Efficient Adversarial Chaff Generation for Challenge-Response Authentication Over Unsecure Networks with an Application to Civilian Radio Networks

Brendan Kitts and Andrew Potter
Verizon
Seattle USA
brendan.kitts@verizonmedia.com

***Abstract*— Challenge Response is one of the cornerstones of online security. The simplest form of Challenge-Response is asking for a password. Much cryptographic work has focused on developing strong forms of encryption, however some networks require transmission over networks which might be monitored. We discuss this problem in the context of a particular kind of open network used by 30,000 users, and which is an important medium supporting emergency services. The current challenge-response implementation on this network relies upon sending information about the password. We calculate the number of observations needed to capture password using brute force attack, replay attack, and version spaces. We show that even strong passwords (completely random set of characters) are at significant risk of discovery in as few as 16 login attempts. We next present an algorithm that adds adversarial "chaff" to the password information designed to minimize relative information gain during challenge-response. We show that, with enough adversarial chaff, unambiguous password recovery from passive data capture may not be possible, although passwords can still be recovered by an attacker actively probing the system. Despite this, better protection of passwords is useful, and would be immediately helpful to people using these services.**

Keywords—challenge, response, authentication, chaff, unsecure, open, APRS, email, amateur, radio (*key words****)***

I. INTRODUCTION

Establishing identity is a common problem in computer security. In most circumstances, this is achieved by issuing a challenge to which the user must respond correctly. A prompt asking for a password is the simplest example. The password is typically encrypted using a variety of strong encryption standards to protect against discovery.

However, in some cases, the network over which the password needs to be transmitted is monitored or unsecure. It would be simple for an attacker to capture the response, and then use it to gain access.

To authenticate in this situation, it is common to create a "shared secret" such as a password, between the user and service, along with an agreed upon mapping function such as a hash. The service will then issue a challenge in the form of a one-time occurring nonce integer. The user uses the challenge plus the secret to compute the answer and sends back a response. The server compares the returned response against their own calculation, and if equal they are authenticated. Assuming a cryptographically secure hash function it will be difficult for an attacker to infer the generating hash function and secret.

In 1998 Ronald Rivest proposed an alternative, Steganographic method, for unsecure networks [8]. Unlike the cipher approach above, in which the message content is encrypted, in Steganography, the message is hidden in plain text, within another message.

Steganography has a long history. The Cardan Grille is based on an ancient technique where a paper template is created with holes in it. When placed over a text, the holes reveal an embedded message [2]. In more modern times, data has been embedded in the low order bits of images.

Rivest proposed that a message could be first broken down into pieces, with each piece being assigned a valid Message Authentication Code (MAC) and a Sequence Number for reconstruction. After this, generated "chaff" – false information with false MACs and Sequence Numbers, would be embedded within the message. The combined message could then be transmitted "in the clear" to a receiver. The receiver would know the algorithm for checking the validity of MACs, and would simply re-assemble the valid message fragments by sequence number.

Chaffing has some curious advantages. Most internet users reside in countries with laws requiring assistance to decrypt communications, but this unfortunately means that it is not inconceivable that encrypted communications may be vulnerable to compromised Back Doors [6]. Chaffing may be immune to this vulnerability, and can be implemented peer-to-peer over monitored networks. Estrada (1999) also noted that Chaffing could be used to embed a covert communication channel within an existing open communication network [3].

Chaffing does result in much larger message; for instance, [5] noted that a 30KB message will generally grow to about 75MB. However, if the message were compressed prior to chaffing, it would decrease message size. This would present an additional advantage, as the resulting message content would be near random, and so random chaff would actually be quite effective in obscuring the real message. Random chaff could enable third parties, with no knowledge of message content, to participate in making messages more secure [8]. Samid (2018) observed that non-randomized chaff might be even more effective [9].

In this paper we use the concept of chaff to improve the security of authentication challenges and responses over open networks. We also use Samid's insight that non-random chaff may be more effective at hiding message content.

The particular problem that we deal with is how to authenticate email users over unencryptable Amateur Radio Networks [4]. We analyze an existing Challenge-Response system and note that passwords can be reliably recovered in just 16 trials. We then propose a chaffing algorithm based on minimizing Information Gain from subsequent responses, designed to slow time to discovery.

