Zero-Knowledge Proofs in M2M Communication

Michael Schukat[1], Padraig Flood[2]

[1]*Department of Information Technology*
NUI Galway, Galway, Ireland
Email: michael.schukat@nuigalway.ie

[2]*Department of Information Technology*
NUI Galway, Galway, Ireland
Email: p.flood1@nuigalway.ie

***Abstract*— The advent of the IoT with an estimated 50 billion internet enabled devices by the year 2020 raises questions about the suitability and scalability of existing mechanisms to provide privacy, data integrity and end-entity authentication between communicating peers. In this paper we present a new protocol that combines zero-knowledge proofs and key exchange mechanisms to provide secure and authenticated communication in static M2M networks, therefore addressing all the above problems. The protocol is suitable for devices with limited computational resources and can be deployed in wireless sensor networks.**

While the protocol requires an a-priori knowledge about the network setup and structure, it guarantees perfect forward secrecy.

***Keywords* – Zero knowledge proof, GMW protocol, Diffie Hellman key exchange**.

I INTRODUCTION

Privacy, data integrity and end-entity authentication are essential features of computer communication. The de-facto internet standard to provide these features is based on TLS [1], digital certificates [2] and public key infrastructures (PKI). However, all these standardisation efforts have been hampered by a range of design and implementation insufficiencies / flaws. Recently disclosed vulnerabilities include Apple's iOS SSL bug [3], null-prefix attack on SSL certificates [4] and the Commodo certificate authority breach [5].

The advent of the Internet of Things with an estimated deployment of 50 billion devices globally by 2020 [6] only will only enlarge these problems, or, expressed in a metaphor: "How are we supposed to ride a motorbike (50 billion estimated networked devices), if we can't even handle a bicycle (a mere 12 billion internet-enabled devices as of today)?"

Another twist is that the IoT revolution will largely be based on low-cost, poorly protected and un-supervised embedded systems (ES) deployed across many different application domains. These ES are potentially far more vulnerable to malicious attacks than for example the standard internet-enabled PC, which at least has some decent defense mechanism in form of a built-in firewall or virus scanner.

Also, many of these deployments will operate on wireless standards based on 802.11 and 802.15, which are prone to eavesdropping, packet injections, selective scrambling, etc., therefore requiring strong link-layer encryption (e.g. AES-128).

There are also question marks over proposed cipher suits that are suitable for resource constrained devices: While elliptical curve cryptography (ECC) requires in principal much shorter keys to provide the same level of security as classical algorithms like for example RSA, there is a certain amount of scepticism in the security community about the underlying elliptic curves [7] as standardised by NIST and other organisations.

Static small to medium sized machine-to-machine (M2M) networks can in principal incorporate privacy, data integrity and end-entity authentication by means of one or more pre-shared symmetric link keys, which are deployed on all network devices or end-entities prior to their deployment. While this approach has been successfully applied in ZigBee [16], it does not provide perfect forward secrecy [11], which is deemed to be a necessity after the recent disclosure of widespread capture and storage of encrypted network communication for retrospective cryptanalysis as done by various government agencies.

In this paper we propose an alternative method for peer-to-peer authentication and encryption based on the Goldreich-Micali-Wigderson graph isomorphism zero-knowledge protocol and the Diffie-Hellman key exchange. It is structurally similar to a pre-shared symmetric link-key approach. While our method lacks the flexibility

of a public-key cryptosystem (e.g. our key directories have to be setup pre-deployment), it avoids the complexity of a public-key infrastructure-based approach_and therefore may have greater potential for small embedded systems.
In summary, it provides end-entity authentication and session key negotiation between peers over a potentially unsecure communication channel. Because of the zero-knowledge property no security credentials are exchanged, while compromised devices can only reveal their own credentials, but not any shared keys.

II ZERO-KNOWLEDGE PROOFS

Zero-knowledge proofs (ZKP) [8] are challenge / response authentication protocols, in which parties are required to provide the correctness of their secrets without revealing these secrets.
During the authentication procedure, a prover (e.g. Alice) must respond to challenges issued by a verifier (e.g. Bob) over a number of accreditation rounds.

