Pairing Free Certified Common Asymmetric Group Key Agreement Protocol for Data Sharing Among Users with Different Access Rights

An Braeken¹

Accepted: 15 June 2021 / Published online: 1 July 2021

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

Abstract

Research into a pandemic like Covid-19 needs a tremendous input of patient histories and characteristics. Patients and doctors are only willing to share these sensitive data when they are ensured that the data are solely used by legitimate research laboratories. Asymmetric group key agreement (AGKA) protocols provide a good cryptographic primitive to address this requirement. The AGKA protocols proposed in literature provide users with a common public group key and a different decryption key by relying on compute intensive pairing operations. In this paper, we propose a new primitive, called the Common AGKA (CAGKA) protocol in which the users share the same private-public key pair, resulting in a more efficient solution. By combining Elliptic Curve Qu Vanstone certificates and a recently proposed Canetti–Krawczyk (CK) secure mutual authentication protocol, a one round self-certified pairing free CAGKA protocol is defined, which can be also globally certified after one additional round.

Keywords Asymmetric group key · Elliptic curve cryptography · Implicit certificates · Canetti–Krawczyk (CK) security

1 Introduction

Group key agreement protocols in literature can be distinguished into two categories, being the ones that lead to the construction of a common shared secret symmetric key and the ones resulting in a common shared public key. This last category is also called the asymmetric group key agreement (AGKA) protocol and has been proposed in [1]. In their definition and also the other follow-up constructions [2–6], the individual users of the group possess their own corresponding private key, which is different from the other participants of the group.

For symmetric group key agreement (SGKA) protocols, there is a group coordinator required, which chooses the key and shares it among the different members based on the

✉ An Braeken
an.braeken@vub.be

¹ Vrije Universiteit Brussel (VUB), Pleinlaan 2, 1040 Etterbeek, Belgium

individual input of the different members using secret sharing techniques (for instance by means of Lagrange interpolation). An SGKA protocol relying on solely symmetric key based operations has been proposed in [7] and a mutual authenticated SGKA protocol based on Elliptic Curve Cryptography (ECC) is presented in [8]. Recently, in [9], another ECC based SGKA scheme has been proposed, offering in addition resistance to ephemeral key compromise attacks. These attacks exploit the knowledge of random values used in the scheme, e.g. leaked by time analysis or other side channel techniques, to retrieve the session key. Besides the vulnerability for key escrow in SGKA protocols, this approach is also not advisable in case not all participants of the group possess “read” rights. For instance, in the use case of data sharing from individuals to a group of trusted laboratories, the individuals should not be able to derive the input of others.

Instead, the AGKA protocols provide a perfect solution to distinguish between users with only “write” rights and users with both “read” and/or “write” rights. The AGKA protocols are distributed one-round protocols, that do not require the different users to stay online to concurrently run the protocol. The latest generation of AGKA protocols is designed to enable the users without additional communication rounds to verify the correctness and authentication of the other participants, taking into account both passive and active attackers [3–6]. There are also some schemes that associate attribute based control to the construction of the AGKA protocols [10–12].

However, all of these schemes are relying on compute intensive pairing operations, used both in the key agreement phase and in the actual encryption/decryption process afterwards. Moreover none of them are able to offer ephemeral key compromise protections and only allow certification of the public key among the group members.

To conclude, in order to offer a viable solution for data sharing, the objective in this paper is to focus on the construction of an AGKA variant protocol to address the above mentioned shortcomings. This results in the following contributions.

- We argue the usefulness of having different decryption keys as in the AGKA protocols in literature and define the Common AGKA (CAGKA) protocols in which users share the same private-public key pair.
- We propose a single round pairing free CAGKA protocol applicable for users being certified by different certificate authorities (CAs) and offering protection in the Canetti–Krawczyk (CK) security model [13], where an adversary is also able to reveal session state specific information, session keys, or long-term private keys. This automatically implies the presence of perfect forward secrecy and protection against the ephemeral key compromise attack.
- We propose an additional round to enable implicit certification, allowing other parties to verify the certificate of the group key by integrating the Elliptic Curve Qu Vanstone (ECQV) certificates [14].

