Physical-layer Identification of RFID Devices

Boris Danev
Dept. of Computer Science
ETH Zürich, Switzerland
boris.danev@inf.ethz.ch

Thomas S. Heydt-Benjamin
IBM Zürich Research
Laboratory, Switzerland
hey@zurich.ibm.com

Srdjan Čapkun
Dept. of Computer Science
ETH Zürich, Switzerland
capkuns@inf.ethz.ch

Abstract

In this work we perform the first comprehensive study of physical-layer identification of RFID transponders. We propose several techniques for the extraction of RFID physical-layer fingerprints. We show that RFID transponders can be accurately identified in a controlled environment based on stable fingerprints corresponding to their physical-layer properties. We tested our techniques on a set of 50 RFID smart cards of the same manufacturer and type, and we show that these techniques enable the identification of individual transponders with an Equal Error Rate of 2.43% (single run) and 4.38% (two runs). We further applied our techniques to a smaller set of electronic passports, where we obtained a similar identification accuracy. Our results indicate that physical-layer identification of RFID transponders can be practical and thus has a potential to be used in a number of applications including product and document counterfeiting detection.

1 Introduction

Passively powered Radio Frequency Identification Devices (RFID) are becoming increasingly important components of a number of security systems such as electronic passports [3], contactless identity cards [4], and supply chain systems [16]. Due to their importance, a number of security protocols have been proposed for RFID authentication [46, 25, 17], key management [31, 28] and privacy-preserving deployment [6, 29, 26, 37, 19, 14, 13]. International standards have been accepted that specify the use of RFID tags in electronic travel documents [3]. Although the literature contains a number of investigations of RFID security and privacy protocols [27, 5] on the logical level, little attention has been dedicated to the security implications of the RFID physical communication layer.

In this work, we focus on the RFID physical communication layer and perform the first study of RFID transponder physical-layer identification. We present a hardware set-up and a set of techniques that enable us to perform the identification of individual RFID transponders of the same manufacturer and model. We show that RFID transponders can be accurately identified in a controlled measurement environment based on stable fingerprints corresponding to their physical-layer properties. The measurement environment requires close proximity and fixed positioning of the transponder with respect to the acquisition antennas.

Our techniques are based on the extraction of the modulation shape and spectral features of the signals emitted by transponders when subjected to both well formed reader signals, and to out of specification reader signals. We tested our techniques on a set of 50 RFID smart cards of the same manufacturer and type and show that these techniques enable the identification of individual cards with an Equal Error Rate of 2.43% (single run) and 4.38% (two runs). We further applied our techniques to a smaller set of electronic passports, where we obtained a similar identification accuracy. We also tested the classification accuracy of our techniques, and show that they achieve an average classification error of 0% for a set of classes corresponding to the countries of issuance. We further show that our techniques produce features that form compact and computationally efficient fingerprints. Given the low frequencies of operation of the transponders in our study, the extraction of the fingerprints is inexpensive, and could be performed using a low-cost purpose-built reader.

Although the implications of physical-layer identification of RFID transponders are broad, we believe that the techniques we present can potentially find their use in the detection of cloned products and identity documents, where the (stored) fingerprints of legitimate documents are compared with those of the presented documents. Our experimental setup corresponds to this application in which the transponders are fingerprinted from close proximity and in a controlled environment.

It has been recently shown that despite numerous protections, RFIDs in current electronic documents can be successfully cloned [18, 34, 33, 47], even if they apply the full range of protective measures specified by the standard [3], including active authentication. We see our techniques as an additional, efficient and inexpensive mechanism that can be used to detect RFID cloning. More precisely, to avoid detection of a cloned document, an adversary has to produce a clone using a transponder with the same fingerprint as the original document. Although, it may be hard to perform such task, the amount of effort required is an open research problem. We discuss two methods of applying RFID physical-layer identification to cloning detection and compare it to other anti-cloning solutions, like those based on physically-unclonable functions (PUFs) [12].

Our results show the feasibility of RFID transponder fingerprinting in a controlled environment. Using the proposed methods is not enough to extract the same or similar fingerprints from a larger distance (e.g., 1 meter). In our experiments, such remote feature extraction process resulted in incorrect identification. Therefore, we cannot assert that chip holder privacy can be compromised remotely using our techniques. This result further motivates an investigation of physical-layer features of RFID transponders that would allow their remote identification, irrespective of (e.g., random) protocol-level identifiers that the devices use on the logical communication level. Our current results do not allow us to conclude that such distinguishable features can be extracted remotely.

The remainder of this paper is organized as follows. In Section 2, we present our system model and investigation parameters. In Section 3, we detail our fingerprinting setup (i.e., a purpose-built reader), signal capturing process and summarize the data acquisition procedure and collected data. The proposed features for transponder classification and identification are explained in Section 4 and their performance is analyzed in Section 5. We discuss an application of our techniques to document counterfeiting detection in Section 6, make an overview of background and related work in Section 7 and conclude the paper in Section 8.

2 Problem and System Overview

In this work, we explore physical-layer techniques for detection of cloned and/or counterfeit devices. We focus on building physical-layer fingerprints of RFID transponders for the following two objectives:

1. RFID transponder classification: the ability to associate RFID transponders to previously defined transponder classes. In the case of identity documents classes might, for example, be defined based on the country that issued the document and the year of issuance.

2. RFID transponder identification: the ability to identify same model and manufacturer RFID transponders. In the case of identity documents, this could mean identifying documents from the same country, year and place of issuance.

A classification system must associate unknown RFID transponder fingerprints to previously defined classes C. It performs "1-to-C" comparisons and assigns the RFID fingerprint to the class with the highest similarity according to a chosen similarity measure (Section 5.1). This corresponds to a scenario in which an authority verifies whether an identity document belongs to a claimed class (e.g., country of issuance).

An identification system typically works in one of two modes: either identification of one device among many, or verification that a device's fingerprint matches its claimed identity [8]. In this work, we consider verification of a device's claimed or assumed identity. This corresponds to a scenario in which the fingerprint of an identity document (e.g., passport), stored in a back-end database or in the document chip, is compared to the measured fingerprint of the presented document. The verification system provides an Accept/Reject decision based on a threshold value T (Section 5.1). Identity verification requires only "1-to-1" fingerprint comparison and is therefore scalable in the number of transponders.

In this study we use a single experimental setup for examination of both classification and identification. Our setup consists of two main components: a signal acquisition setup (i.e., a purpose-built RFID reader) (Section 3) and a feature selection and matching component (Section 4). In our signal acquisition setup we use a purpose-built reader to transmit crafted signals which then stimulate a response from the target RFID transponders. We then capture and analyze such responses. In particular, we consider transponder responses when subjected to the following signals from the reader: standard [4] transponder wake-up message, transponder wake-up message at intentionally out-of-specification carrier frequencies, a high-energy burst of sinusoidal carrier at an out-of-specification frequency, and a high-energy linear frequency sweep.

To evaluate the system accuracy, we make use of two different device populations (Table 1). The first population consists of 50 "identical" JCOP NXP 4.1 smart cards [2] which contain NXP RFID transponders (ISO 14443, HF 13.56 MHz). We chose these transponders since they are popular for use in identity documents and access cards, and because they have also been used by hackers to demonstrate cloning attacks against

e-passports [47]. The second population contains 8 electronic passports from 3 different countries[1]. These two populations allow us to define different transponder classes (e.g., 3 issuing countries, and a separate class for JCOP cards) for classification and include a sufficient set of identical transponders to quantify the identification accuracy of the transponders of the same model and manufacturer.

In summary, in this work, we answer the following interrelated questions:

1. What is the classification accuracy for different classes of transponders, given the extracted features?
2. What is the identification accuracy for transponders of the same model and manufacturer, given the extracted features?
3. How is the classification and identification accuracy affected by the number of signals used to build the transponder fingerprint?
4. How stable are the extracted features, across different acquisition runs and across different transponder placements (relative to the reader)?

3 Experimental Setup and Data

In this section, we first describe our signal acquisition setup. We then detail the different types of experiments we performed and present the collected datasets from our population of transponders.

3.1 Hardware Setup

Figure 1 displays the hardware setup that we use to collect RF signals from the RFID devices. Our setup is essentially a purpose-built RFID reader that can operate within the standardized RFID communication specifications [4], but can also operate out of specifications, thus enabling a broader range of experiments. The setup consists of two signal generators, used for envelope generation (envelope generator) and for signal modulation (modulation generator), and of transmitting and acquisition antennas. The envelope generator is fed with a waveform that represents the communication protocol wake-up command[2] required for initiating communication with RFID transponders. The envelope waveform is then sent to the modulation generator and is modulated according to the ISO/IEC 14443 protocol Type A or B, depending on the transponders being contacted. The modulated signal is then sent over a PCB transmitting antenna. Finally, the wake-up signal and the response from the transponder are received at the acquisition antenna and captured at the oscilloscope. The separation of the envelope generation and modulation steps allowed us to independently vary envelope and modulation characteristics in our experiments.

In order to collect the RF signal response, we built a "sandwich" style antenna arrangement (Figure 2b) where an acquisition antenna is positioned between the transmission antenna and the target RFID transponder. An wooden platform holds the transmission and acquisition antennas in a fixed position to avoid changes in antenna polarization[3]. The platform is separated from the desk by a non-metallic wooden cage. The transmission and acquisition antennas are both connected to an oscilloscope. We used the RF signal on the transmission antenna to trigger the acquisition and then record the transponder's response at the acquisition antenna. It should be noted that we can also observe the transponder's response at the transmission antenna, however as the acquisition antenna had a higher gain than the transmission antenna, we used the described setup to obtain better signal-to-noise ratio.

3.2 Performed Experiments

Using the proposed setup, we performed four major experiments:

Experiment 1 (Standard): In this experiment we initiate communication with the transponders as defined by Type A and B protocols in the ISO/IEC 14443 standard. The envelope generator generates the Type A and B envelopes and the modulation generator modulates the signal at a carrier frequency F_c= 13.56 MHz, using 100% ASK for Type A and 10% ASK for Type B at the nominal bit rate of $F_b \sim 106$kbit/s.[4] The experiment consists of the following steps: a period of unmodulated carrier is transmitted to power the transponder at which time the oscilloscope begins recording the data. The carrier is then modulated according to the envelope such that it corresponds to a WUQA (Type A) or WUQB (Type B) wake-up command. When the commands are no longer transmitted, an unmodulated period of carrier is then sustained while the oscilloscope records the response from the transponder. The carrier is turned off between each

[1] The small quantity of the electronic passports used in the experiments is due to the difficulty of finding people who are in possession of such passports and at the same time willing to allow experimentation on them.

[2] ISO/IEC 14443 for RFID communication defines two different communication protocols, Type A and B, which use different wake-up commands: WUQA and WUQB, respectively.

[3] It has been observed that such changes can reduce the identification accuracy [11].

[4] For 100% ASK modulation we used pulse modulation as standard built-in amplitude modulation (AM) in our generators could not reach the required precision.

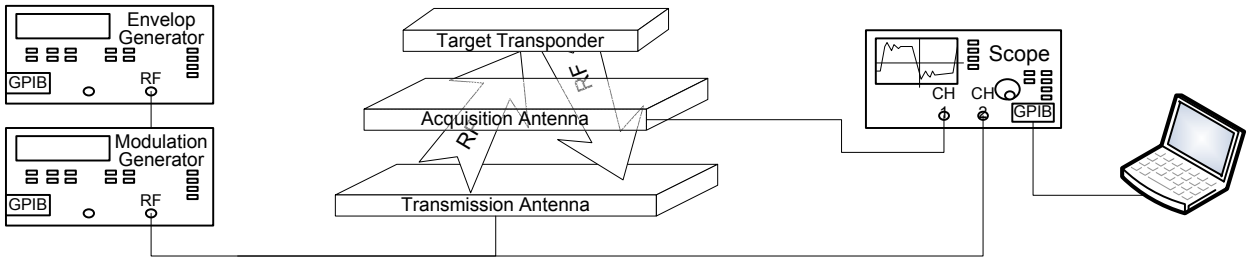


Figure 1: Signal acquisition setup. Envelope and modulation generators generate wake-up signals that initiate the response from the RFID transponder. This wake-up signal is transmitted by the transmitting antenna. The acquisition antenna captures both the wake-up signal and the response from the transponder. The signal from the acquisition antenna is then captured and recorded by the oscilloscope.

(a) (b)

Figure 2: a) Transmission and acquisition antennas. b) An electronic identity document being placed in the fingerprinting setup.

observation to ensure the transponder reboots each time. Figures 3a and 3b show the collected samples from Type A and Type B RFID transponders, respectively. This experiment enables us to test if the transponder's responses can be distinguished when they are subjected to standard signals from the reader.

Experiment 2 (Varied F_c): In this experiment, we test transponder responses to the same signals as in Experiment 1, but on out of (ISO/IEC 14443) specification carrier frequencies. Instead of on F_c=13.56 MHz, our setup transmits the signals on carrier frequencies between F_c=12.96 MHz and 14.36 MHz. Figures 3c and 3d display sample transponder responses to signals on F_c=13.06 MHz. We expect the variation in the transponder responses to be higher when they are subjected to out of specification signals, since the manufacturers mainly focus on transponder responses within the specified frequency range.

Experiment 3 (Burst): In this experiment, we tested transponder responses to bursts of RF energy. We subjected the transponders to 10 cycles (2 μs) of non-modulated 5 MHz carrier at an amplitude of V_{pp}=10 V (the maximum frequency and amplitude supported by our generators while in burst mode). Figure 4a shows a sample transponder response to such an RF burst. This experiment tests the response of transponders to an additional out-of-specification signal. We expect to see variation in different transponders' responses for a variety of reasons. For example since each transponder's antenna and charge pump is unique, we believe that during power-up it will present a unique modulation of an activating field.

Experiment 4 (Frequency Sweep): This experiment consists of observing transponder responses to a non-modulated carrier linear sweep from 100 Hz to 15 MHz at an amplitude of V_{pp}=10 V (as measured at transmitting antenna). The duration of the sweep is fixed to the maximum allowed by our generator, 10 ms. In this test we examine how the transponders react to many different frequencies. Among other things, such an experiment provides information about the resonances of the RF circuitry in each transponder. Figure 4b shows a sample transponder response to a frequency sweep. Note the different shape artifacts.

We found that samples collected from Experiment 2 were well suited for transponder classification, whereas

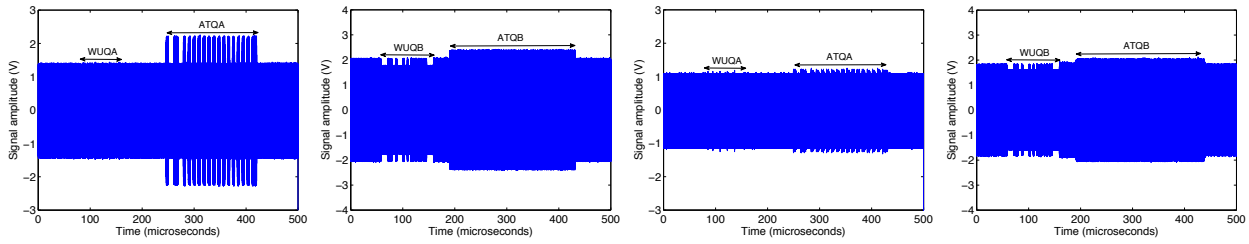


Figure 3: Experiment 1: Type A (a) and Type B (b) RFID transponder responses to WUQA and WUQB commands sent on the ISO/IEC 14443 specified carrier frequency (F_c=13.56 MHz). Experiment 2: Type A (c) and Type B (d) RFID transponder responses ATQA and ATQB to WUQA and WUQB commands respectively sent on an out of ISO/IEC 14443 specification carrier frequency (F_c=13.06 MHz)

.

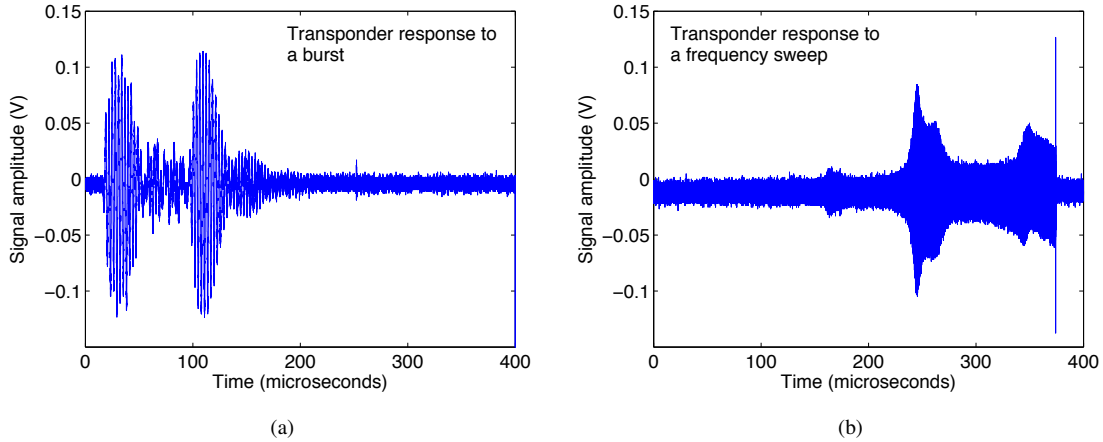


Figure 4: a) Experiment 3: transponder response sample to a non-modulated 5 MHz carrier in duration of 10 cycles. b) Experiment 4: transponder response sample to a non-modulated carrier linear sweep from 100 Hz to 15 MHz. The duration of the sweep is 10 ms.

those collected from Experiments 3 and 4 were better suited for identification of individual RFID transponders. We discuss this result at greater length in Section 4.

3.3 Collected Data

Using the proposed setup, we performed the experiments described in Section 3.2 and collected samples from 8 passports and 50 JCOP NXP 4.1 smart cards (same model and manufacturer). The types of devices used in the experiments are shown in Table 1. For the privacy of our research subjects we arbitrarily labeled the passports as ID1 to 8. To further protect their privacy we give the country and place of issuance under the pseudonyms C1 to C3 and P1 to P6 respectively.

Our data collection procedure for a single experiment "run" was as follows: We positioned the target RFID device on the experimental platform with all other transponders being at an out-of-range distance from the activating field. We then placed a heavy non-metallic weight on top of the transponder to position it firmly and horizontally on the platform. For each device we then performed Experiments 1-4 at fixed acquisition timing offset and sampling rate and saved the samples to a disk for later analysis. For each transponder we performed two runs, completely removing and replacing the target transponder on the experimental platform between runs. This ensures that extracted features are stable across repositioning.

In each iteration of Experiment 2 we collected 4 samples per run for 14 different carrier frequencies starting from F_c=12.96 up to 14.36 MHz with a step of 100 KHz. This resulted in 64 samples per transponder per run. In Experiments 3 and 4 we collected 50 samples per device per run.

Table 1: RFID device populations (passports and JCOP NXP smart cards).

Type	Number	Label	Country	Year	Place of Issue
Passport	2	ID1, ID2	C1	2006	P1
	1	ID3	C1	2006	P2
	1	ID4	C1	2006	P3
	1	ID5	C1	2007	P4
	1	ID6	C2	2008	P5
	1	ID7	C3	2008	P6
	1	ID8	C1	2008	P1
JCOP	50	J1..J50	JCOP NXP 4.1 cards (same model and manufacturer)		

4 Feature Extraction and Selection

The goal of the feature extraction and selection is to obtain distinctive fingerprints from raw data samples collected in the proposed experiments, which most effectively support the two objectives in our work, namely classification and identification. In this section, we detail the extraction and matching procedures of two types of features effective for that purpose: modulation-shape features (Section 4.1) and spectral PCA features (Section 4.2). We also investigated the use of some timing features, such as the time interval within which the transponder responds to an WUQ command and the duration of that response (Figure 5a). These timing features performed poorly in both tasks, hence in this work we focus on the modulation-shape and spectral features.

4.1 Modulation-shape Features

In this section, we describe the extraction and matching procedures for the features extracted from the shape of the modulated signal of the transponder responses at a given carrier frequency F_c (Experiment 1&2). Figure 5 b) shows the shape of the On-Off keying modulation for the JCOP NXP 4.1 card for the first packet in a transponder's response to a wake-up command. All Type A transponders in our study had a logically identical first packet.

For a given transponder, the features of the modulated signal are extracted from the captured transponder response (see Figure 3) denoted as $f(t,l)$, using Hilbert transformation. Here, $f(t,l)$ is the amplitude of the signal l at time t. The Hilbert transformation is a common transformation in signal processing used to obtain the signal envelope [38].

In Step (i), we apply Hilbert transformation on $f(t,l)$ to obtain $H(t,l)$:

$$H(t,l) \quad = \quad \text{Hil}(f(t,l)) \tag{1}$$

where Hil is a function implementing the Hilbert transform [36].

In Step (ii), the starting point of the modulation in $H(t,l)$ is determined using the variance-based threshold detection algorithm described in [40]. The end point is fixed to a pre-defined value (see Section 5) and then the modulation-shape fingerprint is extracted.

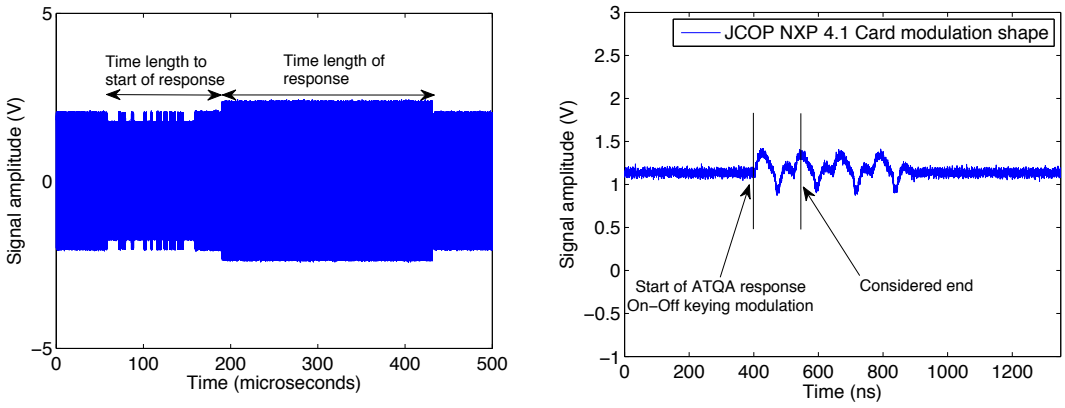
Feature matching between a reference and a test fingerprints is performed using standardized Euclidean distance, where each coordinate in the sum of squares is inverse weighted by the sample variance of that coordinate [35].

4.2 Spectral Features

In this section, we describe the extraction and matching of spectral features from data collected from Experiments 3 (Burst) and 4 (Sweep) (Section 3.2).

Both frequency sweep and burst data samples are extremely high-dimensional: each sweep data sample contains 960000 points (dimensions) and each burst data sample contains 40000. Such high-dimensional data typically contain many noisy dimensions which are detrimental to finding distinctive features. Therefore, it is critical to remove the noise as much as possible from the raw data samples.

We explored two basic approaches to solve the dimensionality problem. In the first approach, we considered transforming the data in the frequency domain by means of the Fast Fourier Transform (FFT) and remove the high frequencies (usually considered noisy) by filtering. However, matching experiments using direct vector similarity measures such as Euclidean and Cosine distance failed to produce distinctive enough features. This may be because in removing the high frequencies we are also removing frequencies that contain discriminative capabilities. Such behavior is commonly noticed in biometrics research [10]. In the second approach we down-sampled the signal at different rates in order to reduce the dimensionality. We then transformed the data in the frequency domain by FFT and applied standard vector similarity measures. Again reducing the dimensionality in this way did not prove to be effective in extracting sufficiently dis-



(a) (b)

Figure 5: a) Timing features extracted from Type B transponder responses. b) Modulation-shape features.

criminative features.

To overcome the above problems, we use a modification of Principal Component Analysis (PCA) for high-dimensional data [7], that reduces data dimensionality by discarding dimensions that do not contribute to the total covariance. Given that the number of dimensions is very high, orders of magnitude higher than the number of data samples we can process, a standard PCA procedure cannot be applied. In the following subsection, we briefly describe the used PCA modification.

4.2.1 Feature Extraction and Matching

For a given RFID device, spectral PCA features are extracted from N captured samples using a linear transformation derived from PCA for high-dimensional data. We denote a signal by $f(t,l)$, where $f(t,l)$ is the amplitude of the signal l at time t. The features are extracted in the following three steps:

In Step (i), we apply a one-dimensional Fourier transformation on $f(t,l)$ to obtain $F(\omega,l)$:

$$F(\omega,l) \quad = \quad \frac{1}{\sqrt{M}} \sum_{m=0}^{M-1} f(t,l) \exp(-2\pi i \frac{t\omega}{M}) \quad (2)$$

where M is the length of signal considered and $0 \leq t \leq M-1$ is time. We then remove from $F(\omega,l)$ its DC component and the redundant part of the spectrum; we denote the remaining part of the spectrum by $\vec{s_l}$.

In Step (ii), a projected vector $\vec{g_l}$, also called a spectral feature, is extracted from the Fourier spectrum using a PCA matrix W_{PCA}:

$$\vec{g_l} = W_{PCA}^t \vec{s_l} \quad (3)$$

The feature extraction from N captured samples for a given transponder is then given by $G = W_{PCA}^t S$ where G is an array of $\vec{g_l}$ and S is a matrix $S = [\ \vec{s_0} \ldots \vec{s_l} \ldots \vec{s_N}\ ]$.

Finally, in Step (iii), the feature template (fingerprint) h used for matching is computed:

$$h = \{\hat{G}; \Sigma_G\} \quad (4)$$

where $\hat{G}$ denotes the mean vector of G and Σ_G denotes the covariance matrix of G. The number of captured samples N used to build the feature template and the number of projected vectors in W_{PCA} (i.e., the subspace dimension) are experimentally determined.

Mahalanobis distance is used to find the similarities between fingerprints[5]. The result of matching a reference h^R and a test h^T feature templates is a matching score, calculated as follows.

$$scr(h^R, h^T) = min(\sqrt{(\hat{G}^T - \hat{G}^R)^t \Sigma_{G^R}^{-1} (\hat{G}^T - \hat{G}^R)}, \sqrt{(\hat{G}^T - \hat{G}^R)^t \Sigma_{G^T}^{-1} (\hat{G}^T - \hat{G}^R)}) \quad (5)$$

Values of the matching score closer to 0 indicate a better match between the feature templates. The proposed matching uses the mean and covariance of both test and reference templates. It also ensures the symmetric property, that is $scr(h^R, h^T) = scr(h^T, h^R)$.

It should be noted that the proposed feature extraction and matching method can be efficiently implemented in hardware as they use only linear transformations for feature extraction and inter-vector distance matchings. These operations have a low memory footprint and are computationally efficient.

