Version 7 29th December 2020

Background to Modern Cryptography

Cryptography has grown in importance due to modern means of communication and increasing security threats. Mathematical methods and powerful computers have come to the aid of both code-makers and code-breakers. Once the preserve of governments and the military, it is now available to businesses and the general public.

This paper sets out the basic principles of modern Public Key Cryptography, with a brief account of how we got here and suggestions of where we may be going. The companion paper *Pros and Cons of Cryptography* summarises the main benefits, drawbacks and costs of cryptography; *Implementing Cryptography* deals with the practical details of choice of software and creation and management of cryptographic keys; and there is also a *Glossary of Cryptographic Terms.*

Traditional Cryptography

Encryption turns a *plaintext* message, that you wish to keep secret from prying eyes, into gibberish known as *ciphertext.* Your chosen recipients can *decrypt* the ciphertext, to retrieve the original plaintext, but snoopers cannot.

Encryption and decryption use a cryptographic *algorithm* together with a *key*. Algorithms, such as Blowfish, IDEA or AES, are in the public domain whereas keys are secrets of the correspondents.

In traditional cryptography, the same key is used for encryption and decryption; this is known as a *symmetric key.* For example, in the ‘book cipher’ the correspondents agree on a book they will use as the key. The algorithm applied by the sender is “replace each word of the message by a reference to a place in the book where that word can be found”. The ciphertext is a list of such references, that only the correspondents – who know what book they refer to – can decrypt.

Over the centuries, new techniques superseded crude ciphers like the book code. The sender still used a key to encrypt the message they wanted to send, though it was now more likely to be electronic data than a book, and the recipient used that same key to decrypt the message and recover the original plaintext.

This meant that the key had to be shared by at least two people – sender and recipient – and possibly by a whole group of correspondents. It was difficult to do this without risk of keys falling into the hands of outsiders, except in tightly managed groups such as military and state institutions.

Public Key Cryptography

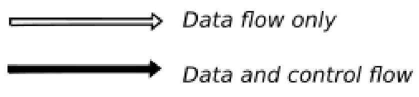
The invention of *Public Key Cryptography*[*] in the mid-1970s solved the problem. In this system, only the specific person to whom a secret message is directed has the key that gives them the power to decrypt the message; no-one else, not even the sender, can do so.

Public Key Cryptography achieves this by using **different,** but related, keys for encryption and decryption. Each participant has their own unique *key pair.* Whichever key of the pair has been used to encrypt a piece of data, that data can only be retrieved by using the **other** key to decrypt the ciphertext, not the same key that was used for encrypting the data. This removes the need for the sender and recipient of a message to share the same key.

One key of the pair is designated as the owner's *Private Key.* It is used to decrypt incoming messages and to sign outgoing messages, which only the owner of the key pair can be allowed to do. The Private Key is critical to its owner's security and must be kept secret; ideally, it should always be kept safe on the computer or device where it belongs and never copied anywhere else.

The other is the *Public Key,* which acts as a token of the key owner's *identity* in cyberspace. The owner's contacts need it to encrypt messages to the key owner and to verify signatures on messages received from the key owner, so the owner should give them copies of it. The Private Key cannot be derived from it.

The next three sub-sections give a procedural description of how the keys are used for sending secret messages securely and for verifying the authenticity and integrity of incoming messages. In the accompanying flowcharts, data and control flows are represented as follows:

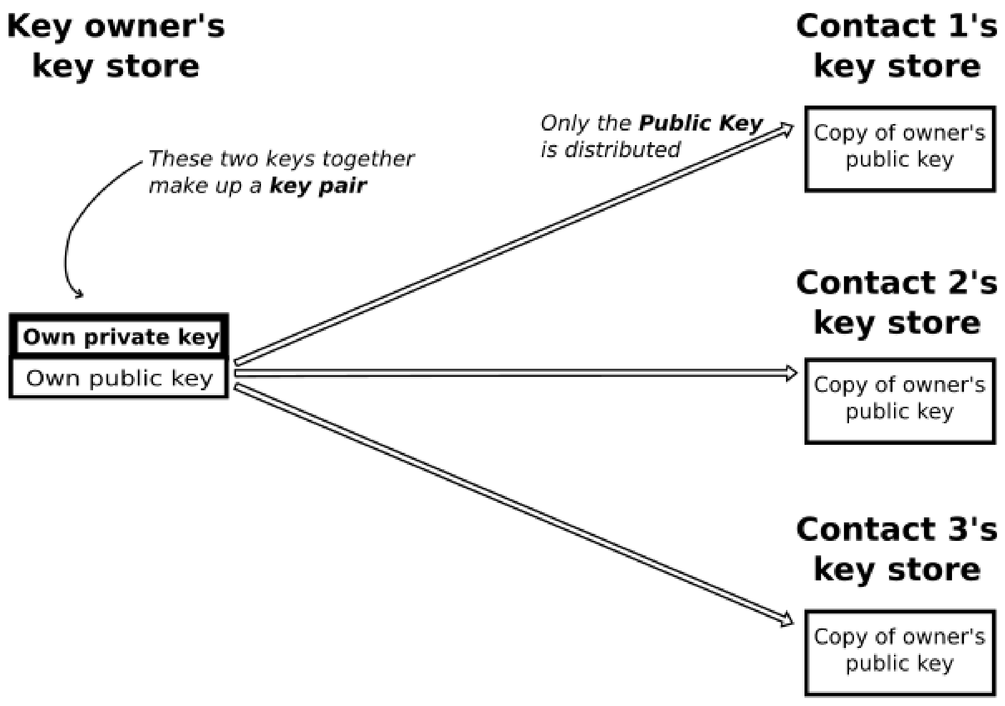


The Appendix to this document demonstrates more formally how Public Key Cryptography protects and authenticates messages so as to be proof against spies and fraudsters.

[*] For a fuller, if slightly dated, description of public key cryptography, see: https://epdf.tips/an-introduction-to-cryptography.html
Chapter 1 covers the basics; chapter 2 throws interesting light on the origins of PGP.

Distributing Public Keys

Each *key owner* distributes copies of their own Public Key to their contacts:


Key owner's key store
These two keys together make up a key pair
Own private key
Own public key
Only the Public Key is distributed
Contact 1's key store
Copy of owner's public key
Contact 2's key store
Copy of owner's public key
Contact 3's key store
Copy of owner's public key