2 Related Work

The AGKA protocol has been introduced by Wu et al. in 2009 [1]. Their proposed protocol consisted of a single round, enabling each participant to publish independently their public key contribution, without the need of being connected during the protocol. First, it was defined for a fixed group, later for a dynamic group [2]. Different types of features have been added to the AGKA protocols in later years. In [5], Ranjani et al. propose an ID-based

approach to protect against active attacks and to avoid reliance on a trusted dealer. Since the group controller distributes the secret keys to the other participants, the scheme is vulnerable for key escrow. In addition, it also suffers from the ephemeral key compromise attack. In [4], Lv et al. proposed the combination with a certificateless public key cryptosystem to avoid key escrow and to enable the members of the group to self-certify the resulting public key in a single round. The scheme is not resistant against the ephemeral key compromise attack and is not able to offer perfect forward secrecy in which the previous session keys are protected even if the secret key of one of the participants leaks. In [3], Zhang et al. propose a cross domain self-certified authenticated group key protocol, in which the users can come from different authorization domains. Their protocol assumes the existence of a so called alliance public key between the different certificate authorities. The construction of this key has not been explained. In fact, its construction can be made by our CAGKA protocol. In addition, no perfect forward secrecy is obtained and there is also no protection against the ephemeral key compromise attack. Finally, there is the scheme of Chen et al. [6], in which also an identity-based cross-domain authenticated asymmetric group key is established. Here the validity of the public keys is guaranteed by the three-layer cross domain network architecture. This scheme satisfies forward secrecy, but offers still no protection against the ephemeral key compromise attack.

In all of the approaches mentioned above, only the members of the group are able to verify the validity of the group key. To conclude, as far as the authors are aware, all the proposed asymmetric group key agreement protocols are pairing based, which are very compute intensive cryptographic operations.

3 Preliminaries

3.1 Cryptographic Operations

The underlying cryptographic operations used in this paper are based on Elliptic Curve Cryptography (ECC). In ECC, an elliptic curve (EC) E_p over F_p is considered with generator G of order q . The security in ECC relies on the hardness of both the Elliptic Curve Discrete Logarithm Problem (ECDLP) and the Elliptic Curve Diffie Hellman Problem (ECDHP). In ECDLP, it is computationally hard for any polynomial-time bounded algorithm to determine a parameter $x \in F_q$ for which $Q = xG$, given Q and G . ECDHP states that given xG, yG , it is computationally hard to derive xyG .

We further utilise two basic symmetric key based primitives, a hash algorithm and symmetric key encryption/ decryption. The hash of a concatenated message M_1, M_2 , is denoted by $H(M_1, M_2)$ and results in an output of fixed length. The hash function should offer protection against collision, pre-image and second-image attacks. The symmetric key encryption encrypts a message M into a ciphertext C such that $C = E_K(M)$ using the shared secret key K . The decryption is denoted by $M = D_K(C)$.

The scheme is built upon the combination or integration of two primitives, described in literature.

3.1.1 Elliptic Curve Qu Vanstone (ECQV) Certificates

ECQV certificates [14] enable a participant to derive a secret key without the need for a secure channel and without the Certificate Authority (CA) knowing this secret. In addition,

based on identity and certificate, any other participant is able to derive the corresponding public key. The protocol consists of two rounds. First the participant U_i with identity ID_i chooses a random variable $r_i \in F_q$ and computes $R_i = r_i G$. The message ID_i, R_i is sent to the CA, who possesses the private-public key pair $(d_c, Q_c = d_c G)$, where Q_c is publicly known and certified to all users. Here, the CA also chooses a random variable $r_c \in F_q$ to derive $R_c = r_c G$ and defines the certificate $Cert_i = R_i + R_c$. Next, the CA determines auxiliary information $a = H(Cert_i, ID_i)r_c + d_c$ and sends $a, Cert_i$ to U_i . Based on this information, U_i is able to derive its private key $d_i = H(Cert_i, ID_i)r_i + a$. If $d_i G = H(Cert_i, ID_i)Cert_i + Q_c$, then U_i accepts the key pair (d_i, Q_i) .