II. Civilian Radio Networks

Amateur radio spans a wide range of frequencies ranging from long wave to high frequency, VHF, UHF and Microwave, where licensed radio operators are allowed to transmit and receive information. Amateur radio is often used during emergencies to provide fall back communications for government entities. RACES (Radio Amateur Civilian Emergency Services) and ACS (Auxiliary Communication Services) are organizations supported by national laws that can be activated during emergencies to assist communications in natural and man-made disasters.

In the United States, radio networks are governed by Part 97 of the FCC regulations, and other countries have similar regulations. Under these regulations, amateur radio data transmission must not be encrypted. Title 47 Part 97.309.B states "data emissions using unspecified digital codes must not be transmitted for the purpose of obscuring the meaning of any communication" [4]. The reasoning behind this is that amateur radio uses public frequencies, and so the data being transmitted should be readable by the public.

This prohibition makes it difficult to support standard communication services like email. If a user wants to read email over a radio network (because regular cellphone tower or fiber networks are down during a disaster), authentication is needed to ensure that the proper user is sending and receiving email for a given email address.

The most widely used email system on amateur radio is APRS and Winlink – these can be used together, or with Winlink alone [12]. This email system has been deployed with great success in US Western Wildfires of 2018, Hurricane Katrina, Rita and Maria [17], and has a record of deployments as long ago as 1991 [16].

APRS (Automatic Packet Reporting System) is widely used with about 30,000 stations transmitting every hour as of 2019 [10]. APRS was developed by Bob Bruninga in the 1980s [1], and the acronym, APRS is partially derived from his call sign WB4APR. APRS is similar in operation to TCP/IP, in that AX25 packets are broadcast via radio usually on a frequency of 144.390 MHz. Receiving stations read the message, and then re-broadcast until the message reaches its intended target, or the message decrement drops to 0 in which case the message is abandoned as undeliverable. Any radio station can receive and interpret the transmitted data [1, 11].

Winlink comprises radio enabled servers for managing, receiving and transmitting emails, and can be standalone or accessed via APRS [12]. Winlink has about 12,000 registered email users, and transmits about 100,000 emails per month [14].

This paper is mainly concerned with accessing Winlink via APRS (APRSLink). In this mode, a radio operator uses APRS to send email commands and messages. Receiving radio stations pass these messages over the air until a radio station with a working internet connection (iGate), is reached. When this is reached, it sends the email over the internet to Winlink servers which implement the email commands. When sending email back to the originating station, the process is reversed [15].

Authentication via password is challenging over radio networks, since the passwords have to be transmitted "in the clear". Winlink creators set up the following challenge-response protocol to provide some security for accessing email via APRS:

(1) A password is created externally on the web.

(2) When logging in over open radio network, the system prompts with "Login [XYZ] where X,Y,Z are three character positions in the password.

(3) The user then responds with the three characters of the password in any order, and also adds another three (3) characters of "chaff" – random characters that are designed to make it harder for external viewers to guess the password.

The password controlling access to the account is currently between 6 and 12 characters and comprise an alphabet of upper case letters (A-Z), numeric digits (0-9), and symbols .!@#$%^&*()_. This alphabet has 48 characters [13].

There are two kinds of vulnerabilities with this system: (1) One-time Access to services: a malicious party may be able to log into email service, and then send/receive emails. (2) Password recovery: If the user's password can be recovered, then the attacker can take control of the account, change the password, and perhaps even log into other services used by the same user. This is a more serious problem.

This paper will address the problem of how to better secure authentication over these networks. We describe the current framework including several attack methods. We show that even strong passwords under APRSLink/Winlink can be captured in as few as 16 observations with 99% probability of success. We next discuss an algorithm for generating adversarial "chaff" that is designed to minimize information gain for attackers. Well-selected chaff may be able to prevent unambiguous recovery of the password, although the possibilities can be narrowed, and the attacker can then still search the remaining possibilities. Despite still being vulnerable, this is a meaningful improvement over the current system.