4.2.2 PCA Training

In order to compute the eigenvalues and corresponding eigenvectors of the high-dimensional data (the number

[5]We discovered that the feature templates are distributed in ellipsoidal manner and therefore use Mahalanobis distance that weights each projected sample according to the obtained eigenvalues.

of samples $\ll$ the number of dimensions), we used the following lemma:

Lemma: For any $K \times D$ matrix W, mapping $x \rightarrow Wx$ is a one-to-one mapping that maps eigenvectors of W^TW onto those of WW^T.

Here W denotes a matrix containing K samples of dimensionality D. Using this lemma, we can first evaluate the covariance matrix in a lower space, find its eigenvectors and eigenvalues and then compute the high-dimensional eigenvectors in the original data space by normalized projection [7]. Based on this description, we compute the PCA matrix W_{PCA}=$[\vec{u_1}\vec{u_2}\ldots\vec{u_i}]$ by solving the eigenvector equation:

$$(\frac{1}{K}X^TX)(X^T\vec{v_i}) = \lambda_i(X^T\vec{v_i}) \quad (6)$$

where X is the training data matrix $K \times D$ and $\vec{v_i}$ are the eigenvectors of XX^T. We then compute the eigenvectors of our matrix $\vec{u_i}$ by normalizing:

$$\vec{u_i} = \frac{1}{\sqrt{K\lambda_i}}(X^T\vec{v_i}) \quad (7)$$

It should be noted that other algorithms like probabilistic PCA (e.g., EM for PCA) can potentially be also used given the fact that we discovered that only 5-10 eigenvectors are predominant. We intend to investigate these as a part of our future work.

5 Performance Results

In this section, we present the performance results of our fingerprinting system. First, we review the metrics that we use to evaluate the classification and identification accuracy.

5.1 Evaluation Metrics

As a metric for classification, we adopt the average classification error rate, defined as the percentage of incorrectly classified signatures to a predefined set of classes of signatures (e.g., countries). We used the 1-Nearest Neighbor rule [7] for estimating the similarity between testing and reference signatures from a given class; that is, a testing signature is matched to all reference signatures from all classes and assigned to the class with nearest distance similarity. It should be noted that more sophisticated classifiers can be devised such as Support Vector Machines (SVM), Probabilistic Neural Networks (PNN) [7]. However these classifiers require more training which we do not consider in this work.

As metrics for identification, we adopt the Equal Error Rate (EER) and the Receiver Operating Characteristic (ROC) since these are the most agreed metrics for evaluating identification systems [8]. The False Accept Rate (FAR) and the False Reject Rate (FRR) are the frequencies at which the false accept and the false reject events occur. The FAR and FRR are closely related to each other in the Receiver Operating Characteristic (ROC). ROC is a curve which allows to automatically compute FRR when the FAR is fixed at a desired level and vice versa [8]. The operating point in ROC, where FAR and FRR are equal, is called the Equal Error Rate (EER). The EER represents the most common measure of the accuracy of identification systems [1]. The operating threshold value at which the EER occurs is our threshold T for an Accept/Reject decision.

To increase the clarity of presentation, we use the Genuine Accept Rate (GAR = 1 - FRR) in the ROC because it shows the rate of Accepts of legitimate identities. In addition, we also compute FRR for common target values of FAR (e.g., FAR = 1%).

5.2 Classification Results

In this section, we present the results of the classification using modulation-shape and spectral features. In this evaluation, we consider all our passport samples and 5 of the JCOP NXP 4.1 cards. Here, the identity documents ID1, ID2, ID3, ID4, ID7, ID8 (see Table 1) and the JCOP cards implement Type A communication protocol, whereas ID5 and ID6 use Type B protocol. It is interesting to notice that within the same country class (C1) we have documents with two different communication protocols (ID1-ID4 and ID8 implement Type A, whereas ID5 implements Type B protocol).

5.2.1 Classification using Modulation-shape Features

The modulation-shape features described in Section 4 show the discriminant artifacts in the transponder's response. In particular, we discovered that these artifacts (shapes) vary from one transponder to another on out-of-specification carrier frequencies.

Figure 6 shows the modulation envelope shapes of the initial sequence of the RFID transponder's response after Hilbert transformation for 4 different classes of Type A protocol devices. These were recorded at an out of specification carrier frequency F_c=13.16MHz. Visual inspection shows that the modulation shapes not only differ from class to class but also are stable within different runs.

In order to quantify these observations more precisely, we considered classification with 3 classes (2 countries + JCOP cards) with all fingerprints from two different runs. The classification process was repeated 8 times with 8 different reference fingerprints per class for validation.

Table 2: Classification using modulation-shape features (Experiment 2)

Number of Classes	Class structure	Average Classification Error Rate
3	(C1),(C2),(JCOP)	0%
4	(ID1,ID3,ID4,ID8), (ID2), (ID7), (JCOP)	0%
2	(ID5-C1),(ID6-C3)	0%

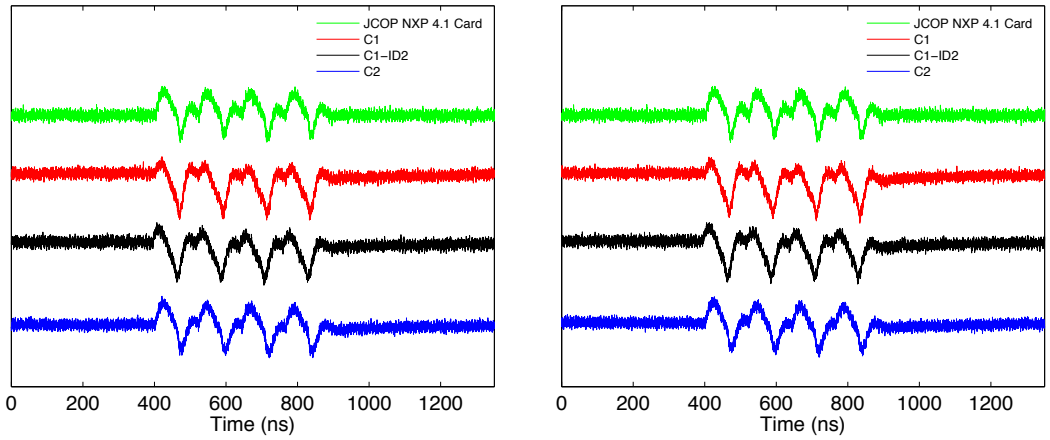


Figure 6: Modulation shape of the responses of 4 different classes (C1),(C1-ID2),(C2),(JCOP): a) first run b) second run. In each run, the sample transponders were freshly placed in the fingerprinting setup. These plots show the stability of the collected modulation-shape features across different runs.

The results show perfect separability of the classes with average classification error rate of 0%. In addition, after detailed inspection of the modulation-shape features we discovered that ID2 from C1 differs significantly from the representatives of that class. We therefore formed a new classification scenario with 5 classes and obtained again a classification error rate of 0%. It is an interesting result given that ID1 and ID2 are issued by the same country, in the same year and place of issue. However, their transponders are apparently different. The modulation-shapes of ID1,ID3 and ID4 from C1 could not be further distinguished using the combination of modulation-shape features and Euclidean matching. Table 2 shows the results.

Similar to Type A, the 2 Type B transponders from two different countries (C1,C3) available in our population showed complete separability with classification error rate of 0%. We acknowledge that our data set is insufficient due to the difficulty of obtaining e-passports. We believe however that our results are promising to stimulate future work with a larger set of e-passports.

In summary, the modulation shapes at an out-of-specification carrier frequency are successful in categorizing different classes of transponders (e.g., countries). They are quickly extractable and stable across different runs. For the classification task, there is no need of statistical analysis in contrast with the proposed spectral features analyzed in the next sections. An additional advantage is that specialized hardware is not required as current RFID readers can be easily adapted.

5.2.2 Classification using Burst and Sweep Spectral Features

We also performed classification using burst and sweep spectral features (Experiment 3 & 4) on the same set of classes as with modulation-shape features (Table 2). Similar to the modulation-shape features, this classification achieved a 0% classification error rate on the proposed classes. Moreover, using the spectral features we were also able to distinguish individually each of our 9 identity documents with an EER=0%, i.e. we were able to verify the identify of each individual document with an accuracy of 100% with FRR=FAR=0%. This result motivated us to estimate the identification accuracy of spectral features on a larger set of identical (of the same make and model) transponders.

5.3 Identification results

In this section we present the results of the identification capabilities of the (burst and sweep) spectral features for

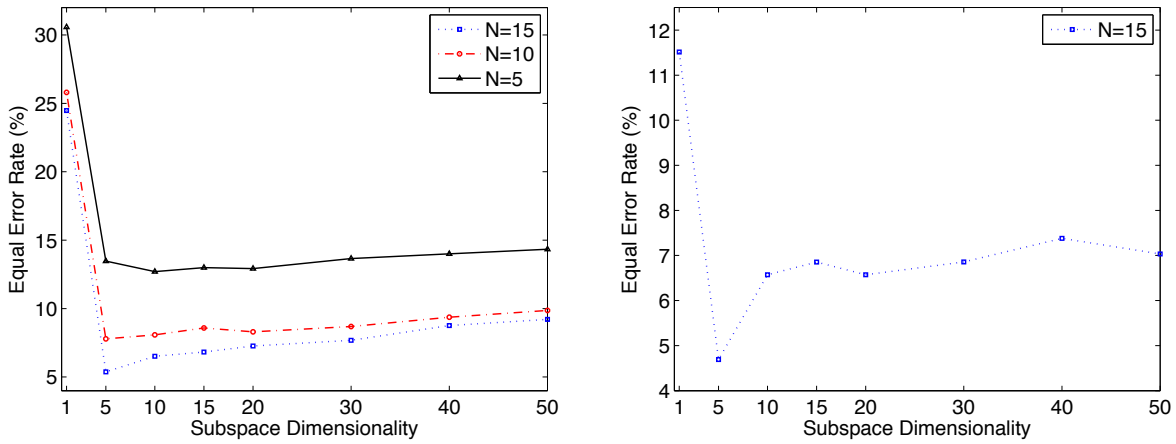


Figure 7: Spectral features identification accuracy for different number of samples N used to built the fingerprint and for different subspace dimensions: a) burst spectral features, b) sweep spectral features. 50 identical (same manufacturer and model) transponders are used in the computation.

our data population (50 identical JCOP NXP 4.1 cards). We adopt the following approach. We first evaluate the accuracy over the data collected in a single run of the experiment (Section 5.3.1 and 5.3.2). We then quantify the feature stability of the spectral features by considering samples from two independent runs together (Section 5.3.3).

We validate our results using cross-validation [7]. We measured 50 samples per transponder per run of which we use 5-10 samples for training and the remaining 40-45 samples for testing depending on the number of samples N used to build the fingerprint. The training and testing data are thus separated and allow validation of the identification accuracy.

5.3.1 Identification using Burst Spectral Features

In this evaluation, we consider the samples from the burst dataset, from a single experiment run (Experiment 3) in order to obtain a benchmark accuracy. We varied two parameters: the number of samples N used to build the feature templates (fingerprints) and the dimension of the PCA subspace used to project the original features into. The dimension of the PCA subspace is also related to the feature template size which we discuss below.

The results of this analysis are presented in Figure 7a for different N and subspace dimensionality. The dimension of the features before the projection is 19998. The results show the EER of the system reaching 0.0537 (5.37%) for N=15. This means that our system correctly identifies individual identical transponders with an accuracy of approximately 95% (GAR at the EER operating point) using the features extracted from the burst samples. We later show that this accuracy is preserved in cross-matchings between different runs. Table 3 summarizes the underlying data, namely the number of samples N, total genuine and imposter matchings performed for EER computation[6], Accept/Reject threshold, EER and confidence interval (CI).

The results in Figure 7a also confirm that using the first 5 eigenvectors to project and store the feature template provides the highest accuracy. Our proposed features therefore form compact and computationally efficient fingerprints (see Section 5.4).

5.3.2 Identification using Sweep Spectral Features

Similarly to the above analysis, we considered the first run of samples from the sweep experiment (Experiment 4) dataset. For computational reasons, we did not consider the entire sample. Instead, we extracted the spectral features from the part of the sample between 220 to 270 microseconds. As it can be seen in Figure 4, this part contains the biggest shape changes in the frequency sweep. This decision reduced the considered space to 100000 points which allowed reasonably fast feature extraction (26 s per sample). This clearly excludes some discriminant information from our analysis, and future work should include other sections of the sample signals.

The results are presented in Figure 7b for N=15 and

[6]The number of genuine and imposter matchings depends on the number of available fingerprints per transponder. For N=10, we are able to built 4 different fingerprints with the testing data within a run. This results in 6 different matchings of fingerprints from the same device (i.e., genuine matchings) and 392 different matchings of fingerprints from different transponders (i.e., imposter matchings). For 50 transponders, this makes 300 genuine and 19600 imposter matchings.

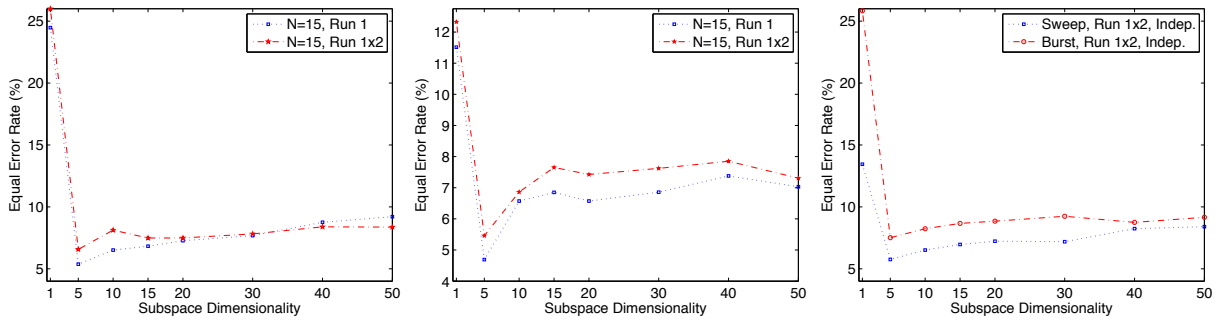


Figure 8: Feature stability in identification: a) burst spectral features b) sweep spectral features. 50 identical (same manufacturer and model) transponders are used in the experiments. c) burst and sweep spectral features on independent transponder sets for training and testing; 20 transponders are used for training and 30 transponders - for testing; N=15.

different subspace dimensions. The dimension of the original features before projection is 49998. We computed the EER for N=15 (see Burst analysis in Section 5.3.1). The obtained EER is 0.0469 (4.69%), when using the first 5 eigenvectors to project and store the feature template. The obtained accuracy is therefore similar to the one obtained with the burst features, i.e. our system correctly identifies the individual identical transponders with an accuracy of approximately 95% (GAR at the EER point). Table 3 shows the confidence intervals.

5.3.3 Feature Stability

In the previous sections we have analyzed the identification accuracy using burst and sweep spectral features within a single experiment run. This allows us to have a benchmark for estimating the stability of the features. In particular, we performed the following stability analysis:

1. Using the linear transformations W_{PCA} obtained in the first run, we selected 4 feature templates (2 from each run) and computed again the EER by considering only the cross matching scores of fingerprints from different runs[7]. The process was repeated 3 times with different feature templates from the two runs to validate the feature stability.

2. We trained the system over the first 20 transponders and then used the obtained linear transformation to estimate the accuracy over the remaining 30 transponders. This analysis tests the stability of the obtained linear transformations to discriminate independent transponder populations[8].

[7]This procedure is required in order to remove any possible bias from cross matching scores of fingerprints from the same run. We point out that this results in a reduced number of genuine and imposter matchings for the EER computation, 200 and 9800 respectively (see Table 3).

[8]The motivation behind this division (20 vs. 30) is that it gives reasonable number of transponders for both training and testing.

Figure 8 compares the EER accuracy obtained with the first run (Run 1) and the accuracy obtained by mixing fingerprints of both runs (Run 1×2) for a fixed N=15. Table 3 displays the confidence interval for subspace dimension of 5 eigenvectors. The obtained EERs do not show a statistically significant difference between the two experiments for both the burst and sweep features using 4-fold validation.

Figure 9 displays the EER accuracy obtained using independent transponder sets for training and testing for a fixed N=15. Here, the fingerprints from both runs are mixed as in the previous analysis. Table 4 summarizes the numeric results together with confidence intervals of the EER. Even though the testing population (30 transponders) is smaller, we observe that the sweep features do not show any significant accuracy deviation from the benchmark accuracy on Run 1×2 (Table 3). On the other hand, the burst features slightly decreased the accuracy on average (Table 3). The reason for this might be that 20 different transponders are not sufficient to train the system; however, we cannot assert this with certainty.

5.3.4 Combining Sweep and Burst Features

Given that the identification accuracies of both burst and sweep spectral features are similar; in order to fully characterize the identity verification we computed the ROC curves for the burst and sweep features as shown in Figure 9b. We notice that while the EERs are similar, the curves exhibit different accuracies at different FARs. In particular, for low FAR$\leq$1% the sweep features show lower GAR.

The burst and sweep features discriminate the fingerprints in a different way, and therefore these features can be combined in order to further increase the accuracy. Such combinations are being researched in multi-modal

Table 3: Summary of accuracy for the 5-dimensional spectral features (50 transponders).

Type	Run	N	Test matchings		Threshold T	EER (%)	EER CI (%)		Validation
			Genuine	Imposter			lower	upper	
Burst	1	15	150	11025	1.88	5.37	4.38	6.36	4-fold
	1	10	300	19600	2.91	7.79	5.29	10.28	4-fold
	1	5	300	19600	7.56	13.47	13.22	13.72	4-fold
	1x2	15	200	9800	2.64	6.57	6.25	6.89	4-fold
Sweep	1	15	150	11025	1.68	4.69	3.65	5.74	4-fold
	1x2	15	200	9800	1.93	5.46	5.08	5.84	4-fold

Table 4: Accuracy when independent sets are used for training (20) and testing (30) transponders.

Type	Run	N	Test matchings		Threshold T	EER (%)	EER CI (%)		Validation
			Genuine	Imposter			lower	upper	
Burst	1x2	15	120	3480	2.78	7.33	6.01	8.65	3-fold
Sweep	1x2	15	120	3480	2.03	5.75	5.45	6.05	3-fold

biometrics [42] where different "modalities" (e.g., fingerprint and vein) are combined to increase the identification accuracy and bring more robustness to the identification process [42].

A number of integration strategies have been proposed based on decision rules [32], logistic functions to map output scores into a single overall score [24], etc. Figure 9 shows the EERs and ROC curves of feature combination by using the sum as an integration function. The overall matching score between a test and a reference template is the sum of the matching scores obtained separately for the burst and sweep features. Table 5 summarizes the results.

For the benchmark datasets (Run 1), we observe significant improvement of the accuracy reaching an EER=2.43%. The improvement is also significant for all target FARs (e.g., 0.1%, 1%) as shown in Figure 9b. We also observe a statistically significant improvement on using fingerprints from both Run 1 and 2. The accuracy is slightly lower (EER=4.38%). These results motivate further research on feature modalities and novel integration strategies.

5.4 Summary and Discussion

In this section, we have experimentally analyzed the classification and identification capabilities of three different physical-layer features with related signal acquisition, feature extraction and matching procedures.

The results show that classification can successfully be achieved using the modulation shape of the transponder's response to a wake-up command at an out-of-specification frequency (e.g., F_c=13.06 MHz). This technique is fast, does not require special hardware and can be applied without statistically training the classification process.

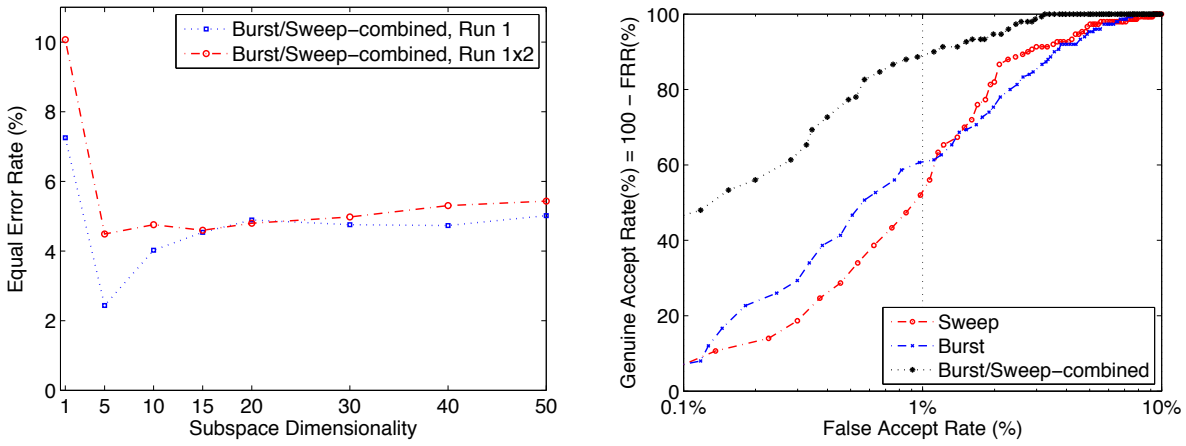
For identification, we proposed using spectral features extracted from the transponder's reaction to purpose-built burst and linear frequency sweep signals. Our proposed signal acquisition and feature extraction/matching techniques achieved separately an identification accuracy of approximately EER=5% over 50 identical RFID transponders. The proposed features are stable across acquisition runs. In addition, our spectral features showed that they can be combined in order to further improve the accuracy to EER=2.43%.

The results also confirm that using the first 5 eigenvectors is sufficient to represent the proposed features while keeping the identification accuracy high. Therefore, our proposed features also form very compact and computationally efficient fingerprints. Typically, if each dimension is represented by a 4-byte floating-point number, the size of the corresponding feature template $h = \{\hat{G}; \Sigma_G\}$ is 20 (5×4) bytes for $\hat{G}$ and 100 (5x5x4) bytes for the square covariance matrix Σ_G resulting in a total of 120 bytes.

In terms of feature extraction performance, given the much lower dimensionality of the burst samples (40000 vs. 960000 for the sweep), they are much faster to digitally acquire and extract with approximately 2 sec. compared to 26 sec. for the sweep data samples. The times are measured on a machine with 2.00 GHz CPU, 2 GB RAM running Linux Ubuntu. It should be noted that all the components of the feature extraction can be implemented efficiently in hardware which would significantly improve the performance.

6 Application to Cloning Detection

The classification and identification results presented in Section 5 indicate that physical-layer fingerprinting can be practical in a controlled environment. In this section,



(a) (b)

Figure 9: a) The identification accuracy combining the sweep and burst features b) Receiver Operating Characteristic (ROC) for N=15 for burst and sweep spectral features and their combination. 50 identical transponders are used. The subspace dimension is fixed to 5. See Table 5 for the underlying data.

Table 5: Summary of accuracy when a combination of burst and sweep features used (50 transponders).

Type	Run	N	Test matchings		Threshold T	EER (%)	EER CI (%)		Validation
			Genuine	Imposter			lower	upper	
Burst/Sweep	1	15	150	11025	1.56	2.43	1.54	3.33	4-fold
Burst/Sweep	1x2	15	200	9800	2.18	4.38	3.9	4.9	4-fold

we discuss how it could be used in the context of product or document cloning detection. We point out however that the cloning detection will obey to the achieved error rates. Despite a number of protective measures, it has been recently shown [18, 34, 33, 47] that even RFID transponders in electronic identity documents can be successfully cloned, even if the full range of protective measures specified by the standard [3], including active authentication, is used. We consider the physical-layer fingerprinting described in this work as an additional efficient mechanism that can be used to detect document counterfeiting.

We foresee two use cases in which fingerprints can be applied for anti-counterfeiting. In the first use case, the fingerprints are measured before RFID deployment and are stored in a back-end database, indexed with the unique transponder (document) identifier. When the authenticity of the document with identifier ID is verified, the fingerprint of the document transponder is measured, and then compared with the corresponding transponder fingerprint of document ID stored in the database. In order to successfully clone the document, the attacker needs to perform two tasks:

1. Obtain the fingerprint template of the transponder in the original document and
2. Produce or find a document (transponder) with the same fingerprint.

In order to extract a fingerprint template the attacker needs to fully control the target document (hold it in possession) for long enough to complete the extraction. Using the methods from our study, it would be hard, if not infeasible, for the attacker to extract the same fingerprints remotely (e.g., from few meters away). In our experiments, such remote feature extraction process resulted in an EER of approximately 50%. We assume that this is due to the change of acquisition antenna orientation and lower signal-to-noise ratio. We do not exclude the possibility that other discriminant features could be found that could be extracted remotely. However, this does not appear to be the case for our features. After obtaining the original fingerprint, the attacker now needs to produce or find an RFID transponder with that fingerprint (i.e., such that it corresponds to the one of the original document), which is hard given that the extracted fingerprints are due to manufacturing process variation. Although manufacturing process variation effects the RFID micro-controller itself, it is likely that the main source of detectable variation lies in the RFID radio circuitry. However, we cannot conclude with certainty which component of the entire transponder circuit contributes most to the fingerprints. We leave this determination to future

work. Because of the complexity of these circuits this is a challenging task in the lab, let alone in "the wild" environment of the attacker.

In the second use case, transponder fingerprints are measured before their deployment as in the first case, but are stored on the transponders instead of in a back-end database. Here, we assume that the fingerprints stored on the transponders are digitally signed by the document-issuing authority and that they are protected from unauthorized remote access; the digital signature binds the fingerprint to the document unique identifier, and both are stored on the transponder. When the document authenticity is validated, the binding between the document ID and the fingerprint stored on the transponder is ensured through cryptographic verification of the authority's signature. If the signature is valid, the stored fingerprint is compared to the measured fingerprint of the document transponder. The main advantage in this use case is that the document authenticity can be verified "off-line". The main drawback is that the fingerprint is now stored on the transponder and without appropriate access protection, it can be remotely obtained by the attacker. Here, minimal access protection can be ensured by means of e.g., Basic Access Authentication [3] although, that mechanism has been shown to have some weaknesses due to predictable document numbers [33]. As we mentioned in Section 5.4, our technique generates compact fingerprints, which can be stored in approximately 120 bytes. This means that they can easily be stored in today's e-passports. The ICAO standard [3] provides space for such storage in files EF.DG[3-14], which are left for additional biometric and future use; transponder fingerprints can be stored in those files. Our proposal does not require the storage of a new public key or maintenance of a separate public-key infrastructure, since the integrity of the fingerprints, stored in EF.DG[3-14] will be protected by the existing passive authentication mechanisms implemented in current e-passports.

The closest work to ours in terms of transponder cloning protection is the work of Devadas et al. [12], where the authors propose and implement Physically Unclonable Function(PUF)-Based RFID transponders. Processors in these transponders are specially designed and contain special circuits, PUFs, that are hard to clone and thus prevent transponder cloning. The main difference between PUF-based solutions and our techniques is that our techniques can be used with existing RFID transponders, whereas PUF-based solutions can detect cloning only of PUF-based transponders. However, PUF-based solutions do have an advantage that they rely on "controlled" randomness, unlike our techniques, that relies on randomness that is unintentionally introduced in the manufacturing of the RFID tags.

