Jamming-resistant broadcast communication without shared keys

Jamming-resistant Broadcast Communication without Shared Keys

ETH Zurich D-INFK Tech. Report 609 – March 12, 2009. Accepted for publication at Usenix Security Symp. 2009

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, relying broadcast anti-jamming communication on either secret pairwise or group keys suffers from serious and sometimes even unsolvable scalability and key-setup problems or from 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 a certificate of the sender's public key, 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 by an experimental evaluation on a prototype implementation.

1 Introduction

Due to the shared use of the communication medium, wireless radio communications are not only vulnerable to traditional attacks such as eavesdropping and message synthesis, but also to active jamming attacks [2,19]. 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 (FH) [22]. 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 both [4] and [23] in the context of pairwise communications.

In this work, we focus on a related, but different problem for broadcast communications: *How to enable robust anti-jamming broadcast without shared secret keys*? Typical broadcast applications share the need for guaranteed 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 may communicate to a dynamic set of *trusted* receivers (i.e., the nodes are honest, but unknown to the sender due to receiver dynamics) or to *untrusted* receivers (which might be interested in obtaining the information themselves but preventing others from getting 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 deployment of the keys or because malicious nodes may misuse shared keys for jamming. We can best illustrate this by an example:

A central governmental authority needs to inform the public about the threat of a (terrorist) attack. For disseminating information about the risk, a message $M = l|t|area|sign_{CA}$ could contain the level of risk l, a timestamp t, the physical $area$ of risk, and the signature $sign_{CA}$ 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. Crucial is the dissemination (broadcast) of *authentic* information to as many receivers as possible within a reasonable timeframe (seconds to minutes). Particular receivers can then make use of different communication channels (voice, sight, etc.) in order to propagate the information further among each other.

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 still 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 already described example, another 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 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 identify navigation services for secure localization and time-synchronization as a typical broadcast application for uncoordinated spread-spectrum techniques.
- 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 BPSK) 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 synchronization 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, 19, 22].

For spreading a message M, the sender uses a *spreading sequence* $c_0 = (c_{0,1}, c_{0,2}, \ldots, c_{0,\ell})$ that is composed of ℓ binary non-return to zero (NRZ) *spreading codes*, $|c_{0,i}| = N$. Typical spreading codes used for DSSS are pseudo-randomly created sequences [22] and codes with well-defined properties such as Walsh-Hadamard [11] or Gold-codes [19]. 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. If we express the codes $c_{0,i}$ 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) denotes the data bit (signal 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 the message and transmits it.

Upon signal reception, each receiver demodulates the signal and samples it using an A/D converter with a sampling rate $R_s \geq 2/T_c$. It stores the samples in a cyclic buffer which has the capacity to store samples for 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 establish 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. 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 [23] and consider an omnipresent but computationally bounded adversary J who 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., 15% 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 [19] 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. 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 a receiver 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 $P_j < P_o$.

4 Jamming-Resistant Broadcast: UDSSS

In this section, we introduce our UDSSS (Uncoordinated DSSS) scheme. UDSSS is an anti-jamming modulation technique that is based on the concept of DSSS. However, UDSSS 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 of spreading sequences, C, 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 C 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.

As a further difference to DSSS, the performance and jamming-resistance of UDSSS can be increased by using multiple senders. 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—each spread with a different spreading code—in parallel or by using m separate sending devices.

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

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 using a D/A converter. We stress that 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 additional techniques: In order to achieve message authentication, sender A signs the message using its private key SK_A. The sender may also include a

