$, then p is not forwarded anymore.
4. The selected forwarder of a packet p (the next sender of p), reduces the ttl field by one

and transmits it.

Steps 2 to 4 are repeated till the ttl field of all the packets in the network expires. Note that when transmitting a packet p the sender sets a timer T_p, which defines a time window for acknowledging. Once T_p has expired, step 3 takes place.

Feedback Phase

For $i = 1 \dots n$:

1. T_i constructs a $1 \times nN$ vector v_i, with a "1" in the jm^{th} position if T_i has received the packet with sequence number m from terminal T_j, and a "0" otherwise.

2. T_i reliably broadcasts v_i into the network, using special packets indicated as *feedback* packets.

3. Each terminal T_j adds to queue $\mathcal{Q}_{ji}$ the identities and contents of the x-packets it shares with terminal $T_{i \neq j}$.

Privacy Amplification Phase

The terminals perform the privacy amplification phase as described in sections 2.3.1 and 2.3.2. Each pair of terminals T_i / T_j can construct up to M_{ij} y-packets, the concatenation of which is their common secret $\mathcal{S}_{ij}$. Regarding the value of variable V_E in Equation 2.1, see section 2.5.2.

Key Points

The ttl determines how far a packet will propagate; thus it enables to control the trade-off between creating a large number of common packets between nodes, while keeping Eve's chances of overhearing low. Each terminal acts as source, so that we generate uniform traffic across the network, and make packet receptions spatially uncorrelated. Terminals transmit at random intervals to incur collisions and at rate $\frac{1}{\lambda}$ so that, as the density of the network increases, we do not cause congestion. By selecting a single forwarder we avoid flooding and exponential replication of packets; instead, each packet follows a single random walk through the network, so that we exploit multi-path erasures. Note also that the dissemination of a packet may stop early, because $\mathcal{R}_p$ may not contain receivers due to lost or late acknowledgments, or because the control message that selects a forwarder is not received. As our protocol does not aim to deliver specific messages but instead to create shared x-packets, such losses do not have a significant effect.

2.4.2 Communication Overhead Analysis

In certain steps of the secret-agreement protocol, each terminal needs to *reliably broadcast* an amount of information to other terminals, notably at (1) step 2 of the feedback phase, (2) step 2 of the privacy amplification phase. The additional communication overhead imposed by these operations varies, depending on the way we choose to implement them.

For the feedback phase, we assume that we use an efficient all-to-all broadcast dissemination scheme; indeed, many such schemes have been explored in the literature [60, 41]. In section 2.7.2, we evaluate the secrecy rate achieved by our protocol taking into account only the overhead of the packet dissemination phase; we do not take, thus, into account the overhead of the feedback phase that would depend on the particular all-to-all scheme employed. To approximately estimate how much this overhead could reduce our secrecy generation rate, we next perform a back of the envelope calculation.

For the dissemination step there are $T_d \simeq n/\lambda \times (NL\log q \times ttl)$ bits transmitted in total[5], with λ here denoting the average number of neighbors. For the feedback step we have $T_f \simeq n[\gamma(n-1)+1] \times nN$ bits, where $0 \leq \gamma \leq 1$, denoting a forwarding factor for each terminal, that depends on the broadcast protocol used. Thus, our secrecy rate would be approximately reduced by a constant factor of $1+\mu$, where μ is defined as follows:

$$\mu = \frac{T_f}{T_d} \simeq \frac{\lambda \times [\gamma(n-1)+1] \times n}{L\log q \times ttl} \tag{2.2}$$

Example: Assume a k-hop network with $k = 3$ and $n = 90$ in which we disseminate x-packets of size 1KB and $ttl = 3$, during the dissemination step. In addition, assume we use a network coding technique as described in [41] for the feedback step, for which $\gamma = 2/\lambda$ yields an almost 100% packet delivery ratio. In that case, $\mu \simeq 0.67$ meaning that the achieved rate should be divided by a factor of $1+\mu = 1.67$. For the same network and for $n = 135$, the rate should be divided by a factor of $1 + 1.51 = 2.51$.

In the privacy amplification phase, a terminal T_i needs to communicate to terminal T_j the coefficients it used for constructing the y-packets, i.e., their shared secret $\mathcal{S}_{ij}$. Depending on how the terminals intend to use this $\mathcal{S}_{ij}$, this operation could be carried out without adding any communication overhead at all. For instance, if T_i uses $\mathcal{S}_{ij}$ as an one-time-pad encryption key to send a confidential message to T_j, these coefficients can be appended at the end of the encrypted message itself. Or, the terminals could just use the same deterministic algorithm, e.g., using as input the unique ids of the two terminals, to compute independently the same MDS matrix A (see Lemma 7 in Appendix A.1); in that case no further communication is needed.

[5] We do not account for re-transmissions, since we assume a MAC layer where re-transmissions are by default disabled in broadcast mode, as in IEEE 802.11.

2.5 Protocol Analysis

2.5.1 Single-hop Networks

We state, in the following, certain properties of the basic protocol and also present an argument on why this particular protocol outperforms a more obvious alternative. We summarize the proofs of Lemmas 1 and 4 in Appendix A.1. We omit the proof of Lemma 2, which is straightforward.

Lemma 1. *If the theoretical network conditions hold, there exists a sufficiently large N for which the basic protocol is information-theoretically secure against a passive adversary.*

From the previous lemma, our protocol is secure; next we examine what efficiency it can achieve. Note that while for $n = 2$, we create a single key $\mathcal{S}$ with some efficiency E, for $n \geq 3$, the efficiency is different for each secret $\mathcal{S}_{ij}$, and depends on the erasure probabilities δ_{ri}, δ_{rj}, and δ_{rE}. In our notation, the efficiency simply corresponds to the ratio

$$E_{ij} = \frac{M_{ij}}{Nn}.$$

To calculate it, we need M_{ij}, to count how many packets a queue contains that Eve (or eavesdropping terminals) have not received. Over the theoretical network conditions, we can estimate M_{ij} using expected values. Lemma 8 in the Appendix provides concentration results showing that our estimation error becomes zero exponentially fast in the number of packets N. Lemma 2 provides such an example calculation.

Lemma 2. *If the theoretical network conditions hold, and we assume non-colluding eavesdroppers, then there exists a sufficiently large N for which the basic protocol achieves:*

- $n = 2$ *terminals,* $E = \delta_E(1-\delta)$,
- $n \geq 3$, *if* $\delta_1 \leq \delta_{ij} \leq \delta_2 \ \forall\ i, j$ *and* $\delta_E = \min_i \delta_{iE}$,

$$E_{ij} \geq \min\left\{\delta_E(1-\delta_2)\left[(1-\delta_2)+\frac{2\delta_2}{n}\right], \delta_1(1-\delta_2)\left[(1-\delta_2)-\frac{1-3\delta_2}{n}\right]\right\}.$$

This lemma verifies an intuitive fact: as the number of terminals (and transmission rounds in the initial phase) n increases, what dominates the size of each queue is the number of packets $(1-\delta_2)^2 N$ jointly overhead by two terminals; the fraction of these (δ_1 or δ_E) that is unknown to our strongest eavesdropper equals the amount of secrecy we can create. In other words, the fact that we keep adding x-packets in each queue during all rounds is the key in the protocol's scalability.

Lemma 3. *Under the conditions of Lemma 2, for $n = 2$ terminals, the basic protocol achieves maximum efficiency.*

Indeed, the efficiency we achieve for $n = 2$ reaches Maurer's upper bound [67].

The basic protocol scales well with the number of terminals because we try to leverage broadcasting as much as possible. If we were, instead, attempting pairwise secret establishment, the efficiency would quickly go to 0 with the number of terminals. To see this, consider the following, conceptually simpler alternative to the basic protocol: Consider a time-division protocol, where we operate in time-slots, and at each time-slot we create the key $\mathcal{S}_{ij}$ between a specific terminal pair, using the best possible protocol that achieves efficiency $\delta_E(1-\delta)$ [67]. Since we have $\binom{n}{2}$ keys to create, and each key is created during only one time-slot, the overall efficiency is $E^{(\text{alt})} = \frac{\delta_E(1-\delta)}{\binom{n}{2}}$ per key. Unlike the efficiency of our protocol that converges to a constant value as n increases, $E^{(\text{alt})}$ goes to zero.

Finally, the most demanding operations a terminal needs to perform is linear combining to create the y-packets. Thus:

Lemma 4. *Each terminal that participates in the basic protocol executes an algorithm that is polynomial in N and n.*

2.5.2 Multi-hop Networks

In contrast to the single-hop network scenario, where the erasure channel model is well defined and allows us to do closed-form computations, the multi-hop wireless networks do not offer this opportunity. The existence of correlated events and non-independent conditions make the task of upper-bounding Eve's reception capabilities very difficult. In fact, the probability of Eve (or any other terminal) receiving an x-packet depends on a multitude of effects, e.g., the various channel erasure probabilities, the probability of collision, the probability of the x-packet traveling though a specific path etc. This fact hinders us from calculating closed-form expressions about the efficiency achieved by our protocol, in the case of an arbitrary, multi-hop network.

Nevertheless, the property of information-theoretic security of our protocol holds also for the case of multi-hop networks, provided that an upper bound on the information that Eve can receive exists. Recall that (see Eq. 2.1) terminals T_i / T_j can construct up to M_{ij} y-packets that are information-theoretically secure (following Lemma 6 in Appendix A.1), and serve as their common secret $\mathcal{S}_{ij}$, provided they know (1) V_l, i.e., the number of x-packets they share and were missed by terminal T_l, and (2) V_E, i.e., the number of x-packets they share and were missed by Eve. The value of V_l can be precisely computed, given that each terminal announces the x-packets it has received during the feedback phase. A lower bound for V_E can also be precisely computed, provided that an upper bound on Eve's reception capability exists and is known to terminals T_i / T_j.

2.6 Adapting to Real Networks

In this section, we adapt our secret-agreement protocols to the scenario where the theoretical network conditions (as defined in section 2.2) do not hold, and an upper bound of how much information Eve knows is not known (for the reasons explained in section 2.5.2).

The challenge with real networks is that we do not know the size of the pairwise secrets (the M_{ij} from section 2.3.2) that we should create. In section 2.3.2, we were able to analytically compute M_{ij} because we assumed that we knew enough about Eve's reception capabilities to compute the expected amount of information missed by Eve, but in a real wireless network this knowledge cannot be assumed with certainty. Instead, we try in practice to conservatively estimate the amount of information missed by Eve based on the amount of information missed by the terminals.

Main Idea

In the case of single-hop networks and the basic protocol we think as follows: Alice and Bob assume that, during each round of the initial phase, Eve learns as much information as any of the other terminals about the x-packets shared by Alice/Bob. Hence, at the end of the initial phase, Eve is assumed to know at least as many of the Alice/Bob shared x-packets as the most knowledgeable terminal.

We chose this based on the following observations: Channel behavior varies significantly over time, to the point where we cannot estimate or even upper-bound how much information Eve collects during one experiment based on how much information she collected during past experiments. Channel behavior also varies over *space*, but less so: if, during an experiment, terminal T_i receives many packets in common with neighbor T_j, then T_i most likely receives many packets in common with its other neighbors as well. It turns out that, by measuring how many packets each pair of neighboring terminals receive in common during one experiment, we can estimate quite accurately how many packets any terminal and Eve receive in common *in the same experiment*. This, of course, is an empirical estimation, thus we cannot guarantee its accuracy theoretically.

In the case of multi-hop networks our intuition is that the fraction of packets, out of the packets shared between a pair T_i/T_j, that was overheard by Eve, depends on how "far from each other" the pair of nodes are: nodes that are further apart may collect less common packets; yet among the packets they collect, Eve is likely to have overheard a smaller amount, since she would not intercept the transmissions in all paths that connect them. In addition, Eve will aim to position herself inside the network so as to maximize her probability of eavesdropping as many paths as possible, i.e., a position through which the majority of the available paths pass by.

Estimating Eve's Knowledge

Single-hop networks: T_i and T_j estimate that, at the end of the initial phase, from their shared x-packets, Eve misses the following number:

$$V_E = \sum_{r \neq i,j,\, r=1}^{n} \min\{ V_1^r, V_2^r, \ldots V_n^r \}, \tag{2.3}$$

where V_l^r is the number of new x-packets shared by terminals T_i / T_j and missed by terminal T_l during round r of the initial phase.

In short, we assume that, in each round of the initial phase, Eve missed as few (of the x-packets newly shared by T_i / T_j in this round) as any other terminal.

Multi-hop networks: We form a set $\mathcal{L}$ of the ℓ nodes that have the largest number of neighbors. Let k_{ij} be the distance between nodes T_i / T_j in hops and let $\mathcal{P}(k_{ij})$ denote the set of all pairs of terminals in the network with the same distance k_{ij}. Then:

$$V_E = \underset{\mathcal{P}(k_{ij})}{\text{avg}} \min_{\ell \in \mathcal{L}} \{ V_1^p, V_2^p, \ldots V_l^p \}, \tag{2.4}$$

where V_l^p is the number of x-packets shared by pair p, with $p \in \mathcal{P}(k_{ij})$, and missed by helper terminal T_l, and $\text{avg}_{\mathcal{P}(k_{ij})}$ denotes average taken over the set $\mathcal{P}(k_{ij})$.

In the above formula we select the ℓ nodes with most neighbors to be conservative (note that the larger the $\mathcal{L}$ the more conservative we are); we also calculate the average taken over all pairs with distance k_{ij}, because a similar behavior is expected from pairs at the same distance.

Key Points

If we do not assume theoretical network conditions, in the case of singe-hop, or the existence of an upper-bound of how much Eve learns, in the case of multi-hop, we cannot offer formal guarantees about the reliability of our protocol, because we do not know exactly how much information Eve collects during the initial phase (resp. the packet dissemination phase): it is theoretically possible that Eve receives more x-packets in common with the terminals than we estimate, which means that she learns something about the pairwise secrets. The amount of information that leaks to Eve depends both on the particular wireless network and the number of terminals we use for our estimations: the more terminals we use, the more we learn about the quality of receptions throughout the network, and the better we can estimate the quality of Eve's receptions capabilities. Hence, the amount of information that leaks to Eve needs to be experimentally assessed in each wireless network.

2.7 Experimental Evaluation

In this section, we experimentally evaluate our adapted secret-agreement protocols (section 2.6). Our goal is to answer two questions: is it feasible to achieve non-negligible secrecy rate in realistic wireless setups by leveraging packet erasures? And how well can we do so using our protocols?

2.7.1 Single-hop Networks

Testbed

We show our testbed in Figure 2.1. It consists of 6 nodes distributed over an indoor office area. Each office is about 2×3 meters. Unless otherwise specified, the nodes are HTC Wildfire Android smartphones. We set the phones to 802.11 ad-hoc mode, and we fixed their transmission rate to 36 Mbps. In some experiments, we also use WARP software-defined radios [10].

In order for our approach to work, the wireless network must provide a certain level of channel variability. The simplest scenario where such variability exists is when the nodes are not in direct line of sight, e.g., they are separated by office walls. This is the scenario we implement in our testbed. Our protocol can work even when the nodes are in direct line of sight, but for that we need to use artificial noise (the terminals create interference and force Eve to miss some of the traffic they exchange). We have experimented with that idea [85], but we do not consider this approach here.

When we refer to an "experiment," we mean that we place one node in each room, and we run one round of our protocol. In each experiment, one node plays the role of Eve, while the rest play the role of 5 terminals that exchange pairwise secrets. There are 6 possible arrangements of 5 terminals and Eve in 6 rooms, and we experiment with 3 different levels of transmit power. Hence, each presented graph summarizes the results of 3×6 experiments (all the combinations of transmit-power levels and node arrangements). For each transmit power level, we use a box plot as a convenient way of graphically depicting different groups of our measurements through their percentiles (we used matlab's `boxplot` function [66]): On each box, the central horizontal line is the median, i.e., half of the measurements are below that level and the other half is above. The lower edge of the box is the 25th percentile (splits off the lowest 25% of measurements from the highest 75%) and the upper edge is the 75th percentile (splits off the lowest 75% of measurements from the highest 25%). The whiskers extend to the most extreme measurements not considered outliers[6], and outliers are plotted individually and marked as +.

[6] In matlab's implementation, points are considered as outliers if they are larger than $q_3 + w(q_3 - q_1)$ or smaller than $q_1 - w(q_3 - q_1)$, where q_1 and q_3 are the 25th and 75th percentiles, respectively. The default value of w is 1.5 and corresponds to approximately $\pm 2.7\sigma$ and 99.3 coverage if the data are normally distributed.

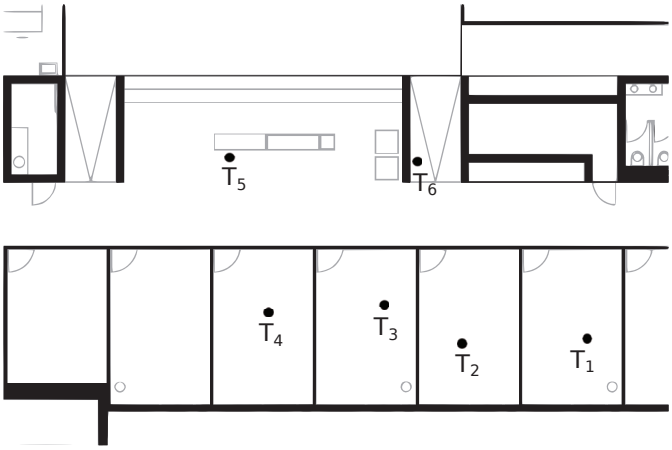


Figure 2.1 – Our 1-hop wireless testbed.

Ideal Secrecy Rate

We start with the ideal efficiency and secrecy rate achievable in this testbed by leveraging packet erasures. In particular, we measure the efficiency and secrecy rate of an *oracle-assisted* protocol; this works like ours, with the only difference that it does not estimate how much information Eve obtains in the initial phase – that knowledge is directly provided by the oracle. More specifically, instead of estimating V_E using Equation 2.3, we set it to the exact number of x-packets shared by terminals T_i/T_j and missed by Eve. This oracle-assisted protocol by construction achieves reliability 1, because it knows exactly how much information Eve obtains in the initial phase and computes the longest secret that is completely unknown to Eve. In Figure 2.2 ("Ideal" label) we plot the efficiency/secrecy rate achieved by any terminal pair in any experiment, using the oracle-assisted protocol, as a function of the transmit power of the terminals.

First, we see that, if we perfectly knew Eve's channel conditions, *all* terminal pairs could create tens of thousands of secret bits per second, of which Eve would have zero information independently from her computational capabilities. This shows that a real wireless network may offer enough channel variability to enable secret generation in non-negligible rates.

Second, we observe a variability regarding the secrecy rates achieved by different pairs of terminals, which reduces as the transmit power increases. This is because for low transmit powers the difference in physical distance between terminal pairs has a greater impact on the terminals' channel qualities than for high powers; as the transmit power increases the channel noise affects in a similar way the terminals' channel qualities, despite their differences in physical distance.

Third, we see that the secrecy rate drops as the transmit power of the terminals increases. This is due to the following reason: As the transmit power of a terminal increases, so does the quality of its channels to both the other terminals and Eve. Hence, higher transmit power

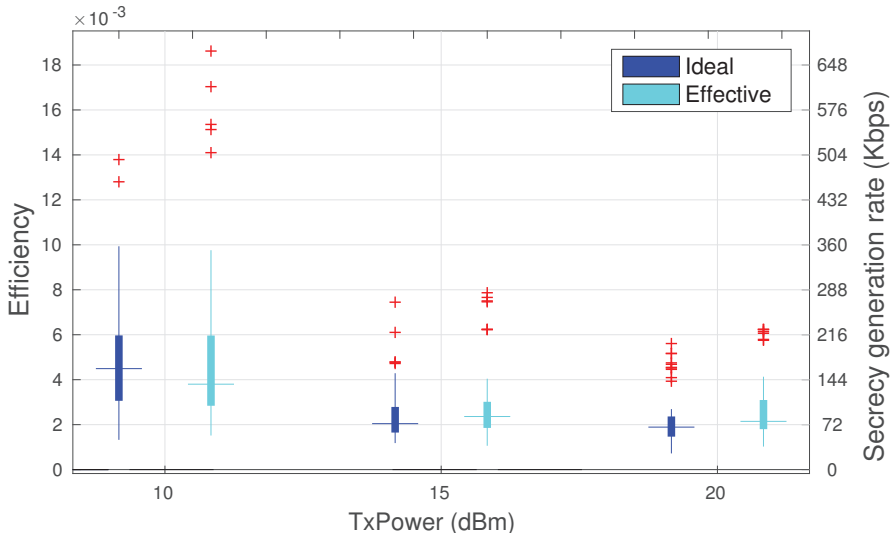


Figure 2.2 – Efficiency and secrecy rate as a function of TX power. “Ideal” corresponds to the oracle-assisted protocol and “Effective” to our protocol.

means that the terminals receive correctly more packets, but also that Eve overhears more of their packets, decreasing, thus, their secrecy rate.

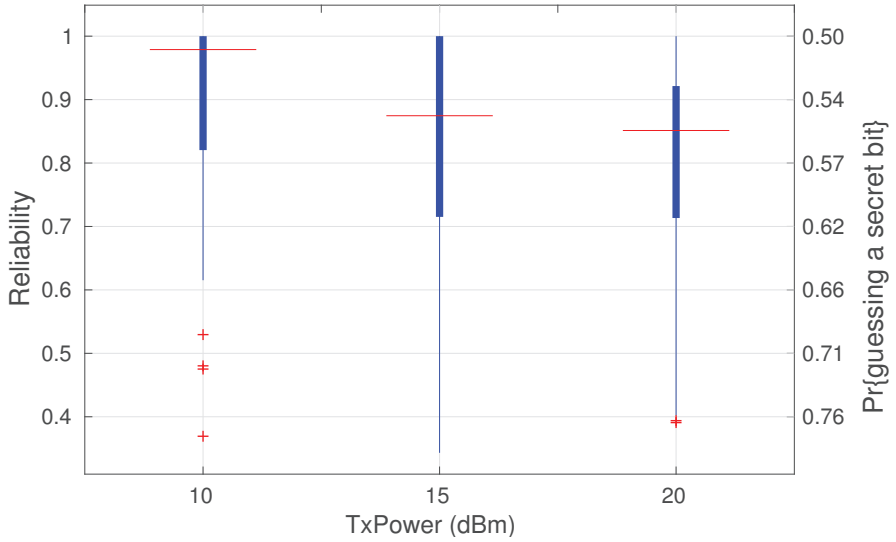


Figure 2.3 – Reliability of our protocol as a function of TX power.

Reliability and Secrecy Rate of our Protocol

Next, we look at the performance of our protocol. Unlike the oracle-assisted protocol, ours needs to estimate how much information Eve obtains in the initial phase. If it overestimates Eve’s knowledge, it creates a shorter secret than it could, achieving lower efficiency/secrecy rate than the oracle-assisted protocol. If it underestimates Eve’s knowledge, it creates a longer

secret than it should, achieving higher secrecy rate than the oracle-assisted protocol, but reliability below 1. Hence, there is a trade-off between secrecy rate (how fast we create new secrets) and reliability (how secure these secrets are).

Ideally, we would want our protocol to behave like the oracle-assisted one (achieve the same secrecy rate and reliability 1). In practice, this is infeasible, as it would require us to always estimate Eve's knowledge with perfect accuracy. Thankfully, it is also unnecessary: Suppose a secret has reliability 0.6, which means that Eve can correctly guess the value of one bit of the secret with probability $2^{-0.6} = 0.66$. The smallest secret that our protocol ever creates is one y-packet (1 KB); reliability 0.6 means that Eve can correctly guess the value of one y-packet with probability $2^{-0.6 \cdot 8000} \approx 0$. Hence, as long as the terminals use their pairwise secrets at the granularity of a y-packet (e.g., they use at least one entire y-packet as an encryption key), they are secure from Eve.

Figure 2.2 ("Effective" label) and Figure 2.3 show the efficiency/secrecy rate and the reliability of our protocol, as a function of the transmit power of the terminals. We see that, using our estimations, we can closely follow the behavior of Eve. Although we tend to slightly underestimate Eve's knowledge as the transmit power increases, on average the secrets we create have reliability above 0.8. This shows that, in a real wireless network, it may be feasible to accurately estimate an adversary's knowledge, if we have a sufficiently dense deployment of collaborating honest nodes. Of course, this estimation will become harder as we consider adversaries with increasingly more sophisticated hardware (e.g., multiple receiving antennas).

Finally, we note that we could increase further the reliability of our secrets by decreasing M_{ij}, the number of y-packets that a terminal pair T_i / T_j can construct, by a constant factor ϕ. This would, of course decrease the efficiency/rate achieved by the terminals by the same factor. For example, we used $\phi = 2$ in our experiments, which translated to half the rate reported in Figure 2.2, but also to a reliability almost (with very few outliers close to 0.7) at 1 for all the produced secrets S_{ij}.

Understanding Erasures

In the above experiments, Eve uses a commodity device, i.e., a smartphone, to eavesdrop the communication medium; she, therefore, gains knowledge only from the information that is successfully delivered to her application layer. There exist packets that reach Eve's receiver, yet are not delivered to her application because they are corrupted beyond what the lower layers can repair. One could argue that, if Eve rooted her phone and gained access to every packet that reaches her physical layer (even the partially corrupted ones), she would improve her knowledge.

To investigate how much Eve's knowledge could be improved, we used three WARP software-defined radios[7], configured with an 802.11-compliant physical layer (16 QAM over OFDM),

[7] The smartphones used for our earlier experiments do not provide access to received data that is discarded

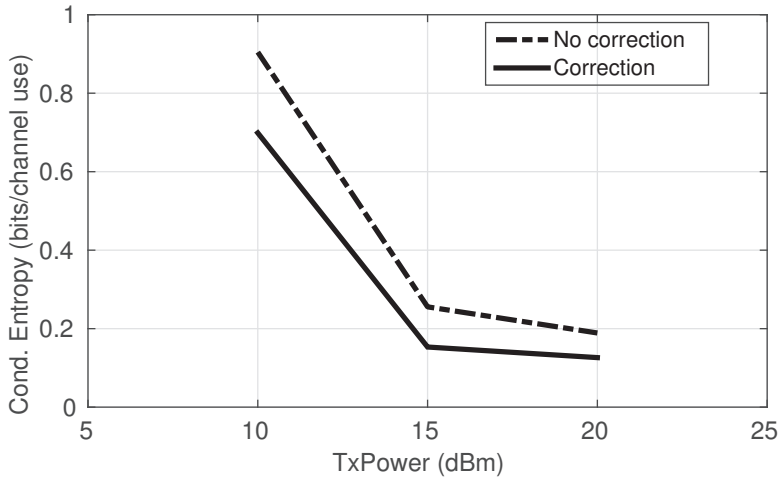


Figure 2.4 – Eve's conditional entropy (in bits per channel use) when she has access to the packets that reach her receiver. In our setup, one "channel use" means sending one 16 QAM symbol. "Correction" corresponds to the case where an oracle corrects all corrupted packets that reach Eve.

and we placed them in our testbed. We make one of them (Alice) send out traffic, while the other two (Bob and Eve) receive. The difference from our earlier experiments is that now Eve tries to use *all* the packets that reach her physical layer (every correctly received packet but also every packet with partially corrupted payload – that would be normally dropped below the application layer, if Eve was using a smartphone) to increase her knowledge. First, we consider the packets that are correctly received by Bob, and we measure Eve's knowledge (conditional entropy) about these packets[8]. Then we repeat the experiment, assuming that an oracle magically repairs the corrupted payload of every packet that reaches Eve's receiver. In the former case, Eve's uncertainty on Bob's information originates from both corrupted and erased symbols – this is equivalent to Eve using a smartphone, whereas in the latter only from erased ones (that do not reach Eve's receiver at all) – this is equivalent to Eve using a specialized radio receiver and to gaining the maximum possible knowledge out of partially corrupted packets.

