Failures of security proofs

Examples and recommendations

Chris J Mitchell

Royal Holloway, University of London
United Kingdom

me@chrismitchell.net
www.chrismitchell.net

15th January 2021

1. Introduction: Security proofs

- Modern cryptographic practice depends to a huge extent on proofs of security.
- New cryptographic primitives and protocols are routinely required to have a security proof, relating security to a ‘hard’ computational problem.
- While this is not a perfect approach, since ‘hard’ problems can sometimes found to be not so hard:
  - in new computing paradigms, or
  - if new algorithms are found for existing paradigms,

  it has undoubtedly reduced the likelihood of the adoption of fundamentally flawed schemes.
- The development of the complexity-based security models in which security proofs are formulated has undoubtedly been a major step forward in cryptography.

Issues

- In practice, the situation is not as rosy as it might be.
- There are many documented cases where ‘proven secure’ schemes turn out to be insecure.
- In this talk we will explore a range of examples of how and why proofs of security have failed, and what lessons we might learn for the future.

What can go wrong?

- Some of the key ways in which security proofs go wrong are:
  - *No security model* — 'theorems' and 'proofs' are given outside the context of a well-defined model;
  - *Wrong security model* — the model is not appropriate for the mechanism or protocol;
  - *Inadequate security model* — the model does not (fully) capture the security properties required;
  - *Erroneous 'proofs'* — the proof of security is incorrect.

Examples and analysis

- In the remainder of this talk we will look at examples of each of these types of failure.
- This then enables us to draw conclusions, including suggesting why things have gone wrong and what might be done to improve matters in the future.
- A caveat: this talk focuses on complexity-theoretic proofs of security, and does not address logic-based approaches to proving security properties of protocols.
- The issues in the latter case are different, notably including capturing the important but subtle features of the use of cryptography.

2. No security model: The simple case

- Sadly I am sure we have all seen cases like this.
- Even though the security model paradigm has been in widespread use for over 20 years, many papers are still being published which make no attempt to offer a security proof even when that would be appropriate.
- Of course, there are cases where no proof can be expected, e.g. for symmetric cryptographic primitives such as block and stream ciphers where provably secure schemes are not really practical.
- Sadly, some (in other respects highly reputable) non-security-specialist conferences and journals are particularly guilty of publishing papers containing schemes which lack security proofs, and the schemes are, in many cases, obviously insecure.

A case study

- To demonstrate the problem, and purely for the purposes of this talk, I took a quick look at a very recent journal issue — the October 2020 issue of *Wireless Networks* (volume **26 #7**), a long-established journal published by Springer.
- I chose this journal because I have seen many flawed security papers appear in its pages previously.
- This issue contains two security protocol papers.
  - (Chilveri and Nagmode, 2020) proposes an ECC-based authentication protocol. No security proof is provided — indeed there is almost no security analysis at all.
  - (Moazami and Safkhani, 2020) proposes an ownership transfer protocol. Brief but unconvincing discussions of claimed ‘proofs of security’ using BAN and Scyther are given.

Are the schemes secure?

- The absence of any security analysis whatever for the Chilveri-Nagmode scheme suggests it is likely to be vulnerable.
- The other day I had a closer look at the Chilveri-Nagmode scheme, and sadly there is so much undefined notation that it is impossible to work out what is being presented; moreover the English is very poor and there are numerous trivial typos. I can only conclude that none of the referees actually tried to read the paper.
- If time allows before the talk I will look at the Moazami-Safkhani scheme, which I am also rather dubious about.
- However, this is not the real point — the main issue is that even today, unproven (and hence potentially vulnerable) schemes are being widely published in major venues.

How can this be?

- Why do all these papers get published?
- On the supply side — clearly many academics working on cryptography have not mastered the necessary tools.
- At the consumer end — many non-specialist journals and conferences do not have security specialists on their editorial boards, and hence do not ensure papers are refereed by knowledgeable experts.
- Also, sadly, it is often difficult to get appropriate experts to review papers.

