The (in)security of some recently proposed lightweight key distribution schemes

Chris J. Mitchell
Information Security Group, Royal Holloway, University of London
www.chrismitchell.net

20th January 2021

Abstract

Two recently published papers propose some very simple key distribution schemes designed to enable two or more parties to establish a shared secret key with the aid of a third party. Unfortunately, as we show, most of the schemes are inherently insecure and all are incompletely specified — moreover, claims that the schemes are inherently lightweight are shown to be highly misleading.

1 Introduction

Two papers recently published by Harn et al. [6, 7] propose some very simple key distribution schemes designed to enable two or more parties to establish a shared secret key with the aid of a third party. Unfortunately, as we show, most of the schemes are inherently insecure and all are incompletely specified. Moreover, claims regarding the efficiency of the schemes, namely that they are inherently lightweight, are shown to be highly misleading.

The remainder of this paper is arranged as follows. Section 2 provides an introduction to the protocols in question, and sets them in the context of the extensive prior art. This is followed by analyses of the various proposed protocols: Sections 3, 4, 5 and 6 provide analyses of the four protocols in the first paper [6], and Section 7 briefly examines the almost identical protocols in the second paper [7]. Finally, Section 8 concludes the paper.

2 Background

The two papers, whilst having very closely-related content and the same set of authors, do not refer to each other, and it is hard to know which

was submitted first. For simplicity we refer throughout to *Paper A* [6], and *Paper B* [7]. Since the two schemes in Paper B [7] both appear to be slightly elaborated versions of two of the schemes in Paper A [6], we consider Paper A first.

2.1 Paper A

Paper A [6] presents a total of four closely related schemes, which we refer to below as A1–A4. They all involve using a combination of a Key Derivation Function (KDF) (see, for example, ISO/IEC 11770-6:2016 [10]) and pre-established shared secret keys to establish new one-time session keys for pairs or groups of participants.

All the schemes work in essentially the same way. Long-term secret keys are first used to create one-time keys, using a KDF. The newly generated one-time keys are then combined using bit-wise exclusive-or to create the desired session keys. We provide details of the individual schemes below.

2.2 Paper B

Paper B [7] presents two schemes which also reply on KDFs and pre-established shared secrets — we refer to these as B1 and B2. As we observe below, both B1 and B2 are essentially the same as scheme A3.

2.3 Threat models

Neither of the two papers explicitly give the threat model in the context of which the protocols are designed to operate. We therefore assume (as is common practice — see, for example, Boyd et al. [2]) that messages are sent between participants via an unreliable channel, i.e. one where an adversary can eavesdrop on messages and also delete, modify or insert spurious messages. Of course, any such channel can be upgraded to a secure channel using well-established technology such as TLS [15], but in such a case the protocols described in the two papers would not be needed — keys could simply be transferred in cleartext.

2.4 The prior art

There is a large and very well-established literature on protocols designed to enable two parties to establish a shared secret session key with the aid of a trusted third party with whom they both share a long-term secret key. A classic example is that of Kerberos, whose design dates back to the mid-1980s [12, 13, 16]. An extensive discussion of such protocols can be found

in the landmark 1990s Handbook of Applied Cryptography [14], and the first edition of a standard for secret-key-based key establishment protocols, ISO/IEC 11770-2, was published in 1996 [8] — the current version of the standard was published in 2018 [11]. For an up-to-date summary of the state of the art, and a review of the history of the subject area, the reader is referred to Chapter 3 of the excellent Boyd et al. [2].

Unfortunately, much of this prior art is either not known to the authors or has been ignored. Certainly, no attempt has been made in either of Papers A or B to provide a comparison of the efficiency and/or the security properties of the proposed schemes with the prior art.

3 Scheme A1

3.1 Operation

The first scheme in Paper A ([6], §3.1) is designed to enable two parties (A and B) to establish a shared secret session key, with the aid of a trusted third party C. A and B both share a long-term secret key with C (which we label L_{AC} and L_{BC}, respectively), although A and B do not, a priori, share a key — explaining the role of C. Third party C is trusted to the extent that it chooses the value of the secret session key to be shared by A and B.