7 Related Work

Besides PUF-based RFIDs [12], that we discuss in the previous section, the following works relate to ours.

In [41], Richter et al., report on the possibility of detecting the country that issued a given passport by looking at the bytes that an e-passport sends as a reply in response to some carefully chosen commands from the reader. This technique therefore enables classification of RFID transponders used in e-passports. Our technique differs from that proposal as it enables not only classification, but also identification of individual passports. Equally, the technique proposed in [41] cannot be used for cloning detection since the attacker can modify the responses of a tag on a logical level.

The proliferation of radio technologies has triggered a number of research initiatives to detect illegally operated radio transmitters [44, 45, 23], mobile phone cloning [30], defective transmission devices [48] and identify wireless devices [20, 22, 43, 40, 39, 9] by using physical characteristics of the transmitted signals [15]. Below, we present the most relevant work to ours in terms of signal similarities, features and objectives.

Hall et al. [20, 21] explored a combination of features such as amplitude, phase, in-phase, quadrature, power and DWT of the transient signal. The authors tested on 30 IEEE 802.11b transceivers from 6 different manufacturers and scored a classification error rate of 5.5%. Further work on 10 Bluetooth transceivers from 3 manufacturers recorded a classification error rate of 7% [22]. Ureten et al. [39] extracted the envelope of the instantaneous amplitude by using the Hilbert transformation and classified the signals using a Probabilistic Neural Network (PNN). The method was tested on 8 IEEE 802.11b transceivers from 8 different manufacturers and registered a classification error rate of 2%-4%. Rasmussen et al. [40] explored transient length, amplitude variance, number of peaks of the carrier signal and the difference between mean and maximum value of the transient. The features were tested on 10 identical Mica2 (CC1000) sensor devices (approx. 15cm from the capturing antenna) and achieved a classification error rate of 30%. Brik et al. [9] proposed a device identification technique based on the variance of modulation errors. The method was tested on 100 identical 802.11b NICs (3-15 m from the capturing antenna) and achieved a classification error rate of 3% and 0.34% for k-NN and SVM classifiers respectively. In [11] the authors demonstrate the feasibility of transient-based Tmote Sky (CC2420) sensor device identification with an EER of 0.24%. The same work considered the stability of the proposed fingerprint features with respect to capturing distance, antenna polarization and voltage, and related attacks on the identification system.

8 Conclusion

In this work we performed the first comprehensive study of physical-layer identification of RFID transponders. We showed that RFID transponders have stable fingerprints related to physical-layer properties which enable their accurate identification. Our techniques are based on the extraction of the modulation shape and spectral features of the response signals of the transponders to the in- and out- of specification reader signals. We tested our techniques on a set of 50 RFID smart cards of the same manufacturer and type and we showed that these techniques enable the identification of individual transponders with an Equal Error Rate of 2.43% (single run) and 4.38% (two runs). We further applied our techniques to a smaller set of electronic passports, where we obtained a similar identification accuracy. We tested the classification accuracy of our techniques, and showed that they achieve 0% average classification error for a set of classes corresponding to manufacturers and countries of issuance. Finally, we analyzed possible applications of the proposed techniques to the detection of cloned products and documents.

Acknowledgements

This work was partially supported by the Zurich Information Security Center. It represents the views of the authors.

References

[1] Fingeprint verification competitions (FVC). http://bias.csr.unibo.it/fvc2006/.

[2] IBM JCOP family. ftp://ftp.software.ibm.com/software/ pervasive/info/JCOP_Family.pdf.

[3] ICAO. http://www.icao.int/.

[4] ISO/IEC 14443 standard. http://www.iso.org/.

[5] RFID security and privacy lounge. http://www.avoine.net/rfid /index.html.

[6] AVOINE, G., AND OECHSLIN, P. RFID traceability: A multilayer problem. In *Financial Cryptography* (2005), A. Patrick and M. Yung, Eds., vol. 3570 of *LNCS*, pp. 125–140.

[7] BISHOP, C. *Pattern Recognition and Machine Learning*. Springer, 2006.

[8] BOLLE, R., CONNELL, J., PANKANTI, S., RATHA, N., AND SENIOR, A. *Guide to Biometrics*. Springer, 2003.

[9] BRIK, V., BANERJEE, S., GRUTESER, M., AND OH, S. Wireless device identification with radiometric signatures. In *Proc. ACM MobiCom* (2008).

[10] COSTEN, N., PARKER, D., AND CRAW, I. Effects of high-pass and low-pass spatial filtering on face identification. *Perception & Psychophysics 58*, 4 (1996), 602–612.

[11] DANEV, B., AND ČAPKUN, S. Transient-based identification of wireless sensor nodes. In *Proc. ACM/IEEE IPSN* (2009).

[12] DEVADAS, S., SUH, E., PARAL, S., SOWELL, R., ZIOLA, T., AND KHANDELWAL, V. Design and implementation of PUF-based "unclonable" RFID ICs for anti-counterfeiting and security applications. *Proc. IEEE Intl. Conf. on RFID* (2008), 58–64.

[13] DIMITRIOU, T. A lightweight RFID protocol to protect against traceability and cloning attacks. In *Proc. ICST SecureComm* (2005).

[14] DUC, D. N., PARK, J., LEE, H., AND KIM, K. Enhancing security of EPCglobal Gen-2 RFID tag against traceability and cloning. In *Proc. Symposium on Cryptography and Information Security* (2006).

[15] ELLIS, K., AND SERINKEN, N. Characteristics of radio transmitter fingerprints. *Radio Science 36* (2001), 585–597.

[16] EPCGLOBAL. Architecture framework v. 1.2. standard, 2007. http://www.epcglobalinc.org/standards/ architecture/architecture_1_2-framework-20070910.pdf.

[17] FELDHOFER, M., DOMINIKUS, S., AND WOLKERSTORFER, J. Strong authentication for RFID systems using the AES algorithm. In *Workshop on Cryptographic Hardware and Embedded Systems* (2004), M. Joye and J.-J. Quisquater, Eds., vol. 3156 of *LNCS*, pp. 357–370.

[18] GRUNWALD, L. Cloning ePassports without active authentication. In *BlackHat* (2006).

[19] HALAMKA, J., JUELS, A., STUBBLEFIELD, A., AND WESTHUES, J. The security implications of VeriChip^{TM}cloning. Manuscript in submission, 2006.

[20] HALL, J., BARBEAU, M., AND KRANAKIS, E. Enhancing intrusion detection in wireless networks using radio frequency fingerprinting. In *Proc. CIIT* (2004).

[21] HALL, J., BARBEAU, M., AND KRANAKIS, E. Radio frequency fingerprinting for intrusion detection in wireless networks. *Submission to IEEE TDSC (Electronic Manuscript)* (2005).

[22] HALL, J., BARBEAU, M., AND KRANAKIS, E. Detecting rogue devices in bluetooth networks using radio frequency fingerprinting. In *Proc. CCN* (2006).

[23] HIPPENSTIEL, R., AND PAYAL, Y. Wavelet based transmitter identification. In *Proc. ISSPA* (1996).

[24] JAIN, A., PRABHAKAR, S., AND CHEN, S. Combining multiple matchers for a high security fingerprint verification system. In *Pattern Recognition Letters* (1999).

[25] JUELS, A. Minimalist cryptography for low-cost RFID tags. In *Intl. Conf. on Security in Communication Networks* (2004), C. Blundo and S. Cimato, Eds., vol. 3352 of *LNCS*, pp. 149–164.

[26] JUELS, A. Strengthening EPC tags against cloning. Manuscript, 2005.

[27] JUELS, A. Rfid security and privacy: A research survey. *IEEE Journal on Selected Areas in Communications 24*, 2 (2006).

[28] JUELS, A., PAPPU, R., AND PARNO, B. Unidirectional key distribution across time and space with applications to RFID security. In *Proc. 17th USENIX Security Symposium* (2008), pp. 75–90.

[29] JUELS, A., RIVEST, R., AND SZYDLO, M. The blocker tag: Selective blocking of RFID tags for consumer privacy. In *Proc. ACM CCS* (2003), pp. 103–111.

[30] KAPLAN, D., AND STANHOPE, D. Waveform collection for use in wireless telephone identification, 1999.

[31] KERSCHBAUM, F., AND SORNIOTTI, A. RFID-based supply chain partner authentication and key agreement. In *Proc. ACM WiSec* (2009).

[32] KITTLER, J., HATEF, M., DUIN, R., AND MATAS, J. On combining classifiers. *IEEE Trans. on Pattern Analysis and Machine Intelligence 20*, 3 (1998).

[33] LAURIE, A. Reading ePassports with predictable document numbers. In *news report* (2006).

[34] M, W. Cloning ePassports with active authentication enabled. In *What The Hack* (2005).

[35] MANLY, B. *Multivariate Statistical Methods: A Primer*, 3rd ed. Chapman & Hall, 2004.

[36] MARPLE, S. Computing the discrete-time analytic signal via FFT. *IEEE Trans. on Signal Processing 47*, 9 (1999).

[37] MITRA, M. Privacy for RFID systems to prevent tracking and cloning. *Intl. Journal of Computer Science and Network Security 8*, 1 (2008), 1–5.

[38] OPPENHEIM, A., SCHAFER, R., AND BUCK, J. *Discrete-Time Signal Processing*, 2nd ed. Prentice-Hall Signal Processing Series, 1998.

[39] O.URETEN, AND N.SERINKEN. Wireless security through RF fingerprinting. *Canadian J. Elect. Comput. Eng. 32*, 1 (Winter 2007).

[40] RASSMUSSEN, K., AND CAPKUN, S. Implications of radio fingerprinting on the security of sensor networks. In *Proc. SecureComm* (2007).

[41] RICHTER, H., MOSTOWSKI, W., AND POLL, E. Fingerprinting passports. In *NLUUG Spring Conference on Security* (2008).

[42] ROSS, A., AND JAIN, A. Multimodal biometrics: An overview. In *Proc. EUSIPCO* (2004).

[43] TEKBAS, O., URETEN, O., AND SERINKEN, N. Improvement of transmitter identification system for low SNR transients. In *Electronic Letters* (2004).

[44] TOONSTRA, J., AND KISNER, W. Transient analysis and genetic algorithms for classification. In *Proc. IEEE WESCANEX* (1995).

[45] TOONSTRA, J., AND KISNER, W. A radio transmitter fingerprinting system ODO-1. In *Canadian Conf. on Elect. and Comp. Engineering* (1996).

[46] VAJDA, I., AND BUTTYÁN, L. Lightweight authentication protocols for low-cost RFID tags. In *Proc. 2nd Workshop on Security in Ubiquitous Computing – Ubicomp* (2003).

[47] VANBEEK, J. ePassports reloaded. In *BlackHat* (2008).

[48] WANG, B., OMATU, S., AND ABE, T. Identification of the defective transmission devices using the wavelet transform. *IEEE PAMI 27*, 6 (2005), 696–710.

CCCP: Secure Remote Storage for Computational RFIDs

Mastooreh Salajegheh[1] Shane Clark[1] Benjamin Ransford[1] Kevin Fu[1] Ari Juels[2]
[1]*Department of Computer Science, University of Massachusetts Amherst*
[2]*RSA Laboratories, The Security Division of EMC*
{*negin, ssclark, ransford, kevinfu*}*@cs.umass.edu, ajuels@rsa.com*

Abstract

Passive RFID tags harvest their operating energy from an interrogating reader, but constant energy shortfalls severely limit their computational and storage capabilities. We propose *Cryptographic Computational Continuation Passing* (CCCP), a mechanism that amplifies programmable passive RFID tags' capabilities by exploiting an often overlooked, plentiful resource: low-power radio communication. While radio communication is more energy intensive than flash memory writes in many embedded devices, we show that the reverse is true for passive RFID tags. A tag can use CCCP to checkpoint its computational state to an untrusted reader using less energy than an equivalent flash write, thereby allowing it to devote a greater share of its energy to computation.

Security is the major challenge in such remote checkpointing. Using scant and fleeting energy, a tag must enforce confidentiality, authenticity, integrity, and data freshness while communicating with potentially untrustworthy infrastructure. Our contribution synthesizes well-known cryptographic and low-power techniques with a novel flash memory storage strategy, resulting in a secure remote storage facility for an emerging class of devices.

Our evaluation of CCCP consists of energy measurements of a prototype implementation on the batteryless, MSP430-based WISP platform. Our experiments show that—despite cryptographic overhead—remote checkpointing consumes less energy than checkpointing to flash for data sizes above roughly 64 bytes. CCCP enables secure and flexible remote storage that would otherwise outstrip batteryless RFID tags' resources.

1 Introduction

Research involving low-energy computing systems has long treated radio as an energy-hungry resource to be used sparingly. Our work uncovers a key resource in which programmable passive RFID tags differ from higher-powered wireless embedded devices such as motes: *radio communication consumes less energy than persistent local storage*. We exploit radio as a resource to amplify the storage capabilities of an emerging class of batteryless, programmable devices called *computational RFIDs* (CRFIDs) [7, 26, 27].

The main idea of this paper is that a CRFID can securely use radio communication as a less energy-intensive alternative to local, flash-based storage. The smaller energy requirements of radio allow the CRFID either to devote more energy to computation or to accomplish the same tasks using less energy, which may translate into a longer operating range. We use established cryptographic mechanisms to protect against untrustworthy RFID readers that could attempt to violate the confidentiality, authenticity, integrity, and freshness of the data on a CRFID. However, the cryptographic overhead threatens to eliminate the energy advantage of remote storage. Thus, the main challenge is to design an energy-saving remote storage system that provides security under the constraints of passive RFID systems.

This paper uses computational state checkpointing as an example of an application that benefits from our techniques. *Cryptographic Computational Continuation Passing* (CCCP) enables CRFIDs to perform sophisticated computations despite limited energy and continual interruptions of power that lead to complete loss of the contents of RAM. CCCP extends the Mementos architecture [26] for execution checkpointing by securely storing a CRFID's computational state on the untrusted RFID reader infrastructure that powers the CRFID, thereby making program execution on CRFIDs robust against loss of power. The design of CCCP is motivated by (1) a desire to minimize the amount of energy devoted to flash memory writes and (2) the observation that a CRFID's backscatter transmission is surprisingly efficient compared to alternatives such as active radio (like that found in motes) or flash memory writes.

Our contribution in this paper is the synthesis of sev-

eral existing ideas with techniques that are specifically applicable to computational RFIDs:

- We describe the design and implementation of CCCP, a *secure remote storage protocol that suits the characteristics and constraints of CRFIDs*, and we show how this protocol can be used in the contexts of execution checkpointing and external data storage on an untrusted RFID reader infrastructure (Sections 3, 4).
- Motivated by a desire to save energy when storing CCCP's numeric counters to nonvolatile memory, we introduce *hole punching* (Section 3.4.4), a unary encoding technique that allows a counter stored in flash memory to be updated economically, minimizing energy- and time-intensive flash erase operations. For a CRFID, less frequent flash erasure means more energy available for computation.

Since CCCP involves communication with a potentially untrustworthy RFID reader, it must ensure the integrity, confidentiality, and data freshness of checkpointed messages. For message integrity, CCCP employs UMAC [4], a Message Authentication Code (MAC) scheme based on universal hash functions (UHF) that involves the application of a cryptographically secure pseudorandom pad. Remotely stored messages in CCCP are encrypted for confidentiality using a simple stream cipher. CCCP's frequent use of key material motivates the use of opportunistic precomputation: when a CRFID is receiving abundant energy, CCCP generates and stores keystream bits in flash memory for later consumption. CCCP maintains a small amount of its own state in local nonvolatile memory, including a counter that must be updated during checkpoint operations when energy may be low. To minimize the energy required to update the counter, CCCP employs hole punching.

Conventional passive RFID tags perform rudimentary computation, often in extremely tight real-time constraints using nonprogrammable finite state machines [1], but CRFIDs offer true general-purpose computational capabilities, broadening the range of their possible applications (Section 6). Although CRFIDs offer more flexibility, they present challenging resource constraints. While sensor motes, which rely on batteries for power, often have an active lifetime measured in weeks or months, a CRFID may be able to compute for less than a second given a burst of energy, and may receive such bursts in quick succession—putting CRFIDs in an entirely different class with regard to energy constraints. Moreover, although CRFIDs have a small amount of flash memory available as nonvolatile storage, writing to this flash memory is energy intensive (Section 2).

Because CRFIDs are new and prototypes are not yet widely available for use in the laboratory, there is little previous work describing their applications or limitations; Section 7 summarizes relevant work that has appeared to date. CCCP extends a recent execution checkpointing system called Mementos [26] by adding remote, rather than local flash-based, storage capabilities to CRFIDs. While systems such as Mementos investigate how to effectively store checkpoints locally in trusted flash memory to achieve computational progress on CRFIDs despite power interruptions, CCCP focuses on using external, *untrusted* resources to increase tag storage capacity in a secure and energy-efficient manner.

2 Computational RFIDs: Background, Observations, Challenges

Consistent with the usage of RFID terminology, the term *Computational RFID* (CRFID) has two meanings: the *model* under which passively powered computers operate in concert with an RFID reader infrastructure, and the passively powered computers themselves. CRFIDs represent a class of programmable, batteryless computers [7, 26, 27]. The small size and low maintenance requirements of CRFIDs make them especially appealing for adding computational capabilities to contexts in which placing or maintaining a conventional computer would be infeasible or impossible. However, CRFID systems require that nearby, actively powered RFID readers provide energy whenever computation is to occur, a requirement that may not suit all applications.

The components of a CRFID are: a low-power microcontroller; onboard RAM; flash memory (on or off the microcontroller); energy harvesting circuitry tuned to a certain frequency (e.g., 913 MHz for EPC Gen 2 RFID); an antenna; a transistor between the antenna and the microcontroller to modulate the antenna's impedance; a capacitor for storage of harvested energy; one or more analog-to-digital converters; and optional sensors for physical phenomena such as acceleration, heat, or light. The first working example of a CRFID is the Wireless Identification and Sensing Platform, or WISP [29], a prototype device slightly smaller than a postage stamp (discounting its inches-long antenna). The WISP is built around an off-the-shelf TI MSP430 microcontroller.

Like passive RFID tags but unlike sensor motes, CRFIDs are powered solely by harvested RF energy and lack active radio components. Instead, such CRFIDs use *backscatter* communication: in the presence of incoming radio waves, a CRFID electrically modulates its antenna's impedance using a transistor, encoding binary information by varying the antenna's reflectivity. While the omission of active radio circuitry saves energy, it gives up the tag's autonomy; a CRFID can send and receive information only at the command of an RFID reader.

A CRFID's lack of autonomy is one of the factors that makes it difficult to protect.

2.1 Frequent Power Loss on Tags, but Plentiful External Resources

Several key observations motivate the development of secure remote storage for computational RFIDs.

Frequent loss of power may interrupt computation. The CRFID model posits computing devices that are primarily powered by RF energy harvesting, a mechanism that is naturally finicky because of its dependence on physical conditions. Any change to a CRFID's physical situation—such as its position or the introduction of an occluding body—may affect its ability to harvest energy. Existing systems that use RF harvesting typically counteract the effect of physical conditions by placing stringent requirements on use. For example, an RFID transit card reader presented with a card may behave in an undefined way unless the card is within 1 cm for at least 300 ms, parameters designed to ensure that the card's computation finishes while it is still near the reader. CRFID applications may preclude such a strategy: programs on general-purpose CRFIDs may not offer convenient execution time horizons, and communication distances may not be easily controlled. Without any guarantees of energy availability, it may be unreasonable to mandate that programs running on CRFIDs complete within a single energy lifecycle. As an extension of the Mementos system [26], CCCP aims to address the problem of suspending and resuming computations to facilitate spreading work across multiple energy lifecycles.

Storing remotely may require less energy than storing locally. Some amount of onboard nonvolatile memory exists on a CRFID, so an obvious approach to suspension and resumption is simply to use this local memory for state storage. However, to implement nonvolatile storage, current microcontrollers use flash memory, which imports several undesirable properties. While reading from flash consumes energy comparable to reading from volatile RAM, the other two flash operations—writing and erasing—require orders of magnitude more energy per datum (Table 3). Our measurements of a CRFID prototype reveal that the energy consumption of storing a datum locally in flash can in fact exceed the energy consumption of transmitting the same datum via backscatter communication.

To illustrate the difference between flash and radio storage on a CRFID and to show how the relationship is different on a sensor mote, we offer Figure 1. The figure helps explain why designers of mote-based systems choose to minimize radio communication; similarly, it justifies our exploration of radio-based storage as an alternative to flash-based storage on CRFIDs.

It should be noted that CCCP, although its primary data storage mechanism is the communication link between CRFIDs and readers, still requires *some* flash writes during storage operations: CCCP maintains a counter in flash to ensure that key material is not reused. However, because CCCP employs hole punching (Section 3.4.4) to maintain the counter, the amount of data written for counter updates is small compared to the amount of data that can be stored at once—small enough not to obviate the energy advantage of radio-based storage—and counter updates do not frequently necessitate erasures.

EPC Gen 2 RFID readers are typically not standalone devices. Rather, they are connected to networks or other systems (for, e.g., control or logging) that can offer computing resources such as storage. The benefit to CRFIDs that communicate with such a reader infrastructure is access to effectively limitless storage. Several kilobytes of onboard flash memory is minuscule compared to the potentially vast amount of storage available to networked RFID readers. While unlimited external storage is not obviously helpful for saving computational state—a CRFID cannot save or restore more state than it can hold locally—its usefulness as general-purpose long-term storage is analogous to the usefulness of networked storage for PCs.

RFID protocols allow arbitrary payloads. While the EPC Gen 2 protocol imposes constraints on the transmissions between RFID tags and RFID readers—for example, the maximum upstream data rate from tag to reader is 640 Kbps [13]—it also offers sufficient flexibility that CCCP can be implemented on top. In particular, the Gen 2 protocol permits a reader to issue a *Read* command to which a tag can respond with an arbitrary amount of data. Previous versions of the EPC RFID standard mandated a small response size that would have imposed severe communication overhead on large upstream transmissions.

CRFID is not married to EPC Gen 2 as an underlying protocol, but the existence of a widespread RFID reader infrastructure and the availability of commodity reader hardware makes for easy prototyping.

2.2 Challenges Due to Energy Scarcity

Several energy-related considerations limit the resources available for computation on CRFIDs, limiting the utility of CRFIDs as a general-purpose computing platform. Ransford et al. [26] discuss the difficulty of effectively utilizing a storage capacitor and enumerate the drawbacks of using capacitors for energy storage; Buettner et al. [7] discuss how energy limitations bear on the deployment of a CRFID-based system. Two key design features of CRFIDs pose energy challenges to a system like

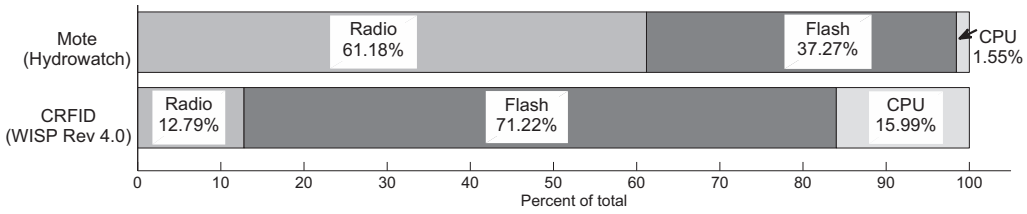


Figure 1: Per-component maximum power consumption of two embedded devices. Radio communication on the WISP requires less power than writes to flash memory. The relative magnitudes of the power requirements means that a sensor mote favors shifting storage workloads to local flash memory instead of remote storage via radio, while a computational RFID favors radio over flash. The numbers for the mote are calculated based on the current consumption numbers given by Fonseca et al. [15]. For the CRFID, we measured three operations (radio transmit, flash write, and register-to-register move) for a 128-byte payload.

CCCP: first, the voltage and current requirements of flash memory constrain the design of flash-bearing CRFIDs and limit the portion of a CRFID's energy lifecycle that is usable for computation. Second, a CRFID's reliance on energy harvesting and backscatter communication means that a CRFID cannot compute or communicate without reader contact.

Flash memory limitations. Microcontrollers that incorporate flash typically have separate threshold voltages: one threshold for computation, and a higher threshold for flash writes and erases. Because of this difference, flash writes cannot be executed at arbitrary times during computation on a CRFID; they require sufficient voltage on the storage capacitor. Without a constant supply of energy, capacitor voltage declines with time and computation, so waiting until the end of a computation to record its output to nonvolatile memory may be risky.

The size of a CRFID's storage capacitor imposes another basic limitation. Flash writes, which owe their durability to a process that effects significant physical changes, require more current and time (and therefore energy) than much simpler RAM or register writes. Per-datum measurements show that, on a WISP's microcontroller, writing to flash consumes roughly 400 times as much energy as writing to a register [26]. Such outflow from the storage capacitor can dramatically shorten the device's energy lifecycles.

Non-autonomous operation. Backscatter communication involves modulating an antenna's impedance to reflect radio waves—an operation that, for the sender, involves merely toggling a transistor to transmit binary data. Such communication cannot occur without a signal to reflect; CRFIDs, like other passive RFIDs, are therefore constrained to communicate only when a reader within range is transmitting. Computation may occur during times of radio silence, but only if sufficient energy remains in the CRFID's storage capacitor. Unlike battery-powered platforms that can operate autonomously between beacon messages from other entities, a CRFID may completely lose power between interactions with an RFID reader. Our experience shows that, lacking a source of harvestable energy, the storage capacitor on a WISP (Revision 4.0) can support roughly one second of steady computation before its voltage falls below the microcontroller's operating threshold. Such limitations constrain the design space of applications that can run on CRFIDs. For example, without autonomy, an application cannot plan to perform an action at a specific time in the future.

Unsteady energy supply. A key challenge CRFIDs face is that their supply of energy can be unsteady and unpredictable, especially under changing physical conditions. RFID readers may not broadcast continuously or even at regular intervals, and they do not promise any particular energy delivery schedule to tags. In our experiments, even within inches of an RFID reader that emitted RF energy at a steady known rate, the voltage on a CRFID's storage capacitor did not appear qualitatively easy to predict despite the fixed conditions. A CRFID's storage capacitor must buffer a potentially unsteady supply of RF energy without the ability to predict future energy availability.

3 Design of CCCP

CCCP's primary design goal is to furnish computational RFIDs with a mechanism for secure outsourced storage that facilitates the suspension and resumption of programs. This section describes how CCCP is designed to meet that goal and several others. Refer to Section 4 for a discussion of CCCP's implementation, and refer to Section 5 for an evaluation of CCCP's design choices and security; in particular, Section 5.3.1 discusses the overhead imposed by cryptographic operations.