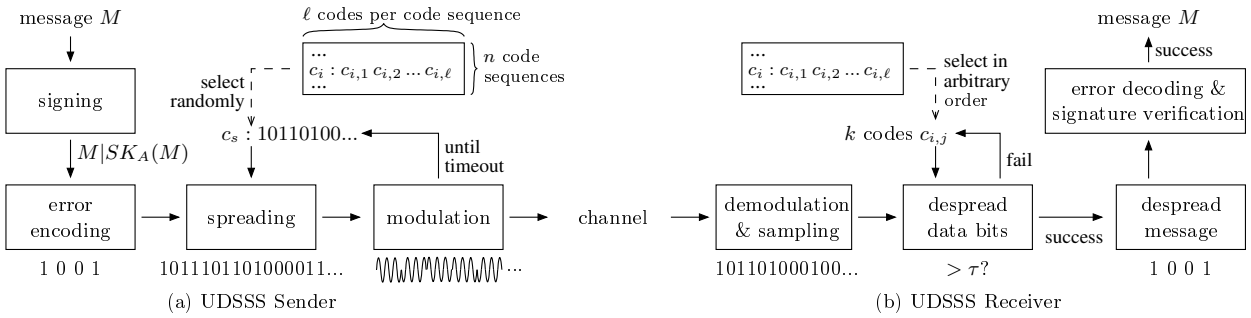


Figure 1: (a) UDSSS transmission. Sender A signs and error-encodes the message M. Then it repeatedly spreads the signed and error-encoded data using a *freshly* selected spreading sequence c_s in each repetition and transmits the modulated signal. (b) UDSSS reception. Receiver B demodulates and samples 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.

timestamp or sequence number in the message in order to achieve replay protection. In order to resist transmission errors, the sender subsequently 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 15% 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 for each sender is displayed in Figure 1(a).

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 or needs to repeat the transmission of each message such that, when the receivers fill their reception buffers (the interval of which is determined by their decoding times), they will receive the message. An example is shown in Figure 2.

4.2 UDSSS Reception

In UDSSS, the receiver[1] samples the radio channel using an A/D converter (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 here 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). When the receiver fills the buffer, it will reject all other signals arriving to its interface (Fig. 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 $1 \leq j \leq \ell$, see Figure 3a) from each code sequence $c_i \in C$ and uses them to despread k data bits, as sketched in Figure 1(b). $k > 1$ 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 mes-

[1]For ease of description, we focus on *one* receiver, however, the description applies for *each* receiver.

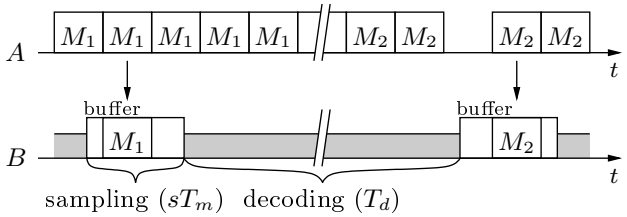


Figure 2: UDSSS message sampling and decoding. After the sampling, the receiver B decodes the message M_1 contained in the buffer. During the decoding, B disregards all further samples.

sage, 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. Depending on the available hardware, the despreading operation can partially be performed in parallel or using a multi-stage solution [19].

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 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

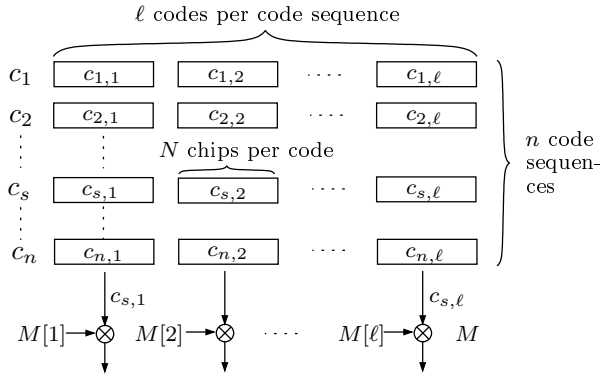
As a crucial component of UDSSS, we now describe how to generate the UDSSS spreading codes that are used by the sender and receivers. In our description, we refer to *one* code set C, however, the same applies for *each* code set in the case of multiple senders. Figure 3a illustrates the spreading code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

UDSSS requires the use of spreading codes that are balanced and have good auto- and cross-correlation properties; good auto-correlation properties are needed for precise synchronization at the receivers and low cross-correlation properties prevent that different spreading codes 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*[2] [17] as well as *pseudo-randomly created sequences* [21].