TABLE I. WINLINK/APRSLINK EMAIL SESSION

Date	time	Path	Message
3/17/19	14:17:53	WLNK-1>[CALLSIGN]	You have 2 Winlink mail messages pending
3/17/19	14:18:00	[CALLSIGN]>WLNK-1	I
3/17/19	14:18:01	WLNK-1>[CALLSIGN]	Login [798]:
3/17/19	14:19:11	[CALLSIGN]>WLNK-1	1UTERW
3/17/19	14:19:12	WLNK-1>[CALLSIGN]	Hello [CALLSIGN]
3/17/19	14:19:22	[CALLSIGN]>WLNK-1	L
3/17/19	14:19:22	WLNK-1>[CALLSIGN]	1) 03/17/2019 21:19:20 test from external 371 bytes
3/17/19	14:19:23	WLNK-1>[CALLSIGN]	2) 03/17/2019 21:19:20 test 590 bytes
3/17/19	14:19:50	[CALLSIGN]>WLNK-1	R1

3/17/19	14:19:51	WLNK-1>[CALLSIGN]	test from external Fm:SMTP:xxxx@xxxx.com Msg:this is an email
3/17/19	14:19:52	WLNK-1>[CALLSIGN]	transmitted over radio.
3/17/19	14:20:40	[CALLSIGN]>WLNK-1	?
3/17/19	14:20:41	WLNK-1>[CALLSIGN]	SP, SMS, L, R#, K#, Y#, F#, P, G, A, I, PR, B (? + cmd for more)
3/17/19	14:21:34	[CALLSIGN]>WLNK-1	? G
3/17/19	14:21:35	WLNK-1>[CALLSIGN]	Return closest RMS Packet Gateway information: G#
3/17/19	14:21:45	[CALLSIGN]>WLNK-1	G1
3/17/19	14:21:46	WLNK-1>[CALLSIGN]	Gateway: W7ACS-10 CN87UO 5 miles -60 deg, 430.850 1200b
3/17/19	14:21:56	[CALLSIGN]>WLNK-1	I
3/17/19	14:21:57	WLNK-1>[CALLSIGN]	APRSLink v5.0
3/17/19	14:22:18	[CALLSIGN]>WLNK-1	B
3/17/19	14:22:18	WLNK-1>[CALLSIGN]	Log off successful

III. Problem Definition

Let an APRS/Winlink Email password W be equal to a sequence of L characters $W = \langle w_1, w_2, w_3, \ldots, w_L \rangle : w_i \in Alphabet$. The cardinality of $Alphabet=A$. The most secure passwords will be random sequences, and we use this assumption for calculating attack time complexity. (Attack times could be reduced further using a dictionary, but this would introduce some arbitrariness to these results, and random sequences also enables closed form solutions in many cases).

Let a challenge on iteration t be defined as a set of positions $Challenge_t = \{p_1, p_2, \ldots, p_P\} : p_i \in [1..L]$ where P is the number of valid positions to send back. Let the response at iteration t be defined as the set of $P+C$ characters from the alphabet $Response_t = \{a_1, a_2, \ldots, a_{P+C}\} : a_i \in Alphabet$. A valid response is one for which the characters in the response are members of the set of valid position POS. $Response_t = \{a_1, a_2, \ldots, a_{P+C}\} : a_i \in POS \cup CHAFF$; $POS = \{w_{p_1}, w_{p_2}, \ldots, w_{p_P}\}$; $CHAFF = \{c_1, c_2, \ldots, c_C\} : \neg(c_i \in POS)$ & $c_i \in Alphabet$ where P is the number of requested positions, C is the number of chaff characters to sent back. If the system receives an incorrect response to the challenge, it will persist with 2 more tries, and after $TRIES=3$ consecutive failures, will disallow further access.

The problem for the Attacker is to use a set of observed Authentications $(Challenge_t, Response_t) \in Authentications$ to infer the user's underlying password W or otherwise gain access to the system by correct response to challenges. The default parameters for this problem are Alphabet size A=48, number of valid position characters P=3, number of chaff characters C=3, password length L = 6, $TRIES$=3.

IV. Attack I: Random Guessing

The expected time to randomly guess a password is $0.5*A^L$. For a length L=6, A=48, random character password, that is $E(T)$ = 6,115,295,232 trials. However, since only part of the password is requested - P position characters – the probability of one-time-access is much higher. The probability of access will equal the chance of drawing P successes (red balls), from $P+C$ trials, given that there are P correct results for the valid positions (red balls), and there are $A-P$ incorrect characters (blue balls); a Hypergeometric distribution. The expected time $E(T)$ is the inverse of this probability. For P=3, C=3, A=48, $E(T) = \left[\frac{\binom{P}{P}\binom{A-P}{P+C-P}}{\binom{A}{P+C}}\right]^{-1}$ = 833 trials.

