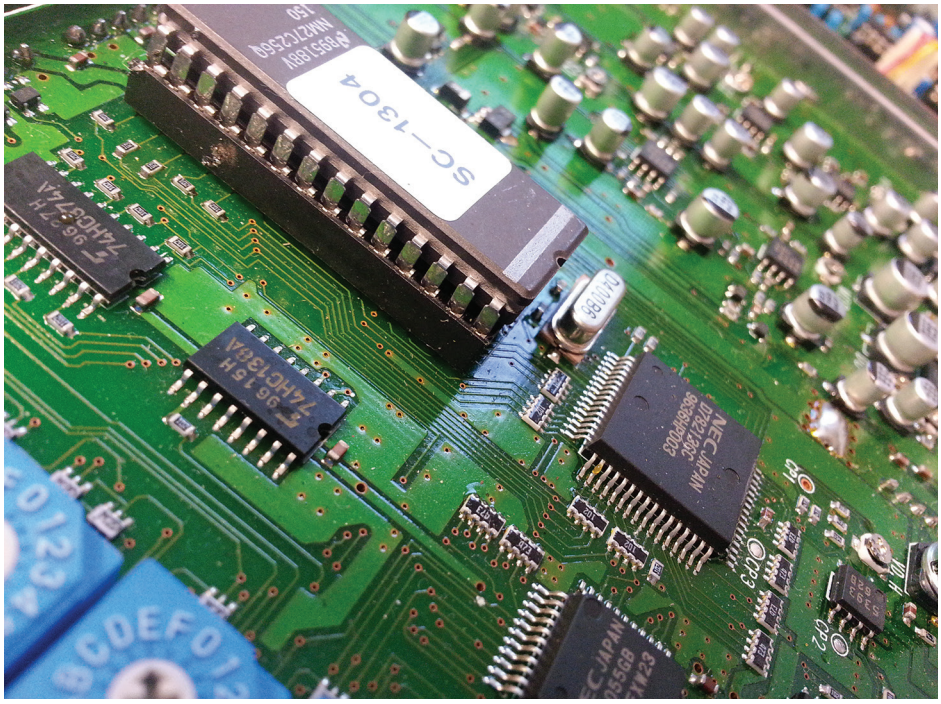
Using authentication in amateur radio

Repeaters and beacons need to be reliably controlled as part of the conditions of their licence. Good repeater sites rarely coincide with amateur's homes and are often difficult to access. An RF control link is an obvious solution – but this raises the spectre of malicious meddling with control functions. Cryptography has a solution that allows a control message to be sent reliably *and unencrypted* (so it is legal to send via amateur radio), which will also detect any attempt to imitate or repeat the message [1].

The two essential ingredients in the process are *cryptographically secure hash functions* and *multi-way handshakes*.



Part of a repeater's control logic (photo by Dave Williams, G8PUO, RILGES repeater group).

Secure hash functions

A simple analogy for a hash function is that of cooking a stew, reducing it, and then liquidising it into a soup. The aroma and taste from a spoonful tell you what the constituents are but without giving a precise description of the ingredients. The addition of a spice to the stew modifies the flavour depending on the spice used.

A hash function is used to compress data of any size down to data of a fixed size. The values generated are referred to as hashes.

At this point you may well ask 'what on earth uses a function that behaves like an electronic black hole?' The answer is *cryptography*, and for this we have to blame the beer served in the Eagle in Cambridge **[2]**. About 1967, Roger Needham and Michael Guy were discussing the dangers of storing passwords on the multi-access Titan computer they were helping to build. Over a pint of inspiration they realised that you could store the *hashes* of the passwords instead of the passwords themselves. When someone logged on, the hash of their password was calculated and compared with the list. So if a naughty person copied the passwords file, they couldn't impersonate another user because they didn't have the password. Since then, hashes have become – along with block and stream ciphers, cryptographic primitives – one of the main building blocks of cryptographic protocols.

A good secure hash function has the following properties:

- It is virtually impossible to calculate a message from its hash
- It is virtually impossible to change a message without changing the hash
- It is virtually impossible to find two different messages with the same hash
- A 1-bit change in the input affects each hash bit with a probability of exactly 0.5.

Secure hash functions – also known as one-way functions **[3, 4]** – have uses such as digital signatures, message authentication codes, and other forms of authentication. They can also be used to detect duplicate data, uniquely identify files, and as checksums to detect data corruption.