3.1.2 Authentication Protocol Secure in CK Model

In [15], a highly efficient elliptic curve (EC) based mutual authentication protocol between a smart meter and service provider satisfying security in the CK security model has been presented. Let the smart meter possess the key pair (d_1, Q_1) together with random EC point $R_1 = r_1 G$ and the service provider possess (d_2, Q_2) with $R_2 = r_2 G$. Only the EC points Q_1, R_1, Q_2, R_2 are publicly available. Then, the protocol defines the session key SK as

$$SK = H(h_1 r_1 + d_1)(h_2 R_2 + Q_2) = H(h_1 R_1 + Q_1)(h_2 r_2 + d_2)$$

with $h_1 = H(ID_1, ID_2, R_1, R_2, Q_1, Q_2)$ and $h_2 = H(ID_2, ID_1, R_2, R_1, Q_2, Q_1)$.

3.2 Security Model

In our setting, we assume n users $U_1, \dots, U_n$ aiming to generate a common private-public key pair, where U_1 is responsible for the selection of the participating users and the initial collection of the key material, which should be made publicly available in order to initiate the actual protocol. Next, the users broadcast their contribution to the group key over an open public channel.

We further consider the existence of a passive and active attacker. As a consequence, messages sent in the open channel can not only be eavesdropped, but can also be captured, removed, and changed and even new messages can be inserted.

In addition, we also assume that the attacker is able to retrieve the long term secret private key or session specific random values of the users participating in the protocol, cf. the CK security model.

Moreover, also the existence of malicious insider users are considered, which try to impersonate legitimate users.

3.3 SGKA, AGKA and CAGKA Protocols

The main underlying difference between AGKA and CAGKA type of protocols is that in CAGKA all users share the same private key. The private key in AGKA is only used for decryption of the messages, which are encrypted with the public group key. In both AGKA and CAGKA, if a user is compromised, a remove user procedure should be installed, where all users need to restart the key agreement procedure. Consequently, no direct advantages, both from a security or efficiency point of view, are linked to AGKA compared to CAGKA.

One could say that SGKA can be easily transformed to CAGKA protocols by considering the resulting shared symmetric key as the private key. However, the main difference between SGKA and CAGKA is that SGKA still remains vulnerable for key escrow attacks.

4 Proposed Scheme

We consider four main phases in the proposed scheme. First, there is the registration phase, followed by the self-certified group key agreement phase and optionally further finished by the CA-certified group key agreement phase. Finally, there is also the group update phase in which a user leaves or joins the group.

4.1 Registration Phase

In the registration phase, the different participants U_i for $i \in \{1, \dots, n\}$ derive a key pair (d_i, Q_i) by its CA via ECQV. Denote the corresponding certificate by $Cert_i$ and public key of the CA by Q_c . For ease in notation, we here consider the existence of one CA, but with additional cost of publishing the CA public key, the participants can be connected also to different CAs like in [3].

4.2 Self-certified Group Key Agreement Phase

To start, each user U_i for $i \in \{1, \dots, n\}$ willing to participate in the group needs to publish its identity ID_i , certificate $Cert_i$ and EC point R_i constructed as $R_i = r_i G$ with r_i a randomly chosen value. The group controller, e.g. U_1 makes sure that this information is available.

Next, any user U_i with $i \in \{1, \dots, n\}$ chooses a random value a_i and computes for all users U_j with $j \in \{1, \dots, n\}, j \neq i$, $Q_j = H(Cert_j, ID_j) Cert_j + Q_c$. Then, the secret shared key $K_{ij} = H(h_1 r_i + d_i)(h_2 R_j + Q_j)$ with hashes $h_1 = H(ID_i, ID_j, R_i, R_j, Q_i, Q_j)$ and $h_2 = H(ID_j, ID_i, R_j, R_i, Q_j, Q_i)$ between U_i, U_j is derived by the CK mutual authentication scheme of [15]. This results in the parameter $x_{ij} = E_{K_{ij}}(a_i)$, which is sent together with $A_i = a_i G$ to all other users U_j .

