Enabling Short Fragments for Uncoordinated Spread Spectrum Communication

Naveed Ahmed[1], Christina Pöpper[2], and Srdjan Capkun[3]

[1] DTU Copenhagen, Denmark
naah@dtu.dk
[2] HGI Ruhr-University Bochum, Germany
christina.poepper@rub.de
[3] ETH Zurich, Switzerland
srdjan.capkun@ethz.ch

Abstract. Uncoordinated spread spectrum (USS) protocols have been proposed for anti-jamming communication in wireless settings without shared secrets. The existing USS protocols assume that fragments of hundreds of bits can be transmitted on different channels in order to identify fragments that belong to the same message. However, such long transmissions are susceptible to reactive jamming. To address this problem, we present a protocol that allows the use of short fragments of a few bits only. This makes our scheme resilient to a large class of reactive jammers. We prove that reassembling the fragmented message is not only feasible but also efficient: it can be completed in polynomial time in the size of the message, even if the jammer is computationally resourceful. We demonstrate the protocol efficiency by simulating the reassembly process at the link layer under different design parameters.

Keywords: Anti-jamming, Spread-spectrum Communication, Wireless Security.

1 Introduction

The primary countermeasure against jamming attacks on wireless communication is spread spectrum (SS) communication. Traditional (coordinated) SS communication between two parties requires shared secrets, however, establishing the secret key is a challenge in itself [1]. If two parties are unknown to each other, such as in wireless ad-hoc communication, emergency alert broadcast, or the dissemination of navigation signals [2], pre-sharing secrets is not feasible. Jamming-resistant key establishment is not only a bootstrapping problem but it reoccurs during re-keying if the old keys have been compromised.

A few years ago, a technique for uncoordinated spread spectrum (USS) communication, which does not require pre-shared secrets, was proposed at the US Air Force Academy [3]. Since then the interest in USS has grown, both for civilian [1, 4, 5] and military applications [6]. In effect, a USS transmitter transmits a long message as a sequence of shorter, fixed-size encoded fragments (frames) on randomly selected channels. A channel here corresponds to a frequency channel in frequency hopping spread spectrum (FHSS) or a chip code (or sequence of chip codes) in direct sequence spread spectrum (DSSS).

M. Kutyłowski and J. Vaidya (Eds.): ESORICS 2014, Part I, LNCS 8712, pp. 488–507, 2014.

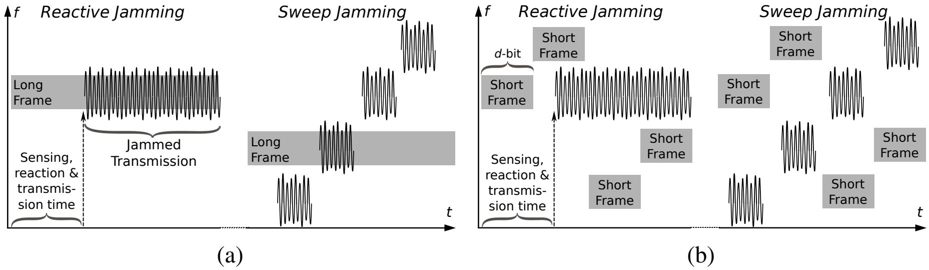


Fig. 1. Long and short frame transmissions. (a) Long frames are an easy target for jammers. Reactive jammers will have sufficient time to sense an ongoing transmission and then react to jam the used channel. Sweep (non-reactive) jammers have a high probability of hitting the channel where a long frame is transmitted. (b) Short frames reduce the risk of successful jamming attacks for both reactive and non-reactive sweep jammers.

The frame size is a key parameter of USS communication because it determines how fast a transmitter can switch the transmission channel. Clearly, if the frames are long then the respective channel will remain active for a long time, which makes it easy for a reactive adversary to locate and jam the channel and which results in high jamming probabilities for sweep jammers, as illustrated in Fig. 1-(a). If the switching frequency is high enough (see Fig. 1-(b)) then the adversary does not get enough time to react.

The feasibility of real-time, reactive radio jamming has recently been demonstrated using software-defined radio equipment [7, 8] with reaction times on the order of few symbol durations for 802.15.4 communication. Although error correction schemes can be used to repair some errors, these schemes are not effective against reactive jammers that can jam the channel if the frames are long. Thus, a USS protocol that supports short frames is highly desirable. Existing USS protocols, however, depend on long frames in order to transmit "linking information" that is used to identify parts that belong to the same message (we further elaborate on this in § 3 and § 7). This identification problem exists due to trivial pollution attacks at the link layer, in which a large number of well-designed fake frames are broadcasted to overburden the reassembly process at a legitimate receiver.

In this paper, we challenge the current assumption that long frames are indeed a necessary requirement for USS schemes. The idea of our solution is based on two insights. First, the payload and the message link do not need to be in the same fragment – instead they can be decoupled. This allows to independently transmit the payload and the link as shorter fragments. Second, if the function that computes a link is secure against a computationally unbounded adversary, then the size of the link can be reduced to a few bits. Using these insights, our protocol is the *first scheme for USS communication that allows for short fragments–down to a few bits.*

In more details, consider a reactive adversary who can jam frames longer than d-bit. To circumvent this reactive jamming, we disassemble a long message into m fragments of d-bit each, which are cryptographically linkable. As we will show, our protocol enables a USS receiver to reassemble the original message in a time that is polynomial in

m, provided $z(z+1) < 2^d$, where z is the number of fake fragments that an adversary can transmit in parallel to each legitimate fragment.

The rest of the paper is arranged as follows. In § 2, we clarify the problem and define the system and adversary models. In § 3, we give an overview of our proposal. We present our proposed solution and its properties in § 4 and prove its security in § 5. In § 6, we describe various performance results that we obtained by a simulation. In § 7, we describe related work. Finally, in § 8, we conclude our work.

2 Problem Statement

Problem Formulation: A number of message fragmentation schemes [1,9–11] for USS communication were proposed (see § 7). The current approaches face a dilemma.

First, the required fragment size is too long to be practical against reactive jamming at the physical layer. That is, the proposed schemes apply linking techniques that not only require embedded linking information within the fragment but also the linking information must be hundred bits long for adequate cryptographic security.

Second, shortening the fragments (to provide more resistance to reactive jammers) conflicts with the schemes' very resistance against computationally powerful adversaries. For instance, the minimum fragment size of hash-based solutions [9] is mainly given by the length of the hash values used in the frame encoding; if the length of the hash values is reduced to make the scheme resistant to faster jammers then the hash function may no longer be second pre-image resistant. A computationally powerful adversary will then be able to break the linking function and introduce exponential complexity in the reassembly process—with the effect of a DoS-attack.

The goal of this work is to identify a way that allows to significantly reduce the frame size compared to prior proposals.

System Model: We consider an environment in which the communication bandwidth is given by a set of channels $\mathcal{C}$, where $|\mathcal{C}| = n$. For instance, $c \in \mathcal{C}$ can be a frequency channel or a spreading code (or a short sequence of chippings codes) that encodes d bits. We consider ad-hoc communication between pairs of devices or from a transmitter $\mathcal{T}$ to unknown receivers in its transmission range (broadcast). $\mathcal{T}$ transmits d-bit fragments, encoded in frames, on randomly selected channels. We do not assume any shared secrets in the system and all protocol specifications are public and known to the receivers.

A receiver $\mathcal{R}$ is located in the transmission range of $\mathcal{T}$ and can receive on all or a subset of channels in parallel (broadband or partial-band receiver). $\mathcal{R}$ does not need to be time synchronized with $\mathcal{T}$ at the fragment level but is assumed to be in reception mode while the sender is transmitting. There can be a large number of receivers, but we do not assume any inter-receiver communication. We do not consider point-to-point communication, i. e., we do not require headers with physical address information.

Adversary Model: The aim of an adversary $\mathcal{A}$ is to prevent $\mathcal{R}$ from reassembling a legitimate message sent by $\mathcal{T}$. The adversary can mount attacks at two different levels.

First, at the radio level, $\mathcal{A}$ can try to jam $\mathcal{T}$'s transmission. We assume that, due to limitations of radio equipments (e. g., required time for sensing and synthesizing the frequency of the carrier wave), $\mathcal{A}$ cannot deterministically jam transmissions of fragments of a few bits. Similar to conventional SS communication, $\mathcal{A}$ can jam a frame probabilistically by guessing as was, e. g., analyzed for longer frames in [1]. Therefore, as typical for conventional SS, we assume that $\mathcal{A}$ can only jam a limited portion of the available bandwidth. We note that number of channels where the attacker can jam may also be larger than the number of receiving channels at the receiver (which increases reception time accordingly).

Second, at a computational level, an all-powerful $\mathcal{A}$ tries to exploit our protocol at the data link layer. To this end, $\mathcal{A}$ transmits fake fragments to make it infeasible for $\mathcal{R}$ to identify legitimate fragments. The adversary may be located within the transmission range of $\mathcal{T}$ and her fake fragments can be a function of the legitimate fragments.