Figure 2.4 shows that – at least in our testbed – Eve's uncertainty mostly depends on the erased in-the-air symbols, i.e., symbols that were not demodulated at all. We observe that, if Alice uses transmit powers of 10, 15 and 20 dBm, in the second experiment (where all payload corruption is corrected by the oracle), Eve learns only an extra 0.2, 0.15 and 0.08 bit-per-channel-use, respectively, relative to the first experiment. This indicates that the number of partially corrupted packets that reach Eve's receiver is relatively small, hence they do not significantly increase Eve's knowledge (or reduce the secrecy rate achieved by our protocol).

below the application layer.

[8] We do so by calculating the joint empirical distribution of the 16-QAM symbols in the payload of Bob's packets and the symbols that Eve receives – correctly or not.

2.7.2 Multi-hop Networks

Simulation Environment

We use the Java-based, discrete event-driven simulator JiST [18], along with the SWANS library [19], that builds on top of JiST and provides all the elements needed to simulate ad-hoc wireless networks. We also used the extensions and bug-fixes proposed in [57]. In Table 2.4 we summarize the configuration parameters of the simulation setup. We use an IEEE 802.11b/g compliant MAC configuration and an SNR frame reception model with an SNR threshold value appropriate for high data rates [79]. The RTS/CTS functionality is by default disabled.

The signal interference model used in the JiST/SWANS simulator is equivalent to the physical model of successful receptions as defined by Gupta *et al.* in [48]. This feature enables to simulate the hidden-terminal effect and exploit collisions and frame erasures for secrecy.

We simulate a wireless ad-hoc network as a set of n nodes uniformly at random placed on a square area of dimension x meters. All nodes have the same communication capabilities that yield a transmission range of r meters. Under the configuration parameters described above $r \approx 200m$. Therefore, for a k-hop network we set $x = k * \frac{r}{\sqrt{2}}$. We consider networks with fixed network density per unit area, that is, for a k-hop area and a given density d we have in total $n = k^2 * d$ nodes.

In our protocol, we set $ttl = k$, the maximum distance in the network, and the packet payload to 1KB, so that the resulting MAC frame (including the necessary headers of our protocol and of other layers) does not get fragmented. We also position Eve in each configuration to be in the network center, where we verified that she would have the highest probability to overhear the largest amount of packets. We also verified that the simulator produces very similar results, in the case of a single-hop network, to these produced in our testbed.

Ideal Secrecy Rate

As in the case of single-hop, we measure the efficiency/secrecy generation rate achieved by the oracle-assisted protocol. Figures 2.5a, 2.6a, 2.7a (label "Ideal") show the efficiency/rate achieved by the oracle alternative, over k-hop networks, with $k = 1 \ldots 5$, and for network densities $d = 10, 15, 20$, respectively.

First, we observed that in all cases we simulated, we could generate non-zero rates across (almost) all pairs in the network. Notably, we observed that in all our simulations, only 56 pairs of nodes in total experienced zero rate (in particular configurations of 500 nodes, where in each configuration there exist 124750 possible pairs). Second, for every density, we observe that, as the size of the network increases, namely for $k \geq 3$, the rate significantly drops. This is the aggregated result of two conflicting effects: (1) to create shared randomness over a k-hop network, each packet needs to be transmitted at least k times, which correspondingly reduces the rate; moreover the amount of common packets that a pair of terminals collects during

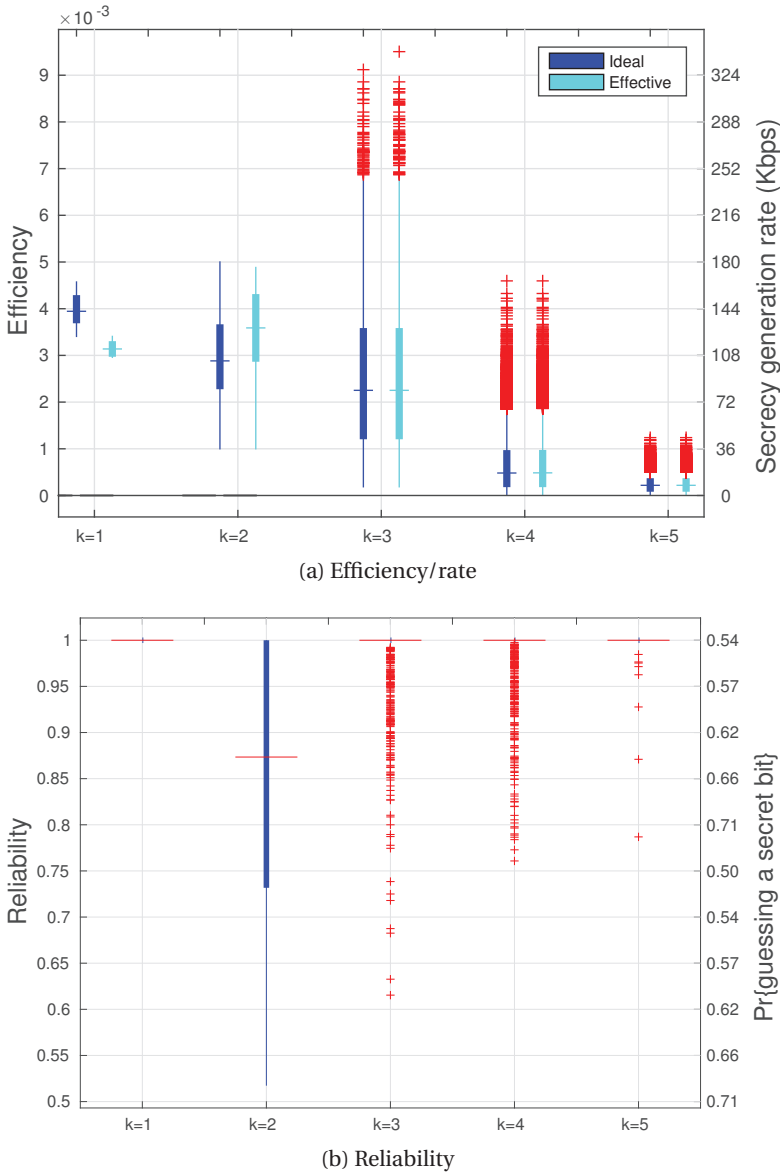
MAC Layer	Slot Time	$20\mu s$
	W_{min}	31 slots
	W_{max}	1023 slots
	SIFS	$10\mu s$
	DIFS	50 μs
	PHY header	192 bits
	MAC header	272 bits
	DATA frame header	464 bits
	ACK frame	304 bits
PHY Layer	Frequency	2.4 GHz
	Basic Rate	1 Mbps
	Data Rate	36 Mbps
	Tx Power	15 dBm
	Sensitivity Threshold	-81 dBm
	Reception Threshold	-71 dBm
	Reception Model	SNR
	SNR Threshold	15 dB
Channel Model	Propagation Model	TwoRay
	Fading Model	Rayleigh
	Interference Model	AdditiveNoise

Table 2.4 – Configuration of simulation setup.

the packet dissemination phase is smaller, because a smaller percentage of the generated packets reaches both, which in turn reduces the rate; (2) due to the existence of interference and multiple paths between two terminals in larger networks, Eve observes a smaller fraction of the common random packets that both terminals collect, which boosts the rate. We verified these effects in our simulations; we show here in Fig. 2.8 the second effect: we examine what percentage of packets shared between two nodes Eve has also observed (on average), and we find that this percentage decreases with the network size. Finally, for the 2-hop network, we observe that as the density increases, the rate also increases; this is because we have more nodes acting as sources, thus creating more interference and hindering Eve from collecting the same packets as her close neighbors. The existence of multiple paths, boosts further the rate, as demonstrated by the rates for a 2-hop network when comparing to the rate for an 1-hop network, for high network densities: the more nodes we have the more probable is that two nodes are connected through more than one paths, out of which Eve does not observe at least one.

Reliability and Secrecy Rate of our Protocol

Fig. 2.5a, 2.6a, 2.7a (label “Effective”) demonstrate the efficiency/rate achieved by our protocol, and Fig. 2.5b, 2.6b, 2.7b the corresponding achieved reliability. In contrast to the oracle-assisted protocol, in our protocol the terminals need to estimate how many packets Eve



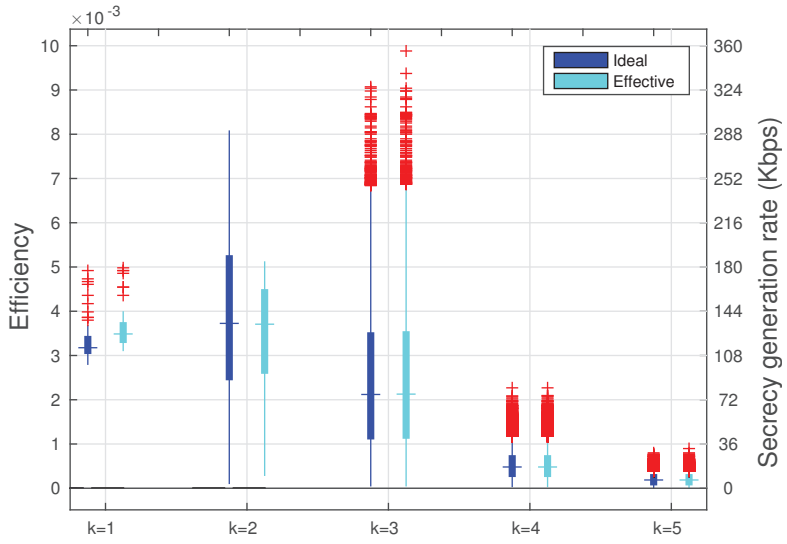
(a) Efficiency/rate

(b) Reliability

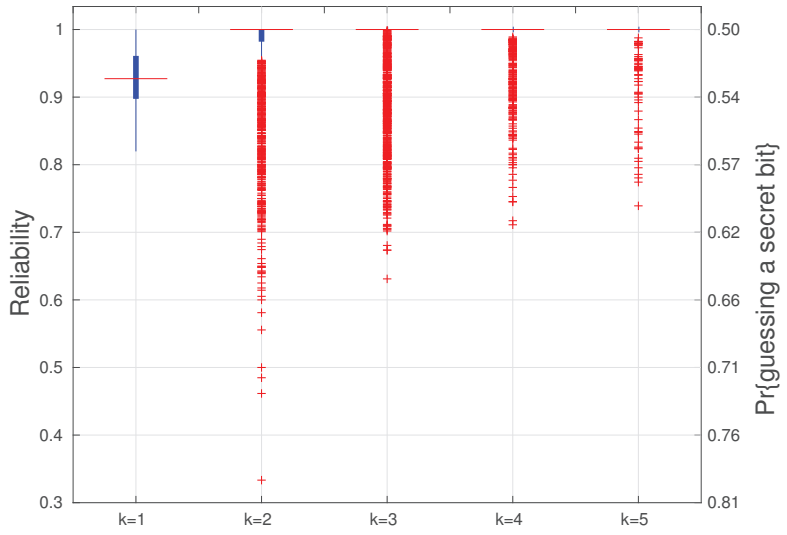
Figure 2.5 – Measurements for network density $d = 10$, over arbitrary k-hop networks.

misses, using the technique in section 2.6.

We observe that our protocol can closely follow the oracle-assisted protocol's performance, i.e., our estimator yields rather accurate estimations on Eve's knowledge. In some cases, namely for small densities and small networks, i.e., $k \leq 2$, we underestimate Eve's knowledge, which yields reliability values around 0.7. Despite this, we observe that as the network increases in density and size, the terminals compute very good estimations; this is of course due to the fact that the more terminals there exist, the more side information on packet receptions is available, the easier it becomes to accurately estimate a terminal's (Eve's) behavior.



(a) Efficiency/rate

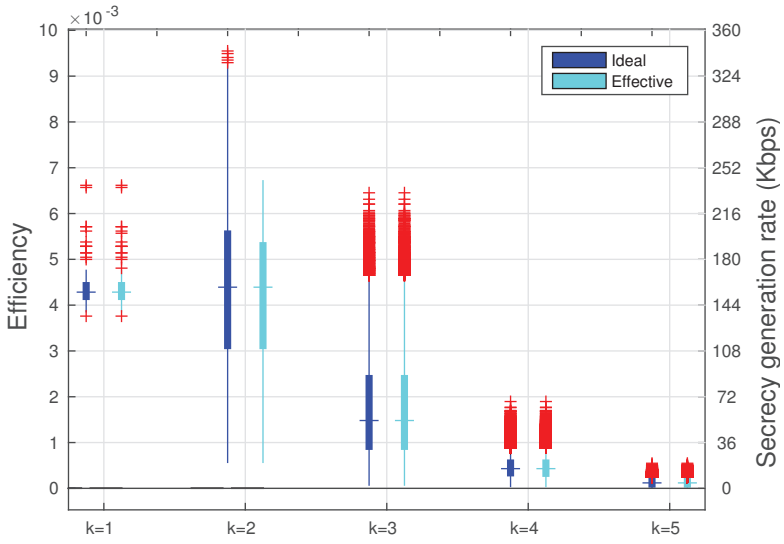


(b) Reliability

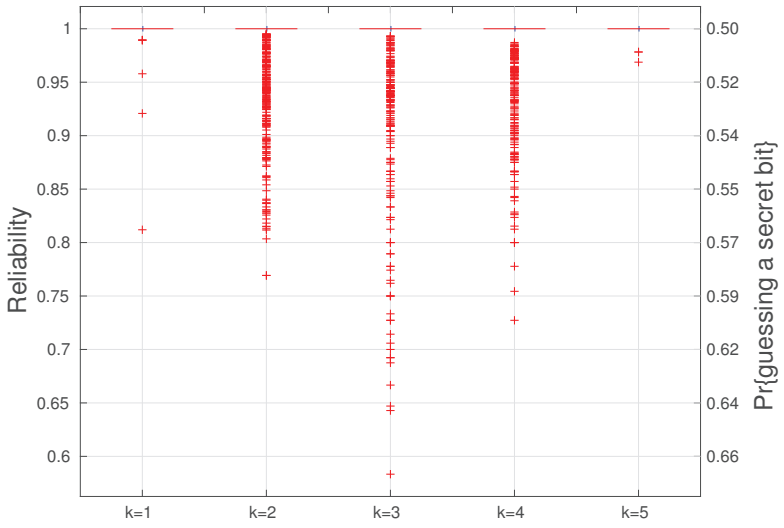
Figure 2.6 – Measurements for network density $d = 15$, over arbitrary k-hop networks.

2.8 Discussion

A main assumption we do is that Eve is a *passive* adversary. In the case that Eve is an *active* adversary (tries to impersonate a terminal), the terminals need to share some bootstrap information to authenticate each other when they first communicate. The need for this bootstrap information is fundamentally unavoidable: without it, there is no way for Alice to know she is talking to Bob until they have established their first secret. Authentication is orthogonal to our secret agreement and can happen in different ways, e.g., by requiring the terminals to initially share bootstrap information and use it to construct authentication codes for the x-packets (and the feedback packets) they transmit the first time they run our protocols.



(a) Efficiency/rate



(b) Reliability

Figure 2.7 – Measurements for network density $d = 20$, over arbitrary k-hop networks.

After the terminals have established their first pairwise secrets using our protocols, they can use these to construct new authentication codes, which do not depend on the bootstrap information.

One might argue: if the terminals have to share bootstrap information anyway to defend against active adversaries, they might as well share pairwise secrets to begin with and not run our protocol at all. The advantage of our protocols is that they enable the terminals to keep generating *new* secrets, independent from the previous ones, and continuously refresh their encryption and authentication keys. Unless the adversary can break into one of the terminals *while they run our protocols,* she has a small window of opportunity to compromise their

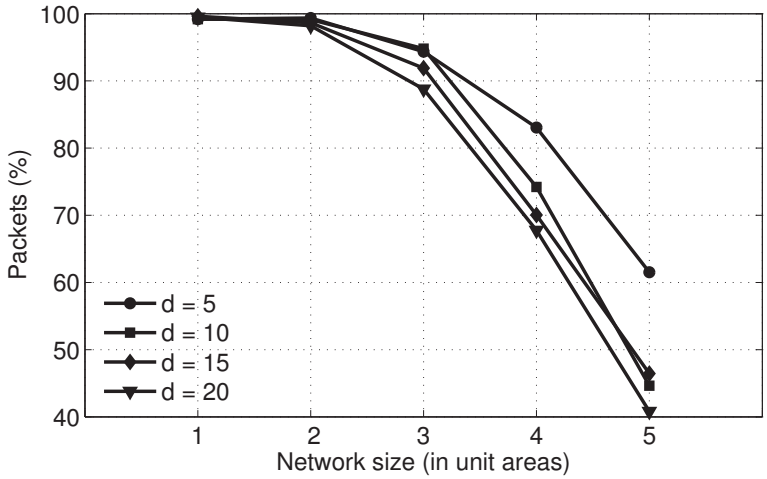


Figure 2.8 – Eve's knowledge on shared packets.

communication: she has to steal the bootstrap information and impersonate a terminal *while the terminals are running our protocols for the first time.*

2.9 Related Work

Existing information theoretical results characterize the largest achievable secrecy rate under a variety of idealized channel models [98, 67, 37, 39]. The most common setting considers pairwise secret key generation over a single channel with a single sender and one or more receivers. Some results are available for a network setting, most notably secure network coding for an error-free wired network [24]. The secrecy capacity of wireless erasure networks is investigated in [69], but no complete characterization is provided. A rich literature exists in designing practical codes for achieving the theoretical secrecy bounds (see [50] and references therein), but the proposed schemes typically aim in providing weak information theoretic security and in single message delivery (e.g., [14, 74, 49, 56]). Coding for strong secrecy usually yields low achievable rates and builds on the fact that Eve has a degraded channel compared to the legitimate nodes [50].

Several practical protocols were recently also proposed that build on the symmetry and the randomness extracted from the wireless channel to set up strong information theoretically secure pairwise keys [100, 62, 80, 17, 99]. These achieve modest secret-generation rates (in modified 802.11 or 802.15 environments) and require node proximity, hence, they do not naturally translate to multi-hop networks/multiple keys creation. iJam [43] utilizes artificial interference (specific to OFDM) to increase Eve's uncertainty and it achieves a secret-generation rate up to 18 Kbps (in a modified 802.11).

We differ in the following ways: to the best of our knowledge, our work is the first to consider

multi-terminal pairwise secret-agreement, where broadcast is leveraged to efficiently create multiple shared secrets at the same time. The existing protocols focus on a single pair of nodes, hence they are not designed to leverage broadcast, and they would not scale well with the number of terminals (if applied to the multi-terminal scenario). Moreover, our protocols achieve a secret-generation rate of tens of Kbps, without requiring any custom physical-layer operations that are specific to OFDM (or any other transmission scheme). More importantly, as noted earlier, the extension for a multi-hop network requires new techniques and also brings new secrecy opportunities. To our best knowledge the current work is the first to develop protocols for secret key exchange in a multi-hop network that simultaneously exploits channel and network properties, and to report secrecy rates of Kbps, through experimentation in realistic wireless setups.

2.10 Summary

In this chapter we presented two protocols for enabling a group of n wireless nodes to create pairwise secrets, in the presence of a passive adversary, with limited network presence, without assuming anything about her computational and memory capabilities. Our basic secret-agreement protocol operates in single-hop networks, it is information-theoretically secure and leverages broadcast to create secrets simultaneously between all terminal pairs. Our protocol for arbitrary, multi-hop networks, builds on the basic protocol and includes new designs, e.g., a custom packet dissemination protocol, to leverage the benefits of multi-hop for secrecy generation.

On the practical side, we evaluated our protocols through testbed experimentation and extensive simulations, and we showed that it is feasible to generate secrets at non-negligible rates, both on single-hop and multi-hop networks.

In summary, the contributions in this chapter are:

1. We design practical secret-agreement protocols for simultaneously generating $\binom{n}{2}$ secrets in:
   a) single-hop networks, by leveraging channel properties,
   b) arbitrary multi-hop networks, by leveraging both channel and network properties.
2. We evaluate the performance of our protocols through experimentation in realistic wireless environments.

3 A Tor-like Traffic Anonymization Scheme

3.1 Introduction

The Tor anonymity network [38] is an overlay network that combines Onion Routing with a lightweight system design for Internet traffic anonymization, and it is rapidly becoming the prevalent approach to anonymity today. In the core of the Tor system, basic cryptographic primitives are used, e.g., the Diffie-Hellman key-agreement, RSA and AES encryption. In this chapter we aim in designing an alternative, Tor-like communication scheme for wireless networks, that offers a level of anonymity comparable to the level of anonymity that Tor does, yet without assuming anything about the computational and memory capabilities of an adversary, who is trying to de-anonymize the observed traffic in the network.

Similarly to the Tor anonymity approach, our goal is to enable nodes to connect to the Internet, while hiding their identity within a set of potential users. Tor achieves anonymity by bouncing encrypted communications around a distributed network of relays; we similarly bounce encrypted communications among the wireless network nodes. In our use-case scenario, an information packet travels from the source node along a randomly selected path towards a final hop to the Internet. We use layered, one-time pad encryption to both secure the messages against eavesdropping, and ensure that each relay along the path is aware of only a fraction of the entire communication path, in a fashion similar to Tor.

The main contribution of this chapter is in the design of a traffic anonymization protocol, that exploits the security properties of the shared secrets, which we can generate with the techniques described in chapter 2. Our privacy analysis demonstrates that we can achieve a Tor-like level of anonymity and our experimental evaluation shows that we can achieve almost perfect anonymity within a group of approximately half the network size. We note that the traffic anonymization protocol presented in this chapter is not bounded to the specific secret-generation technique we described in chapter 2. Any secret-agreement procedure, that enables nodes in a multi-hop wireless network to establish secure pairwise and group keys, under the presence of an adversary, would serve as the base of our traffic anonymization protocol.

The work presented in this chapter has been presented in [83].

3.2 Setup and Background

System and Adversary Model

We consider a network of n wireless nodes that form a k-hop ad-hoc network, where k refers to the maximum distance (in hops) between any two nodes. From the nature of wireless, each node's transmission can potentially be received by its neighbors, i.e., all nodes within its transmission radius. We assume that every node has a unique identifier that is revealed to all other nodes in the network. We also assume that every node is an honest-but-curious node: it legitimately participates in the protocols used, but tries to breach security using the information at its disposal.

In the network there exists also a passive adversary, Eve, who eavesdrops but does not reveal her presence with any form of communication, and can be located anywhere inside the network, at an unknown location. We assume that Eve has access to the same physical layer (radio technology, number of antennas etc.) as the legitimate nodes, and is not omni-present in the network. However, we assume that Eve may have infinite memory as well as unbounded computational capabilities at her disposal; this would follow the model of an adversary that does not want to reveal her identity by using specialized equipment, yet has offline access to unbounded resources to breach security. In the following we will call the adversary Eve, without specifying (unless needed) if she is a passive eavesdropper or an honest-but-curious node.

Possible Use-case Scenario

As a use-case, consider a street protest, where participants use local communication (e.g. WiFi) to cooperate and hide the identity of someone who needs to use cellular Internet connectivity to send reports to the media (and thus might be a target for the authorities eavesdropping the local communication). In addition to eavesdropping, the authorities might interrogate any participant and force them to reveal their knowledge on the on-going communications. While Tor preserves anonymity as long as the cryptographic primitives used remain unbreakable, we aim, with our approach, to ensure anonymity even if the adversary has unlimited computational power.

The Basic Tor Operations

We here summarize the basic Tor [38] operations, without describing in full detail the whole system architecture; we rather focus on the key agreement procedure and the use of the keys for anonymous communication.

A node S wants to send a message m to a public destination D (e.g. a web-server) using the Tor anonymization network, i.e., a set of collaborating nodes, the so-called Onion Routers (OR), that will relay m toward its final destination. In a first phase, S negotiates a symmetric key with each relay. Assume S selects two nodes (the minimum required, assuming S is an OR as well) R_1, R_2, as shown in Fig. 3.1, and agrees on two symmetric keys with each one of them:

1. S sends to R_1 the first half of the Diffie-Hellman handshake g^{x_1}, encrypted with the public key $K^+_{R_1}$ of R_1. R_1 responds back with the other half of the handshake g^{y_1}, and a hash of the negotiated key (with $F(\cdot)$ denoting a secure hash function). S and R_1 compute the key $K_{SR_1} = g^{x_1 y_1}$.

2. S sends to R_1 the packet $K_{SR_1}\{R_2,\ K^+_{R_2}\{g^{x^2}\}\}$, that is a request to negotiate a symmetric key with R_2, encrypted with the key K_{SR_1} (128-AES encryption). R_1 and R_2 perform the same actions as S and R_1 respectively in step 1.

In a second phase, S communicates a message m to D by sending the packet $K_{SR_1}\{R_2,\ K_{SR_2}\{D, m\}\}$ to R_1, which extracts the first layer of encryption and forwards the inner packet $K_{SR_2}\{D, m\}$ to R_2; finally, R_2 extracts the second layer of encryption and sends m to D.

We note two fundamental properties of the Tor design:

- **Property 1:** R_1 cannot compute the key K_{SR_2}, since g^{x_2} is protected with the public key of R_2. It cannot, namely, decrypt the packet $K_{SR_2}\{D,\ m\}$ and reveal the message m and its final destination D.

- **Property 2:** R_2 does not know if it is setting up a symmetric key with R_1 or any other node in the network (in our example, node S). In other words, it does not know which is the originator of the packet $K_{SR_2}\{D,\ m\}$; from R_2's perspective the originator could be R_1, S or any other network node with equal probability.

These two properties ensure the basic premise of Tor: a relay knows only two nodes along the communication path, its predecessor and its successor, but cannot ultimately link S to D and m. Anonymous communication is, therefore, preserved under the presence an adversary Eve, who in this case has bounded computational power and cannot breach the security of the cryptographic primitives used.

