Zero-Knowledge and Identity-Based Authentication and Key Exchange for Internet of Things

Irfan Simsek, Erwin P. Rathgeb
Computer Networking Technology Group
University of Duisburg-Essen
Essen, Germany
{irfan.simsek, erwin.rathgeb}@uni-due.de

***Abstract*—The *Internet of Things* (IoT) is a dynamic and large-scale infrastructure of uniquely identifiable, potentially resource-constrained, and heterogenous things sensing and influencing their environment to provide services with or without direct human intervention. Moreover, the de facto method applied so far to provide communication confidentiality and integrity is cryptography. This requires an authentic key exchange between two communicating entities, which in turn needs a mutual authentication of the entities. In this paper, we introduce a novel approach supplying zero-knowledge and identity-based authentication with integrated key exchange while meeting the IoT challenges. Our approach is resistant to active man in the middle attacks and does not include any costly cryptographic operations. Also, it does not require any predistribution or pre-sharing with regard to authenticating entities. Additionally, our identity-based scheme allows that a thing can operate autonomously in a secure manner. Furthermore, this scheme provides application-independence without requiring additional components and procedures. After discussing our approach, this paper also presents our prototype implementation and its evaluation.**

***Index Terms*—Internet of Things, security, authentication, key exchange, zero-knowledge proof, identity-based key generation.**

I. Introduction

According to the existing discussions about the *Internet of Things* (IoT) [1], it can be stated that IoT is a network of dynamic networks of a huge number of addressable, uniquely identifiable, potentially resource-constrained, and heterogenous things sensing and influencing their environment to provide services with or without direct human intervention. This statement implies the IoT challenges to be met by IoT approaches. The first challenge is that IoT is a highly dynamic and very large-scale network of resource-constrained devices. Moreover, IoT is an infrastructure of heterogenous things. This means that there do not exist any fixed communication workflows, e.g., like in the Web. Thus, IoT approaches to be applied universally have to be application-independent. Additionally, IoT is an autonomous infrastructure. However, this does not mean that IoT can operate completely without human intervention. E.g., each thing has an owner/operator and needs a certain startup setup by its owner. After this setup, things have to be able to operate autonomously. Furthermore, things have to be uniquely identifiable. Thus, it is an advantage that IoT approaches follow an identity-based scheme. In addition, IoT is a service-providing infrastructure. Consequently, IoT needs infrastructure supporting instances for diverse functionalities such as service mapping and authorisation, just like in each service-providing infrastructure such as the conventional Internet.

978-1-5386-4980-0/19/$31.00 ©2019 IEEE

Besides these considerations for IoT, communication confidentiality and integrity are quite important security requirements. The de facto method applied so far is cryptography. However, this needs an authentic key exchange between two communicating entities, which in turn requires a mutual authentication of the entities. Since IoT is a highly dynamic and very large-scale infrastructure, an approach providing authentication and key exchange for IoT cannot be based on public data predistribution or secret pre-sharing between authenticating things. Moreover, it cannot rely on human interventions due to the autonomous character of IoT. With regard to the heterogeneity in IoT, the approach has to be application-independent without requiring additional components or procedures, except the de facto existing ones such as thing owner, infrastructure supporting instance, thing startup setup, service mapping, and authorisation. Additionally, the approach cannot include costly cryptographic operations due to resource-constrained devices in IoT. Furthermore, the approach has to be resistant to active man in the middle attacks which can be classified as the strongest adversary model.

This paper introduces a novel approach providing mutual authentication with integrated key exchange while meeting the IoT challenges and security requirements introduced above. Here, we combine zero-knowledge [2] and identity-based [3] schemes and thereby leverage the existing components as well as procedures described above. Especially, we expand the Goldreich-Micali-Wigderson (GMW) zero-knowledge protocol [4] for a mutual authentication and for an authentic key exchange. We choose the GMW protocol, since it does not include any costly cryptographic operations and is perfect zero-knowledge, i.e. resistant to malicious prover and verifier [4]. This implies that this protocol is suitable for resource-constrained devices and is resistant to active man in the middle attacks, which also apply to our approach. For authentication public data and secret generation, we follow an identity-based scheme. This provides that we do not need any public data predistribution or secret pre-sharing with regard to authenticating things. Moreover, this scheme allows that