We synthesize the above discussion in two assumptions:

Assumption 1 (Minimum Reaction Time). *The reaction time of $\mathcal{A}$, i. e., the time required to sense an ongoing transmission on a channel and then jam that channel, is longer than the time required to transmit a frame containing a d-bit fragment ($d \geq 1$).*

In short, we assume that communication using conventional FHSS with fragments of d bits would resist the considered reactive attacker $\mathcal{A}$ (but cannot be used due to the lack of shared secrets). The reaction time depends on $\mathcal{A}$'s distance to $\mathcal{T}$ and to $\mathcal{R}$ and on the response times of $\mathcal{A}$'s radio equipments; channel switching can easily account for tens of microseconds [7, 12] and channel sensing by energy detection may be up to an order of milliseconds [13]. Given the bit rates of common wireless standards for comparison (e. g., 11 Mbit/s for 802.11b or 5-15 Mbit/s in 3G-UMTS) and the resulting bit duration of around one μs, it is reasonable to assume that the combined attack time is longer than the frame transmission time for a d-bit fragment when d is small.

Assumption 2 (z-channel Adversary). *$\mathcal{A}$'s transmission power is limited to $z < n$ channels (on which $\mathcal{A}$ can transmit in parallel) such that $z(z+1) < 2^d$.*

Assumption 2 limits the bandwidth on which the adversary can transmit; e. g., our proposed scheme will work efficiently for $d = 5$-bit fragments if the attacker transmits on up to $z = 4$ channels in parallel. Note that unlimited computation power cannot be used to overcome the limitations of the communication hardware.

The effect of non-reactive jamming strategies (e.g., sweep jamming) on $\mathcal{R}$ is the same as randomly corrupting some of the fragments, because $\mathcal{T}$ sends the fragments on randomly selected channels; for UFH, random selection is the optimum strategy [9]. In principle, error-correcting schemes [14] can be used to counter random errors. Traditional error correction increases the frame size, though. Therefore, we propose a scheme based on the repetition of fragments in order to tolerate random errors.

Clearly, $\mathcal{A}$'s attacks are not effective at the radio level given the design parameters, z and d, correctly capture the capabilities of $\mathcal{A}$'s radio equipment. Next, we introduce the details of our protocol and show that $\mathcal{A}$, even with infinite computational power, cannot devise a jamming strategy that exploits our protocol at the data link layer.

3 Solution Overview

In this section, we provide an overview of our protocol, which we call the collision detection protocol (CDP or CD protocol). In the CDP, a message $\mathbf{T}_{\mathrm{m}}^{\mathcal{T}}$ to be broadcasted

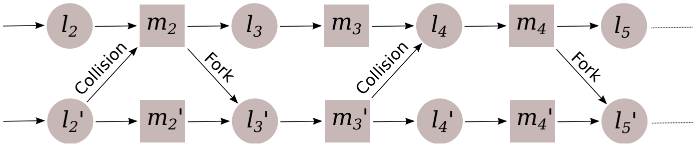


Fig. 2. Collisions and forks among paths significantly increase the number of paths that $\mathcal{R}$ must reassemble. Although forks and collisions are unavoidable for short fragments, collisions can be detected, which suffices to efficiently decode a legitimate message.

by transmitter $\mathcal{T}$ is assembled as a list of m message fragments (M-fragments), each of which is d bits long: $\mathbf{T}_{\mathtt{m}}^{\mathcal{T}} = \mathtt{m}_1, \ldots, \mathtt{m}_m$. We define $\mathbf{T}_{\mathtt{m}}^{\mathcal{T}}[i \to j] = \mathtt{m}_i, \ldots, \mathtt{m}_j$, where $1 \leq i \leq j \leq m$. Each M-fragment is sent on a randomly selected SS channel. Since an adversary can transmit fake fragments, we link the M-fragments together, so that the CDP receiver $\mathcal{R}$ can reassemble the original message. For this purpose, we use a link certificate $\mathbf{T}_{\mathtt{l}}^{\mathcal{T}}$, which consists of m linking fragments (L-fragments) each of size d bits. The i-th L-fragment is computed from the message fragments using a CDP linking function: $\mathtt{l}_i = link(\mathbf{T}_{\mathtt{m}}^{\mathcal{T}}[1 \to i])$; we specify this function in § 4.1. We then interleave the L-fragments and the M-fragments, to obtain a transmission schedule: $\mathbf{T}^{\mathcal{T}} = \mathtt{m}_1, \mathtt{l}_2, \mathtt{m}_2, \ldots, \mathtt{l}_m, \mathtt{m}_m$. The schedule $\mathbf{T}^{\mathcal{T}}$ is followed by $\mathcal{T}$ who sends fragments sequentially on randomly selected channels.

Since an L-fragment is a function of prior M-fragments, $\mathcal{R}$ can perform an online verification of the incoming L-fragments. A *path* is a plausible reconstruction of the transmission schedule $\mathbf{T}^{\mathcal{T}}$ and it consists of a list of interleaved L-fragments and M-fragments upon which the linking function can be verified. We say that $\mathcal{R}$ is tracking a path at an index i if the online verification of the i-th L-fragment $\mathtt{l}_i$ succeeds for the path $\mathtt{m}_1 \to \mathtt{l}_2 \to \cdots \to \mathtt{l}_i \to \mathtt{m}_i$. A path is complete, when the tracking terminates successfully with the verification of the m-th L-fragment.

A z-channel adversary can simultaneously transmit on z channels, and thus $\mathcal{R}$ may receive $z+1$ fragments at each time instant. In this case, finding the path of the original message (consisting of m fragments) involves searching among $(z+1)^m$ plausible paths, which can be infeasible, e. g., with $m = 128$ and $z = 2$. In this paper, we show that if an appropriate linking function is used then the number of paths are significantly reduced, which enables $\mathcal{R}$ to efficiently reassemble a legitimate message. Ideally, if $z+1$ fragments are received per time unit then no more than $z+1$ paths should be tracked by $\mathcal{R}$. In reality, however, the number of trackable paths could be more than $z+1$ due to a combination of *collisions* and *forks*. We say that a collision occurs when two paths merge together (at an L- or M-fragment) and a fork occurs when a path splits into different paths, as illustrated in Fig. 2. Ideally, one would expect that the linking function can resist both collisions and forks in order to limit the number of paths, however, this may be impossible to achieve; e. g., if $d = 4$ then the probability that two random paths will have the same subsequent L-fragment is at least 2^{-4}.

Fortunately, the following two insights lead to a technique that reduces the number of search paths for short fragments. First, by preventing collisions only, we can avoid an exponential number of plausible paths. If there are no collisions then $\mathcal{R}$ never tracks more than $z+1$ paths when receiving $z+1$ fragments per time unit. Without collisions, the search space for tracking can be visualized as a tree-shaped structure (due to the forks), which will have less than $z + 1$ complete paths. Second, a linking function that only takes the current and the previous fragment into account (such as a hash chain in [1]) is not sufficient to detect collisions, because if a collision is not detected in the first link that arrives after the collision then the collision will remain undetected in the subsequent tracking. Hence, the number of paths would be high for such a function.

We propose a linking function that takes all (or a large number of) the previous fragments into account, which enables to detect collisions in the subsequent tracking (hence the name *collision detection protocol*). Although an adversary may be able to create collisions between her path and the legitimate path, these collisions can be detected once the tracking progresses with the arrival of more fragments. Towards the end of the legitimate path, collisions are less likely to be detected, but this does not exponentially increase the message reassembly time, since only few fragments are concerned.

4 Collision Detection Protocol

We next present our proposed scheme on the sender (§ 4.1) and receiver (§ 4.2-4.3) side.

4.1 Sender Side: Message Transmission

Let the message $\mathbf{T}_{\mathrm{m}}^{\mathcal{T}}$ to be sent by $\mathcal{T}$ come from a uniformly distributed encoding of the message payload, making $\mathbf{T}_{\mathrm{m}}^{\mathcal{T}}$ unpredictable for the adversary, e.g., $\mathbf{T}_{\mathrm{m}}^{\mathcal{T}}$ can be computed by a symmetric encryption function: $\mathbf{T}_{\mathrm{m}}^{\mathcal{T}} \leftarrow \mathcal{E}_K\big(N_{\mathcal{T}}, \mathcal{M}, \mathcal{S}_{Sk_{\mathcal{T}}}(\mathcal{M})\big)$. Here, K is a publicly known key, $N_{\mathcal{T}}$ is a secretly generated nonce used to randomize $\mathbf{T}_{\mathrm{m}}^{\mathcal{T}}$, $\mathcal{M}$ is the payload, and $\mathcal{S}_{Sk_{\mathcal{T}}}(\mathcal{M})$ is the signature[1] computed on $\mathcal{M}$ using $\mathcal{T}$'s private key $Sk_{\mathcal{T}}$. The payload may contain a time-stamp to provide freshness of $\mathbf{T}_{\mathrm{m}}^{\mathcal{T}}$.