V. Attack II: Replay

Given that the network is monitorable, better performance can be achieved by recording all of the challenges and their successful responses into a codebook. After this is done, the attacker can lookup the valid response for any challenge. There are $Q = L!/(L-P)!$ unique possible challenges or queries, and so storage would comprise every query with the chaff filled response $Q \cdot (P+C)$. The expected time needed to sample all challenges will equal: $E(T) = Q \cdot \sum_{i=1}^{Q} \left(\frac{1}{i}\right) = Q \cdot H_Q$ where H_Q is the Harmonic sum. Assuming L=6, P=3, it will take 644 trials to fill this codebook. It is, however, possible to speed up further by using an *incomplete* codebook, since the user has $TRIALS$=3 trials to enter a correct response to a challenge. The formula is below where f is the codebook fill rate needed. The trials needed to capture challenge response entries that would have probability of success $crit$=1.00, 0.99, 0.95 and 0.90 is 644, 186, 120 and 93 trials.

$$\text{E[T]} = Q \cdot \left[H_Q - H_{(Q-fQ)}\right] \text{ where } f = 1 - \exp\left(\frac{\ln(1-crit)}{TRIES}\right)$$

VI. Attack III: Version Spaces

In order to speed password capture even further, we introduce the idea of *Version spaces*. A Version space is a set of hypotheses, comprising a set of disjunctive propositions $v_{i,1} \vee v_{i,2} \vee \ldots \vee v_{i,A}$ [7]. For each password character position i, define an array of size A which represents 1 if the character is possible, and 0 if it is not. $v_{i,a} \in \{0,1\}$: $a \in A$ and $i \in [1..L]$. Initialize $v_{i,a} = 1 \; \forall i,a$ (all hypotheses are possible) and L = 12. Next every time that we observe a challenge and response, update the version space as follows:

$$v_{i,a} = v_{i,a} \wedge 0 \; \forall a \in A - (Response_t \cup Challenge_t)$$

$$L = \max(L, \max(Challenge_t))$$

The space needed to store the Version Space is modest; at most $O(L \cdot A)$ bits where L=12 is the maximum length of password. Given default parameters, only 12*48 = 576 bits are needed per user in order to execute this attack. The speed at which Version Spaces can be used to capture passwords can be calculated as follows: Assume that the user is adding random chaff. The expected time to recover password can be calculated as the expected time to sample all positions multiplied by expected time to eliminate chaff:

$$E(T) = \left[L \cdot \sum_{i=1:P:L} \left(\frac{1}{i}\right)\right] \cdot \left[\ln(1-crit)/\ln\left(\frac{P+C}{A}\right)\right]$$

Using Stirling's Approximation, the left-hand side of the equation grows as a function of the $L \cdot \ln(L)$, making this an extremely fast algorithm. Given a desired $crit$=0.90 and 0.99 chance of recovering password, the time to recovery is equal to 8.3 and 16.6 trials.

VII. Adversarial Chaff

We can now use Version Spaces to create chaff designed to defeat attackers. A simple algorithm is to select the chaff character which results in the least loss of hypotheses. This is, equivalently, the character a with the highest match count with

existing hypotheses: $a: \max \sum_{p \in POS} v_{p,a}$. However, this approach won't protect a critical character whose loss might reveal the correct password character. A better approach is to calculate the Entropy before $E[v]$ versus after removal of each candidate chaff character $E[v \backslash a]$, and select the character a which minimizes Information Gain $\Delta E[a]$

$$a: \min \Delta E[a] = E[v] - E[v \backslash a]$$

$$E[v] = \sum_{p \in POS} \sum_{a \in A: v_{p,a}=1} \left(\frac{1}{\#v_p}\right) \cdot \ln\left(\frac{1}{\#v_p}\right)$$