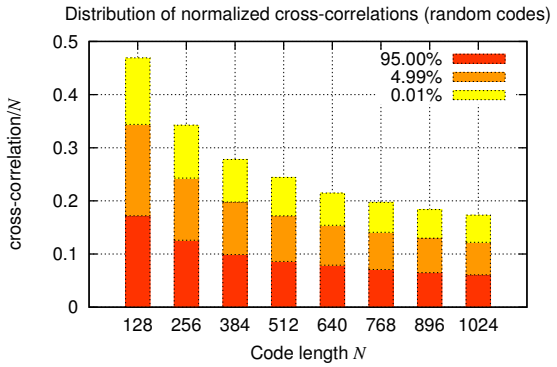
Due to their straight-forward generation, we focus on pseudo-random codes in the following. A specific code set C is then given by a (unique) seed sd, used as input to a well-defined pseudo-random number generator. Given a sufficiently large code length N, pseudo-random codes will have good auto- and cross-correlation properties[3]. Figure 3b displays the cross-correlation values of pseudo-random codes and

[2]Gold- and Kasami-codes have the same correlation properties and both approach the Welch lower bound in their cross-correlation values [25]. 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., $257_{N=255}$ and $1025_{N=1023}$), the number of Kasami-codes of length N is much higher: $\approx 2^{\frac{3}{2}log_2(N+1)}$ (e.g., $4112_{N=255}$ and $32800_{N=1023}$) [17]. Depending on the number of required codes in the spreading code set C and, thus, on the assumption of the attacker strength, Kasami-codes are more appropriate for UDSSS.

[3]$\forall c_{i,j} \in C, \forall t \in \{0, \ldots, N\text{-}1\}$, and a small $\varepsilon \ll N$, the *auto-correlation* of the codes is $\sum_{q=0}^{N-1} c_{i,j}[q]c_{i,j}[q + t \bmod N] \gg \varepsilon$ 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, $\forall c_{i,j}, c_{i',j'} \in C, t \in \{0, \ldots, N\text{-}1\}$ the *cross-correlation* is $\sum_{q=0}^{N-1} c_{i,j}[q]c_{i',j'}[q +$



(a) Code set C



(b) Cross-correlations of random code sequences

Figure 3: **(a)** 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. **(b)** Distribution of cross-correlations for 10^3 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.

confirms the desired property; for a more comprehensive analysis of the properties of pseudo-random sequences refer to [21]. 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 to 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), $\varepsilon \lessapprox 250$ (Figure 3b). This enables us to set τ used as integration threshold by the receiver: $\tau = k\varepsilon$, $k \in \mathbb{Q} \geq 1$.

Furthermore, the probability that any two random codes of length N from a set of $n\ell$ codes agree is approximately $1 - e^{-(n\ell)^2/2^N}$ (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 $c_{i,j}$. In other words, if the attacker successfully identifies the code that was used for spreading any particular message bit, she will likewise know the entire code sequence that was used and can thus jam the remainder of the message. This must be taken into account for the analysis of the attacker's decoding strength (see Section 5.2). Section 6.5 will further provide numbers for the size n of the code set to be used in practical applications and will displays 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 derive the probability that a message is jammed during its transmission for different types of attackers. As will show, our UDSSS scheme 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 communications: (1) she can guess the spreading code and try to jam the signal using this code, (2) she can repeat the recorded signal, thus trying to create a collision with the original transmission, and (3) she can try to find the code by despreading (part of) the spread signal and then using the identified spreading sequence for

$t \bmod N] \leq \varepsilon$.

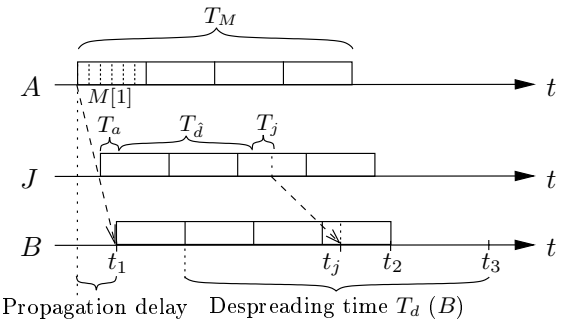


Figure 4: UDSSS attack scenario for reactive jammers. Sender A sends message M during the transmission time T_M. Receiver B and a reactive attacker J start to despread the (same) signal samples after having recorded the first data bit $M[1]$. J's jamming attack will succeed only if $t_j < t_2$, i.e., if the attacker succeeds to despread $M[1]$ and to send its jamming signal such that it reaches B before B receives the whole message M; otherwise the jamming fails and B despreads 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, $< (t_2 - t_1)$, to despread the message, whereas the receiver can despread the message long after having recorded it.

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 or random attacker); in the second and third cases, the attacker bases her jamming signal on the detection (and analysis) of the spread signal (reactive attacker). A hybrid jammer can combine these actions.