Assuming the encryption function has pseudo-random properties [15], the output $\mathbf{T}_{\mathrm{m}}^{\mathcal{T}}$ is uniformly distributed for the adversary until $\mathbf{T}_{\mathrm{m}}^{\mathcal{T}}$ is transmitted. This randomization is required in order to make the value of a fragment unpredictable for the adversary before the fragment is actually transmitted (details will follow later). The signature $\mathcal{S}_{Sk_{\mathcal{T}}}$ is required so that $\mathcal{R}$ can distinguish a legitimate message from fake messages.

As described in § 3, $\mathcal{T}$ assembles $\mathbf{T}_{\mathrm{m}}^{\mathcal{T}}$ as a list of M-fragments, computes the link certificate $\mathbf{T}_{\mathrm{l}}^{\mathcal{T}}$, and interleaves it with $\mathbf{T}_{\mathrm{m}}^{\mathcal{T}}$: $\mathbf{T}^{\mathcal{T}} = \big[\mathbf{T}_{\mathrm{l}}^{\mathcal{T}}[i], \mathbf{T}_{\mathrm{m}}^{\mathcal{T}}[i] : 1 \leq i \leq m\big]$. The i-th fragment $\mathbf{T}^{\mathcal{T}}[i]$ is transmitted at the i-th time instant t_i on randomly selected channels.

Resilience to Fragment Loss. To tolerate possible fragment loss (due to, e.g., adversarial jamming or channel noise), the front-end of $\mathcal{T}$ operates with a repetition factor of ρ: each fragment $\mathbf{T}^{\mathcal{T}}[i]$ is repeated ρ times between time instants t_i and $t_i + \Delta T$ on randomly selected channels, as illustrated in Fig. 3. Here, ΔT is the window size in which $\mathcal{R}$ senses the incoming signals for data. The period size ΔT is fixed but $\mathcal{R}$ may not know the start and end of a period.

[1] Existing digital signature schemes (DSS) are only secure against computationally bounded adversaries, but this does not affect the security of the link layer, which is responsible for re-assembling the message from received fragments.

Properties of the Linking Function. The core of our scheme is the linking of message fragments using a linking function: $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^d$. This function is computed on the current fragment and all prior fragments. We require the following property to hold for $\mathcal{H}(\cdot)$ in order to efficiently identify the legitimate message $\mathcal{T}_{\mathrm{m}}$ at $\mathcal{R}$.

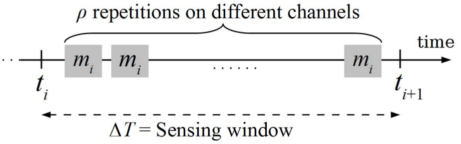

ρ repetitions on different channels
time
m_i
m_i
m_i
t_i
t_{i+1}
ΔT = Sensing window

Fig. 3. Repetitions enable $\mathcal{R}$ to tolerate fragment losses, as common for USS techniques

Property 1. *Let x_l, x_l' and x_h be three bit strings such that $x_l \neq x_l'$, $|x_l| = |x_l'| \leq md$, and $|x_h| = d$. Let x_l be known to the adversary, x_l' be chosen by the adversary, and x_h comes from the uniform distribution. The function $\mathcal{H}(\cdot)$ is a* linking function *if $\mathcal{H}(x_l) = \mathcal{H}(x_l') \Rightarrow \mathcal{H}(x_l, x_h) = \mathcal{H}(x_l', x_h)$ only holds with probability 2^{-d}.*

As an intuition, x_h stands for the current fragment of a transmitted message, which is connected to all prior fragments x_l using the link $\mathcal{H}(x_l, x_h)$. Similarly, x_l' stands for all prior fragments of an adversarial message. $\mathcal{A}$ may well compute x_l' such that $\mathcal{H}(x_l) = \mathcal{H}(x_l')$ holds for $x_l' \neq x_l$, which results in a collision of the two search paths in the decoding process on $\mathcal{R}$. This collision, however, is likely to be detected during the consideration of the next fragment x_h, because the corresponding links, $\mathcal{H}(x_l, x_h)$ and $\mathcal{H}(x_l', x_h)$, can only be equal with a low probability (2^{-d}). This probability is further decreased when more fragments arrive: Property 1 not only specifies the collision detection probability with the current fragment x_h, but also with all subsequent fragments of the message.

Instantiation of the Linking Function. A simple instantiation of $\mathcal{H}(\cdot)$ is a cryptographic hash function, such as SHA-256. For this purpose, one can treat ',' as string concatenation, pad the resultant string to make it compliant to the hash function, and truncate the digest to d-bit. $\mathcal{H}(\cdot)$ can also be constructed using a set of random tables (especially if d is a few bit long), or using the truncated output of an encryption or signature function. The bitwise XOR function however, does *not* satisfy Property 1 and collisions will propagate (i. e., remain undetected) with probability 1.

If SHA-256 type hash function is used, then new fragments must be prepended to the existing string of fragments. We explain this requirement in the following. Due to the Merkle-Damgård construction, SHA-256 computes the digest from an input iteratively, by taking 512-bit at a time. Let s_1 and s_2 be two bit strings that are multiples of 512-bit[2] with $s_1 \neq s_2$. Let s_3 be another bit string of arbitrary length. If a collision occurs, namely $\mathcal{H}(s_1) = \mathcal{H}(s_2)$, then we also get another collision $\mathcal{H}(s_1, s_3) = \mathcal{H}(s_2, s_3)$. Therefore, to avoid this problem and to cause re-computation of the whole chain of compression functions inside SHA-256, new fragments must be prepended.

Note that Property 1 does not imply one-wayness, second pre-image resistance, or conventional collision-resistance of $\mathcal{H}(\cdot)$. This is important because, for the link layer, we consider a computationally unbounded adversary, for whom these assumptions may not hold. With a small value of d, such as 4 bits, the standard assumptions of a hash function are not even realistic for a computationally limited adversary.

[2] The internal "chunk size" of SHA-256 and SHA-512 is 512 bit.

If the fragments are very short and the adversary can introduce a large number of parallel fragments, a single link certificate may not be enough to detect collisions efficiently. This problem can be addressed by using $\alpha > 1$ link certificates; we call α *amplification factor*. With $\alpha = 2$, the encoding becomes: $\mathtt{m}_1, \mathtt{l}_2, \mathtt{l}'_2, \mathtt{m}_2, \ldots, \mathtt{m}_m$. Each additional link certificate uses a different linking function. When we describe the receiver side, we assume $\alpha = 1$, but the results can be extended for $\alpha > 1$ and we investigate its impact on the performance of the CDP in § 6.

4.2 Receiver Side: Reception of Fragments

Handling of Fragment Loss and Fake Fragments. The CDP tolerates a situation in which up to a certain threshold of the transmitted fragments—typically half of them—get corrupted or lost due to adversarial jamming or channel characteristics. To achieve time synchronization, $\mathcal{R}$ uses a sliding-window technique to get alignment for the ΔT window, which can be achieved by repeating the decoding process $\rho/2$ times, possibly in parallel. To illustrate the decoding process, we consider two special cases of a receiver antenna. In both cases, the purpose is to make sure that a z-channel adversary can only make $\mathcal{R}$ accept, at most, one value per adversarial channel.

First, we consider a broadband antenna, namely $\mathcal{R}$ can receive data on all channels in parallel. In this case, the top $z + 1$ values that occur the most in ΔT are marked as received. Clearly, a z-channel adversary cannot make $\mathcal{R}$ to ignore the legitimate transmission, because for doing so the adversary would need to transmit $z + 1$ different values, such that each of them occurs at least ρ times.

In the second case, the receiver antenna is narrow-band, namely $\mathcal{R}$ can only receive on one or a few channels. In this case, our solution is very similar to the technique from [2, 9]: $\mathcal{R}$ listens on randomly selected channels. The probability that $\mathcal{R}$ listens on the correct channel(s) (where a fragment is being transmitted) is n'/n, where n is the total number of channels and n' is the number of channels where $\mathcal{R}$ can listen on in parallel. Since a fragment is repeated ρ times, $\mathcal{R}$ is expected to receive $\rho n'/n$ fragments. With an argument similar to the first case, the top $z + 1$ values occurring the most are marked as received. To make this scheme work, we further require $\rho \gg n/n'$ such that $\rho n'/n$ is sufficiently large (such as 32).