Performance Metrics

The goal of our traffic anonymization protocol is to create uncertainty to Eve about the sender and the receiver of a given message m. Let $\mathcal{S}, \mathcal{D}$ denote the random variables that describes who the actual sender and receiver is, and E Eve's knowledge on the protocol and the produced traffic.

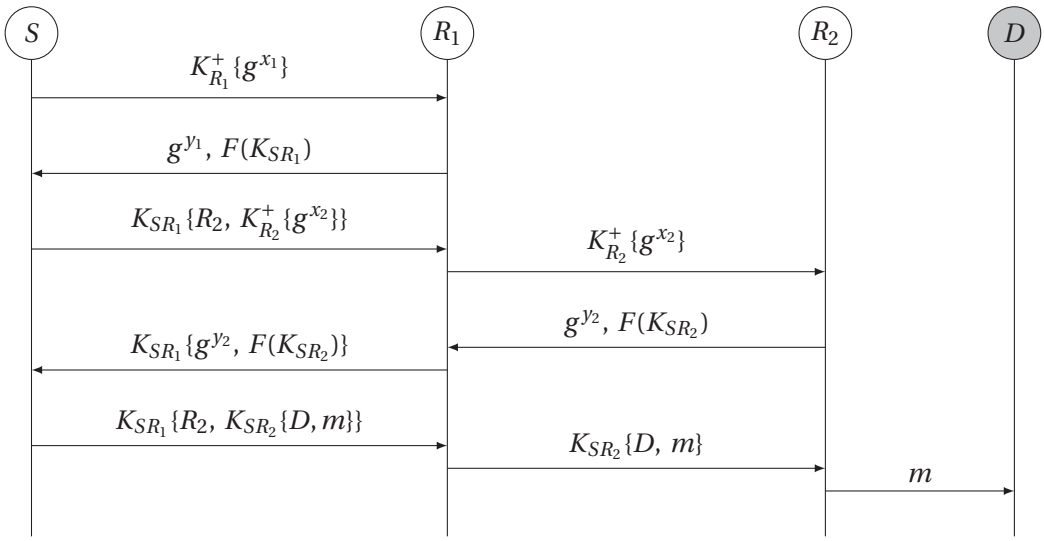


Figure 3.1 – Tor anonymization protocol – example.

- The *sender uncertainty* $U_{\mathcal{S}}$ and *destination uncertainty* $U_{\mathcal{D}}$ are measured as the conditional entropies:

$$U_{\mathcal{S}} = H(\mathcal{S}|E) \text{ and } U_{\mathcal{D}} = H(\mathcal{D}|E).$$

- The *sender-receiver uncertainty* expresses the uncertainty about the communication pair and equals

$$U_{\mathcal{S}-\mathcal{D}} = H(\mathcal{S}|E) + H(\mathcal{D}|E).$$

  $U_{\mathcal{S}-\mathcal{D}}$ gives the entropy of the joint distribution of $(\mathcal{S},\mathcal{D})$ in case the two random variables are independent from Eve's perspective. The maximum source uncertainty within a group would be achieved if Eve believes each group member to be the source with equal probability.

3.3 Traffic Anonymization Protocol

We here describe a communication scheme aiming to provide a level of anonymity that is comparable with the anonymity level of the Tor system [38], albeit also secure against computationally unbounded, but presence-limited, adversaries. The design of our protocol aims in satisfying the two fundamental Properties 1 and 2 of Tor, that we described in section 3.2.

The steps of negotiating the symmetric keys in Tor, are essentially replaced by the secret-generation protocol for multi-hop networks as described in chapter 2: In an initial step, legitimate nodes produce and transmit *random* packets; next, they publicly announce to each

other which packets they correctly received. In a second step, the nodes linearly combine their common packets to create keys, on request. We now describe how they can use these resources for anonymous communication, as depicted in Fig. 3.2.

Example

S wants to communicate message m to D:

1. S randomly selects two nodes R_1 and R_2 in the network that will act as relays. The message m will travel to D by following the path $S-R_1-R_2-D$, as depicted in Figure 3.2, where each link in this path is conceptual, i.e., the underlying connection may employ multiple hops.

2. S uses one-time pad encryption to send to R_1 the message m and the identities D and R_2 through the packet:

$$K_{SR_1}\{R_2, K_{G_2}\{D, m\}\} = K_{SR_1} \oplus \{R_2, K_{G_2} \oplus \{D, m\}\},$$

   where K_{SR_1} is a secure pairwise key between S and R_1 (we will call this link encryption), and K_{G_2} is a random packet that all nodes in a group G_2 have successfully received, with $\{S, R_2\} \subset G_2$ but $R_1 \notin G_2$, i.e., this packet is secret from R_1 (we will call this group encryption).

3. R_1, that has the pairwise key K_{SR_1}, removes it to find out that it needs to forward to R_2; it then re-encrypts using the pairwise key $K_{R_1R_2}$ and sends the packet:

$$K_{R_1R_2}\{K_{G_2}\{D,\ m\}\} = K_{R_1R_2} \oplus K_{G_2} \oplus \{D,\ m\}.$$

   R_1 does not possess K_{G_2} and thus does not learn D and m.

4. R_2 removes both $K_{R_1R_2}$ and K_{G_2}, and sends m to D; R_2 does not know that S originated message m, since it could have been any node in group G_2.

Key Points

The link and the group keys serve complimentary roles in ensuring anonymity. The role of the link keys, K_{SR_1} and $K_{R_1R_2}$, is to hide R_2, D, and m from intermediate relays as well as external eavesdroppers, similarly to the symmetric encryption in Tor. The role of the group key K_{G_2} is threefold. First, it hides the identity of the destination from R_1, who only learns the identity of the next relay R_2. Second, because it also hides the message m from R_1, even if R_1 overhears the unencrypted message m that R_2 transmits, it cannot link m to packet $K_{G_2}\{D, m\}$ and thus again will not learn the destination. Third, it hides the identity of the sender within the group G_2 for R_2, who only knows that $S \in G_2$. In other words, the role of the group keys is to provide

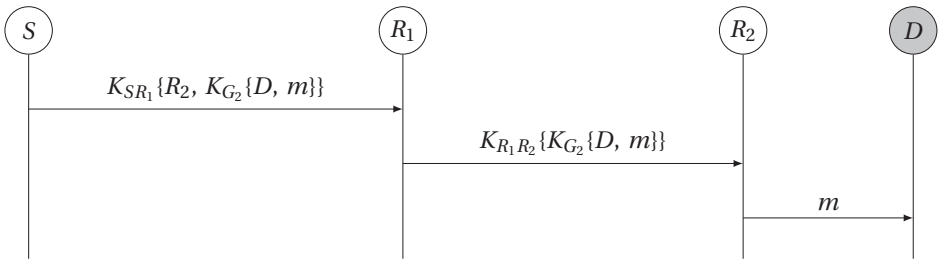


Figure 3.2 – Traffic anonymization protocol – example.

the basic anonymity property: each relay knows only its predecessor and its successor in the communication path; similarly to Tor.

How we have created keys has significant implications on the anonymity protocol we have designed. Our protocol essentially combines the layered (onion) encryption of Tor with one-time pad encryption. We can afford to use one-time encryption, exploiting the high key-generation rates of our secret-agreement protocol for multi-hop networks; S can randomly select R_1 and R_2 because we can create keys between all pairs of nodes; and because we distribute random packets to create shared randomness, we can easily find large sets G_2 that share common random packets (see section 3.5). The size of G_2 is important as it determines the amount of anonymity: the larger it is, the harder it is for the adversary to correctly guess the originator of a packet.

3.3.1 Algorithm

The protocol we described in the previous example naturally extends to multiple relays, as described next.

1. S selects randomly t relays $R_1, \ldots, R_t$.
2. S creates each group key K_{G_i} by randomly selecting a packet from the packet dissemination phase among the ones that (a) are not known by R_{i-1}, (b) are known by R_i, (c) are known by at least σ other nodes, where the parameter σ defines the minimum size of G_i.
3. S sends to R_1 a packet of the form:

   $$K_{SR_1}\{R_2, K_{G_2}\{R_3, K_{G_3}\{\ldots K_{G_t}\{D, m\}\}\}\}$$

   such that $\{S, R_i\} \subset G_i$, $R_{i-1} \notin G_i$.
4. The first relay R_1 decrypts the packet using the link key K_{SR_1} and encrypts the encapsulated packet destined for R_2, using the link key $K_{R_1R_2}$, and sends the packet:

   $$K_{R_1R_2}\{K_{G_2}\{R_3, K_{G_3}\{\ldots K_{G_t}\{D, m\}\}\}\}.$$

5. The relay R_i sends to R_{i+1} the packet:

$$K_{R_i R_{i+1}}\{K_{G_{i+1}}\{R_{i+2}, K_{G_{i+2}}\{\ldots K_{G_t}\{D, m\}\}\}\},$$

   which is produced as follows: (1) After removing the two outermost encryption layers (first with a link key $K_{R_{i-1}R_i}$ and then with a group key K_{G_i}), the received packet reveals the next relay R_{i+1} on the path and an encapsulated packet that is encrypted with $K_{G_{i+1}}$. (2) R_i encrypts the encapsulated packet with $K_{R_i,R_{i+1}}$.

6. The last relay R_t simply forwards m to D, after removing the two remaining encryption layers.

Note that this protocol can be used to also support two-way communication: since every relay knows the preceding relay along a path, they can forward a response from D by applying the same type of encryptions but now in the reverse direction.

3.4 Privacy Analysis

We use the term *flow* to describe the set of all the packets that are exchanged to support the communication of a specific S-D pair. We are interested in four forms of *unlinkability*:

- *Unlinkability of packets:* Eve is not able to tell whether two (or more) overheard packets belong to the same flow.
- *Unlinkability with the destination:* Eve is not able to tell which is the destination of an overheard packet. We measure this with the metric $U_{\mathcal{D}} = H(D|E)$.
- *Unlinkability with the source:* Eve is not able to tell which is the source of an overheard packet. We measure this with the metric $U_S = H(S|E)$.
- *Source-Destination unlinkability:* Eve does no learn which source communicates with which destination.

Recall that for us Eve may be a passive external eavesdropper, or an honest-but-curious node in our network.

Unlinkability of Packets

Clearly, we need to have more than one flows in our network, as the uncertainty, to which flow a packet belongs, is constrained by the number of flows. We will next assume that a large number of flows share the network; this is also a basic premise of Tor.

If Eve overhears a packet, she could learn which node transmitted it and which node received it (she could learn one link of the path); if she could overhear multiple packets that she found

were part of the same flow, she could piece together parts of the path, and thus her uncertainty about the communicating parties would reduce. Packet unlinkability is essential to avoid giving such side information to Eve.

In our protocol, the link keys together with the group keys, ensure that all transmitted packets are statistically independent and thus, even if Eve observes multiple of them, she cannot correlate them. The use of link keys ensure that packets appear statistically independent of each other whether or not they belong to the same flow. This property holds also against a relay: the content of a packet a relay can see, by knowing its own link key, is independent of the same packet encrypted with a different link key. It means relays cannot recognize packets that they themselves forwarded earlier along the path.

The only packet that is not protected with a link key is the last packet of the flow. Hence, it remains to protect the last message from a relay, who knows also a link key. The group key plays a role here: it encrypts the message from relays, which makes also the last packet independent and thus unlinkable with its previously seen encrypted version. It follows that for all nodes (including Eve) packets remain unlinkable with each other in the network.

Unlinkability with the Destination

If Eve overhears the transmission of R_t (of the last relay on the path), then she learns the destination of the packet, and thus $U_{\mathcal{D}} = 0$; trivially, this is the case if Eve is the node R_t. The leakage of this information is unavoidable, since D is outside the network. This is also the case in Tor.

If Eve overhears the transmission of any other packet, the packet remains unlinkable with its destination. Indeed, link keys protect the identity of the destination from any node who is not a relay on the path; and group keys protect the identity of the destination from the nodes that are relays. It follows that for any node, including relays, the destination remains unlinkable with any packet of the flow except for the last-hop unencrypted packet.

Unlinkability with the Sender

We here need to distinguish cases depending on which node Eve is. First, assume Eve is not one of the R_i relays on the path; then the link keys make the different packets of a flow indistinguishable, i.e., Eve cannot tell if an overheard packet is the first packet of the flow, and cannot learn anything about the sender. Next, assume Eve is R_1. Then Eve knows that S is the source, and thus for the first packet $U_{\mathcal{S}} = 0$. This is also the case in Tor, if the adversary manages to compromise the first onion-router, to which the user's onion-proxy connects.

Assume now Eve is a relay R_i on the path. R_i knows that $S \in G_i \cap G_{i+1}$, since the source has to be a member of both groups. Thus it can link S with the group $G_i \cap G_{i+1}$. In the example of Fig. 3.2, R_2 learns that $S \in G_2$. Ideally, any node in $G_i \cap G_{i+1}$ would appear equally likely to be

the actual source, i.e., Eve would infer a uniform distribution over these nodes. However, the selection of the groups G_i does not guarantee this property; the distribution will be skewed from the uniform. We numerically evaluate the uncertainty $U_{\mathcal{S}}$ in the evaluation section 3.5, and find that it is very close to uniform.

Sender-Receiver Unlinkability

From the previous arguments it follows that the uncertainty about the communicating pair $U_{\mathcal{S}-\mathcal{D}} = U_{\mathcal{S}} + U_{\mathcal{D}}$ is never 0. Moreover, $U_{\mathcal{S}-\mathcal{D}}$ is the largest possible when Eve is not one of the relays R_i and she does not overhear the last packet of the flow. It is reasonable to assume that the uncertainty about the destination $H(\mathcal{D})$ is larger than about the source $H(\mathcal{S})$, since the destination could be any server on the Internet. Thus $U_{\mathcal{S}-\mathcal{D}}$ takes its smallest value if Eve is the last relay R_t on the path. In our evaluation we assume this worst-case situation and numerically evaluate the sender-receiver uncertainty under this condition.

Side Information Attacks

When analyzing the unlinkability properties that our protocol provides, we only considered the information that the content of the transmitted packets can reveal to an adversary. However, an adversary may also observe additional side-information; the amount and type of this information depends on the actual implementation of the protocol and also on its interplay with other protocols (e.g. the routing protocol used). Such side information is present irrespective of the applied anonymizer solution; indeed most known attacks against Tor are of this kind (e.g. [71, 13]). The possible sources of side information include traffic analysis (timing information, number of sent/received packets), topology (routing and location information), and application level analysis. For an overview of side-channel attacks we refer to [72]. Although it is not possible to conceal all the side information, by its design, our protocol offers a level of anonymity comparable to that of Tor.

3.5 Experimental Evaluation

We use the simulation environment described in section 2.7.2, with the configuration parameters summarized in Table 2.4. We consider networks with fixed network density per unit area, that is, for a k-hop area and a given density d (nodes per unit area) we have in total $n = k^2 * d$ nodes. We first run the secret-agreement for multi-hop networks described in chapter 2, and then our traffic anonymization protocol, described above.

Fig. 3.3 and 3.4 numerically evaluate the sender-receiver unlinkability that our protocol achieves, for different network densities and as a function of the number of nodes in the network. We assume that Eve is the worst-case node for us relay R_t (as we explain in section 3.4); in this case, the sender-receiver uncertainty equals the source uncertainty $U_S = H(S|E)$. We

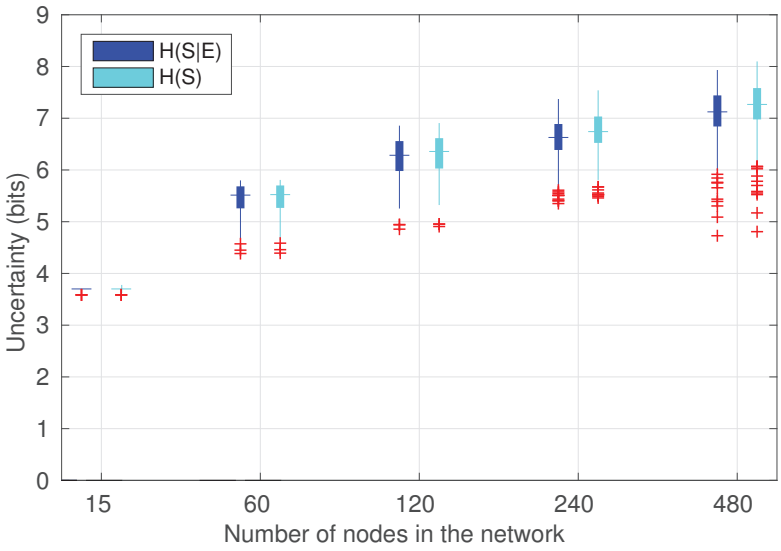


Figure 3.3 – Anonymity for density $d = 15$ and $k \in \{1,2,3,4,5\}$.

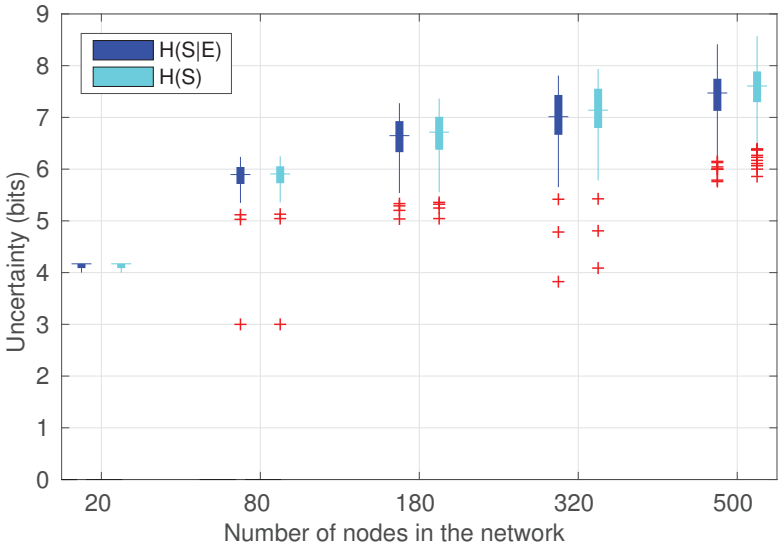


Figure 3.4 – Anonymity for density $d = 20$ and $k \in \{1,2,3,4,5\}$.

compare this to the ideal uncertainty $H(S)$ Eve would have, if each group member would be the source with equal probability (note that $H(S) = 5$ amounts to uniform probability within a group of size 2^5). We find that with our protocol we can restrict Eve to only learn that the source belongs in a set of size approximately half the network population; moreover, Eve perceives each node in the group to be the source with probability very close to uniform.

3.6 Related Work

Our anonymity protocol combines the onion routing of Tor [38] with one-time pad encryptions, to provide protection against computationally unbounded adversaries. There also exist alternative anonymous routing protocols specially designed for ad-hoc networks (e.g. [103, 58]) but they all build on computational limitations. In our design, we considered privacy in the presence of a passive adversary; an active adversary might for instance intentionally introduce timing patterns that she can later identify [71]. Introducing latency and mixing [32] can make timing attacks more difficult but at the same time decreases throughput.

3.7 Summary

In this chapter we presented the design of a traffic anonymization scheme that exploits the security properties of the shared secrets, which we can generate using the techniques described in chapter 2. Our privacy analysis demonstrates that we can achieve a Tor-like level of anonymity, yet without relying on the computational limitations of Eve. We experimentally evaluated the performance of our design over various network configurations, and we showed that we can achieve almost perfect anonymity within a group of roughly half the network size.

4 A Lightweight Encryption Protocol for Sensor Networks

4.1 Introduction

In this chapter we propose a lightweight data encryption protocol suitable for wireless sensor networks. Differently to our approach so far, we are not interested in strong information-theoretic security but rather in exploring what additional security we can achieve, when constrained not to use (or use very few) additional resources to those used for a data collection task in a wireless sensor network.

We consider a wireless sensor network where individual nodes want to send data to a single collection point in the network, the sink. We focus on data collection protocols employing network coding for increased data reliability and in particular to one, SenseCode [55]. Our goal is to complement SenseCode by adding a layer on top of its operations, so as to enable encrypted data delivery between each node and the sink during the data collection task.

Our data encryption protocol tries to balance the following requirements:

1. We want to avoid the overhead of a dedicated "key generation/discovery phase" to construct a pairwise sensor node – sink key before each communication round.
2. We want to use a one-time-pad approach for encryption, as encoding and decoding has low complexity; that is, at every communication round, use a pairwise secret key known to each sensor node and the sink, as one-time pad to encrypt the sensor data.
3. We want to use a different pairwise key per communication round, so that an adversary that captures the key of one round cannot unlock subsequent rounds.

To address these requirements, we propose to construct each key from past data, collected or overheard in previous communication rounds; in particular, keys are constructed as random linear combinations of the past data that both the sensor node and the sink have. An important aspect of this approach is that we can create these keys with low complexity both computationally and in terms of memory requirements. Thus, we can efficiently renew our keys at each

communication round, with very low cost, as we essentially reuse the communication of past data for our key generation.

Similarly to our approach in the previous chapters, we design our protocol building on the fact that, an adversary will not have overheard *exactly* the same transmissions as any sensor node in the network. This can happen because the adversary may not be present at all communication rounds, or may be physically separated from the sensor node. Even if none of the above happens, with high probability a passive eavesdropper will not overhear exactly the same transmissions as a node due to the random wireless channel variation and losses. We strengthen this effect by combining past data across multiple communication rounds to construct our keys, which would require the adversary to overhear the same data as a node over multiple communication rounds as well.

Network coding naturally offers weak security, as observed in the literature in the case of multicasting (the same arguments naturally extend for data collection) [20]. Our protocol can be viewed as enhancing this network coding security, where we now use linear combinations not only of current but also past communication rounds. Our main contribution is on how exactly to perform the mixing across rounds so that we maintain low overhead, that is suitable for sensor networks. In our evaluation, we explicitly compare the security benefits that our protocol offers with the security inherently provided by network coding.

The work presented in this chapter is joint work with Emre Atsan and has been presented in [16].

4.2 Setup and Background

System and Adversary Model

We consider a sensor network of N sensor nodes and one single collection point, the sink. The network operates in rounds, and in each round t, every node i would like to reliably communicate a message x_i^t, $i = 1...N$, to the sink. Each source message x_i^t is a sequence of symbols over a finite field $\mathbb{F}_q$. We assume a data collection protocol that enables network coding; for the sink to decode the linear combinations of source messages, in every packet an N-dimensional coding vector $\mathbb{F}_q^N$ is appended. We also assume that our protocol enables node overhearing; we will use SenseCode [55] to illustrate this effect.

Our adversary, Eve, is a passive eavesdropper or an honest and curious sensor node,i.e., a node that honestly follows the employed protocols but at the same time attempts to extract information regarding the other nodes' messages, using the information she has at her disposal. Any node in our network could be Eve, we have no information regarding her location in the network. Although Eve is assumed to be limited in network presence, that is, she is restricted to one (any one) location in the network, e.g., next to one of our sensor nodes, we do not make any assumptions about her computational and memory capabilities.

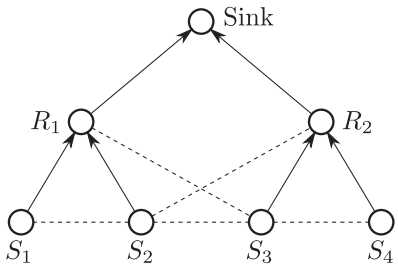


Figure 4.1 – Example of a tree-structured wireless sensor network. Each source routes information toward the sink, through its parent node. Overhearing links are depicted with dashed lines.

Our goal is to design a practical message encryption protocol, on top of the SenseCode operations, that enables each node to securely communicate its messages to the sink, under the presence of Eve.

Possible Use-case Scenarios

A use case could be when the collected sensor data are to be sold to customers; a customer, to avoid paying, could potentially setup an eavesdropping node and try to acquire the data while it is transmitting towards the sink. Our scheme aims to constrain the stingy customer to learn at most a small fraction of the data. In another use case, perhaps the sensor nodes themselves would like, if possible, not to reveal their individual measurements to the other participating nodes. Our scheme would in this case increase the privacy of the participating nodes.

SenseCode Basics

SenseCode is a data collection protocol, that uses network coding techniques to increase the reliability of data collection in wireless sensor networks, and it has been developed and implemented in [55]. SenseCode creates and maintains a tree structure in the network for the routing of messages towards the sink, as depicted for example in Fig. 4.1.

At every communication round t, each node that has a new message x_i^t generates and sends out r packets, where r is the redundancy factor, through its parent node in the tree path towards the sink. Out of the r packets sent, 1 packet is the plain message x_i^t (uncodable SenseCode packet) and $r-1$ packets are linear combinations of x_i^t's, i.e., a mixing of node's own message and other *overheard* messages (codable SenseCode packets). Each intermediate node having a packet to forward to a next hop along the path, before doing so, linearly combines it in an opportunistic manner with its own message and messages from other nodes, i.e., messages from its children (if any) and *overheard* messages from its neighbors. Clearly, the overheard information from neighboring nodes can be plain messages x_i's or linear combinations of these (uncodable and codable SenseCode packets, respectively). At the end of round t, the sink uses all the received packets and tries to decode, so as to obtain

all the x_i^t's. For the sink to be able to successfully decode and eventually obtain all the x_i^t's, it needs to have collected at least N linear independent combinations (given that there were N different messages produced during round t) of the x_i^t's.

Performance Metrics

We are interested in a form of weak security. That is, Eve gets no meaningful information about a specific message x_i. In particular, if x_i and x_k take i.i.d. binary values and Eve receives $x_i \oplus x_k$ we consider both x_i and x_k to be *secure* from Eve as she gets no meaningful information about them.

The *reliability* for a node i is the percentage of messages that were communicated securely to the sink from node i during the whole operation of the network, given that one message x_i^t is generated at each round t. It is defined as:

$$\rho_i = \frac{\#\, secure\ x_i{}'s\ from\ any\ other\ node\ j}{\#\, rounds}, \forall j \neq i.$$

The *average reliability* ρ, captures the performance of our protocol with the respect to the whole network, that is, $\rho = \text{avg}(\rho_1, \ldots, \rho_N)$.

Note that our security metric is pessimistic in two ways: (i) we consider a packet x_i to be not secure even if it is secure from all other sensor nodes in the network but one; (ii) at every round, we implicitly assume that Eve is the node that can at that round decode, thus we give her a strong advantage.

4.3 Message Encryption Protocol

Our protocol aims to exploit the fact that each node of the network will overhear (and have in common with the sink) a random subset of linear combinations of the source symbols, as dictated by the network topology and the channel conditions. More precisely, every node and the sink share a common collection of x_i's and linear combinations of these, over multiple communication rounds. We can use this common information to encrypt future data of node i from an adversary that does not have full knowledge of the shared data.

We define the following parameters for our protocol:

- μ : number of rounds in the past from which we select packets to be combined to create an encryption key.
- q : the field size of the vector space used.