$$\text{where } \#v_p = \sum_{a \in A} v_{p,a}$$

For example, let the correct characters for p_1,p_2,p_3 = (a,b,c). The hypothesis space for each position is v_1 = {a,b,c,d}; and p_1 = a; v_2 = {b,e} and p_2=b; v_3 = {c,a,b,d} and p_3 = c. Password Entropy = Entropy([1/4 1/4 1/4 1/4]) + Entropy([1/2 1/2]) + Entropy([1/4 1/4 1/4 1/4]) = 3.4657. The character with the maximal coverage (and so minimizes reduction in hypotheses) is "d". If "d" is selected for chaff then set of hypotheses only decrease by 1. v_1 = {a,b,c,d}, v_2={b}, v_3={c,a,b,d}. However the password character for position 2 has been revealed exactly as "b". The new PasswordEntropy = [entropy([1/4 1/4 1/4 1/4]) + entropy([1]) + entropy([1/4 1/4 1/4 1/4])] = 2.7726. The Entropy drop is 3.4657-2.7726 = 0.6931. If "e", instead, is selected, then the total hypotheses decrease by 2. However now the password character in position 2 remains unknown; it is either "b" or "e". After deployment of "e" as chaff, v_1 = {a,b,c}, v_2={b,e}, v_3 = {c,a,b}. The new Password Entropy = [entropy([1/3 1/3 1/3]) + entropy([1/2 1/2]) + entropy([1/3 1/3 1/3])] = 2.8904. Entropy drop is equal to 3.4657-2.8904 = 0.5753. Although more hypotheses were invalidated, this resulted in less information gain.

Using Adversarial Chaff, it is often possible to reach a state where chaff can continue to be selected with Zero Information Gain, so leading to a discovery time which is infinite as long as Adversarial Chaff is selected each time. We observe this by noting some cases: If $\#v_p > P+C$ then any valid Response must reduce the hypothesis space by at least 1, so Information Gain will occur. However if the length of hypotheses for any position p drops below $P+C$, then it may be possible to construct Adversarial Chaff which produces 0 Information Gain for position p. Two common cases where this occurs are: (1) If $v_{p,a} = 1: a \in POS$ then a Response with any amount of chaff will not result in any ambiguity reduction for this position p, since all *POS* characters must be part of the response. Setting $Response_t$ = *POS* ∪ *CHAFF* where *CHAFF* can be any characters, will result in no further hypothesis reduction and information gain of 0 each time. (2) If $\sum_{a \in POS} v_{p,a} \leq P \wedge \sum_{b \in CHAFF} v_{p,b} \leq \frac{C}{P} \wedge b \neq a: \forall p$ then setting the Response equal to $POS \cup \{v_{p,b}\}$ will result in no further hypothesis reduction for each position p.

An attacker observing Adversarial Chaff can therefore end up stuck in a loop of observations that provide no further information. Faced with Zero Information observations, they would need to run their own "probes" with their own Responses to disambiguate from the alternative hypotheses. They would logically pursue this by using "Collusive Chaff" to maximize information gain at each iteration. Collusive Chaff uses the same Entropy calculation, but each iteration, characters are chosen that *maximize* Information Gain.

However actively probing radio email services is significantly more resource intensive than passively capturing passwords by observing challenges and responses. When passively mining passwords, the attacker just needs to issue a query against online APRS traffic, which is equivalent to reading a text file and processing the text to look for logins. They can capture an enormous amount of data. For an active probe, the attacker would need to actually issue queries to Winlink, under an FCC issued radio callsign, and both Winlink and APRS have radio bandwidth limitation mechanisms that slow the number of queries that can be issued per minute, and also halts logon attempts to the service after 3 unsuccessful challenge-response *TRIES*.

Therefore, while an attacker could pursue direct queries, it would be slower. Winlink limits login attempts to 3 queries per hour, versus passive mining where it is possible to capture 30,000 messages per hour. In summary, using Adversarial Chaff to close off direct capture of passwords from ultra-efficient, "passive mining" efforts, is useful for protecting radio operator passwords.

VIII. Simulation

A simulation was implemented to explore how the problem scales. The simulation generated a random password string, and then ran through different combinations of number of position characters *P*, number of chaff characters *C*, password length *L*, and Adversarial Chaff versus Random Chaff, recording the length of time before the password was recovered, and also if the password was recovered. If the hypothesis set was such that further chaff couldn't reduce it further, recovery was halted and recovery was reported as 0.

Longer passwords take longer to crack, as would be expected (Table II). However, as we observed in our Version Space derivation, time only grows as a function of $L \cdot \ln(L)$ (Fig. 1). For a length *L*=6 password, simulation average for discovery time was 8.5 steps. For a password length *L*=12, discovery grows only slightly to 18.9 observations.

One might think that adding more random chaff characters would slow discovery, since only *C*=3 chaff characters are being used, which leaves a lot of hypotheses being invalidated each step. However, increasing the chaff doesn't fundamentally curb the exponential speed with which hypotheses are invalidated. Increasing chaff from 3 to 10 only extends the time to recovery from 8.5 to 13.2 observations (Table III). High levels of chaff also have the side-effect that they increase the chance of accidental one-time-access. For instance, 10 chaff characters results in an expected one-time-access occurring every 25 responses. Increasing chaff from 3 to 6 keeps one-time-access probability at lower than one time in 100 and is probably a better tradeoff.