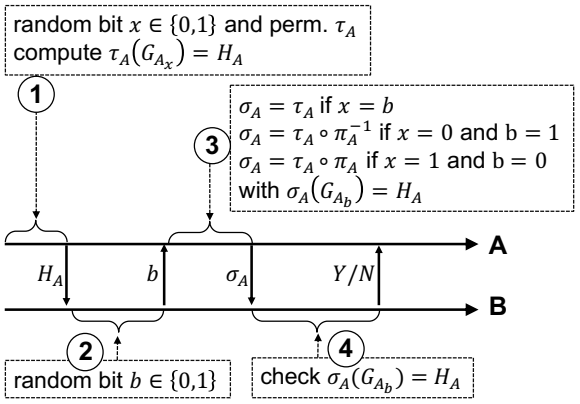

Fig. 1. The GMW zero-knowledge protocol.

a thing can operate autonomously in a secure manner after a startup setup by its owner. Additionally, our identity-based scheme supplies application independence without requiring additional components and procedures.

The rest of this paper is structured as follows. In Section II, we sketch the GMW protocol and discuss related work. Section III presents and Section IV analyses our approach. Our prototype implementation and our evaluation results are discussed in Section V. Finally, we conclude our paper and outline our future work in Section VI.

II. Related Work

While [5] gives a survey of IoT, overviews about IoT security and authentication protocols for IoT can be found in [6] and [7] respectively. Key exchange protocols, identity-based cryptography, and zero-knowledge protocols are studied in [8], [3], and [2] respectively.

Zero-knowledge protocols provide that an entity can prove the knowledge of its secret associated to public data without revealing any information about the secret. In general, the prover A randomly generates a witness and sends it to the verifier B. The witness defines challenges which A claims to be able to meet by means of its secret. B challenges A by randomly selecting one of the challenges defined by the witness. A supplies its answer, and B checks it for correctness. This can be iterated to decrease the cheating probability. Thus, zero-knowledge protocols are designed to be resistant to chosen-text attacks [9].

In identity-based cryptography, an entity A gets its public and private key pair from a trusted Key Generation Center (KGC). Here, KGC generates A's key pair by means of its own key pair and A's identity information. Moreover, A's public key is created in such a manner that any other entity can also generate A's public key by using KGC's public key and A's identity information. In this way, only KGC's public key has to be distributed authentically.

The Goldreich-Micali-Wigderson (GMW) zero-knowledge protocol [4] based on the graph isomorphism problem is one of the most popular zero-knowledge protocols. Two graphs $G_0 = (V_0, E_0)$ and $G_1 = (V_1, E_1)$ with n vertices are isomorphic, if there exists a permutation π with $(\pi(u), \pi(v)) \in E_1$ for each $(u, v) \in E_0$. There is no known polynomial time algorithm solving this problem for regular graphs. In the GMW protocol, an entity A with the secret π_A associated to the public data (G_{A_0}, G_{A_1}) with $\pi_A(G_{A_0}) = G_{A_1}$ proves to the entity B that it knows the secret without revealing any information about its secret. For that, A has to complete r rounds successfully, and each round proceeds as follows (see Figure 1):

1) First, A picks a random bit x and creates a random permutation τ_A. Subsequently, A generates $H_A = \tau_A(G_{A_x})$ and sends it to B.
2) Upon receiving H_A, B picks a random bit b and sends it to A.
3) Afterwards, A computes σ_A with $\sigma_A(G_{A_b}) = H_A$ and sends it to B. Here, $\sigma_A = \tau_A$ if $x = b$, $\sigma_A = \tau_A \circ \pi_A^{-1}$ if $x = 0$ and $b = 1$, and $\sigma_A = \tau_A \circ \pi_A$ if $x = 1$ and $b = 0$. [1]
4) After receiving σ_A, B checks whether $\sigma_A(G_{A_b}) = H_A$. If so, B sends YES. Otherwise, B sends NO and breaks up the round.