During the rest of this section, we use the operation $(a||b)$ to represent the concatenation of

two given vectors, a and b.

4.3.1 Data Structures

- x_i^t: source message generated by node i at round t. The size of each message x_i^t is fixed and equal to L bits.
- s_i^t: encryption key created by node i which is used at round t.
- $y_i^t = s_i^t + x_i^t$: encrypted message of node i at round t. $+$ represents the addition operation over a given finite field.
- w_i^t: encryption coefficients vector for secret key s_i^t.[1]
- $p_\ell^t = (c_\ell^t || d_\ell^t)$: a SenseCode packet at round t. It is a concatenation of its coding (coefficients) vector (c_ℓ^t) and payload (d_ℓ^t) as defined in [55].

$$d_\ell^t \triangleq \sum_{j=1}^{N} c_\ell^t[j](w_j^t || y_j^t) = \left(\sum_{j=1}^{N} c_\ell^t[j]\, w_j^t\right) || \gamma_\ell^t,$$

  where $\gamma_\ell^t = \sum_{j=1}^{N} c_\ell^t[j]\, y_j^t$ and $c_\ell^t \in \mathbb{F}_q^N$.
- $\mathcal{P}_i^t$: set of packets p_ℓ^t overheard by node i at round t.
- Q_i: a FIFO (first-in first-out) bounded queue of size μ. Its elements are encryption keys (s_i^t) and their encryption coefficients (w_i^t) at node i for the next μ rounds.
- $\mathcal{Y}$: a list of all encrypted messages y_i^t for the last μ rounds. This list will be used for the reconstruction of encryption key s_i^t at the sink.

4.3.2 Algorithm

Message Creation & Encryption

1. Node i at round t generates a source message x_i^t to be communicated to a common collecting sink.
2. Encrypted message $y_i^t = s_i^t + x_i^t$ is prepared using the key s_i^t pulled from the top of queue Q_i at round t.
3. y_i^t and the encryption coefficients w_i^t of s_i^t are encapsulated into a SenseCode message ($w_i^t || y_i^t$) which will be communicated to the sink.

[1] The size of each w_i^t is $\mu \times N(\log_2(q))$ bits.

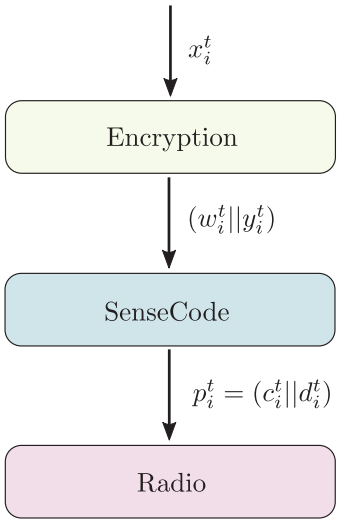


Figure 4.2 – Protocol stack.

Message Collection

1. At each round t, node i communicates the encapsulated message $(w_i^t||y_i^t)$ to the sink using the SenseCode collection protocol (see Fig. 4.2).

2. In order to decode all the encapsulated messages, the sink waits until it receives at least N linearly independent combinations of them. At the end of round t, the sink tries to recover as much encapsulated message as possible from all the packets it receives and overhears.[2]

3. After recovering y_i^t and w_i^t, the sink runs the key reconstruction and message decryption phase to obtain the source message x_i^t.

Key Reconstruction & Message Decryption at Sink

1. The sink updates its list $\mathcal{Y}$ of encrypted messages with the new y_i^t.

2. Then, the sink needs to reconstruct the secret s_i^t using the encapsulated encryption coefficients vector w_i^t and the list of encrypted messages list $\mathcal{Y}$ as follows:

$$s_i^t = \sum_{k=1}^{\mu} \sum_{j=1}^{N} w_i^t[(k-1)N + j]\, y_j^{t-k} \tag{4.1}$$

3. After reconstructing the key s_i^t, node i can obtain the source message $x_i^t = y_i^t - s_i^t$.

[2]The payload d_l^t of the packets received at the sink is random linear combinations of encapsulated messages $(w_i^t||y_i^t)$ of each node $i \in 1...N$.

$\mathbf{t+k}$	$\mathbf{w_1^{t+k}}$				$\mathbf{s_1^{t+k}}$
5	c_l^{t+1}	c_l^{t+2}	c_l^{t+3}	c_l^{t+4}	$\gamma_l^{t+1}+\gamma_l^{t+2}+\gamma_l^{t+3}+\gamma_l^{t+4}$
6	c_l^{t+2}	c_l^{t+3}	c_l^{t+4}		$\gamma_l^{t+2}+\gamma_l^{t+3}+\gamma_l^{t+4}$
7	c_l^{t+3}	c_l^{t+4}			$\gamma_l^{t+3}+\gamma_l^{t+4}$
8	c_l^{t+4}				γ_l^{t+4}

Table 4.1 – Contents of queue $\mathcal{Q}_1$ at round $t+4$.

Key construction at node i

1. (*Initialization*) Node i initializes its key queue Q_i with a predefined set of initial (possibly insecure and all zero) μ vectors.

2. At every round t, for each overheard packet $p_\ell^t \in \mathcal{P}_i^t$, node i updates all the elements in $\mathcal{Q}_i = \{(w_i^{t+1}||s_i^{t+1}),...,(w_i^{t+\mu}||s_i^{t+\mu})\}$, $\forall k \in 1,2,..,\mu$:

$$w_i^{t+k}(k) = w_i^{t+k}(k) + \alpha_{\ell,i}^t c_\ell^t \tag{4.2}$$
$$s_i^{t+k} = s_i^{t+k} + \alpha_{\ell,i}^t \gamma_\ell^t, \tag{4.3}$$

   where $\alpha_{\ell,i}^t \in \mathbb{F}_q$ is the random coefficient generated for p_ℓ^t during this update by node i and $w_i^{t+k}(k) = \left[w_i^{t+k}[(k-1)N+1] \ \ldots \ w_i^{t+k}[kN]\right]$.

3. When a node i pulls an encryption key s_i^t, $(w_i^t||s_i^t)$ from the top of Q_i, it pushes a new all zero element $(w_i^{t+\mu}||s_i^{t+\mu})$ to the bottom of the queue. This ensures the size of Q_i is always fixed and equals to μ.

A Key Construction Example

Let $\mu = 4$ and suppose we are interested in the key construction procedure at node 1, starting from round $t+1$ up to $t+4$. Assume that node 1, collects only one packet over these rounds, i.e., p_l^{t+1}, p_l^{t+2}, p_l^{t+3} and p_l^{t+4}. Table 4.1 shows the contents of the queue $\mathcal{Q}_1$ at round $t+4$. The first row represents the head of the queue, the last row the tail, the first four columns the contents of vector w_1^{t+k} and the last column the encryption key s_1^{t+k}. For example, at round $t+5$, the key s_1^{t+5} is going to be removed from the top of the queue to be used for encrypting the source message x_1^{t+5} and a new key s_1^{t+9} will be initialized and added to the end of the queue. Once a key s_1^{t+k} is used for encryption, its corresponding coefficients vector w_1^{t+k} is attached to the encrypted message y_1^{t+k}. For simplicity, in this example we used $\alpha_{\ell,1}^t = 1, \forall t, \forall \ell$.

4.3.3 Cost Analysis

Memory requirements: Each node i at any given round t has to keep the queue Q_i in its memory. The size of Q_i is fixed and $\mu \times (L + \mu(N \times \log_2(q)))$ bits. In other words, the queue has

μ elements of size $(L+\mu(N\times\log_2(q)))$. An element of the queue $(w_i^t||s_i^t)$ has an encryption key (s_i^t) of L bits and encryption coefficients of size $\mu\times N\times\log_2(q)$ bits.

In order to regenerate the encryption keys at a given round t, the sink should keep a list of encrypted messages $\mathcal{Y}$ for all the N nodes in the last μ rounds. The size of this list in memory is $(\mu\times L\times N)$ bits.

Communication overhead: The size of a packet transmitted by our protocol is $((\mu+1)\times N\times\log_2(q)+L)$ bits, where L is the size of a plain message x_i^t. On the other side the size of a SenseCode packet (without any encryption of message) is $(N\times\log_2(q)+L)$ bits. In other words, encrypting messages at the nodes costs an extra $N\times\mu\times\log_2(q)$ bits per packet transmission.

Note that we are actually using the standard SenseCode protocol with larger messages. We can, therefore, claim that the number of packets transmitted in the network does not change compared to SenseCode without encryption (we found that larger packet sizes do not increase the packet error rates substantially). Moreover, we can significantly compress the coding vectors using techniques similar to [52].

Operational complexity overhead: For every overheard packet in $\mathcal{P}_i^t$ at round t, every node i updates (an addition operation over finite field $\mathbb{F}_q$) μ vectors in its key queue Q_i. Thus, the main computational overhead introduced (per node per round) by encrypting messages is:

$$|\mathcal{P}_i^t|\times\mu \text{ additions of 2 vectors of size } N\times\log_2(q)+L$$

On the sink side, the overhead for reconstructing all N keys s_i^t is $N^2\times\mu$ multiplications of a vector (size L bits) and a scalar (see Equation 4.1). After reconstructing the keys, sink should compute the source messages x_i^t, which requires an extra N additions of 2 vectors of size L.

4.4 Protocol Analysis

We start our analysis by observing that the payload of all packets sent by the protocol are linear combinations of source messages. We can therefore uniquely represent every packet as a vector that collects the coefficients used for linear combining. In this section we will call this vector the *coding vector* of the packet. We define the vector such that the coefficient used to linearly combine x_i^t is at position $Nt+i$ of the vector. The length of the vector is in principle unbounded, but in the following we will always be able to think about the vector as having a sufficient length, as it will be clear from the context.

Our analysis in the following assumes that the x_i^t's are statistically independent across sources and rounds and are uniformly distributed. This could be because we use distributed source coding or because the nature of the application data is so. If this is not the case, we will have a

corresponding reduction in the expected secrecy as determined by the specific correlation patterns.

We denote as $\Pi_i^t \in \mathbb{F}_q^N$ the subspace spanned by the coding vectors of the packets node i collected at round t. We define Z_i^t the subspace spanned by the basis vectors of $\{\Pi_i^1, \ldots, \Pi_i^t\}$, Z_i^t represents all information that a node i can potentially collect up to round t and therefore use to recover messages sent by other nodes. We define W_i^μ the subspace spanned by the basis vectors of $\{\Pi_i^{t-\mu+1}, \ldots, \Pi_i^t\}$, this is the subspace from which the encryption keys s_i^t is chosen.

In the following we will exploit the fact that if the vector $\mathbf{e}_{Nt+i}$ is not in $Z_j^{t'}$, i.e., node j cannot reconstruct X_i^t from the linear combinations it has overheard then X_i^t is *weakly secure* from node j up to time t', i.e., $H(X_i^t|Z_j^{t'})$. To prove this it is sufficient to observe that $\mathbf{e}_{Nt+i}$ can be used to extend a base of Z_j^t and obtain a set of linearly independent vectors, then observe that the corresponding linear combinations of source messages are statistically independent, which implies that X_i^t is statistically independent from what node j knows and, therefore, secure. In this section we want to make two points:

- If the subspace overheard by the adversary doesn't contain the subspace used to create the secret key of round t on node i then with a non-zero probability the data sent by node i will be secure even if the adversary overhears all the packets sent in this round.
- By using linear combinations of past data to create keys our protocol doesn't compromise the security of the data sent in the previous rounds.

For the first point, we define a function g_i^t as follows:

$$g_i^t(\mu) \triangleq \min_{j \neq i} [\dim(W_i^\mu) - \dim(W_i^\mu \cap Z_j^t)].$$

We will show that $g_i^t(\mu)$ determines the probability of picking a key that makes the transmission of node i in round t secure. The actual value of $g_i^t(\mu)$ depends on the network conditions. Here, we want to show that if it is bigger than 1 (i.e.the adversary does not collect everything that node i collected) then our protocol improves the security.

We first observe that our protocol chooses keys uniformly at random over W_i^μ. Indeed what our protocol does is to create a linear combination with random coefficients of the packets it has overheard. The following lemma shows that this linear combination is distributed uniformly.

Lemma 5. *Given a set of k vectors* $\mathbf{v}_1, \ldots, \mathbf{v}_k \in \mathbb{F}_q^N$ *and k uniform and independent random variables* $c_1, \ldots, c_k \in \mathbb{F}_q$*, then* $\sum_{i=1}^k c_i \cdot \mathbf{v}_i$ *is uniformly distributed over* $\langle \mathbf{v}_1, \ldots, \mathbf{v}_k \rangle$.

Now we can use the following lemma to find what is the probability that a given key is not in the subspace overheard by the adversary as stated in Proposition 1:

Lemma 6. *Let two subspaces Π_i and Π_j of $\mathbb{F}_q^N$, for which $\Pi_i \not\subseteq \Pi_j$. Let also $\mathbf{v}$ a vector uniformly chosen in Π_i. Then*

$$\Pr[\mathbf{v} \in \Pi_j] = \frac{1}{q^{\beta}},$$

where $\beta = \min_j[\dim(\Pi_i) - \dim(\Pi_i \cap \Pi_j)]$, $\forall i \neq j$.

Proposition 1. *An encryption key s_i^t produced by our protocol by node i at round t is secure by an other node j with probability $\delta = 1 - \frac{1}{q^{g_i^t(\mu)}}$.*

Now, what remains to be proven is that by using a key that is not in Z_j^{t-1} we actually secure the data being transmitted at round t, against node j.

Proposition 2. *If $s_i^t \notin Z_j^{t-1}$ then $H(X_i^t|Z_j^t) = H(X_i^t)$,$\forall j \neq i$.*

Proof. Node i selects an encryption key s_i^t at round t to encrypt its message x_i^t, i.e. $y_i^t = s_i^t + x_i^t$. For simplicity we will write the coding vectors of the packets as elements of $\mathbb{F}_q^{Nt}$. Now, let $\mathbf{v}_i = [\mathbf{w}_i\ \mathbf{e}_i]$, where $\mathbf{w}_i \in \mathbb{F}_q^{N(t-1)}$, represent the coding vector of the key s_i^t. Also, let $\mathbf{b}_1 \ldots \mathbf{b}_m$ a basis of Z_j^{t-1} and assume that node j overhears all the y_i^t for round t. Then $Z_j^t = \langle \mathbf{b}_1 \ldots \mathbf{b}_m, \mathbf{v}_1, \ldots, \mathbf{v}_N \rangle$, where vectors $\mathbf{b}_i$ have zeros in the last N entries.

We want to show that node j cannot decode data from node i, i.e., $e_{N\cdot(t-1)+i} \notin Z_j^t$, $\forall i \neq j$. Suppose that this is not the case, then we can find α_r and β_r such that:

$$\sum_{k=1}^{Nt} \left(\sum_{r=1}^{m} \alpha_r \mathbf{b}_r[k] + \sum_{r=1}^{N} \beta_r \mathbf{v}_r[k] \right) \mathbf{e}_k = \mathbf{e}_{N(t-1)+i} \tag{4.4}$$

The above equation holds if:

$$\sum_{r=1}^{m} \alpha_r \mathbf{b}_r[N(t-1)+i] + \sum_{r=1}^{N} \beta_r \mathbf{v}_r[N(t-1)+i] = 1$$

and

$$\sum_{\substack{k=1 \\ k \neq N(t-1)+i}}^{Nt} \left(\sum_{r=1}^{m} \alpha_r \mathbf{b}_r[k] + \sum_{r=1}^{N} \beta_r \mathbf{v}_r[k] \right) \mathbf{e}_k = 0.$$

The first equation implies that $\beta_r = 1$, for $r = i$, since $\mathbf{b}_r[N(t-1)+i] = 0$ for all r, and $\mathbf{v}_r[N(t-1)+i] = 1$ for $r = i$ and 0 otherwise. The second equation implies that all β_r, for $r \neq i$, must be zero, for we will not be able to cancel out the corresponding $\mathbf{e}_{N(t-1)+r}$ term. Now, Equation 4.4 can be rewritten as:

$$\sum_{\substack{k=1 \\ k \neq N(t-1)+i}}^{Nt} \left(\sum_{r=1}^{m} \alpha_r \mathbf{b}_r[k] \mathbf{v}_i[k] \right) \mathbf{e}_k + \mathbf{e}_{N(t-1)+i} = \mathbf{e}_{N(t-1)+i} \Rightarrow$$

$$\sum_{r=1}^{m} \alpha_r \mathbf{b}_r + \sum_{\substack{k=1 \\ k \neq N(t-1)+i}}^{Nt} \mathbf{v}_i[k]\mathbf{e}_k = 0 \tag{4.5}$$

However, Equation 4.5 implies that $[\mathbf{w}_i \ \mathbf{0}_N]$ should have been in the span of $\mathbf{b}_i \dots \mathbf{b}_m$, which is a contradiction. Therefore node j cannot decode x_i^t and so $H(X_i^t|Z_j^t) = H(X_i^t)$. □

In the next proposition, we argue that using an encryption key at round t, produced by our protocol, does not compromise the security of the data sent in previous rounds, for which the adversary already had maximum uncertainty.

Proposition 3. *If* $H(X_i^t|Z_j^t) = H(X_i^t)$ *then* $H(X_i^t|Z_j^{t+1}) = H(X_i^t)$, $\forall j \neq i$.

Proof. Let $\mathbf{b}_1 \dots \mathbf{b}_m$ a basis of $\mathbb{F}_q^{Nt}$ that spans Z_j^t and $\mathbf{c}_1 \dots \mathbf{c}_m$ a basis of $\mathbb{F}_q^{Nt+1}$, where $\mathbf{c}_i = [\mathbf{b}_i \ 0]$.

By assumption it holds that $\mathbf{e}_l \notin \langle \mathbf{b}_1 \dots \mathbf{b}_m \rangle$, where $\mathbf{e}_l \in \mathbb{F}_q^{tN}$. Given that, it is straightforward to show that also $\mathbf{e}_l \notin \langle \mathbf{c}_1 \dots \mathbf{c}_m \rangle$, where $\mathbf{e}_l \in \mathbb{F}_q^{Nt+1}$, and vice versa.

Let the vector $\mathbf{v} = [\mathbf{w} \ 1]$, where $\mathbf{w} \in \mathbb{F}_q^{Nt}$, represent a coding vector of the key to be used in round $t+1$, and assume that $\mathbf{e}_l \in \langle \mathbf{c}_1 \dots \mathbf{c}_m, \mathbf{v} \rangle$. We can write:

$$\sum_{i=1}^{m} \sum_{r=1}^{Nt+1} \alpha_i \mathbf{c}_i[r]\mathbf{e}_r + \sum_{r=1}^{Nt+1} \gamma \mathbf{v}[r]\mathbf{e}_r = \mathbf{e}_l \Rightarrow \tag{4.6}$$

$$\sum_{i=1}^{m} \sum_{r=1}^{Nt} \alpha_i \mathbf{b}_i[r]\mathbf{e}_r + \sum_{r=1}^{Nt} \gamma \mathbf{w}[r]\mathbf{e}_r + \gamma \mathbf{e}_{Nt+1} = \mathbf{e}_l. \tag{4.7}$$

For $l \leq Nt$, for Equation 4.7 to hold, it should be $\gamma = 0$. What remains cannot hold, because it contradicts the assumption $\mathbf{e}_l \notin \langle \mathbf{b}_1 \dots \mathbf{b}_m \rangle$. For $l = Nt+1$, Equation 4.7 does not hold because by construction, the basis $\mathbf{c}_1 \dots \mathbf{c}_m$ cannot span the vector $\mathbf{e}_{Nt+1}$. We conclude that $\mathbf{e}_l \notin \langle \mathbf{c}_1 \dots \mathbf{c}_m, \mathbf{v} \rangle \ \forall 1 \leq l \leq Nt+1$, and therefore the uncertainty about message x_i^t after the observation of round $t+1$ remains the same. □

4.5 Experimental Evaluation

Simulation Environment and Parameters

SenseCode is implemented as a TinyOs module and tested with the TOSSIM simulator [61]. We implemented our encryption protocol in Java, and we evaluate its performance using the TOSSIM simulation results and the *nc-utils* toolbox [3]. We used a fixed field $\mathbb{F}_{2^4}$ for the network coding operations. Each TOSSIM simulation consists of 100 consecutive communication rounds. We consider the following topologies:

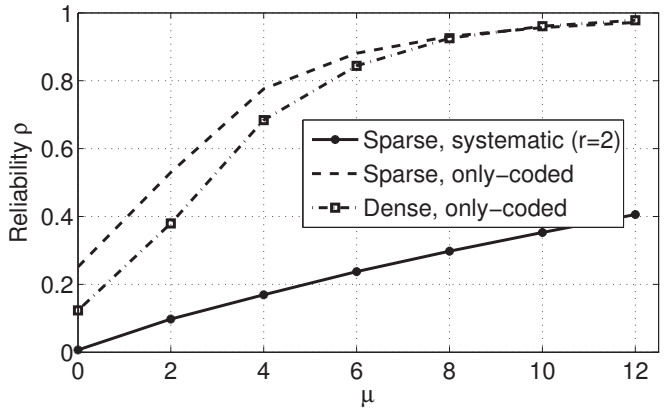


Figure 4.3 – Reliability - 7 × 7 Square Topology.

1. a 7 × 7 square grid ($N = 49$), with the sink located in the middle of the grid,
2. a 3 × 16 rectangular grid ($N = 48$), with the sink located in the middle of the short edge of the grid.

We configure each topology with two different inter-node distances: 20 m (sparse deployment) and 10 m (dense deployment), each of them yielding a different density of the network. For all the above network deployments, we test the performance of our scheme under the following scenarios:

- Every node is permanently present in the network.
- Node i can be in either connected or disconnected state. We set the *mean time between failures* (MTBF) to 400 sec, and the *mean time to repair* (MTTR) to 40 sec[3].

For the first scenario, we consider coded communication, for which nodes introduce a single codable packet per round (in [55] we describe as *codable* the packets that we allow the sensor nodes to linearly combine with other packets before forwarding them to the sink). For the second scenario, we consider redundancy $r = 2$, where each node injecting in network one uncodable and one codable packets per round. We assume that the uncodable packet is encrypted while the codable is not.

Evaluation Results

We present average reliability results as a function of μ to observe the effect of using larger windows when creating secret keys. Note that $\mu = 0$ corresponds to the inherent security provided from the network coding operations in SenseCode. We emphasize that our reliability metric (see section 4.2) considers as secure only messages that are secured concurrently from

[3]Selected from [55], as a meaningful use-case for SenseCode.

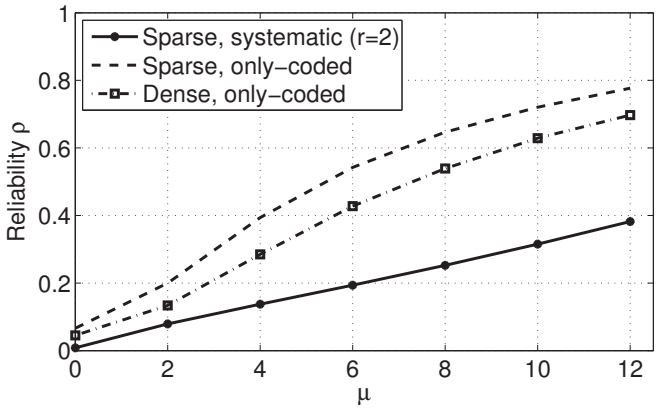


Figure 4.4 – Reliability - 3 × 16 Rectangular Topology.

all possible eavesdroppers, i.e., a communication is secure if none of the network nodes can recover what was sent.

Fig. 4.3 presents the average reliability ρ as a function of μ, for the square grid topology. We observe that our protocol increases the average number of secretly communicated messages by exploiting past communications. In other words, when we increase the parameter μ, we increase the time window of secrecy accumulation over time. As expected, for higher values of μ, we provide a better reliability.

For only one coded packet, we see that in a dense deployment the reliability is less, in comparison with the sparse, since the nodes overhear more common packets, but still increases fast with μ. For $r = 2$, the reliability is degraded due to fact that every node sends at least one uncoded SenseCode packet, facilitating in that way the task of the adversary. Nevertheless, our scheme achieves 20% improvement in reliability for small values of μ.

Fig. 4.4 presents similar results for the rectangular topology. In general, this topology provides less secrecy compared to a square grid, because the information flow (collection tree) is more concentrated to several points (nodes) in the network compared to a square grid. Even in this challenging topology, we provide up to 40% increase in terms of reliability for as low as $\mu = 4$.

Note that in both Fig. 4.3 and 4.4, our approach achieves higher reliability as compared to $\mu = 0$, the reliability achieved by SenseCode alone. For completeness, we provide in Table 4.2 the average percentage of y-packets decoded correctly to the sink per round with each scheme.

4.6 Related work

Key distribution and establishment in wireless sensor networks has different characteristics, requirements and limitations than in traditional networks mainly due to the limited resources on sensor nodes. As a result, the widely accepted key management schemes for traditional

sparse, 3×16, $r = 2$	79.0243%
sparse, 3×16, only-coded	64.4745%
sparse, 7×7, $r = 2$	95.8544%
sparse, 7×7, only coded	79.6517%
dense, 3×16, only-coded	76.5313%
dense, 7×7, only coded	90.8671%

Table 4.2 – Measured average delivery ratio per scheme.

networks have drawbacks for sensor network environments [33], [28], [25]. A popular approach is the scheme of Eschenauer and Gligor (EG) [40] and follow up works (e.g., [27]), where each node is provided with a set of cryptographic keys (key ring) randomly selected from a common pool and uses these as common randomness to create keys with other nodes. Like EG-based schemes, our nodes also collect random keys, yet our randomness does not come from pre-distribution but from the randomness of the wireless channel conditions and topology; this enables us to easily refresh our keys at every round and use them for one-time-pad encryption, making, therefore, our scheme suitable for protecting against computationally powerful adversaries.

Our work can also be seen as offering weak security through network coding [20]; lightweight protocols have been developed for weak security as in [76], [101] yet not applied to sensor networks. The work in [95] looks at security for network coded sensor networks with keys distributed by a mobile agent that visits the nodes.

4.7 Summary

In this chapter we presented a data encryption protocol for wireless sensor networks that builds on top of the operations of a data collection protocol that employs network coding. Our protocol leverages the shared information, between nodes and the sink across communication rounds, to enable secure data delivery under the presence of network limited, yet computationally unconstrained adversaries. Our approach can be viewed as an enhancement of the weak security that network coding inherently offers, with low additional operational complexity.

In summary, the contributions in this chapter are:

1. We design a data encryption protocol that integrates well with SenseCode [55], as well as other protocols that employ network coding over sensor networks, and offers increased security at low additional complexity.
2. We experimentally evaluate its performance, using the TOSSIM simulator [61] over several settings.
3. We offer an analysis that supports the trends we observe experimentally.