Consequently, each user U_j with $j \in \{1, \dots, n\}, j \neq i$ is now able to derive from the available information $R_i, ID_i, Cert_i$ of U_i the shared key by computing $K_{ij} = H(h_1 R_i + Q_i)(h_2 R_j + Q_j)$ with $h_1 = H(ID_i, ID_j, R_i, R_j, Q_i, Q_j)$ and $h_2 = H(ID_j, ID_i, R_j, R_i, Q_j, Q_i)$ between U_i, U_j and retrieve a_i by $a_i = D_{K_{ij}}(x_{ij})$. If $a_i G$ corresponds with the published EC point A_i , user U_j is guaranteed of the integrity and authentication of the value. After repeating this process for the $n - 1$ users, a legitimate participant is then able to derive the asymmetric private group key $d_g = a_1 + \dots + a_n$ and the corresponding public group key by $Q_g = A_1 + \dots + A_n$ where $Q_g = d_g G$.

4.3 CA-Certified Group Key Agreement Phase

In the previous process, only the members of the group are ensured about the legitimacy of the other participants. An outsider receiving the group public key Q_g cannot verify the relation with the involved entities. Therefore, the ECQV mechanism is now applied on top of the previous phase.

For this, similar as in ECQV, the CA chooses a random value r_c and computes $R_c = r_c G$. The resulting certificate becomes $Cert_g = Q_g + R_c = A_1 + \dots + A_n + R_c$. Define $ID_g = H(ID_1, \dots, ID_n)$. The auxiliary value equals to $a = H(Cert_g, ID_g)r_c + d_c$ and sends $a, Cert_g$ to $\{U_1, \dots, U_n\}$. Based on this information, each U_i is able to derive the private key $d_{gc} = H(Cert_g, ID_g)d_g + a$. If $Q_{gc} = d_{gc}G = H(Cert_g, ID_g)Cert_g + Q_c$, the U_i accepts the key pair (d_{gc}, Q_{gc}) .

4.4 Group Update Phase

In case a user leaves or enters the group, the other users U_i can keep the random value R_i , resulting in the same session keys K_{ij} with the other still remaining users U_j . Only the value a_i should be updated in order to guarantee perfect forward and backward security. Note that after a fixed time or a certain amount of key update phases, each user needs to refresh the random key R_i . It is of course a trade-off in storing these common shared keys (efficiency) and recomputing the keys/redefining random values (security) and is mainly determined by the frequency of updates. The new user follows the same procedure as mentioned in the self-certified group key agreement phase. For the resulting group key, a CA-certified group key agreement phase can also be defined.

5 Security Analysis

An asymmetric group key agreement protocol is secure if it is able to guarantee that only the intended users of the group are able to compute the private group key. For this type of protocol, several generally accepted desirable security properties are defined [16].

- Known-key security ensures that if the protocol is successfully finished, each legitimate participant is able to compute the unique private key. Even if group private keys of some previous sessions are leaked, the security of the private keys in the other sessions is not compromised. This feature is clearly valid in our proposed scheme due to the usage of different random values a_i, r_i in each protocol run and the protection against the ephemeral key compromise attack, inherently present in the CK security model applicable in the construction of the keys K_{ij} .
- Unknown key-share security guarantees that all users are the real users as claimed during the group construction. The established session keys K_{ij} among the pairs of users U_i, U_j can only be derived by the true users as they claimed to be. This follows immediately from the construction of the symmetric keys used to encrypt the values a_i , which is based on both the mutual CK secure authentication protocol of [15] and the ECQV protocol.
- Key compromise impersonation security ensures that if an attacker is able to compromise the long term private key of a user and thus to impersonate that user, the impact should be limited to that user. This feature is inherently present in the CK security model and thus valid thanks to the usage of the mutual CK secure authentication protocol of [15].
- Key control security states that none of the users is capable to force the session key to be a preselected value. The other users are not able to change the chosen parameters a_i of the other users U_i without being noticed. Although, a legitimate user U_k (with $k \neq \{j, i\}$) is able to derive the input a_i of the published outputs x_{ij} , it is not able to

retrieve K_{ij} since symmetric key cryptographic protocols are designed to resist such known-plaintext attacks.