Given a chunk of serialized computational state on a CRFID, CCCP sends the state to the reader infrastructure for storage. (CCCP is designed to work independently of the state serialization method, and does not prescribe a specific method.) In a subsequent energy lifecycle, an RFID reader that establishes communication with the tag

Design goal	Approach
Computational progress	Communicating checkpoints via radio to untrusted RFID readers
Security: authentication, integrity	UHF-based MAC
Security: data freshness	Key non-reuse; counter stored by hole punching in nonvolatile memory
Security: confidentiality	Symmetric encryption with keystream precomputation

Table 1: CCCP's design goals and techniques for accomplishing each of them.

sends back the state, CCCP performs appropriate checks, and the CRFID resumes computation where it left off. CCCP provides several operating modes that allow an application designer to increase security—by adding authentication alone, or authentication and encryption—at the cost of additional per-checkpoint energy consumption. Table 1 describes how CCCP meets each of the goals discussed in this section.

3.1 Design Goal: Computational Progress on CRFIDs

CCCP remotely checkpoints computational state to make long-running operations robust against power loss—i.e., to enable their *computational progress*. We define computational progress as change of computational state toward a goal (e.g., the completion of a loop). While CRFIDs are able to finish short computations in a small number of energy lifecycles (e.g., symmetric-key challenge-response protocols [9, 19]), the challenges described in Section 2 make it difficult for a CRFID to guarantee the computational progress of longer-running computations.

If a CRFID loses power before it completes a computation, all volatile state involved in the computation is lost and must be recomputed in the next cycle. If energy availability is similarly inadequate in subsequent cycles, the CRFID may never obtain enough energy to finish its computation or even to checkpoint its state to flash memory. We refer to such vexatious computations as *Sisyphean tasks*. (Sisyphus was condemned to roll a large stone up a hill, but was doomed to drop the stone and repeat hopelessly forever [20].) A major goal of CCCP is to prevent tasks from becoming Sisyphean by shifting energy use away from flash operations and toward less energy-intensive radio communication.

3.2 Checkpointing Strategies: Local vs. Remote

We consider two strategies for the nonvolatile storage of serialized checkpointed state. The first, writing the state to flash memory, involves finding an appropriately sized region of erased flash memory or creating one via erase operations. The second strategy, using CCCP, requires a CRFID to perform zero or more cryptographic operations (depending on the operating mode) and then transmit the result via backscatter communication.

The obvious advantage of flash memory is that its proximity to the CRFID makes it readily accessible. On-chip flash has the further advantage that it may be inaccessible to an attacker. However, the operating requirements of flash are onerous in many situations. With unlimited energy, a CRFID could use flash freely and avoid the complexity of a radio protocol such as CCCP. Unfortunately, energy is limited in ways described elsewhere in this paper, and several disadvantages of flash memory diminish its appeal as a store for checkpointed state. The most obvious disadvantage is an imbalance between the requirements for reading and writing. Write and erase operations require more time and energy per bit than reading (Table 3); additionally, the minimum voltage and current requirements are higher. For example, in the case of the MSP430F2274, read operations are supported at the microcontroller's minimum operating voltage of 1.8 V, but write and erase operations require 2.2 V. Finally, flash memory (both NOR and NAND types) generally imposes the requirement that memory segments be erased before they are written: if a bit acquires a zero value, the entire segment that contains it must be erased for that bit to return to its default value of 1. Aside from burdening the application programmer with inconvenience, erase-before-write semantics complicate considerations of energy requirements. These disadvantages are minor afflictions for higher-powered systems, but they pose serious threats to the utility of flash memory on CRFIDs.

Backscatter transmission, since it involves modulating only a single transistor to encode data, requires significantly less energy than transmission via active radio. In fact, our measurements (Figure 4) show that backscatter transmission of an authenticated, encrypted state checkpoint (plus a small amount of bookkeeping in flash) can require less energy than exclusively writing to flash memory, even after including the energy cost of encrypting and hashing the checkpointed state. Because of its consistent behavior throughout the microcontroller's operating voltage range, backscatter transmission is an especially attractive option when the CRFID receives radio contact frequently but cannot harvest energy efficiently,

in which case writing to flash may be infeasible because of insufficient energy in the storage capacitor. These circumstances may occur far from the reader, or in the presence of radio occlusions, or when a computation uses energy quickly as soon as the CRFID wakes up.

Despite its advantages over flash storage and active radio, CCCP's reliance on backscatter transmission has drawbacks. Bitrate limitations in the EPC Gen 2 protocol cause CCCP's transmissions to require up to twice as much time per datum as flash storage on some workloads. The best choice of storage strategy depends on an application's ability to tolerate delay and the necessity of saving energy.

3.3 Threat Model

We define CCCP's threat model as a superset of the attacks that typically threaten RFID systems [21]. The most obvious way an attacker can disrupt the operation of a CRFID is to starve it of energy by jamming, interrupting, or simply never providing RF energy for the CRFID to harvest. Because they depend entirely on harvestable energy, CRFIDs cannot defend against such denial-of-service (DoS) attacks, so we consider these attacks as a problem to be dealt with at a higher system level. We instead focus on two types of attacks that a CRFID can use its resources to address: (1) active and passive radio attacks and (2) attacks by an untrusted storage facility.

An adversary may attempt to:

- Eavesdrop on radio communication in both directions between a CRFID and reader.
- Masquerade as a legitimate RFID reader in order to collect checkpointed state from CRFIDs. Because CRFIDs do not trust reader infrastructure, such an attack should allow an attacker to collect only ciphertext.
- Masquerade as a legitimate RFID reader in order to send corrupted data or old data (e.g., a previous computational state) to the CRFID. Such invalid data should not trick the CRFID into executing arbitrary or inappropriate code.
- Masquerade as a specific legitimate CRFID in order to retrieve that CRFID's stored state from the reader. This state should be useless without access to the keystream material that encrypted it—keystream material that is stored in the legitimate owner's nonvolatile memory and never transmitted.

We additionally assume that an adversary cannot physically inspect the contents of a CRFID's memory.

3.4 Secure Storage in CCCP

Because computational RFIDs depend on RFID readers for energy—if a CRFID is awake, there is probably a reader nearby—readers are a natural choice for storing information. But a reader trusted to provide energy should not necessarily be trusted with sensitive information such as checkpointed state.

CCCP involves communication with untrusted reader infrastructure, so we establish several security goals:

- Authenticity: a CRFID that stores information on external infrastructure will eventually attempt to retrieve that information, and the authenticity of that information must be cryptographically guaranteed. Under CCCP, the only party that ever needs to verify the authenticity of a CRFID's stored information is the CRFID itself.
- Integrity: an untrusted reader may attempt to impede a CRFID's computational progress by providing data from which the CRFID cannot resume computation (e.g., random junk). While CCCP cannot prevent a denial of service attack in which a reader provides only junk, it guarantees that CRFIDs will compute only on data they recognize.
- Data freshness: just as a reader can provide corrupted data instead of usable data, it can replay old state in an attempt to hinder the computational progress of a computation. Under CCCP, a CRFID recognizes and rejects old state.
- Confidentiality: in certain applications, the leaking of intermediate computational state might be a critical security flaw. For other applications, confidentiality may not be necessary.

3.4.1 Keystream Precomputation

Because CCCP's threat model assumes a powerful adversary that can intercept all transmissions, CCCP never reuses keystream material when encrypting data or computing MACs. We use CCCP's refreshable pool of pseudorandom bits (a circular buffer in the CRFID's nonvolatile memory) as a cryptographic keystream to provide confidentiality and authentication.

CCCP stores keystream material on the CRFID because we assume that the CRFID trusts only itself; a CRFID cannot extract trustworthy keystream material from a reader it does not trust, nor from any observable external phenomenon (which, in our threat model, an attacker would be able to observe equally well). Because a CRFID can reserve only finite storage for storage of keystream material, the material must be periodically refreshed. CCCP opportunistically refreshes the keystream material with pseudorandom bits, following Algorithm 3.

To provide unique keystream bits to cryptographic operations (encryption and MAC), CCCP uses an existing implementation [9] of the RC5 block cipher [28] in counter mode to generate pseudorandom bits and store them to flash. The choice of a block cipher in counter mode means that the resulting MAC and ciphertext are secure against a computationally bounded adversary [6]. A stream cipher would work equally well in principle, but in implementing CCCP, we found that those under consideration required a large amount of internal read-write state. For example, the stream cipher ARC4 requires at least 256 bytes of RAM [30], whereas RC5 requires only an 8-byte counter. The RC5 key schedule is preloaded into flash memory the first time the device is programmed, and the keystream materials are generated during periods of excess energy (or *power seasons*; see § 3.5). One such period of excess energy is the CRFID's initial programming, at which time the entire keystream buffer is filled with keystream bits. To avoid reusing keystream bits, CCCP maintains several variables in nonvolatile memory. Table 2 summarizes the variables CCCP stores in nonvolatile memory.

Variable	Description
chkpt_counter	Counter representing the number of checkpoints completed; used to calculate the location of the first unused keystream material; updated each time keystream material is consumed; unary representation
kstr_end	Pointer to the end of the last chunk of unused keystream bits in keystream memory; updated during key refreshment
rc5counter	Incrementing counter used as an input to RC5 while filling keystream memory with pseudorandom data; updated during key refreshment

Table 2: Variables CCCP stores in nonvolatile memory.

3.4.2 UHF-based MAC for Authentication and Integrity

CCCP uses a MAC scheme based on universal hash functions (UHF) [8] to provide authentication and integrity. CCCP constructs the MAC by first hashing the message and then XORing the 80-bit hash with a precomputed cryptographic keystream. Because of the resource constraints of CRFIDs, it is critical to use a scheme that consumes minimal energy, and according to recent literature [4, 14], UHF-based MACs are potentially an order of magnitude faster than MACs based on cryptographic hash functions. We chose UMAC [4] as the MAC function after evaluating several alternatives. Our experiments on WISP (Revision 4.0) CRFIDs determined that UMAC takes on average 18.38 ms and requires 28.79 μJ of energy given a 64-byte input.

3.4.3 Stream Cipher for Confidentiality

To provide confidentiality, a CRFID simply XORs its computational state with a precomputed cryptographic keystream. This encryption scheme is low-cost in terms of computation and energy, but it relies on using each keystream bit at most once. CCCP ensures that the encryption and MAC functions never reuse keystream bits by keeping track of the beginning and end of fresh keystream material in flash memory. The keystream pool is represented as a circular buffer. The address of the first unused keystream material is derived from the value of *chkpt_counter* (Table 2), and the last unused keystream material ends just before the address pointed to by *kstr_end*.

If the application using CCCP demands confidentiality at all times, then if CCCP cannot satisfy a request for unused keystream bits, it pauses its work to generate more keystream bits. This behavior is inspired by that of the blocking `random` device in Linux [17].

3.4.4 Hole Punching for Counters Stored in Flash

To avoid reusing keystream material, CCCP maintains a counter (*chkpt_counter*) from which the address of the first unused keystream bits can be derived. The counter is stored in flash memory because it is used for state restoration after power loss. However, incrementing a counter stored in binary representation always requires changing a 0 bit into a 1 bit (Figure 2(a)). On segmented flash memories, changing a single bit to 1 requires the erasure (setting to 1) of the entire segment that contains it—at least 128 bytes on the MSP430F2274—before the new value can be written. An additional cost that varies among flash cells is that they wear out with repeated erasure and writing [18].

To avoid energy-intensive erasures and minimize the energy cost of writing counter updates, CCCP represents *chkpt_counter* in complemented unary instead of binary. CCCP interprets the value of such a counter as the number of 0 bits therein. Because 1 bits can be changed to 0 bits without erasure, incrementing a counter requires a relatively small write, with erasures necessary only if the unary counter must be extended into unerased memory. We call this technique *hole punching* after the visual effect of turning 1 bits into 0 bits. Since *chkpt_counter* is simply incremented at each remote checkpoint, updating the counter generally requires writing only a single

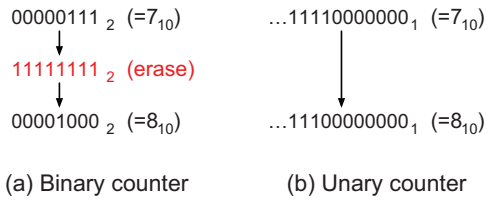


Figure 2: Illustration of hole punching. While incrementing a binary counter (a) in flash memory may require an energy-intensive erase operation, complemented unary representation ((b), with the number of zeros, or "holes," representing the counter value) allows for incrementing without erasure at a cost of space efficiency.

word. Table 3 illustrates the energy cost of erasing an entire segment and the energy cost of writing a single word.

Operation	Seg. erase	Write	Read	Write
Size (bytes)	128	128	128	2
Energy (μJ)	46.81	56.97	0.64	0.96

Table 3: Comparison of energy required for flash operations on an MSP430F2274. Hole punching often allows CCCP to use a single-word write (2 bytes on the MSP430) instead of a segment erase when incrementing a complemented unary counter.

To minimize the length of the unary *chkpt_counter*'s representation and to facilitate simple computation of offsets, CCCP assumes a fixed size for checkpointed state; in practice an application designer can choose an appropriate value for the fixed checkpoint size.

3.4.5 Extension for Long-Term Storage

Under CCCP, readers can act not only as outsourced storage for computational state, but also as long-term external storage. Because of their ultra-low-power microcontrollers, CRFIDs are likely to have only a small amount of flash available for data storage. Moreover, since flash operations are energy intensive, depending exclusively on flash memory as a storage medium is undesirable. CCCP could enable a CRFID to instead use the reader infrastructure as an external storage facility with effectively limitless space.

Long-term storage requires a different key management strategy than checkpointing data. With a temporary checkpointing system, the CRFID needs access only to the keystream material used to prepare the last checkpoint sent to a reader. However, in the case of long-term storage, the CRFID may require access to all of the data it has ever stored on the reader and therefore must remember all of the cryptographic keys from those stores. To avoid this unrealistic requirement, a potential extension to CCCP allows the CRFID to generate keys on demand when long-term storage is required.

There are two operations that CCCP can provide to a CRFID application for this purpose:

- To satisfy a STORE(*data*) request, CCCP provides a keystream generator in the form of a block cipher in counter mode; this requires a monotonically increasing counter in addition to CCCP's *chkpt_counter*. CCCP XORs the given data with the generated keystream and then constructs a MAC, then sends the ciphertext and MAC to the reader for storage. CCCP then sends the counter value back to the application.
- To satisfy a RETRIEVE(*index*) request, CCCP asks the RFID reader for the data at the given index. CCCP then generates the same keystream it used to encrypt the data by passing the index to the block cipher. Finally, CCCP verifies the MAC provided by the reader and returns the decrypted data to the application.

3.5 Power Seasons

If a CRFID could predict future energy availability, then it would be able to schedule its generation of keystream bits and ensure that it never exhausted its supply of pseudorandomness during normal operation. However, because CRFIDs lack autonomy and cannot depend on RFID reader infrastructure to provide a steady energy supply, we roughly classify the energy availability scenarios a CRFID faces into two *seasons*. We assume that the general case is a *winter* season, in which a CRFID cannot consistently harvest enough energy to perform all of its tasks. During winter, the CRFID must focus on minimizing checkpoints and wasted energy. The other season is *summer*, during which harvested energy is plentiful and the CRFID can afford to perform energy-intensive operations such as precomputation and storage of keystream material for later use.

CCCP can identify a summer season if one of two conditions is true. First, the CRFID may find itself awake with no computations left to complete, for example after it has finished a sensor reading. Second, the CRFID may find itself communicating with a reader that does not understand CCCP and simply provides harvestable energy.

4 Implementation

The components of CCCP span two environments: CRFIDs and RFID readers. On a CRFID, CCCP accepts data from an application and uses the CRFID's backscatter mechanism to ship the data to a reader. The reader

(which we consider as an RFID reader plus a controlling computer) is programmed to participate in the CCCP protocol and return computational state where necessary. This section describes the CRFID-side components, the reader-side components, and the protocol that ties them together.

The CRFID side of CCCP is implemented in the C programming language on WISP (Revision 4.0) prototypes. At its core are three primary routines, which we present in pseudocode: CHECKPOINT (Algorithm 1), RESUME (Algorithm 2), and KEY-REFRESH (Algorithm 3). CHECKPOINT and RESTORE refer to a counter called *chkpt_counter* from which CCCP derives the address of the first unused keystream material. For routines that require radio communication, we borrow radio code from Intel's WISP firmware version 1.4. Note that, since a CRFID cannot assume that a reader is listening at an arbitrary time, the TRANSMIT subroutine waits for an interrupt indicating that the CRFID has received a go-ahead message from the reader.

The RFID reader side of CCCP consists only of code to drive the reader appropriately for communication events. Because of the Gen 2 protocol's complexity, we have not completely implemented the reader side of the CCCP protocol. Rather than write a large amount of code for the reader, we chose to use simple control programs for the reader and inspect the exchanged messages manually, a strategy that allowed us to concentrate on the more resource-constrained CRFID side of the system while avoiding porting applications from one proprietary reader to another. (The WISP [Revision 4.0] is nominally compatible with only the Alien ALR-9800 and Impinj Speedway readers; we chose to use a desktop PC to program these readers for the sake of simplicity and portability.) A full implementation of the reader side would properly parse each message received from the CRFID and manage storage for checkpointed state.

4.1 Communication Protocol

The CRFID model places a number of restrictions on communication. The only communication hardware on a CRFID is a backscatter circuit involving an antenna and a modulating transistor; an active radio would require significantly more energy. Since backscatter simply reflects an incoming carrier signal, a prerequisite for communication is that the reader emits an appropriate carrier signal. In our experiments, we used two different EPC Gen 2-compatible RFID readers that are readily available as off-the-shelf products; we used no nonstandard reader hardware or antennas.

CCCP's communication protocol is based on primitives provided by the EPC Gen 2 RFID protocol (the RFID protocol the WISP understands). Specifically, CCCP makes use of three EPC Gen 2 commands:

- A reader issues a *Query* command to a specific tag (in our case, a CRFID). The *Query* command comprises a 4-tuple: $\langle$*action*, *membank*, *pointer*, *length*$\rangle$. While a conventional RFID tag may require reasonable values for all four tuple members, a CRFID need examine only the fourth member to learn the maximum reply length the reader will accept. The reader can use the other three fields to encode meta-information such as whether the reader wants to offer checkpointed state to the CRFID.

- A reader issues a *Read* command to a specific tag to request an arbitrary amount of data from an RFID tag's memory. A CRFID can respond to a coordinated *Read* command with a chunk of checkpointed state.

- A reader issues a *Write* command to send data for storage in a specific tag's memory. Because RFID tags tend to have fewer resources even than CRFIDs, *Write* commands transmit only a small amount (16 bits) of data. A CRFID can request a series of *Write* commands from the reader to retrieve checkpointed state, then reassemble the results in memory and restore its state from the checkpoint.

Figure 3 gives an overview of CCCP's message types and their ordering. CCCP does not require protocol changes to the EPC Gen 2 standard, but it requires that an RFID reader be controlled by an application that understands CCCP. While a proprietary radio protocol for CCCP could be more efficient than one built atop an existing RFID protocol, a goal of CCCP—inherited from the design goals of the WISP CRFID—is to maintain compatibility with existing RFID readers.

5 System Evaluation

This section justifies our design choices and offers evidence for our previous claims. We evaluate the security properties of four distinct checkpointing strategies—three based on CCCP's radio transmission and one on local flash storage—and describe how CCCP provides data integrity with or without confidentiality. We describe our experimental setup and methods, then provide empirical evidence that CCCP's radio-based checkpointing requires less energy per checkpoint than a flash-based strategy. Finally, we characterize the overhead incurred by CCCP's cryptographic operations in terms of both energy and the keystream material that they consume.

Algorithm 1 The CHECKPOINT routine encrypts, MACs, and transmits a fixed-size (*STATE_SIZE*, selected by the application designer) chunk of computational state. $\langle A, B \rangle$ means the concatenation of A and B with a delimiter in between. 80 bits is the fixed output size of NH, the hash function used by UMAC. For arithmetic simplicity, this pseudocode treats the *keystream* pool as an infinite array.

CHECKPOINT(*state*, *keystream*, *chkpt_counter*)

1 ≻ Compute the (constant) amount of keystream material that will be used in this invocation
2 $chkpt\_size = STATE\_SIZE + \text{LENGTH}(\langle state, chkpt\_counter \rangle) + 80$ bits
3
4 $k \leftarrow chkpt\_counter \times chkpt\_size$ ≻ $keystream[k]$ holds unused keystream material
5 $chkpt\_counter \leftarrow chkpt\_counter + 1$ ≻ Update *chkpt_counter* in nonvolatile memory
6
7 $C \leftarrow state \oplus keystream[k \ldots k + STATE\_SIZE - 1]$ ≻ Encrypt *state* by XORing with keystream material
8 $k \leftarrow k + STATE\_SIZE$ ≻ . . . and advance k
9
10 $H \leftarrow \text{NH}(\langle C, k \rangle, keystream[k \ldots k + \text{LENGTH}(\langle C, k \rangle) - 1])$ ≻ Hash the encrypted state
11 $k \leftarrow k + \text{LENGTH}(\langle C, k \rangle)$ ≻ . . . and advance k
12
13 $M \leftarrow H \oplus keystream[k \ldots k + \text{LENGTH}(H) - 1]$ ≻ Construct an 80-bit MAC
14
15 TRANSMIT(C, M) ≻ Note: TRANSMIT blocks until a reader is detected

5.1 Security Semantics

CCCP trades the physical security of local storage for the energy savings of remote storage, but its use of radio communications introduces different security properties. We consider CCCP's four operating modes in increasing order of cryptographic complexity. Note that the algorithm listings (Algorithms 1–3) describe the most computationally intensive operating mode; the other modes involve subsets of its operations.

- Under CCCP's threat model, storing checkpointed state only in local flash memory is the most secure option, since it involves no radio transmission at all. However, for reasons detailed elsewhere in this paper, writing to flash memory is not always possible or desirable. We call the flash-only approach *Mementos* after the system [26] that inspired CCCP.
- In a mode called *CCCP/NoSec*, a CRFID sends computational state in plaintext. Under CCCP's threat model, CCCP/NoSec allows an attacker to intercept computational state and trivially recover the information it contains.
- In a mode called *CCCP/Auth*, the CRFID computes a message authentication code (MAC), attaches it to plaintext computational state, and transmits both. To trick a CRFID into accepting illegitimate state, an attacker must craft a message that incorporates a MAC that the CRFID can verify. However, since CCCP's MAC routine incorporates keystream material that is local to the CRFID, the attacker must guess the contents of a chunk of the CRFID's keystream memory, which requires brute force under our threat model.
- In a mode called *CCCP/AuthConf*, CCCP encrypts computational state, computes a MAC, and transmits both (Algorithm 1). As with CCCP/Auth, an attacker who wants to trick a CRFID into accepting illegitimate state must find a hash collision; however, part of her colliding input must be a valid *encrypted* computational state from which the CRFID would be able to resume. Since CCCP does not reuse keystream material, the attacker is limited to brute-force search to find such an encrypted state.

5.2 Experimental Setup & Methods

We used a consistent experimental setup to obtain timing and energy measurements for a prototype CRFID. We programmed a WISP with a task (e.g., a flash write) and set a GPIO pin to toggle immediately before and after the task. We then charged the WISP's capacitor to 4.5 V using a DC power supply, disconnected the power supply so that the storage capacitor was the only source of energy for the WISP, and observed the task's execution and storage capacitor's voltage on an oscilloscope. We delivered energy directly from a DC power supply when taking measurements because the alternative, providing an RF energy supply, results in unpredictable and unsteady

Algorithm 2 The RESUME routine receives an encrypted checkpoint C and a message authentication code M from a reader, then restores the computational state of the CRFID if the received data pass an authenticity test. *chkpt_counter* is the value stored in nonvolatile memory at the beginning of CHECKPOINT (Algorithm 1). We assume that, since k and *chkpt_counter* are both numbers, their in-memory representations have the same length. As in Algorithm 1, this pseudocode treats the *keystream* pool as an infinite array for arithmetic simplicity. $\langle A, B \rangle$ means the concatenation of A and B with a delimiter in between. 80 bits is the fixed output size of NH, the hash function used by UMAC.

```
RESUME(C, M, keystream, chkpt_counter)
 1  ≻ Find the first unused keystream material, then backtrack to find the keystream material CHECKPOINT used to
    hash and MAC the ciphertext
 2  chkpt_size = STATE_SIZE + LENGTH(⟨C, chkpt_counter⟩) + 80 bits      ≻ N.b.: STATE_SIZE = LENGTH(C)
 3  k ← chkpt_counter × chkpt_size
 4  k ← k − (LENGTH(⟨C, k⟩) + 80 bits)
 5
 6  H ← NH(⟨C, k⟩, keystream[k ... k + LENGTH(⟨C, k⟩) − 1])      ≻ Compute the ciphertext's hash
 7  k ← k + LENGTH(⟨C, k⟩)                                       ≻ ... and advance k to point to the MAC
 8
 9  if M = H ⊕ keystream[k ... k + LENGTH(H) − 1]                ≻ If the MAC is OK, then...
10     then k ← k − (LENGTH(C) + LENGTH(⟨C, k⟩))                 ≻ backtrack further ...
11          state = C ⊕ keystream[k ... k + LENGTH(C) − 1]       ≻ and decrypt C to yield state
12          RESTORE-STATE(state)
13     else ≻ Do nothing
```

charge accumulation, making it difficult to shut off the energy supply at a precise capacitor voltage.

After watching the GPIO pin signal the beginning and end of the task, we calculated the task's duration and the corresponding change in the storage capacitor's voltage. When an operation completed too quickly to observe clearly on the oscilloscope, we repeated it in an unrolled loop and divided our measurements by the number of repetitions. Finally, we calculated per-bit energy values by subtracting the baseline energy consumption of the WISP with its MSP430 microcontroller in the LPM3 low-power (sleep) mode. We subtract the WISP's baseline energy consumption in order to discount the effects of omnipresent consumers such as RAM and CPU clocks. For all measurements that we present, we give the average of five trials.

5.3 Performance

Figure 4 shows that, for data sizes greater than 16 bytes, a checkpoint operation under CCCP/NoSec requires less energy than a checkpoint to flash. Under CCCP/AuthConf, which adds encryption and MAC operations, a similar threshold exists between 64 and 128 bytes. Checkpointing via flash has an additional cost: if the checkpointing mechanism needs to overwrite existing data (e.g., old checkpoints) in flash memory, it must erase the corresponding flash segments and potentially replace whatever data it did not overwrite. Even if a flash write does not necessitate an immediate erasure, it makes less space available in the flash memory and therefore increases the probability that a long-running application will eventually need to erase the data it wrote—that is, it incurs an *energy debt*. In the ideal case, an application can pay its energy debt easily if erasures happen to occur only when energy is abundant—i.e., in summer power seasons. However, since CCCP is designed to address scenarios in which energy availability fluctuates, we consider the case in which each write incurs an energy debt. Factoring in debt, we characterize the energy cost of a write of size *dsize* as

$$\text{Cost}^*(\text{write}(dsize)) = \text{Cost}(\text{seg. erase}) \times \frac{dsize}{\text{Size}(\text{seg.})} + \text{Cost}(\text{write}(dsize)).$$

In practice, because some erasures will likely occur in summer power seasons and some in winter power seasons, the energy cost of a flash write of size *dsize* falls between $\text{Cost}(\text{write}(dsize))$ (the ideal cost) and $\text{Cost}^*(\text{write}(dsize))$ (the worst-case cost), inclusive.