Does it matter?

- I am sure many would argue that it doesn't really matter, since neither of the example papers would be accepted by a respectable security-specialist journal or conference.
- Also, they might argue that it is not their job to 'police' publications.
- In response, I would say:
  - it is highly damaging to the literature as a whole that suspect papers are routinely published — especially if the schemes then get used;
  - it is precisely our job to police the academic literature — nobody else will, and if the reputation of the literature is damaged then so is ours as academics;
  - many readers and future authors will take the work seriously and then publish more papers in a similar vein, thus making the problem worse.

What can we do?

- If we all agreed to review a couple of these ‘suspect’ papers every year, then the problem would be pushed into the fringes (where it belongs).
- Reviewing a paper without a security model and proof where such a proof is needed is easy — it can simply be rejected with a short explanation!
- Reviewing apparently good papers is where the hard work is . . .

3. Wrong security model

- (Xia et al., 2020), presented at ICICS 2019, describes a ‘provably secure group authentication [protocol] in the asynchronous communication model’.
- However the protocol is subject to a serious attack, as shown in (Mitchell, 2020).
- Examination of the security theorem provided in (Xia et al., 2020) reveals that it is not exactly what it seems to be at first sight.

The scheme: security goals

- Here a *group authentication protocol* is one where ‘each user acts both roles of the prover and the verifier, and all users in the group are authenticated at once’ (Xia et al., 2020).
- Main goal is to assure members of a group that all members and no other parties are actively involved in the protocol.
- ICICS paper refers to both insider and outsider attacks, i.e. protocol is intended to be secure against both; also claims an outside adversary cannot impersonate a group member without detection, even if it computes a token after seeing all other tokens (communication assumed to be asynchronous).
- However, no reference to trust assumptions for broadcast channel used for communications, apart from being asynchronous — it is standard practice when analysing authentication protocols to assume attacker can manipulate the communications channel, including to intercept, delete, insert and modify messages (see, for example, (Boyd et al., 2020)) — we therefore assume this here.

The scheme: Overview

- Scheme can be divided into two phases:
  - *initialisation*, when the *Group Manager* (GM) equips each participant with the credentials needed to perform group authentication, and
  - the *group authentication phase* where a subset of the participants simultaneously authenticate each other as a group.
- Suppose that there are n participants $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$.

The scheme: Initialisation

The GM chooses/computes:

- parameters t and ℓ, where at most $t-1$ insiders collaborate, and ℓ is the number of sessions with these credentials.
- cyclic group G (expressed multiplicatively) with order a large prime q, and randomly selects $g_1, g_2, \ldots, g_\ell$ to be ℓ independent generators of G (one per session).
- cryptographic hash function H with domain G.
- secret $s \in \mathbb{Z}_q$, and the ℓ values $H((g_i)^s)$, $1 \leq i \leq \ell$.
- secret polynomial $f(x) = \sum_{i=0}^{t-1} a_i x^i$ over $\mathbb{Z}_q$ of degree $t-1$, where $a_0 = s$.
- credential $s_i = f(x_i)$ for each participant U_i ($1 \leq i \leq n$), where $x_i \in \mathbb{Z}_q$ is a unique identifier for U_i.

Using an out-of-band secure channel, GM equips participant U_i ($1 \leq i \leq n$) with t, G, q, H, the identifiers $\{x_1, x_2, \ldots, x_n\}$, the generators $\{g_1, g_2, \ldots, g_\ell\}$, the hash codes $\{H((g_1)^s), H((g_2)^s), \ldots, H((g_\ell)^s)\}$, and the participant's secret credential $s_i (= f(x_i))$.

The scheme: Group authentication: Notation

- Suppose some subset $\mathcal{U}' \subseteq \mathcal{U}$ of the participants (where $|\mathcal{U}'| = m \leq n$) wish to authenticate each other in a group-wise fashion, where $\mathcal{U}' = \{U_{z_1}, U_{z_2}, \ldots, U_{z_m}\}$.
- Suppose every participant in $\mathcal{U}'$ is aware of the membership of $\mathcal{U}'$.
- Also suppose that the set of participants has reached session number σ during use of a particular credential set, where $1 \leq \sigma \leq \ell$; each session must be conducted using a new value of σ, and σ determines which generator g_σ from the set of generators will be used in this particular protocol instance.