The formal security of the proposed protocols described in Sect. 4 can be easily deduced from the security of the two underlying primitives.

Theorem 1 *The self-certified group key agreement phase offers session key security under the CK adversary model [13] and in the random oracle model.*

The self-certified group key agreement phase between n participants is built upon $\binom{n}{2}$ applications of the mutual CK secure authentication protocol of [15]. The security of this protocol has been proven in the random oracle model in [15] and is strongly related to the computational hardness of the ECDH problem and the collision resistance of the hash function. The same conclusions of [15] on the advantage of the adversary in attacking the protocol can thus be made here.

Theorem 2 *The CA-certified group key agreement phase is secure in the random oracle model.*

This protocol is a direct application of the ECQV protocol, whose security has been formally proven in [17] and relies on the fact that the ECDLP is intractable.

Table 1 provides an overview of the differences in security features between several other AGKA and SAGKA protocols in literature. From this table, we can conclude that our proposed protocols are the only ones offering CK security, i.e. satisfying both resistance to the ephemeral key compromise attack and perfect forward secrecy, among the AGKA protocols. In addition, our CA certified group key agreement protocol is the only one in literature offering a global certificate to the resulting group key.

6 Efficiency Analysis

6.1 Computational Costs

For the efficiency analysis, we consider 160-bit security level and use the performance results of the different cryptographic operations as mentioned in [3, 18]. There, the results are obtained after implementation of the cryptographic primitives on an Intel R Core RM 2 Duo E8400 CPU3 (3.00GHz) ubuntu 10.04.

Table 1 Comparison of security features, where V1 denotes the self certified version and V2 the global CA certified version (Y=Yes, N=No)

	[3]	[4]	[5]	[6]	[9]	V1	V2
Ephemeral key attack	N	N	N	N	Y	Y	Y
Perfect forward secrecy	N	N	Y	Y	Y	Y	Y
No key escrow attack	Y	Y	N	N	N	Y	Y
Different CAs	Y	N	N	Y	Y	Y	Y
Global certified	N	N	N	N	N	N	Y

Fig. 1 Performance comparison for a varying number of participants (Note that the results of [5] are omitted due to the very high numbers.)

Table 2 Comparison of computational complexity, where V1 denotes the self certified version and V2 the global CA certified version

	Key construction	Encryption	Decryption
[3]	$2nT_p + (3n + 2)T_m$	$1T_{e2}$	$1T_p$
[4]	$nT_p + (3n - 1)T_m$	$1T_{e2} + 2T_m$	$2T_p$
[5]	$nT_p + 4nT_{e1}$	$1T_{e2} + 2T_{e1}$	$3T_p$
[6]	$3T_p + 5nT_m$	$2T_m + 1T_{e2}$	$1T_m + 2T_p$
[9]	$(9n + 2)T_m$	–	–
V1	$7nT_m$	$2T_m$	$1T_m$
V2	$(9n + 1)T_m$	$2T_m$	$1T_m$

Table 2 compares the number of most compute intensive operations between the proposed CAGKA protocol and [3–6, 9] for the key agreement phase, the encryption and decryption using the derived key. Denote the pairing operation as a function $e : G_1 \times G_1 \rightarrow G_1$ and define the EC also in G_1 . We then denote by T_m the time cost for a modular multiplication in G_1 , T_{e1} , T_{e2} the time cost for a modular exponentiations over G_1 , G_2 and T_p the time cost for a pairing operation. Note that for our protocol, we here consider the usage of the ECIES encryption protocol [19] for the encryption/decryption, while in [9] only a symmetric key encryption is required. Since this is negligible compared to the rest, we keep the corresponding entries empty in the table.