The energy measurements we present in this paper (e.g., in Figure 4) fail in some cases to strongly support the hypothesis that radio-based checkpointing is consistently less energy intensive than flash-based checkpointing. The imbalance is due to a missed opportunity for optimization on the WISP prototype. The transistor used

Algorithm 3 The KEY-REFRESH replaces used keystream material with new keystream material in nonvolatile memory. Unlike in CHECKPOINT and RESUME, this pseudocode treats the *keystream* pool as a fixed-size circular buffer. This allows us to treat keystream material between k and *kstr_end* as unused, and the rest—between *kstr_end* and k—as used. This pseudocode omits two subtleties for simplicity: first, the routine must not erase keystream material that is waiting to be used by RESUME. Second, because flash erasure affects entire segments at once, the ERASE-MEMORY-RANGE routine must sometimes restore data that should not have been erased.

KEY-REFRESH(*keystream*, *kstr_end*, *chkpt_counter*, *rc5counter*)

```
1  ≻ Find the first unused keystream material in the circular keystream buffer
2  chkpt_size = STATE_SIZE + (STATE_SIZE + LENGTH(⟨null, chkpt_counter⟩)) + 80 bits
3  k ← chkpt_counter × chkpt_size (mod LENGTH(keystream) / chkpt_size)
4
5  ≻ Erase all used keystream memory, then write pseudorandom data to it
6  ERASE-MEMORY-RANGE(keystream[kstr_end ... k])
7  i ← kstr_end
8  while (i < k)
9      do rc5counter ← rc5counter + 1          ≻ Update counter in nonvolatile memory
10        keystream[i] ← RC5(rc5counter − 1)   ≻ Write keystream material into nonvolatile memory
11        kstr_end ← i + 1                     ≻ Update kstr_end in nonvolatile memory
12        i ← i + 1
```

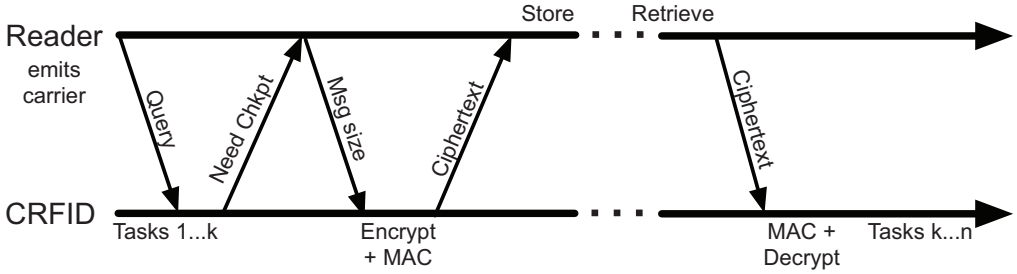


Figure 3: Application-level view of the CCCP protocol. The CRFID sends a request to checkpoint state while in the presence of a reader, and the reader specifies the maximum size of each message. The CRFID then prepares the checkpoint and transmits it in a series of appropriately sized messages. The reader stores the checkpoint data for later retrieval by the CRFID. All messages from the reader to the CRFID also supply power to the CRFID if the latter is within range.

for backscatter modulation on the WISP (Revision 4.0) draws 500 μW of power, far more than is typical of a comparable mechanism on a conventional RFID tag. Alien's Higgs 3, a conventional RFID tag, draws only 15.8 μW of power [2] (total) during operation—an order of magnitude difference that supports an alternative design choice for future CRFIDs.

5.3.1 System Overhead

An application on a CRFID can balance energy consumption against security by choosing one of CCCP's operating modes:

- CCCP/NoSec imposes the least overhead because it does not encrypt data or compute a MAC; it requires no computation and consumes no keystream material. However, CCCP/NoSec imposes a time overhead to receive computational state from a reader at power-up and to transmit new state at checkpoint time.
- CCCP/Auth avoids encryption overhead (like CCCP/NoSec) but requires time, energy, and keystream bits to compute a MAC over the plaintext checkpoint. However, it requires no energy or keystream bits for encryption because it does not encrypt the plaintext checkpoint.
- CCCP/AuthConf offers the most security, since it adds confidentiality to CCCP/Auth, but the extra security comes at the expense of time, energy, and keystream bits. In this mode, CCCP encrypts the computational state before computing a MAC and

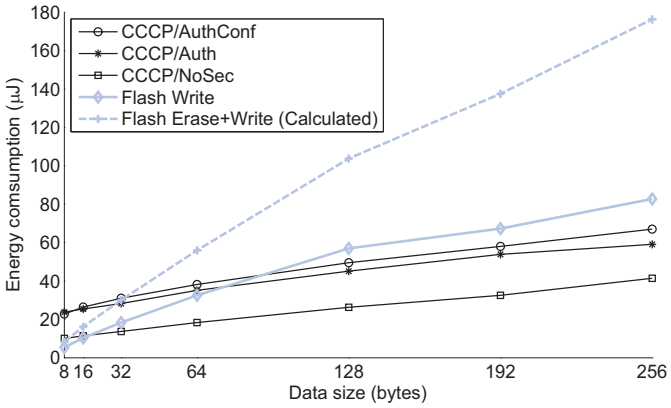


Figure 4: Energy consumption measurements from a WISP (Revision 4.0) prototype for all considered checkpointing strategies. Under our experimental method, we are unable to execute flash writes larger than 256 bytes on current hardware because larger data sizes exhaust the maximum amount of energy available in a single energy lifecycle. The average and maximum percent error of the measurements are 5.85% and 14.08% respectively.

transmitting both. It requires as much keystream material as the size of the state plus a constant amount for authentication.

6 Applications

The outsourced memory introduced by CCCP expands the design space for applications on a computational RFID. This section offers some illustrative example applications.

CRFIDs as low-maintenance sensors. Consider a *cold-chain monitoring* application for pharmaceutical supplies, in which a CRFID carries an attached temperature sensor and stores in flash memory a temperature reading each time it is scanned. To prevent exhaustion of its flash memory, the CRFID periodically computes aggregate statistics on, then discards, stored readings. Some statistical computations (e.g., computation of quartiles) require memory-intensive manipulation of the data set. If the flash memory on the CRFID considerably exceeds the size of RAM, computation of such statistics would require many writes to flash, an energy-intensive operation. An alternative is to use outsourced memory for the computation. (In the case of cold-chain monitoring, maintaining privacy of harvested data with respect to the reader may be unessential, but the *integrity* of the statistical computation is important.)

RFID sensor networks. Recent work [7] describes *RFID sensor networks* (RSNs) that combine RFID reader infrastructure with sensor-equipped computational RFIDs. RSNs do not simply replace traditional sensor networks because of several limitations. First, they require an infrastructure of readers that provide power to sensor nodes. Second, they are constrained by the distances (several meters) at which CRFIDs currently operate. Third, because RFID communication is asymmetric, the nodes of an RSN cannot exchange information with each other except through a more powerful reader. However, there are applications for which short-range networks of batteryless sensors would be appropriate; Yeager et al. offer several examples [34].

Computational RFIDs as smartcards. Some passive RFID tags are capable of executing strong cryptographic primitives. For example, various models of the Mifare DESfire can perform triple-DES or AES, while other RFID devices can compute elliptic-curve and RSA signatures, such as the RF360 introduced by Texas Instruments [32]. The RF360 is designed to allow public-key authentication in RFID-enabled identification documents, such as e-passports.

The RF360 incorporates an MSP430, but also includes a cryptographic co-processor, and is designed to operate at relatively short range as a high-frequency, ISO 14443 device. As we show in this paper, CCCP creates the possibility of a more lightweight device. Such a "CCCP smartcard" has two notable benefits: (1) a CCCP smartcard eliminates the cost of cryptography-specific hardware; and (2) a CCCP smartcard can operate in a mode compatible with EPC Gen 2 and achieve read ranges beyond those of a high-frequency device like the RF360.

Some smartcards are capable of performing biometric authentication—generally fingerprint verification. Match-on-card, i.e., verification of the validity of a fingerprint through computation exclusively within the smartcard, has long stood as a technical challenge. The U.S. National Institute of Standards and Technology (NIST) recently conducted an evaluation of a range of such algorithms in contactless cards [11]. CCCP is a promising tool for expanding the class of radio devices for which match-on-card is feasible. While CCCP does

not follow a strict match-in-device paradigm—given that it outsources data to a reader—it nonetheless provides comparable security assurances.

Trusted computing: outsourcing computation via TPMs. CCCP permits a computational RFID to use external memory via an RFID reader. It can support an even broader design space if we use CCCP instead for secure outsourcing not of memory, but of *computational tasks*.

Trusted platform modules (TPMs) [3, 33] offer support for such outsourcing. A TPM is a hardware device, standard in the CPUs of modern PCs and servers, that can provide a secure attestation to the software configuration of the computing platform on which it operates. Briefly stated, an attestation takes the form of a digital signature on a digest of the software components loaded onto the device. (An attestation does not provide assurance against hardware tampering or subversion of running software.)

A computational RFID can in principle make use of a TPM-enabled reader—or platform communicating with the reader—to gain secure access to a more powerful external computer. The process for such use of a TPM is subtle. The operations of verifying a TPM attestation and creating a secure session are both cryptographic operations that require computationally intensive modular exponentiation. Hence the computational outsourcing process requires CCCP as a bootstrapping mechanism.

7 Related Work

CCCP is closely related to Mementos [26] in that both systems provide checkpointing of program execution on CRFIDs. Whereas Mementos relies purely on flash memory and focuses on finding optimal checkpoint frequencies via static and dynamic analysis, CCCP relies primarily on untrusted remote storage via radio and focuses on low-power cryptographic protections to ensure that remotely stored data is as secure as if it were stored locally.

Several systems share CCCP's goal of exploiting properties of RFID systems to enhance security and privacy. For instance, Shamir's SQUASH hash algorithm [31] exploits the underutilized radio link between a tag and a reader to reduce the amount of cryptographic computation necessary on a tag. While number-theoretic hash functions typically require significant computational resources for modular arithmetic, the SQUASH function eliminates costly modular reductions and produces large (unreduced) hash outputs that a tag can send directly to a reader. Tags can thus use the SQUASH function to engage in secure challenge-response protocols with minimal computational resources on the tag. The scheme is provably as one-way as Rabin encryption. Like SQUASH, CCCP exploits the relatively low cost of radio communication between a tag and a reader to increase security. While SQUASH increases radio communication to reduce computation, CCCP increases radio communication to reduce writes to flash memory.

CCCP uses cryptographic techniques from past work on secure file systems and secure content distribution. CFS [5], the SFS read-only file system [16], and Plutus [22] investigated how to provide secure storage layered on various degrees of untrusted infrastructure. The key generation techniques in secure file systems help CCCP to precompute keystream materials during power seasons. While scalability and throughput are the main challenges in such file systems, CCCP primarily addresses energy and memory constraints. The semantics of CCCP storage are similar to the semantics of secure file systems. None of the systems explicitly and directly prevent denial of service. Storing information on untrusted RFID readers trades off the gain in storage capacity and energy conservation versus the risk of losing data due to compromise or destruction of the external storage. To mitigate the risk against denial of service, CCCP could choose to replicate data as do secure file systems.

CCCP shares some goals with power-aware encryption systems such as that proposed by Chandramouli et al. [10]. Both systems are designed to consume little energy while offering the security of well-known cryptographic primitives and both are motivated by a study of power profiling results, but they have different goals. Chandramouli et al. focus on deriving an energy consumption model and establishing a relationship between energy consumption and security, and they offer an encryption scheme that might allow CCCP to consume less energy during its precomputation of keystream bits. However, CCCP's opportunistic precomputation occurs during periods of abundant energy, when the choice of encryption scheme is not of the utmost importance. CCCP's precomputation allows it to use time- and energy-efficient XOR operations at checkpoint time, when energy is low; an alternative encryption scheme would have to save time or energy over simple XOR operations to be useful when energy consumption matters.

CCCP shares a number of properties with systems built for sensor networks. Storage-centric sensor networks [12, 24] have focused on reducing radio communication and increasing writes to flash memory to conserve energy. One of our motivating observations is that this relationship is inverted in the CRFID model: CCCP reduces writes to flash memory in favor of increasing radio communication. Performing cryptography is hard on both a CRFID and its elder cousin the sensor mote. Previous systems, such as SPINS [25] and TinySec [23] for sensor networks, have faced design choices similar to CCCP's. SPINS and TinySec use RC5 because of its small code size and efficiency, but the battery-powered

platform underlying these systems differs in fundamental ways from batteryless computational RFIDs. For a side-by-side comparison of such embedded systems, see Table 1 of Chae et al. [9].

CCCP provides secure storage for CRFIDs, and CRFIDs are closely related to existing passively powered RFID tags conforming to the EPC Gen 2 standard [13]. At times the RFID and sensor world fuse together. Buettner et al. [7] propose *RFID sensor networks* (RSNs) as a replacement for wireless sensor networks in applications where batteries are inconvenient, and the authors describe RSNs built on WISP CRFIDs. However, the RSN work does not consider remote storage options for CRFIDs.

8 Future Work

Our future work includes enhancements to the CCCP protocol. Most pressingly, the protocol currently suffers from a potential atomicity problem. In CHECKPOINT (Algorithm 1), *chkpt_counter* is updated before the checkpointed state is transmitted, so that even if the transmission fails, *chkpt_counter* will point to unused keystream material the next time CHECKPOINT runs. However, if CHECKPOINT updates the offset but terminates before transmission succeeds, then the next RESUME operation will see a value of *chkpt_counter* from which its normal backtracking operation will not find the correct keystream material. CCCP cannot currently recover from such a mismatch.

An unacceptable solution is for CHECKPOINT to update *chkpt_counter after* a successful transmission; such a strategy opens the possibility that, if power loss occurred between the transmission and the counter update, CCCP would reuse keystream material. A more reasonable solution (which we have not implemented) is to use a separate *commit bit* that is set in nonvolatile memory after both the *chkpt_counter* update and the transmission; this solution avoids both problems mentioned above. Minimizing the energy cost of maintaining a commit bit is an opportunity for hardware optimization.

A number of implementation enhancements are also future work. For instance, shortfalls in over-the-air RFID protocols and a lack of drivers on the WISP make the restore procedure unnecessarily complicated and difficult to implement. We also plan to extend the borders of CCCP from checkpointing towards long-term storage as described in Section 3.4.5. Key management makes long-term storage more challenging than checkpointing. Another area for further investigation is modifying the checkpoint function to operate at lower voltages. Writing the counter value to flash memory restricts checkpoints to periods where the available energy can support at least one write to flash memory. Our future work seeks to circumvent these minimum voltages in order to accomplish secure remote storage for CRFIDs whenever their processors have sufficient energy to compute. Finally, for simplicity, CCCP's communication protocol currently addresses only the scenario in which a single tag communicates with a single reader. We plan to discard that simplifying assumption during further testing in multi-reader infrastructures.

9 Conclusion

CRFIDs enable pervasive computing in places where batteries are difficult to maintain. However, the high energy necessary to erase and write to flash memory makes storage difficult without a constant energy source. CCCP extends Mementos [26] by exploiting the backscatter transmission common on passive RFID systems to remotely store checkpoints on an untrusted RFID reader infrastructure. CCCP protects data with UHF-based MACs, opportunistic precomputation of keystream material for symmetric cryptography, and hole punching to store a counter used to enforce data freshness. Our measurements of a prototype implementation of CCCP on the WISP tag shows that radio-based, remote checkpoints require less energy than local, flash-based checkpoints—despite the overhead of the cryptography to restore the security semantics of local, trusted storage. CCCP gives a CRFID increased storage capacity at low energy cost and enables long-running computations to make progress despite continual power interruptions that destroy the contents of RAM. Moreover, the abstraction provided by CCCP allows application developers to focus on computation rather than space, energy, and security management. Flash memory generally requires a coarse-grained, high-power erase operation before writing a new value. Our hole punching technique allows CCCP to partially reuse unerased flash memory, thus reducing the frequency with which flash memory must be erased.

10 Acknowledgments

We thank Robert Jackson for guidance on RF power consumption; Berk Sunar and Christof Paar for advice on UHF-based MACs; Mike Todd for help with circuit-level aspects of flash memory; Mankin Yuen for assisting with measurements; and John Brattin for identifying algorithmic flaws. We also thank the members of the SPQR group from UMass Amherst CS and ECE for reviewing early drafts of this work. This research was supported by NSF grants CNS-0520729, CNS-0627529, and the NSF REU program. This research is supported in part by UMass through the CVIP Technology Development Fund. This material is based upon work supported by

the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

References

[1] AHSON, S. A., AND ILYAS, M., Eds. *RFID Handbook: Applications, Technology, Security, and Privacy*. CRC Press, 2008.

[2] ALIEN TECHNOLOGY. Product Overview: Higgs-3 EPC Class 1 Gen 2 RFID Tag IC, July 2008.

[3] BERGER, S., CÁCERES, R., GOLDMAN, K. A., PEREZ, R., SAILER, R., AND VAN DOORN, L. vTPM: virtualizing the trusted platform module. In *Proceedings of the 15th USENIX Security Symposium* (2006), USENIX Association.

[4] BLACK, J., HALEVI, S., KRAWCZYK, H., KROVETZ, T., AND ROGAWAY, P. UMAC: Fast and secure message authentication. In *CRYPTO* (1999), Springer-Verlag, pp. 216–233.

[5] BLAZE, M. A cryptographic file system for UNIX. In *1st ACM Conference on Communications and Computing Security* (November 1993), pp. 9–16.

[6] BRASSARD, G. On computationally secure authentication tags requiring short secret shared keys. In *CRYPTO* (1982), pp. 79–86.

[7] BUETTNER, M., GREENSTEIN, B., SAMPLE, A., SMITH, J. R., AND WETHERALL, D. Revisiting smart dust with RFID sensor networks. In *Proceedings of the 7th ACM Workshop on Hot Topics in Networks (HotNets-VII)* (October 2008).

[8] CARTER, L., AND WEGMAN, M. Universal hash functions. In *Journal of Computer and System Sciences* (1979), Elsevier, pp. 143–154.

[9] CHAE, H.-J., YEAGER, D. J., SMITH, J. R., AND FU, K. Maximalist cryptography and computation on the WISP UHF RFID tag. In *Proceedings of the Conference on RFID Security* (July 2007).

[10] CHANDRAMOULI, R., BAPATLA, S., SUBBALAKSHMI, K. P., AND UMA, R. N. Battery power-aware encryption. *ACM Trans. Inf. Syst. Secur. 9*, 2 (2006), 162–180.

[11] COOPER, D., DANG, H., LEE, P., MACGREGOR, W., AND MEHTA, K. *Secure Biometric Match-on-Card Feasibility Report*, 2007.

[12] DIAO, Y., GANESAN, D., MATHUR, G., AND SHENOY, P. Rethinking data management for storage-centric sensor networks. In *Proceedings of the Third Biennial Conference on Innovative Data Systems Research (CIDR)* (January 2007).

[13] EPCGLOBAL. EPC Radio-Frequency Identity Protocols, Class-1 Generation-2 UHF RFID. http://www.epcglobalinc.org/standards/uhfc1g2/, 2008.

[14] ETZEL, M., PATEL, S., AND RAMZAN, Z. Square hash: Fast message authentication via optimized universal hash functions. In *In Proc. CRYPTO 99, Lecture Notes in Computer Science* (1999), Springer-Verlag, pp. 234–251.

[15] FONSECA, R., DUTTA, P., LEVIS, P., AND STOICA, I. Quanto: Tracking energy in networked embedded systems. In *8th USENIX Symposium of Operating Systems Design and Implementation (OSDI'08)* (2008), pp. 323–328.

[16] FU, K., KAASHOEK, M. F., AND MAZIÈRES, D. Fast and secure distributed read-only file system. *ACM Transactions on Computer Systems 20*, 1 (February 2002), 1–24.

[17] GUTTERMAN, Z., PINKAS, B., AND REINMAN, T. Analysis of the Linux random number generator. In *IEEE Symposium on Security and Privacy* (2006), IEEE Computer Society, pp. 371–385.

[18] HADDAD, S., CHANG, C., SWAMINATHAN, B., AND LIEN, J. Degradations due to hole trapping in flash memory cells. In *Electron Device Letters* (March 1989), IEEE, pp. 117–119.

[19] HALPERIN, D., HEYDT-BENJAMIN, T. S., RANSFORD, B., CLARK, S. S., DEFEND, B., MORGAN, W., FU, K., KOHNO, T., AND MAISEL, W. H. Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In *IEEE Symposium on Security and Privacy* (May 2008), IEEE Computer Society, pp. 129–142.

[20] HOMER. *Odyssey*, vol. XI. ca. 750 B.C.

[21] JUELS, A. RFID security and privacy: A research survey. *IEEE Journal on Selected Areas in Communications 24*, 2 (February 2006), 381–394.

[22] KALLAHALLA, M., RIEDEL, E., SWAMINATHAN, R., WANG, Q., AND FU, K. Plutus: Scalable secure file sharing on untrusted storage. In *Proc. USENIX Conference on File and Storage Technologies* (San Francisco, CA, December 2003).

[23] KARLOF, C., SASTRY, N., AND WAGNER, D. TinySec: A link layer security architecture for wireless sensor networks. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys)* (November 2004).

[24] MATHUR, G., DESNOYERS, P., GANESAN, D., AND SHENOY, P. CAPSULE: An energy-optimized object storage system for memory-constrained sensor devices. In *Proceedings of the Fourth ACM Conference on Embedded Networked Sensor Systems (SenSys)* (November 2006).

[25] PERRIG, A., SZEWCZYK, R., WEN, V., CULLER, D., AND TYGAR, J. D. SPINS: Security protocols for sensor networks. *Wireless Networks 8*, 5 (Sept. 2002), 521–534.

[26] RANSFORD, B., CLARK, S., SALAJEGHEH, M., AND FU, K. Getting things done on computational RFIDs with energy-aware checkpointing and voltage-aware scheduling. In *Proceedings of USENIX Workshop on Power Aware Computing and Systems (HotPower)* (December 2008).

[27] RANSFORD, B., AND FU, K. Mementos: A secure platform for batteryless pervasive computing, August 2008. USENIX Security Works-in-Progress Presentation.

[28] RIVEST, R. L. The RC5 encryption algorithm. *Dr Dobb's Journal—Software Tools for the Professional Programmer 20*, 1 (1995), 146–149.

[29] SAMPLE, A. P., YEAGER, D. J., POWLEDGE, P. S., MAMISHEV, A. V., AND SMITH, J. R. Design of an RFID-based battery-free programmable sensing platform. In *IEEE Transactions on Instrumentation and Measurement* (2008).

[30] SCHNEIER, B. *Applied cryptography (2nd ed.): protocols, algorithms, and source code in C*. John Wiley & Sons, Inc., New York, NY, USA, 1995.

[31] SHAMIR, A. SQUASH—a new MAC with provable security properties for highly constrained devices such as RFID tags. In *Proceedings of the 15th International Workshop on Fast Software Encryption (FSE)* (2008), Springer-Verlag, pp. 144–157.

[32] TEXAS INSTRUMENTS INCORPORATED. http://www.ti.com/rfid/shtml/news-releases-11-12-07.shtml.

[33] Trusted computing group. http://www.trustedcomputinggroup.org.

[34] YEAGER, D., POWLEDGE, P., PRASAD, R., WETHERALL, D., AND SMITH, J. Wirelessly-Charged UHF Tags for Sensor Data Collection. In *IEEE International Conference on RFID* (2008), pp. 320–327.

Jamming-resistant Broadcast Communication without Shared Keys

Christina Pöpper
System Security Group
ETH Zurich, Switzerland
poepperc@inf.ethz.ch

Mario Strasser
Communication Systems Group
ETH Zurich, Switzerland
strasser@tik.ee.ethz.ch

Srdjan Čapkun
System Security Group
ETH Zurich, Switzerland
capkuns@inf.ethz.ch

Abstract

Jamming-resistant broadcast communication is crucial for safety-critical applications such as emergency alert broadcasts or the dissemination of navigation signals in adversarial settings. These applications share the need for guaranteed authenticity and availability of messages which are broadcasted by base stations to a large and unknown number of (potentially untrusted) receivers. Common techniques to counter jamming attacks such as Direct-Sequence Spread Spectrum (DSSS) and Frequency Hopping are based on secrets that need to be shared between the sender and the receivers before the start of the communication. However, broadcast anti-jamming communication that relies on either secret pairwise or group keys is likely to be subject to scalability and key-setup problems or provides weak jamming-resistance, respectively. In this work, we therefore propose a solution called Uncoordinated DSSS (UDSSS) that enables spread-spectrum anti-jamming broadcast communication without the requirement of shared secrets. It is applicable to broadcast scenarios in which receivers hold an authentic public key of the sender but do not share a secret key with it. UDSSS can handle an unlimited amount of receivers while being secure against malicious receivers. We analyze the security and latency of UDSSS and complete our work with an experimental evaluation on a prototype implementation.

1 Introduction

Due to the shared use of the communication medium, wireless radio communication is not only vulnerable to traditional attacks such as eavesdropping and message synthesis but also to active jamming attacks [2, 20]. In a signal jamming attack, the attacker emits a jamming signal while the legitimate transmission is taking place, thus achieving a denial-of-service (DoS) by blocking, modifying, annihilating, or overwriting the original signal. Well-known, effective countermeasures against signal jamming attacks are spread-spectrum techniques, in particular Direct-Sequence Spread Spectrum (DSSS) and Frequency Hopping Spread Spectrum (FHSS) [23]. For these techniques to work, the receivers are required to share secret keys with the sender prior to their anti-jamming communication; these keys enable them to derive identical spreading codes or hopping sequences. Shared secrets are also the basis of proposed anti-jamming broadcast schemes [6, 8].

The requirement of pre-shared secret keys, however, imposes limits on the use of common spread-spectrum techniques for anti-jamming communication in scenarios where such secret keys cannot be pre-shared (but which instead rely on, e.g., public-key certificates). This problem (i.e., the lack of techniques for jamming resistance without shared secret keys) was recently observed in [4] and [24] in the context of pairwise communication.