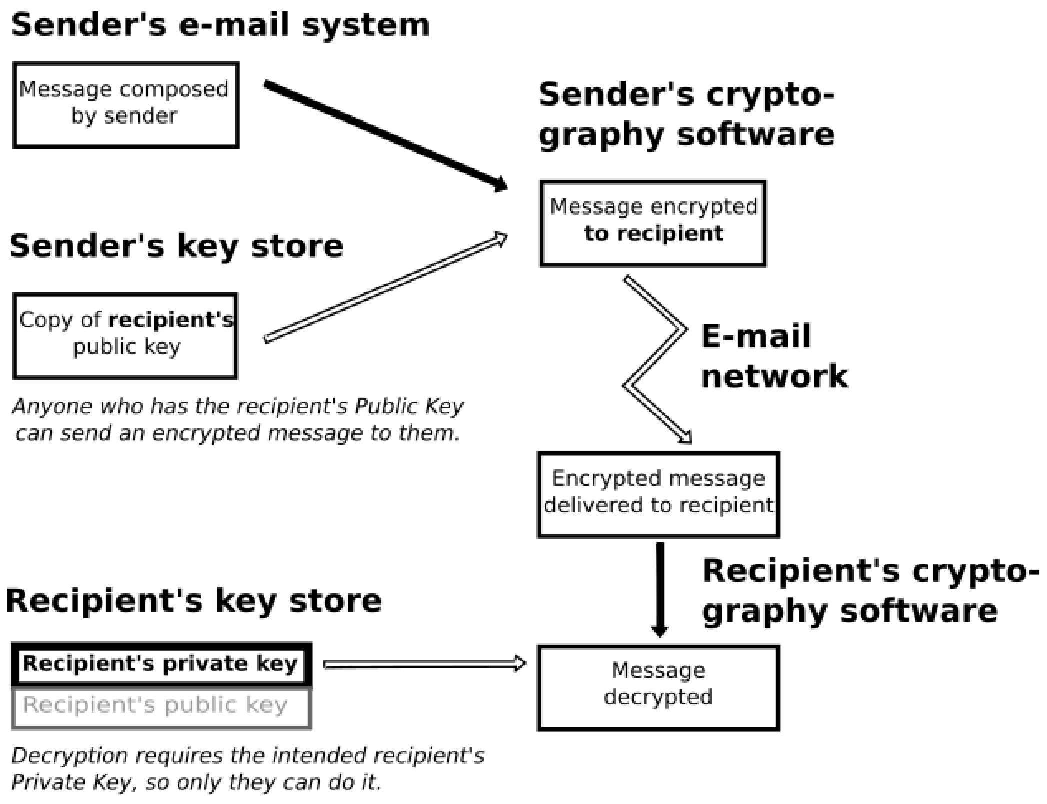
Distributing Public Keys

The key owner will receive copies of their contacts' Public Keys in the same way and will store them in their own key store along with their own keys.

Encrypting Confidential Messages

Encryption protects confidential messages from snoopers. The sender can encrypt the message, before sending it, using the intended recipient's Public Key* – a copy of which the recipient will have distributed to them as shown above. The recipient can decrypt the message by using their own Private Key.

* In practice, the sender's software generates a one-time symmetric key and uses that to encrypt the message content. It sends this one-time key, itself encrypted using the recipient's Public Key, to the recipient along with the encrypted message content. The recipient's software decrypts the one-time key by using their own Private key and then uses it to decrypt the message content. This is just a performance optimisation and is equivalent to using the Public Key directly.


Sender's e-mail system
Message composed by sender
Sender's crypto-graphy software
Message encrypted to recipient
Sender's key store
Copy of recipient's public key
Anyone who has the recipient's Public Key can send an encrypted message to them.
E-mail network
Encrypted message delivered to recipient
Recipient's crypto-graphy software
Recipient's key store
Recipient's private key
Recipient's public key
Message decrypted
Decryption requires the intended recipient's Private Key, so only they can do it.