Receiver's Search Space. The described handling of arriving fragments during ΔT results in at most $z + 1$ fragments. In this way, $\mathcal{R}$ gets up to $z + 1$ fragments for each t_i. We use capital letters to denote variables corresponding to received fragments, e. g., M_1 where the first M-fragment is expected. Note that $\mathcal{R}$ does not have a-priori knowledge about the type of fragments it receives. The set of *received* fragments between t_i and $t_i + \Delta T$ is called the i-th *reception set*, denoted by σ_i. For a z-channel adversary, the size of a reception set is $|\sigma_i| \leq z + 1$, in which up to z fragments can be from the adversary. In the following, we denote the adversary's transmission schedule as $\mathbf{T}^{\mathcal{A}}$, where $\mathbf{T}^{\mathcal{A}}[i]$ is the set of M/L-fragments transmitted by $\mathcal{A}$ in the i-th time period.

To achieve fragment alignment, $\mathcal{R}$ makes a random guess to mark odd time instants for the reception of L-fragments and even time instants for the reception of M-fragments. The guess is correct if it matches the position of legitimate fragments. If the decoding fails, $\mathcal{R}$ switches the role of even and odd fragments. In this way, two

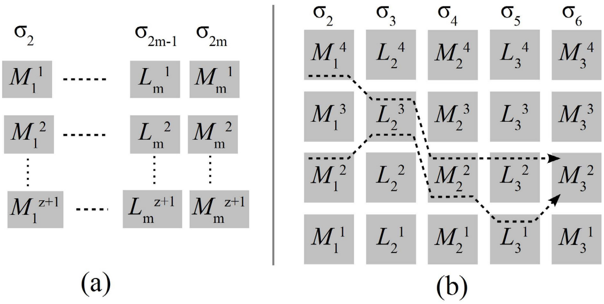


Fig. 4. (a) The search space $\mathcal{S}$ for a CDP receiver $\mathcal{R}$ (b) Two example paths: $\Pi_3 = M_1^2, L_2^3, M_2^2, L_3^1, M_3^2$ and $\Pi_3' = M_1^4, L_2^3, M_2^2, L_3^2, M_3^2$

decoding attempts are enough to achieve correct alignment. All received fragments are arranged in a search space:

Definition 1 (Search Space). *The search space is $\mathcal{S} = [\sigma_1, \ldots, \sigma_{2m}]$, where $\sigma_{2i} = \mathbf{T}_m^{\mathcal{T}}[i] \cup \mathbf{T}_m^{\mathcal{A}}[i]$, and $\sigma_{2i-1} = \mathbf{T}_l^{\mathcal{T}}[i] \cup \mathbf{T}_l^{\mathcal{A}}[i]$, for $1 \leq i \leq m$.*

As illustrated in Fig. 4-(a), reception sets on each odd index correspond to L-fragments and sets on each even index correspond to M-fragments. Certainly, an adversary does not need to abide by the rule and can transmit an L-fragment where an M-fragment is expected (and vice versa). This, however, does not help her because the linking function that connects L- and M-fragments is not symmetric, namely if $\mathtt{l}_i = \mathit{link}(\mathtt{m}_1, ..., \mathtt{m}_i)$ holds then $\mathtt{m}_i \neq \mathit{link}(\mathtt{l}_2, ..., \mathtt{l}_i)$.

4.3 Receiver Side: Decoding Algorithm

The receiver's goal is to efficiently identify the fragments of the legitimate transmission from the search space. For this purpose, we next introduce the notion of a *path*:

Definition 2 (Path). *Let $L_i \in \sigma_{2i-1}$ and $M_i \in \sigma_{2i}$. A path of length j is $\Pi_j = M_1, L_2, \ldots, L_j, M_j$, which is a sequence of j interleaved M-fragments and L-fragments, such that all the linking functions in Π_j can be verified.*

Two example paths are shown in Fig. 4-(b), one being $\Pi_3 = M_1^2, L_2^3, M_2^2, L_3^1, M_3^2$. The path Π_3 is tracked by verifying the relations $L_2^3 = \mathcal{H}(M_1^2, M_2^2)$ and $L_3^1 = \mathcal{H}(M_1^2, M_2^2, M_3^2)$. A complete path, Π_m, represents an entire CDP transmission, which is then considered as a candidate for the legitimate transmission $\mathbf{T}^{\mathcal{T}}$. A candidate message on which the signature verification succeeds represents $\mathbf{T}^{\mathcal{T}}$.

Algorithm. The CDP decoding algorithm—denoted by $search(\mathcal{S}, Pk_{\mathcal{T}})$—reassembles the legitimate message from all received fragments. Its inputs are the search space $\mathcal{S}$ and the sender's public key $Pk_{\mathcal{T}}$. The algorithm is based on a recursive depth-first search and terminates within a time polynomial in m and z, which we will prove in § 5.

Each of the M-fragments of $\mathcal{S}$ is associated with a visited-counter (v-counter), initialized to zero. Let θ be a constant representing the threshold (maximum) of the v-counters. Its value will be determined in § 5 in the security analysis.

The function $search(\cdot)$ consists of the following steps:

1. By exhaustive search, find the *roots* of all paths in $\mathcal{S}$. A root is a path of type $\Pi_2 = M_1, L_2, M_2$, where $L_2 = \mathcal{H}(M_1, M_2)$ for $M_1 \in \sigma_2, L_2 \in \sigma_3$, and $M_2 \in \sigma_4$.
2. Start a depth-first search (*dfs*) from each root path Π_2 found in Step 1 by calling a recursive function $b = \mathit{dfs}(\Pi_2)$ (defined below). If $b = 1$ then the legitimate message has been found, so terminate $search(\cdot)$ successfully. If $b = 0$, repeat Step 2 with a new root.
3. Terminate $search(\cdot)$ with an error signal.

The recursive function $b = \mathit{dfs}(\Pi_i)$ is defined as follows:

1. From the path Π_i of length i, compute a new *extended* path $\Pi_{i+1} = \Pi_i$, L_{i+1}, M_{i+1}, where $L_{i+1} = \mathcal{H}(M_1, \ldots, M_{i+1})$ for $L_{i+1} \in \sigma_{2i+1}$, and $M_{i+1} \in \sigma_{2i+2}$. If no such new pair (L_{i+1}, M_{i+1}) can be found then return 0.
2. If the v-counter associated with the new M-fragment M_{i+1} is equal to θ then backtrack to the path Π_i, i. e., go back to Step 1.
3. If the v-counter of M_{i+1} is less than θ then increment the v-counter. If $i + 1 = m$ go to Step 4; otherwise make a recursive call to compute b: $b = \mathit{dfs}(\Pi_{i+1})$. If the returned value b is 0 (indicating a failure to find the legitimate message) then backtrack, i. e., go back to Step 1. If $b = 1$ then return 1.
4. The path is complete. Extract the message from path Π_m. Verify the signature of the message. If the signature verification fails then backtrack to Step 1. If the signature verification succeeds (meaning that this is a legitimate path) then return 1.

Running Time. We derive an upper bound on the running time of the decoding algorithm $search(\cdot)$. Let T_l be the time to compute the hash function $\mathcal{H}(\cdot)$. In $\mathcal{S}$, an upper bound on the number of root paths is $(z + 1)^2$. Each root path requires one computation of the hash function. Therefore, in Step 1 of $search(\cdot)$, the upper bound on the time to compute all root paths is $(z + 1)^2 T_l$.

For Step 2 of $search(\cdot)$, each M-fragment in $\mathcal{S}$ is associated with a v-counter that is upper-bounded by θ. Therefore, each M-fragment can cause at most θ computations of the hash function. The total number of M-fragments in $\mathcal{S}$ is $m(z + 1)$. Hence, the time to compute Step 2 of $search(\cdot)$ is $m\theta(z + 1)T_l$. Due to our repetition scheme (Fig. 3), the decoding process may need to be repeated ρ times. An upper bound on the running time of the algorithm is, therefore, as follows:

$$\tau_r < \rho\big(m\theta(z + 1) + (z + 1)^2\big)T_l. \tag{1}$$

Since the hash function is efficiently computable, T_l represents a polynomial time. If θ is polynomial in m then the running time of the decoding algorithm, τ_r, has an upper bound that is polynomial in z and m, as given by Eq. 1.

Clearly, if we do not set a threshold, i. e., $\theta = \infty$, then the decoding algorithm reduces to an exhaustive search, which may require an exponential amount of time. On the other hand, if the number of fake paths is small then θ can be set to a small number. In the next section, we show that there is indeed a limit on the number of fake paths, which allows us to determine the value of θ, thus guaranteeing that the legitimate message can be reassembled efficiently.

5 Security Analysis

We now show that our protocol cannot be successfully attacked on the link layer.

5.1 Definitions

Security of the CDP is defined as adversary's inability to jam a legitimate transmission.

Definition 3 (Security of CDP). *The CDP protocol is secure if a z-channel adversary, under Assumptions 1-2, cannot prevent the CDP receiver $\mathcal{R}$ from receiving and reassembling a legitimate message, consisting of m fragments, within an amount of time that is polynomial in m and z.*

As described earlier, with an appropriate value of d, radio level jamming by $\mathcal{A}$ can be prevented. At the link layer, however, $\mathcal{A}$ may prevent $\mathcal{R}$ from reassembling a legitimate message. To show the link layer security, namely the efficiency of reassembly process, the notion of collision is important, and in the following we formally define this notion. Let $\Pi_{i \to j}$, with $i < j$, denote a partial path from L_i, M_i to L_j, M_j.