The essential feature of hash functions is that you can calculate the hash of any stream of bits or bytes and get a unique digital fingerprint of that item. If you include your own secret key as part of that input, you'll get a different fingerprint that is unique to that item *and your key*.

Hence two people who share a secret key can communicate with an extreme degree of certainty that their messages have not been subject to modification or interference, by simply appending a secure hash to the message – *but without hiding the message.* A keyed hash system called the Message Authenticator Algorithm was developed by Donald Davies and David Clayden of NPL in 1983 and became part of the ISO 8731-2 Banking standard.

Here, we are assuming the hash-based message authentication code (HMAC) method of combining a key and data, using the SHA-1 algorithm **[5]**. SHA-1 outputs a 160-bit (20 byte) hash.

Handshaking, or 'who am I talking to?'

If you've ever been on either end of a misdialled phone call you will realise how difficult it can be to identify who you are

talking to. We all use certain clues that make us trust that we know who is on the other end of the line. Tone of voice, accent, subject matter, time of day all contribute to that trust.

You may go as far as to ask the other end about some shared private information to confirm their identity. However, assuming you've been overheard, that secret must be considered to be in the public domain and cannot be used again.

Context is important too – as the conversation continues, the context will change and form a subtle backbone to the whole dialogue.

Challenge-response handshaking [6] is a method of generating trust between two parties prior to them conducting business. When there are hackers who want to subvert the process, special measures have to be taken to ensure that a high level of trust is maintained throughout the dialogue. For simplicity, each packet has the same four field format and its integrity is guaranteed by making the fourth field the keyed hash of the first three fields.

Initially, each party must challenge the other so that the response is only known to the challenger. This method uses a shared secret key along with a hash function.

Consider the example where Alice needs to tell Bob to do a task and report back on the results. Initially, Alice challenges Bob by sending a number that is used once only (known as a 'nonce' – see later) and Bob replies with the keyed secure hash of the nonce and also a nonce of his own. In this case accurate timestamps can be used as nonces.

In the data packet shown in **Figure 1** we've added a message, a digest (see later) and all three are processed to produce the Keyed Hash value for that packet.

Figure 2 shows Bob's reply, which is both a response to Alice's challenge and also a challenge to Alice. After Alice has verified Bob's response by checking the has his correct and the digest matches what she previously sent, Alice can trust Bob.

Only when Alice's second message to Bob has been verified by Bob can Bob trust Alice. He can then confidently act on the other contents of that message – which can be information, a request or a command etc.

If the first message to Bob was a replay attack (recording of a previous packet), the attack would fail because the new Timestamp 2 and hence the new Keyed Hash 2 would be different and so the attacker would not be able to create a valid message 3 packet without the secret key.

From now on, the back and forth dialogue can be taken as trustworthy as long as the messages contain fresh material (nonces) each time and are verified by keyed secure hashes. **Figure 4** shows a response to message 3.

	Field 1	Field 2	Field 3	Field 4
Alice → Bob	Timestamp 1	Message 1	Digest 1 (KH(T1))	Keyed hash 1

FIGURE 1: Format of a simple initial message from Alice to Bob.

	Field 1	Field 2	Field 3	Field 4
Bob → Alice	Timestamp 2	Message 2	Digest 2 (KH1)	Keyed hash 2

FIGURE 2: Bob's response to Figure 1.

	Field 1	Field 2	Field 3	Field 4
Alice → Bob	Timestamp 3	Message 3	Digest 3 (KH2)	Keyed hash 3

FIGURE 3: Alice's reply to Bob, establishing trust both ways.

	Field 1	Field 2	Field 3	Field 4
Bob → Alice	Timestamp 4	Message 4	Digest 4 (KH3)	Keyed hash 4

FIGURE 4: Bob replying to Alice.

The chain of digests or context in these messages confirms the sequence of the messages. When each message is sent, the sender notes what digest they expect to see in the next reply. The timestamps introduce freshness at every stage so that no packet is ever repeated, meaning that the keyed hash has to be recalculated for every packet.

The digest is a smaller version (64-bit) of its hash input. In this case the suggestion is a simple accumulator, where every input byte is added into the accumulator and the accumulator is rotated by 8 bits after each addition. Initialising the first digest field with the hash of the initial timestamp enhances the uniqueness of the chain of digests.