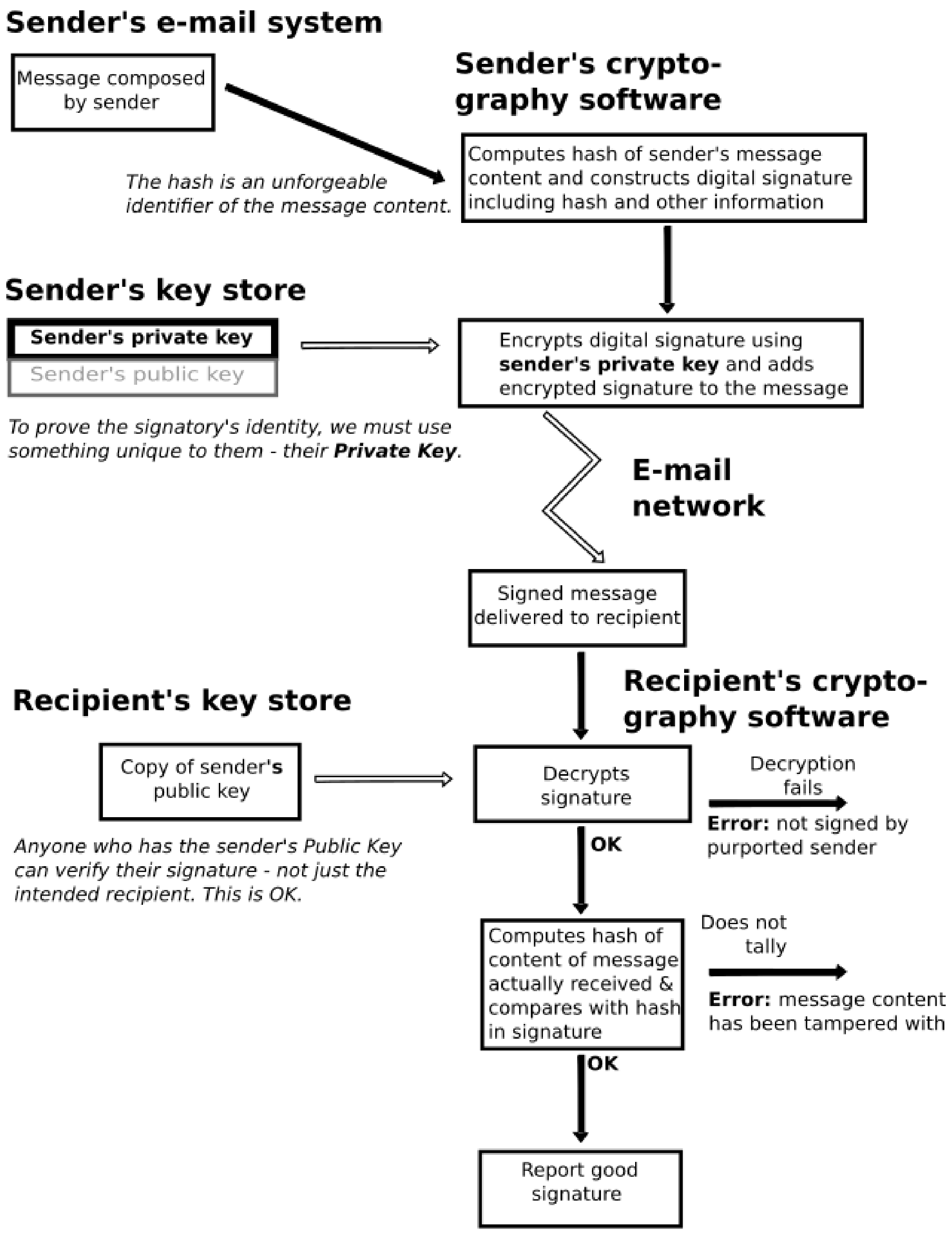
Sending an Encrypted Message

Signing and Verifying Messages

The author of a message can *digitally sign* it to prove that it was written by them – just like signing a letter – and also to guard against forgers altering the message content in transit.

The author's software first computes a *hash* of the message content; this identifies the content, as sent. The hash forms part of the signature, which is then encrypted using the author's Private Key to protect it from tampering and tie it uniquely to the author.

The recipient's software decrypts the signature using the author's Public Key, to check the author's identity. It also computes a hash of the received message content and checks that it matches the hash of the sent content given in the signature.


Sender's e-mail system
Message composed by sender
Sender's crypto-graphy software
The hash is an unforgeable identifier of the message content.
Computes hash of sender's message content and constructs digital signature including hash and other information
Sender's key store
Sender's private key
Sender's public key
Encrypts digital signature using **sender's private key** and adds encrypted signature to the message
To prove the signatory's identity, we must use something unique to them - their **Private Key**.
E-mail network
Signed message delivered to recipient
Recipient's key store
Recipient's crypto-graphy software
Copy of sender's public key
Decrypts signature
Decryption fails
Error: not signed by purported sender
Anyone who has the sender's Public Key can verify their signature - not just the intended recipient. This is OK.
OK
Computes hash of content of message actually received & compares with hash in signature
Does not tally
Error: message content has been tampered with
OK
Report good signature

Signing and Verifying a Message

Usually, cryptographic messages are both digitally signed and encrypted. Some authorities advise signing, encrypting and signing again to avoid a potential flaw[*].

Adoption of Public Key Cryptography

Public Key Cryptography is used for secure web-sites (where you see a padlock icon in your browser's address bar), for code signing and for secure messages. The public, as well as state institutions, have been able to use it since Phil Zimmermann published the PGP (Pretty Good Privacy) program in the 1990s. After a run-in with the US Department of Justice, it became generally available and, according to reports[†], neither the UK's GCHQ nor the USA's NSA has – as yet – cracked it.

Secure web-sites caught on rapidly, driven by e-commerce companies' need for secure transactions over the world-wide web. Most e-businesses now use them.

Secure messaging by e-mail did not take off as the originators had hoped. Perhaps people saw no real need for it, or were waiting for others to adopt it first so that they would have someone else to talk to. Development of cryptography support for secure e-mail slowed down; it was still based largely on the 1990s versions and has been criticised, e.g. by Green[‡] and Marlinspike[#], for not keeping up-to-date with changing technology and expectations. Complaints include:

- clumsy key management and does not support *forward secrecy;*
- lack of automatic key synchronisation across devices, so user has to copy them manually;
- obscure trust model so authentication is often not done properly;
- difficulty of adding desired features without a total redesign; and
- complex and old-fashioned *user interface,* requiring either training or recourse to bulky and unreadable documentation.