Compared to these countermeasures, Adversarial Chaff works much better for slowing discovery. For a *L*=6 password, Random chaff results in discovery in 8.5 steps. Adversarial Chaff runs for 25 steps forcing limited reduction in hypotheses, before the simulation terminates because the password hypotheses cannot be further reduced (Table IV). Thus,

Adversarial Chaff keeps hypotheses alive longer, reaches a set of final hypotheses which are ambiguous.

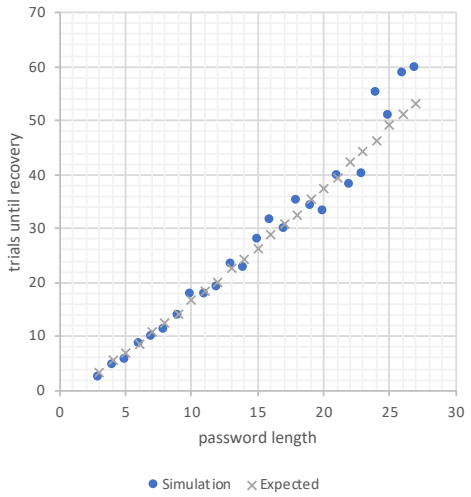

Fig. 1. The number of observations needed to recover password using a Version Space data structure for tracking viable hypotheses. Time complexity scales approximately as a function of L*ln(L); and note the curve above is just slightly worse than linear. Even long passwords can be recovered very quickly (L∈[3..30]; C=3; P=3; Chaffing Strategy = Random).

TABLE II. PASSWORD LENGTH VERSUS TIME TO RECOVER USING VERSION SPACE ATTACK (L∈[1..30], C= 3, P= 3; CHAFF = RANDOM)

L **Password length**	**E(*T*) Time to recover password**	***L*** **Password length**	**E(*T*) Time to recover password**	***L*** **Password length**	**E(*T*) Time to recover password**
3	2.2	11	17.7	21	39.7
4	4.5	12	18.9	22	37.9
5	5.5	13	23.1	23	39.8
6	8.5	14	22.5	24	55.0
7	9.8	15	27.7	25	50.7
8	11.1	16	31.4	26	58.7
9	13.6	17	29.6	27	59.6
10	17.7	18	35.1	28	53.7

TABLE III. QUANTITY OF CHAFF VERSUS TIME TO RECOVER USING VERSION SPACE ATTACK (L=6, C∈[1..10], P= 3; CHAFF = RANDOM)

C **Number of Chaff characters**	**E(*T*) Time to recover password**
1	8.6
2	7.7
3	8.5
4	9.9
5	8.7
6	9.6
7	10.7
8	11.0
9	13.1
10	13.2

TABLE IV. CHAFFING STRATEGY VERSUS TIME TO RECOVER USING VERSION SPACE ATTACK (L∈[3..20], C= 3, P= 3)

Password length *L*	**Random Chaff E(*T*)**	**Adversarial Chaff* E(*T*)**	**Password length *L***	**Random Chaff E(*T*)**	**Adversarial Chaff* E(*T*)**
3	2	1	12	19	26
4	5	5	13	23	26
5	6	11	14	23	25
6	9	25	15	28	27
7	10	25	16	31	35
8	11	18	17	30	30
9	14	20	18	35	33
10	18	17	19	34	39
11	18	35	20	33	48

* Adversarial Chaff simulation code halts when it detects that it can not reduce hypothesis space any further. The number reported above is the number of steps until this state was reached.

IX. DISCUSSION

A general solution for secure authentication over radio networks is of great interest for the 822,000 radio operators in the United States. Access to standard communication methods such as email and SMS, generally needs some kind of authentication in order to work. FCC Regulations prohibiting encryption have been in place for decades, and there is little likelihood of regulatory changes.

However new information security methods are available which would enable transmission of information in the clear, but whilst still being able to properly establish the identity of the user. One promising approach is a *Zero Knowledge Proof* [18]. In this scheme, the user shows that they can solve a problem, without revealing the details to the service or other monitoring parties. The service asks the user a question, and they answer. The information being sent is not a cipher and is sent in the clear. However only the user knows the correct answer for the question being sent by the service. The fact that one radio operator knows an answer to a question, and another does not, should not count in any way as encryption and so should be consistent with FCC regulations; in the same way that George may know my favorite sports team, but Jane may not.