All parties are also assumed to have access to a KDF f, the nature of which is not precisely specified. For the purposes of this discussion, and in line with common practice, we suppose that f takes two inputs, namely (a) a secret key, and (b) some one-time data (which may be public), and outputs a secret key (this is what is referred to in ISO/IEC 11770-6 as a *key expansion function*). The nature of this function is also not specified, but typically it is instantiated as a CBC-MAC, i.e. a Message Authentication Code based on a block cipher in CBC mode (see, for example, ISO/IEC 9797-1 [9]). It could also be instantiated using a cryptographic hash function, although the computational complexity is likely to be very similar to use of a block cipher. Most importantly, regardless of how it is implemented, it must be computationally infeasible to recover the input secret key even if a number of outputs are known. Note that all the schemes in both papers use such a function f, and we implicitly assume throughout that this has been agreed in advance.

The scheme has two phases.

1. In Phase 1, which is only briefly sketched in §3.1.2 of Paper A:
    - A, B and C (by some unspecified means) agree on a public nonce value N;

- A and C use a public nonce N and the KDF f to compute a one-time shared secret key as $K_{AC} = f(L_{AC}, N)$;
- B and C make an analogous calculation to compute a one-time shared secret key as $K_{BC} = f(L_{BC}, N)$.

It is implicit that N should be generated in such a way that it is only ever used once.

2. In Phase 2, C chooses a one-time session key K_{AB} to be shared by A and B, which must have the same bit-length as K_{AC} and K_{BC}, and distributes it to them both in the following way:

   - C computes $C_A = K_{AB} \oplus K_{AC}$ (where here and throughout $\oplus$ denotes bit-wise exclusive-or) and sends this value to A;
   - A computes $C_A \oplus K_{AC} = K_{AB}$ to recover the session key;
   - in parallel, C computes and sends $C_B = K_{AB} \oplus K_{BC}$ to B, who can compute $C_B \oplus K_{BC} = K_{AB}$.

   That is, at the end of Phase 2, A and B will share the session key K_{AB}.

3.2 Efficiency and comparisons

The authors of Paper A make the following claims (see [6] §3.1.2).

> The operation of this proposed scheme is lightweight since it only needs to evaluate the positions of matching bits between two keys which is equivalent to the computation of a logical exclusive or (XOR) operation. In summary, the scheme is very efficient in computation since logic XOR is the simplest operation and it is also efficient in communication since it is non-interactive.

It is true that Phase 2 of the scheme does involve bit-wise exclusive-or operations. However, Phase 1 requires both A and B to compute the function f, i.e. to perform at least one block cipher encryption or hash function computation, which is nowhere near so lightweight as an XOR — typically, a hash function computation is of the same order of complexity as a block cipher encryption. The claim made in the paper is thus extremely misleading.

It is helpful to consider how the scheme compares with various well-established protocols of this general type, i.e. in which a shared secret session key is established between two parties with the help of a trusted third party, with whom both parties share a long-term secret key. ISO/IEC 11770-2 [11] specifies four protocols of this type, namely Mechanisms 7–10, which are referred to as *Mechanisms using a Key Distribution Centre*. These protocols

have been extensively analysed over the 25 years since the standard first appeared, and (after some minor modifications) there is now robust evidence that they meet the claimed security properties (see, in particular, Cremers and Horvat [3, 4]).

The four protocols have slightly varying security properties, and two of them also include an optional step designed to enable A and or B to confirm that their counterpart has successfully received the shared key. Since such a property is not provided by scheme A1, we ignore the optional steps. Table B.1 in Annex B of ISO/IEC 11770-2 [11] helpfully provides a detailed comparison of the protocols, stating that all four protocols require A and B to perform one encryption operation, with the exception of Mechanism 10 which requires one of the two parties to perform two such operations. A superficial analysis therefore suggests that the complexity of these existing standardised protocols is actually directly comparable with Scheme A1.