The impact on the efficiency between the different protocols increases with increasing number of participants and has been shown in Fig. 1. For instance, in our self-certified protocol, a group of 100 participants is able to derive the group key in only 11 ms, while it takes 14 ms, 21 ms, 440 ms, 876 ms and 1990 ms in [3–6, 9] respectively. Due to the high number of [5], the performance of this scheme is omitted in Fig. 1. The global CA certified version takes approximately 111 ms, which is still better than most of the other schemes. It should be noted that the scheme of [6] is very efficient, compared to the other pairing based schemes as it only requires a fixed number of pairing operations in the key construction, independent of the group size. For values of n larger than 410, the scheme of [6] even

Fig. 2 Number of transmitted bits (Note that the results of [5] are omitted due to the very high numbers.)

Fig. 3 Number of received bits

outperforms the proposed CAGKA scheme. To summarize, our protocol behaves best for the key agreement phase, even better than the SGKA protocol for group sizes smaller than 410. Also, with respect to encryption and decryption, the difference is huge compared to the other AGKA protocols.

6.2 Communication Costs

For analyzing the communication costs, we consider again the 160-bit security level.

Next, we also derived the communication cost with respect to the number of sent and received bits. Figures 2, 3 and 4 show the evolution in terms of groups size for the number of bits sent, received and total amount of bits sent and received respectively. With respect to sent bits, our both schemes outperform the other schemes, followed by the schemes of

Fig. 4 Total amount of sent and received bits

[3, 4, 6]. The number of bits sent by [9] is almost 10 times higher than CAGKA. The number of bits sent by [5] is almost 15 times higher than the CAGKA scheme. For the amount of received bits, the scheme of [3] slightly outperforms the CAGKA schemes. The scheme of [4] has also similar results with respect to number of transmitted bits. The other schemes have a significantly higher communication overhead. Combining then the numbers of sent and received bits, it can be concluded that the proposed CAGKA protocols outperform the other protocols. Only the protocol of [3] is in the same range, followed by [4]. The other schemes [5, 6, 9] have significant higher costs.

7 Conclusion

We have introduced a new cryptographic primitive, called the common asymmetric group key agreement protocol (CAGKA), which has clear advantages compared to both AGKA and SGKA protocols. First, it avoids key escrow, which is inherently present in SGKA protocols and second, it is much more efficient compared to AGKA protocols. We rely on ECC based mechanisms and develop by means of ECQV and a CK resistant mutual authentication protocol of [15], a one round self-certified pairing free CAGKA protocol. A global certification can be added after one additional round by applying the ECQV mechanism again. We show that these primitives outperform the state of the art with respect to security strength, computation and communication cost.

The scheme is very useful for scenarios of information sharing between users with different access rights, like eg. sharing of medical information among a group of experts. This can also be generalised to applications in wireless sensor networks, where the cluster heads take the role of participants and the sensors only need to securely transmit their information to this group. It can for instance result in facilitating the handover process of dynamic sensors crossing regions covered by the different cluster heads.

Declarations

Conflicts of interest This research has been funded by the Tetra grant OpenCloudEdge. There are no conflicts of interest. Additional data can be given on demand.

References

1. Wu, Q., Mu, Y., Susilo, W., Qin, B., & Domingo-Ferrer, J. (2009). Asymmetric group key agreement. In *Proceedings EUROCRYPT, Lecture Notes in Computer Science* (Vol. 5479, pp. 153–170). Springer.
2. Wu, Q., Mu, Y., Susilo, W., Qin, B., & Domingo-Ferrer, J. (2011). Provably secure one-round identity based authenticated asymmetric group key agreement protocol. *Information Sciences*, 181(19), 4318–4329.
3. Zhang, Q., Gan, Y., Zhang, Q., Wang, R., & Tan, Y. A. (2018). A dynamic and cross-domain authentication asymmetric group key agreement in telemedicine application. *IEEE Access*, 6, 24064–24074.
4. Lv, X., Li, H., & Wang, B. (2014). Authenticated asymmetric group key agreement based on certificateless cryptosystem. *International Journal Computer Mathematics*, 91(3), 447–460.
5. Ranjani, R. S., Bhaskari, D. L., & Avadhani, P. S. (2015). An extended identity based authenticated asymmetric group key agreement protocol. *International Journal Network Security*, 17(5), 510–516.
6. Chen, A., Hu, C., Chen, A., & Zheng, Q. (2021). An identity-based cross-domain authenticated asymmetric group key agreement. *Information*, 12, 112.
7. Carlier, M., Steenhaut, K., & Braeken, A. (2019). Symmetric-key-based security for multicast communication in wireless sensor networks. *Computers*, 8(1), 27.
8. Porambage, P., Braeken, A., Schmitt, C., Gurtov, A., Ylianttila, M., & Stiller, B. (2015). Group key establishment for enabling secure multicast communication in wireless sensor networks deployed for IoT applications. *IEEE Access*, 3, 1503–1511.
9. Mandal, S., Mhohant, S., & Majhi, B. (2020). CL-AGKA: Certificateless authenticated group key agreement protocol for mobile networks. *Wireless Networks*, 26, 3011–3031.
10. Zhao, X., Zhang, F., & Tian, H. (2011). Dynamic asymmetric group key agreement for ad hoc networks. *Ad Hoc Networks*, 9(5), 928–939.
11. Zhang, Q., Ma, Z., & Tan, Y. (2014). An authenticated asymmetric group key agreement for imbalanced mobile networks. *Ad Hoc Networks*, 23(4), 827–835.
12. Gan, Y., Wang, B., Zhuang, Y., Gao, Y., Li, Z., & Zhang, Q. (2020). An asymmetric group key agreement protocol based on attribute threshold for Internet of Things. *Trans Emerging Tel Tech*, 2020, e417.
13. Canetti, R., & Krawczyk, H. (2001). Analysis of key-exchange protocols and their use for building secure channels. In *EUROCRYPT, Lecture Notes of Computer Science* (Vol. 2045, pp. 453–474). Springer.
14. Qu, M., & Vanstone, S. (1997). Implicit Certificate Schemes, preprint
15. Braeken, A., Kumar, P., & Martin, A. (2018). Efficient and provably secure key agreement for modern smart metering communications. *Energies*, 11(10), 2662.
16. Burmester, M., & Desmedt, Y. (1994). A secure and efficient conference key distribution system. *Eurocrypt*, 1994, 275–286.
17. Brown, D. R. L., Gallant, R., & Vanstone, S. A. (2001). Provably secure implicit certificate schemes. In *Financial cryptography, lecture notes in Computer Science* (Vol. 2339, pp. 56–165). Springer.
18. Wei, G., Yang, X., & Shao, J. (2012). Efficient certificateless authenticated asymmetric group key agreement protocol. *KSII Transactions Internet Information Systems*, 6(12), 3352–3364.
19. Smart, N. P. (2001). The exact security of ECIES in the generic group model. In *Cryptography and coding, lecture notes in computer science* (Vol. 2260, pp. 73–84). Springer.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

An Braeken obtained her MSc Degree in Mathematics from the University of Gent in 2002. In 2006, she received her PhD in engineering sciences from the KULeuven at the research group COSIC (Computer Security and Industrial Cryptography). She became professor in 2007 at the Erasmushogeschool Brussel (currently since 2013, Vrije Universiteit Brussel) in the Industrial Sciences Department. Prior to joining the Erasmushogeschool Brussel, she worked for almost 2 years at the management consulting company Boston Consulting Group (BCG). Her current interests include security and privacy protocols for IoT, cloud and fog, blockchain and 5G security. She is (co-)author of over 120 publications. She has been member of the program committee for numerous conferences and workshops (IOP2018, EUC 2018, ICNS 2018, etc.) and member of the editorial board for Security and Communications magazine. She has also been member of the organizing committee for the IEEE Cloudtech 2018 conference and the Blockchain in IoT workshop at Globecom 2018. In addition, she is since 2015 reviewer for several EU proposals and ongoing projects, submitted

under the programs of H2020, Marie Curie and ITN. She has cooperated and coordinated more than 12 national and international projects. She has been STSM manager in the COST AAPELE project (2014–2017) and is currently in the management committee of the COST RECODIS project (2016–2019).