---

* A signed message after forwarding still bears its original author's signature, not the forwarder's. See Davis: *Defective Sign & Encrypt in S/MIME, PKCS#7, MOSS, PEM, PGP, and XML* (http://world.std.com/~dtd/sign_encrypt/sign_encrypt7.html).

† Edward Snowden.

‡ See Matthew Green: *What's the Matter with PGP?* https://blog.cryptographyengineering.com/2014/08/13/whats-matter-with-pgp/

# See Moxie Marlinspike: *Blog >> GPG and Me* (https://moxie.org/2015/02/24/gpg-and-me.html). He is criticising GnuPG, a competitor to his company's Signal product.

Cryptography in the Age of Social Media

A revival of consumer demand came from the world of social media. Users wanted secure communication, to keep their activities secret from snoopers or from the police or security services; they wanted it on mobile devices; and they wanted to use it as an appliance, not needing a high degree of IT skills.

To meet this demand, new cryptographic applications (see Appendix III of the *Implementing Cryptography* paper) were developed to make secure communication easier to use and more modern in style by combining:

– instant messaging etc. over the internet – familiar to Social Media users – and other features that users want; and

– Public Key Cryptography – as described earlier, but with a modernised user interface and extended cryptographic support.

Some of these applications provide robust cryptography suitable for serious business and professional users, for whom high security is a priority; others are aimed at the recreational market and provide popular add-ons but less emphasis on security.

Each application comes as part of a proprietary *platform;* to use it, you sign up for an account, which lets you communicate with other users of that platform. You can extend your circle of contacts by opening accounts on several platforms. This is less open than e-mail, which let you communicate freely with anyone else, anywhere on the internet, using any e-mail software.

Other possible drawbacks are that instant messaging lacks the quoting and threading features of e-mail and could make document control more difficult.

Multi-device Support

Many users now own more than one device – for example desktop computer, laptop computer, tablet computer or mobile phone. They may use them in different places and at different times – for instance send an enquiry from a desktop computer at work, receive a reply by mobile phone on the train and send a follow-up query from a laptop at home. They want this conversation to be as seamless as if they were conducting it from a single device in a single place.

To do this, each device's key store must have a copy of your own Private/Public key pair and your correspondents' Public keys. In early Public Key Cryptography, you had to propagate them manually from device to device but this was tricky. The new applications synchronise keys between devices automatically.

Forward Secrecy

Forward secrecy can protect you against leakage of commercially sensitive, compromising or embarrassing information, that you probably did not know – or had forgotten – was on your computer. The biggest risk is often from working copies of data, that computers make without your knowing as it passes through the system; tracking down and purging all such copies is next to impossible.

The cryptography software does not attempt to purge this data; instead, it keeps each item of data encrypted with its own unique, one-off or short-term key. Once the key has expired, no-one can ever decrypt the data again.

In an online conversation with forward secrecy, the software generates a unique key for each successive message. Even if an attacker cracks the key for one message, previous messages remain secure.

Forward secrecy is not a water-tight guarantee of security. If the sender's device has been infected with malicious software that can spy on the information before it has been encrypted, or the recipient's device with software that can stash away a copy of the information after it has been decrypted, the attacker can still obtain a copy of it; and there is no way to stop the recipient taking a photograph of their screen while the information is on display.

Cryptographic Support for New Application Features

To support these features, applications need extensions to the basic cryptography described in the Public Key Cryptography section. You still have a key pair, comprising a Private and a Public Key, which is created automatically by the software when you sign up to the platform:

– the Private Key is secret to you and never leaves the device on which it was created; and

– the Public Key has to be available to your contacts on the platform. Your software publishes it to them via a key store, which may be centralised or distributed according to the application.

But the new features also require short-term dynamic keys. To implement Forward Secrecy, for example, you need a unique key for encrypting each message. The Signal protocol[*] software does this by combining your Public Key – which authenticates the message as being from you – with a one-time ‘pre-key’.