Finally note that, whilst I am not aware that the precise protocol has previously been proposed (there are so many possible variants), something rather similar was proposed by Gong over 30 years ago [5]. It is interesting to note that an issue was found with this latter scheme by Boyd and Mathuria in 1997 [1].

3.3 Security analysis

Unfortunately it appears that Scheme A1 is subject to a simple attack. We sketch one possible attack scenario. The attack hinges on the fact that there is no specified means for the parties to agree on who is involved in the protocol at the beginning of Phase 1. We assume that the protocol set-up (including setting up the nonce) takes place via a public channel, with no protection for the exchanged messages; of course, such an exchange could be made secure but this would require further use of cryptography, making the protocol even less 'lightweight'.

In line with the assumed threat model stated in Section 2.3, we suppose that a malicious third party E can control the communications channel with respect to a victim user A. We suppose that E convinces A that a secret key is being established with party B (who is actually not involved). The third party C believes it is establishing a secret key between A and another (legitimate) party D. All active parties (i.e. A, C and D) agree on the nonce N.

C computes a one-time key for A as $K_{AC} = f(L_{AC}, N)$, chooses the session key K_{AD} and sends $K_{AC} \oplus K_{AD}$ to A (the malicious party E does not interfere with this message). A can now also compute K_{AC} and uses it to recover K_{AD}, which A believes is shared with B. Simultaneously C computes a one-time key for D as $K_{AD} = f(L_{AC}, N)$, and computes and

sends $K_{DC} \oplus K_{AD}$ to D; D now recovers K_{AD}, which D (correctly) believes is shared with A.

That is, at the end of the protocol A has a session key which it believes is shared with B but is actually shared with D. This is clearly a breach of the protocol objectives, especially if we observe that D could be the same as the malicious entity E. Indeed, by repeating this deception with party B, E could end up with session keys shared with A and B, which A and B believe are shared by each other. This would enable E to act as an eavesdropping 'man-in-the-middle', reading messages sent between A and B which A and B believe are securely encrypted.

There are a range of possible 'fixes' to the protocol. One possibility would be to include the identifiers for A and B in the inputs to the KDF f, i.e. so that one-time keys are computed as function of both a nonce and unique identifiers for the parties involved. However, as is well-known, proposing ad hoc fixes to protocols without providing rigorous evidence of security is a dangerous path, and so even a fixed-up version should only be considered for real-world use subject to the recommendations below.

3.4 Recommendations

If such a protocol is to be seriously considered for use then the following points need to be addressed.

- A modification to address possible attacks, such as that outlined above, needs to be fully specified.
- A threat model needs to be carefully defined, and a formal security model for the properties of the protocol needs to be given that matches the threat model.
- A proof of security in the chosen security model needs to be given.
- A detailed performance and security comparison with existing protocols, such as those specified in ISO/IEC 11770-2 [11] needs to be given, so that any performance advantages over the prior art can be verified.

4 Scheme A2

4.1 Operation

This scheme is described in §3.2 of Paper A [6]. It is even simpler than the first scheme. It simply describes how an entity A can be equipped by

a trusted party S with a new secret session key, assuming A and S share a long-term secret key, L_{AS} say.

There are four steps (although the first two are only briefly sketched in §3.2.2 of Paper A).

1. A and S (by some unspecified means) agree on a public nonce value N;
2. A and S use the public nonce N and the KDF f to compute a one-time shared secret key as $K_{AS} = f(L_{AS}, N)$;
3. S chooses a one-time session key K to be shared by A and S, which must have the same bit-length as K_{AS}, computes $C_S = K_{AS} \oplus K$, and sends this value to A;
4. A recovers $K = C_S \oplus K_{AS}$.

4.2 Efficiency and comparisons

Analogously to Scheme A1, the authors of Paper A make the following claims ([6], §3.2.2).

> In summary, the scheme is very efficient in computation since it needs only one logical XOR operation for both server and user and is efficient in communication since it is non-interactive.

Just as for Scheme A1, the protocol requires A and S to compute the function f, i.e. at least one block cipher encryption or hash function computation, which is nowhere near so lightweight as a single bit-wise XOR. The efficiency claim is thus again extremely misleading.

Interestingly, steps 1 and 2 of the mechanism are essentially identical to the steps in Key Establishment Mechanism 1 of ISO/IEC 11770-2 [11].

4.3 Security analysis and recommendations

As the standard makes clear, ISO/IEC 11770-2 Key establishment mechanism 1 (which corresponds to steps 1 and 2 of Scheme A2) does not provide authentication of the one-time key — i.e. of the key K_{AS} using the above notation. This is because there is no means provided for A and S to securely agree on the value of the nonce N — moreover, C_S is also not authenticated. As a result, it follows that the session key K established by Scheme A2 is not authenticated, i.e. it could even be the case that this key is known to a party other than A or S.

Therefore, unless additional mechanisms are put in place to guarantee the timeliness and origin of the nonce C and the value C_S, the mechanism should not be used. A set of six mechanisms of this general type with well-understood security properties are given in Clause 6 of ISO/IEC 11770-2 [11], and depending on the context and precise security requirements, one of these should be used in preference to Scheme A2; that is, of course, unless the communications channel is already made secure by other means.

5 Scheme A3

This slightly more complex scheme can be found in §3.3 of Paper A. It is designed to enable a group of three or more users to establish a shared session key using pre-established long-term shared secret keys. Two versions are given: first a scheme designed specifically for three parties, and subsequently a scheme for an arbitrary-sized group. We consider them in turn.

5.1 The three-party scheme

The scheme is a simple derivative of Scheme A1. The only difference is that pre-established long-term secret keys are assumed to be shared by every pair of the three parties (labeled A, B and C). These are then combined with an agreed nonce using a KDF to generate three one-time shared secret keys K_{AB}, K_{AC} and K_{BC} (shared by A and B, A and C, and B and C, respectively).

Any one of the parties, which we assume to be A (without loss of generality), computes $C_A = K_{AB} \oplus K_{AC}$ and sends it to both B and C. Both B and C can now recover the pair of keys (K_{AB}, K_{AC}) — which are also known to A — and a combination of these two keys, e.g. using a KDF, forms the group key.

Since it is essentially the same as Scheme A1 it suffers from the same serious security issues. Moreover, while Paper A makes the usual claim about the protocol being lightweight, its (hidden) dependence on use of the KDF for every instance of the protocol means that this claim is highly misleading.

5.2 The multi-party version

This is an elaboration of the three-party version, which is only very briefly sketched in §3.3.2 of Paper A. One of the parties is appointed as the initiator, and somehow all of the n parties agree on the identity of the initiator and a nonce. The initiator is also assumed to pre-share a long-term secret key with all the other $n - 1$ parties. To start the protocol, one-time keys are

generated as a function of the nonce and these long-term shared keys; as a result the initiator will share a one-time secret key with every other party.

The n parties are assumed to be arranged in a binary tree, rooted at the initiator (at the 'top' of the tree). The initiator then engages in a series of instances of the three-party protocol, systematically working 'down' the tree, at each stage establishing a group key shared by a larger set of participants. Finally, when the 'bottom' of the tree is reached, all participants share a single group key.

Since the three-party protocol suffers from the same major security issues as Scheme A1, as discussed above, similar issues arise with the n-party version of the scheme. Yet again the authors make the highly misleading claim that 'the scheme is very efficient in computation since it requires only one logical XOR operation for each user'.

6 Scheme A4

6.1 Goals of scheme

The final scheme is described in §3.4 of Paper A [6]. The scheme has the same primary objective as the three-party version of Scheme A3, namely to establish a shared group key amongst three parties (A, B and C) who have pre-established long-term secret shared keys. However, one difference is that this scheme is claimed to provide an *authenticated* group key, although what this means, and the precise threat model, are left unspecified. However, it is stated that the scheme 'should be able to prevent any outside attacker to impersonate to be any legitimate user and discover the group key'.

This implicitly means that the other three schemes are *not* authenticated — indeed, this is clear from the analyses of Schemes A1–A3 above. It could therefore be argued that to claim the previous schemes are insecure is unfair, as they were not designed to be secure against active attackers. However, the following points need to be taken into account when evaluating such an argument.

- At no point in Paper A (or Paper B) is there any effort to make clear that the schemes A1–A3 (and B1 and B2) are insecure and should only be used when the communications channels are secure, e.g. using an underlying protocol such as TLS. Indeed, the requirement for such an underlying security protocol would completely invalidate any claims about lightweight properties. Moreover, the authors state the following in §3.1 of Paper A (this is the part of the paper describing Scheme A1 — the 'basic scheme' in the language used below).

> This basic scheme can be applied to many practical network applications. For example, in a wireless sensor network, each sensor has been pre-loaded with a subset of keys before deploying sensors to a geographical area in a random key distribution solution. After deploying these sensors, two neighboring sensors intend to negotiate a common key between them to establish a secure communication. In order to increase the probability of sharing an overlapping key in two different subsets of sensors, researchers have proposed various solutions. Our proposed key distribution scheme provides an alternative solution for this application. In our scheme, two sensors with no pre-shared key can establish a one-time common key through a third sensor in which both sensors have pre-shared keys with the third sensor separately.

This text clearly recommends use Scheme A1 directly, i.e. without any additional security measures, over a wireless sensor network, where such networks are likely to be deployed in an environment where interception and manipulation of data transmissions will be feasible.

- Possible attacks on Scheme A4 are described below, which enables the value of an agreed group key to be changed, i.e. so that one participant ends up with a key different to that of the two other participants. That is, despite being described as 'authenticated', the scheme does not offer significantly more protection against malicious adversaries than any of the other schemes.

- It is far from clear what the authors mean by 'authenticated' key establishment; it would seem (see below) that the only additional security property for Scheme A4 is that it can detect if A and B send inconsistent values to C.

6.2 Operation

As for the three-party variant of Scheme A3, it is assumed that pre-established long-term secret keys are shared by every pair of the three parties. These are then combined with an agreed nonce using a KDF to generate three one-time shared secret keys K_{AB}, K_{AC} and K_{BC} (shared by A and B, A and C, and B and C, respectively). We suppose that the versions of the keys held by A are K^1_{AB}, K^1_{AC}, the versions of the keys held by B are K^2_{AB}, K^2_{BC}, and the version of the keys held by C are K^3_{AC}, K^3_{BC}. Of course if everything is working correctly then $K^1_{AB} = K^2_{AB}$, $K^1_{AC} = K^3_{AC}$, $K^2_{BC} = K^3_{BC}$, but the protocol is designed to detect such inconsistencies.

Two of the three parties now jointly act as protocol initiators (which, without loss of generality, we assume are A and B). Both A and B now conduct the three-party version of Scheme A3, which enables all parties to agree on the group key $K = K^1_{AB} = K^2_{AB}$.

- A computes $C_A = K^1_{AB} \oplus K^1_{AC}$ (using its copies of these two keys) and sends it to C;
- B computes $C_B = K^2_{AB} \oplus K^2_{AC}$ (again using its copies of these two keys) and sends it to C;
- C computes $K^1 = C_A \oplus K^3_{AC}$ and $K^2 = C_B \oplus K^3_{BC}$. If $K^1 = K^2$, which should be true if all parties hold the same pre-shared keying material, the same nonce, and all messages are exchanged correctly, then C accepts $K^1 = K^2$ as the group key (as do A and B).

6.3 Analysis

As for the three previous schemes, §3.4.2 of Paper A makes a claim about the efficiency of the scheme.

> Each initiator needs to execute one logic XOR operation to compute the positions of matching bits between two pairwise shared keys and the third user needs to execute two XOR operations to extract and compare a pairwise shared key which the user has no prior knowledge of it. The overall performance of this scheme is lightweight.