The scheme: Group authentication: Steps

Each participant $u_{z_i} \in \mathcal{U}'$ proceeds as follows.

- Choose $u_{z_i} \in \mathbb{Z}_q$ uniformly at random, and broadcasts it.
- Once the values $\{u_{z_1}, u_{z_2}, \ldots, u_{z_m}\}$ received, compute:

$$\gamma_i = \prod_{\substack{j \in \{1,2,\ldots,m\} \\ z_j < z_i}} (g_\sigma)^{u_{z_j}} \prod_{\substack{j \in \{1,2,\ldots,m\} \\ z_j > z_i}} (g_\sigma)^{-u_{z_j}},$$

$$L_i = \prod_{\substack{j \in \{1,2,\ldots,m\} \\ z_j \neq z_i}} \frac{x_{z_j}}{x_{z_j} - x_{z_i}},$$

and

$$c_{z_i} = (g_\sigma)^{s_{z_i} L_i} (\gamma_i)^{u_{z_i}}.$$

- Broadcast c_{z_i} to all members of $\mathcal{U}'$.
- Once values $\{c_{z_1}, c_{z_2}, \ldots, c_{z_m}\}$ received, compute $\prod_{r=1}^{m} c_{z_r}$.
- If $H(\prod_{r=1}^{m} c_{z_r}) = H((g_\sigma)^s)$ then all users authenticated.

Analysis: Preliminary observation I

- We consider what can be learnt by observing a single value c_{z_i} in a single instance of the protocol, together with the initial broadcasts of the values $\{u_{z_1}, u_{z_2}, \ldots, u_{z_m}\}$.
- We suppose that the (outside) observer has access to the system parameters, i.e. the values provided by the GM to all participants, namely:
  - t, G, q, H,
  - the identifiers $\{x_1, x_2, \ldots, x_n\}$,
  - the generators $\{g_1, g_2, \ldots, g_\ell\}$, and
  - the hash codes $\{H((g_1)^s), H((g_2)^s), \ldots, H((g_\ell)^s)\}$.

Analysis: Preliminary observation II

- By definition:

$$c_{z_i} = (g_\sigma)^{s_{z_i} L_i} (\gamma_i)^{u_{z_i}}.$$

- Again by definition:

$$\gamma_i = \prod_{\substack{j \in \{1,2,\ldots,m\} \\ z_j < z_i}} (g_\sigma)^{u_{z_j}} \prod_{\substack{j \in \{1,2,\ldots,m\} \\ z_j > z_i}} (g_\sigma)^{-u_{z_j}},$$

- I.e. computing γ_i does not involve any secret credential values and hence can be derived by anyone with the system credentials.
- If u_{z_i} is intercepted, the observer can thus compute

$$c_{z_i} \cdot (\gamma_i)^{-u_{z_i}} = (g_\sigma)^{s_{z_i} L_i}.$$

- Yet again by definition

$$L_i = \prod_{\substack{j \in \{1,2,\ldots,m\} \\ z_j \neq z_i}} \frac{x_{z_j}}{x_{z_j} - x_{z_i}},$$

so L_i is also available to anyone with the system credentials.

Analysis: Preliminary observation III

- Having derived L_i, the observer now computes a value M such that $ML_i \equiv 1 \pmod{q}$, a calculation which is simple to perform given that q is known. Note that M is guaranteed to exist since q is prime.
- It follows immediately that

$$[c_{z_i}.(\gamma_i)^{-u_{z_i}}]^M = (g_\sigma)^{s_{z_i} L_i M} = (g_\sigma)^{s_{z_i}}.$$

- That is, an observer of c_{z_i} and the values $\{u_{z_1}, u_{z_2}, \ldots, u_{z_m}\}$ can compute $(g_\sigma)^{s_{z_i}}$, where s_{z_i} is the secret credential for user u_{z_i}.
- Moreover, the only occasion s_{z_i} is used in the protocol is to compute $(g_\sigma)^{s_{z_i}}$, i.e. knowing $(g_\sigma)^{s_{z_i}}$ is essentially equivalent to knowing s_{z_i}, at least for this session.