Definition 4 (Collision). *Consider two paths Π_j and Π'_j in a search space. We can write the two paths as $\Pi_j = \Pi_i, \Pi_{i+1 \to j}$ and $\Pi'_j = \Pi'_i, \Pi'_{i+1 \to j}$, for $i < j$. The path Π'_j is said to generate a (i, Π_j)*-collision *in Π_j if $\Pi_{i+1 \to j} \overset{m}{=} \Pi'_{i+1 \to j}$ and $\Pi_i \overset{m}{\neq} \Pi'_i$, where $\overset{m}{=}$ means that the corresponding M-fragments in two paths are equal.*

For example, in Fig. 4, the two paths are $\Pi_3 = M_1^2, L_2^3, M_2^2, L_3^1, M_3^2$ and $\Pi'_3 = M_1^4, L_2^3, M_2^2, L_3^2, M_3^2$. The path Π'_3 creates a $(1, \Pi_3)$-collision in Π_3, and the path Π_3 creates a $(1, \Pi'_3)$-collision in Π'_3. From Def. 4 it is clear that a (i, Π_j)-collision implies (i, Π_{i+1})-, …, (i, Π_{j-1})-collisions, e. g., in Fig. 4 the $(1, \Pi_3)$-collision implies the $(1, \Pi_2)$-collision. If one of the (i, Π_{i+1})-, …, (i, Π_{j-1})-collisions does not occur then a (i, Π_j)-collision cannot occur. This fact is later used in the security proof.

In Def. 4, a (i, Π_j)-collision can be generated due to an adversarial strategy or purely by chance. When the subsequent pair of fragments are added to the path, i. e., Π_j grows to Π_{j+1}, the (i, Π_j)-collision can only propagate to the (i, Π_{j+1})-collision with a probability that is negligible in d. This happens due to Property 1 of our linking function $\mathcal{H}(\cdot)$, and we formally prove this fact. The low probability of collision propagation is exemplified below.

Example 1. *Consider a legitimate path, $\Pi_2 = M_1, L_2, M_2$, and a 1-channel adversary who generates a $(1, \Pi_2)$-collision. For this purpose, she computes M'_1, L'_2, such that $M'_1 \neq M_1$ and $L'_2 = \mathcal{H}(M'_1, M_2)$. In this way, her fake path merges into Π_2, but a propagation of the $(1, \Pi_2)$-collision to a $(1, \Pi_3)$-collision requires $L_3 = \mathcal{H}(M'_1, M_2, M_3)$. Since M_3 was not sent by $\mathcal{T}$ when the $(1, \Pi_2)$-collision was generated, M_3 was unknown to her for $(1, \Pi_2)$-collision. Due to Property 1, she can only guess the value of L_3 (or M_3) with probability 2^{-d}. Hence, L_3 cannot be used by the adversary to generate a $(1, \Pi_2)$-collision. The probability of a successful propagation of the $(1, \Pi_2)$-collision to a $(2, \Pi_3)$-collision is thus 2^{-d}. The probability of propagation decreases further as more L-fragments are added to the legitimate path.* ■

On the other hand, $\mathcal{A}$ can create both collisions and forks between her own z paths, which may result in an exponential number of fake paths. Therefore, our decoding algorithm (§ 4.3) uses v-counters for the M-fragments when exploring the search space. Each time a path from the search space is decoded, the v-counters associated with the path are incremented. If a counter reaches the threshold θ, the associated M-fragment is not used in the subsequent decoding process. In this way, the M-fragments of the adversary start becoming unavailable as the decoding proceeds.

We proceed in two steps. First, we model a *benign adversary*, who relies on the transmission of random messages as attack strategy, and quantify an upper bound on the probability of collisions in the search space (Claims 1 and 2). Later we quantify the advantage of a computationally unbounded adversary over the benign adversary (Claim 3), which brings us finally to argue on the security of our protocol (Claim 4).

Definition 5 (Random Transmission). *The random transmission of a benign adversary is $\mathbf{T}^{rnd} = [\mathbf{T}_{\mathit{l}}^{rnd}[i], \mathbf{T}_{\mathit{m}}^{rnd}[i] : 1 \leq i \leq m]$, where $\mathbf{T}_{\mathit{m}}^{rnd}[i]$ is a set consisting of z (uniformly distributed) d-bit random strings and $\mathbf{T}_{\mathit{l}}^{rnd}[i]$ is a set consisting of the i-th L-fragments of the link certificates of $\mathbf{T}_{\mathit{m}}^{rnd}$.*

Claim 1. *Let $\Pi_j^{\mathcal{T}}$ be the path of the legitimate transmission. For a z-channel benign adversary under Assumption 2 (i. e., with $z(z+1) < 2^d$), the following relation holds for $i < j \leq m$ and $\eta = (1+z)2^{-d}$:*

$$\textit{Number of } (i, \Pi_j^{\mathcal{T}})\textit{-collisions} \leq 2z\eta^{j-i}.$$

Claim 2. *The total number of collisions in a partial legitimate path $\Pi_j^{\mathcal{T}}$ in the presence of $\mathbf{T}^{rnd}$ is $\aleph_j^{rnd} < 2z\eta(j-1)$.*

Claim 3. *The total number of collisions in a partial legitimate path $\Pi_j^{\mathcal{T}}$ in the presence of $\mathbf{T}^{\mathcal{A}}$, which is due to a computationally unbounded adversary $\mathcal{A}$, is $\aleph_j^{\mathcal{A}} \leq \aleph_j^{rnd} + z$.*

The proofs of Claims 1, 2, and 3 can be found in Appendix A.

Claim 4. *The transmission $\mathbf{T}^{\mathcal{T}}$ of the CD protocol as specified in § 4.1 is secure as per Def. 3.*

Proof. From Eq. 1, we know that an upper bound on the running time of the decoding algorithm is $\rho\big(m\theta(z+1) + (z+1)^2\big)T_l$. The term T_l (the time to compute a link) is polynomial in m, due to our choice of a hash function as a linking function. The only unknown value is the threshold θ, which we determine in the following.

Using Claim 3, we can calculate an upper bound on the number of collisions into a legitimate full path Π_m. A fragment on a path Π_m can be visited by the decoding algorithm not only by full-length fake paths but also by partial fake paths (of length less than m). Therefore, an upper bound on the threshold θ is the total number of paths that can pass through a legitimate fragment:

$$\theta \leq \sum_{j=2}^{m} \aleph_j^{\mathcal{A}} + 1 = 2z\eta\big[1 + 2 + \cdots + (m-1)\big] + z + 1 = z\eta m(m-1) + z + 1. \quad (2)$$

Here, the constant 1 is due to the legitimate path itself. Clearly, the upper-bound is polynomial in m and z. Hence, with $\theta = z\eta m(m-1) + z + 1$ the decoding is guaranteed to succeed in polynomial time in the adversarial environment.

6 Performance Evaluation

We evaluate the theoretical results by a simulation that addresses the link-layer reassembly process on the receiver side. Physical-layer issues, such as signal strengths, modulation types, and the actual transmission of the signals are not part of this simulation.

Setup. The simulation code is written in C and is parameterizable for the adversarial power (z), length of fragments (d), number of fragments (m), and amplification factor (α). The simulation randomly generates a legitimate message consisting of m fragments and mixes the message with z other random messages to simulate the benign adversary[3]. The resulting search space has $(z+1)^m$ plausible combinations to search for a legitimate message. For a typical set of values, say $z = 2$ and $m = 128$, the infeasibility to explore this search space is clear.

The CDP protocol makes the search of the legitimate message tractable. To demonstrate this, we encode the legitimate message as per the CDP protocol. The linking function in the simulation is the truncated output of SHA-256. The CDP decoder efficiently reassembles all CDP messages from the search space. The output consists of one legitimate message, z adversarial messages, and a small number of accidental messages formed by pure chance. In the simulation, this whole process is called a *run*.

Metrics. We consider three metrics to demonstrate the performance of the CDP. The first one is Ω, the number of fake paths per legitimate path in a given search space. This metric is independent of the computer and language used to implement the CDP. The second metric is the running time τ_r of the decoding algorithm, which depends on the computer used for the simulation; in our case, it is a Thinkpad T400s laptop with 3GB memory and P9400 Intel Core 2 processor at 2.4GHz clock. This measure is machine specific and should be interpreted in a relative sense. The third metric is the maximum value of θ (see § 4.3), which determines the theoretical upper bound on the running time of the decoding algorithm, as per Eq. 1.