5 A Steganographic Mechanism for Private Messaging

5.1 Introduction

In this chapter we consider the problem where two communication parties, Alice and Bob, wish to exchange short secret messages such that an adversary, Eve, who observes their communication, does not know that they are exchanging secret messages, i.e., they wish to hide from Eve the fact that they are exchanging secret messages. We are interested in enabling them to do so by hiding their secret message into another innocuous text, and in investigating if Eve can distinguish between normal messages and messages carrying hidden information.

Linguistic, or text-based, steganography is concerned with the problem of hiding information within natural text. The majority of existing approaches to linguistic steganography follow the automated *covertext* modification strategy (see also section 5.7 on related work): given a piece of natural text referred to as the *covertext*, hidden bits are embedded by applying modifications to the covertext, so as its original meaning and grammatical correctness is preserved, that result into a *stegotext* object. Despite the benefit of zero user-effort required, these techniques have significant drawbacks. First, the automatically generated stegotexts are typically vulnerable to steganalysis attacks – methods for detecting the existence of steganography [90, 35, 102]; automation is highly likely to introduce easily detectable syntactic and semantic unnaturalness. Second, they usually require off-line access to a large amount of linguistic resources and sophisticated Natural Language Processing tools, which makes them rather impractical to use. Third, the covert rate achieved, the number of hidden bits per stegotext word, is quite low; typically, only a few bits are embedded in a very long stegotext.

We design our linguistic steganographic mechanism aiming to address the aforementioned drawbacks of existing approaches. We refrain from the purely automated covertext modification technique and we propose a *semi*-automated scheme that (a) drops the need for a dedicated covertext and simply produces a new stegotext for each new message, (b) involves human interaction: the hidden message is automatically embedded in a sequence of words that the user edits to create the final stegotext. The "user-in-the-loop" involvement has indeed the benefit of producing high quality stegotexts; the user irons out elements that do not feel

like natural language. Despite the continuous advances in the NLP field, enabled by contributions from domains such as Deep Machine Learning, we believe that we are still far from being able to automatically compose perfect natural language. For applications like steganography, where the slightest hint of unnaturalness could compromise the security, we believe that the "user-in-the-loop" involvement is a reasonable approach.

The main challenge, for an approach involving the user, is in minimizing the user-effort required. In our design, we address this by building a dictionary of words and a language model for producing sequences of words, such that the user can easily combine them into natural language sentences. We do not require for our encoding process to have off-line access to linguistic resources, but we rather build the dictionary and the language model out of small text corpora, as needed. In addition, we design our approach in order to achieve high covert rate. Note that a low covert rate yields a long stegotext for a secret message of just a few characters; this enables Eve with a very low complexity detection test: given that the users normally exchange short messages, a very long message is very likely to be a stegotext.

We implement our steganographic mechanism and we experimentally evaluate its performance. We use Amazon's Mechanical Turk [15], an on-line platform for Human Intelligence Tasks, to ask human users to modify various stegotexts such that they become meaningful natural texts. We measure the amount of user-effort required and we demonstrate how our design choices facilitate the users in their task. We also measure the covert rate achieved and we show that it is feasible to embed a significant amount of bits in small stegotexts. We evaluate the quality of our stegotexts by using two steganalysis methods that we design for detecting unnaturalness in sentences. Our observations indicate that, at least by applying our detection methods, it is rather difficult for our user-enhanced stegotexts to be detected as suspicious. This corroborates our intuition on the benefits of involving the "user-in-the-loop".

The work presented in this chapter is also presented in [82].

5.2 Setup

Problem Statement

We consider two communications parties, Alice and Bob, that wish to communicate short secret messages to each other by using a popular third-party messaging service. Alice and Bob frequently use this service for their normal communications, but wish to hide from the service provider that they might occasionally use it for communicating secret messages.

We assume that there exists an adversary, Eve, that has access to the internal infrastructure of the messaging service. Eve can inspect all the messages passing through the provider and tries to identify those that might be hiding a secret message. Eve can be the provider itself or an external adversary that has gained access to the provider's infrastructure. We assume that the load of messages that passes through the provider's infrastructure is large enough to

prevent Eve from visually inspecting each one of them; she instead runs detection algorithms to identify the existence of steganography. Note that the last assumption does not imply that our approach is de-facto vulnerable to human inspection attacks – the "user-in-the-loop" is a significant step toward successfully mitigating these attacks. It rather implies that the load is such that Eve does not have any better strategy for identifying the steganographic messages than running a detection algorithm on each message passing through her infrastructure.

Our goal is to design a steganographic mechanism that enables Alice and Bob to exchange short secret messages, under the presence of Eve, by masking them as innocuous messages. Eve should not be able to distinguish between normal messages and those carrying hidden information.

Possible Use-case Scenarios

Users frequently exchange messages over messaging services, e.g., e-mail, chat, tweets, offered by popular providers, e.g., Google, Facebook, Microsoft, Tweeter, etc. These services are usually offered for free since the companies' business models depend mainly on advertising. That is, the service providers analyze the user data being carried over their infrastructure to infer information about them and eventually project relevant advertisements through their applications or collaborating ones. If the users start encrypting their messages to protect them from providers, they risk to experience degraded quality, or even interruption, of the free service. The users would like to be a able to use the free service and receive ads tailored for them, while maintaining the privacy for a small subset of their messages.

As another possible use-case scenario consider an Eve empowered by law that gains access to the service provider's infrastructure, despite the fact that the provider is against this. Such an adversary could be a powerful government or organization. Eve knows that users often use the messaging service to communicate sensitive information to each other, relying on their trust in the provider to not share these with unauthorized third-parties; Eve wants to gain access to this information and potentially use it against them. The provider wishes for plausible deniability regarding sensitive information of its users. It is possible to offer plausible deniability by applying steganography, while impossible by simply using encryption.

5.3 Steganographic Mechanism

In this section we describe the high-level components of our steganographic mechanism and we provide an explanation about the role of each component. In section 5.4 we explicitly describe how we choose to implement the functionality of each component. Our steganographic scheme consists of an encoder and a decoder and operates as depicted in Fig. 5.1. We assume in this section, that Alice and Bob have already pre-shared the material they need to perform the operations described here.

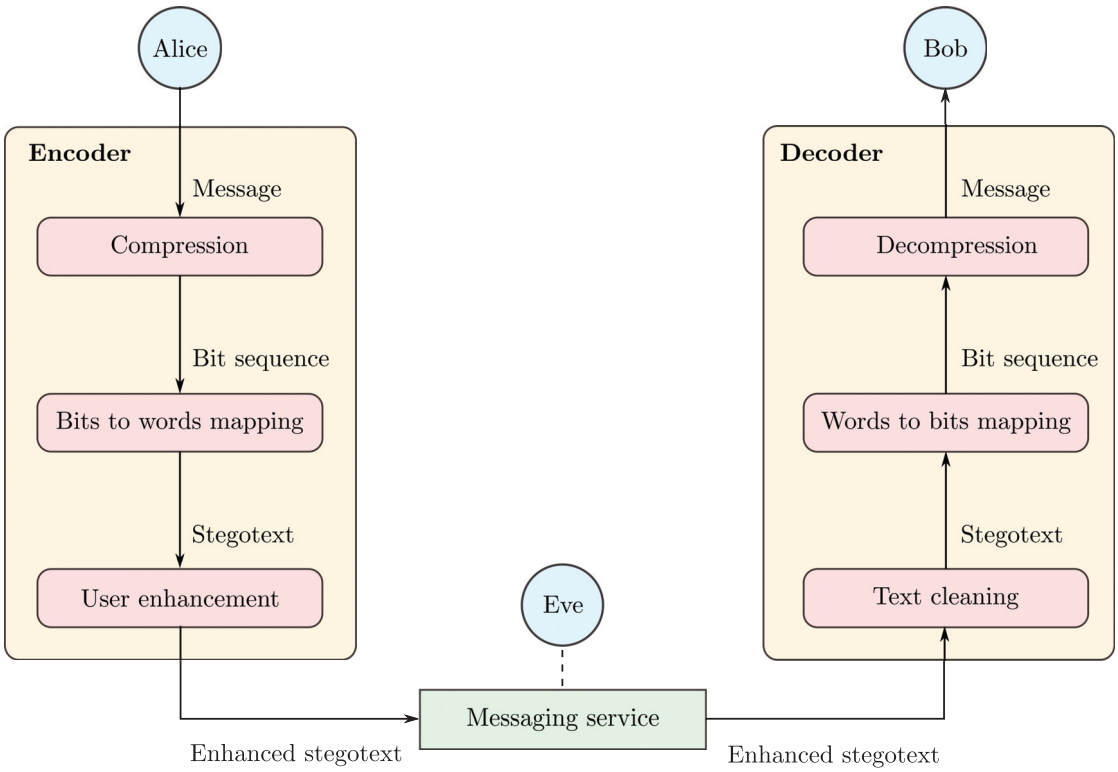


Figure 5.1 – The basic encoder and decoder components.

Compression/Decompression

The first operation we perform is compressing the secret *message* into a *bit sequence*. Clearly, by the nature of the application, a lossless compression technique is required. Assume that the message is composed by a sequence of n random variables x^n taking values in an alphabet $\mathcal{X}$ (e.g. the English alphabet), i.e., $x_i \in \mathcal{X}$, $1 \leq i \leq n$. The compression component implements the operation:

$$C : x^n \rightarrow \{0,1\}^*$$

Respectively, the decompression component implements the inverse operation:

$$C^{-1} : \{0,1\}^* \rightarrow x^n$$

Note that we use compression and not just any mapping between source symbols and bits in order to increase the covert rate of our system. An obvious alternative would be to use the ASCII (or Unicode) representation of the characters in the message for converting it into a bit sequence, but this approach would produce a longer sequence than the compressed one (assuming a compression scheme based on the statistics of the source symbols). As we will explain next, a longer bit sequence yields a longer stegotext. We require our stegotext to be as small as possible for the following reasons:

1. One of our basic premises is the "user-in-the-loop" involvement. We want to make the task of the user as easy as possible, that is, reduce the effort and time he spends on modifying a stegotext by reducing its length.

2. The end goal of our system is to produce an "as innocent as possible" stegotext that will be sent through the third-party provider to Bob. The length of the stegotext may give away the existence of steganography. For example, assume that Alice and Bob communicate over chatting, where they usually sent 2-3 sentences per message; a message that suddenly appears to be much longer, say 20-30 sentences, is an good candidate to be flagged as suspicious.

Bits-to-words/Words-to-bits mapping

The second operation we perform is mapping the bit sequence to a sequence of linguistic words, which we refer to as stegotext, and to the words of the stegotext embedding the hidden message as *stego words*. Assume that $\mathcal{W}$ is a set of linguistic words and let w^m be a sequence of m random variables taking values in $\mathcal{W}$, i.e., $w_i \in \mathcal{W}$, $1 \leq i \leq m$. The bits-to-words mapping component implements the operation:

$$F: \{0,1\}^* \rightarrow w^m$$

Respectively, the words-to-bits mapping component implements the inverse operation:

$$F^{-1}: w^m \rightarrow \{0,1\}^*$$

This is, essentially, the stage at which the encoding/decoding of a hidden message happens. Clearly, the mapping can be implemented in various ways. In section 5.4, we explain how we perform this operation in two steps: first, we use a *dictionary* that maps bit sequences of length b bits to sets of words, and then we use a *language model* that selects words from the sets, so as to form the sequence w^m. For a compressed bit sequence of length c bits, $m = \lceil c/b \rceil$ stego words are produced. That is, for achieving high covert rate, we need b to be as big as possible. In section 5.4.5 we compute an upper bound for b that depends on the input resources we use for building our dictionary and on other parameters of our design.

User enhancement/Text cleaning

The last operation performed produces the *enhanced stegotext* that gets sent to Bob. This step involves Alice manually modifying the stegotext produced from the previous component, in order to create an "as meaningful as possible" enhanced stegotext, according to her judgment. Certainly, the modifications that Alice introduces should comply to some requirements, so as to enable decoding at the receiver of Bob. Given a sequence w^m of stego words from set $\mathcal{W}$, the user enhancement component should ensure that Alice is *not* be able to:

1. *delete* or *modify* any of the m words,
2. *change* the order of the sequence elements.
3. *insert* a word from set $\mathcal{W}$ in the sequence.

The last requirement is not a strict one, and we discuss later its significance. Clearly, Alice is allowed to insert in the sequence any word $w' \notin \mathcal{W}$. The component "Text cleaning" at the decoder of Bob, filters out the extra words that Alice has inserted.

The enhancement by the user does not only make the stegotext look like natural language text but also it helps to *personalize* it. A possible attack that Eve could launch is to look for user's writing patterns: type of words used, punctuation, capitalization etc. Given that Alice and Bob have been using the third-party messaging service in the past, it is reasonable to assume that Eve possesses a significant fraction of their written communication that she could use to infer writing patterns. A stegotext that lacks user's writing style is a good candidate to be flagged as suspicious.

5.4 Design Choices

In this section we explain how we implement each building block of the steganographic mechanism we described in section 5.3, and we provide the intuition behind our design choices. Our goal is to design a mechanism that is easily implementable and practical to use.

5.4.1 Compression

The nature of the English language, i.e., the predictability of the English letter frequencies, makes the Huffman coding a reasonable choice for compressing the secret message into a bit sequence, in a lossless and efficient way. Another straightforward observation is that some words are more *likely* to occur than others in the English written language; e.g., the words "the", "of", "and", "is", "to", etc. are much more often used in written English texts than others. In particular, studies have shown that the top 100 most frequently used words in English make up around 50% of all written material, and the top 300 make up about 65% [42].

We choose to use a *mixed* Huffman coding scheme [51], where the symbol alphabet $\mathcal{X}$ includes all the printable ASCII characters (lower and capital case letters, numbers, space, digits, punctuation) and a set of *frequently used* English words $\mathcal{E} = \{w_1, \ldots, w_r\}$:

$$\mathcal{X} = \{\text{printable ASCII characters}\} \cup \mathcal{E}$$

A given mixed Huffman codebook $\mathcal{C}$, implements the compression function C and the decompression function C^{-1}. Note that the Huffman code is a prefix-free and thus uniquely

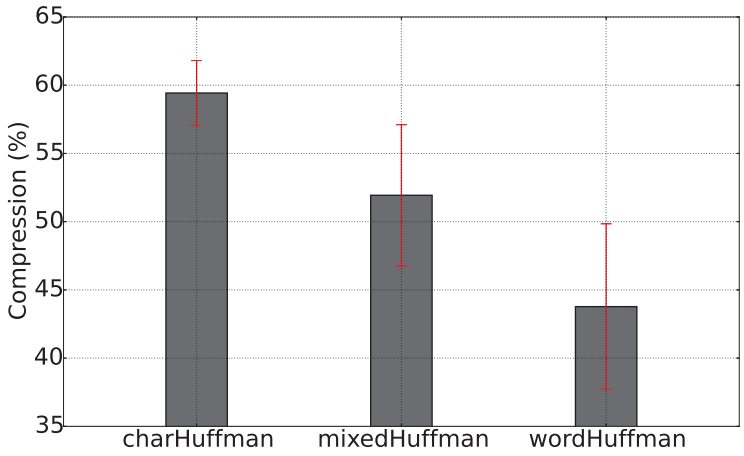


Figure 5.2 – Average percentage of compression achieved by different Huffman codebooks.

decodeable code; in other words, the following property holds for function C (and C^{-1}, respectively):

$$C(x_1,\ldots,x_n) = C(x_1)\ldots C(x_n)$$

We motivate our choice by investigating the performance of three different Huffman-based compression techniques:

(a) A character-level Huffman, where the symbol alphabet includes only the printable ASCII characters.

(b) A mixed Huffman, where the symbol alphabet includes the printable ASCII and the words from a set $\mathcal{E}$. We form the set $\mathcal{E}$ by concatenating the set of the 300 most frequently used words in English [42] and the NLTK's stop words set [21]. The result is a set comprising 330 frequently used words.

(c) A word-level Huffman, where the symbol alphabet includes all the words that *will* appear in the messages to be compressed; this is an idealistic scheme, which we use as a lower bound for comparison.

We derive the frequencies of the characters and the words we need from a large training corpus which we form by concatenating the *reuters*, the *brown* and the *wikipedia* corpora (approximately 78M characters, 13M words – for more details about the corpora see Table 5.3). We use the *overheard* corpus to compress; we derive 4825 sentences of length between 4 and 15 words and we compress each one using the three different techniques.

In Fig.5.2 we show the average compress ratio (in percentage) achieved by each technique. First, we observe that by using a mixed Huffman approach we gain a non-negligible 7% in compression w.r.t the character-level Huffman. Second, we see that the ideal scheme of the word-level Huffman achieves a 44% average compression; this is achieved by considering

2863 words in the dictionary (these are the unique words appearing in the sentences we chose to compress). Interestingly, by including only 330 words in the mixed Huffman codebook, we reach half-way through the gain in performance that the ideal scheme offers. The key differences are that the mixed Huffman codebook (a) is ten times smaller in size (in KB), (b) does not need a-priori knowledge of the messages to be compressed, (c) can compress any message.

Note that Alice and Bob may use the same pre-computed codebook to compress/decompress messages that can be included in the resources of the implementation or our steganographic mechanism.

5.4.2 Dictionary

A *dictionary* $\mathcal{D}$ is a unique mapping between bit sequences of length b bits and sets of linguistic words $\mathcal{B}$. In the following, we will refer to such a set as a *bin* and to the words included in the bins of a dictionary as *bin words*. The number of entries in a dictionary, i.e., the number of unique mappings between sequences of b bits and bins, is the size of the dictionary. We consider dictionaries of size 2^b. Each bin is populated with p words in total.

A given dictionary $\mathcal{D}$, implements the bit-sequence to bin mapping:

$$D : \{0,1\}^b \rightarrow \mathcal{B}$$

Respectively, the bin to bit-sequence mapping:

$$D^{-1} : \mathcal{B} \rightarrow \{0,1\}^b$$

For convenience, we denote the bin corresponding to the j^{th} entry as $\mathcal{B}_j$, where $j \in \{0,\ldots,2^b - 1\}$. We also denote the set of bin words as $\mathcal{W}_B$, with $\mathcal{W}_B = \bigcup \mathcal{B}_j$, $\forall\, j$. If a word in not included in the bin words set, we refer to it as a *stop word* and we denote the set of stop words as $\mathcal{W}_S$. Note that a dictionary can be viewed as a fixed-length coding scheme, thus the following property holds for function D (for D^{-1}, respectively):

$$D(\{0,1\}^*) = D(\{0,1\}^b) \ldots D(\{0,1\}^b)$$

A compressed bit sequence of $c \geq b$ bits is mapped into a bin sequence as follows:

1. The compressed bit sequence is parsed into $\lceil c/b \rceil$ sub-sequences of length b bits each[1].
2. Each sub-sequence is mapped to a bin $\mathcal{B}_j$ according to a dictionary $\mathcal{D}$.

The resulting bin sequence can be converted back to the original bit sequence in a determinis-

[1] If needed, the compressed bit sequences is padded with 0's at the end. This operation is reverted while retrieving the hidden message.

Bit sequence	Bin
00	$\mathcal{B}_0 = \{$ to, the, an $\}$
01	$\mathcal{B}_1 = \{$ I, we, you $\}$
10	$\mathcal{B}_2 = \{$ music, piano, weather $\}$
11	$\mathcal{B}_3 = \{$ like, hate, hear $\}$

Table 5.1 – An example dictionary of size 4, with 3 words in each bin.

tic fashion by using the same dictionary $\mathcal{D}$.

Example: Assume we use $b = 2$ and $p = 3$. An example of a dictionary of size 4 is shown in Table 5.1. Assume a message that has been compressed into the bit sequence "01110010". The mapping to the corresponding bin sequence is shown below:

$$\begin{array}{lcl} \text{Bit sequence} & : & \underbrace{01}\ \underbrace{11}\ \underbrace{00}\ \underbrace{10} \\ \text{Bin sequence} & : & \mathcal{B}_1\ \ \mathcal{B}_3\ \ \mathcal{B}_0\ \ \mathcal{B}_2 \end{array}$$

Any selection of words from the bins $\mathcal{B}_1$, $\mathcal{B}_3$, $\mathcal{B}_0$, $\mathcal{B}_2$ carries the compressed bit sequence. For example, the words "I like the weather" embed the "01110010" bit sequence. In section 5.4.3, we describe how the selection of words from a bin sequence is performed.

Dictionary Building Algorithm

Let $\mathcal{W}_R$ denote the set of unique words derived from a text corpus R, $\mathcal{E}$ a set of frequently used English words, $\mathcal{W}_S$ the set of stop words and $\mathcal{W}_B$ the set of bin words. A dictionary $\mathcal{D}_R$, of size 2^b entries, is created using the corpus R as follows:

1. The sets $\mathcal{W}_B = \mathcal{W}_R - \mathcal{E}$ and $\mathcal{W}_S = \mathcal{W}_R \cap \mathcal{E}$ are computed [2].
2. For every word $w_i \in \mathcal{W}_B$, its number of occurrences o_i in the corpus R is counted, and its frequency of occurrence f_i is calculated:
$$f_i = \frac{o_i}{\sum_{j=0}^{|\mathcal{W}_B|} o_j}.$$
3. A vector $\mathbf{v}$, with $|\mathbf{v}| = p \cdot 2^b$, is constructed be repeating $\lceil f_i \cdot p \cdot 2^b \rceil$ times each word $w_i \in \mathcal{W}_B$. Finally, the vector $\mathbf{v}$ is randomly shuffled.
4. All the elements between $\mathbf{v}[i]$ and $\mathbf{v}[i+p]$, $i \in \{0, p, 2p \ldots, (p-1)2^b\}$, are placed in the bin B_j, $j \in \{0, 1, \ldots, 2^b - 1\}$ of the dictionary D_R.

[2] Due to the rounding operation at step 3 of the dictionary building process, some words with very low frequency might not be eventually placed in a bin. At the end of step 4, the sets $\mathcal{W}_B$ and $\mathcal{W}_S$ are accordingly updated to contain all words within the bins and words from the corpus not placed in a bin, respectively.

At this point, each word $w_i \in \mathcal{W}_B$ is, in the worst case, placed in $\max(f_i \cdot p \cdot 2^b, 2^b)$ bins in total.

Key Points

We choose to exclude from the bin words set some frequently used English words (e.g., the words "a", "and", "to", etc.) in order to give flexibility to the user while enhancing the stegotext. If a word has been included in the dictionary, it should not be used while enhancing since it would introduce bits in the original compressed bit sequence and create, thus, ambiguity at the decoder. As discussed earlier, a small set of words gets very frequently used in English; it is, therefore, reasonable to assume that with high probability the user will use these words while enhancing the stegotext, meaning that they should be excluded from the dictionary. In section 5.6, we demonstrate how we verified this assumption.

We allow a bin word to appear in more than one bins. If it was required that each bin word appeared only in one bin, the dictionary size would be restricted to be less or equal to $\frac{|\mathcal{W}_B|}{p}$; reducing the dictionary size, i.e., the parameter b, automatically reduces the achieved covert rate for a given compressed bit sequence. Another reason for repeating words in different bins is for increasing the probability of finding words between adjacent bins of a bin sequence, that match well in natural language. For example, assume that the word "sunny" is placed only in bin $\mathcal{B}_1$, and the word "weather" in bin $\mathcal{B}_2$; unless the bins occur in that order there is no chance that these words could get selected together. Instead, if both words appeared in more than one bins, there exist more combinations of bin sequences that allow these words to get selected.

A word to be placed in a bin is drawn according to the frequency distribution of the words in R. Note that the way we construct the vector $\mathbf{v}$, implies that a given word w_i appears in it with probability f_i. Also, shuffling $\mathbf{v}$ and assigning p elements to a bin $\mathcal{B}_j$, corresponds to independently sampling p times the vector $\mathbf{v}$. In other words, the p words chosen come from the frequency distribution of the words in R. The intuition behind this design choice, is that by having inside a bin words from different frequencies we increase the probability of finding suitable combination of words among the bins of a sequence.

The repetition of bin words in more than one bins can be regarded as "introducing noise" in the communication channel between Alice and Bob. For a given stegotext, i.e., a sequence of words received, Bob has to infer the bin sequence that these words came from. Clearly, if each word appears in only one bin of his dictionary $\mathcal{D}$, the conversion to the correct bin sequence is straightforward. If each word appears in more than one bins of $\mathcal{D}$, there exist more than one possible bin sequences that correspond to the received stegotext. In section 5.4.4, we describe how the decoder of Bob decides on the most probable bin sequence.

Note that Alice and Bob need to either share the dictionary $\mathcal{D}$, or independently compute it by using the same corpus R plus a secret key $\mathcal{K}$ (needed for appropriately shuffling in step 3 of the dictionary building process). This is an unavoidable requirement since sharing bootstrap

Symbol	Meaning
R	A text corpus
$\mathcal{D}_R$	A dictionary based on corpus R
$\mathcal{E}$	A set of frequently used English words
$\mathcal{W}_R$	The set of unique words appearing into corpus R
$\mathcal{W}_B$	The bin words set; words from R included in the dictionary
$\mathcal{W}_S$	The stop words set; words from R not included in the dictionary
L_R	An n-gram model based on corpus R
p	Number of words in each dictionary bin
b	Length in bits of the compressed bit sub-sequences
k	Length in bins of the bin sub-sequences

Table 5.2 – Commonly used symbols in our steganographic scheme.

secret materials is a basic premise in any security approach. Nevertheless, care should be taken on how this sharing is done because any wrong move could make Eve suspicious. A possible approach could be the following: assume that Alice and Bob already share a symmetric key $\mathcal{K}$, and that at some point Alice sends a link to an interesting article she read on-line as a message to Bob through the messaging service. Bob perceives this action as a hint from Alice to use this article to build a dictionary. Next time Bob observes a message from Alice on a similar topic to the one in the article (the stegotext includes words appearing in the article), he can use his dictionary to scan the received message for hidden messages.

5.4.3 Word Choosing

Language Modeling with n-grams

Statistical language models are a popular technique for modeling natural language and are used to assign probabilities to sequences of words. The n-gram model [23] is a type of probabilistic language model for predicting the next word in a sequence of words, in the form of a (n-1)–order Markov model.

The name of the model comes from the fact that n-grams are used to calculate probabilities of sequences of words. An n-gram is a sequence of n words derived from a linguistic source. For example, assume the sentence “the sun is shinning”; the 2-grams derived from this sentence are “the sun”, “sun is”, “is shinning”. The *count* of a given n-gram is the number of its occurrences inside a training corpus.