An outsider impersonation attack: Scenario

- The above observation leads to a very simple and powerful attack, enabling impersonation of a participant in any group.
- Suppose an (outsider) adversary controls the broadcast channel with respect to 'victim' participant U_{z_i}, i.e. the adversary can:
  - prevent messages sent by other legitimate participants from reaching U_{z_i}, and
  - send messages to U_{z_i} on this channel that appear to have come from other legitimate participants.
- Also assume that it is 'time' for a session using the group generator g_σ.

An outsider impersonation attack: Gathering data

- Suppose the adversary observes a group of participants $\mathcal{U}'' \subseteq \mathcal{U}$ (where $U_{z_i} \notin \mathcal{U}''$) engaging in the protocol.
- The adversary:
  - intercepts all the u_{z_j} and c_{z_j} values sent by each $U_{z_j} \in \mathcal{U}''$;
  - uses these intercepted values, together with the system parameters, to compute $(g_\sigma)^{s_{z_j}}$ for each $U_{z_j} \in \mathcal{U}''$;
  - prevents any of the messages reaching U_{z_i} (since these messages are not intended for U_{z_i}, U_{z_i} should ignore them anyway).

  That is, the adversary now knows information equivalent to the secret credentials for all participants in $\mathcal{U}''$ for session σ

An outsider impersonation attack: Completion

- Suppose that the adversary persuades the victim U_{z_i} that it is being invited to join a group of participants $\mathcal{U}' \subseteq \mathcal{U}'' \cup \{U_{z_i}\}$, where $U_{z_i} \in \mathcal{U}'$, e.g. by sending 'fake' messages from members of $\mathcal{U}'$ to U_{z_i}.
- Adversary chooses arbitrary values u_{z_j} for every $U_{z_j} \in \mathcal{U}' - \{U_{z_i}\}$, and sends these values to U_{z_i} as if they come from U_{z_j}.
- Once U_{z_i} sends its value u_{z_i}, the adversary can use the complete set of values $\{u_{z_j}\}$ and the computed values $(g_\sigma)^{s_{z_j}}$ (which it has for every $U_{z_j} \in \mathcal{U}' - \{U_{z_i}\}$) to compute the 'correct' values c_{z_j} for every $U_{z_j} \in \mathcal{U}' - \{U_{z_i}\}$, which it sends to the victim participant U_{z_i}.
- Since all the received values are 'correct', the victim will falsely believe that it is part of a group authentication with a set of participants, of whom none believe they are being authenticated to the victim.

Other attack scenarios

- There are many other scenarios that could be used to launch an attack on the protocol.
- For example, if an attacker could control the broadcast network with respect to two victims, a range of conflicting beliefs about who has been authenticated to whom could be established.
- That is, once an attacker has observed a participant U_{z_j} output a value c_{z_j}, this can be used to impersonate U_{z_j} in any group the attacker chooses (assuming control over the broadcast channel).

But there is a proof of security ... I

- The attack described above clearly breaks the claimed ‘no impersonation’ property.
- Theorem 4 of (Xia et al., 2020) states that ‘The proposed group authentication scheme satisfies the no impersonation property, assuming that H is a preimage resistant hash function and the DDH assumption holds in G’.
- The attack does not invalidate the assumptions of the theorem, and hence the theorem must be false.

But there is a proof of security ... II

- How can this be true?
- Examination of the proof of Theorem 4 suggests why.
- The sort of manipulation of messages and beliefs involved in the attack do not appear to be covered by the proof.
- That is, while the mathematics may be correct, the result does not establish that the protocol would actually be secure in a real-world deployment (which, of course, it would not).