Alice must be able to answer all challenges successfully to prove her identity. Bob's confidence in Alice's identity increases with every single round.
In zero-knowledge proof protocols, the verifier cannot learn anything from the authentication procedure. Moreover, the verifier is unable to cheat the prover because he cannot calculate the prover's secret. Furthermore, the verifier cannot cheat the prover because the protocol is repeated as long as the verifier is not convinced; due to the random selection of a challenge selection the verifier cannot pretend to be the prover to a third party.
The computational overhead required for Alice to prove her identity can potentially be significantly less than that required for other approaches of authentication without the need for a trusted third party such as that of RSA [15], while still remaining very difficult for an intruder to cheat (due to being based upon NP problems). Due to these characteristics, ZKPs were immediately considered to be very well suited to resource-limited systems (e.g. smart cards) [13][14].

III THE GOLDREICH-MICALI-WIGDERSON ZKP

The GMW protocol is based on graph isomorphism [12]. Two graphs $H = (V_1,E_1)$ and $G_0 = (V_2,E_2)$ that have the same number of vertices are isomorphic, if there exists a permutation π_0 on vertices of H, so that any edge between vertices (u, v) in H can be mapped onto G_0. An example is depicted in Figure 1 with the corresponding edges labelled.

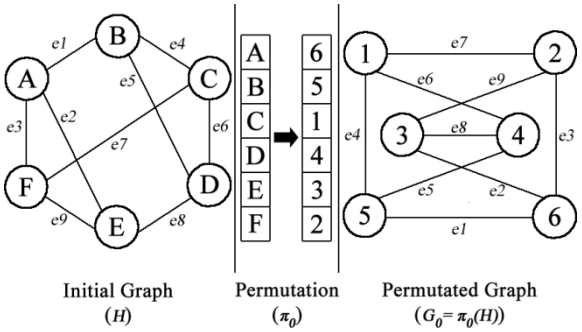


Figure 1: Building an isomorphic graph of H (G_0) using a permutation (π_0)

The graph isomorphism problem is NP, there is no known polynomial time algorithm that solves it. For example, an exhaustive search for an isomorphism between 2 regular graphs with 34 vertices is computationally as complex as the exhaustive search for a 128 bit symmetric key (~3E38 steps).

In the GMW protocol the prover's secret is a graph permutation π that is the isomorphism between two publically known graphs G_0 and G_1. At the beginning of an accreditation round the prover generates a random permutation ϱ and transforms either of the 2 graphs (e.g. a= 0 or a = 1 in Figure 2). The resulting *initiation graph* H is sent to the verifier. The verifier in turn picks randomly either graph G_0 or G_1 (e.g. b = 0 or b = 1 in Figure 2) and asks the prover to show an isomorphism between this graph and H. Depending on the (randomly picked) values for a and b the prover might or might not require the secret graph permutation π to provide the correct answer (e.g. permutation σ), which is sent back to the verifier and validated by him.

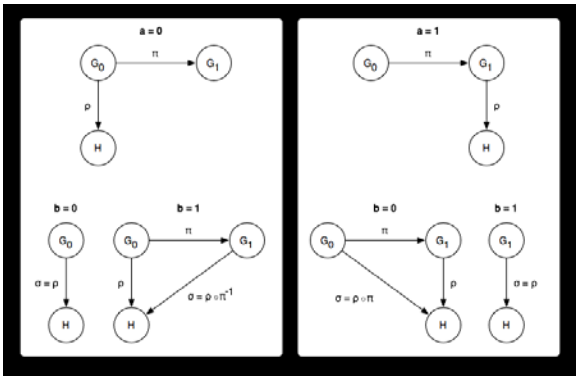


Figure 2: The GMW Protocol

IV THE KEY NEGOTIATION PROTOCOL

The GMW protocol allows end-entity authentication, but provides no further means to authenticate or encrypt data for inter-link communication.