Here, a dishonest prover can cheat if he guesses b before sending the permuted graph. Thus, the cheating probability for each round is $1/2$, and it is $(1/2)^r$ for the entire protocol (e.g., $\approx 10^{-9}$ for $r = 32$). This protocol is perfect zero-knowledge, i.e. resistant to malicious prover and verifier, since b cannot depend on x, as τ_A is completely random [4].

The GMW zero-knowledge protocol is leveraged by diverse approaches. The most closely related one to our work is [10]. This approach provides authentication with key exchange but is only suitable for static networks due to predistribution, and thus does not fit the IoT challenges. Moreover, a certain part of τ_A is used for encoding of a public key part into H_A, and τ_A is thus not completely random. Furthermore, this approach can encode only $n/4$ bits in each round at most.

III. Construction

Our approach consists of multiple parts. First, we discuss how an unique permutation can be generated from a given identifier. The second part introduces identity-based generation of a public graph tuple and a secret permutation. After that, we describe the encoding of a given bit sequence into a graph by permuting a given graph. The next part expands the GMW zero-knowledge protocol for an authentic key exchange and for a mutual authentication. After outlining the thing startup setup and service registration as well as authorisation, the last part uses the introduced components to set up a secure channel between two things.

A. Identifier Permutation Generation

Any integer number $N < n!$ can be represented by an unique n-digit factoradic number $(d_{n-1}...d_0)$ of the

[1] $\circ$ denotes the product of two permutations, where the rightmost permutation is applied first.

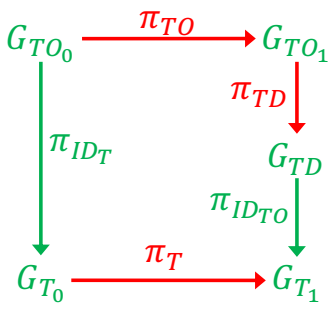

Fig. 2. Identity-based public data and secret generation.

form $N = d_{n-1} \cdot (n-1)! + ... + d_0 \cdot 0!$ with $d_i \leq i$. Lehmer [11] shows the correspondence between permutations and factoradic numbers. Thus, an unique permutation π of elements $\{0, ..., n-1\}$ can be generated from a factoradic number $(d_{n-1}...d_0)$. Here, for each element $0 \leq i \leq n-1$ we set $\pi(i)$ to $d_{n-1-i}{}^{\text{th}}$ element and remove this element from the element list.

However, up to 40! has to be computed, e.g., for a 160-bit identifier, which is not appropriate for a resource-constrained device. Therefore, the identifier is split into 4-bit blocks, and a factoradic number is generated for each 4-bit block. Thus, each 4-bit block value is smaller than 16, and only up to 3! has to be computed for each factoradic number. The permutation elements are also split into 4-element blocks. At last, a 4-sub-permutation is generated from each factoradic number as described above. Thus, the identifier permutation consists of the sub-permutations.

B. Identity-Based Public Data and Secret Generation

A thing owner TO creates a random k-regular undirected graph G_{TO_0} with $n = 2k+1$ vertices. Alternatively, the owner can permute a default k-regular undirected graph with his identifier n-permutation $\pi_{ID_{TO}}$ generated as described in Section III-A to create G_{TO_0}. After that, the owner generates a random n-permutation π_{TO} and computes $G_{TO_1} = \pi_{TO}(G_{TO_0})$. Here, $pub_{TO} = (G_{TO_0}, G_{TO_1})$ is the public data of the owner, and π_{TO} is its secret.

For each thing T with the identifier ID_T (thus, π_{ID_T} is T's identifier n-permutation) the owner

- chooses a random n-permutation π_{TD},
- creates $G_{TD} = \pi_{TD}(G_{TO_1})$,
- generates $G_{T_0} = \pi_{ID_T}(G_{TO_o})$ as well as $G_{T_1} = \pi_{ID_{TO}}(G_{TD})$, and
- computes $\pi_T = \pi_{ID_{TO}} \circ \pi_{TD} \circ \pi_{TO} \circ \pi_{ID_T}^{-1}$.

Here, $pub_T = (G_{T_0}, G_{T_1})$ is T's public data, and π_T is its secret with $\pi_T(G_{T_0}) = G_{T_1}$. The relationship between the graphs and the permutations is given in Figure 2. The owner gives $\{G_{TD}, (G_{T_0}, G_{T_1}), \pi_T\}$ to the thing but keeps π_{TD} secret. Here, we want to point out that π_{TD} is needed to provide that the thing cannot find out the secret of the owner by using its own secret. Additionally, π_T cannot be computed in polynomial time without knowing π_{TO} and π_{TD}. Thus, π_T can only be created by the thing owner. Furthermore, everyone having ID_{TO}, ID_T, and (G_{TO_0}, G_{TD}), which can be known publicly, can generate (G_{T_0}, G_{T_1}).

C. Bit Encoding through Graph Permutation

For the adjacency matrix $G[][]$ of a given k-regular undirected graph G with $n = 2k+1$ vertices and for a given bit sequence $(b_0...b_{k-1})$ our goal is to compute a permutation κ called *key permutation* with $\kappa(G) = H$ and $(H[0][1]...H[0][k]) = (b_0...b_{k-1})$. It is clear that such a permutation exists, since G is a k-regular graph with $2k+1$ vertices and thus has exact k set bits in its first row. First, we set $\kappa(0)$ to 0, i.e. $\kappa(0) := 0$. For each b_i with $0 \leq i \leq k-1$ we check whether $G[0][i+1] = b_i$. If so, we set $\kappa(i+1) := i+1$. Otherwise, we look for an unused column $j > k$ with $G[0][j] = b_i$ and set $\kappa(i+1) := j$ and $\kappa(j) := i+1$. Thus, the column is marked as used. The complete procedure is given by Algorithm 1. In this way, we can encode $(n-1)/2$ bits into a k-regular graph with $n = 2k+1$ vertices.

Algorithm 1 Bit encoding

```
b := {b_0, ..., b_{k-1}}
pb := {G[0][k+1], .., G[0][n-1]}
c := {k+1, ..., n-1}
κ(0) := 0
for i = 0 to k-1 do
  if G[0][i+1] == b[i] then
    κ(i+1) := i+1
  else
    for l = 0 to pb.length do
      if pb.at(l) == b[i] then
        κ(i+1) := c.at(l)
        κ(c.at(l)) := i+1
        pb.remove(l)
        c.remove(l)
        break
```

D. Mutual Authentication with Key Exchange

The entities A and B aim to mutually authenticate each other and to authentically exchange their public keys pk_A and pk_B with a bit length of $r \cdot k$. Here, they mutually prove the knowledge of their secrets π_A and π_B associated to their public data $pub_A = (G_{A_0}, G_{A_1})$ and $pub_B = (G_{B_0}, G_{B_1})$ with $\pi_A(G_{A_0}) = G_{A_1}$ and $\pi_B(G_{B_0}) = G_{B_1}$. The public data and secrets are generated as described in Section III-B. We first assume that A and B already know their public data (G_{A_0}, G_{A_1}) and (G_{B_0}, G_{B_1}). In the next sections, we discuss how they can generate the public data of each other by means of their identifiers. Before proceeding with the authentication and key exchange, A and B split their public keys into k-bit blocks $\{pk_{A_0}, ..., pk_{A_{r-1}}\}$ and $\{pk_{B_0}, ..., pk_{B_{r-1}}\}$. The entities have to complete r rounds successfully, and each round R_i proceeds as follows (see Figure 3):

1) A picks a random bit x as well as a random permutation τ_A and generates $\tau_A(G_{A_x}) = \widetilde{G}_{A_x}$. Additionally, A

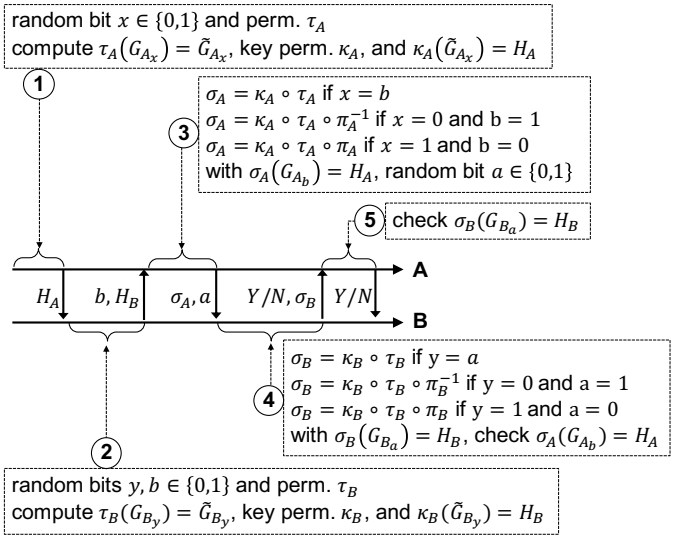

Fig. 3. Mutual zero-knowledge authentication with key exchange.

computes the key permutation κ_A for $\widetilde{G}_{A_x}$ and for its public key part pk_{A_i} as described in Section III-C. Furthermore, A creates $\kappa_A(\widetilde{G}_{A_x}) = H_A$ and sends it to B.

2) After receiving H_A, B proceeds analogously and thus creates H_B. Additionally, B picks a random bit b and sends $\{b, H_B\}$ to A.
3) Upon receiving $\{b, H_B\}$, A computes the permutation σ_A with $\sigma_A(G_{A_b}) = H_A$. Here, $\sigma_A = \kappa_A \circ \tau_A$ if $x = b$, $\sigma_A = \kappa_A \circ \tau_A \circ \pi_A^{-1}$ if $x = 0$ and $b = 1$, and $\sigma_A = \kappa_A \circ \tau_A \circ \pi_A$ if $x = 1$ and $b = 0$. Additionally, A picks a random bit a and sends $\{\sigma_A, a\}$ to B.
4) Analogously, B computes σ_B with $\sigma_B(G_{B_a}) = H_B$. After that, B checks whether $\sigma_A(G_{A_b}) = H_A$. If so, B sends $\{YES, \sigma_B\}$ to A. Otherwise, B sends $\{NO\}$ and breaks up the round.
5) If A gets the YES message, it checks whether $\sigma_B(G_{B_a}) = H_B$. If so, A sends $\{YES\}$ otherwise A sends $\{NO\}$ and breaks up the round.

E. Thing Startup Setup

In each service-providing infrastructure such as IoT, there exist infrastructure supporting instances (ISI) for various functionalities such as service mapping and authorisation management. In our approach, a thing owner TO registers himself with such an instance (see Figure 4). To do so, the owner sets up an authentic channel (e.g., by means of Transport Layer Security (TLS) [12]) to the instance ISI with the identifier ID_{ISI} and sends his identifier and public data $pub_{TO} = (G_{TO_0}, G_{TO_1})$. After receiving that, ISI sends its own public data $pub_{ISI} = (G_{ISI_0}, G_{ISI_1})$ to the owner. TO and ISI generate their public data as introduced in Section III-B. Thus, π_{TO} is TO's secret with $\pi_{TO}(G_{TO_0}) = G_{TO_1}$, and π_{ISI} is ISI's secret

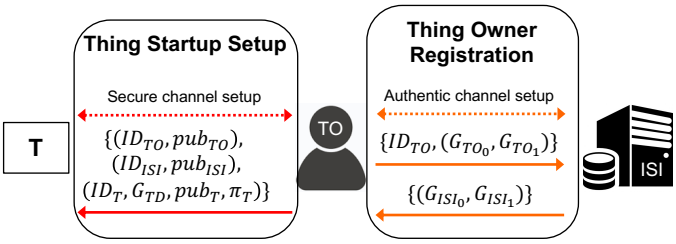

Fig. 4. Startup setup.

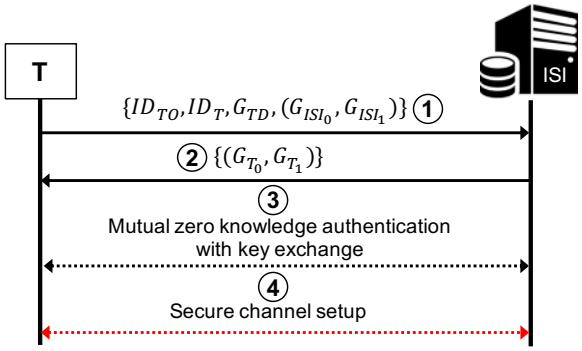

Fig. 5. Thing registration.

with $\pi_{ISI}(G_{ISI_0}) = G_{ISI_1}$. Here, we want to point out that the owner performs this registration only once for himself.

Each device providing a service needs a certain startup setup to perform its functionalities as desired. Thus, for each thing T with the identifier ID_T its owner generates G_{TD} as well as T's public data $pub_T = (G_{T_0}, G_{T_1})$ and secret π_T as described in Section III-B. After that, the owner sets up a secure channel (e.g., based on a one-time password, or by using near field communication) to the thing and sends $\{(ID_{TO}, pub_{TO}), (ID_{ISI}, pub_{ISI}), (ID_T, G_{TD}, pub_T, \pi_T)\}$ (see Figure 4). By means of (ID_{ISI}, pub_{ISI}), the thing gets to know with which instance it has to register its services and their authorisation.

F. Thing Service Registration and Authorisation

After the startup setup (see Section III-E), services provided by a thing T with the owner TO and authorisations for service use have to be securely registered with the instance ISI. In our approach, the thing can autonomously perform this as follows (see Figure 5):

1) T sends $\{ID_{TO}, ID_T, G_{TD}, (G_{ISI_0}, G_{ISI_1})\}$ to ISI.
2) If the thing owner TO is successfully registered, ISI generates (G_{T_0}, G_{T_1}) by using ID_{TO}, ID_T, and (G_{TO_0}, G_{TD}) (see Section III-B) and sends it to the thing. Thus, each of T and ISI gets a challenge, i.e. proving to know their respective secrets π_T and π_{ISI} associated to their respective public data $pub_T = (G_{T_0}, G_{T_1})$ and $pub_{ISI} = (G_{ISI_0}, G_{ISI_1})$.
3) Subsequently, T and ISI mutually authenticate each other and authentically exchange their public keys as

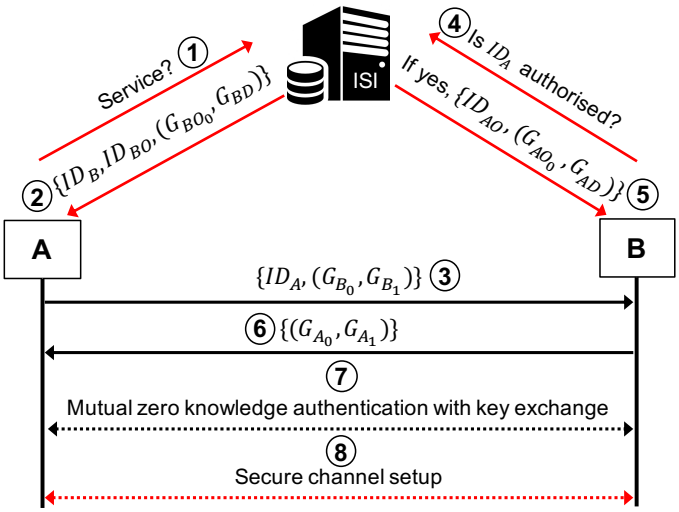

Fig. 6. Thing-to-thing communication.

introduced in Section III-D.

4) T and ISI set up a secure channel by generating a symmetric session key by using their public keys, e.g., just like in TLS.

The thing can then use this channel to securely register itself with the instance. Here, we want to point out that the owner can also set up a secure channel with his thing in a similar way, e.g., to securely reconfigure it.

G. Thing-to-Thing Communication

We assume that the things A and B already have set up a secure channel with the instance ISI and registered themselves with ISI as introduced in Section III-F. A with the identifier ID_A aims to securely use a service provided by B with the identifier ID_B. Here, the identifiers of A's owner AO and B's owner BO are ID_{AO} and ID_{BO} respectively. Moreover, AO's and BO's public graph tuples are (G_{AO_0}, G_{AO_1}) and (G_{BO_0}, G_{BO_1}). Additionally, G_{AD} and G_{BD} are generated and registered with ISI as introduced in Section III-B and in Section III-F. To set up a secure channel between A and B, the following procedure is used (see Figure 6):

1) A queries ISI for a thing providing the service.
2) Upon receiving the request, ISI responds with $\{ID_B, ID_{BO}, (G_{BO_0}, G_{BD})\}$.
3) By using ISI's response, A generates B's challenge, i.e. B's public data $pub_B = (G_{B_0}, G_{B_1})$ (see Section III-B) and queries B for the service by sending a request message containing $\{ID_A, (G_{B_0}, G_{B_1})\}$.
4) For the incoming request, B asks ISI whether A is authorised to use the service.
5) If A is authorised, B gets $\{ID_{AO}, (G_{AO_0}, G_{AD})\}$.
6) Subsequently, B analogously generates A's challenge, i.e. A's public data $pub_A = (G_{A_0}, G_{A_1})$ and sends it to A.
7) A and B mutually authenticate each other and exchange their public keys as described in Section III-D.
8) A and B set up a secure channel based on a symmetric session key generated by using their public keys.

Afterwards, A can begin to securely use B's service.

IV. Analysis

We have expanded the GMW protocol for a mutual authentication and for an authentic key exchange. Since here the bits selected by authenticating entities still cannot depend on each other, and their τ-permutations are still completely random, our approach is also perfect zero-knowledge like the GMW protocol and thus is resistant to malicious prover and verifier. Due to this property, our approach is resistant to an active man in the middle attack which can be classified as the strongest adversary model.

Our construction also allows that authenticating entities can encode parts of their public keys into the H-graphs during the mutual authentication. Since the H-graphs act as witnesses for the knowledge of the secret permutations associated to the public graph tuples of the entities, any modification of the H-graphs by a third party leads to an authentication failure. Thus, our approach does not only provide mutual authentication but also supplies authentic exchange of public keys, e.g., generated by using *Elliptic Curve Cryptography* (ECC) [13]. Moreover, successfully authenticated entities can utilise the public keys of each other to generate a symmetric session key in order to set up a secure channel. This can provide confidentiality and integrity for communication between a thing and its owner, between a thing and an infrastructure supporting instance, as well as between two things. Here, we want to point out that our approach also allows to integrate a key agreement protocol such as the Diffie-Hellman protocol [14].

Our approach follows an identity-based scheme and leverages only the de facto existing components such as thing owner and infrastructure supporting instance as well as procedures such as thing startup setup, service registration and authorisation. In this way, we do not need any public data predistribution or secret pre-sharing with regard to authenticating things. Thus, our approach is very well suited for highly dynamic and large-scale networks such as IoT and is application-independent without requiring additional components or procedures.

Due to the identity-based scheme, another benefit provided by our approach is that a thing owner has to register himself with an ISI only once, and his things can autonomously register their services by the ISI in a secure manner. Thus, our approach requires fewer human intervention, which is a quite important benefit for IoT.

V. Implementation and Evaluation

In order to demonstrate the feasibility of our construction, we have implemented the mutual authentication with key exchange and the related parts (Section III-A, III-B, and III-C) in Contiki [15] which is one of the widespread operating systems for IoT. In our implementation, a graph with n vertices requires $n \cdot \lceil n/8 \rceil$ bytes. Moreover, we have only used simple list operations of polynomial time to generate permutations,

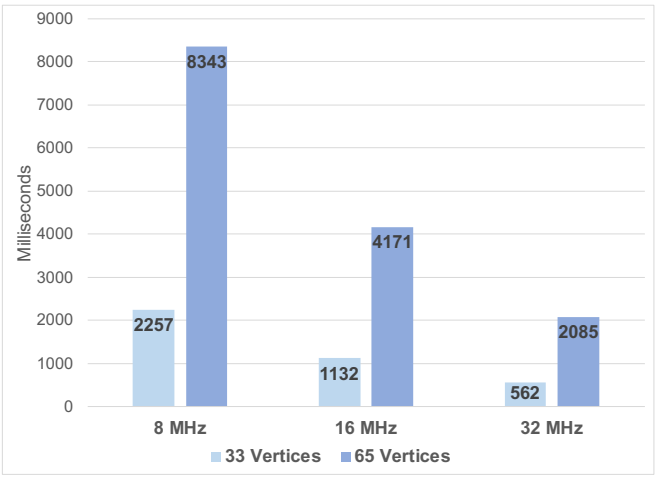

Fig. 7. Execution times for mutual authentication with key exchange.

to permute graphs, and to encode bit sequences.