But there is a proof of security ... III

- This issue is (partly) admitted in (Xia et al., 2020).
- In the concluding section it is stated that:
  - ‘There are two distinct approaches to defining security for cryptographic protocols: simulation proof and reduction proof.
  - The former is more intuitive because it models security of the targeted problem via an ideally trusted third party. However, the definitions will become complicated once all details are filled in.
  - In contrast, the reduction proof yields definitions that are simpler to describe and easier to work with. However, the adequacy for modelling the problem is less clear. In this paper, we followed the latter approach, and it is still open how to provide formal security treatment for group authentication using the simulation proof.’
- Of course the final claim is questionable; there are well-established security models for analysing group security protocols — see, e.g., §2.7.1 of (Boyd et al., 2020).

Conclusions

- The fundamental flaw exists despite the fact that theorems are provided asserting its security.
- In fact the authors admit that the security model used is not sufficient to establish security other than in a special case — perhaps this was missed by reviewers?
- This clearly suggests that reviewers need the time to carefully review precise details of claims of security.
- This flies in the face of the modern obsession with speedy publication, both for conferences and many journals (e.g. *IEEE Access* allows referees only a week to complete a review, making it little more authoritative than a preprint site like arXiv).
- Perhaps we, as the research community, need to think more carefully about finding ways to allow reviewers time and space to write carefully considered and detailed reviews.

4. Inadequate security models

- One major problem in applying the provable security technique is in ensuring that the security model used captures all the desired properties of the scheme.
- It is particularly problematic when real-world protocols employ specially-designed modes of operation for reasons of efficiency or usability.
- A ‘nice’ example of this is provided by SSH, as we next describe (although there are plenty of other examples in the literature, as we briefly discuss later).
- Much of the discussion here is based on the excellent magazine article (Degabriele et al., 2011).

SSH: a quick introduction

- SSH (Secure Shell) provides a secure communications channel.
- It was designed in the mid 1990s as a replacement for Telnet and other unsecured remote shell protocols.
- The main objective was to hide secret information, such as passwords, which were previously sent in cleartext.
- The SSH Binary Packet Protocol (BPP) (see RFC 4253) is the part of SSH responsible for message encryption — it uses a special purpose *encode-then-encrypt-then-MAC* construction.

BPP — details

- The message payload is first encoded and then encrypted.
- The MAC is calculated on a combination of the encrypted message and a sequence number.
- Encoding adds three fields: a packet-length field, a padding-length field, and some padding bytes.
- The packet-length field specifies the combined length of the padding-length field, the payload message, and the padding field.
- It is encrypted to protect against traffic analysis — this has a significant effect on protocol security.

Formal analysis

- (Bellare et al., 2004) provides a formal analysis of the SSH BPP.
- Because of a distinguishing attack due to Dai that exploits the use of initial packet chaining when using CBC mode encryption, (Bellare et al., 2004) does not directly prove security for SSH BPP as defined in RFC 4253.
- Instead, it proposes several minor SSH BPP variants, and proves them secure in an extended version of the IND-CCA model, where this new model takes into account the stateful nature of SSH decryption.

An attack!

- However, SSH BPP turns out to be vulnerable to attack, despite having a mathematically correct security proof.
- (Albrecht et al. 2009) describes plaintext-recovery attacks that exploit the use of an encrypted packet-length field, its reliance on CBC mode, and the attacker's ability to send ciphertext data in small chunks and observe how the recipient reacts.
- These and other attacks rely on the fact that the recipient decrypts and acts upon the decrypted data prior to verifying the MAC.

Attack details

- An attacker observes a ciphertext and chooses one block to attack.
  - The attacker sends this target block to the recipient in such a way that the recipient interprets it to be the start of a new packet.
  - The recipient must immediately decrypt this block to retrieve the packet-length field, to know how much data it must wait for before it receives and verifies the MAC.
  - The attacker then proceeds by sending random blocks one at a time until the recipient outputs a MAC error.
  - By counting how many random blocks have been sent, the attacker can deduce the new packet's packet-length field and, by the properties of CBC mode, deduce the corresponding bits in the target plaintext block.
  - In practice, this attack is complicated by checks performed on the packet-length field once the recipient recovers it.
- This attack can be applied to one of the provably secure variants of the SSH BPP proposed by (Bellare et al., 2004).