To get statistically significant measurements, each of the presented results is based on $10,000$ independent simulation runs. We report three values over these $10,000$ samples: the minimum value Ω_{min}, the arithmetic mean Ω_{avg}, and the maximum value Ω_{max}. Alongside, we also report the values of the standard deviation (SD) and the 95% confidence intervals (CI).

Results. The first set of results shown in Table 2a indicates the variation of Ω with respect to z. In this set, the message size is 768-bit, which is divided into 128 fragments of 6 bits. In these simulation runs, one link certificate is used ($\alpha = 1$). The results show that Ω increases with z, but it remains tractable as long as the threshold $z(z+1) < 2^d$ (cf. Assumption 1) is respected, which occurs at $z = 7$ in this case. Furthermore, the value of θ is consistent with the theoretical upper bound of Eq. 2.[4]

[3] From Claim 3, we know that a reactive adversary can generate at most z additional fake paths compared to the benign adversary. Hence, we can limit the simulation to the benign adversary for the performance evaluation and there from derive the results for the worst-case adversary.

[4] The upper bound of θ in Eq. 2 is large when compared to the actual value of θ reported in Table 2a. This is due to the extensive use of over-approximations in our security analysis.

Table 1. Performance results

z	0	1	2	3	4	5	6	7
Ω_{min}	1	2	3	4	5	6	8	37
Ω_{avg}	1	2.067	3.329	4.98	7.435	11.9	23.3	132.34
Ω_{max}	1	4	7	11	17	24	56	351
SD	0	0.256	0.565	0.997	1.637	2.81	5.94	44.968
95% CI	1–1	2–3	3–5	4–7	5–11	7–18	13–36	63–234
θ	1	2	3	4	5	6	9	32
τ_r [ms]	0.57	1.71	3.71	6.68	11.68	21.17	47.59	261.75

(a) z =variable, $d = 6$, $m = 128$, $\alpha = 1$

α	1	2	3	4	5	6
Ω_{min}	33	8	8	8	8	8
Ω_{avg}	131.66	8.8751	8.0943	8.0104	8.0011	8.0002
Ω_{max}	419	14	10	9	9	9
SD	46.228	0.9221	0.3082	0.1014	0.0331	0.0141
95% CI	62–238	8–11	8–9	8–8	8–8	8–8
θ	30	3	2	2	1	1
τ_r [ms]	308.75	177.17	289.0	437.667	562.5	711

(b) α =variable, $d = 6$, $m = 128$, $z = 7$

The second set of results is shown in Table 2b and indicates the variation of Ω with respect to the amplification factor α, for a 7-channel adversary. Once again, the message size is 768-bit divided into 128 fragments of 6-bit each. Ideally, there should be 8 paths, i. e., $\Omega = 8$, corresponding to one legitimate and 7 adversarial paths. As expected, the value of Ω quickly converges to 8 with an increase in α. In this particular configuration, $\alpha = 2$ is optimum for the decoding time. These simulation runs demonstrate that the use of additional link certificates can improve the security level of the CDP, however, increasing α too much can in turn degrade the decoding time, due to software overhead.

The third set of results is shown in Table 2 and indicates the variation of Ω with respect to the fragment size d. In these simulation runs, we use a 7-channel adversary and one link certificate. The message size is $128d$, which is formatted as 128 fragments of d bits. The results show that increasing d also increases the security level, namely the number of fake paths decreases. This is due to the fact that by increasing d the threshold on z also increases, due to the relation $z(z+1) < 2^d$. We also note that the machine we use for testing has a 32-bit architecture, which means increasing d from 6 to 13 does not significantly increase the software overhead, because each fragment is internally stored as a 32-bit memory word. Therefore, the decrease in τ_r is almost entirely due to a decrease in the number of fake paths.

Table 2. Performance results for d =variable, $\alpha = 1$, $m = 128$, $z = 7$

d	6	7	8	9	10	11	12	13
Ω_{min}	37	8	8	8	8	8	8	8
Ω_{avg}	132.339	14.674	10.291	9.007	8.467	8.223	8.113	8.053
Ω_{max}	351	28	17	15	12	11	11	10
SD	44.9682	2.9075	1.5322	1.0103	0.6836	0.467	0.339	0.233
95% CI	63–234	10–21	8–14	8–11	8–10	8–9	8–9	8–9
θ	32	6	4	3	3	2	2	2
τ_r [ms]	261.75	34.9	33.867	30.788	29.457	29.0	28.67	28.64

Security Performance Trade-Off. The CD protocol allows trade-offs between security and performance by changing d, α, and the linking function. As a typical design flow, consider a reactive adversary who takes at least $10\mu s$ to sense and jam an active SS channel. If each of the channels supports a data rate of 800-$kbps$ or more then it is safe to assume that 8 bits cannot be deterministically jammed within a 1-μs window, which implies $d \leq 8$-bit. From the threshold $z(z + 1) < 2^d$, we can derive the maximum value of z to be $z \leq 15$. Since z is the ratio of the absolute power (in Watts) of the adversary's antenna to that of $\mathcal{T}$'s antenna, $\mathcal{T}$'s transmission power can be changed to an optimal level that meets the constraint of $z \leq 15$, as proposed by Xu *et al.* [16].

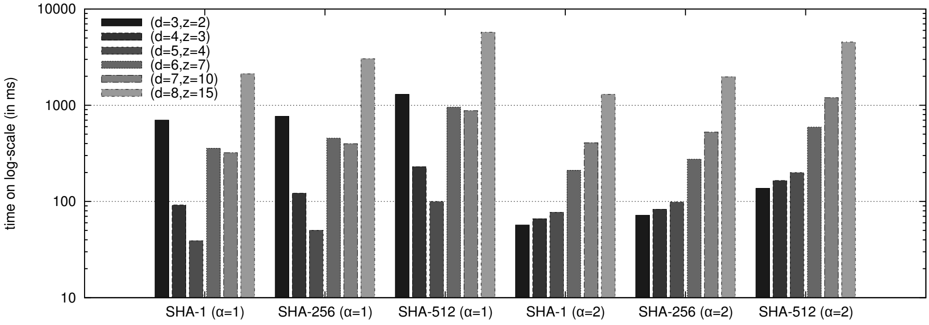


Fig. 5. Visualization of security and performance trade-offs. For the left three sets of data, $\alpha = 1$, for the right three, $\alpha = 2$. The parameters d and z respectively model the levels of protection against reactive and pollution attacks (the smaller d and the larger z, the more protection the scheme provides). The decoding times are specific for our implementation; they must be interpreted in a relative sense. The message size in all these cases is about 1024-bit: for instance, $m = 128$ for $d = 8$, and $m = 171$ for $d = 6$.

The next step is the design of a CDP receiver with $z \leq 15$ and $d \leq 8$. Fig. 5 shows how different linking functions and values of α change the decoding time in our simulation. The best decoding time is achieved with SHA-1, $\alpha = 1$, $d = 5$ for a 4-channel adversary with a $1ms$ reaction time. Fig. 5 also shows that the use of more than one link ($\alpha \geq 2$) is beneficial if the fragment length d is very small and that the selection of the hash-function may have a non-negligible impact on overall running time.

7 Related Work

Many protocols have been proposed for uncoordinated anti-jamming communication. Many of these protocols [17–19] assume that packets of arbitrary size can be received with a certain probability, which, however, is only realistic if the jammer is non-reactive or the combined transmission power of senders is greater than that of the jammer.

The first scheme for keyless anti-jamming broadcast was proposed in 2007 by Baird *et al.* [3] using concurrent codes. Its security depends on the pre-image resistance of the hash function and the adversary's inability to delete or overshadow $1s$ in transmitted packets. For impulse radios employing time hopping [20], this scheme is promising; for DSSS and FHSS, however, it requires specialized radio transceivers.

Strasser *et al.* [1] propose a scheme for UFH using the hash function to achieve anti-jamming key-establishment. The fragment sizes of this scheme are in the order of hundreds of bits, e. g., SHA-1 based fragments are to be longer than 160 bits. Since the security of this scheme depends on the computational power of the adversary, a truncated hash of a small size, such as 16 bit, will make the scheme insecure.

To make the solutions more efficient, Strasser *et al.* [9], Slater *et al.* [10], and Pöpper *et al.* [2] propose alternatives to the hashes, but the fragment size remains a limiting factor. For example, for the scheme using short signatures based on bilinear maps [9], the linking information is $4k$-bit, where k is the security level, again resulting in packets of few hundred bits (even for short-term security levels). The decoding of messages in these schemes is (partially) offline. The decoding process in our protocol is online, i. e., the decoding can start as soon as fragments are arriving. Wang *et al.* [21] model UFH transmissions as a multi-armed bandit problem. Since they assume the same verification schemes as proposed by Strasser *et al.* [1, 9], this scheme is vulnerable to reactive jamming due to long packets. In our protocol, the requirement $z(z+1) < 2^d$ implies the fragment size to be $2log_2(z+1)$-bit (e.g., 2-bit for $z = 1$). This means that a single hop in FHSS systems or a sequence of codes in DSSS systems is used to transmit only $2log_2(z+1)$ bits, thus allowing a higher switching frequency.