In this work, we focus on a related but different problem for broadcast communication: *How to enable robust anti-jamming broadcast without shared secret keys*? Typical broadcast applications share the need for authenticity and availability of messages that are transmitted by base stations (senders) to a large, unknown number of potentially untrusted (malicious or selfish) receivers. In such settings, a sender communicates to a dynamic set of *trusted* receivers (i.e., the nodes are honest but may be unknown to the sender due to receiver dynamics) or to *untrusted* receivers (which might be interested in obtaining the information themselves but depriving others of it). In both cases, basing the anti-jamming communication on pre-shared keys is not an option because (honest) nodes join the setting *after* the key deployment or because malicious nodes may misuse shared keys for jamming. We can best illustrate this by an example:

A governmental authority needs to inform the public about the threat of an imminent attack. For disseminating information about the risk, a message could contain the level of risk, a timestamp, the physical area of risk,

and the signature of the central authority (CA). Note that if DSSS was used with a (public) spreading code that is known to the attacker or if no spreading was used at all for the transmission, the attacker could easily disrupt the transmission of the message by jamming, thus blocking the propagation of the warning within her transmission radius. The information transferred in this setting is not secret, hence eavesdropping is not considered a risk. What is crucial is the dissemination (broadcast) of *authentic* information to as many receivers as possible within a reasonable timeframe (seconds to few minutes).

As a solution to the described problem, we propose a scheme called *Uncoordinated DSSS* (UDSSS) that enables authentic spread-spectrum anti-jamming broadcast without the requirement of shared secrets. UDSSS follows a similar approach as DSSS, it differs, however, in the following aspect: the spreading code is not predefined but chosen by the sender randomly out of a set of publicly available codes. Since no receiver can predict the choice of the sender, UDSSS prevents dishonest receivers from interfering with the communication (to other receivers) while it enables them to obtain the information themselves. After a certain time, every receiver will succeed in identifying the correct spreading code and its synchronization, thus despreading the signal. The required despreading time depends on the coding strategy, the size of the spreading code set, and on the receivers' processing capabilities; we analyze this in detail. Although UDSSS is inherently less efficient than DSSS, it enables broadcast anti-jamming communication in scenarios in which DSSS cannot be used. Besides the example described above, an important application of UDSSS is the jamming-resilient dissemination of navigation signals. As we will show in Section 7, UDSSS enables not only anti-jamming localization for broadcast navigation systems (GPS or similar systems), but it also inherently protects them against a wide range of location-spoofing attacks. We will also show that UDSSS can achieve the same performance as DSSS in the absence of jamming.

In summary, the main contributions of this work are:

- We identify anti-jamming broadcast without shared keys as a relevant problem and we show that it can be addressed using uncoordinated spread-spectrum techniques.
- We propose a scheme called *Uncoordinated DSSS* that supports broadcast anti-jamming communication without shared keys and enables communication in scenarios in which DSSS cannot be used.
- We analyze the performance of UDSSS. We show that a performance comparable to DSSS can be achieved in the absence of jamming and that the expected time for a message transmission to ten receivers takes less than 30 s on state-of-the-art systems under high jamming-probabilities.
- We demonstrate the feasibility of UDSSS by a prototype implementation on a software-defined radio platform [10]; the reception of a typical message takes well below 20 s for 21 dB processing gain on this system. We note that this time can further be significantly reduced on a purpose-built platform (e.g., like the ones used for GPS receivers).

The remainder of the paper is organized as follows: We give background information on DSSS in Section 2 and describe the system and attacker models in Section 3. In Section 4, we present our UDSSS scheme. We analyze its security in Section 5 and its performance in Section 6, including the presentation of our implementation results. In Section 7, we discuss possible applications of UDSSS. Finally, in Section 8, we describe related work and we conclude our paper in Section 9.

2 Background: DSSS

In DSSS, the data signal is modulated with a continuous, pre-defined spreading signal of a higher frequency, also called the *chipping sequence*. During the modulation, the data signal gets spread in the frequency domain and thus becomes resistant against (narrow-band) interference. The resulting signal is modulated (e.g., using phase-shift keying) and – given a sufficiently high frequency of the spreading signal – becomes hidden in the noise of the wireless channel. The processing gain of the communication system (indicating the ratio by which interference can be suppressed relative to the original signal) defines the required length N of the DSSS spreading code, determining the spreading signal. More precisely, given a certain data bit time T_b and a target processing gain defined as $10 \log_{10} \frac{T_b}{T_c}$ in decibel (dB), we get $N = T_b/T_c$, where T_c is the time of a modulated signal chip (a low signal-to-noise ratio requires $T_c \ll T_b$). A typical processing gain of spread-spectrum systems is between 20 dB and 60 dB and results from a chip length $N \in \{100, \ldots, 10^6\}$.

In anti-jamming applications, the DSSS spreading signal is secret and shared only by the sender and legitimate receivers. This can be achieved by a shared secret key that is used to seed a pseudo-random generator at the sender and the receivers. The generator outputs a (pseudo-random) chipping sequence which is used to spread the message. In order to despread the signal, the receivers apply a symmetric operation and correlate the received signal with a synchronized replica of the spreading code. Except for the secret code, all other communication parameters (modulation, frequency band, etc.) are public. For the discussion of DSSS we assume that the receivers are synchronized to the sender (later, we will show how we remove this assumption in UDSSS). The synchronization includes both bit and chip time syn-

chronization to the sent signal as well as synchronization with respect to the used spreading code, i.e., the receivers know which code to apply at which point in time in order to despread the received signal. We refer to related literature for a comprehensive discussion of efficient synchronization techniques [2, 20, 23].

In more details, for spreading a message M, the sender uses a *spreading sequence* $c_0 = (c_{0,1}, c_{0,2}, \ldots, c_{0,\ell})$ composed of ℓ binary NRZ (non-return to zero) *spreading codes*, $|c_{0,i}| = N$. Typical spreading codes used for DSSS are pseudo-randomly created sequences [23] and codes with well-defined properties such as Walsh-Hadamard [11] or Gold-codes [20]. The sender spreads M by applying code $c_{0,1}$ to the first b bits of M, $c_{0,2}$ to the second b bits and so forth, where b denotes the repetition factor in the use of the spreading codes. By expressing the codes in the time domain, we can define a function $c_0(t) = c_{0,i}[j]$ for $i = \lfloor t/bT_b \rfloor \mod N$ and $j = \lfloor t/T_c \rfloor \mod N$, where T_b (T_c) is the data bit (chip) time. The spreading operation can then be written as $d(t) \cdot c_0(t)$, where $d(t)$ is the data signal that carries the message. The sender modulates and transmits the result.

Upon signal reception, each receiver demodulates the signal and samples it (sampling rate $R_s \geq 2/T_c$). It stores the samples in a cyclic buffer which has the capacity to store samples of several message bits, (i.e., for the duration of $T_s = kT_b$, $k > 1 \in \mathbb{N}$). Then, the receiver despreads the data stored in the buffer by computing $\overline{s} := \sum_{i=0}^{T_b R_s} s[i]c_0(t_i)$ for each data bit, where $s[i]$ denotes the i-th value in the buffer and t_i the time when it was sampled. Finally, the result of the bit integration $\overline{s}$ is used to determine the received bit $\hat{d}_i$. We assume a simple bit decoder that outputs 1 (0) if the integration yields a value greater (lower) than 0 (i.e., $\hat{d}_i = \lceil \overline{s} \rceil$). Finally, after all data bits have been despread, the correctness of the despreading operation is verified by means of the message decoding.

3 System and Attacker Models

3.1 System Model

Our system consists of a sender A and a set of receivers. The goal of the sender is to enable anti-jamming broadcast to the receivers in the presence of communication jamming. We assume that each device is equipped with a radio frontend with transmission and reception capabilities in a corresponding frequency band and that the receivers are computationally capable of efficiently performing (e.g., ECC-based) public-key operations. In addition, each receiver holds an authentic public key of the sender or of the central authority (CA) that can certify the sender's public key. The CA may be off-line at the time of communication.

In our model, P_A denotes the strength of A's signal arriving at a receiver B; P_A depends on the strength of the signal sent by A, on the distance between the sender and the receiver as well as on large- and small-scale fading and interference effects [25,26]. We denote by P_t the minimal required signal strength at the receiver B such that B can successfully decode the signal. In this context, the transmission between A and B in a setting without (active) interference will be successful if $i)$ $P_A \geq P_t$, if $ii)$ A and B use the same spreading code, and if $iii)$ B uses the correct synchronization in its despreading operation (code time and carrier frequency synchronization).

3.2 Attacker Model

We adopt the attacker model from [24] and consider an omnipresent but computationally bounded adversary J with unlimited power supply that is able to eavesdrop and insert messages arbitrarily but can only alter or erase messages by adding her own (energy-limited) signals to the wireless medium; that is, she cannot disable the communication channel by blocking the propagation of signals (e.g., by placing a Faraday cage around a node). The goal of the attacker is to prevent all communication between the sender A and all or some of the receivers. In order to achieve this, the attacker is not restricted to message jamming but can also modify existing or insert new messages. More precisely, the attacker can choose among the following actions:

- She can *jam messages* by transmitting high-power signals that cause the original signal to become unreadable by the receiver. The fraction of the message that the attacker has to interfere with to successfully jam depends on the used coding scheme (e.g., 13% of the message size [16]).
- She can *modify messages* by either flipping single message bits or by entirely overshadowing original messages. In either case, in this attack the messages remain readable by the receiver.
- She can *insert messages* that she generated herself or reuse previously overheard messages. Depending on the signal strength and used spreading codes, the inserted messages might interfere with regular transmissions.

In addition to these types of *attacks*, we follow previous classifications [20] and distinguish different types of *attackers*: static, sweep, random, and reactive jammers. Static, sweep, and random jammers do not sense for ongoing transmissions but jam the channel permanently; they only differ in the regularity of their jamming signals. Reactive jammers initially solely sense for ongoing transmissions and start jamming only after the detection of a message transfer; we express the strength of reactive jammers by their despreading performance $\Lambda_B(N)$,

denoting the number of spreading codes the attacker can apply and check per time unit. Repeater jammers [12] are a subclass of reactive jammers that intercept the signal, low-noise amplify, filter and re-radiate it without requiring or getting knowledge of the used spreading codes. Hybrid jammers are a combination of the above types that jam while searching for message transmissions.

For all attacker types, we assume a finite maximal transmission power and bandwidth. We denote by P_J the maximal signal strength that the attacker is able to achieve at a receiver B; the attacker can split P_J over an arbitrary number of parallel signal transmissions. Given P_A, the strength of A's signal at B, we denote by P_j and P_o the minimal required strength of the attacker's signal at B in order to jam or overshadow a message sent from A to B, respectively, provided that the attacker is aware of the used code sequence and its synchronization. We assume $P_t \leq P_A$ and $P_j < P_o$. We further assume that $P_J < \mu P_t$, where μ denotes the number of possible transmissions, i.e., the attacker is not capable of jamming all possible transmissions in parallel; μ depends on the number of available spreading codes and on the attacker's bit and chip synchronization.

4 Jamming-Resistant Broadcast: UDSSS

In this section, we introduce our UDSSS (Uncoordinated DSSS) scheme. UDSSS is an anti-jamming modulation technique based on the concept of DSSS, however, it does *not* rely on pre-shared spreading sequences. In contrast to anti-jamming DSSS communication, where the spreading sequence is secret and shared exclusively by the communication partners, in UDSSS, a public set C of spreading sequences is used by the sender and the receivers. C is not secret and may be known to the attacker. To transmit a message, the sender randomly selects a spreading sequence from the code set and spreads the message with this sequence. The receivers record the signal on the channel and despread the message by applying sequences from C using a trial-and-error method.

The receivers using UDSSS are not time-synchronized to the sender with respect to the spread signal, i.e., they do not know the message bit or chip synchronization. In order to compensate for this (as well as for message losses due to jamming), the sender sends the message repeatedly and the receivers apply a sliding window approach to synchronize to the transmission. The efficiency of UDSSS is therefore determined $i)$ by the time that the receivers need to find the right spreading code and its synchronization (we will analyze this in detail in Section 6) and $ii)$ by the attacker's jamming success (analyzed in Section 5). Given that, in UDSSS, the receivers need to search through a set of codes and synchronization windows in order to despread the received message, UDSSS is inherently less efficient than DSSS. However, it provides important advantages over DSSS:

- UDSSS enables anti-jamming communication between nodes that are within each others' transmission ranges but do not share a secret, and
- UDSSS supports broadcast anti-jamming communication for dynamic groups of untrusted receivers.

UDSSS requires the receivers to store all chips received and to analyze them retrospectively to find the used spreading code. The time this takes defines the latency of the communication. The performance and jamming-resistance of UDSSS can be increased by using multiple senders (in contrast to DSSS). More precisely, we consider $m \geq 1$ parallel broadcast transmissions of the same message with *different* spreading codes. This can be achieved by one sender transmitting m signals in parallel—each spread with a different spreading code—or by using m separate sending devices.

In what follows, we describe the details of the UDSSS operations at the sender(s) and the receivers and discuss suitable choices of the UDSSS spreading code set.

4.1 UDSSS Transmission

We envisage *one* sending device, but for generality, our description includes one or multiple senders that transmit the same message in parallel on $m \geq 1$ channels using the code sets $C_1, \ldots, C_m$ (not necessarily distinct). For transmitting message M, $|M| \leq \ell$, each sender repeatedly selects a *fresh*, i.e. randomly selected, code sequence $c_s \in C_i$, spreads M using c_s, and transmits the resulting modulated signal. For each transmission a new code sequence is chosen; repeated messages thus get encoded with a different code sequence on each transmission (with high probability). All spreading codes are chosen to be (nearly) orthogonal (strong auto- and low cross-correlations), hence parallel transmissions of multiple senders do not (significantly) interfere with each other; multiple transmissions using the same spreading code and code synchronization can be excluded by agreements between the senders that are, e.g., linked by wires. Each sender repeats the spreading and sending operation either for a well-defined number of iterations (e.g., for emergency alert broadcasting) or continuously (for longer-term applications, e.g. for navigation signals).

Before the UDSSS modulation, each sender applies the following techniques: In order to achieve message authentication, sender A signs the message using its private key SK_A. The sender may also include a timestamp or sequence number in the message in order to achieve replay protection. In order to resist transmission errors, the sender then error-encodes the message before the spreading operation; error-coding makes a message resistant to a certain number of bit errors (e.g., up to 13%

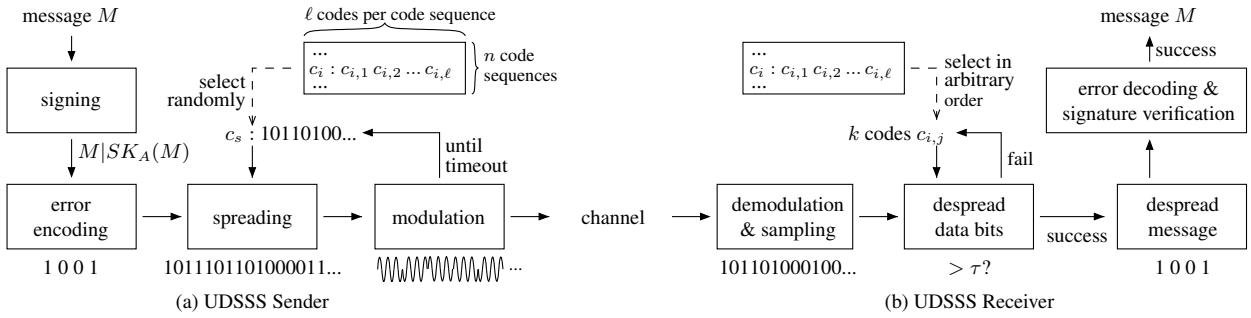


Figure 1: (a) UDSSS transmission. Sender A signs and error-encodes the message M. Then it repeatedly spreads the signed and error-encoded data using a *freshly* selected spreading sequence c_s in each repetition and transmits the modulated signal. (b) UDSSS reception. Receiver B demodulates and samples the radio channel. Then B repeatedly selects a spreading sequence $c_i \in C$, picks k codes $c_{i,j}$ from c_i and tries to despread one data bit (using the integration threshold τ). On success, B despreads the entire message. A failure during the error-encoding check or signature verification restarts the despreading process.

for concatenated Reed-Solomon codes [16]). In combination with bit interleaving, error-encoding increases the resistance of a message to targeted jamming attacks. The entire sending process is displayed in Figure 1a.

There are two reasons why the UDSSS transmission requires message repetitions: i) to enable the receivers that are not synchronized to the beginning of the transmission to receive the message and ii) due to the risk that the attacker guesses the used code sequence and thus jams the transmission (this risk is also present in DSSS anti-jamming systems). UDSSS receivers will therefore not try to decode all received signals but only those signals that are received in the time intervals when the sender is expected to transmit. For this, the sender either needs to have a (public) transmission schedule (and the receivers need to be precisely time-synchronized to the sender) or the sender has to repeat the transmission of each message such that, when the receivers fill their reception buffers, they will receive the message (Fig. 2).

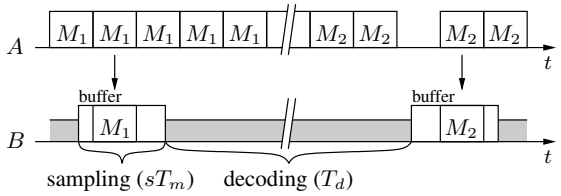


Figure 2: UDSSS message sampling and decoding. A's repeated transmissions ensure that each receiver B can sample an entire message. After the sampling, B decodes the message M_1 contained in the buffer. During the decoding, B disregards all further samples.

4.2 UDSSS Reception

In UDSSS, each receiver samples the radio channel (sampling rate $R_s \geq q/T_c$, $q \geq 2$, sample resolution b_s bits) during the sampling period $T_s = sT_M$, and stores the samples in a buffer; T_M denotes the message transmission time and $s \geq 2$ is the number of messages that can be stored in the buffer; given a continuous message transmission, for $s \geq 2$, the signal stored in the buffer will always contain an entire message. After the buffer has been filled, the receiver will reject all signals arriving to its interface (Figure 2) until the message in the buffer is successfully despread and its authenticity is verified.

After the sampling, the receiver tries to decode the data in the buffer by applying a sliding-window protocol in which the current window is shifted in intervals of T_c/q; a complete run of the despreading operation is denoted as one *decoding*. For this purpose, the receiver chooses k spreading codes $c_{i,j}$ ($1 \leq i \leq n$ and $1 \leq j \leq \ell$) from each code sequence $c_i \in C$ (see Figure 3) and uses them to despread k data bits, as sketched in Figure 1b. We check each spreading sequence on multiple ($k > 1$) data bits in order to compensate for transmission or decoding errors. If, during this process and while applying codes from c_r, the absolute value of a bit integration exceeds a threshold τ, i.e. $\overline{s} := \sum_{i=0}^{T_b R_s} s[i] c_{r,j}(t_i) \geq \tau$, the receiver uses the code sequence c_r for despreading the entire message, now benefiting from the identified chip synchronization. τ can be derived from the cross-correlation properties of the used codes and depends on the code length (see Section 4.3). Depending on the available hardware, the despreading operation can partially be performed in parallel or using a multi-stage solution [20].

The bits resulting from this trial-and-error approach are disinterleaved and verified by means of the error-encoding of the message. The receiver accepts those

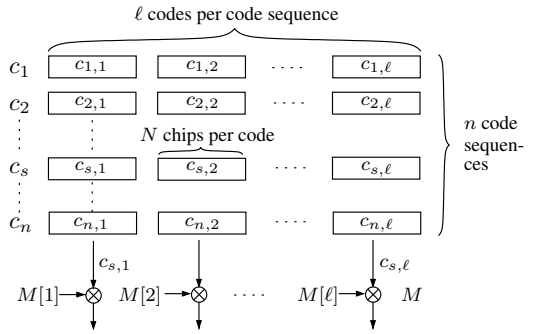


Figure 3: The set C of code sequences. Each sequence $c_i \in C$ is composed of ℓ spreading codes: $c_i = (c_{i,1}, \ldots, c_{i,\ell})$, where $|c_{i,j}| = N$. A message M is then spread using a randomly selected code sequence $c_s \in C$; $M[i]$ denotes the i-th bit of M.

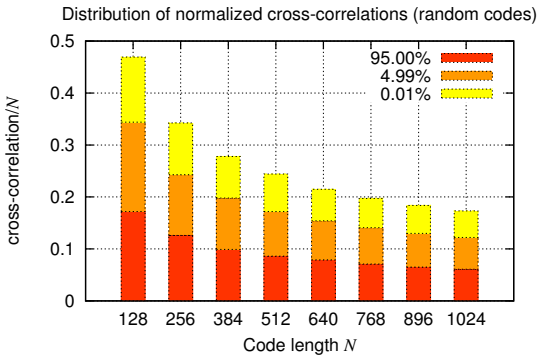


Figure 4: Distribution of cross-correlations for 1000 pseudo-randomly created codes, depending on the code length N. The values are normalized, i.e. divided by N (the peak auto-correlation). The simulations allow to set reasonable limits to the detection threshold τ.

messages that pass the error-encoding check and hands them on to the signature verification. Due to possible message insertions by an attacker, the receiver does not stop analyzing the buffer after having successfully despread a message with valid error-encoding, but continues scanning the buffer using the remaining code sequences (until a despread message also passes the signature verification). Thus, the receiver may detect one or more messages per buffer, coming from the original transmissions or from message insertions by the attacker. In any case, the receiver will only pass those messages to the application layer that contain a valid signature.

4.3 UDSSS Spreading Codes

As a crucial component of UDSSS, we now describe how to generate the UDSSS spreading codes that are used by the sender and receivers. In our description, we refer to *one* code set C, however, the same applies for *each* code set in the case of multiple senders. Figure 3 illustrates the code set. Every spreading code $c_{i,j}$ is used to spread *one* bit of the message M (repetition factor $b = 1$), $\ell = |M|$.

UDSSS requires the use of balanced spreading codes that have good auto- and cross-correlation properties; good auto-correlation properties are needed for precise synchronization at the receivers and low cross-correlation properties have the effect that transmissions with different spreading codes do not interfere with each other. We thus exclude the following codes that are typically used in DSSS systems: codes of insufficient length (e.g., Barker codes), codes with large cross-correlation properties (e.g., Walsh-Hadamard), and unbalanced codes resulting in high auto-correlation values (e.g., optical orthogonal codes [7]). Codes for UDSSS that satisfy the above properties are shift-register sequences, in particular *Gold-* and *Kasami-codes*[1] [17], and *pseudo-random sequences* [22].

Due to their straight-forward generation, we focus on pseudo-random codes in the following. A specific code set C is then given by a (public) seed, used as input to a well-defined pseudo-random number generator. Given a sufficiently large code length N, pseudo-random codes have good auto- and cross-correlation properties[2]. Figure 4 displays the cross-correlation values of pseudo-random codes and confirms the desired property; for a more comprehensive analysis of the properties of pseudo-random sequences we refer to [22]. Consequently, the attacker has to use the correct code sequence $c_s \in C$ in order to interfere with a transmission; using a spreading sequence $c' \neq c_s$, $c' \in C$ will not have a relevant impact on the transmission. We can calculate reasonable limits of the parameter ε that specifies the quality of the correlations. Our simulations suggest that, e.g., for random codes of length $N = 512$ (27 dB), $\varepsilon \lessapprox 150$ (Figure 4). This enables us to set τ used as integration threshold by the receiver: $\tau = a\varepsilon$, $a \in \mathbb{R} \geq 1$. We refer to Section 6.4 for details on the parameter choices.

Furthermore, the probability that any two random codes of length N from a set of $n\ell$ codes agree is approx. $1 - e^{-(n\ell)^2/2^{(N+1)}}$ (cf. birthday paradox). For typical values of N, n, and ℓ (i.e., $N \geq 64$ and $n\ell \leq 2^{20}$) this probability is negligible. Hence, each code sequence c_i is uniquely identified by any of its codes $c_{i,j}$. While this is beneficial for the legitimate receivers, the attacker will likewise know the entire code sequence if she can successfully identify the code that was used for spreading any particular message bit and might thus be able to jam the remainder of the message. This will be taken into account for the analysis of the attacker's decoding strength (Section 5.2). Section 6.5 will further display the impact of n and N on the system performance.

5 Security Evaluation of UDSSS

In this section, we analyze the points of attack on UDSSS communication and, for various attacker types, derive the probability that a message is jammed during its transmission. As we will show, UDSSS provides resistance even to reactive attackers, a very strong type of attacker.

5.1 Jamming Attacks on UDSSS

An attacker has the following options for performing a code-based jamming attack on UDSSS communication: i) she can guess the spreading code and try to jam the signal using this code, ii) she can repeat the recorded signal, thus trying to create a collision with the original transmission, and iii) she can try to find the code by despreading (part of) the spread signal and then use the identified spreading sequence for jamming the rest of the message *during* its transmission. In the first case, the attacker's jamming signal is independent of the transmission she is trying to jam (representing a static, sweep, or random attacker); in the latter two cases, the attacker is reactive and bases her jamming signal on the detection (and analysis) of the spread signal. In the following we refer to reactive jammers that simply repeat the recorded signal as *repeater jammers* and to reactive jammers that aim at finding the used spreading code as *decoding jammers*. A hybrid jammer can combine non-reactive and reactive actions.

For non-reactive (static, sweep or random) attackers (case i), the attacker's success probability depends on the number of codes that she chooses from for composing her jamming signal and on the accuracy of her synchronization to the spread signal. Although (U)DSSS signals are usually hidden in noise, they can be detected by means of energy detectors or by their modulation-specific characteristics [9,20]. Depending on the strength of the attacker and the processing gain achieved by the modulation, the attacker might therefore be able to recover a message transmission and its chip synchronization without having to decode a message; however the attacker still needs to guess the used spreading code in order to jam the signal. In all cases, the jamming success probability of a non-reactive attacker depends on the number of codes in the code set; this probability is further decreased if the attacker cannot detect the chip synchronization (Sec. 5.2).

The purpose of using a different spreading code for each message bit ($b = 1$) is to prevent successful replay attacks from repeater jammers [12] (case ii). Due to the low auto-correlation properties of the codes, the attacker's repeated signal would have to arrive at the receiver within one chip length T_c in order to affect the transmission; this requires the attacker to have an (almost) zero processing delay (e.g., for signal inversion) and to be positioned very close to the signal's path of travel (e.g., within a typical $T_c = 10\,ns$, the signal travels less than $3\,m$). More details are provided in Sec. 5.2.