What went wrong?

- (Bellare et al., 2004) implicitly assumes that ciphertexts are self-describing in terms of their lengths.
- In reality, recipients must decrypt the first block of a packet as soon as they receive it to obtain the packet length.
- RFC 4253 actually states that implementations SHOULD decrypt the length after receiving the first 8 (or cipher block size, whichever is larger) bytes of a packet.
- Also, the (Bellare et al., 2004) model doesn't allow for the possibility that the amount of data needed to complete the decryption process is governed by data produced during the decryption process.
- In the analysis, ciphertexts and plaintexts are handled as atomic strings.
- However, the attack exploits the fact that an attacker can send data in small chunks to the recipient.
- Many implementations use a buffer to store data until it is needed, but (Bellare et al., 2004) doesn't model this.

Fixing things up I

- (Paterson and Watson, 2010) provided a new formal analysis of the SSH BPP using counter-mode encryption, with the intention of addressing the shortcomings of the previous analysis.
- They defined a new version of SSH-CTR that accurately captures how the SSH BPP with counter-mode encryption is defined in the RFCs and coded in practice in OpenSSH and other implementations.
- They also extended the previous security model to account for the manner in which the SSH BPP buffers as-yet-unprocessed ciphertext bytes, and to let the attacker deliver ciphertext to a decryption oracle in a byte-by-byte fashion, and proved their new definition of SSH-CTR is secure in the new model.
- This work was extended by (Boldyreva et al., 2012), in particular by extending the security model to capture ciphertext fragmentation.

Fixing things up II

- This is not the end of the story.
- (Albrecht et al., 2016) examines and formally proves secure some of the plethora of new SSH-specific modes of operation that arose after the attack we have just outlined was published.
- They also contributed to the further extension and refinement of security models necessary to capture the fine details of how SSH works in practice — improving and correcting (Boldyreva et al., 2012).
- It turns out that proving the required properties of an apparently simple protocol has necessitated much new work on security models to capture the real-world need for ciphertext fragmentation — superficially a minor issue.

Other examples

- (Degabriele et al., 2011) provide a list of related examples of failures of this type, which we briefly summarise.
- It was possible to mount passing oracle attacks against the SSL/TLS MAC-then-encrypt mode, despite the existence of a security proof: the security proof did not consider error messages leaking information.
- Attacks against the IPsec MAC-then-encrypt mode were found that exploited features not captured in the security proof, notably data fields not covered by the MAC calculation.

5. Erroneous ‘proofs’: A bit of a shock

- The recently revealed attacks on OCB 2.0 came as something of a shock to the cryptographic community.
- OCB 2.0 (Rogaway, 2004) is the second of a series of three block cipher modes of operation designed to provide authenticated encryption.
- It was standardised in ISO/IEC 19772:2009.
- All three members of the OCB series have proofs of security.

Cryptanalysis

- In October 2018, much to everyone's surprise, Inoue and Minematsu presented an existential forgery attack against OCB 2.0 requiring only a single prior encryption query and minimal computation.
- The attack does not affect the other two versions of OCB.
- Poettering and Iwata independently improved the Inoue-Minematsu attack to a full plaintext recovery attack very shortly after the October 2018 announcement.
- Full details of the attacks are provided in a Crypto 2019 paper (Inoue et al. 2019).
- A revised version of ISO/IEC 19772, about to be published, will omit OCB 2.0.

What went wrong?