In an n-gram language model, the probability $\mathbb{P}(w_1,\ldots,w_M)$ of observing the sequence of words $w_1,\ldots,w_M$ is approximated as follows:

$$\mathbb{P}(w_1,\ldots,w_M) = \prod_{i=1}^{M} \mathbb{P}(w_i|w_1,\ldots,w_{i-1}) \approx \prod_{i=1}^{M} \mathbb{P}(w_i|w_{i-(n-1)},\ldots,w_{i-1})$$

In other words, the underlying assumption is that the probability of observing the word w_i, given that words $w_1, \ldots, w_{i-1}$ have been observed in the past, can be approximated by the probability of observing it considering only a shorter period of the past, i.e., considering only the preceding $n-1$ words.

The conditional probabilities, i.e., the probabilities of the n-grams, are calculated using maximum likelihood estimates (MLE), based on the n-gram frequency counts:

$$\mathbb{P}_{ML}(w_i | w_{i-(n-1)}, \ldots, w_{i-1}) = \frac{\text{count}(w_{i-(n-1)}, \ldots, w_{i-1}, w_i)}{\text{count}(w_{i-(n-1)}, \ldots, w_{i-1})}$$

In our scheme, we build an n-gram model L_R based on a text corpus R; we use R to extract the possible n-grams, count their occurrences and compute their probabilities. Note that R is the same corpus we use to build the dictionary $\mathcal{D}_R$.

Smoothing

An obvious problem with using MLE for approximating the n-gram probabilities is that the n-grams that have not been observed within a training corpus will be assigned a zero probability. Given that natural language is highly diverse, even if we consider a very large training corpus for deriving the counts of the n-grams, the probability of having unseen n-grams (n-grams that may appear in natural language but *not* in a specific training corpus) is rather high. For this reason, n-gram model probabilities are not in practice directly calculated from the n-gram frequency counts, but instead by using *smoothing* techniques, designed to address this problem.

There exists a variety of smoothing methods in the literature [34], that build on different ideas for approximating the probabilities of unseen n-grams. We use the Witten-Bell smoothing technique, which belongs to the family of methods that use linear interpolation between higher and lower order n-gram counts in order to assign probabilities to unseen n-grams. Our choice is motivated by the results of the empirical study in [34], where it is demonstrated that interpolated models outperform other techniques on small training corpora with low counts. We also use the parametric version of the Witten-Bell algorithm as suggested in [26].

Encoding n-gram Models as FSAs

We use the OpenGrm library [78], a popular and powerful C++ library, that builds on the functionality of the OpenFst library [77] for generating n-gram language models encoded as cyclic weighted finite-state automata (FSA). More precisely, a generated n-gram model is represented in the form of an *acceptor*, i.e., an automaton with the input and output labels of a transition being equal. Finite-state acceptors are used to represent sets of strings; in the case of n-gram models, the set of n-grams observed in a corpus.

To build an n-gram model L_R we use a text corpus R. In the produced model, every sentence in R corresponds to a valid path in the FSA. Every transition in the automaton is associated with a weight that represents the probability of the transition. For every valid path in the automaton, an associated probability can be computed as the accumulation of the transition weights along the path. As discussed above, we appropriately smooth the model which means that there exists valid paths in the model that do *not* correspond to any sentence in corpus R; they model unseen word sequences based on the statistics of the n-grams observed in corpus R. The precise details of the n-gram format in the OpenGrm library are presented in [81].

The OpenGrm library provides an efficient implementation for *intersecting* two acceptor FSAs. The result of the intersection between two acceptors is an acceptor in which strings that are accepted by both automata are retained. We note this useful operation since we use it in the task of selecting words from a bin sequence with the help of an n-gram model L_R, as we present next.

Word Choosing Algorithm

Given a bin sequence $\mathcal{B}_1,\ldots,\mathcal{B}_m$ of m elements, and a language model L_R:

1. The sequence $\mathcal{B}_1,\ldots,\mathcal{B}_m$ is parsed into $\lceil\frac{m}{k}\rceil$ sub-sequences, each of k elements long.

2. For every sub-sequence, a corresponding FSA is generated; we refer to this FSA as the *bin model* associated with the given sub-sequence of bins:

   (a) For every bin $\mathcal{B}_i$, a state S_i is created. A state S_{k+1} is also created.

   (b) For every word $w_j \in \mathcal{B}_i$, a transition is added from state S_i to destination state S_{i+1}. The transition is decorated with input label w_j and output label w_j.

   (c) For every state S_i and for every word $w_r \in \mathcal{W}_S$, a transition is added from state S_i to destination state S_i (a self-loop). The transition is decorated with input label w_j and output label w_r.

3. For every bin model generated, a sequence of $q \geq k$ words is produced:

   (a) The bin model gets *intersected* with the language model L_R; the resulting FSA is referred to as the *combined model*.

   (b) The shortest path in the combined model is computed, i.e., the path with the highest probability. The output labels of the shortest path is the sequence $w_{i1},\ldots,w_{iq}$

The final stegotext $w_1,\ldots,w_{q'}$, with $q' \geq m$, is formed as the concatenation of the sequences $w_{i1},\ldots,w_{iq}$, in the order they were produced.

Key Points

The FSA constructed for a given sub-sequence of bins (a *bin model*), is an acceptor that represents a set A of all strings of words that can be derived by performing the cartesian product between the bins of the bin sub-sequence. In other words, all the possible combinations of the bin words of the sub-sequence. This acceptor also represents a set B of strings of words, which is a proper superset of set A. Set B includes all the strings of A augmented by an arbitrary number of stop words (words from set $\mathcal{W}_S$), placed in-between the bin words and at arbitrary places.

The *combined model* model is an acceptor that represents strings of words accepted by both the language model and the bin model. These strings are essentially the ones appearing in the language model and (a) contain words from the bins in the given sub-sequence (in the order of the sub-sequence), (b) may or may not include stop words. The probabilities of L_R are retained in the combined model; by selecting, thus, the most probable path we select the most probable string of the combined model that satisfies (a) and (b), and also is the most likely to appear in natural language (according to language model L_R).

We parse a given bin sequence into sub-sequences of length k bins, and select words from each sub-sequence using the language model, independently among sub-sequences. As the length of the bin sequence m increases, the stricter our requirements from the language model become: we seek long strings of words containing specific words in a specific order. The probability of succeeding in finding one drops dramatically as m increases, especially when considering corpora R of moderate sizes for building L_R. We introduce the parameter k to tune this effect. Finally, the stegotext, i.e., the sequence of stego words $w_1, \ldots, w_{q'}$, is possible to be of length $q' \geq m$. This is because there might exist a number of stop words in-between the bin words selected.

5.4.4 Hidden Message Retrieval

As described in section 5.4.2, the way we build the dictionary can be viewed as "introducing noise" in the communication channel between Alice and Bob. In this section, we describe the steps performed by the decoder of Bob in order to retrieve the hidden message from a received stegotext. Our decoder aims in removing the "noise" introduced by the encoding process.

Main Ideas and Examples

Assume that Bob receives a stegotext consisting of three stego words w_1, w_2, w_3, and assume also that each word appears in two bins of Bob's dictionary $\mathcal{D}$: w_1 appears in bins $\mathcal{B}_1, \mathcal{B}_2$, w_2 in bins $\mathcal{B}_3, \mathcal{B}_4$ and w_3 in bins $\mathcal{B}_5, \mathcal{B}_6$. There exist in total eight possible bin sequences, out of which Bob has to infer the one that is the *most probable* to have been sent by Alice.

A naive, but effective, approach that Bob could apply would be to convert each of the eight bin

sequences into messages and then decide, by *visually observing* the content of each message, which one was originally sent by Alice. Given that in the English language not every character combination is possible, i.e., not every compressed bit sequence and, thus, not every bin sequence is possible, it might be that out of the eight bin sequences that Bob possesses, only one yields a valid English message. The obvious problem with this naive approach is that as the length of the stegotext and the number of bins that a word appears in increase, it becomes impractical because (a) Bob has to go through possibly hundreds of decoded messages to discover the one sent, (b) each and every bin sequence, out of which the majority is most probably gibberish, has to be decoded, thus adding operational complexity.

As an alternative approach Bob could start decoding only *partially* the bin sequences and gradually reject some of these as *improbable*. The main idea here is that the decoder of Bob does not have to decode each sequence till the end, but instead break the decoding process into steps; at each step the decoder assigns probabilities to the sequences, using only the part of the sequence available up to this step, and continues to decode the most probable sequences at the next step. At the final step, the decoder outputs the message corresponding to the bin sequence with the highest probability.

Notation

Given a bin word w_i and a dictionary $\mathcal{D}$ with bins $\mathcal{B}_0,\ldots,\mathcal{B}_{2^b-1}$, we define a set $\mathcal{A}_i$ as the set of bins in which word w_i appears. Clearly, the size of a set $\mathcal{A}_i$ is not fixed and depends on the dictionary $\mathcal{D}$. Given a sequence of bin words $w_1,\ldots,w_m$ there exists a unique sequence $\mathcal{A}_1,\ldots,\mathcal{A}_m$.

We define a set of states $\mathcal{S}_t$, $\forall\, t \in \{1,\ldots,\lceil\frac{m}{r}\rceil\}$ and $r \leq m$, such that:

$$\mathcal{S}_t = \mathcal{A}_{r(t-1)+1} \times \mathcal{A}_{r(t-1)+2} \ldots \times \mathcal{A}_{r(t-1)+r},$$

where $\times$ denotes the cartesian product over sets. We refer to r as the *grouping factor*.

By construction, every state in a set $\mathcal{S}_t$ is a distinct sequence of r bins. We refer to state $i \in \mathcal{S}_t$, as state i at *step* t. Each state in every step is associated with a *state probability* ϵ_i.

We define a *transition probability* a_{ij}, between two states $i \in \mathcal{S}_t$ and $j \in \mathcal{S}_{t-1}$, such that:

$$a_{ij} = \mathbb{P}(s_t = i \mid s_{t-1} = j)$$

where s_t denotes the state at the current step and s_{t-1} the state at the previous step.

Decoding Algorithm

Given a dictionary $\mathcal{D}$ and a sequence of bin words $w_1,\ldots,w_m$, the decoder performs the following steps:

1. For every word w_i, a set $\mathcal{A}_i$ is constructed.

2. From the sequence $\mathcal{A}_1, \ldots, \mathcal{A}_m$ the sequence of sets $\mathcal{S}_1, \ldots, \mathcal{S}_T$, is constructed, where $T = \lceil \frac{m}{r} \rceil$, for some grouping factor $r \leq m$.

3. For every state $i \in \mathcal{S}_1$, its *initial* state probability is computed: $\epsilon_i = \mathbb{P}(s_1 = i)$

4. For every state $i \in \mathcal{S}_t$, $1 < t \leq T$, its state probability ϵ_i is computed:

$$\epsilon_i = \max_{j \in \mathcal{S}_{t-1}} \epsilon_j \, a_{ij}$$

Once all the state probabilities have been computed, the most probable state sequence $s_1^*, \ldots, s_T^*$ is derived as follows:

$$s_T^* = \arg\max_{i \in \mathcal{S}_T} \epsilon_i$$

$$s_{t-1}^* = \arg \max_{j \in \mathcal{S}_{t-1}} \epsilon_j \, a_{j s_t}$$

Finally, the estimation of the most probable message sent is computed as follows:

$$m^* = C^{-1}(D^{-1}(s_1^*, \ldots, s_T^*)).$$

Probabilities & Approximations

We assume that the state and transition probabilities can be approximated by the probabilities given by an m-order Markov model of the English characters. In other words, we use probabilities of sequences of English characters to approximate the probabilities of bin sequences, i.e., of the states at each step t.

Recall that a bin sequence can be converted back to a bit sequence using a dictionary $\mathcal{D}$, and consecutively to a message in English using a Huffman codebook $\mathcal{C}$. We assume a m-order Markov model M over characters $x \in \mathcal{X}$, where $\mathcal{X}$ denotes here all the printable ASCII characters.

The initial state probabilities are computed as follows:

$$\begin{aligned}
\epsilon_i = \mathbb{P}(s_1 = i) &\approx \mathbb{P}(\mathcal{B}_1, \ldots, \mathcal{B}_r) \\
&= \mathbb{P}(D^{-1}(\mathcal{B}_1, \ldots, \mathcal{B}_r)) \\
&= \mathbb{P}(C^{-1}(D^{-1}(\mathcal{B}_1, \ldots, \mathcal{B}_r))) \\
&= \mathbb{P}(x_1, \ldots, x_n) \\
&= \prod_{j=1}^{n} \mathbb{P}(x_j | x_{j-(m-1)}, \ldots, x_{j-1})
\end{aligned}$$

The transition probabilities are computed as follows:

$$\begin{aligned} a_{ij} = \mathbb{P}(s_t = i \mid s_{t-1} = j)) &\approx \mathbb{P}(\mathcal{B}_{r+1}, \ldots, \mathcal{B}_{r'} \mid \mathcal{B}_1, \ldots, \mathcal{B}_r) \\ &= \mathbb{P}(D^{-1}(\mathcal{B}_{r+1}, \ldots, \mathcal{B}_{r'}) \mid D^{-1}(\mathcal{B}_1, \ldots, \mathcal{B}_r)) \\ &= \mathbb{P}(C^{-1}(D^{-1}(\mathcal{B}_{r+1}, \ldots, \mathcal{B}_{r'})) \mid C^{-1}(D^{-1}(\mathcal{B}_1, \ldots, \mathcal{B}_r))) \\ &= \mathbb{P}(x_{n+1}, \ldots, x_{n'} \mid x_1, \ldots, x_n) \\ &= \prod_{j=n+1}^{n'} \mathbb{P}(x_j | x_{j-(m-1)}, \ldots, x_{j-1}) \end{aligned}$$

Key Points

We make two assumptions: (a) The message sent by Alice is valid English language, (b) Bob has at his disposal a Markov model M that accurately models the English language. The first assumption implies that messages with typos, very rare words, loans from spoken language, etc. may be very difficult or impossible to retrieve. The second assumption implies that the Markov model of the English characters has been trained over a large English text corpus and also for sufficiently large values of m, i.e., character sequences. In section 5.6, we evaluate the performance of our decoder design, using a 6-order Markov model and we demonstrate that it is possible to achieve good performance.

5.4.5 Parameter Selection

In this section we derive an upper bound on the value of parameter b, given a corpus R and a Huffman codebook $\mathcal{C}$, under a complexity constrain at the decoder side. We would like to maximize the value of b so as to maximize the covert rate of our encoder.

Assume we use a Huffman codebook $\mathcal{C}$, with average codeword length C_{avg}. Then the expected number of characters [3] in a state s_t would be $\frac{b \cdot r}{C_{avg}}$, where r is the grouping factor used in the decoding process. In order to compute the state probability ϵ_{s_t}, we need at least $\frac{m}{2}$ characters within a state, where m is the order of the Markov model of English characters used. Therefore, a lower bound for r is given by the following equation:

$$r \geq \frac{m/2 \cdot C_{avg}}{b} \tag{5.1}$$

Note that the higher the value of r, the higher the number of characters within a state, but also the higher the number of states at a given step of the decoding process.

The state space at time t, i.e., the cardinality N_t of the set $\mathcal{S}_t$, can be upper bounded as follows:

[3] If a mixed Huffman code is used, the expected number of characters is higher since a symbol can be a word and may contribute more than one character per codeword.

Corpus	Description
wikipedia	Various texts extracted [53] from Wikipedia. 2615 texts, 9M words.
brown	The Brown corpus [1]; texts from 500 sources categorized into 15 genres (e.g., news, editorial, reviews, hobbies etc.), totaling 1.15M words.
reuters	The Reuters corpus [8]; a collection of 10K news documents that appeared on Reuters newswire in 1987, 1.3M words, various categories.
novels	Two publicly available novels through Project Gutenberg [6]; "A tale of two cities", by C. Dickens, and "Siddhartha", by H. Hesse, totaling 54K words.
overheard	Conversations overheard in the city of New York [4].

Table 5.3 – Useful text corpora.

$$N_t = |\mathcal{A}_{r(t-1)+1}| \cdot |\mathcal{A}_{r(t-1)+2}| \cdot \ldots \cdot |\mathcal{A}_{r(t-1)+r}| \leq \left(f_{max} \cdot p \cdot 2^b\right)^r, \tag{5.2}$$

where $f_{max} \cdot p \cdot 2^b$ (with $f_{max} = \max_i f_i$ and f_i denoting the frequency of occurrence of bin word w_i in corpus R) is the maximum number of bins any bin word may appear in dictionary D_R, i.e., the maximum cardinality of any set $\mathcal{A}_i$, assuming that we use a corpus for which $f_{max} \cdot p < 1$.

Note that the number of states at time t grows exponentially with the grouping factor r. The same holds for the total number of transitions computed between steps $t-1$ and t.

Assume that, at the decoder there exists a constrain N_{max} on the maximum number of states per step t, i.e., $N_t \leq N_{max}$. Taking the minimum r from Eq. 5.1 and the maximum N_t from Eq. 5.2, we upper bound b as follows:

$$b \leq \frac{m/2 \cdot \mathcal{C}_{avg} \cdot \log_2(f_{max} \cdot p)}{\log_2 N_{max} - m/2 \cdot \mathcal{C}_{avg}} \tag{5.3}$$

5.5 Steganalysis Methods

In this section we describe two steganalysis methods, for attacking our steganographic scheme, that aim in detecting unnaturalness in the stegotexts and flag them as suspicious. The methods we design work at the granularity of a sentence, as opposed to the majority of existing steganalysis techniques [35, 102, 36] that operate at a text granularity and are usually ineffective for short texts. We assume that an adversary Eve, as modeled in section 5.2, has the possibility to launch these attacks on each message that crosses her infrastructure. We note

that the methods we design may not be the optimal way to detect unnaturalness in a sentence, but we believe our approach is a first step toward solving a non-trivial NLP problem.

Stanford Parser

The Stanford Parser [47, 65] is a popular natural language parser, developed from the Stanford Natural Language Processing Group, and its purpose is to infer the grammatical structure of sentences. It falls into the category of probabilistic parsers that use knowledge on language, acquired through hand-annotated sentences, for producing the most likely analysis of new sentences. The lexicalized PCFG parser for the English language, included in the distribution of the parser, has been trained on the collection of syntactically annotated data from the Penn Treebank Project [5]. The parser produces various information on a tested sentence, including the part-of-speech (POS) tagging and the syntactical dependencies of its words, the grammatical analysis of the sentence in a form of a tree, the probability (or *score*) of the most likely parsing of the sentence, etc.

We use the score and the POS tags produced on tested sentences to construct statistical tests for detecting unnaturalness in sentences.

5.5.1 Classification Using Prediction Intervals

A prediction interval (PI) is an interval that can be computed based on observations $z_1, \ldots, z_n$, such that a future observation will take a value within this interval with probability γ. They are practically used when the distribution, or its parameters, from which the observations take values, is unknown. Given an iid sequence $z_1, \ldots, z_n, z_{n+1}$, a prediction interval at level $\gamma = 1 - \alpha$ can be computed, using Theorem 2.4.1 from [59], as follows:

$$\left[z^n_{\lfloor (n+1)\frac{\alpha}{2} \rfloor},\ z^n_{\lceil (n+1)(1-\frac{\alpha}{2}) \rceil} \right],$$

where $z^n_{(1)}, \ldots, z^n_{(n)}$ denotes the ordered statistic of $z_1, \ldots, z_n$ and $\alpha \geq \frac{2}{n+1}$.

We assume that our adversary has access to a large quantity of natural language (NL) texts that she uses to train her attack algorithm. More precisely, the adversary uses a parser to infer the scores of the sentences in the natural language texts, and computes a prediction interval at level $\gamma = 0.95$ based on the observations of the scores, normalized by the sentence length. The observations have been checked for the iid property, using the Ljung-Box statistical test [63], and have been appropriately down-sampled when the property did not hold, before being used to compute the prediction interval.

The adversary can perform the following simple test, in order to flag on-the-fly an observed message as suspicious or not: she uses a parser to infer the scores, normalized by length, of the sentences in the message, and checks if they fall within the pre-computed prediction interval.

If many of them fall outside, then this message is flagged as suspicious.

We imitate the behavior of the adversary: we first use the Stanford Parser to infer the normalized scores of approximately 155K sentences, derived from a *training* corpus (concatenation of the *wikipedia*, *reuters*, *brown* and *novels* corpora – see Table 5.3 for a description of each corpus), and then we compute the prediction interval. We also create three different *testing* sets of sentences:

(a) 10K natural language sentences excluded from the initial training corpus.

(b) 10K *random* sentences, of length between 5 and 40 words, produced using an on-line random sentence generator [4] [7].

(c) 10K *pseudo-random* sentences, produced by performing random walks in a 5-gram language model encoded as an FSM and trained on the *brown* corpus.

We find that 98% of the observations from testing set (a) falls within the intended prediction interval, i.e., they are classified as NL sentences, while only 4% of the scores from set (b) do; the prediction interval test works rather good in telling apart NL to random sentences. Interestingly, for set (c) we get 83% of the observations being inside the prediction interval; the partial underlying structure of the sentences makes the scores being very close to natural language scores, and hence confusing the classifier that, nevertheless, still classifies a 17% of these as non NL.

In the evaluation section, we use the prediction interval technique to classify the sentences of the stegotexts. We refer to this technique as the PI classifier.

5.5.2 Classification Using SVMs

Support vector machines (SVMs) are supervised learning models with associated learning algorithms used for data classification. Given a set of training observations $\mathbf{z}_1, \ldots, \mathbf{z}_n$, with $\mathbf{z}_i \in \mathbb{R}^d$, each marked for belonging to a category (or *class*), an SVM learning algorithm computes a decision function that predicts the class of new observations. An SVM is a representational model of the training observations as points in space, mapped such that observations from different classes are separated by a clear gap that is as wide as possible. In linear classification, the decision function is the optimal hyperplane in $\mathbb{R}^d$. In non-linear classification, a non-linear decision function is computed by projecting the observations through a function ϕ to a space with a higher dimension, commonly referred to as the feature space F, and by computing the optimal hyperplane in F.

The One-Class SVMs [87] is a category of SVMs, where all the training observations belong to only one class. Such a model separates the training points from the origin (in feature space

[4] The sentences are random only in terms of grammar and syntax; the words used are normal English words.

F) and maximizes the distance from this hyperplane to the origin. The result is a binary decision function which captures regions in the input space where the probability density of the training observations lives. The function returns 1 in a "small" region (capturing the training points) and -1 elsewhere. If for a new observation the decision function returns -1, this observation is labeled as out-of-class. In Appendix B, we include the mathematical description of the One-Class SVMs.

We assume that our adversary, Eve, uses an One-Class SVM to identify if a given observed message is a stegotext or not. As previously, we assume that Eve has at her disposal a large quantity of NL sentences that she uses to train her SVM. More precisely, for each sentence i she uses a parser to infer its score and its POS tags, and she constructs the following vector:

$$\mathbf{z}_i =< \text{score, \#nouns, \#verbs, \#adverbs, \#adjectives, length} >,$$

that is, a feature vector with the score, the number of nouns, verbs, adjectives and adverbs (each normalized by the sentence length in words) and the length in words, as features.

Eve uses her observations $\mathbf{z}_1, \ldots, \mathbf{z}_n$, to train an One-Class SVM, using a Gaussian Radial Base Function (RBF) kernel function, and $\nu = 0.10$. The feature vectors are scaled properly per dimension [5]. For every newly observed message she analyzes its sentences and decides if the message should be flagged as suspicious or not.

We imitate the behavior of Eve: we use the sentences in the *training* corpus, as defined before, and the Stanford Parser to infer the scores and the POS tags, in order to construct the training set of observations. For sanity check, we test the performance of our SVM using the *testing* sets of sentences (a), (b), (c).

We find that 95% of the sentences in set (a) are classified as NL. For the sentences in set (b) this percentage is only 3%: the classifier successfully classifies the random sentences as out-of-class, i.e., as *non* NL sentences. The sentences in set (c) are classified as natural language with a percentage of 65%: the SVM classifier has a better performance regarding the pseudo-random sentences, in comparison to the PI classifier.

In the evaluation section, we use the One-Class SVM classifier described here to classify the sentences of our stegotexts. We refer to this technique as the SVM classifier.

5.6 Experimental Evaluation

5.6.1 Encoder

In this section we evaluate the performance of our encoder. Our goal is to answer the following questions: How much effort is required from human users for enhancing our stegotexts? Do

[5] We use the same scale for scaling the feature vectors of the testing sets.

Corpus					Parameters		
Name	Description	$\|\mathcal{W}_R\|$	$\|\mathcal{W}_B\|$	$\|\mathcal{W}_S\|$	b	p	k
dreams	A long-form article from "The Guardian" about dreams and academic anxiety.	425	277	148	6	5	4
animals	A collection of stories for kids about animals.	1600	823	777	6	15	3
facebook	A long-form article from BuzzFeed about Facebook.	1423	461	962	5	15	3

Table 5.4 – Input parameters for the MTurk experiments.

our design choices assist them into doing so? What is the covert rate achieved by our scheme? Finally, do the stegotexts and the user enhanced stegotexts pass our steganalysis attacks?

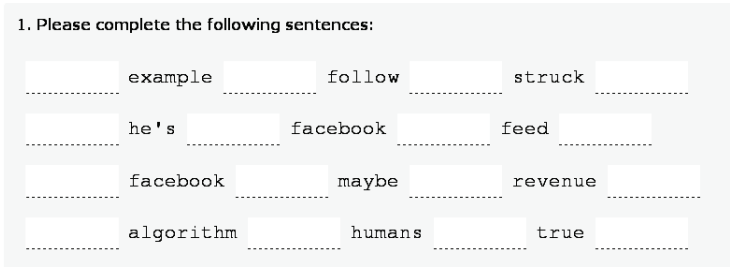
MTurk Platform

Amazon's Mechanical Turk (MTurk) [15] is a on-line platform for enabling the use of human intelligence in various tasks. Businesses, researchers or individuals (referred to as Requesters) post short tasks, known as Human Intelligence Tasks (HITs), which become available through the MTurk platform to a pool of users, known as Workers. The Workers have the possibility to browse among the available tasks and choose the ones they wish to complete in exchange for a payment from the Requester.

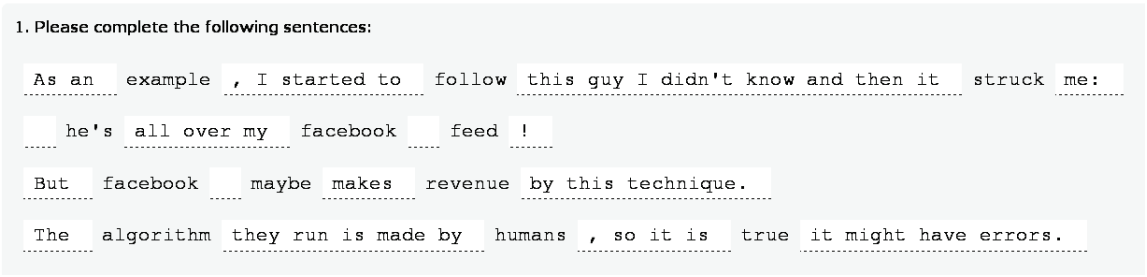
We use the MTurk platform for evaluating the effort required by human users to turn various stegotexts produced by our encoder into meaningful natural language texts. We prepare a set of HITs, with each HIT consisting of a set of *incomplete* sentences that we ask the users to complete so as to become meaningful. Each set of incomplete sentences essentially consists of the stego words, produced by our encoder, that embed a given hidden message. We provide an interface for completing sentences, that hinders the users from modifying the stego words and altering their order, but allows them to introduce any word or punctuation mark among the stego words, as they wish. In Fig. 5.3 we show our interface and an example of sentence completion; the dashed lines between the displayed words represent the gaps into which the user can insert words.