By using a new spreading code for each message bit ($b = 1$), successful replay attacks from repeater jammers [12] (case 2) are prevented due to the low auto-correlation properties of the codes used. For non-reactive (static or random) attackers (case 1), the attacker's success probability depends on the number of codes that she chooses from for composing her jamming signal. At the same time, it depends 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, 19]. 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 she 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 synchronization.

Figure 4 displays the attack scenario for case 3. The time that a reactive attacker has to detect the spreading code used by the sender and to exploit this knowledge to jam the message is limited by the transmission time of the message and the fraction that needs to be jammed. For this reason, the attacker's success probability depends on the time that she needs to identify the used spreading code and its synchronization with respect to the received signal. Hence, 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 (larger C is better). 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 (Section 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) (Section 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, Random, and Sweep Jamming

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 random jammers the jamming probability is $p_j(n_j) \leq 1 - (1 - \frac{n_j}{n|M|N})^{T_M/T_{\overline{M}}}$, and for sweep jammers it is $p_j(n_j) \leq \min\{\frac{n_j}{n|M|N}\frac{T_M}{T_{\overline{M}}}, 1\}$. Note that the attacker has to hit both the right code sequence (out of n sequences) and the right chip synchronization ($N|M|$).

Reactive and Hybrid Jamming

A reactive 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. The attacker's ability to jam a message is thus determined by the time that the attacker needs to guess 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; despreading one bit requires Nq additions and multiplications, where q is the number of samples per chip. 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

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

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 [23], the probability that a message is successfully jammed is

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

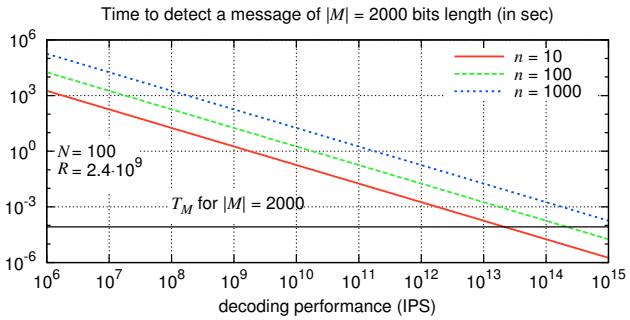


Figure 5: Message detection and decoding performance of the attacker, as function of her computing power. The attacker's detection is based on one code ($k = 1$). Since the acceptable time T_d for the receiver to decode a message is in the order of seconds whereas the time T_M to transmit a message is in the order of ms, an attacker with a decoding power one order of magnitude larger than the one of the slowest receiver can be tolerated.

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

The attacker's message detection performance is depicted in Figure 5; in Section VI, we will compare it to the receiver's message decoding performance.

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.

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 6 displays their decoding performances). We start by analyzing the original UDSSS scheme in the absence of jamming and therefrom 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

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

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, 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

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

The approximation holds if $T_s \ll T_d$, that is, if $s|M|N \ll R$ and $1 \ll nN$. In the same setting, DSSS—where the used spreading code and synchronization are known to the receiver—would achieve a throughput of $\frac{|M|}{T_M} = \frac{R}{N}$. The bottom graph in Figure 5 depicts the receiver's message decoding performance as a function of its computing power (i.e., number of instructions that it can perform per second, IPS). 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. Although the throughput of UDSSS is about one order of magnitude lower than that of DSSS, we need to consider that DSSS cannot be used for broadcast anti-jamming communication. 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 $\leq n_j/(nN)$. 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

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

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 $\frac{1-p_e^{s^*-1}}{1-p_e} \leq$