In addition, the user interface has been modernised by a method of creating and managing keys behind the scenes, more in keeping with the social media model (see *Implementing Cryptography* section TOFU).

The Future of Cryptography

Cryptography has always been at risk of keys being cracked, so making the current cryptographic algorithms useless. This is a particular concern now because of the development of powerful Quantum Computers.

Theoretically, the widely used RSA algorithm is vulnerable to cracking but this is impracticable by classical methods. However it is feasible – though difficult and expensive – using quantum methods, that could be within the reach of large organisations such as governments. A defence[†] has been proposed but, while cheap compared with the attack, it is beyond the reach of normal users.

There is also active research into new communication techniques, that exploit the quantum properties of novel materials and so provide security that would be very hard for an attacker to defeat.

Conclusions

1. The theory behind Public Key Cryptography is sound. The method is in wide use and, so far as known, has not yet been cracked. There is a risk that it may be rendered useless by advances in code-cracking. We can only hope that cryptographers continue to keep ahead of the crackers.

2. The social media based applications are gaining ground, among recreational and business users, over the earlier PGP based ones. They have a superb solution for multi-device usage, partly cure the usability difficulties of the older software (though perhaps creating some new ones) and support forward secrecy (which may not be relevant to us).

[*] See the video presentation at: https://www.youtube.com/watch?v=7WnwSovjYMs.

[†] See Bernstein, Heninger, Lou & Valenta: *Post-quantum RSA* https://cr.yp.to/papers/pqrsa-20170419.pdf.

However, users may need to sign up for several different platforms and install the appropriate software for each, in order to communicate widely on the internet; and the social media applications do not, at present, cater for some likely needs of business users – for example, integrated e-mail support (or perhaps something better).

The problems are soluble but whether, when and how they will be solved is not clear. Interfaces between rival platforms could be standardised (but this may be unlikely in the competitive business environment); and e-mail support has been mentioned (but not, so far as I know, implemented).

3. Whichever of these styles of cryptography is adopted, the security that it provides depends on:

   - guarding the secrecy of the Private Key;

   - not allowing interlopers to gain access to a password-protected session on an unattended device; and

   - verifying that the Public Keys received for use in communicating with contacts really belong to those contacts.

Richard Stonehouse

Appendix – Why Public Key Cryptography Works

The Basic Rules

Private and Public Key work together according to the rules:

1. you do not share your Private Key with anyone. You may share your Public Key freely, but the people you give it to cannot use it to find out your associated Private Key; and

2. whichever key was used to encrypt something, it can only be decrypted by the **other** key of the **same** pair – not by the key used for encrypting it or by a key from any other key pair.

You want …

… your intended correspondent to be able to:

– decrypt a ciphertext message that you have sent them; or

– sign a message that they send you, to prove they are the author,

… but an interloper **not** to be able to:

– decrypt a ciphertext, sent by you to your intended correspondent but eavesdropped on by the interloper; or

– impersonate a correspondent you trust by signing a message, from themself to you, as having come from that correspondent.

Which implies that …

… decryption and signing must depend on something that your intended correspondent has but the interloper does not.

Therefore …

… they must require your intended correspondent’s Private Key (rule 1).

Which implies that …

… the other end of the transaction, performed by you:

- encrypting a plaintext message into ciphertext before sending it to your correspondent; or

- verifying a signed message after receiving it from them,

must be done by you using the counterpart of your correspondent's Private Key, i.e. their Public Key (rule 2).

Why can you do this?

Because you have a copy of their Public Key.

Can anybody else do the same?

Yes. (Provided they have a copy of the necessary Public Key).

Can a fraudster …

… pass off a fake message as being from one of your genuine contacts by:

- intercepting one of your contact's genuine messages, changing its content and sending the altered message on to you; or

- writing a message to you but, instead of signing it in their own name, copying a signature from one of your contact's messages?

No. You will know it is a fake because the hash in the signature will not match that of the content actually received.

Can they get round this by …

… falsifying the signature as above, but then making it match the fake message content by inserting a hash computed from that content?

No. The original author's signature is protected; it is encrypted using their Private Key. The fraudster can decrypt it using the author's Public Key and substitute the hash, but cannot re-encrypt the altered signature because they do not have the author's Private Key.