In Table 5.4 we show the corpus and the parameter values we use in each experiment. For a corpus R, we produce a dictionary D_R and a 5-gram language model L_R, and we encode 50 different messages of length between 4 and 15 words, derived from the *overheard* corpus. We encode the messages three times, each time using one of the following techniques:

1. *Random*: Instead of using a language model for selecting words from bins, we just pick uniformly at random a bin word from each bin of a given bin sequence.



(a) Before completion.



(b) After completion.

Figure 5.3 – The user interface for sentence completion in MTurk.

2. *N-grams*: We select words from the bins, using the language model L_R, as described in section 5.4.3. We do not output the stop words produced by the encoder, only the bin words.

3. *Complete*: The words from the bins are selected using L_R. The resulting stegotext consists of the bin words and the stop words selected by the encoder.

Note that for the third technique we modify appropriately the user interface for the MTurk experiments, so as to display the stop words pre-inserted in the gaps. The users are free to delete/reuse/modify these at their will.

In every experiment we use the same mixed Huffman codebook, in which we include 330 frequently used English words, as explained in section 5.4.1, with average codeword length 12.73 bits. For each dictionary, we computed the value of parameter b using Eq. 5.3. We also note that we use either $k = 3$ or $k = 4$, since initial results from the MTurk platform suggested that users find it rather difficult to complete sentences of 5 words and more.

Finally, we mention that our payment for each HIT was \$1.00, which translated into approximately \$0.2 per minute and was well above the average MTurk wage; this an important aspect of the study since Workers are usually negatively biased regarding the difficulty of a task, if the ratio of the time spent for a HIT over the payment is large. We restricted the maximum number of HITs that a user could do from each experiment to 3.

Performance Metrics

We evaluate the amount of user-effort required for completing a HIT using the following metrics:

- *Difficulty*: The users are asked to evaluate the difficulty of a HIT, once completing it, by assigning a score between 1 and 5, with 1 representing a "Very Easy" task and 5 a "Very Difficult" task.
- *Completion time*: The amount of time needed for a user to complete a HIT. We measure the time between the moment the user starts reading the given sentences and the moment they declare they are done enhancing them.
- *Extra words*: The number newly inserted words by the user while completing a set of sentences.

We also measure the *covert rate* achieved by our encoder: the amount of hidden bits per word of enhanced stegotext. It is defined as follows:

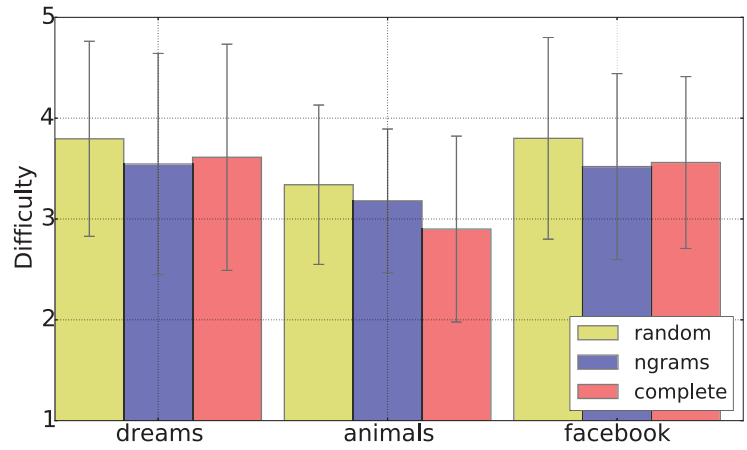
$$Covert\ Rate = \frac{\#\ hidden\ bits}{\#\ words\ in\ enhanced\ stegotext}$$

Results on User-effort

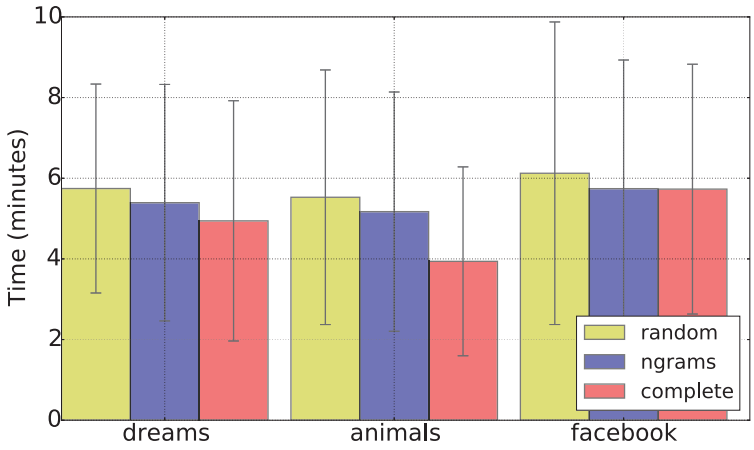
In Fig, 5.4a, 5.4b, 5.4c we demonstrate the average difficulty, completion time (in minutes) and number of user inserted extra words, respectively, for each corpus and for each encoding technique. The values reported are averaged over the number of HITs and the error bars indicate the standard deviation of the measurements.

First, we observe that for all three corpora the *ngrams* and *complete* approaches outperform the *random* one, indicating that our word-choosing approach offers some benefits in assisting the user to complete the sentences. Even though the users report a level of difficulty comparable to *random* (Fig. 5.4a), the completion time they need and the number of words they introduce is always smaller. Especially for the case of *complete*, where stop words are pre-inserted between stego words, the users introduce up to roughly half the amount of words they do for the *random*, and usually in less time.

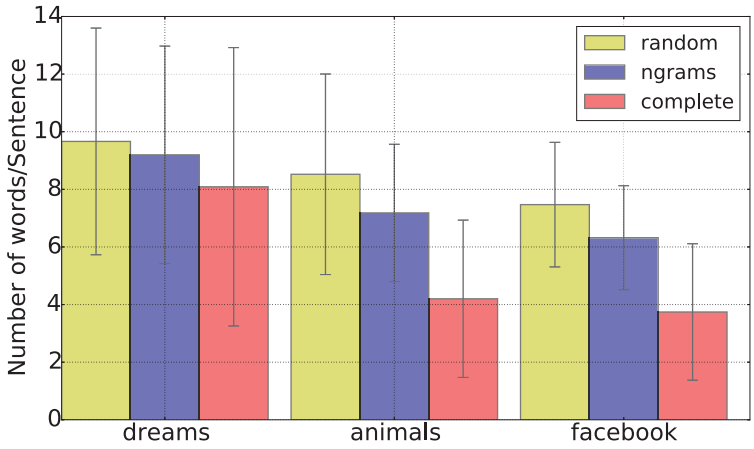
Second, we observe that there exist differences among corpora. The *dreams* corpus seems to require more effort than the others. This is mainly due to three reasons: (a) The vocabulary of the *dreams* corpus was more domain specific, in comparison to the other two; the task of combining words of this domain in sentences was more difficult for the average user. The *animals* and *facebook* topics are closer to the general knowledge. (b) The dictionary used for *dreams* includes only $p = 5$ words per bin, in contrast to $p = 15$ words per bin used for the other



(a) Average reported difficulty per HIT.



(b) Average completion time per HIT.



(c) Average number of extra words inserted by users per HIT.

Figure 5.4 – Measurements on the user-effort required for completing our MTurk HITs.

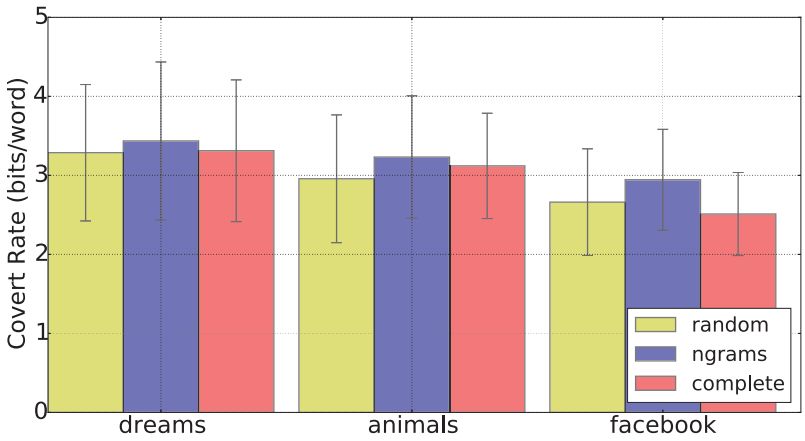


Figure 5.5 – Average covert rate achieved per HIT.

two experiments. By increasing the number of words per bin we increase the probability of finding suitable words from consecutive bins that the users can combine easier in a sentence. (c) The sentences produced for *dreams* included $k = 4$ stego words, whereas for the other two experiments only $k = 3$, which users found easier to complete.

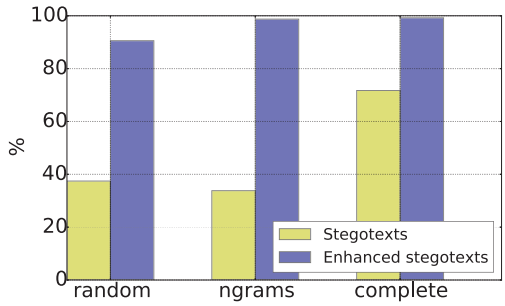
Third, we note that the users report, on average, a difficulty score between 3 and 4, which translates to "Moderate" to "Difficult" task. This fact should be explained in the context of the MTurk platform: the opinion of the users regarding these HITs, is possible to be biased from their opinion on other MTurk HITs. The HITs including writing are usually regarded as the most demanding ones in comparison to others available in the platform. Out of all the HITs we published, a user would do on average 5 of them, and an average MTurk user does tens of them in a day. Therefore, we believe that the difficulty reported is mainly in comparison to other HITs and does not reflect purely the opinion of the users on our HITs.

Finally, we mention here that we also measured the rate at which a user inserts words that appear in the dictionary of the encoder. Note that our user interface in MTurk, does not hinder the users from inserting bin words, as explained in section 5.3, allowing us to measure this rate. We observed that by excluding from a dictionary words that are in the top 100 most frequently used words in English, we get an approximate insertion rate of 1 word per sentence. That is, in every sentence that a user enhances he will introduce one bin word. This rate indicates that such insertions can be prevented by a smart user interface that consults the user to avoid the use of this word without decreasing dramatically the user experience.

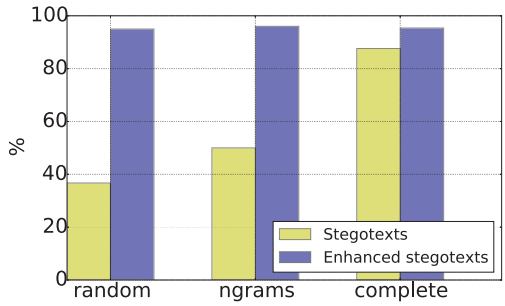
Results on Covert Rate

In Fig. 5.5 we demonstrate the average covert rate achieved, in bits per word, for all corpora and encoding techniques.

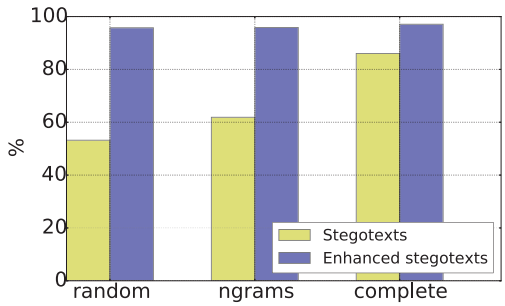
First, we observe that on average the covert rate achieved by our scheme is roughly 3 bits per



(a) The *dreams* corpus.



(b) The *animals* corpus.



(c) The *facebook* corpus.

Figure 5.6 – Percentage of sentences classified as NL by the PI classifier.

word of enhanced stegotext. This means that for sending out a hidden message consisting of 5 words, each one consisting on average of 5.5 characters, we need an enhanced stegotext of around 73 words. This is well above the covert rate achieved usually from existing text-based steganographic approaches (see also section 5.7 on related work).

Second, we observe that the *ngrams* slightly outperforms the *random* approach; this is due to the fact that the users introduce less words per sentence while enhancing. We note that the rate for *complete* appears to be lower than the *ngrams* because the final length of the enhanced stegotext includes the newly inserted words by the user (as shown in Fig. 5.4c) plus the already pre-inserted words, that the user may or may not use.

Finally, we observe the effect of the values of parameters b and k on the covert rate. For *dreams* and *animals* we used $b = 6$ whereas for *facebook* $b = 5$; for a given message, the higher the b, the smaller the corresponding bin sequence length, the smaller stego text produced, the less sentences to enhance. Note that despite the users inserting less words per sentence for *facebook* (Fig. 5.4c), since for $b = 5$ the users have to enhance more sentences, the total rate drops. The difference between *dreams* and *animals* is explained due to the different values of k used ($k = 4$ and $k = 3$, respectively): the higher the value of k, the less sentences the user needs to enhance, resulting into an increased overall rate.

Results on Steganalysis Attacks

We investigate the performance of the attacks described in section 5.5. In Fig. 5.6 and 5.7 we demonstrate the results for the PI and the SVM classifiers, respectively. For each experiment configuration and each encoding technique used, we show the percentage of sentences from stegotexts and enhanced stegotexts that were classified as natural language (NL) sentences.

First, we look at the performance of the PI classifier (Fig. 5.6). We observe that, before enhancement, the *complete* approach clearly outperforms the other two: the sentences produced with the help of the language model, that include stop words, are resembling NL sentences, at least while using this specific classifier. Also, the selection of words from the bins with *ngrams* seems to produce more NL sentences than when choosing randomly. Given that the MTurk users write NL sentences, the majority (more than 95% in all cases) of the sentences in the enhanced stegotexts are classified as natural. Of course, the PI classifier has limited capabilities, and the results indicate that, by just looking at the sentence's score, we cannot detect unnaturalness in pure stegotexts with high confidence.

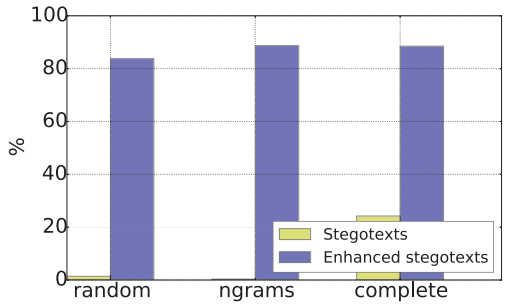
Next, we look at the performance of the SVM classifier (Fig. 5.7). Clearly, the SVM classifier performs better in detecting unnaturalness since a very high percentage of sentences from the *random* and the *ngrams* stegotexts are classified as *non* NL; this is expected since the stego words produced by these two techniques are not truly NL sentences, especially when using random selection of words from bins. Again, the sentences we produce with the *complete* technique are closer to NL sentences, but not more than 50%. This is another indication that this technique produces stegotexts that require less enhancing effort. Regarding the enhanced stegotexts, we again observe that a large fraction of these are classified as NL, with the percentage being around 90% – a bit lower compared to the PI classifier; this is expected since the SVM classifier is more powerful and, thus, able to detect some weird sentences composed by the users, who did not put the maximum effort into enhancing the stegotexts.

5.6.2 Decoder

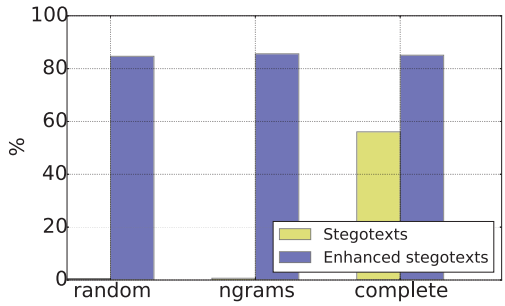
The decoder we presented in section 5.4.4, is a type of probabilistic decoder that tries to infer the most probable message sent by Alice, out of all the possible decodings available. We implemented our decoder design and, in this section, we evaluate its performance.

As explained in section 5.4.4, our decoder uses an m-order Markov model of the English characters to approximate the probabilities of states and transitions. We build such a 6-order model by counting the occurrences of character tuples[6] in a large *training* text corpus and by using these to compute the conditional probabilities of the model. We note that, the compressed data of the model amounts to 43MB.

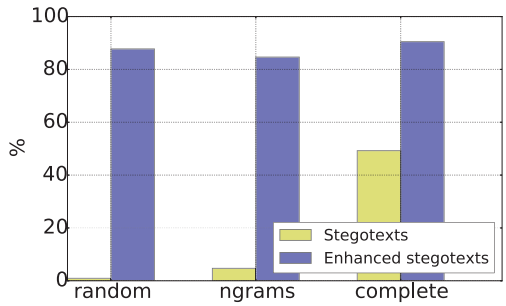
[6] We count the occurrences of all tuples of length l, $\forall\, l \in \{1, \ldots, m+1\}$.



(a) The *dreams* corpus.



(b) The *animals* corpus.



(c) The *facebook* corpus.

Figure 5.7 – Percentage of sentences classified as NL by the SVM classifier.

Performance Metric

We evaluate the performance of our decoder by measuring the *error rate* of decoded messages. It is defined as follows:

$$Error\ Rate = \frac{\#\ erroneously\ decoded\ tokens}{\#\ total\ decoded\ tokens},$$

where as tokens we use *characters* and *words*. I.e., we measure the *character error rate* and *word error rate* of a decoded message.

Results

In Table 5.5 we show the percentage of messages that were correctly decoded, i.e., that had a *zero* error rate, for each experiment, considering all stegotexts produced with each encoding technique. We see that our decoder was able to decode correctly a very high percentage of the encoded messages. We note that this percentage was consistently around 95%, for many off-line experiments we run, indicating that this error is due to the limited capabilities of the 6-order Markov model.

For the few cases of erroneously decoded messages, we demonstrate in Fig. 5.8 and 5.9 the average character and word error rate, respectively. We see that the erroneously decoded

Experiment	Correctly decoded (%)
dreams	96%
animals	93%
facebook	95%

Table 5.5 – Percentage of messages decoded correctly.

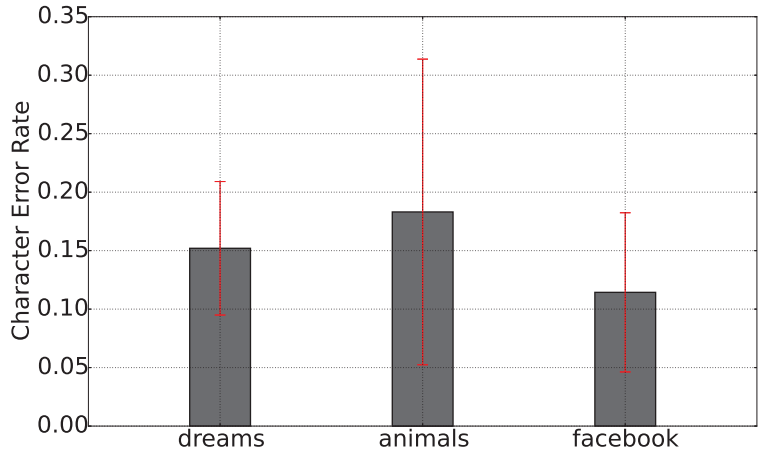


Figure 5.8 – Average character error rate of messages not decoded correctly.

messages are not completely corrupted but only partially, i.e., on average 15% of characters were corrupted, influencing up to 18% of the words in the messages. We note that for the majority of the erroneously decoded messages, the correct message was included in the list of the top ten most probable decodings.

5.7 Related Work

The majority of existing approaches to linguistic steganography follow the traditional strategy of automated covertext modification. Popular such techniques include word substitution [97, 31, 93, 30], sentence structure manipulation [73, 68], phrase paraphrasing [29], semantics transformations [91, 94], hiding information in errors [92], etc. Many of these techniques require access to sophisticated NLP tools, e.g., semantic role parsers, POS taggers, anaphora resolution tools, etc., and large linguistic datasets, e.g., the WordNet lexical database [12], the Web 1T corpus [11] and the Google n-grams database [2]. Moreover, they usually introduce grammatical, syntactic and semantic anomalies in the stegotext, easily detectable by steganalysis methods [90, 35, 102, 36]. Finally, the existing techniques achieve a covert rate of less than 1 bit per covertext word.

Recently, some approaches have appeared that include the “user-in-the-loop”. Closer to our work is the approach presented in [46] and followed by [45], where the effectiveness of the user intervention on the stegotext is demonstrated using human judges. Similarly to us, the

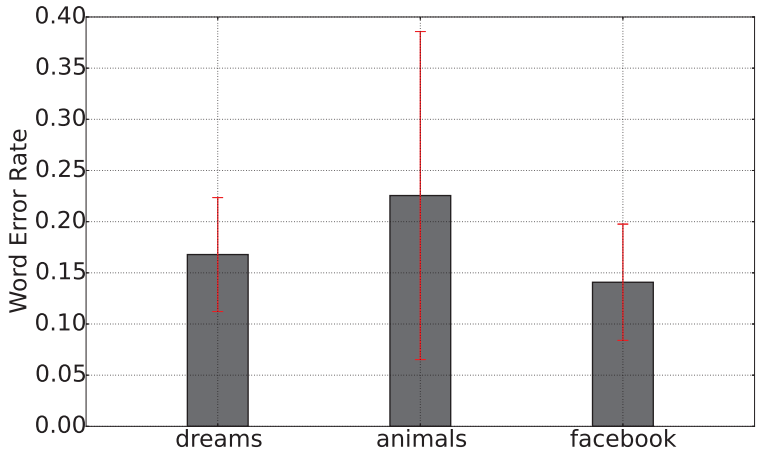


Figure 5.9 – Average word error rate of messages not decoded correctly.

proposed approach consists of mapping the hidden message to a sequence of words which the user modifies to produce text. Differently to us, no significant attention is given in the way the stego word sets are formed or in the way words get chosen from sets, so that less user-effort is required. In addition, no thorough evaluation is provided on quantifying the amount of effort needed, and on measuring the achieved covert rate; a demonstrated example implies a covert rate of 0.5 bits per word. In [96], the authors propose a technique for Twitter that embeds 4 bits per tweet by modifying previous tweets, partially aided by the human user.

An interesting technique is the on-line tool Spammimic [9] that maps the hidden message to a set of natural language sentences that are eventually combined to form a spam e-mail. Alice can copy-paste the output of the tool and send it as a normal e-mail to Bob, who, upon reception, uses the on-line decoder to retrieve the hidden message. Although it requires zero user-effort, Spammimic has some significant drawbacks. First, modern e-mail providers usually filter out spam e-mails, so Bob risks to miss it. Second, it is rather unusual that normal users, like Alice and Bob, will exchange spam e-mails, so this action immediately classifies the e-mail as suspicious. Third, the output of Spammimic is rather long for very short input messages. For example, the message “hello” produces an e-mail of 15 sentences and 185 words in total, which yields a covert rate of 0.3 bits per word.

Our approach differs from the existing ones since its design aims in achieving high covert rate, while remaining practical and usable. The “user-in-the-loop” design choice we do may require higher amount of user-effort but it also introduces elements that make robust our stegotexts to sophisticated detection attacks. We believe that privacy-aware users would be willing to put in a reasonable amount of effort in exchange for secretly communicating short phrases and not just a few bits.

5.8 Summary

In this chapter we presented a linguistic steganographic mechanism for enabling two communication parties to exchange private messages such that an adversary, who observes their communication, is not aware of this fact. First, we designed a semi-automated encoder that embeds a hidden message into a sequence of words that the user manually enhances to produce natural language text. Our encoder does not require off-line access to large linguistic resources and sophisticated NLP tools and it is easily implementable. Second, we presented two steganalysis methods for detecting unnaturalness in text, that operate at the granularity of a sentence. Finally, we designed a probabilistic decoder suitable for removing the "noise" introduced by the encoding process.

We implemented our steganographic mechanism and we experimentally evaluated its performance. First, we designed and launched an on-line campaign on the MTurk platform, in order to evaluate the amount of effort needed by human users to enhance the output of our encoder. We observed that our design choices assist the human users in their task, without requiring an unreasonable amount of user-effort, and that our approach achieves a good covert rate. Second, we applied the steganalysis methods we designed both on the raw and the enhanced output of the encoder, and we demonstrated that the "user-in-the-loop" helps in hiding the existence of steganography. Finally, we evaluated the performance of our decoder under restricted linguistic resources, and we demonstrated that it performs well.

In summary, the contributions in this chapter are:

- We design a steganographic mechanism for masking hidden messages as innocuous text, in order to enable private communications over untrustworthy messaging providers.
- We evaluate our approach by experimenting with human users through the MTurk platform and by applying steganalysis methods that we design.

A Proofs for Chapter 2

A.1 Proof of Lemma 1

We consider our eavesdropper to be either Eve or one of the participating terminals. Our eavesdropper has two occasions to obtain information about the secret $\mathcal{S}_{ij}$: by overhearing a fraction of the transmitted x-packets in the initial phase; or because a terminal knows the source packets it transmitted. Both these effects are captured in the calculation of the number M_{ij}. Under the theoretical conditions of the erasure channel model, we can approximate these numbers with their average value; Lemma 8 shows that this approximation can become arbitrarily good exponentially fast in N. Given that we use any value M_{ij} smaller or equal to the exact, the following Lemma 7 gives a construction that does not allow the eavesdropper to obtain any information about $\mathcal{S}_{ij}$. □

Lemma 7. *Consider a set of N x-packets, say $x_1, \ldots, x_N$, and assume an eavesdropper, Eve, has a subset of size N_E of the x-packets. Construct $M = N - N_E$ y-packets, say $y_1, \ldots, y_M$, as*

$$Y = AX,$$

*where matrix X has as rows the N x-packets, matrix Y has as rows the $N - N_E$ y-packets, and A is the generator matrix of a Maximum Distance Separable (MDS) linear code with parameters $[N, N - N_E, N_E + 1]$ (*e.g.*, a Reed-Solomon code [64]). Then the M y-packets are information-theoretically secure from Eve, irrespective of which subset (of size N_E) of the x-packets Eve has.*

Proof. Let W be a matrix that has as rows the packets Eve has. To prove that the y-packets are information-theoretically secure from Eve, we must show that:

$$H(Y|W) = H(Y).$$

We can write

$$\begin{bmatrix} Y \\ W \end{bmatrix} = \begin{bmatrix} A \\ A_E \end{bmatrix} X \stackrel{def}{=} BX,$$

where A_E is a $N_E \times N$ matrix of $\text{rank}(A_E) = N_E$, which specifies the N_E distinct x-packets that are known to Eve. A_E is *not known* to us, however we know is that in each row of A_E there is only one 1 and the remaining elements are zero; so all of the vectors in the row span of A_E have Hamming weight (the number of nonzero elements of a vector [64]) less than or equal to N_E. On the other hand, from construction, $\text{rank}(A) = N - N_E$, and each vector in the row span of A has Hamming weight larger than or equal to $N_E + 1$ [64]; thus the row span of A and A_E are disjoint (except for the zero vector) and the matrix B is full-rank, i.e. $\text{rank}(B) = N$.