As an alternative to UFH, Pöpper *et al.* [22] propose a UDSSS scheme that transmits messages of fixed length without splitting them into fragments; large messages and reactive jammers are also a problem for this scheme. Jin *et al.* [23] propose a DSSS-based scheme using time-reversed message extraction and key scheduling. This scheme can only be applied for a known receiver, it requires offline decoding, and the size of the first fragment is half of the full message size. Liu *et al.* propose a UDSSS protocol based on the theory of finite projective planes [24] and another protocol called randomized differential (RD)-DSSS [4]. Both are quite robust against reactive jammers, but cannot directly be applied to FHSS-based communication. RD-DSSS assumes a computationally bounded adversary, as opposed to our CDP.

Xu *et al.* [25] propose the use of timing-based covert channels for anti-jamming transmission. They, however, assume that the transmission power of an adversary is comparable to that of a legitimate transmitter, i. e., $z \approx 1$. To cope with insertion or pollution attacks, where an adversary transmits fake packets, the authors assume shared secrets. Xiao *et al.* [5] propose a UFH-based broadcast scheme that assumes collaboration of receivers, namely after having received a message the receiver helps to broadcast it to other receivers. This is orthogonal to our work. Strasser *et al.* [26] propose a complementary technique to detect jamming attacks using received signal strength. If the presence of a jammer is detected then the receiver can take additional countermeasures, such as changing the location to reduce the strength of the jamming signals. In contrast to BitTrickle proposed by Liu *et al.* [12], our protocol does not rely on physical-layer authentication approaches to verify individual bits nor on specific hardware.

Xu *et al.* [16] propose an adaptive protocol for point-to-point UFH transmission, which is modelled as a game played between a sender, a receiver, and a jammer. The transmission power of the transmitter and jammer may vary. As compared to our solution, the jammer is assumed to be weaker, not being able to jam in the current time-slot. The authors assume bi-directional, time-slotted communication between the sender and receiver, in which the sender can always receive acknowledgement (ACK) from the receiver in the same time-slot and frequency channel.

8 Conclusion

Avoiding DoS-threats is important for all dependable wireless networks. USS protocols are link-layer protocols that are required in situations where parties who do not share a secret need to communicate under the threat of (reactive) jamming attacks. Existing USS protocols, however, are not effective enough against fast reactive jammers due to their dependency on long fragments. In this context, the presented protocol is the first USS protocol that supports short fragments of a few bits. Our protocol uses the idea of collision detection to efficiently link legitimate fragments, even when the jammer pollutes the communication channels with a large number of fake fragments using concurrent transmissions. We hope that the presented protocol will serve as an important technique for bootstrapping spread spectrum communication in cases where shared secrets do not exist or have been compromised.

References

1. Strasser, M., Pöpper, C., Capkun, S., Cagalj, M.: Jamming-resistant key establishment using uncoordinated frequency hopping. In: Proc. of S&P, pp. 64–78. IEEE (2008)
2. Pöpper, C., Strasser, M., Capkun, S.: Anti-jamming broadcast communication using uncoordinated spread spectrum techniques. IEEE Journal on Selected Areas in Communications 28(5), 703–715 (2010)
3. Baird, L., Bahn, W., Collins, M., Carlisle, M., Butler, S.: Keyless jam resistance. In: Information Assurance and Security Workshop (IAW), pp. 143–150. IEEE (2007)
4. Liu, Y., Ning, P., Dai, H., Liu, A.: Randomized differential DSSS: Jamming-resistant wireless broadcast communication. In: Proceedings of INFOCOM, pp. 1–9. IEEE (2010)
5. Xiao, L., Dai, H., Ning, P.: Jamming-resistant collaborative broadcast using uncoordinated frequency hopping. IEEE Trans. on Information Forensics & Security 7(1), 297–309 (2012)
6. Hamilton, S., Hamilton, J.: Secure jam resistant key transfer: Using the DOD CAC card to secure a radio link by employing the BBC jam resistant algorithm. In: Military Communications Conference (MILCOM), pp. 1–7. IEEE (2008)
7. Wilhelm, M., Martinovic, I., Schmitt, J., Lenders, V.: Short paper: Reactive jamming in wireless networks. In: Proc. of WiSec, pp. 47–52. ACM (2011)
8. Wilhelm, M., Martinovic, I., Schmitt, J., Lenders, V.: Wifire: a firewall for wireless networks. In: Proceedings of the ACM SIGCOMM Conference, pp. 456–457. ACM (2011)
9. Strasser, M., Pöpper, C., Capkun, S.: Efficient uncoordinated FHSS anti-jamming communication. In: Proc. of MobiHoc, pp. 207–218. ACM (2009)
10. Slater, D., Tague, P., Poovendran, R., Matt, B.J.: A coding-theoretic approach for efficient message verification over insecure channels. In: Proc. of WiSec, pp. 151–160. ACM (2009)
11. Slater, D., Poovendran, R., Tague, P., Matt, B.J.: Tradeoffs between jamming resilience and communication efficiency in key establishment. SIGMOBILE Mobile Computing and Communications Review 13, 14–25 (2009)
12. Liu, Y., Ning, P.: BitTrickle: Defending against broadband and high-power reactive jamming attacks. In: Proceedings of INFOCOM, pp. 909–917. IEEE (2012)
13. Cabric, D., Tkachenko, A., Brodersen, R.W.: Experimental study of spectrum sensing based on energy detection and network cooperation. In: Proc. of TAPAS. ACM (2006)
14. Bhargava, V.: Forward error correction schemes for digital communications. IEEE Communications Magazine 21(1), 11–19 (1983)
15. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)
16. Xu, K., Wang, Q., Ren, K.: Joint UFH and power control for effective wireless anti-jamming communication. In: Proceedings of INFOCOM, pp. 738–746. IEEE (2012)
17. Awerbuch, B., Richa, A., Scheideler, C.: A jamming-resistant MAC protocol for single-hop wireless networks. In: Proc. of PODC, pp. 45–54. ACM (2008)
18. Dolev, S., Gilbert, S., Guerraoui, R., Newport, C.: Gossiping in a multi-channel radio network. In: Pelc, A. (ed.) DISC 2007. LNCS, vol. 4731, pp. 208–222. Springer, Heidelberg (2007)

19. Dolev, S., Gilbert, S., Guerraoui, R., Newport, C.: Secure communication over radio channels. In: Proc. of ACM Symp. on Principles of Distributed Computing, pp. 105–114 (2008)
20. Win, M., Scholtz, R.: Impulse radio: How it works. IEEE Communications Letters 2(2), 36–38 (1998)
21. Wang, Q., Xu, P., Ren, K., Li, X.: Towards optimal adaptive UFH-based anti-jamming wireless communication. Journal on Selected Areas in Communications 30(1), 16–30 (2012)
22. Pöpper, C., Strasser, M., Capkun, S.: Jamming-resistant broadcast communication without shared keys. In: Proceedings of the USENIX Security Symposium, pp. 231–247 (2009)
23. Jin, T., Noubir, G., Thapa, B.: Zero pre-shared secret key establishment in the presence of jammers. In: Proc. of MobiHoc, pp. 219–228. ACM (2009)
24. Liu, A., Ning, P., Dai, H., Liu, Y., Wang, C.: Defending DSSS-based broadcast communication against insider jammers via delayed seed-disclosure. In: Proceedings of the 26th Annual Computer Security Applications Conference, pp. 367–376. ACM (2010)
25. Xu, W., Trappe, W., Zhang, Y.: Anti-jamming timing channels for wireless networks. In: Proceedings of Wireless Network Security (WiSec), pp. 203–213. ACM (2008)
26. Strasser, M., Danev, B., Capkun, S.: Detection of reactive jamming in sensor networks. ACM Transactions on Sensor Networks (TOSN) 7(2), 16 (2010)

A Proofs of Claims

A.1 Proof of Claim 1

Proof. Let p_j be the probability of a $(i, \Pi_j^{\mathcal{T}})$-collision due to one fake path Π_i'. Let q_i be the total number of fake paths of length i; each of the q_i paths can cause a $(i, \Pi_j^{\mathcal{T}})$-collision. Therefore, we have

$$\text{Number of } (i, \Pi_j^{\mathcal{T}})\text{-collisions} \leq p_j q_i. \tag{3}$$

We next determine the values of p_j and q_i.

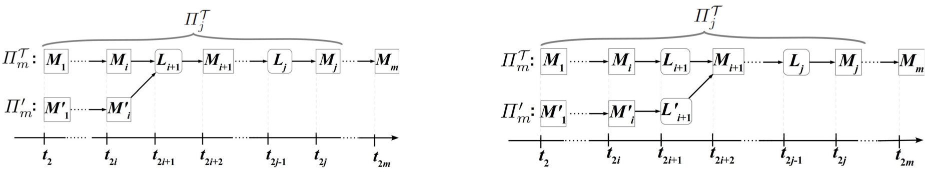
Value of p_j: According to Def. 4, a $(i, \Pi_j^{\mathcal{T}})$-collision occurs if a fake path Π_j' merges into the legitimate path $\Pi_j^{\mathcal{T}}$ after index i, i. e., $\Pi_{i+1\to j}' = \Pi_{i+1\to j}^{\mathcal{T}}$. By definition, a $(i, \Pi_j^{\mathcal{T}})$-collision implies $(i, \Pi_{i+1}^{\mathcal{T}})$, …, $(i, \Pi_j^{\mathcal{T}})$-collisions. The required conditions corresponding to these collisions, $\Re_{i+1} \dots \Re_j$ are as follows:

$$\begin{aligned} \Re_{i+1} &\overset{\text{def}}{=} \exists L_{i+1} \in \sigma_{2i+1} : L_{i+1} = \mathcal{H}(M_1, \dots, M_{i+1}) = \mathcal{H}(M_1', \dots, M_i', M_{i+1}) \\ &\vdots \\ \Re_j &\overset{\text{def}}{=} \exists L_j \in \sigma_{2j-1} : L_j = \mathcal{H}(M_1, \dots, M_j) = \mathcal{H}(M_1', \dots, M_i', M_{i+1}, \dots, M_j). \end{aligned} \tag{4}$$

Here, $\{M_i, M_i'\} \subseteq \sigma_{2i}$ and $\{M_j\} \subseteq \sigma_{2j}$. Let $p_{i+1}, \dots, p_j$ be the probabilities corresponding to the $(i, \Pi_{i+1}^{\mathcal{T}}), \dots,$ $(i, \Pi_j^{\mathcal{T}})$-collisions. First we claim that $p_{i+1} \leq \eta$ because a path Π_i' can use any of the L_{i+1} in σ_{2i+1} to cause a $(i, \Pi_{i+1}^{\mathcal{T}})$-collision. For a given value of L_{i+1}, the collision probability is 2^{-d} due to Property 1 of the linking function. We have $|\sigma_{2i+1}| = z + 1$, therefore, $p_{i+1} \leq (z + 1)2^{-d} = \eta$. On the next pair of fragments, once again for $z + 1$ values of L_{i+2}, we have $Pr\big[(i, \Pi_{i+2}^{\mathcal{T}})\text{-collision}|\ (i, \Pi_{i+1}^{\mathcal{T}})\text{-collision}\big] \leq \eta$. Thus, $p_{i+2} = p_{i+1} \cdot Pr\big[(i, \Pi_{i+2}^{\mathcal{T}})$

Table 3. Maximum number (#) of fake paths along the legitimate path

Index i	1	2	3	...	$j-1$
q_i (max. # of fake paths of length i)	z	$\leq z+\frac{1}{2}z=\frac{3}{2}z$	$\leq z+\frac{3}{4}z=\frac{7}{4}z$	...	$\leq (\frac{2^{j-1}-1}{2^{j-2}})z$
Prob. of (i, Π_{i+1})-collision per fake path	$p_{i+1} \leq \eta \leq \frac{1}{2}$	$p_{i+1} \leq \eta \leq \frac{1}{2}$	$p_{i+1} \leq \eta \leq \frac{1}{2}$	...	$p_{i+1} \leq \eta \leq \frac{1}{2}$
Total # of (i, Π_{i+1})-collisions	$\leq \eta z = \frac{1}{2}z$	$\leq \frac{3}{4}z$	$\leq \frac{7}{8}z$	...	$\leq (\frac{2^{j-1}-1}{2^{j-1}})z$



(a) (b)

Fig. 6. Long and short fragment transmissions. (a) Long fragments are an easy target for jammers. Reactive jammers will have sufficient time to sense an ongoing transmission and then react to (deterministically) jam the used channel. Sweep (non-reactive) jammers have a high probability of hitting the channel where a long fragment is transmitted. (b) Short fragments reduce the risk of successful jamming attacks for both reactive and non-reactive sweep jammers.

-collision$|$ $(i, \Pi_{i+1}^{\mathcal{T}})$-collision$\big] \leq \eta^2$. Re-applying the above arguments until index j, we get this result:

$$p_j \leq \eta^{j-i}. \tag{5}$$

Value of q_i: We assume $z(z+1) < 2^d$, which implies $\eta \leq 1/2$.[5] Table 3 shows upper bounds on the value of q_i (the number of fake paths of length i). For the first pair of fragments, there are no collisions, but there are z adversarial paths due to $\mathcal{T}^{rnd}$, thus $q_1 = z$. Next, q_2 is the sum of the number of $(1, \Pi_2)$-collisions and the number of adversarial paths (z). Repeating this procedure, we can calculate the rest of the table. For convenience, we use a simpler expression $2z$ to represent an upper bound on q_i:

$$q_i \leq (\frac{2^{j-1}-1}{2^{j-2}})z < 2z. \tag{6}$$

Substituting Eq. 5 and Eq. 6 in Eq. 3 proves the claim.

[5] Solving η from $z(z+1) < 2^d$ results in $\eta < 1/z$. Clearly, for $z > 1$, we have $\eta \leq 1/2$. For the remaining two values, $z = 1$ and $z = 0$, the minimum values of d that satisfy the relation $z(z+1) < 2^d$ are 1 and 2 respectively, which implies that $\eta = 1/2$ for these two values.

A.2 Proof of Claim 2

Proof. An upper bound on $\aleph_j^{rnd}$ is obtained by summing up all collisions on path $\Pi_j^{\mathcal{T}}$:

$$\begin{aligned}\aleph_j^{rnd} &\leq \sum_{i=1}^{j-1} \text{Number of } (i, \Pi_j^{\mathcal{T}})\text{-collisions} = \sum_{i=1}^{j-1} q_i p_j \quad \text{(using Claim 1)}\\ &< \sum_{i=1}^{j-1} 2z\eta^{j-i} = 2z\eta^j \left[\eta^{-1} + \eta^{-2} + \cdots + \eta^{-(j-1)}\right]\\ &< 2z\eta^j \left[\eta^{-(j-1)} + \cdots + \eta^{-(j-1)}\right] = 2z\eta(j-1).\end{aligned}$$

This completes the proof.

A.3 Proof of Claim 3

Proof. The proof is based on the following observation: The adversary is constrained by the fact that she must send her fragment at t_i before receiving $\mathcal{T}$'s fragment at t_{i+1} because fragments are transmitted in order. The conditions to create collisions are the same as in Eq. 4. There are two cases of collisions, which we analyze in the following.

First, we consider case (a) of the $(i, \Pi_j^{\mathcal{T}})$-collision as shown in Fig. 6-(a). Here, the fragment M_{i+1} is unknown before $t_{2i+1}$and, hence, the output of the linking function, L_{i+1}, appears as a random value to the adversary. Before t_{2i+1}, there is no strategy to generate $[M_1', \ldots, M_i']$ deterministically in such a way that a collision can be created.

Next, we consider case (b) of the collision, as in Fig. 6-(b). We analyze the situation at t_{2i+1} when $\mathcal{T}$ transmits L_{i+1}. On reading L_{i+1}, the adversary's task is to generate L_{i+1}' such that the condition $\Re_{i+1}$ holds. The condition $\Re_{i+1}$ can be satisfied by computing a new L_{i+1}', but a z-channel adversary cannot generate more than z values of L-fragments at a time. Therefore, $\mathcal{A}$ can only create z number of $(i, \Pi_{i+1}^{\mathcal{T}})$-collisions.

If the adversary is not active at t_{2i+3}, a $(i, \Pi_{i+1}^{\mathcal{T}})$-collision cannot deterministically propagate to a $(i, \Pi_{i+2}^{\mathcal{T}})$-collision because, at t_{2i+1}, the value of M_{i+2} (required to compute L_{i+2}) is still unknown. If the adversary is active at t_{2i+3}, the above arguments for $\Re_{i+1}$ can be applied to $\Re_{i+2}$ to generate either a new $(i+1, \Pi_{i+2}^{\mathcal{T}})$-collision or make a $(i, \Pi_{i+1}^{\mathcal{T}})$-collision propagate as a $(i, \Pi_{i+2}^{\mathcal{T}})$-collision. The same arguments can be applied for the rest of the conditions up to $\Re_j$.

Therefore, we conclude that the best $\mathcal{A}$ can do is to create z number of $(i, \Pi_j^{\mathcal{T}})$-collisions. Creating z collisions, however, does not affect the number of collisions that are inherently present in the search space due to random transmission, because any attack strategy can be considered as an instance of random transmissions in which the adversary provides the output of coin tosses. Hence, the total number of collisions is $\aleph_j^{\mathcal{A}} \leq \aleph_j^{rnd} + z$. This completes the proof.