As in the previous cases, there is no mention of the cost of applying a KDF, which means that the efficiency claims are highly misleading.

Even more seriously, the scheme is not significantly more secure than Schemes A1 and A3, as somewhat analogous attacks apply. We sketch two possible attack scenarios. In both cases, in line with the assumed threat model stated in Section 2.3, we suppose that a malicious third party E can control the communications channel with respect to a victim user C.

In the first attack, E first chooses a block of bits M of the same length as the key K_{AB}. E now modifies the messages sent from A to C and B to C as follows:

- C_A is replaced with $C_A \oplus M$; and
- C_B is replaced with $C_B \oplus M$.

It is not hard to see that C will compute $K^1 = K^2 = K_{AB} \oplus M$. That is, the value accepted by C will be different to the value held by A and B, i.e.

the method does not provide key authentication as claimed. Of course, this attack could probably be 'fixed' by providing explicit authentication for the messages containing C_A and C_B, e.g. by adding a MAC to the two messages. However, this would make the protocols much less efficient, voiding any justification for departing from the well-established state of the art.

The second attack requires E to be able to manipulate the nonce value agreed by the participants. This is a plausible assumption — there is certainly no discussion in either of the two papers on a secure method for agreeing this nonce, and it is therefore reasonable to assume it is agreed by insecure messaging (adding an exchange to agree the nonce in a secure way would essentially require a security protocol to be run in advance of the one proposed). By comparison, this issue *is* addressed in previously proposed nonce-based protocols, by making the nonce exchange an explicit part of the protocol. The attack proceeds in three stages.

1. Suppose E convinces A to run the protocol with itself and party C, where A and E are the initiators and N is the nonce. Note that C is not actually involved — E can send to A messages that are apparently from C agreeing to use of the protocol, as required. Suppose also that E intercepts the message $C_A = K_{AE} \oplus K_{AC}$ sent from A to C (and prevents it from reaching C). Since E knows K_{AE}, E can now recover the value of K_{AC} for this value of the nonce N.

2. In a very similar way, E convinces B to run the protocol with itself and party C, where B and E are the initiators and the value of N is the nonce. Again C is not actually involved, and as previously we suppose that E intercepts the message $C_B = K_{BE} \oplus K_{BC}$ sent from B to C (and also as in the previous step prevents it from reaching C). Since E knows K_{BE}, E can recover the value of K_{BC} for this value of the nonce N.

3. Finally, E persuades C to run the protocol with A and B, where A and B are the initiators and the same value N is the nonce. The two parties A and B are not actually involved, and E send messages to C that are apparently from A and B, as necessary. E now chooses an arbitrary key K^* and sends both $C^*_A = K_{AC} \oplus K^*$ to C (impersonating A), and $C^*_B = K_{BC} \oplus K^*$ to C (impersonating B), where K_{AC} and K_{BC} are the values learnt from the first two steps of the attack. It is straightforward to see that C will accept K^* as a valid group key shared with A and B, whereas in fact it is known to E (and not to A and B).

It is important to note that no party is asked to use the same nonce value twice, so even if all parties keep a record of all the nonce values they have used (and refuse to reuse a value) the attack can still operate.

7 Schemes B1 and B2

The schemes are essentially the same as Scheme A3 in Paper A, with some slight variations in the wording. More precisely the schemes in Paper B have the following relationship to the schemes in Paper A.

- Scheme B1 (given in §3 of Paper B) is precisely the same as the three-party version of Scheme A1.
- Scheme B2 (described in §4 of Paper B) is a group key distribution scheme and comes in two variants. The first variant (see §4.1 of Paper B) involves multiple uses of Scheme A1, in a somewhat inefficient way. The second variant (§4.2) is the same as the n-party variant of Scheme A2, i.e. it involves use of a binary tree.

Just as in Paper A, claims are made about the efficiency of the schemes, completely ignoring the fact that use of the scheme involves computing a KDF.

8 Concluding remarks

We have analysed a range of key establishment schemes proposed in two recent papers. There are a number of major issues with these schemes, which we now summarise.