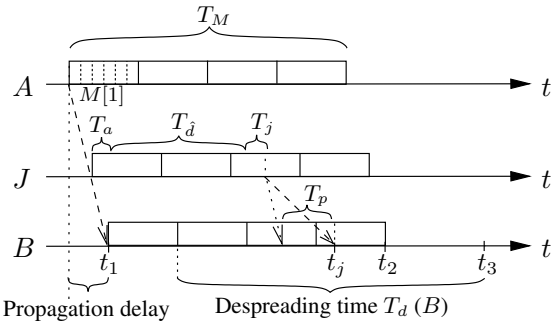


Figure 5: UDSSS attack scenario for reactive (decoding) jammers. Sender A sends message M with transmission time T_M. Receiver B and a reactive jammer J start to despread the (same) signal samples after having recorded the first chips. J's jamming attack may only succeed if $t_j < t_2$, i.e., if the attacker succeeds to compose and send its jamming signal such that it reaches B before B has received the entire message M; otherwise the jamming fails and B will despread the message at time t_3. The main advantage for the receiver over the attacker comes from the fact that the attacker only has very short time ($< T_M$) to despread the message, whereas the receiver can despread the message long after having recorded it (within the latency that the application can tolerate).

Although decoding-based attacks (case iii) are considered infeasible in DSSS, the probability of such an attack is non-negligible for UDSSS communication due to the restricted number of possible spreading codes. Figure 5 displays the attack scenario for decoding attackers. A decoding attacker needs time to acquire the signal, to detect the spreading code used by the sender, and to exploit this knowledge to compose and transmit the jamming signal. Her reaction time is limited by the message transmission time (T_M) and the fraction that needs to be jammed ($T_{\overline{M}}$). More precisely, the attacker's response time (with effect at the receiver) is composed of:

- T_a time for signal acquisition (min. number of chips)
- $T_{\hat{d}}$ expected time for detecting the spreading code
- T_j time for jamming signal generation & transmission
- T_p propagation time difference via the attacker (see Figure 6).

T_a and T_j are mainly determined by the attacker's device, $T_{\hat{d}}$ by her computational capabilities (Figure 7), and T_p is given by the attacker's position relative to the sender and receiver. A reactive attacker can be successful with her jamming attack only if $T_a + T_{\hat{d}} + T_t + T_p < T_M - T_{\overline{M}}$.

For this reason, the success probability of a decoding

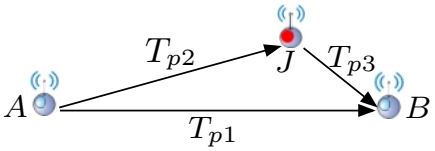


Figure 6: Propagation delay of the jamming signal (for reactive jammers). The displayed times represent propagation delays. $T_p = T_{p2} + T_{p3} - T_{p1}$ is the time that the attacker's signal will be delayed at the receiver B due to propagation delay; $T_p = 0$ if J is positioned on the signal path between A and B.

attacker depends on the time that she needs to identify the used spreading code and its synchronization with respect to the received signal. The code set must limit the search space for the receiver (smaller C is better), while it must still be sufficiently large to prevent the attacker from guessing or systematically finding an effective jamming signal within the message transmission time T_M. Although this might appear as a strong gain in favor of a well-equipped attacker, we stress that the time that the attacker has to find the missing spreading code and its synchronization is small (i.e., limited by T_M, in the order of hundred μs for small messages) while the time for the receiver to despread the recorded message is long (only limited by the application requirements, $\mathcal{O}(s)$). In Section 6, we study how the size of the code set impacts the communication performance of UDSSS.

5.2 Jamming Performance of the Attacker

We now derive the jamming probability for different types of attackers. We use the maximal signal strength P_J that the attacker is able to achieve at the receiver if she transmits with maximal transmission power (Sec. 3.2). Since P_J can be distributed over an arbitrary number of simultaneously transmitted signals, the attacker is allowed to freely choose how much of this power she will use to insert, jam, or overshadow messages as long as the overall signal strength received at B does not exceed P_J. Consequently, given the minimal required signal strength at B such that a message is successfully received (P_t), jammed (P_j), or overshadowed (P_o) (Sec. 3.2), we can derive $n_i := \lfloor \frac{P_J}{P_t} \rfloor$, $n_j := \lfloor \frac{P_J}{P_j} \rfloor$, and $n_o := \lfloor \frac{P_J}{P_o} \rfloor$ as upper bounds for the number of messages that the attacker can insert, jam, and overshadow in parallel.

Static, Sweep, and Random Jamming

We now consider an attacker that tries to guess the used spreading sequence. Let $T_{\overline{M}}$ be the minimum jamming period during which the attacker has to interfere with the transmission of a message M such that it cannot be decoded by the receiver. The length of this period depends on the used coding scheme: the more bit errors it can tolerate, the longer is $T_{\overline{M}}$. We next compute the probability p_j that a message is jammed for static, sweep, and random jammers (Section 3.2). Sweep and random jammers switch their jamming signal (i.e., the set of code sequences $C_j \subseteq C$ that is jammed) after a duration of $T_{\overline{M}}$ whereas static jammers use the same signal for a time $t \gg T_M$. Moreover, sweep jammers do not reuse a code sequence until all sequences from C have been used once, whereas random jammers always choose the set C_j at random and might thus select the same code sequences more than once in subsequent jamming attempts. For both the sweep and the random jammer, the number of jamming attempts per message is $T_M/T_{\overline{M}}$. Hence, the probability that a message is successfully jammed by a static jammer is $p_j(n_j) \leq \frac{n_j}{n|M|N}$; for sweep jammers the jamming probability is $p_j(n_j) \leq \min\{\frac{n_j}{n|M|N}\frac{T_M}{T_{\overline{M}}}, 1\}$, and for random jammers it is $p_j(n_j) \leq 1 - (1 - \frac{n_j}{n|M|N})^{T_M/T_{\overline{M}}}$. Note that the attacker has to hit both the right code sequence (out of n sequences) and chip synchronization ($N|M|$).

Reactive and Hybrid Jamming

A reactive decoding jammer tries to find the sender's spreading sequence by performing a search over C. When successful, the attacker knows both the sender's spreading sequence c_s as well as its synchronization and uses this knowledge to jam the remainder of the message M. Throughout this analysis, we make the (worst case) assumption that successfully decoding a single bit of M reveals to the attacker the code sequence that the sender used to spread M (Sec. 4.3). The attacker's ability to jam a message is thus determined by the time that the attacker needs to identify the sender's code sequence and by the time that she then has left to (partially) jam the same message. Let $\Lambda_J(N)$ denote the number of bits that the attacker can despread per second (possibly benefiting from hardware parallelization). The number of code sequences that the attacker is able to verify during the transmission of M such that she detects M's spreading sequence early enough to be able to successfully jam the message is then $\leq (T_M - T_{\overline{M}})\Lambda_J(N)$. Thus, the probability that a message transmission is detected and jammed is

$$p_j(n_j) \leq \min\left\{\frac{(T_M - T_{\overline{M}})\Lambda_J(N)}{n|M|N}, 1\right\}.$$

The despreading performance of a decoding (reactive) attacker is exemplified in Figure 7; in Section 6, we will compare it to the receiver's message decoding performance. Figure 7 shows the expected time that a decoding

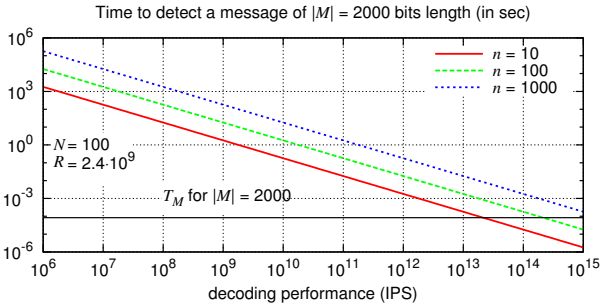


Figure 7: Message detection performance of a decoding attacker as a function of her computing power. This plot depicts the decoding capabilities of a perfect decoding jammer that is able to identify the used spreading code after decoding a single bit (i.e., $k = 1$). The effective impact of the attacker's computing power on the jamming resistance of a message depends on the message transmission time T_M. For a given code set size of $n = 10$ code sequences, in this example, the attacker can block a message of $|M| = 2000$ bits if her computing power exceeds approx. $2 \cdot 10^{13}$ IPS ($2 \cdot 10^{15}$ IPS for $n = 1000$).

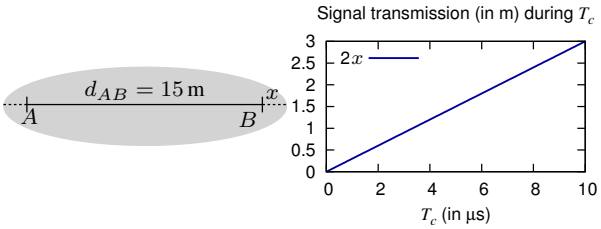
attacker (using the receivers' trial-and-error approach) will need to identify the used spreading code sequence; hardware parallelization in the decoding operation can be mapped to a higher decoding performance. The attacker can only be successful if her time to identify the right code sequence is shorter than the message transmission time (intersection with T_M). We point out that even if the attacker uses more elaborate correlation or deconvolution algorithms, her decoding strength can still be expressed by the expected number of bit decodings per second that her algorithm achieves.

Hybrid jammers are a combination of non-reactive and reactive jammers: while searching for the right spreading code, they simultaneously emit a jamming signal. For the most powerful hybrid jammer type, the reactive-sweep jammer [24], the probability that a message is successfully jammed is

$$p_j(n_j) \leq \frac{\eta}{n|M|N} + \left(1 - \frac{\eta}{n|M|N}\right) \min\left\{\frac{(T_M - T_{\overline{M}})\Lambda_J(N)}{(n-\eta)|M|N}, 1\right\},$$

where $\eta = \min\{n_j T_M / T_{\overline{M}}, n\}$.

Message Overshadowing: Following the above analysis we can also derive the probability that the transmission of a message is overshadowed by the attacker by substituting n_j with n_o in the above expressions for p_j.



(a) (b)

Figure 8: Position of repeater jammers. (a) Area within which a repeater jammer needs to be located in order to successfully jam the communication from A to B (based on signal transmission times). The ellipse is defined by major diameter $d_{AB} + 2x$ and minor diameter $2\sqrt{x^2 + x d_{AB}}$ around A and B; $2x$ is the distance that the signal travels during the chip time T_c ($2x = 3$ m for $T_c = 10^{-8}$ s). (b) Transmission delay ($2x$) of the signal path via the attacker. The smaller T_c, the smaller the area will be in which a repeater jammer needs to be located in order to jam successfully.

Repeater Jammers

As a special case of reactive jammers, *repeater jammers* have $p_j > 0$ only if their signal acquisition, processing, transmission, and propagation delay via the attacker is less than the chip time (i.e., $T_a + T_{pr} + T_j + T_p < T_c$); otherwise the jamming signal will not interfere with the legitimate transmission due to the auto-correlation properties of the spreading codes. For a sample chip rate of 100 Mb/s, the signal travels around 3 m during the transmission of one chip ($T_c = 10^{-8}$ s). This requires the attacker to be positioned within an ellipse with major diameter $d_{AB} + 3$ m around the sender and the receiver in order to jam their communication successfully, see Figure 8. Note that this example considers only the transmission delay of a chip; the attacker's position is even more restricted for $T_a + T_{pr} + T_j > 0$. Hence, repeater jamming implies stringent conditions both on the attacker's position and on her hardware reaction times. Additionally, repeater jamming affects coordinated DSSS and UDSSS equally and we therefore focus on (UDSSS-specific) decoding jammers in the following evaluation.

6 Performance Evaluation of UDSSS

We next evaluate the performance of UDSSS. For simplicity, we first evaluate the scheme for one receiver only and then generalize the results to multiple receivers (Figure 9 displays their decoding performances). We start by analyzing the original UDSSS scheme in the absence of jamming and from that we derive the entire analysis. We will show in Section 6.4 how—in the absence of

jamming—UDSSS can easily be enhanced to yield the same performance as DSSS.

6.1 Communication without an Attacker

In the absence of malicious interference, we can expect that a UDSSS receiver will (on average) successfully decode a message once it has tried a fraction of $\frac{1}{m+1}$ of all codes, where m is the number of parallel transmissions that each use different codes. The expected time for message recovery at the receiver is therefore

$$T_r \approx T_s + T_d = \frac{s|M|N}{R} + \frac{\left(\frac{n}{m+1}Nkq + 1\right)|M|(s-1)}{\Lambda_B(N)}, \quad (1)$$

where $T_s = sT_M$ is the sampling period, $T_M := \frac{|M|N}{R}$ is the time to transmit a message, T_d is the time to decode a message, $R := 1/T_c$ is the chip rate, q is the number of samples per chip, $\Lambda_B(N)$ is the number of bit despreading operations that the receiver B can perform per second (despreading one bit requires Nq additions and multiplications), and k is the number of bits that are despread in order to decide whether the code sequence and synchronization are correct. Thus, the throughput of UDSSS is

$$L = \frac{|M|}{T_r} = \frac{|M|}{\frac{s|M|N}{R} + \frac{\left(\frac{n}{m+1}Nkq+1\right)|M|(s-1)}{\Lambda_B(N)}} \approx \frac{2\Lambda_B(N)}{nNkq(s-1)}. \quad (2)$$

The approximation holds if $T_s \ll T_d$, that is, if $s|M|N \ll R$ and $1 \ll nN$. For a state-of-the-art system that can execute about 10^{10} IPS, the time T_d to decode a message is in the order of seconds, whereas the time T_M to transmit a message is in the order of hundred μs. In the same setting, DSSS—where the used spreading code and synchronization are known to the receiver—would achieve a throughput of $\frac{|M|}{T_M} = \frac{R}{N}$, which is about one order of magnitude higher than that of UDSSS. However, UDSSS is only used when (coordinated) DSSS cannot be applied for broadcast anti-jamming communication (e.g., due to lack of shared keys). The low throughput of UDSSS should therefore be compared to zero throughput of DSSS. Furthermore, since $\Lambda_B(N) = \mathcal{O}(N^{-1})$ we get $T_r = \mathcal{O}(|M|N^2n)$ and $L = \mathcal{O}(N^{-2}n^{-1})$, showing that increasing the processing gain (i.e., N) is more harmful to the latency/throughput than increasing the code set (i.e., n). Thus, by raising n, an increase of the attacker's processing power can be counteracted with less impact on the message latency than an increase of the attacker's bandwidth and jamming power (which would require raising N).

6.2 Communication in the Presence of an Attacker

We now analyze the impact of message insertion, jamming, and overshadowing on the performance of UDSSS by using the probability p_j (p_o) that a message is jammed (overshadowed), as derived in Section 5.2. Attacker's messages whose signal strengths at the receiver are less than P_j have no impact on regular messages. Consequently, the attacker can insert only up to $n_j := \lfloor \frac{P_J}{P_j} \rfloor$ messages that will interfere with regular message transmissions, provided that they use the same spreading code sequence and synchronization as the sender. The probability that a message inserted by the attacker prevents the successful decoding of a regular message is thus $\leq n_j/(n|M|N)$. Since we assume that all messages are authenticated and integrity-protected with a signature and that the attacker is unable to forge signatures, partially modified messages will be recognized and ignored by the receivers. The only way for the attacker to effectively modify a message is thus to replace it (e.g., by replaying an overheard message).

Let ρ_i, ρ_j, and ρ_o such that $0 \leq \rho_i, \rho_j, \rho_o \leq P_J$ and $\rho_i + \rho_j + \rho_o \leq P_J$ be the power at the receiver that the attacker uses to insert, jam, and overshadow messages, respectively. The expected time to receive a message is then $T_r \leq \mathcal{T}(\lfloor \frac{\rho_i}{P_t} \rfloor, \lfloor \frac{\rho_j}{P_j} \rfloor, \lfloor \frac{\rho_o}{P_o} \rfloor)$, where

$$\mathcal{T}(n_i, n_j, n_o) = \sum_{i=0}^{\infty} p_e^{(s-1)i}(T_s + T_d) = \frac{T_s + T_d}{1 - p_e^{s-1}} = \left(\frac{sN}{R} + \frac{nNkq(s-1) + sn_i}{\Lambda_B(N)}\right)\frac{|M|}{1 - p_e^{s-1}} \approx \frac{Nkq|M|(s-1)}{\Lambda_B(N)}\frac{n}{1 - p_e^{s-1}}, \quad (3)$$

where T_s is the sampling time, T_d the time to decode a message if all codes are tried, and $p_e := (p_j(n_j) + p_o(n_o))^m$; the last approximation holds if $sn_i \leq sn \ll nNkq(s-1)$ and $s|M|N \ll R$.

Theorem 1 (Optimal Choice of the Sampling Buffer Size)**.** *Assuming that the sender is continuously broadcasting the same message, in order to capture the message, the receiver needs to have a buffer capacity of $s = T_s/T_M \geq 2$ messages. In other words, after the sampling, the buffer must contain an entire message for the despreading. Provided that $Nkq \gg 1$, a buffer capacity of $s = 2$ messages is optimal with respect to the expected time to receive a message.*

Proof. Let, by contradiction, $s^* > 2$ be the optimal capacity for the buffer. Hence, from Equation 3, $\frac{(s^*-1)nNkq|M|}{\Lambda_B(N)(1-p_e^{s^*-1})} < \frac{nNkq|M|}{\Lambda_B(N)(1-p_e)}$ must hold, i.e., $\frac{1-p_e^{s^*-1}}{1-p_e} > s^* - 1$. However, for $s^* \geq 2$ we have

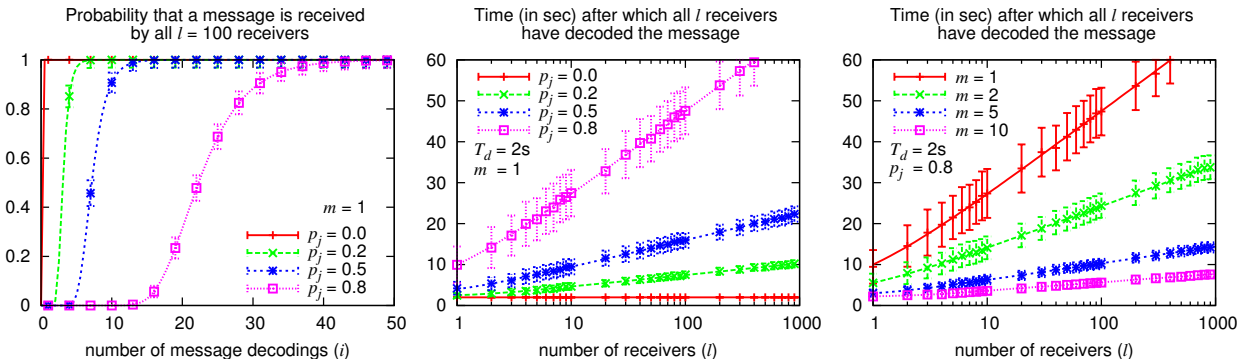


Figure 9: (a) Probability that a UDSSS message has been successfully received by all receivers as a function of the number of message decodings. (b) Expected time to disseminate a message as a function of the number of receivers; the decoding time T_d of the receivers is assumed to be 2 s. (c) Expected time to disseminate a message as a function of the number of parallel message transmissions for a decoding time T_d of 2 s. For (a) – (c), the lines show the expected result according to our analytical analyses, the points and σ-confidence intervals display simulation results.

$\frac{1-p_e^{s^*-1}}{1-p_e} \leq \lim_{p_e \to 1} \frac{1-p_e^{s^*-1}}{1-p_e} = s^* - 1$, leading to a contradiction. □

Theorem 2 (Optimal Attacker Strategy)**.** *Given that* $Nk \gg 1$*, the optimal attacker strategy against UDSSS by which the attacker maximizes the message latency is jamming. That is, for all* ρ_i*,* ρ_j*, and* ρ_o *such that* $0 \leq \rho_i, \rho_j, \rho_o \leq P_J$ *and* $\rho_i + \rho_j + \rho_o \leq P_J$*:* $\mathcal{T}(\lfloor \frac{\rho_i}{P_t} \rfloor, \lfloor \frac{\rho_j}{P_j} \rfloor, \lfloor \frac{\rho_o}{P_o} \rfloor) \leq \mathcal{T}(0, \lfloor \frac{P_J}{P_j} \rfloor, 0)$.

Proof. Since $P_j < P_o$ and by definition of p_j and p_o $\forall \alpha_1, \alpha_2 \geq 0 : p_o(\alpha_1) \leq p_j(\alpha_1)$ and $p_j(\alpha_1) + p_j(\alpha_2) \leq p_j(\alpha_1 + \alpha_2)$ it holds that $p_e = p_j(\lfloor \frac{\rho_j}{P_j} \rfloor) + p_o(\lfloor \frac{\rho_o}{P_o} \rfloor) \leq p_j(\lfloor \frac{\rho_j}{P_j} \rfloor) + p_j(\lfloor \frac{\rho_o}{P_j} \rfloor) \leq p_j(\lfloor \frac{\rho_j + \rho_o}{P_j} \rfloor)$. Hence, $\mathcal{T}(\lfloor \frac{\rho_i}{P_t} \rfloor, \lfloor \frac{\rho_j}{P_j} \rfloor, \lfloor \frac{\rho_o}{P_o} \rfloor) \leq \mathcal{T}(0, \lfloor \frac{\rho_i + \rho_j + \rho_o}{P_j} \rfloor, 0) \leq \mathcal{T}(0, \lfloor \frac{P_J}{P_j} \rfloor, 0)$; i.e., spending all power on jamming is optimal for the attacker. □

6.3 Generalization for Multiple Receivers

If two receivers are synchronized (i.e., sample the same message transmissions) and are positioned appropriately, they will encounter the same attacker-caused errors and require the same amount of time to receive the message (here we assume that the attacker is strong enough to jam all receivers with the same probability, regardless of their relative position to the sender). Moreover, the expected duration $T_r(2)$ until both receivers have successfully received the message equals the single receiver scenario (i.e., $T_r(2) = T_r$). Thus, without loss of generality, any group of receivers that sample the same message transmissions can be regarded as a single receiver.

Now, let l be the number of receivers that sample message transmissions independently (e.g., due to asynchronous sampling schedules, different propagation conditions, or differing distances from the attacker). The probability that at least one of the receivers has not yet successfully received the message once each receiver has sampled i transmissions is $1 - (1 - p_e^i)^l$. Hence, the expected duration $T_r(l)$ until all l receivers have received the message is $T_r(l) \leq \mathcal{T}(n_i, n_j, n_o, l)$, where

$$\mathcal{T}(n_i, n_j, n_o, l) = \sum_{i=0}^{\infty} \left(1 - \left(1 - p_e^i\right)^l\right) (T_s + T_d) \approx \frac{nNkq|M|}{\Lambda_B(N)} \sum_{i=0}^{\infty} \left(1 - \left(1 - p_e^i\right)^l\right). \quad (4)$$

The impact of the number of receivers on the number of required message decodings and on the time to disseminate a message by UDSSS is depicted in Figures 9a and 9b. We observe that even for a high jamming probability of 80%, all receivers have received a message with probability $\geq$ 90% after about 30 message decodings. Furthermore, the time for all l receivers to receive and decode a message is logarithmic in the number of receivers.

6.4 Optimization and Discussion

One limitation of the UDSSS scheme proposed in Section 4 is its inflexibility to the attacker's strength so that the latency will be high even if no attacker is present. In the following, we analyze techniques to improve the performance of UDSSS. We will show i) that selecting a uniform code distribution is optimal and ii) that stopping

the decoding process once a valid message was found decreases the message latency. We also show that *iii*) splitting a large code set into smaller, distinct sets for multiple senders does not decrease the message latency in general. For simplicity, we consider one receiver only but the results also hold for multiple receivers.

Theorem 3 (Optimal Code Distribution)**.** *Let $p(c_i)$ denote the probability with which code sequence $c_i \in C$ is selected by the sender. Without loss of generality, let further $1 \geq p(c_1) \geq p(c_2) \geq \ldots \geq p(c_n) \geq 0$ and $\sum_s p(c_s) = m$. Selecting c_i under a uniform distribution from a set of n^* codes (i.e., $p(c_i) = m/n^*$ for $1 \leq i \leq n^*$ and $p(c_i) = 0$ for $n^* < i \leq n$) is optimal with respect to the expected time T_r to receive a message.*

Proof. The best strategy for the attacker is to focus her jamming on those codes that are the most likely to be used. Given a code distribution function $p(\cdot)$, $\sum_{i=1}^{n} p(c_i) = m$, and $\tilde{n}_j = np_j(n_j)$ as the expected number of codes that the attacker can use in parallel to effectively block ongoing transmissions, we get $p_e := \prod_{i=\tilde{n}_j+1}^{n}(1 - p(c_i))$. It follows from Eq. 3 that T_r is minimized if p_e is minimized, that is, if $p(c_i) = m/n^*$ for $1 \leq i \leq n^*$ and $p(c_i) = 0$ for $n^* < i \leq n$; the optimal number n^* of codes can (numerically) be derived from (3) once $p(c_i)$ and $\tilde{n}_j$ are given. □

Early Termination at the Receiver. The expected time to receive a message can be reduced if the receiver stops the despreading process once it verified a valid message. Here,

$$T_r \leq \sum_{i=1}^{\infty} p_e^{mi}(T_s + T_d) + \frac{Nkq|M|}{\Lambda_B(N)} \sum_{i=0}^{m-1} p_e^i \frac{n}{m+1}$$
$$\approx \frac{Nkq|M|}{\Lambda_B(N)} \left(\frac{np_e^m}{1-p_e^m} + \frac{n}{m+1} \frac{1-p_e^m}{1-p_e} \right), \quad (5)$$

where the first term accounts for the number of unsuccessful transmission rounds and the second term is the expected time for the decoding in the last, successful round. Figure 10 compares the expected despreading times of the original UDSSS scheme and the scheme with early termination for multiple senders depending on the jamming probability.

Theorem 4 (Multiple Code Sets)**.** *Consider m sending devices with code sets $C_1, \ldots, C_m$, where $C_i \cap C_j = \varnothing$ for $i \neq j$, $|C_1| \leq |C_2| \leq \cdots \leq |C_m|$, and $\sum_i |C_i| = n$, which are broadcasting messages in parallel; the probability for each code sequence $c_j \in C_i$ to be used in the current transmission is $p(c_j) = \frac{1}{|C_i|}$. The expected time T_r to receive a message is equal to the case where the m messages are chosen from one common set C of size n such that $p(c_j) = \frac{m}{n}$.*

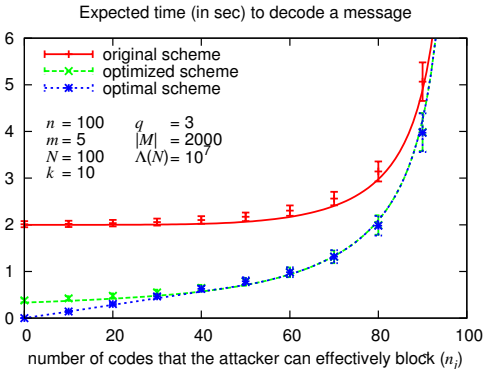


Figure 10: Expected time to disseminate one of the messages from five senders to one receiver. Shown are the original scheme (Equation 3) and the optimization using early termination of the despreading (Equation 5); the optimal scheme uses n^*. We observe that the optimized scheme is close to optimal for $\tilde{n}_j < n/2$ and optimal if $\tilde{n}_j \geq n/2$.

Par.	Definition	Value Range
N	spreading code size	$\geq$ 128 chips (21 dB)
n	#code sequences	up to performance limits
l	#codes per spr. sequence	$\|M\|$
k	#despread data bits per spreading sequence	$2 \leq k \leq$ #bits the error-encoding can correct
s	buffer size sT_M	2
τ	integration threshold	$N/2$ or 2ε

Table 1: UDSSS parameter settings. The larger N and n are, the more jamming-resistant the scheme is. k, s, and τ do not affect the jamming resistance but the decoding performance. A rough, but good estimate for τ is $N/2$; more precise values can be determined by simulations (see Fig. 4), e.g., τ = 90 (200) for N = 256 (1024).

Proof. Let a_i denote the number of codes the attacker blocks from the set C_i. The attacker's optimal strategy is to select each a_i such that she maximizes the probability $\tilde{p}_e = \prod_{i=1}^{m} \frac{a_i}{|C_i|}$ that all m messages are blocked, under the constraints $\sum_{i=1}^{m} a_i \leq \tilde{n}_j$ and $a_i \leq |C_i|$ $\forall i \in \{1, .., m\}$. Hence, $\tilde{p}_e$ is maximized if $\frac{a_i}{|C_i|} = \frac{a_j}{|C_j|}$, i.e., if the attacker jams each code set with the same probability. Then, $|C_i| = \frac{n}{m}$ (Th. 3) and $a_i = \frac{\tilde{n}_j}{m}$, thus $\tilde{p}_e = \prod_{i=1}^{m}(\frac{\tilde{n}_j m}{nm}) = (\frac{\tilde{n}_j}{n})^m$. This probability is equal to the probability $p_e = p_j(n_j)^m = (\frac{\tilde{n}_j}{n})^m$ that m messages are blocked if the codes are chosen out of a set of size n where the attacker can block $\tilde{n}_j$ codes. □

Although splitting a large code set into smaller sets for multiple senders is not beneficial for the latency in general, we can achieve the same message latency as (non-synchronized) DSSS in the absence of jamming by

Figure 11: Experimental hardware setup of the UDSSS implementation, consisting of a Universal Software Radio Peripheral (USRP) and a Lenovo T61 ThinkPad.

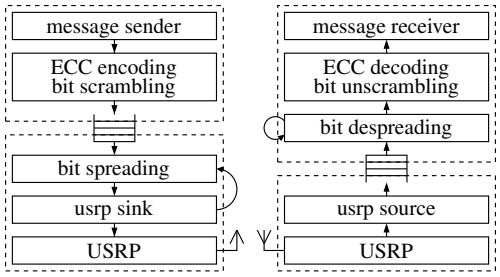


Figure 12: Schematic description of our UDSSS sender and receiver application.

choosing $m = 2$ with $C_1 = \{c_1\}$, $p(c_1) = 1$, and $p(c_2) = \frac{1}{|C_2|}$. In the absence of jamming, the first code $c_1 \in C_1$ used by the receiver will succeed.

Parameter Selection. The exact UDSSS parameter values depend on the hardware in use and on the assumed attacker strength. The values presented in Table 1 may therefore vary depending on the hardware and application. In general, the product $nN|M|$ represents the security parameter of UDSSS and should at least be in the order of 10^6; the smaller $|M|$ is the more jamming-resistant the scheme is.

6.5 Implementation Results

In this section, we demonstrate the feasibility of our UDSSS scheme by means of a prototype implementation based on Universal Software Radio Peripherals (USRPs) [10] and GnuRadio [1] (see Figure 11). The USRPs include a A/D (D/A) converter that provides an input (output) sampling rate of 64 Mb/s (128 Mb/s) and an input (output) sample resolution of 12 bits (14 bits); the employed RFX2400 daughterboards were configured to use a carrier frequency of 2.4 GHz. In our experiments, two USRPs (one being used as a UDSSS sender, the other as a UDSSS receiver) were each connected via a 480 Mbps USB 2.0 link to a Lenovo T61 ThinkPad (Intel Core 2 Duo CPU @ 2.20 GHz) running Linux (kernel 2.6.27) and GnuRadio (version 3.0.3). For performance reasons and for ease of deployment, our UDSSS sender and receiver applications were written entirely in C++, which required porting some GnuRadio libraries from Python to C++. A schematic scheme of our implementation is given in Figure 12.

The sender first encodes the message with a (8,4) Hamming code and scrambles (interleaves) the bits according to a public pseudo-random permutation. Next the sender chooses a spreading code sequence uniform at random, spreads the (encoded and scrambled) message with this code, and sends the resulting chip sequence to the USRP using a differential encoding: the current phase of the baseband signal remains unchanged for a $+1$ and its phase is shifted by 180° for a -1. This step (i.e., choosing a code, spreading, and sending the message) is repeated until the sender stops the message transmission.

The receiver samples the channel for a duration of $2T_M$, where T_M is the transmission time of a message, decodes the samples into a chip sequence, and stores the sequence into a FIFO buffer. A second thread reads the sequences from the FIFO buffer, decodes all possibly included messages by trying all n code sequences on all $N|M|$ positions. To decide whether a code and position pair is valid, a two-level test is used: The sender first despreads two randomly selected bits. If the absolute value of the code-bit correlation for at least one of the bits is $\geq N/2$, it decodes (i.e., despreads, unscrambles, and error-corrects) the first 8 bytes of the message. If these 8 bytes are also valid, the whole message is decoded and the included signature verified.

In our experiments, we positioned the UDSSS sender and receiver indoors at a distance of about 5 m and performed a series of message transfers using UDSSS from the sender to the receiver. The size of the transmitted messages was 256, 512, 1024, 1536, and 2048 bit. The code sets contained up to 500 pseudo-random code sequences and the length of these codes was in the range from 32 to 512 chips. Figures 13a and 13b display the decoding times as a function of the message size $|M|$, code length N, and code set size n. We observe that the decoding time increases linearly with the message size and code set size but quadratic with the code length; this observation is in line with our analytical model. The results further show that, even with this non-optimized (software-based) system, the expected time to receive and decode a typical message ($|M| \leq 2048$ bit) is well below 20 s (for a processing gain of 21 dB and $n = 100$).

We point out that the main purpose of this USRP/CPU-based system is to demonstrate the feasibility of UDSSS. The achieved decoding times should thus not be considered as performance benchmarks. As the operations to decode a bit can easily be executed in parallel, decoding a bit is typically a single-step operation on hardware-based

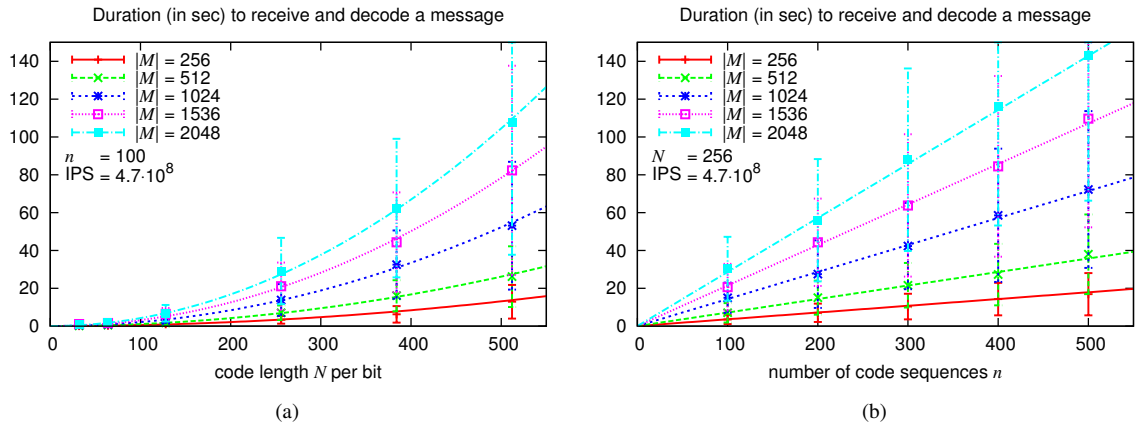


Figure 13: Implementation Results. The plots show the time to receive and decode a message with our UDSSS implementation as a function of the message size $|M|$, code length N, and code set size n. The points and σ-confidence intervals represent the measurements results, the lines the analytical results for a processing speed of about 470 MIPS. We observe that the decoding time increases linear with the message size and code set size but quadratic with the code length. Even with this non-optimized (software-based) system, the expected time to receive and decode a typical message ($|M| \leq 2048$ bit) is well below 20 s (for a processing gain of 21 dB and $n = 100$). On hardware-based DSSS transceivers, bit decoding operations are usually executed in parallel. Purpose-built UDSSS receivers are thus likely to achieve decoding times that are about $O(N)$ times (i.e., 10-1000 times) lower than the times presented in this figure.

DSSS receivers. Realistic decoding times of purpose-built UDSSS transceivers will thus be $O(N)$ times (i.e., 10-1000 times) lower than what we achieve with the presented implementation. As a next step, we intend to optimize our implementation by adding Streaming SIMD Extensions (SSE) support to the core despreading functions and by offloading some of the work to the GPU of the graphic card.

7 Outline of UDSSS Applications

In this section, we present applications for UDSSS broadcasts. The scenarios we will describe share a risk of jamming and of potentially malicious users; in these settings, DSSS communication would either be infeasible or could easily be disrupted by jammers. We demonstrate that the delays which are introduced by the UDSSS trial-and-error reception still enable practical and security-relevant applications.

We consider one or multiple senders that want to disseminate information by broadcasting messages to a set of receivers in a jammed environment. Each receiver holds the authentic public key of the sender but does not share a secret key with it. Such a situation can occur if the sender wants to communicate to a set of *untrusted* receivers that may want to deprive other receivers from obtaining the information broadcasted by the sender, or if a set of *trusted* receivers is dynamic, unknown, or even unpredictable (hence, authentic secret keys between sender and receivers cannot be established beforehand).

Examples for such settings are i) *emergency notification (pager) systems* (e.g., Plectron [19]) used to activate emergency response personnel and disaster warning systems or ii) *central (governmental) authorities* that need to inform the public about the threat of an imminent or ongoing (terrorist) attack. The danger that attackers jam the alert transmission needs to be minimized. Information dissemination in this setting is clearly time-critical, however, being able to distribute the information within seconds to few minutes is clearly preferred over not being able to disseminate any information at all under jamming. We further argue that, in the absence of jamming, UDSSS permits delays as short as DSSS does (see Section 6.4) and that, once the information has been received by some devices, other communication means (e.g., speech or landline) may additionally support its dissemination to more people concerned.

Another notedly well-suited application for UDSSS is the broadcast of *navigation signals* which are foremost used for time synchronization and localization. Examples of navigation systems include satellite navigation (e.g., GPS [27]) and terrestrial systems such as Loran [13] (based on TDoA) and DME-VOR [5] (based on distance/angle measurements). Localization and time-synchronization systems require the reception of navigation signals from multiple base stations; in general, at least three or four different signals are necessary for most localization methods [5]. The broadcast stations are precisely time-synchronized (e.g., via wired links) and located at static or predetermined positions. Each broad-

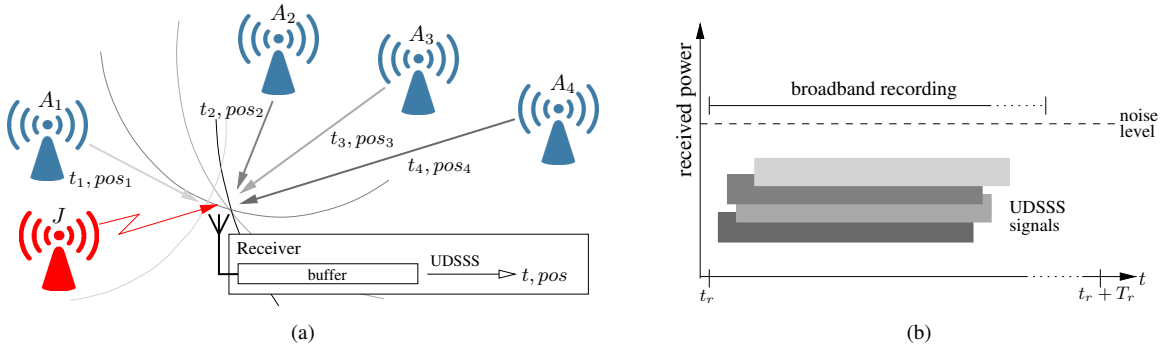


Figure 14: (a) Possible application of UDSSS: jamming- (and spoofing-)resistant reception of navigation signals used for positioning (pos) and/or time-synchronization (t). The receiver records the signals of multiple senders which were spread using randomly selected spreading sequences and uses UDSSS decoding to retrieve the sent messages and compute its position and/or local time. (b) UDSSS signals are highly resistant against narrow-band jamming attacks (by jammer J) because they are sent entirely below noise level. UDSSS likewise prevents signal-delay attacks, because the attacker can only delay individual navigation signals after her decoding, i.e., after having identified the used spreading sequences.

cast station transmits navigation signals either continuously due to a fixed schedule (GPS, Loran-C) or sends replies to individual localization requests (DME-VOR, WLAN-localization), based on which the localized device determines its position.

Without appropriate protection, navigation signals are vulnerable to signal spoofing, synthesis, and jamming attacks [14, 21]. E.g., while current civilian implementations using GPS satellite signals [27] or terrestrial WLAN signals [3] (based on the 802.11b standard) apply spreading to make the transmissions resistant to *unintentional* interference, they do not provide any means to counteract targeted Denial-of-Service (DoS) attacks because their spreading codes are public and can thus be misused for jamming.

UDSSS offers an enhancement to the dissemination of navigation signals that counters targeted jamming. Navigation messages are typically in the order of several hundred bits (e.g., 1.5 kb for GPS messages [18]) and will—even comprising authentication credentials—fit into the considered UDSSS message lengths (in our evaluation in Section 6, the messages were up to 2048 bits long). Each base station uses randomly selected code sequences to spread the messages using UDSSS. The property of the wireless channel enables the receivers to record samples of several navigation signals in parallel in one buffer (same principle as multi-user CDMA). The receivers can then use UDSSS decoding in order to extract three (or more) individual messages (along with their precise arrival times) in one decoding, verify their authenticity, and therefrom derive position and/or time information (see Figure 14a). Unlike DSSS, UDSSS cannot decode navigation signals in real time, but decodes them with a delay T_r, which is largely determined by the processing speed of the receiver. Depending on the implementation and underlying hardware, this delay may vary up to several seconds. However, even if UDSSS causes a delay, the computed position and time are accurate since UDSSS still enables the receiver to record the exact arrival times of the signals it receives.

The delay introduced by the UDSSS decoding is of little importance for pure *time-synchronization* because time represents a rather stable property of a device: Once it is accurately determined, time may slowly degrade by clock drift depending on the clock quality, but it is usually not reset as abruptly as a new position for a mobile device. In this case, after the decoding and processing of the navigation signals, the local time t of the device will be set to $t = t_s + d_p + T_r$, where t_s is the timestamp derived from the base station signals, d_p is the aggregated signal propagation delay (estimated or calculated using the position information, around $30\,\mu s$ for 10 km), and T_r is the local time needed for decoding and processing at the receiver (measured time between the first bit filled into the buffer and the moment the time is reset).

So far, we have only discussed the implications of UDSSS on navigation signals in terms of anti-jamming. We now further show that UDSSS equally helps to secure navigation against *spoofing* attacks. In [14], Kuhn showed that time-of-arrival-based navigation systems (like GPS) can be secured against signal-synthesis and selective-replay attacks in which the attacker inserts navigation signals as they would be received at the spoofed location. Without protection, an attacker can manipulate the (nanosecond) relative arrival times by pulse-delaying or replaying of (individual) navigation signals with a delay of Δ, which results in a distance error $c(\delta + \Delta)$ with respect to the true location (where c is the speed of light

and δ accounts for synchronization imprecisions). The asymmetric scheme proposed in [14] is made resistant against these kinds of attacks by decoupling the time-critical signal transmission from a delayed disclosure of the applied spreading code; the first signal is spread and hidden below noise level whereas the second signal (spreading code along with time and position information) is transmitted above the noise level after a delay ρ. A replay attack can now be performed only with a delay $> \rho$. By choosing ρ large enough (e.g., several seconds), even receivers with a low-quality clock can discover the delay in the received timestamps.

UDSSS achieves a similar anti-spoofing protection as the scheme in [14]. Due to the steganographic properties of the UDSSS signal, the attacker can only extract and delay individual navigation signals after having successfully identified the used spreading sequences. Due to a comparison of the received timestamp with the local time, the receiver can identify signal delays that exceed a certain accepted threshold; the threshold basically depends on the accuracy of the receiver's clock. This (probabilistic) approach secures against attacks in which the attacker's decoding takes longer than this threshold.

In contrast to the scheme in [14], which is susceptible to DoS-attacks since data and code are disclosed above noise level, UDSSS provides resistance against jamming because the entire navigation signals are sent with (temporarily) unknown code sequences below noise level (see Figure 14b).

8 Related Work

The impact and detectability of jammers according to their capabilities (e.g., broad- or narrowband) and behavior (e.g., constant, random, reactive) has been widely studied [2, 15, 20, 28]. Spread-spectrum techniques such as DSSS and FHSS are common jamming countermeasures [2, 20]. In [6, 8], the respective authors address broadcast jamming mitigation based on spread-spectrum communication. Additionally, the use of specific coding and interleaving strategies [16] can strengthen the jamming resistance of transmitted messages. Common to these countermeasures is that they all rely on secret keys, shared between the sender and receiver(s) prior to their communication. As argued in prior work [24], preloading keys on devices in ad-hoc settings for subsequent jamming-resistant communication suffers from scalability and receiver dynamics problems. Furthermore, if some of the receivers are not trustworthy, relying on pre-shared keys allows malicious receivers to obtain messages themselves while withholding them for others [14].

Recent observations [4, 24] identify the lack of methods for jamming-resistant communication without shared secrets and propose solutions to this problem. The solution proposed by Baird et al. [4] uses concurrent codes in combination with UWB pulse transmissions. The jamming resistance achieved by their scheme is not one-to-one comparable to common spread-spectrum-based techniques: While the attacker of spread-spectrum techniques must have enough transmission power to overcome the processing gain, in [4] the limiting factor is the number of pulses that the attacker can insert, i.e., her energy. The solution previously proposed based on Uncoordinated Frequency Hopping (UFH) [24] chooses the frequencies of packet transmissions at random from a fixed frequency band. UFH and UDSSS differ significantly in the following aspects: UDSSS is deterministic (apart from the randomness introduced by the attacker) and its performance (the transmission latency) mainly depends on the receiver's processing capabilities. UFH, in contrast, is probabilistic (even in the absence of jamming) and its performance depends on the number of hopping channels (determined by the processing gain). Unlike UFH, UDSSS decouples the processing gain from the spreading uncertainty and allows to fine-tune the scheme (without complex message fragmentations). Finally, due to the unpredictability in the message reception, UFH is unsuitable for applications that require accurate time-stamping of signals, as it is required for many navigation systems.

In [29], an algorithm to estimate the code sequence of a direct spread-spectrum sequence in non-cooperative communication systems is proposed. This algorithm, however, does not leverage the knowledge of the code set used and further assumes that the same code sequence is used repetitively. This approach is therefore not suitable to counter targeted jamming attacks because the communication will no longer be protected once the code sequence has been identified by the attacker.

9 Conclusion

In this paper, we elaborated the problem of broadcast anti-jamming communication without shared secrets, which can, e.g., be used to secure navigation systems. As a solution to this problem we proposed a scheme called Uncoordinated DSSS (UDSSS) that enables DSSS-based broadcast communication *without* pre-shared keys. UDSSS leverages the fact that the sender can transmit a certain amount of spread (hidden) data to the receivers before a (reactive) jammer is able to identify the used code and to jam the transmission. We evaluated the performance and jamming resistance of our DSSS scheme analytically, through a prototype implementation, and by means of simulations for single and multiple receivers. For a state-of-the-art system (about 6000 MIPS), the expected time for a message transfer to a group of 10 receivers takes less than 30 s for a high jam-

ming probability of 80%. We accent that this time is reasonably short, given that with common (key-dependent) anti-jamming techniques the devices would not be able to broadcast jamming-resistant messages at all.

10 Acknowledgments

We are grateful to Fabian Monrose for his valuable input. We also thank the anonymous reviewers for their suggestions. The work presented in this paper was partially supported by the Swiss National Science Foundation under Grant 200021-116444.

References

[1] GNU Radio Software. http://gnuradio.org/trac.

[2] ADAMY, D. *A first course in electronic warfare*. Artech House, 2001.

[3] BAHL, P., AND PADMANABHAN, V. N. RADAR: An In-Building RF-Based User Location and Tracking System. In *Proceedings of the IEEE Conference on Computer Communications (InfoCom)* (2000), vol. 2, pp. 775–784.

[4] BAIRD, L. C., BAHN, W. L., COLLINS, M. D., CARLISLE, M. C., AND BUTLER, S. C. Keyless Jam Resistance. In *Proceedings of the IEEE Information Assurance and Security Workshop* (June 2007), pp. 143–150.

[5] BENSKY, A. *Wireless Positioning Technologies and Applications*. GNSS Technology and Applications Series. Artech House, 2008.

[6] CHIANG, J., AND HU, Y.-C. Dynamic jamming mitigation for wireless broadcast networks. In *Proceedings of the IEEE Conference on Computer Communications (InfoCom)* (2008).

[7] CHUNG, F. R. K., SALEHI, J. A., AND WEI, V. K. Optical orthogonal codes: Design, analysis and applications. *IEEE Transactions on Information Theory 35*, 3 (1989), 595–604.

[8] DESMEDT, Y., SAFAVI-NAINI, R., WANG, H., CHARNES, C., AND PIEPRZYK, J. Broadcast anti-jamming systems. In *Proceedings of the IEEE International Conference on Networks (ICON)* (1999).

[9] DILLARD, R. A., AND DILLARD, G. M. *Detectability of Spread-spectrum Signals*. Artech House Publishers, 1989.

[10] ETTUS. USRP – Universal Software Radio Peripheral. http://www.ettus.com.

[11] GOLDSMITH, A. *Wireless communications*. Cambridge University Press, 2005.

[12] HANG, W., ZANJI, W., AND JINGBO, G. Performance of DSSS against repeater jamming. In *Proceedings of the IEEE International Conference on Electronics, Circuits, and Systems (ICECS)* (2006), pp. 858–861.

[13] INTERNATIONAL LORAN ASSOCIATION. LORAN: LOng Range Aid to Navigation. http://www.loran.org.

[14] KUHN, M. G. An asymmetric security mechanism for navigation signals. In *Proceedings of the Information Hiding Workshop* (2004), pp. 239–252.

[15] LI, M., KOUTSOPOULOS, I., AND POOVENDRAN, R. Optimal jamming attacks and network defense policies in wireless sensor networks. In *Proceedings of the IEEE Conference on Computer Communications (InfoCom)* (2007), pp. 1307–1315.

[16] LIN, G., AND NOUBIR, G. On link layer denial of service in data wireless LANs: Research articles. *Wireless Communications & Mobile Computing 5*, 3 (2005), 273–284.

[17] NATARAJAN, B., DAS, S., AND STEVENS, D. An evolutionary approach to designing complex spreading codes for DS-CDMA. *IEEE Transactions on Wireless Communications 4*, 5 (2005), 2051–2056.

[18] NAVSTAR SPACE AND MISSILE SYSTEMS CENTER. Navstar Global Positioning System: Interface Specification IS-GPS-200. http://www.losangeles.af.mil, 2006.

[19] PARNASS, B. Plectron R-700 Monitor Receivers. *Monitoring Times Magazine* (October 1999).

[20] POISEL, R. A. *Modern Communications Jamming Principles and Techniques*. Artech House Publishers, 2006.

[21] RASMUSSEN, K. B., ČAPKUN, S., AND ČAGALJ, M. SecNav: secure broadcast localization and time synchronization in wireless networks. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)* (2007), pp. 310–313.

[22] SARWATE, D. V., AND PURSLEY, M. B. Crosscorrelation properties of pseudo-random and related sequences. In *Proceedings of the IEEE* (May 1980), vol. 68, pp. 593–619.

[23] SKLAR, B. *Digital communications: fundamentals and applications*. Prentice-Hall, 2001.

[24] STRASSER, M., PÖPPER, C., ČAPKUN, S., AND ČAGALJ, M. Jamming-resistant key establishment using uncoordinated frequency hopping. In *Proceedings of the IEEE Symposium on Security and Privacy* (2008), pp. 64–78.

[25] TJHUNG, T. T., AND CHAI, C. C. Multitone Jamming of FH/BFSK in Rician Channels. *IEEE Transactions on Communications 47*, 7 (July 1999).

[26] TSE, D., AND VISWANATH, P. *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[27] U.S. GOVERNMENT. Global positioning system. http://www.gps.gov, March 2008.

[28] XU, W., TRAPPE, W., ZHANG, Y., AND WOOD, T. The feasibility of launching and detecting jamming attacks in wireless networks. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)* (2005), pp. 46–57.

[29] ZHAN, Y., CAO, Z., AND LU, J. Spread-spectrum sequence estimation for DSSS signal in non-cooperative communication systems. In *IEE Proceedings Communications Magazine, IEEE* (Aug. 2005).

Notes

[1] Gold- and Kasami-codes have the same correlation properties and both approach the Welch lower bound in their cross-correlation values. However, Gold- and Kasami-codes differ in the number of codes that can be created. While the number of Gold-codes of length N that can be constructed is $N+2$ (e.g., 257 for $N = 255$ and 1025 for $N = 1023$), the number of Kasami-codes of length N (in the large set) is considerably higher: $\approx 2^{\frac{3}{2}log_2(N+1)}$ (e.g., 4112 for $N = 255$ and 32800 for $N = 1023$) [17]. Kasami-codes are therefore more appropriate for UDSSS, although even Kasami codes may have to reoccur in multiple code sequences (if $n\ell > 2^{\frac{3}{2}log_2(N+1)}$).

[2] $\forall c_{i,j} \in C$, $\forall t \in \{0, 1, \ldots, N-1\}$, and a small $\varepsilon \ll N$, the *auto-correlation* of the codes is $\sum_{q=0}^{N-1} c_{i,j}[q]c_{i,j}[q+t \bmod N] \approx N$ if $t = 0$ and $\leq \varepsilon$ else, where $c_{i,j}[q] \in \{-1, +1\}$ denotes the q-th value of the spreading code $c_{i,j}$ and ε indicates the quality of the auto-correlation (the less the better). Similarly, $\forall c_{i,j}, c_{i',j'} \in C, t \in \{0, 1, \ldots, N-1\}$ the *cross-correlation* is $\sum_{q=0}^{N-1} c_{i,j}[q]c_{i',j'}[q+t \bmod N] \leq \varepsilon$.