There are some practical challenges involved in implementing Zero Knowledge Proofs and the Chaffing approach in this paper, in a way that amateur radio operators would be able to use in practice. Radio operators often work in poor conditions, and may not always have access to a computer to perform the calculations needed to respond to an algorithmic challenge. It is possible that a calculation feature could be added to Software Defined Radios. However, there is a much easier approach. During an emergency, smartphones probably won't be able to access the internet or send/receive calls. However, these devices, never-the-less, are superb computers with excellent power consumption, and tend to always be on their person. They could be used as a portable app for computing a response from a challenge sent over the air via radio.

We've shown that Adversarial Chaff can be employed to good effect in a domain where clear transmission is required. We've also shown that it can be used to improve security for Winlink/APRS email authentication. Clear transmission methods such as Chaffing and Zero Knowledge Proofs are a

good match for the requirements of Amateur Radio where clear transmission is required. These methods help to ensure that messages remain open and readable, and also that authentication can be performed so as to support emergency services.

REFERENCES

[1] B. Bruninga, "Introduction to APRS", APRS.Org Website, 2019, http://www.aprs.org/APRS-docs/ARTICLES.TXT. APRS

[2] A. Cheddad, J. Condell, K. Curran, P. McKevitt, "Digital Image Steganography: Survey and Analyses of Current Methods", Signal Processing, Vol. 90, Iss. 3, March 2010, pp. 727-752

[3] N. Estrada, N. Feamster, M. Freedman, "An Application Layer Covert Channel: Information Hiding with Chaffing", (1999), http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.58.4318&rep=rep1&type=pdf

[4] Amateur Radio Service, Code of the Federal Regulations, Federal Communications Commision, Part 97, Title 47, 2009, https://www.govinfo.gov/content/pkg/CFR-2009-title47-vol5/pdf/CFR-2009-title47-vol5-part97.pdf

[5] J. Larkin, "Implementation of Chaffing and Winnowing: Providing Confidentiality without Encryption", Computer Science Technical Reports No. CSBU-2006-10, Department of Computer Science, University of Bath, 2006.

[6] J. Lewis, D. Zheng, W. Carter, "The Effect of Encryption on Lawful Access to Communications and Data", Technical Report, Center for Strategic and International Studies, February, 2017.

[7] T. Mitchell, "Generalization as Search", Artificial Intelligence, Vol. 18, No. 2, 1983, pp. 203–226.

[8] R. Rivest "Chaffing and Winnowing: Confidentiality without Encryption", 1998, http://theory.lcs.mit.edu/~rivest/publications.html

[9] G. Samid, "Rivest Chaffing and Winnowing Cryptography Elevated into a Fully Fledged Cryptographic Strategy", Proceedings of the IEEE International Conference on e-Learning, e-Business, Enterprise Information Systems and e-Government, Las Vegas, July 30, 2018.

[10] S. Simai, "APRS Stations Real-Time Activity Monitor", APRS.Link website, 2019, https://aprs.link/app/aprs/statistics/aprsstations

[11] I. Wade, (ed) "APRS Protocol Reference Protocol Version 1.0", APRS Working Group, Tucson Amateur Packet Radio Corp, Tucson, 2019, http://www.tapr.org

[12] Winlink Development Team, Winlink website, 2019, http://www.winlink.org

[13] Winlink Development Team, "Guidelines on Winlink Password Change", Winlink website, 2019, https://winlink.org/WL2Kforms/passchange

[14] Winlink Development Team, "Winlink Statistics", Winlink webpage, 2019, https://winlink.org/RMSChannels

[15] Winlink Development Team, "APRSLink", Winlink website, 2019, https://winlink.org/APRSLink

[16] Winlink Development Team, "Last Voice Kuwait", WANE-TV/CBS News Documentary, 1991, https://winlink.org/content/last_voice_kuwait

[17] Winlink Development Team, "Winlink Was There… Lives and property saved, damage mitigated, Volunteer Stories", Winlink Website, 2019, https://winlink.org

[18] M. Jakobsson, K. Sako, R. Impagliazzo, "Designated verifier proofs and their applications", International Conference on the Theory and Applications of Cryptographic Techniques, 1996, pp. 143-154. Springer, Berlin, Heidelberg.