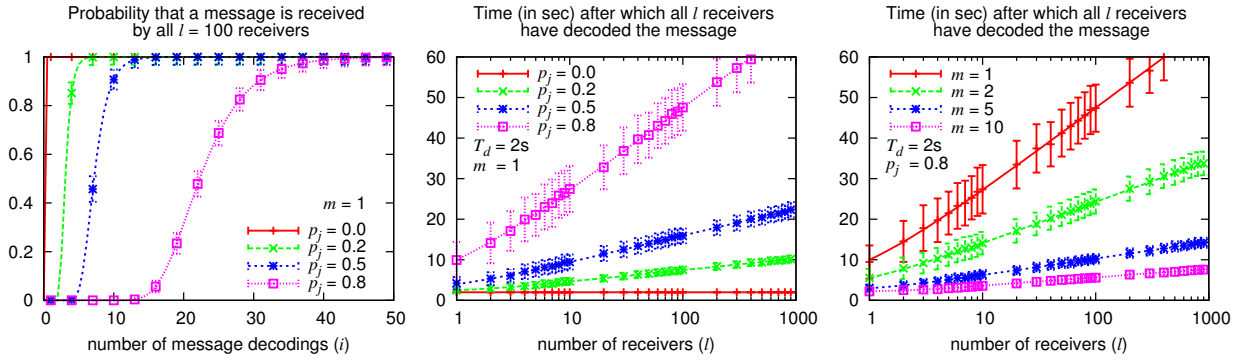


Figure 6: (a) Probability that a UDSSS message has been successfully received by all receivers as a function of the number of message decodings. (b) Expected time to disseminate a message as a function of the number of receivers; the decoding time T_d of the receivers is assumed to be 2 s. (c) Expected time to disseminate a message as a function of the number of parallel message transmissions; the decoding time T_d is assumed to be 2 s. For (a) – (c), the lines show the expected result according to our analytical analyses, the points and σ-confidence intervals display the simulation results.

$\lim_{p_e \to 1} \frac{1-p_e^{s^*-1}}{1-p_e} = s^* - 1$, 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 positioned appropriately, they will encounter the same attacker-caused errors and require the same amount of time to receive the message[4]. 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

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

[4]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.

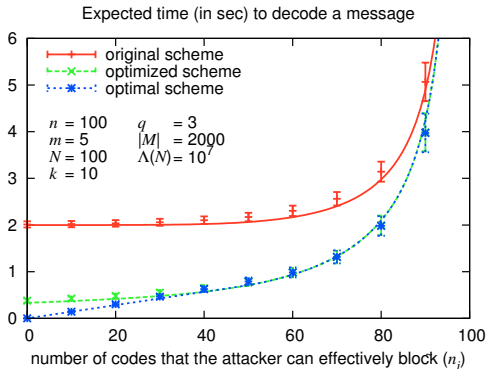


Figure 7: Expected time to disseminate one of the messages from five senders to one receiver. Shown are the original scheme (Equation 3) and the optimization using early termination of the despreading (Equation 5); the optimal scheme uses n^*. We observe that the optimized scheme is close to optimal for $\tilde{n}_j < n/2$ and optimal if $\tilde{n}_j \geq n/2$.

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 6a and 6b. We observe that even for a high jamming probability of 80%, all receivers have received a message with a probability of $\geq$ 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 (1) that selecting a uniform code distribution is optimal and (2) that stopping the decoding process once a valid message was found decreases the message latency. We also show that splitting a large code set into smaller, distinct sets for multiple senders (3) 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 7 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 $p(c_j) = \frac{m}{n}$.*

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

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 8a). 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 8b.

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 and 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.

(a) (b)

Figure 8: UDSSS implementation. Figure (a) shows the experimental hardware setup consisting of a Universal Software Radio Peripheral (USRP) and a Lenovo T61 ThinkPad. Figure (b) depicts a schematic description of our UDSSS sender and receiver application.

In our experiments, we positioned the UDSSS sender and receiver indoors at a distance of about $5\,\mathrm{m}$ 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 9a and 9b 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 $20\,\mathrm{s}$ (for a processing gain of 21 dB and $n = 100$).