We have evaluated the performance of our implementation on TI MSP430x [16], an ultra-low power microcontroller (MCU) emulated by the Contiki COOJA simulator [17]. Figure 7 shows the execution times (without communication overheads) for the mutual authentication with key exchange for graphs with 33 and 65 vertices as well as for the MCU frequencies 8 MHz, 16 MHz, and 32 MHz. Here, we can observe that the execution time decreases when the MCU frequency increases and that the execution time increases when the number of vertices increases. In our evaluation, authenticating entities have to complete 32 rounds successfully, which implies a cheating probability of more less than 10^{-9}. Moreover, brute-force searches for secret permutations between graphs with 33 and 65 vertices require more steps than 10^{36} and 10^{90}. Thus, this basic experiment proves the general feasibility and the quite good performance of our approach with a strong security.

In sum, it can be stated that our approach provides mutual authentication as well as authentic key exchange with regard to a strong adversary model while meeting the IoT challenges such as dynamic, scalability, application independence, autonomy, and resource constraint.

VI. Conclusion and Future Work

This paper discussed the challenges as well as the security requirements in IoT and introduced a novel approach for zero-knowledge and identity-based authentication with key exchange while meeting the IoT challenges and security requirements. Moreover, we have demonstrated the feasibility and performance of our approach by implementing it in Contiki and by evaluating it on an ultra-low power microcontroller emulated by the Contiki COOJA simulator.

In the future, we are going to optimise our approach with regard to the number of rounds to be performed during the authentication while retaining the same security level. Moreover, we aim to integrate our optimised approach into common IoT protocols and architectures [5]. Furthermore, we want to develop systems for thing service registration, discovery, and authorisation as well as for thing identifier structure and management on the basis of our approach.

References

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] O. Goldreich, "Zero-knowledge twenty years after its invention." *IACR Cryptology ePrint Archive*, vol. 2002, p. 186, 2002.

[3] J. Baek, J. Newmarch, R. Safavi-Naini, and W. Susilo, "A survey of identity-based cryptography," in *Proc. of Australian Unix Users Group Annual Conference*, 2004, pp. 95–102.

[4] O. Goldreich, S. Micali, and A. Wigderson, "Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems," *Journal of the ACM (JACM)*, vol. 38, no. 3, pp. 690–728, 1991.

[5] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[6] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, "Internet of things security: A survey," *Journal of Network and Computer Applications*, vol. 88, pp. 10–28, 2017.

[7] M. A. Ferrag, L. A. Maglaras, H. Janicke, J. Jiang, and L. Shu, "Authentication protocols for internet of things: a comprehensive survey," *Security and Communication Networks*, vol. 2017, 2017.

[8] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2001, pp. 453–474.

[9] J. Katz, A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. CRC press, 1996.

[10] P. Flood and M. Schukat, "Peer to peer authentication for small embedded systems," in *Proceedings of 10th international conference on digital technology*, 2014, pp. 68–72.

[11] D. H. Lehmer, "Teaching combinatorial tricks to a computer," in *Proc. Sympos. Appl. Math. Combinatorial Analysis*, vol. 10, 1960, pp. 179–193.

[12] E. Rescorla, "The transport layer security (tls) protocol version 1.3," RFC 8446, Tech. Rep., 2018.

[13] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.

[14] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[15] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*. IEEE, 2004, pp. 455–462.

[16] *MSP430x5xx and MSP430x6xx Family User's Guide*, Texas Instruments, 2018.

[17] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in *Local computer networks, proceedings 2006 31st IEEE conference on*. IEEE, 2006, pp. 641–648.