- We summarise the explanation from (Inoue et al. 2019).
- All members of the OCB family can be seen as modes of operation of a tweakable block cipher.
- Each message block is enciphered independently of the others using a tweak that reflects the position of the block in the message.
- Special tweaking rules are deployed for the last (possibly padded) message block and the checksum used for tag generation.
- In OCB 2.0, the tweakable block cipher itself is derived from an underlying 'regular' block cipher (e.g. AES) using the XEX* transform, a hybrid of XE ('XOR-encipher', i.e. $C = E_K(\Delta \oplus M)$) and XEX ('XOR-encipher-XOR', i.e. $C = \Delta \oplus E_K(\Delta \oplus M)$), where it is decided on a per-evaluation basis which of the two is used.

What went wrong (continued)?

- The flaw in OCB 2.0 is not because it is constructed from a tweakable block cipher; there is also no issue with the security of the XEX* primitive.
- The problem is that if XEX* is ever evaluated twice on the same input, but in different modes (i.e. one in XE and one in XEX), all its proven security properties no longer hold.
- This was clearly known by Rogaway (Rogaway, 2014); however, he overlooked that OCB 2.0 does not always satisfy this requirement.
- An attacker can arrange that an XEX evaluation occurring when encrypting a regular message block and an XE evaluation occurring when decrypting a (padded) last block of an unauthentic ciphertext are on the same inputs.
- This fact was overlooked by the cryptographic community for 15 years; it not only invalidates the formal security argument for OCB 2.0 but leads to attacks that completely break its security.

Analysis

- In this case it seems that a subtle mistake in the logic of the security proof for OCB 2.0 was missed by everyone for 15 years.
- Indeed, I understand that the problem with the proof was discovered first, and then the attacks were found.
- The fact that an error in a proof exists is not surprising — humans, even world-renowned experts, are fallible.
- Perhaps the problem here was with the reviewing process — should the problem have been spotted?

6. Analysis and recommendations

- I guess the first thing to say is that none of these examples in any way invalidate the ‘provable security’ paradigm.
- Indeed, they only reinforce the need to insist upon security proofs wherever possible.
- However, we need ways to ensure that the proofs are correct, and that the security models cover what is needed.
- Unfortunately, some of the pressures imposed by our current publication model do not always support these requirements.

The need to check!

- First and foremost, the proofs and the appropriateness of the models need to be carefully checked prior to publication.
- Indeed, that is one of the main roles of the refereeing process.
- However, it is not clear that this process is doing its job effectively.
- Why?

Publication model

- The prevalent publication model for papers in cryptography and cryptographic protocols is via conferences.
- Inevitably this means the reviewing period is compressed — programme committee members are often asked to review 5–10 papers in just a few weeks.
- This often makes careful and detailed reviews of models and proofs difficult, if not impossible.
- Often the main role of reviewers is simply to filter out the obvious nonsense, the badly written papers, the duplicated work, or the uninteresting results.
- This may be sufficient for some topics, but clearly it isn't when the details really matter.

Too many publications

- We also face an ever growing volume of publications, both in cryptography and more generally across all areas of academia.
- There would appear to be two main pressures driving this.
  1. commercial (and learned society) publishers are keen to both continuously expand the number of papers published in their journals and also increase the number of published conference proceedings.
  2. academics need to publish to progress in their career.
- Even worse, journals will often refuse to accept papers pointing out that work they published is wrong — this is highly irresponsible.

Subject area cultures

- There are also cultural issues at play.
- In Mathematics, where journals are the main venue for publication, it is generally accepted that reviewers should have two or three months to review a paper, and that a review should involve checking (as much as possible) details of proofs.
- This idea seems to be alien for many of us who work in Computer Science.
- The main excuse seems to be that the subject is progressing so rapidly that a delay of a few weeks is not acceptable.
- This is, in my view, complete bollocks — most findings are disseminated using preprint sites such as arXiv and the cryptology eprint archive, so there really is no need to shortcut detailed reviewing.