Nonces / timestamps

Nonces are numbers that are used once only. In cryptographic protocols they are often used as a way of distinguishing one transaction from another. In the present suggested protocol a timestamp will suffice, assuming that the time is of such accuracy that the timestamps of successive transmissions will always be different (eg a resolution of say 1 millisecond where the transmissions may take several milliseconds).

Key security

Keys, like all secrets, only remain secret for a certain length of time. In WW II, the US cipher machine the M-209 was designed for a key life of 24 hours. This meant that it was assumed that the enemy was able to deduce the key after a day and that all previous traffic was open to view.

Ofcom's latest guidance (October 2015) on encryption is in the context of RAYNET activities and insists that keys must be written in logbooks. It must be assumed that as soon as the logbooks are able to be seen, all the transmissions become public knowledge. All parties involved must understand this and tailor their messages accordingly. Also, this raises privacy issues where medical records are involved and those involved should consider extra logbooks that can be kept under lock and key.

Because the method of authentication described here does not hide the message, those rules do not and must not apply. The authentication keys *must* be kept safely locked away.

Keys should be long, preferably in excess of 64 bytes. This is especially important where the key is an ASCII string as there are fewer than 96 printable characters in each byte. It's not important that you remember the key, so a jumble of random words and numbers from a newspaper is ideal.

Keith Lockstone, M0KIL
klockstone@yahoo.co.uk

Multi-Station Use

When more than two entities are involved, the message should include identifiers (eg callsigns) showing who is transmitting and who is receiving. Both identities should be checked for correctness before the message sequence can continue. This can then be logged (electronically) by both parties. With repeaters and beacons, a Keeper and up to four other licenced, named operators are allowed to switch the unit on and off. An unlicensed person may turn the system off but is not allowed to turn it back on. This can be accomplished by assigning different levels of access to each person. Logs are also important as indicators that the system is or has been under attack.

Implementation

Who or what is Alice and who or what is Bob? So far, the assumption is that both are programs running on different computers and that they share a secret key. Even simple computers such as the Raspberry Pi have operating systems and can run multiple copies of a program simultaneously. If Alice and Bob start talking to each other at the same time then two copies of the programs will talk to each other independently. This condition has been the basis for a 'reflection attack' [7], which should to be avoided.

To avoid this schizophrenic situation, only one version of the program must run in simplex mode at either end of the link. It is further suggested that dedicated single chip microprocessors without operating systems are designated as Alice and Bob. The program should be designed to run in either Master (initiator) or Slave mode and if errors or abnormal behaviours are detected, it should return to a quiescent state.

There should be a timeout check for each reply packet so that if a 'man in the middle' decides to add a large delay in relaying the packets (eg time taken to process an attack) this can be detected and flagged up. Also, a *minimum* turn around delay should be specified, to allow transceivers to change mode.

It's best to follow Christopher Strachey's dictum: 'It is impossible to foresee the consequences of being clever, so you try to avoid it wherever you can.'

Conclusion

An Authentication Protocol has been outlined that should protect Repeater and Beacon control links from malicious interference – and in a manner that does not break the rules or the spirit of amateur radio. This scheme also has the advantage of zero running costs when compared with systems that use mobile phone (SMS or DTMF) links.

Acknowledgements

Bruce Christianson, Alex Shafarenko, M0SFR (University of Hertfordshire), Mark Lomas (Capgemini), Anthony Hodge, G4UPY, Andy Holden, M6GND, David Jardine, G0FDV, Tom Pitcher, M6ONX, Barry Pollard-Wilkins, G8DXU (Southdown ARS), Peter Hutchison, G4URT and Alastair Turner, G4RUL (Eastbourne R&EC) have all generously contributed their time and expertise to this article. However, yours truly must take full responsibility.

Websearch

[1] https://en.wikipedia.org/wiki/Replay_attack
[2] https://en.wikipedia.org/wiki/The_Eagle,_Cambridge
[3] https://en.wikipedia.org/wiki/One-way_function
[4] https://en.wikipedia.org/wiki/Cryptographic_hash_function
[5] https://en.wikipedia.org/wiki/Hash-based_message_authentication_code
[6] https://en.wikipedia.org/wiki/Challenge-response_authentication
[7] https://en.wikipedia.org/wiki/Reflection_attack