- The schemes have a range of serious security weaknesses, which potentially allow active opponents of the protocols to learn keys that they should not — this arises partly because at no point are the security assumptions on which the protocols rest made clear.
- All the schemes are claimed to be highly efficient ('lightweight') since they rely on computation of bit-wise exclusive-or operations to obtain the session key. However, in no case is a reference made to the fact that all the schemes require all parties to perform at least one key derivation computation, the cost of which makes the schemes no more lightweight than many rival schemes based on symmetric cryptography.
- Paper B is essentially a subset of paper A. Paper A does not refer to Paper B and neither does Paper B refer to Paper A. It appears they were submitted in parallel, although it is difficult to be sure since the published version of Paper A does not disclose when the first version was submitted.

In conclusion, neither of the papers add anything significant to the literature, especially as no attempt is made to compare the schemes to the prior art, provide a detailed description of (let alone prove) their security properties, or give a fair description of their computational complexity. Last but not least the issues raised by publishing the same material twice merit careful consideration.

References

[1] C. Boyd and A. Mathuria. Systematic design of key establishment protocols based on one-way functions. *IEE Proceedings — Computers and Digital Techniques*, 144:93–99, March 1997.

[2] C. Boyd, A. Mathuria, and D. Stebila. *Protocols for Authentication and Key Establishment*. Information Security and Cryptography. Springer, 2nd edition, 2020.

[3] C. Cremers and M. Horvat. Improving the ISO/IEC 11770 standard for key management techniques. In L. Chen and C. J. Mitchell, editors, *Security Standardisation Research — First International Conference, SSR 2014, London, UK, December 16-17, 2014. Proceedings*, volume 8893 of *Lecture Notes in Computer Science*, pages 215–235. Springer, 2014.

[4] C. Cremers and M. Horvat. Improving the ISO/IEC 11770 standard for key management techniques. *Int. J. Inf. Sec.*, 15(6):659–673, 2016.

[5] L. Gong. Using one-way functions for authentication. *ACM Computer Communication Review*, 19(5):8–11, October 1989.

[6] L. Harn, C. Hsu, and Z. Xia. Lightweight and flexible key distribution schemes for secure group communications. *Wireless Networks*, page page numbers to be assigned, 2020.

[7] L. Harn, C. Hsu, and Z. Xia. Lightweight group key distribution schemes based on pre-shared pairwise keys. *IET Commun.*, 14(13):2162–2165, 2020.

[8] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 11770-2:1996, Information technology — Security techniques — Key management — Part 2: Mechanisms using symmetric techniques*, 1996.

[9] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 9797–1, Information technology — Security techniques —*

Message Authentication Codes (MACs) — Part 1: Mechanisms using a block cipher, 2nd edition, 2011.

[10] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 11770-6:2016, Information technology — Security techniques — Key management — Part 6: Key derivation*, 1st edition, 2016.

[11] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 11770-2:2018, IT Security techniques — Key management — Part 2: Mechanisms using symmetric techniques*, 3rd edition, 2018.

[12] J. Kohl and C. Neuman. *RFC 1510, The Kerberos Network Authentication Service (V5)*. Internet Engineering Task Force, September 1993. `https://tools.ietf.org/html/rfc1510`.

[13] J. T. Kohl. The use of encryption in Kerberos for network authentication. In G. Brassard, editor, *Advances in Cryptology — CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20–24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 35–43. Springer-Verlag, Berlin, 1990.

[14] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, 1997.

[15] E. Rescorla. *RFC 8446, The Transport Layer Security (TLS) Protocol Version 1.3*. Internet Engineering Task Force, August 2018. `https://tools.ietf.org/html/rfc8446`.

[16] J.G. Steiner, C. Neuman, and J.I. Schiller. Kerberos: an authentication service for open network systems. In *Proceedings: Usenix Association, Winter Conference, Dallas 1988*, pages 191–202. USENIX Association, Berkeley, California, February 1988.