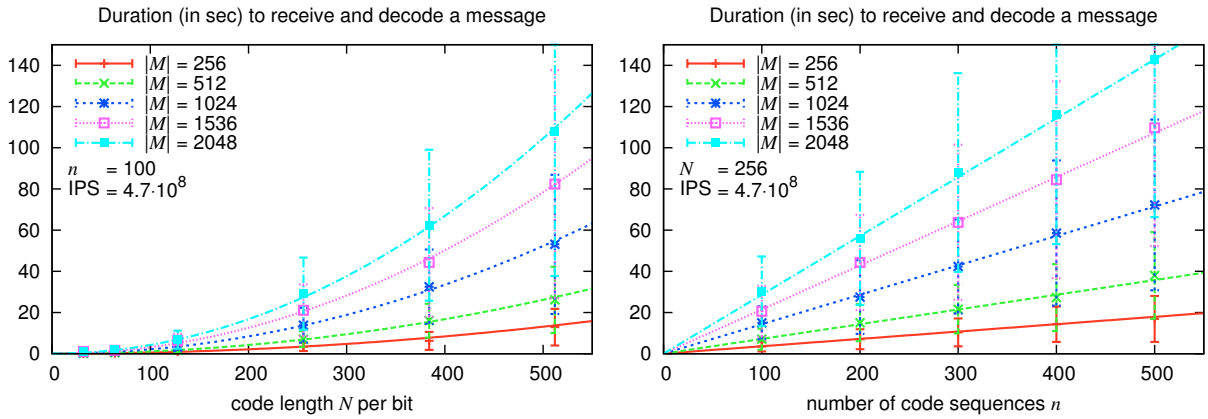
We point out that the main purpose of this USRP/CPU-based system is to demonstrate the feasibility of UDSSS. The achieved decoding times should thus not be considered as performance benchmarks. As the operations to decode a bit can easily be executed in parallel, decoding a bit is typically a single-step operation on hardware-based 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 (Section 6) 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 (and, hence, authentic secret keys between the sender and the receivers cannot be established beforehand).

As mentioned in the Introduction, one example for such a setting is a *central (governmental) authority* that needs to *inform the public about the threat* of an imminent or ongoing (terrorist) attack while minimizing the danger that the attackers jam the alert transmission. Information dissemination in this setting is clearly



(a) (b)

Figure 9: 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; 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 our USRP/CPU-based system is to demonstrate the feasibility of UDSSS. On hardware-based DSSS transceivers, the operations to decode a bit 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.

time-critical, however, being able to distribute the information within seconds to few minutes under jamming is clearly preferred over not being able to disseminate any information at all. 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 [24]) 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 broadcast 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, 20]. E.g., while current civilian implementations using GPS satellite signals [24] 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 easily 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 mes-

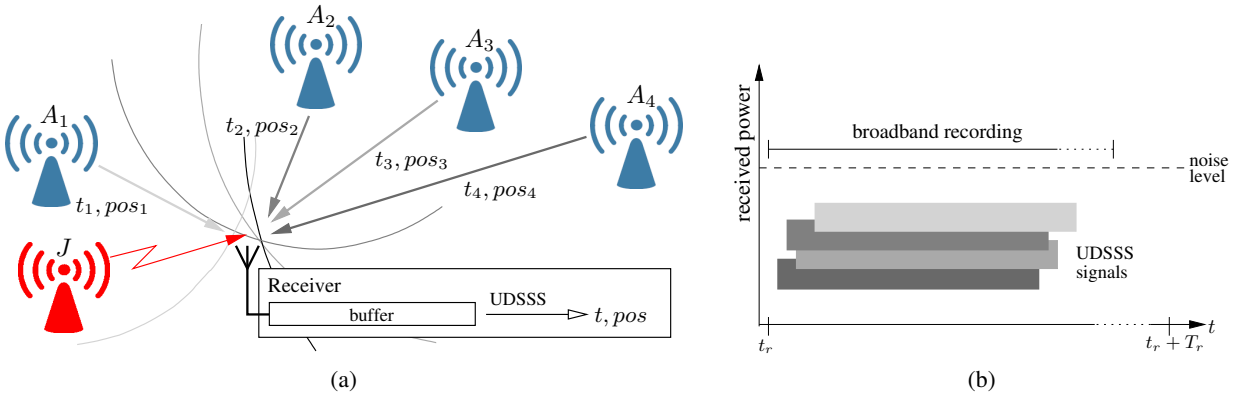


Figure 10: **(a)** Possible application of UDSSS: jamming- (and spoofing-)resistant reception of navigation signals used for positioning (pos) and/or time-synchronization (t). The receiver records the signals of multiple senders which were spread using randomly selected spreading sequences and uses UDSSS 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 (Sec. 5). 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.

sages [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 10a). 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 μs 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 also provides resistance against jamming because the entire navigation signals are sent with (temporarily) unknown code sequences below noise level (see Figure 10b).

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, 19, 26]. Spread-spectrum techniques such as DSSS and FHSS are common jamming countermeasures [2, 19]. 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 [23], pre-loading 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 or modifying it for others [14].