If the packets x_i have length L, we have that:

$$\begin{aligned} &H(Y|W) = H(Y,W) - H(W) = \\ &= \text{rank}(B)\,L - \text{rank}(A_E)L = (N - N_E)L \\ &= \text{rank}(A)L = H(Y). \end{aligned}$$

□

A.2 Concentration to expected values

Lemma 8. *The values of the random variables M_{ij}, U_{rE}, V_l, as defined in section 2.3.2 and used in Lemma 1, converge exponentially fast in N to their expected values.*

Proof. Consider the random variable U_{rE} denoting the number of x-packets transmitted by T_r and received by both T_i / T_j but not Eve. We use a standard argument to show it concentrates exponentially fast to its average. Define the random variable $\eta_q^{(l)}$ as

$$\eta_q^{(l)} = \begin{cases} 1 & \text{if the } q\text{th } x\text{-packet is received} \\ & \text{by terminals } T_i / T_j \text{ and missed by Eve,} \\ 0 & \text{otherwise.} \end{cases}$$

Then we can write $U_{rE} = \sum_{q=1}^{N} \eta_q^{(l)}$ and we have

$$\mu \triangleq E(U_{rE}) = (1 - \delta_{ri})(1 - \delta_{rj})\delta_{rE} N.$$

For $0 < \epsilon \leq 1$ we can write $\mathbb{P}\left[U_{rE} - \mu \geq \epsilon\mu\right] \leq \exp\left(-\frac{\epsilon^2\mu}{3}\right)$, where in the last inequality we use Chernoff bound [70, Chapter 4]. We can also write, for $0 < \epsilon \leq 1$, $\mathbb{P}\left[U_{rE} - \mu \leq \epsilon\mu\right] \leq \exp\left(-\frac{\epsilon^2\mu}{2}\right)$. Similar arguments hold for the remaining variables in section 2.3.2. □

B Mathematical Formulation of One-Class SVMs

The Support Vector Method For Novelty Detection by Schölkopf *et al.* [87], essentially maps all the data points into a feature space F and separates them from the origin (in feature space F) with maximum margin. Consider a training data set $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$, where $n \in \mathbb{N}$ is the number of observations and $\mathcal{X}$ a compact subset of $\mathbb{R}^d$. Let ϕ be a feature map $\mathcal{X} \to F$, i.e., a map into a dot product space F such that the dot product in the image of ϕ can be computed by evaluating some function

$$K(\mathbf{x}, \mathbf{y}) = (\phi(\mathbf{x}) \cdot \phi(\mathbf{y})).$$

The $K(\mathbf{x}, \mathbf{y})$ is known as the *kernel function*. Popular choices for the kernel function are linear, polynomial, sigmoidal but mostly the Gaussian Radial Base Function (RBF):

$$K(\mathbf{x}, \mathbf{y}) = e^{||\mathbf{x}-\mathbf{y}||^2/c},$$

where c is a kernel parameter and $||\mathbf{x}-\mathbf{y}||^2$ is the dissimilarity measure.

To separate the observation points set from the origin, the following quadratic program is solved:

$$\begin{aligned} \min_{w \in F,\, \xi \in \mathbb{R}^n,\, \rho \in \mathbb{R}} \quad & \frac{1}{2}||w||^2 + \frac{1}{\nu n}\sum_i \xi_i - \rho \\ \text{subject to:} \quad & (w \cdot \phi(\mathbf{x}_i)) \geq \rho - \xi_i \qquad \forall i \in \{1, \ldots, n\} \\ & \xi_i \geq 0 \qquad \forall i \in \{1, \ldots, n\} \end{aligned}$$

The parameter $\nu \in (0, 1)$ characterizes the solution:

- It defines an upper bound on the fraction of outliers (training observations considered as out-of-class)
- It is a lower bound on the number of training observations used as Support Vector.

The decision function (classification) rule for a data point $\mathbf{x}$ is:

$$f(\mathbf{x}) = \text{sgn}((w \cdot \phi(\mathbf{x})) - \rho) = \text{sgn}\left(\sum_i \alpha_i K(\mathbf{x}, \mathbf{x}_i) - \rho\right),$$

where α_i are the Lagrange multipliers, computed by deriving and solving the dual problem. The non-zero a_i's are called the Support Vectors.

Bibliography

[1] The brown corpus. http://www.essex.ac.uk/linguistics/external/clmt/w3c/corpus_ling/content/corpora/list/private/brown/brown.html. Accessed April 6, 2016.

[2] The google books ngram viewer. http://storage.googleapis.com/books/ngrams/books/datasetsv2.html. Accessed April 6, 2016.

[3] Nc-utils: Network coding utilities toolbox. http://lokeller.github.io/ncutils/. Accessed April 6, 2016.

[4] Overheard in new york. http://www.overheardinnewyork.com/. Accessed April 6, 2016.

[5] The penn treebank project. https://www.cis.upenn.edu/~treebank/. Accessed April 6, 2016.

[6] Project gutenberg. https://www.gutenberg.org/. Accessed April 6, 2016.

[7] Randomtext. http://www.randomtext.me/. Accessed April 6, 2016.

[8] Reuters corpora. http://trec.nist.gov/data/reuters/reuters.html. Accessed April 6, 2016.

[9] Spammimic. http://spammimic.com/. Accessed April 6, 2016.

[10] Warp: Wireless open access research platform. https://warpproject.org/trac. Accessed April 6, 2016.

[11] Web 1t 5-gram version 1. https://wordnet.princeton.edu/. Accessed April 6, 2016.

[12] Wordnet: a lexical database for english. https://wordnet.princeton.edu/. Accessed April 6, 2016.

[13] Timothy G. Abbott, Katherine J. Lai, Michael R. Lieberman, and Eric C. Price. Browser-based attacks on Tor. In *Privacy Enhancing Technologies*, Lecture Notes in Computer Science, pages 184–199. Springer, 2007.

[14] Majid Adeli and Huaping Liu. On the inherent security of linear network coding. *Communications Letters, IEEE*, 17(8):1668–1671, 2013.

[15] Amazon. Mechanical turk. https://www.mturk.com/mturk/welcome. Accessed April 6, 2016.

[16] Emre Atsan, Iris Safaka, Lorenzo Keller, and Christina Fragouli. Low cost security for sensor networks. In *Network Coding (NetCod), 2013 International Symposium on*, pages 1–6. Ieee, 2013.

[17] Babak Azimi-Sadjadi, Aggelos Kiayias, Alejandra Mercado, and Bulent Yener. Robust key generation from signal envelopes in wireless networks. In *ACM conference on Computer and communications security*, 2007.

[18] Rimon Barr, Zygmunt J Haas, and Robbert van Renesse. JiST: An efficient approach to simulation using virtual machines. *Software: Practice and Experience*, 35(6):539–576, 2005.

[19] Rimon Barr, Zygmunt J Haas, and Robbert Van Renesse. Scalable wireless ad hoc network simulation. *Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad hoc Wireless, and Peer-to-Peer Networks*, 2005.

[20] Kapil Bhattad and Krishna R Narayanan. Weakly secure network coding. *NetCod, Apr*, 104, 2005.

[21] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, 2009.

[22] Sergio Boixo, Troels F Rønnow, Sergei V Isakov, Zhihui Wang, David Wecker, Daniel A Lidar, John M Martinis, and Matthias Troyer. Evidence for quantum annealing with more than one hundred qubits. *Nature Physics*, 10(3):218–224, 2014.

[23] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.

[24] Ning Cai and Raymond Yeung. Secure network coding on a wiretap network. *IEEE Transactions on Information Theory,*, 57(1), 2011.

[25] Seyit A Camtepe and Bülent Yener. Key distribution mechanisms for wireless sensor networks: a survey. *Rensselaer Polytechnic Institute, Troy, New York, Technical Report*, pages 05–07, 2005.

[26] Bob Carpenter. Scaling high-order character language models to gigabytes. In *Proceedings of the Workshop on Software*, pages 86–99. Association for Computational Linguistics, 2005.

[27] Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. In *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, pages 197–213. IEEE, 2003.

[28] Haowen Chan, Adrian Perrig, and Dawn Song. Key distribution techniques for sensor networks. *Wireless sensor networks*, pages 277–303, 2004.

[29] Ching-Yun Chang and Stephen Clark. Linguistic steganography using automatically generated paraphrases. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 591–599. Association for Computational Linguistics, 2010.

[30] Ching-Yun Chang and Stephen Clark. Practical linguistic steganography using contextual synonym substitution and vertex colour coding. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1194–1203. Association for Computational Linguistics, 2010.

[31] Mark Chapman and George Davida. Plausible deniability using automated linguistic stegonagraphy. In *Infrastructure Security*, pages 276–287. Springer, 2002.

[32] David L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2), 1981.

[33] Chi-Yuan Chen and Han-Chieh Chao. A survey of key distribution in wireless sensor networks. *Security and Communication Networks*, 2011.

[34] Stanley F Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393, 1999.

[35] Zhi-li Chen, Liu-sheng Huang, Zhen-shan Yu, Xin-xin Zhao, and Xue-ling Zhao. Effective linguistic steganography detection. In *Computer and Information Technology Workshops, 2008. CIT Workshops 2008. IEEE 8th International Conference on*, pages 224–229. IEEE, 2008.

[36] Zhili Chen, Liusheng Huang, Zhenshan Yu, Wei Yang, Lingjun Li, Xueling Zheng, and Xinxin Zhao. Linguistic steganography detection using statistical characteristics of correlations between words. In *Information Hiding*, pages 224–235. Springer, 2008.

[37] Ivan Csiszar and Prakash Narayan. Secrecy capacities for multiterminal channel models. *Information Theory, IEEE Transactions on*, 54(6):2437–2452, 2008.

[38] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th Usenix Security Symposium*, 2004.

[39] Ersen Ekrem and Sennur Ulukus. Secrecy Capacity of a Class of Broadcast Channels with an Eavesdropper. *EURASIP J. Wireless Comm. and Networking*, 2009.

[40] Laurent Eschenauer and Virgil D Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 41–47. ACM, 2002.

[41] Christina Fragouli, Jorg Widmer, and Jean-Yves Le Boudec. A network coding approach to energy efficient broadcasting: from theory to practice. Technical report, 2005.

[42] Edward B Fry and Jacqueline E Kress. *The reading teacher's book of lists*, volume 55. John Wiley & Sons, 2012.

[43] S. Gollakota and D. Katabi. Physical layer wireless security made fast and channel independent. In *INFOCOM*, 2011.

[44] Google. Requests for user information. https://www.google.com/transparencyreport/. Accessed April 6, 2016.

[45] Michael Grosvald and C Orgun. Human-versus computer-generated text-based steganography: real-world tests of two algorithms. *Journal of Information Hiding and Multimedia Signal Processing*, 3(1), 2012.

[46] Michael Grosvald and C Orhan Orgun. Free from the cover text: a human-generated natural language approach to text-based steganography. *Journal of Information Hiding and Multimedia Signal Processing*, 2(2), 2011.

[47] The Stanford Natural Language Processing Group. The stanford parser: A statistical parser. http://nlp.stanford.edu/software/lex-parser.shtml. Accessed April 6, 2016.

[48] Piyush Gupta and Panganmala R Kumar. The capacity of wireless networks. *Information Theory, IEEE Transactions on*, 46(2), 2000.

[49] Willie K Harrison, João Almeida, Steven W McLaughlin, and João Barros. Physical-layer security over correlated erasure channels. In *Communications (ICC), 2012 IEEE International Conference on*, pages 888–892, 2012.

[50] Willie K Harrison, Jorge Almeida, Matthieu R Bloch, Steven W McLaughlin, and Joao Barros. Coding for secrecy: An overview of error-control coding techniques for physical-layer security. *Signal Processing Magazine, IEEE*, 30(5):41–50, 2013.

[51] David A Huffman et al. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[52] Mahdi Jafari, Lorenzo Keller, Christina Fragouli, and Katerina Argyraki. Compressed network coding vectors. In *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*, pages 109–113. IEEE, 2009.

[53] Evan Jones. Extracting text from wikipedia. http://www.evanjones.ca/software/wikipedia2text.html. Accessed April 6, 2016.

[54] Bhavana Kanukurthi and Leonid Reyzin. Key agreement from close secrets over unsecured channels. In *Advances in Cryptology-EUROCRYPT 2009*, pages 206–223. Springer, 2009.

[55] Lorenzo Keller, Emre Atsan, Katerina Argyraki, and Christina Fragouli. Sensecode: Network coding for reliable sensor networks. *ACM Transactions on Sensor Networks*, 9(2), 2013.

[56] Amjad Saeed Khan, Andrea Tassi, and Ioannis Chatzigeorgiou. Rethinking the intercept probability of random linear network coding. *Communications Letters, IEEE*, 19(10):1762–1765, 2015.

[57] Gabriel Kliot. Technion extensions of the jist/swans simulator. http://www.cs.technion.ac.il/~gabik/Jist-Swans/. Accessed April 6, 2016.

[58] Jiejun Kong and Xiaoyan Hong. ANODR: Anonymous on demand routing with untraceable routes for mobile ad-hoc networks. In *MobiHoc*, 2003.

[59] Jean-Yves Le Boudec. *Performance Evaluation of Computer and Communication Systems*. EPFL Press, Lausanne, Switzerland, 2010.

[60] Sung-Ju Lee, William Su, Julian Hsu, Mario Gerla, and Rajive Bagrodia. A performance comparison study of ad hoc wireless multicast protocols. In *INFOCOM*, 2000.

[61] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. Tossim: Accurate and scalable simulation of entire tinyos applications. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 126–137. ACM, 2003.

[62] Hongbo Liu, Yang Wang, Jie Yang, and Yingying Chen. Fast and practical secret key extraction by exploiting channel response. In *INFOCOM*. IEEE, 2013.

[63] Greta M Ljung and George EP Box. On a measure of lack of fit in time series models. *Biometrika*, 65(2):297–303, 1978.

[64] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-holland Publishing Company, 2nd edition, 1978.

[65] Dan Klein Christopher D Manning. Natural language parsing. In *Advances in Neural Information Processing Systems 15: Proceedings of the 2002 Conference*, volume 15, page 3. MIT Press, 2003.

[66] MathWorks. Boxplot. http://mathworks.com/help/stats/boxplot.html. Accessed April 6, 2016.

[67] Ueli M Maurer. Secret key agreement by public discussion from common information. *Information Theory, IEEE Transactions on*, 39(3):733–742, 1993.

[68] Hasan M Meral, Emre Sevinc, Ersin Ünkar, Bülent Sankur, A Sumru Özsoy, and Tunga Güngör. Syntactic tools for text watermarking. In *Electronic Imaging 2007*, pages 65050X–65050X. International Society for Optics and Photonics, 2007.

[69] A. Mills, B. Smith, T.C. Clancy, E. Soljanin, and S. Vishwanath. On secure communication over wireless erasure networks. In *IEEE International Symposium on Information Theory (ISIT)*, pages 161–165, 2008.

[70] Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[71] Steven Murdoch and George Danezis. Low-cost traffic analysis of Tor. In *IEEE Symposium on Security and Privacy*, pages 195, 183, 2005.

[72] Steven J Murdoch. *Covert channel vulnerabilities in anonymity systems*. PhD thesis, University of Cambridge, 2007.

[73] Brian Murphy and Carl Vogel. The syntax of concealment: reliable methods for plain text information hiding. In *Electronic Imaging 2007*, pages 65050Y–65050Y. International Society for Optics and Photonics, 2007.

[74] Hao Niu, Masayuki Iwai, Kaoru Sezaki, Li Sun, and Qinghe Du. Exploiting fountain codes for secure wireless delivery. *Communications Letters, IEEE*, 18(5):777–780, 2014.

[75] National Institute of Standards and Computer security division Technology. Post-quantum crypto project. http://csrc.nist.gov/groups/ST/post-quantum-crypto/. Accessed April 6, 2016.

[76] Paulo F Oliveira and Joao Barros. A network coding approach to secret key distribution. *Information Forensics and Security, IEEE Transactions on*, 3(3):414–423, 2008.

[77] OpenFst. Openfst library. http://www.openfst.org. Accessed April 6, 2016.

[78] OpenGrm. Ngram library. http://www.opengrm.org/. Accessed April 6, 2016.

[79] Guangyu Pei and Thomas R Henderson. Validation of OFDM error rate model in ns-3, 2010. http://www.nsnam.org/~pei/80211ofdm.pdf.

[80] Sriram Nandha Premnath, Jessica Croft, Neal Patwari, and Sneha Kumar Kasera. Efficient high-rate secret key extraction in wireless sensor networks using collaboration. *ACM Transactions on Sensor Networks (TOSN)*, 11(1):2, 2014.

[81] Brian Roark, Richard Sproat, Cyril Allauzen, Michael Riley, Jeffrey Sorensen, and Terry Tai. The opengrm open-source finite-state grammar software libraries. In *Proceedings of the ACL 2012 System Demonstrations*, pages 61–66. Association for Computational Linguistics, 2012.

[82] Iris Safaka, Katerina Argyraki, Christina Fragouli, David Applegate, and Balachander Krishnamurthy. Matryoshka: Hiding secret communication in plain sight. *Under submission*, 2016.

[83] Iris Safaka, László Czap, Katerina Argyraki, and Christina Fragouli. Towards unconditional tor-like anonymity. In *Network Coding (NetCod), 2015 International Symposium on*, pages 66–70. IEEE, 2015.

[84] Iris Safaka, László Czap, Katerina Argyraki, and Christina Fragouli. Creating Secrets out of Packet Erasures. *IEEE Transactions on Information Forensics and Security*, 11(6):1177–1191, 2016.

[85] Iris Safaka, Christina Fragouli, Katerina Argyraki, and Suhas Diggavi. Creating shared secrets out of thin air. In *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, pages 73–78. ACM, 2012.

[86] Iris Safaka, Christina Fragouli, Katerina Argyraki, and Suhas Diggavi. Exchanging pairwise secrets efficiently. In *INFOCOM, 2013 Proceedings IEEE*, pages 2265–2273, 2013.

[87] Bernhard Schölkopf, Robert C Williamson, Alexander J Smola, John Shawe-Taylor, John C Platt, et al. Support vector method for novelty detection. In *NIPS*, volume 12, pages 582–588. Citeseer, 1999.

[88] M Jafari Siavoshani, Shaunak Mishra, Suhas N Diggavi, and Christina Fragouli. Group secret key agreement over state-dependent wireless broadcast channels. In *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, pages 1960–1964. IEEE, 2011.

[89] Mahdi Jafari Siavoshani, Christina Fragouli, Suhas Diggavi, Uday Pulleti, and Katerina Argyraki. Group secret key generation over broadcast erasure channels. In *Signals, Systems and Computers (ASILOMAR), 2010 Conference Record of the Forty Fourth Asilomar Conference on*, pages 719–723. IEEE, 2010.

[90] Cuneyt M Taskiran, Umut Topkara, Mercan Topkara, and Edward J Delp. Attacks on lexical natural language steganography systems. In *Electronic Imaging 2006*, pages 607209–607209. International Society for Optics and Photonics, 2006.

[91] Mercan Topkara, Umut Topkara, and Mikhail J Atallah. Words are not enough: sentence level natural language watermarking. In *Proceedings of the 4th ACM international workshop on Contents protection and security*, pages 37–46. ACM, 2006.

[92] Mercan Topkara, Umut Topkara, and Mikhail J Atallah. Information hiding through errors: a confusing approach. In *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, volume 29, 2007.

[93] Umut Topkara, Mercan Topkara, and Mikhail J Atallah. The hiding virtues of ambiguity: quantifiably resilient watermarking of natural language text through synonym substitutions. In *Proceedings of the 8th workshop on Multimedia and security*, pages 164–174. ACM, 2006.

[94] Olga Vybornova and Benoit Macq. A method of text watermarking using presuppositions. In *Electronic Imaging 2007*, pages 65051R–65051R. International Society for Optics and Photonics, 2007.

[95] Yawen Wei, Zhen Yu, and Yong Guan. Efficient weakly-secure network coding schemes against wiretapping attacks. In *Network Coding (NetCod), 2010 IEEE International Symposium on*, pages 1–6. IEEE, 2010.

[96] Alex Wilson, Phil Blunsom, and Andrew D Ker. Linguistic steganography on twitter: hierarchical language modeling with manual interaction. In *IS&T/SPIE Electronic Imaging*, pages 902803–902803. International Society for Optics and Photonics, 2014.

[97] Keith Winstein. Lexical steganography through adaptive modulation of the word choice hash. *Unpublished. http://www. imsa. edu/˜ keithw/tlex*, 1998.

[98] Aaron D Wyner. The wire-tap channel. *Bell System Technical Journal, The*, 54(8):1355–1387, 1975.

[99] Sheng Xiao, Weibo Gong, and Don Towsley. Secure wireless communication with dynamic secrets. In *INFOCOM*, 2010.

[100] Chunxuan Ye, Suhas Mathur, Alex Reznik, Yogendra Shah, Wade Trappe, and Narayan B Mandayam. Information-theoretically secret key generation for fading wireless channels. *IEEE Transactions on Information Forensics and Security*, 5(2):240–254, 2010.

[101] Peng Zhang, Yixin Jiang, Chuang Lin, Yanfei Fan, and Xuemin Shen. P-coding: secure network coding against eavesdropping attacks. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

[102] Chen Zhi-li, Huang Liu-Sheng, Yu Zhen-shan, Li Ling-jun, and Yang Wei. A statistical algorithm for linguistic steganography detection based on distribution of words. In *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, pages 558–563. IEEE, 2008.

[103] Bo Zhu, Zhiguo Wan, M.S. Kankanhalli, Feng Bao, and R.-H. Deng. Anonymous secure routing in mobile ad-hoc networks. In *IEEE International Conference on Local Computer Networks*, pages 102–108, 2004.

Iris Safaka

Greek Nationality – Swiss Permit C
Ch. de Maillefer 141, 1052
Le Mont-Sur-Lausanne, Switzerland
iris.safaka@gmail.com
people.epfl.ch/iris.safaka
+41 78 854 06 42

EXPERIENCE

Research and Teaching Assistant, 2011 – 2016
IC-ARNI & NAL, Swiss Federal Institute of Technology (EPFL), Switzerland

Research Intern, 2010 – 2011
IC-ARNI, Swiss Federal Institute of Technology (EPFL), Switzerland

Research Intern, Spring 2010
VERIMAG Research Center, Grenoble, France

EDUCATION

Phd in Computer, Communication and Information Sciences, 2011 – 2016
Swiss Federal Institute of Technology (EPFL), Switzerland
Thesis: Building Security Protocols Against Powerful Adversaries.
(Advisers: Prof. Christina Fragouli & Prof. Katerina Argyraki)

MSc in Computer Science, 2009 – 2010
University Joseph Fourier, Grenoble I, France (GPA: B+)
Specialization in Distributed, Embedded, Parallel and Wireless Systems
Thesis: Translation of MATLAB/Simulink models into BIP models

Diploma in Electrical and Computer Engineering, 2004 – 2009
Polytechnic School of University of Patras, Greece (GPA: 8.4/10)
Specialization in Electronics and Computer Systems
Thesis: Development of communication protocols for large-scale WSN applications

Awards: Scholarship and award from the State Scholarships Foundation for my academic records in my first and fifth year of studies.

TEACHING EXPERIENCE

Teaching Assistant, Programming in C, EPFL, Fall 2014
Teaching Assistant, Computer Networks, EPFL, Fall 2012, 2013
Teaching Assistant, Circuits & Systems, EPFL, Spring 2012, 2013, 2014

Bachelor Project Supervisor, Arno Schneuwly, EPFL, Fall 2014
Bachelor Project Supervisor, Malik Beytrison, EPFL, Fall 2013

Awards: Outstanding Teaching Assistant Award 2013, from the School of Computer and Communication Sciences of EPFL.

PUBLICATIONS

- *"Matryoshka: Hiding Secret Communication in Plain Sight".* I. Safaka, K. Argyraki, C. Fragouli, D. Applegate, B. Krishnamurthy. Under submission, 2016.

- *"Creating Secrets Out Of Packet Erasures".* I. Safaka, L. Czap, K. Argyraki and C. Fragouli. IEEE Transactions on Information Forensics and Security, 2016.

- *"Towards Unconditional Tor-Like Anonymity".* I. Safaka, L. Czap, K. Argyraki and C. Fragouli. IEEE Sumposium on Network Coding (NETCOD), Sydney, 2015.

- *"Exchanging Pairwise Secrets Efficiently".* I. Safaka, C. Fragouli, K. Argyraki, S. Diggavi. IEEE Conference on Computer Communications (INFOCOM), Turin, 2013.

- *"Low Cost Security for Sensor Networks".* E. Atsan, I. Safaka, L. Keller, C. Fragouli. IEEE Symposium on Network Coding (NETCOD), Calgary, Canada, 2013.

- *"Creating Shared Secrets out of Thin Air".* I. Safaka, C. Fragouli, K. Argyraki, S. Diggavi. ACM Workshop on Hot Topics in Networks (HotNets), Redmond, USA, 2012.

- *"Optimizing the Location Obfuscation in Location-Based Mobile Systems".* I. Safaka Technical report, 2011. http://infoscience.epfl.ch/record/196957.

COMPUTER SKILLS

Java, Python, C, MATLAB, shell scripting
Windows, Linux, OpenFST, OpenWrt, Android SDK, Software-Defined Radios (WARP)

KEY SKILLS

- Good background in various security and privacy aspects of wireless systems.
- Theoretical and practical knowledge on networked systems.
- Hands-on experience with popular classification techniques (eg. SVM)
- Good knowledge of performance evaluation techniques and statistics
- Modeling and analysis of theoretical problems and testing on practical systems.
- Development of network applications for Android OS and wireless LANs.
- Familiar with Natural Language Processing techniques and tools.

- Excellent communication skills, team spirit and collaborative attitude.
- Fast learner and ability to work independently.

LANGUAGES

English (C2 level), French (B2 level), German (B1 level), Greek (Mother language)

REFERENCES

- Prof. Christina Fragouli (EPFL/UCLA) - christina.fragouli@ucla.edu
- Prof. Katerina Argyraki (EPFL) - katerina.argyraki@epfl.ch
- Dr. Emre Atsan (Swisscom) - eatsan@gmail.com