Diffie-Hellman (see Figure 3) on the other hand is a common key negotiation protocol. For our considerations it is assumed that the multiplicative group of integers modulo p (with p being a prime) as well as the primitive root g is known to all devices.

RFC 2412 [9] defines a set of recommended parameters with p being either 768 bit or 1024 bit long.

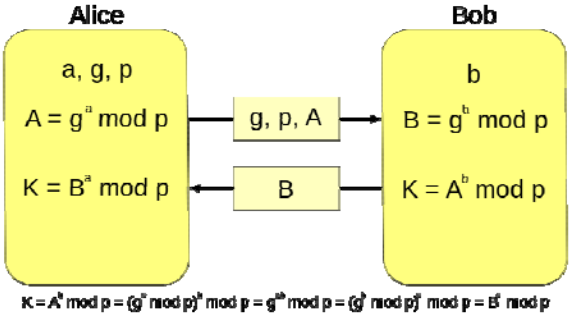


Figure 3: Diffie-Hellman Key Exchange

V THE PROPOSED PROTOCOL

The proposed protocol is designed to operate on static deployments consisting of a fixed number of interconnected devices.

Prior to deployment a network manager creates a random regular public graph G_P (in a regular graph all vertices have the same degree, e.g. the same number of edges) that is stored in all devices. Each device i also receives a random secret permutation s_i. The resulting graph permutation $G_i = s_i(G_p)$ is stored in a hash table (with a device's MAC address as the hash key). The completed hash table (consisting of n entries for n devices) is stored on all devices.

After the deployment of a network a peer-to-peer connection begins with a handshake initiated by a device. Since we have a mutual authentication each device is both prover and verifier. During the handshake both entities extract the other party's MAC address to retrieve its public graph G_i. They negotiate the number of authentication rounds and define a session identifier ***Id***.

In each round a device sends an authentication frame to the other device. It has the following format:
< IG || i || Id || Success || SOL || CHL >

- **IG** is the initiation graph for round i.
- i is the round index.
- **Id** is the unique session identifier.
- ***Success*** is a flag indicating if the other device could successfully answer the last challenge.
- ***SOL*** is the device's response to the other side's last challenge.
- ***CHL*** is the new challenge for the other device.

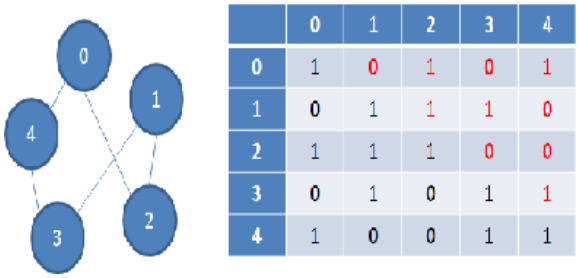
Note that the first authentication frame does not contain the fields ***Success*** and ***SOL***, while the last frame does not contain the fields ***IG*** and ***CHL***.

The handshake fails, if any side cannot respond to the other side's challenge correctly, e.g. if ***Success*** is set to zero.

VI PROTOCOL OPTIMISATION

Following the GMW protocol in each round a permutation σ needs to be send from the prover to the verifier, therefore the permutation must be serialised.

σ can be interpreted as a byte vector of length y, with y being the number of vertices of the graph.
Alternatively, a memory-efficient encoding scheme for $|\sigma| = 2^X$ can be used, as only X bits per vertice are required.
For example, a permutation for a graph with 32 vertices requires either 256 bits if stored uncompressed in a simple array, or 160 bits if stored efficiently.



	0	1	2	3	4
0	1	0	1	0	1
1	0	1	1	1	0
2	1	1	1	0	0
3	0	1	0	1	1
4	1	0	0	1	1

Figure 4: Graph Representation

Multiple graphs have to be stored efficiently in a device's hash table, while in each round an initiation graph has to be transmitted from the prover to the verifier.

A graph can be represented in a binary matrix as shown in Figure 4. Because of the symmetry of the matrix only the upper right triangle has to be considered (e.g. serialised), therefore only requiring $l_g = (y^2 / 2) – (y / 2)$ bits of storage for a graph with y vertices.