A possible way forward I

- When constructing this talk it seemed obvious that it should end with suggestions for addressing some of the issues identified.
- However, it is not so easy!
- It would be a major shift, but we could follow standard practice for many other subject areas and use conferences as a way of disseminating new findings, and not require conference proceedings — detailed publications of results can be left to after the conference, and can use journals with an inherently less intensive review schedule.
- In any event, Covid-19 has taught many of us that there is no need to spend several weeks a year flying round the world, at huge cost to the planet — we can instead hold less formal online meetings to discuss new findings.

A possible way forward II

- What do we do about the sheer volume of publications?
- Is it reasonable to simply ignore anything that is not published in our favourite conference or journal?
- I would argue not — of course there will always be ‘fringe’ and vanity publications, but we as a community should try to drive up the quality of papers published in mainstream journals.
- This would be easy to achieve if we all took our fair share of the load — as I mentioned earlier in this talk, rejecting bad papers is easy!

References I

- (Albrecht et al., 2009): Albrecht, M. R., Paterson, K. G., and Watson, G. J.: Plaintext Recovery Attacks against SSH. Proc. IEEE Symposium on Security and Privacy 2009: pp.16–26. IEEE (2009).
- (Albrecht et al., 2016): Albrecht, M. R., Degabriele, J. P., Hansen, T. B., and Paterson, K. G.: A Surfeit of SSH Cipher Suites. Proc CCS 2016: pp. 1480–1491. ACM (2016).
- (Bellare et al., 2004): Bellare, M., Kohno, T., and Namprempre, C.: Breaking and Provably Repairing the SSH Authenticated Encryption Scheme: A Case Study of the Encode-then-Encrypt-and-MAC Paradigm. *ACM Trans. Inf. Sys. Sec.* **7**:206—241 (2004).
- (Boldyreva et al., 2012): Boldyreva, A., Degabriele, J. P., Paterson, K. G. and Stam, M.: Security of Symmetric Encryption in the Presence of Ciphertext Fragmentation. Proc. EUROCRYPT 2012, LNCS 7237: pp.682–699. Springer (2012).
- (Boyd et al., 2020): Boyd, C., Mathuria, A., and Stebila, D.: Protocols for Authentication and Key Establishment, 2nd edition. Springer (2020).

References II

- (Chilveri and Nagmode, 2020): Chilveri, P. G. and Nagmode, M. S.: A novel node authentication protocol connected with ECC for heterogeneous network. *Wirel. Networks* **26(7)**: 4999–5012 (2020).
- (Degabriele et al., 2011): Jean Paul Degabriele, Kenneth G. Paterson, Gaven J. Watson: Provable Security in the Real World. IEEE Secur. Priv. 9(3): 33–41 (2011)
- (Inoue et al., 2019): Inoue, A., Iwata, T., Minematsu, K., and Poettering, B.: Cryptanalysis of OCB2: Attacks on Authenticity and Confidentiality. Proc. Crypto 2019, LNCS 11692: pp. 3–31. Springer (2019).
- (Mitchell, 2020); Mitchell, C. J.: Provably insecure group authentication: Not all security proofs are what they claim to be. Proc. ICSP 2020, LNCS, to appear. (Also CoRR abs/2005.05376 (2020))

References III

- (Moazami and Safkhani, 2020): Moazami, F. and, Safkhani, M.: SEOTP: a new secure and efficient ownership transfer protocol based on quadric residue and homomorphic encryption. *Wirel. Networks* **26(7)**: 5285–5306 (2020).
- (Paterson and Watson, 2010): Paterson, K. G., and Watson, G.: Plaintext-Dependent Decryption: A Formal Security Treatment of SSH-CTR. Proc. EUROCRYPT 2010, LNCS 6110: 345–361. Springer (2010).
- (Rogaway, 2004): Rogaway, P.: Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC, Proc AsiaCrypt 2004, LNCS 3329, pp. 16–31. Springer (2004).
- (Xia et al., 2020): Xia, Z., Harn, L., Yang, B., Zhang, M., Mu, Y., Susilo, W., and Meng, W.: Provably secure group authentication in the asynchronous communication model. In: Proc. ICICS 2019, LNCS 11999, pp. 324–340. Springer (2020).