Recent observations [4, 23] identify the shortcoming of non-existing 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) [23] 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 [27], 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. The performance and jamming resistance of our DSSS scheme was evaluated analytically, through a prototype implementation, and by means of simulations for single and multiple receivers. Our evaluation results show that even with our prototype (achieving about 470 MIPS), the expected time to receive and decode a message with UDSSS is well below 20 s (for a processing gain of 21 dB). For a state-of-the-art system with a message decoding time of about 2 s (i.e., about 6000 MIPS), the expected time for a message transfer to a group of 10 receivers takes less than 30 s for a high jamming probability of 80%. We point out that these times are reasonably short, given that with common (key-dependent) anti-jamming techniques the devices would not be able to broadcast jamming-resistant messages at all, and that purpose-built hardware enables to decrease the delays significantly.

References

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

[2] David Adamy. *A first course in electronic warfare*. Artech House, 2001.

[3] Paramvir Bahl and Venkata N. Padmanabhan. RADAR: An In-Building RF-Based User Location and Tracking System. In *Proceedings of the IEEE Conference on Computer Communications (InfoCom)*, volume 2, pages 775–784, 2000.

[4] L. C. Baird, W. L. Bahn, M. D. Collins, M. C. Carlisle, and S. C. Butler. Keyless Jam Resistance. In *Proceedings of the IEEE Information Assurance and Security Workshop*, pages 143–150, June 2007.

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

[6] Jerry Chiang and Yih-Chun Hu. Dynamic jamming mitigation for wireless broadcast networks. In *Proceedings of the IEEE Conference on Computer Communications (InfoCom)*, 2008.

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

[8] Y. Desmedt, R. Safavi-Naini, H. Wang, C. Charnes, and J. Pieprzyk. Broadcast anti-jamming systems. In *Proceedings of the IEEE International Conference on Networks (ICON)*, 1999.

[9] Robin A. Dillard and George M. Dillard. *Detectability of Spread-spectrum Signals*. Artech House Publishers, 1989.

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

[11] Andrea Goldsmith. *Wireless communications*. Cambridge University Press, 2005.

[12] W. Hang, W. Zanji, and G. Jingbo. Performance of DSSS against repeater jamming. In *Proceedings of the IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, pages 858–861, 2006.

[13] International Loran Association. LORAN: LOng Range Aid to Navigation. http://www.loran.org.

[14] Markus G. Kuhn. An asymmetric security mechanism for navigation signals. In *Proceedings of the Information Hiding Workshop*, pages 239–252, 2004.

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

[16] Guevara Lin and Guolong Noubir. On link layer denial of service in data wireless LANs: Research articles. *Wireless Communications & Mobile Computing*, 5(3):273–284, 2005.

[17] B. Natarajan, S. Das, and D. Stevens. An evolutionary approach to designing complex spreading codes for DS-CDMA. *IEEE Transactions on Wireless Communications*, 4(5):2051–2056, 2005.

[18] Navstar Space and Missile Systems Center. Navstar Global Positioning System: Interface Specification IS-GPS-200. http://www.losangeles.af.mil, 2006.

[19] Richard A. Poisel. *Modern Communications Jamming Principles and Techniques*. Artech House Publishers, 2006.

[20] K. B. Rasmussen, S. Čapkun, and M. Čagalj. SecNav: secure broadcast localization and time synchronization in wireless networks. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, pages 310–313, 2007.

[21] Dilip V. Sarwate and Michael B. Pursley. Crosscorrelation properties of pseudo-random and related sequences. In *Proceedings of the IEEE*, volume 68, pages 593–619, May 1980.

[22] Bernard Sklar. *Digital communications: fundamentals and applications*. Prentice-Hall, 2001.

[23] M. Strasser, C. Pöpper, S. Čapkun, and M. Čagalj. Jamming-resistant key establishment using uncoordinated frequency hopping. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 64–78, 2008.

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

[25] Lloyd Welch. Lower bounds on the maximum cross correlation of signals. In *IEEE Transactions on Information Theory*, volume 20, pages 397–399, 1974.

[26] W. Xu, W. Trappe, Y. Zhang, and T. Wood. 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)*, pages 46–57, 2005.

[27] Y. Zhan, Z. Cao, and J. Lu. Spread-spectrum sequence estimation for DSSS signal in non-cooperative communication systems. In *IEE Proceedings Communications Magazine, IEEE*, August 2005.