VII KEY EXCHANGE

So far the protocol provides zero-knowledge authentication of two peers, but it does not provide any means for the encryption (using a symmetric key) or authentication (using an encrypted hash) of exchanged data after the completion of the authentication phase.

Since all peers share a potential unsecure or open network medium, replay, injection and packet manipulation attacks by a third party are possible. While this has limited consequences for the ZK authentication (i.e. it results eventually in the termination of the authentication protocol because an incorrect response has been injected) it means that the key exchange (e.g. public keys A and B in Figure 2) has to be tightly knitted into the authentication phase. If both authentication and key exchange are independent (and for example only share a common single data frame for data transport), a potential attacker could capture this frame, replace the public key material with its own public key, and re-inject

the frame into the network, therefore creating a classical man-in-the-middle attack situation.

This problem can be averted by piecewise encoding the public key of a given device into its serialised initiation graph. Over multiple rounds the entire public key will be transferred and because the key is tied to the ZKP a manipulation by a third party is not possible.

The implementation of this concept is as follow: Assuming a public key segment consists of k bits with $k << l_g$ (e.g. $k < l_g / 4$) we pick a permutation (and consequently build an initiation graph), so that the first k bits of the serialised graph match the public key segment.

	0	1	2	3	4
0		1_2	0_3	0_4	1_5
1			1_7	1_8	0_9
2				0_{11}	0_{12}
3					1_{14}
4					

Figure 5: Initial Graph before Permutation

Figure 5 to Figure 7 show in a simple example how a given 2-bit public key segment {0, 1} can be embedded into an initiation graph. All cell values are indexed to increase readability, while swapped cells are highlighted in red.

The first bit (1_2) needs to be 0, therefore vertice 1 is swapped with vertice 3. The resulting graph can be seen in Figure 6.

	0	1	2	3	4
0		0_4	0_3	1_2	1_5
1			0_{11}	1_8	1_{14}
2				1_7	0_{12}
3					0_9
4					

Figure 6: Initial Graph after 1st Permutation

In the next step the second bit has to be set to 1 and therefore vertice 2 is swapped with vertice 4 (Figure 7).

The resulting permutation vector is {0, 3, 4, 1, 2}.

In contrast to the example above the majority of permutations are purely random and do not encode

	0	1	2	3	4
0		0_4	1_5	1_2	0_3
1			1_{14}	1_8	0_{11}
2				0_9	0_{12}
3					1_7
4					

Figure 7: Final Graph after 2nd Permutation

parts of the public key. Also, every vertice can only be swapped once and only vertices with an index > k can be swapped to accommodate the encoded public key. The number of vertices of the graph must be significantly larger than k in order to provide sufficient swap candidates; the example in Figure 5 – Figure 7 has been deliberately chosen to work for the public key segment {0, 1} – it would fail for the public key segment {0, 0}.

VIII PERFORMANCE CHARACTERISTICS

The generation of a permutation and the generation of a permutated graph are simple list operations that can be done in polynomial (e.g. linear) time. From this perspective the protocol is well suited for resource-constrained devices.

However, the authentication protocol creates some communication overheads as shown in Table 1. The results shown in the table are based on

- an authentication phase consisting of 32 rounds (which results in a verifier confidence of 100 x (1 - 1 / 2^{32}) % (~99.99999999%) about the true identity of the prover),
- an uncompressed response permutation and
- protocol overheads (like for example the session ID) of another 10 bytes per packet.

In the current implementation we utilise graphs with the order 128 to transport 24 bits of a public key per round (e.g. 768 bits over 32 rounds).

On the other hand an initiation graph of the order 64 seems to be the most economic solution, while providing an algorithmic complexity (in terms of an exhaustive search of a graph isomorphism), which is in the order of 10E11 larger than a brute force attack on a 256 bit symmetric key.

However, any key exchange algorithm (like Diffie Hellman) creates computational overheads that are well documented in literature [17] and in this regard a conventional pre-shared link-key approach is superior. On the other hand, pre-shared link-keys do not provide perfect forward secrecy [11].

#IG	IG Serialised Length [Bytes]	Response Permutation [Bytes]	Protocol Overheads [Bytes]	Total per round in both Directions [Bytes]	Total over 32 rounds [Bytes]
32	126	32	10	336	10752
64	252	64	10	652	20864
128	1016	128	10	2328	74496
256	4080	256	10	8692	278144

Table 1: Protocol Communication Overheads

IX CONCLUSION AND FUTURE WORK

In this paper we propose a new method to provide privacy, data integrity and end-entity authentication among peers in a static M2M network. It is based on a zero-knowledge proof and has two unique features, e.g. it provides mutual authentication based on the GMW protocol, while integrating a public key transport mechanism for a complementary key negotiation protocol. In its current implementation the protocol uses Diffie-Hellman key-exchange. Cryptographically strong Diffie-Hellman implementations require public keys in the order of 768 or 1024 bits (as recommended in RFC2412), which result in combination with the protocol in either large graphs or an appropriate number of rounds for key exchange, therefore resulting in communication overheads as indicated in Table 1.

As a result we are currently investigating the use of an alternative key negotiating protocol based on Curve25519. Curve25519 was first published by Daniel J. Bernstein [10] as an alternative to NIST approved ECC curves. Curve25519 is designed for the elliptic curve Diffie–Hellman key agreement scheme (ECDH), but requires only parameters of 256 bit length, while providing similar levels of security as RFC2412.

Another possibility to improve the performance and bandwidth requirements of the proposed protocol is an optimal balance between the length of public key segments and the order of the initiation graph. This requires further analysis of the dependency between the size of the graph, the order of its vertices, the length of a public key segment and its composition.

REFERENCES

[1] Dierks, T. and E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.2,” RFC 5246, August 2008

[2] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,” RFC 5280, May 2008

[3] Apple Computers, “About the security content of iOS 7.0.6”, http://support.apple.com/kb/HT6147

[4] M. Marlinspike, “More Tricks For Defeating SSL”, DEFCON 17, Las Vegas, 2009

[5] Comodo Group, “Report of incident on 15-MAR-2011”, , https://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html

[6] Cisco, “The Internet of Things”,http://share.cisco.com/internet-of-things.html

[7] B.Schneier, “Schneier on Security” https://www.schneier.com/blog/archives/2013/11/elliptic_curve.html#c2200076

[8] S. Goldwasser, S. Micali, and C. Rackoff, “The knowledge complexity of interactive proof-systems”. In STOC '85: Proceedings of the seventeenth annual ACM symposium on Theory of computing, pages 291-304, New York, NY, USA, 1985. ACM Press.

[9] H. Orman, “The OAKLEY Key Determination Protocol” RFC 2412, November 1998

[10] D. J. Bernstein. “Curve25519: new Diffie-Hellman speed records”, Proceedings of PKC 2006

[11] A. Sui, “An improved authenticated key agreement protocol with perfect forward secrecy for wireless mobile communication”, Wireless Communications and Networking Conference, 2005 IEEE

[12] O. Goldreich, S. Micali, A. Widgerson, “Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems” Journal of the ACM, Vol. 38, No. 3, July 1991, pp. 691-729

[13] H. Aronsson, “Zero Knowledge Protocols and Small Systems” Department of Computer Science, Helsinki University of Technology (1995).

[14] T. Beth, "Efficient zero-knowledge identification scheme for smart cards." Advances in Cryptology—EUROCRYPT’88. Springer Berlin Heidelberg, 1988.

[15] C.P. Schnorr, "Efficient signature generation by smart cards." Journal of cryptology 4.3 (1991): 161-174.

[16] E. Holohan, M. Schukat, “Authentication using Virtual Certificate Authorities - A new Security Paradigm for Wireless Sensor Networks”, The 9th IEEE International Symposium on Network Computing and Applications (IEEE NCA10), July 2010

[17] P.C. Kocher, “Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems”, CRYPTO '96 Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology