From Dining Cryptographers to dining things: Unobservable communication in the IoT

Conference Paper · June 2017

DOI: 10.1109/CAMAD.2017.8031529

2 authors:

Johannes Bauer
Universität Passau
5 PUBLICATIONS **56** CITATIONS
SEE PROFILE

Ralf C. Staudemeyer
University of Applied Sciences Schmalkalden
34 PUBLICATIONS **414** CITATIONS
SEE PROFILE

From Dining Cryptographers to Dining Things: Unobservable Communication in the IoT

Johannes Bauer
University Passau
94032 Passau, Germany
bauerjo@fim.uni-passau.de

Ralf C. Staudemeyer
University Passau
94032 Passau, Germany
ralf.staudemeyer@uni-passau.de

ABSTRACT

The increasing adoption of the Internet of Things adds new risks for the privacy of everybody. Current privacy enhancing technologies are not sufficient for the new threats. This papers explains one of the most powerful protocols, the DC-net. The DC-net protocol, while more than 30 years old, was never used in real-life due to its overhead and sensitivity against disrupters. However, many improvements have been contributed to the original protocol, which will be summarized in this paper. The main contribution of this work is the implementation of a proof-of-concept DC-net suited for constrained microcontroller-based devices. The challenges faced and the optimizations done throughout the implementation will be presented.

Keywords

Security, Unobservable communication, Privacy, Confidentiality, Unlinkability, Internet-of-Things

1. INTRODUCTION

Privacy is often simply reduced to confidentiality. However, an attacker who is able to mine big quantities of data like network traffic can extract useful information and communication patterns that may reveal sensitive information of users. Powerful Big Data technologies combined with the ubiquity of sensors in the Internet of Things can threaten the privacy of everyone. Simple encryption doesn't solve these problems, as they can't hide the linkability between sender and receiver. Hiding the information who talked to whom or which thing communicated with which things yields into the topic unobservable communication. While there are currently technologies in use like the TOR network [1] which is based on Mixing networks, that try to enable its users anonymous and unobservable communication, a number of attacks [2,12] have shown, that these can't hold their promises. It's a good idea to look out for other protocols which basic security offerings are much stronger than current technologies, like the Dining Cryptographers net.

David Chaum presented in [4] a powerful communication protocol that provides unconditional secure unobservable communication: the Dining Cryptographers Net (DC-net). While the anonymity level guaranteed by a DC-net is much more higher than the anonymity of networks like TOR, DC-nets haven't been adopted in reality due to its practical problems like scalability and sensitivity for disrupters.

[1] `https://www.torproject.org`, last visited 27.02.2017

This paper examins the general applicability of a DC-net in a typical Internet of Things environment.
Section 2 explains the general concept and the initial idea behind DC-nets, followed by a detailed overview of the followed work. The sections ends with the presentation of two real DC-net implementations. Section 3 then introduces the IoT specific needs, problems and challenges in terms of unobservable communication. After a short introduction of the state-of-art in terms of privacy and anonymity preserving techniques, the approaches presented in Section 2 are reviewed for their applicability for a IoT DC-net. A custom DC-net protocol, suited to the use-case presented, will be developed, discussed and implemented. Based on this prototype, the insights gained through the implementation will be explained and the challenges will be discussed. The section ends with an evaluation of this proof-of-concept implementation

2. UNOBSERVABLE COMMUNICATION

This section now presents the Dining Cryptographers Net (DC-net). Starting with the original idea by David Chaum in 1988 [4], various improvements to the original DC-net have been made to fix vulnerabilities and improve the overhead. While DC-nets haven't yet been adopted in reality, however, in 2003 a first implementation was presented. Later, another implementation appeared - the first one that is open sourced and addresses many problems of DC-nets in practice. Those implementations will be presented and are serving as a basis for the IoT based DC-net presented in the next section.

Chaum introduced its idea with a little story: three cryptographers were having dinner in a restaurant and finally someone paid the dinner - either one of the cryptographers or their employer, the NSA. Now, they want to find out who paid the bill, either one of them or the NSA. In case one of the cryptographers paid, however, they don't want to reveal who exactly. They think about the following protocol in order to achieve this.

The cryptographers are sitting on a round table (depicted in figure 1) and each one of them is flipping an unbiased - that is heads or tail are equally likely - coin secretly and show the outcome to his right neighbor. We refer to heads with "1" and to tails with "0". Now, each cryptographer combines the coin toss on his left and the one on the right (the coin the tossed) with the XOR operation. A cryptographer who was

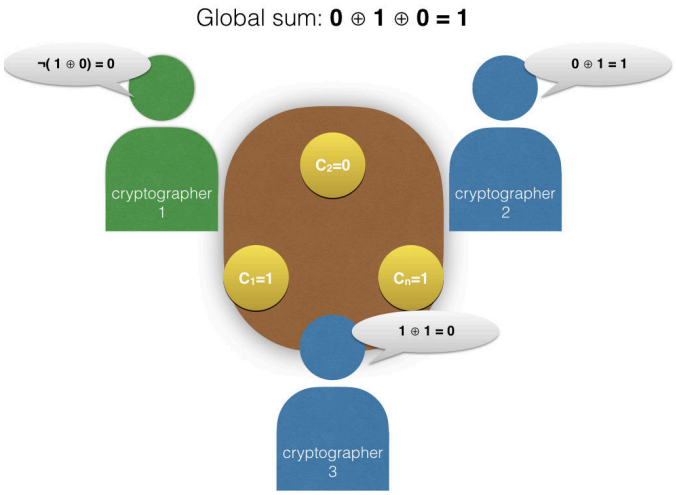

Figure 1: One of the cryptographers has paid (green). Here: $0 \oplus 0 \oplus 1 = 1$ which shows that one of the cryptographers paid and not the NSA.

not paying the bill, writes the result on his napkin in front of him, whereas a cryptographer who paid the bill, writes the negation of the result. The cryptographers see the local results from the others, and combine these results again with XOR, which we are referring as the global sum.

If, and only if, one of the cryptographers paid and is thus inverting the result of his local XOR operation, the global sum will be 1, since there exists now an odd number of ones. Assuming, that the coins were unbiased and no pair of cryptographers is colluding [2] this protocol is unconditionally secure. In case the NSA paid, the global sum is 0 and there is no problem with anonymity. In case one of the cryptographers was paying, it is equally likely for one cryptographer who wants to investigate the payer, that his left or right neighbor was paying. The XOR operation plays an important role here (later it will explained, that also other mathematical structures can be used): since every coin toss $C_n, n \in 1, 2, 3$ exists twice in the global sum. Due to the commutativity and associativity of the XOR operation, the individual coin tosses

$$\begin{aligned} &Local_{crypto1} \oplus Local_{crypto2} \oplus Local_{crypto3} \\ &= (C_1 \oplus C_2) \oplus (C_2 \oplus C_3) \oplus (C_3 \oplus C_1) \\ &= (C_1 \oplus C_1) \oplus (C_2 \oplus C_2) \oplus (C_3 \oplus C_3) \\ &= 0 \oplus 0 \oplus 0 = 0 \end{aligned} \quad (1)$$

cancel out in the global sum yielding always 0 as the global result. Only if one cryptographer who wants to send a (1-bit) message inverses his local result, the global sum gets 1.

Coin tossing and sharing the result with the neighbors practically means exchanging secrets through a secure channel. For transmitting longer messages, many keys can be tossed beforehand and the outcomes can be shared. If the participants of a DC-net share secrets through a unconditional secure channel, the DC-net provides unconditionally security. If key exchange is done through a public-key cryptography system, the security of a DC-net reduces to the degree of computationally security (relying on the security of public-key cryptography). The anonymity of a single node in a DC-net is expressed as the anonymity set for this node, that is the set of honest nodes which the node is sharing secrets. Chaum proposes a ring as a possible network topology, that has compared to a traditional ring where messages often only travel have through the ring before the recipient gets them, a fourfold increase in bandwidth. To reduce bandwidth, not the whole message should be redirected by each ring node, instead, every node can build the XOR of his message and keys with the message he got from the previous node. This way, one global broadcast round has to travel twice through the ring in order to received completely by the participants: in the first round every participant forwards the incoming message together with his local output and in the second round, the global sum is broadcasted to all members. Another problem Chaum points out, are disrupters: malicious nodes, or nodes with faulty behavior can disturb the whole communication by sending for example random data and thus making the global message unreadable. DC-net provides untraceability - also for those attacker - which makes it hard to detect them. Chaum proposes a trap mechanism, in which honest sender reverse a sending slot but instead of sending actual messages, they place traps in it by sending a random message with a secret key. If the attacker tried to disrupt the communication in this round by sending in the reserved slot of another node, the honest node that placed the trap detects this and signals this to the other participants together the with his secret for this round and his decrypted message. Waidner and Pfitzmann generalize and improves the concept of DC-nets [20, 21]. They generalize the concept of sender untraceability of superposed sending. The set of participants $P = P_1, ..., P_n$ are connected through a graph G. Instead of relying on the XOR operation, any abelian finite group $(F, \oplus)$ can be chosen, which is called "alphabet". Participants share common secrets, that is, participant P_i and P_j share the secret key $K_{i,j} = K_{j,i}$. G denotes the key sharing graph, containing all nodes who shared secret keys with each other. Then each participant P_i chooses a message character M_i: 0 or the actual message if it wants to send (and owns the current slot). Then, it combines the message M_i together with all keys sharing with each other in this round. The *sign* function is used to create the vanishing effect (like $x \oplus x = 0$ with the XOR operation).

$$O_i = M_i \oplus \sum_{P_i, P_j \in G} sign(i - j) \cdot K_{i,j} \quad (2)$$

This local output of one node is then broadcasted to all other nodes, and finally each node can compute the global result S of this messaging round:

[2] There is never anonymity if all participants are colluding against one single victim.

$$S := \sum_{j=1}^{n} O_j \tag{3}$$

Finally, all keys in occur twice in the sum with different signs leading to vanishing of all keys and empty messages and only the actual message of the sender who sends a message $M_i \neq 0$ is visible to all participants while preserving the anonymity of the actual sender.

Waidner also generalizes the reservation technique describe by Chaum to additive groups of integers modulo m. Each participant who wants to send randomly choose a position in a reservation vector with r slots, and puts 1 into this position. After broadcasting those vectors, each participant can sum the positions up: 0 in a position means no-one reserved this slot, 1 means one participant successfully reserved this slot without collisions and $n > 1$ means more than one participant wants to send in this slot. All slots with collisions are skipped and only successfully reserved slots are used by the specific participant.

Bos and den Boer presented another reservation approach based on superposed sending of a polynomial function [3]. As pointed out in [21] Chaum's construction of DC-net relies on the "reliable broadcast assumption": A reliable broadcast means that a broadcast message sent by a sender is received by every honest receiver and second is received without changes.

In [10] Golle and Juels propose a new approach to detect disrupters in a DC-net with less overhead than the original technique used by Chaum [4]. Instead on relying on additional (costly) trap rounds where no actual communication happens, Golle and Juels instead propose a technique that is based on cryptographic proofs. Furthermore, their approach allows to recover from a corrupted message - occurred either through jamming of a disrupter or network faults - with only one additional broadcast round. Their improved DC-net protocol still maintains the property of non-interactivity, that is, once two nodes have shared secret keys, there is no more direct communication needed between them.

2.1 Implementations

2.1.1 The first implementation: Herbivore

Herbivore [9] is the first implementation of a DC-net based network protocol in practice. The goal of Herbivore is to build a DC-net based network over the Internet and tries to mitigate the scalability issues and the impact of disrupters on DC-nets. The main idea is to partition the network into smaller sub-groups in the size of tens of nodes which perform the "Round Protocol" which is based on the DC-net protocol. The "Global Topology Control" protocol manages the connection between the cliques and the entry of new nodes to the networks.

Global Topology Control - Partitioning the network into subgroups.

New nodes joining the Herbivore network contact any of the participants of the Herbivore network. It is important that a newly arriving node cannot freely choose the subgroup to join: attackers could try to compromise one specific clique and try to deanonymize the nodes in it by colluding. Instead, the authors propose a challenge-based protocol: a new node creates a key pair $(K_{public}, K_{private})$ and then randomly generates vectors y such that $y \neq K_{public}$ and the lower m_k bits of $f(K_{public})$ are equal to those of f(y) for a given one-way function f. The value $g(K_{public}, y)$ with another one-way function g is then used to contact the node, whose node key is closed numerically.

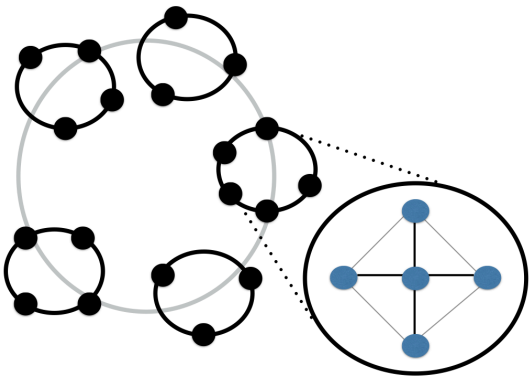

Figure 2: Network topology of Herbivore. On the left the global chord that connects the subgroups ("cliques"). On the right a single subgroup, communcation rounds are organized in a star (the grey lines indicate the key graph - each node shares keys with each others). Adapted from [9].

After contacting that specific clique, the clique's nodes can check that $f(K_{public})$, $f(y)$ and check that the lowest m_k bits match. They check that the value $g(K_public, y)$ matches to their group and was not used before by another node. With this method, an attacker cannot freely choose a clique key $g(K_{public}, y)$ as it would need to compute the inverse of the one-way function g. By increasing the size of m_k it is possible to increase the costs for entering the network and therefore the costs for attackers who are trying to guess a right key by computing many keys beforehand. The authors further increase the security by requiring nodes to choose y with the lower bits encoded with the current date as a time stamp. Attackers then lose the opportunity to generate a large key dictionary (rainbow table) beforehand. The cliques are globally organized in a Chord [18]. If single subgroups get to big, they get split into two new cliques. To maintain anonymity when the size of a clique falls under a certain threshold, the clique gets destroyed and the nodes have to join again the network as described above.

Communication within a subgroup.

After successfully joining a clique, a new node has to establish common secrets with every node member in the clique. The communication is then divided into three phases: reservation, transmission and exit.

In the reservation phase, transmission slots are assigned to nodes that want to communicate their messages. This is done through a reservation vector which is zero in every

component except one position that gets randomly picked by the node. Each node prepares such a reservation vector and broadcasts it to the other members. Reservations can collide and can be detected in the reservation phase if an even number of nodes trying to use the same slot, or in case of an odd number of nodes are trying to reserve the same slot, the collision will show up in the transmission phase.

In the transmission phase, every node transmitting sets the bit to 1 in its transmission slot. Every node is sending its vector and receiving the vector from others. Hence, a node sending in slot i can detect if also other malicious nodes are trying to send in its slot. It therefore tries to retransmit its messages in another round and after a fixed number of unsuccessful retires, it rejoins the Herbivore at a different location.

The exit phase tries to make sure that long-running network transactions can be processed and received by clients who want to leave through a signaling mechanism. This ensures that attackers can't perform intersection attacks on nodes that have long-running transactions in place and change the network. The key graph in a Herbivore subgroup is a fully connected graph, that is every node shares secrets with every other node. Message exchanges, however, are done in a star topology, where every node send its keystreams and message, combined with XOR, to the center of the star which directs those messages to all other nodes. In order to get equal load on the participating nodes, the center of the star is changing after every round. This ensures that the latency of transmissions is independent of the size and the bandwidth of each node, however it reduces basically to the slowest node. With the design of Herbivore, the authors consider many possible attacks:

Topology Attacks: A Chord as the global topology and random, challenge-based entry protocol tries to ensure that attacker can't explicit compromise a specific clique. For cliques of the size 128 and 90% of the total network compromised, there's still only a chance of $0.9^{127} \sim 1.5 \times 10^{-6}$ for taking over the whole network. However, as noted in [8] an attacker could try to isolate single victims by disturbing groups which it can't fully compromise and "behave well" in groups in which it wants to isolate the victim. This could increase the probability for the victim to change into the group the attacker wants it to be over the long term.

Sybil Attacks: Attackers could try to compromise and disturb the network by entering under different identities to the network. Herbivore address this by limiting the rate at which new nodes can enter the network with the challenges-based protocol which increases the cost to join the network.

Intersection Attacks: Intersection Attacks on long running transmissions are considered in the exit phase. Also without malicious attackers, nodes can disappear due to various failures. With a clique size of 128 nodes and a mean node lifetime of 10hours, a node can anonymously transmit for 1 hour and be sure that no more than 25 nodes will fail statistically. Evaluations on Herbivore with clique sizes ranging from 10 to 40 show that the bandwidth utilization is linear with increasing clique sizes. Herbivore was the first approach of implementing the DC-net protocol in reality. However, to my knowledge, there is no further development on the implementation since the initial papers [9, 17] and no source code is available.

2.1.2 *Dissent*

Dissent [3] is another implementation first presented by Corrigan-Gibbs and Ford in 2010 in [5] and later improved in [6, 22]. Dissent is built upon Verifiable Shuffles and the DC-net protocol. However, instead of the complete decentral approach of the DC-net described in [4], Dissent is uses a client-server architecture, where each client node only shares keys with the servers, but not with other clients (see figure 3. The communication in Dissident consists of the following phases: Each client creates a new secret pseudonym key k and sends it encrypted to a server, which creates a secret permutation (that is, no-one else can see the positions of other clients) of all keys received by the clients, decrypts the keys an assigns a (secret) slot to every node. With a known scheduling function, each client can determine the slots assigned to it (depicted in figure 4). Each client creates a message with zeroes in all position except the slots she was assigned. In

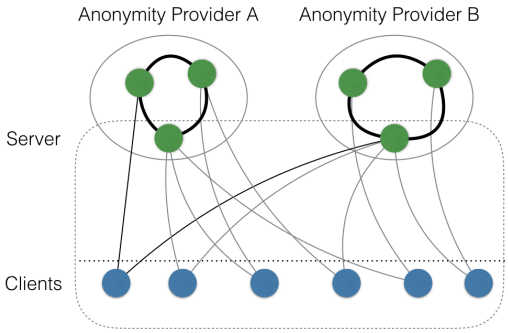

Figure 3: The client-server architecture of Dissent. Clients communicate directly and share keys only with servers. Adapted from [22].

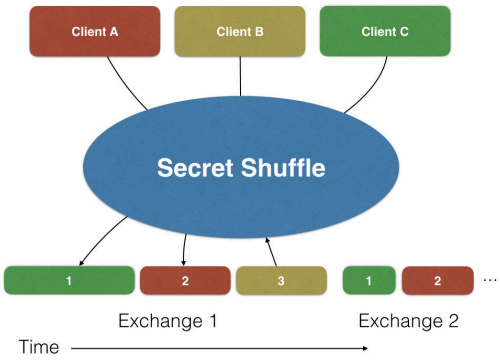

Figure 4: Dissent's scheduling mechanism, realized with a verifiable shuffle. Adapted from [22].

[3] http://dedis.cs.yale.edu/dissent/

those slots she can put its message. Afterwards, the client creates pseudo-random strings from the secret keys the client shares with every server and combines those strings with XOR. Each client sends his output to at least one server. Afterwards, each server collects the messages from the last step. Contrary to the original DC-net protocol, there's a fixed deadline - clients who send their messages to late or don't send anything at all, are left out in this round. The server synchronizes the received messages from each other (redundant messages from the same client can be recognized and trimmed through the pseudonym keys). The server now continue with the protocol only if the number of participating clients in this round are above a certain threshold (to not threaten the anonymity of the clients sending in this round). Each clients receives the final output of this round, which is signed by every server. After verifying the signatures, the client can extract the message from this round. Each node in Dissent shares secrets with each server. Malicious server can therefore not partition the key share graph and thus the anonymity set (as described in [4]). Therefore, if there is at least one honest server, the anonymity of a client can't be compromised easily. The client doesn't even have to know which one of the servers need to be the honest server, which they call the "anytrust-model".

A significant difference to the communication in a Herbivore subgroup is, that nodes don't communicate directly to each other. In Herbivore, each node of subgroup alternating forms the center of star and is responsible to redirect every message of the other nodes. Hence, the latency and bandwidth of a single node can be a bottle-neck and reduce the overall performance of the local group. The decision to choose a fixed receiving window in Dissident tackles this problem, by simply ignoring clients that are too late. In the traditional DC-net, this would not be possible, as the global result relies that every participant in this round committed its message. The possibility, that attackers could misuse this behaviour to DoS clients and try to isolate single clients currently sending, is circumvented by adding a threshold for a minimum number of participating clients in single rounds.

Look at the Source Code of Dissent.
Fortunately, the authors made their implementation of Dissent open source [4]. together with a good documentation [5] of the specific components. Dissent is implemented in the `C++` language with the `QT`-Framework. In practice, normal TCP traffic can be tunneled through the SOCKS server provided by Dissent and therefore normal application level protocol can transparently use the DC-net implementation.

3. INTERNET OF THINGS

The last section presented the DC-net protocol with manifold improvements and two implementations. This section starts with a short overview of the state of the art of privacy enhancing technologies in the Internet of Things. Internet of Things environments differ significantly from traditional network environments: Wireless Sensor Networks containing constrained sensor nodes have different characteristics compared to "traditional" network environments like PCs connected through the Internet. These differences will be explained and the specific challenges for a DC-net in an IoT environment will be outlined. Based on the insights gained from the last section, a DC-net protocol and communication network suited for a WSN will be developed and implemented as a proof-of-concept. The resulting implementation will be evaluated and the general applicability of DC-nets on constrained devices in an IoT scenario will be discussed.

3.1 Constrained environments

The Internet of Things is becoming reality: "smart" sensors and actors are all around us in our homes, public buildings, cities, cars, factories etc. Those "things" are often lower power and low cost hardware, embedded - often invisibly - in sensor network environments and communicating wirelessly. They differ from traditional PCs and laptops in several aspects: equipped with energy-efficient, but less powerful microcontrollers, little memory capacity and they are running on battery or low-power energy supplies like solar panels. Encryption, Origin Authentication and other security mechanisms come at a cost - costs that often have little impact on powerful hardware equipped with modern CPUs and gigabytes of RAM. However, for constrained devices, this overhead can have a dramatic impact on the overall performance and battery life and may at worst lead to security techniques getting neglected instead.

3.2 IoT Scenario

In this paper, a Smart Home scenario is used for embedding the DC-net implementation. The characteristics of such a scenario - compared to hosts communicating over the Internet - are constrained, wireless devices installed in a house. Those devices can be sensors, like a door or window sensor, a temperature sensor or a smoke sensor. The data collected from those sensor is then transmitted wirelessly, processed and decisions are made upon it. For example, the door control sensor registers a new person entering the house, informs the other nodes and the light actor switches on the lights, or the heating turns on. While those scenario seem harmless at the first point, interesting information can be gathered through long term eavesdropping: the habits of the individual residents, the time periods where they are at home or not and so on. Encryption protects the content of the messages exchanged, but it can't hide the sheer fact, that communication has happened. Now, an attacker eavesdropping over a longer period can learn various patterns in communication and may be able to correlate certain events: e.g. if the light actors didn't get any messages in the morning, this may indicate, that no person is currently in the house. Or every time the door control sends a message can be correlated to residents leaving the house and so on. Message encryption alone can't hide the communication patterns and the meta-data. DC-nets however can hide meta-data of this kind: if nodes are continuously sending and receiving, the attacker doesn't learn anything from this "noise" and thus may not be able anymore to gain more knowledge.

3.3 IoT Stack

A typical IoT protocol stack may consist of a RESTful interface on top of the `CoAP` [16] protocol, which can be informal

[4]https://github.com/dedis/Dissent, last visited on 12/17/2016
[5]https://github.com/dedis/Dissent/blob/master/DESIGN, last visited on 12/17/2016

described as a compressed version of HTTP for constrained devices. `CoAP` is used on top of the unreliable but more lightweight (compared to `TCP`) `UDP` protocol. On the network layer, IPv6 is typically used with the `6LoWPAN` adaption layer, which adapts `IPv6` for the needs for constrained sensor environments (e.g. fragmentation on the network layer is avoided and carried through upper layers for less complexity for the sensing nodes) [11, 14]. Finally, on layer 1 and 2 for WSN, the `IEEE 802.15.4` low-power radio interface is used, which is an important wireless protocol used in the IoT.

In order to realize confidentiality, end-to-end encryption can be achieved by Datagram Transport Layer Security (`DTLS`) [15]. Strong origin authentication through digital signatures is feasible with tolerable overhead through Elliptic Curve Digital Signature Algorithms (`ECDSA`) [1, 13]. However, to my knowledge there are currently no specifications or implementations available that try to bring unobservable communication to wireless sensor networks.

3.4 Communication costs

Chaum [4] proposed a ring topology, where every node is receiving the message from its neighbor and in order to reduce bandwidth combines his local output directly to the message received. For one DC-net communication round, the message has to travel twice through the ring: the first time is used to get all local outputs from participating nodes (sending) and the second one for broadcasting the final, global message to every node (receiving).

In *Herbivore* [9] the local subgroups who are actually performing the DC-net protocol, are organized in a star topology. Every node is forming the center of the star in new rounds in an alternating way. Note, that while the communication is done in a star, every participant shares secrets with every other node, that is the key graph is a fully connected graph. The downside of this architecture in network with n nodes is that a node with low bandwidth and high latency acting as the star in a round has to receive $n-1$, combine those results (it doesn't have to, but this decreases the bandwidth requirements) and send then the message to $n-1$ nodes. In a WSN this leads to a high load that a single node has to process and could further increase the latency.

The topology in *Dissent* differs quite significantly: instead of a decentral architecture, *Dissent* uses a client-server architecture [5, 22]. Communication always happens among client-server and server-server, but never among two clients directly. Also the key graph differs from the full connected graph in the *Herbivore* net or in the ring topology: clients share only secrets with the servers, but not with other clients. This design of Dissent has a big advantage: it allows for one communication round to left out clients that were leaving the network or clients that are too "slow" (in the current communication round). Compared to Herbivore or a ring topology, where the slowest client determines the latency and bandwidth characteristics, Dissent increases the performance and scalability (the consequence for the anonymity through this practice are considered through thresholds, i.e. for a communication round, a minimum number clients have to participate, see Section 2).

Wireless networks, however, have one big advantage in terms of topology: broadcasts can be done "naturally" over the physical medium and therefore are "cheaper" in terms of communication overhead. Assuming, every node reaches all other nodes wirelessly, a broadcast message reaches all other nodes without further interaction like forwarding etc. required (such a fully connected network can be achieved through current radio technologies, at least in environments like a single house). The decoupling between the global message and the clients that *Dissent* provides, can be hardly realized in a pure decentral, fully connected network graph: if a single node doesn't broadcast his local message in a given round, the local output of his neighbors (that is all other nodes in the network) depends on it and therefore the global sum of this round can't be calculated.

The first aspect is important to determine which load one single, constrained node device has to cope with. The second aspect indicates the bandwidth and latency requirements that a low-power and lossy network like a wireless sensor network has to offer. Assuming a DC-net with n nodes, with a fully connected key graph, the calculations are done over n rounds of communication. In the ring topology, every node receives the temporary global sum from its predecessor, combines its output and forwards this message. After the temporary global sum has travelled once through the ring and every participant has committed his part, the global sum has again travel through the ring in order to get received by every node. The number of sending and receiving steps after n rounds for one node are thus $2n$ for each. In the star topology, for $(n-1)$ rounds, one node has only to send his local message and receive the final global sum. But in the round where it has to form the center, it has to receive $n-1$ and also send $n-1$ message which sums up to $2n-2$ sending and receiving steps for each. In the fully connected wireless network, every node has a constant sending and receiving rate in every rounds, yielding in n sending steps and $n \cdot (n-1) = n^2 - n$ receiving steps in total.

Assuming there is only one communication round happening in the ring, only one message travels through it. The load on the ring is thus constant, however, with increasing number of nodes, the number of hops is increasing linearly and so does the latency. In the star topology, every client sends his message to the star and the star sends the accumulated message back again yielding in $(n-1)^2$ total messages exchanged in one round and though quadratic costs. For the fully connected, wireless network, exactly n message are exchanged in total in one round.

Summarizing the costs for individual clients for n rounds, the star topology seems to win, however with the downside, that every single node has to cope with the round where it has to build the star center. Nodes in the fully connected network have a constant sending rate of one message per round independent from the network size, must however receive n messages. The receiving costs are thus increasing linearly with the size of clients. The total number of messages exchanged in the fully connected network and the ring are linear with the number, whereas the costs are quadratic in the star.

3.5 Attacker Model

The attacker model for this scenario is a global attacker, however, who is bound computationally, that is an attacker who can listen to every communication (this is very realistic in a locally limited, wireless scenario) and who can insert arbitrary messages (same here). However, Denial of Service attacks through radio jamming are not considered in this paper. Given an attacker with a powerful enough radio equipment, jamming the whole radio frequencies used by the sensor nodes, there is hardly any possibility to prevent this kind of attack. *Dissent* considers these types of attack through defining thresholds of a minimum number nodes who have to participate in a given round, otherwise the round is aborted by the servers (or the receiving windows is increased). Unfortunately, this is much more complicated to realize in a decentral DC-net, because each node participating in a given communication round doesn't know beforehand, how much nodes will (or can) participate. If an attacker therefore is able to simultaneously prevent all other nodes from sending - except the victim node - it is possible to isolate it by effectively reducing its anonymity set to 1.
Since a wireless sensor network like in a Smart Home is bound geographically to a specific location, special care has

3.6 DC-net architecture and communication protocol

In the following paragraphs, the steps forming the basic DC-net protocol implemented here will be presented.

Entry protocol.

As mentioned, there should be an entry burden for newly connecting nodes to reduce the risk of Sybil attacks [9].. For attacker nodes who are already in the DC-net this adds a penalty if they get detected and kicked out of the DC-net, since reconnecting (maybe with another identity) is more costly. This is especially important, because in many IoT/ WSN environments such as Smart Cities, attackers can't be easily removed physically (or even located). Each new node that enters the network has to establish secrets with all other node. This can be done through Diffie-Hellman key exchange. Diffie-Hellman through key exchange through elliptic curves (`ECDH`) can be done efficiently with the *MicroECC* library. The entry protocol is currently not implemented in the proof-of-concept implementation. In the prototype, there are currently only two *RE-Mote* nodes with hardcoded shared-secrets.

Communication.

In contrast to the original DC-net, two important optimizations for the message exchange were implemented. Instead of sending 1 Bit messages in every message round, messages with bigger payloads get exchanged. This allows it for the receiver to spend less effort on putting together the individual messages from the clients. A message round is a single round, where each participant commits the combination of its shared secrets and where one single client sends its "actual" communication payload. For sake of clarity, the term "active sending" for a client means the client actual sends information it wants to communicate where "passive sending" refers to taking part in a message round (i.e. committing the combination of the secrets) but setting the own payload to 0.

The prototype doesn't utilize a TCP nor UDP stack. Instead it builds upon the *RIME* stack which is a network stack of Contiki with very low overhead. *RIME* itself offers different layers with certain services, but the prototype uses the only the "anonymous broadcast" layer which offers only the service of sending broadcasts. This *RIME* layer builds directly upon the *802.15.4* which allows only to send at most 127 Bytes. Therefore it increases the efficiency in terms of energy consumption and overhead to utilize the packets as good as possible. Instead of transporting only one message round per packet (i.e. 1 Bit) multiple message rounds are carried in only one broadcast packet in the "sending vector". The prototype currently uses for message rounds per packet, but this is a rather arbitrary number that needs optimizations in the future.

In every message round, every participant commits his local message: the combination of its shared secrets together with its payload. But instead of the initial DC-net protocol with single bits and the *XOR* operation, an abelic group g modulo a prime p is used (see section, 2, improvements by Waidner and Pfitzmann [20, 21]). For this first proof-of-concept I decided to choose 8 Byte for the message size. The biggest prime that can be expressed in 8 Bytes is 18446744073709551557. Rounded down to 7 Bytes, every message payload with 7 Bytes can be encoded as a number n with $n \in G(\mathbb{N}/18446744073709551557\mathbb{N}, +)$. An abelic group modulo a prime was chosen as it allows to easily calculate the inverse of a number n: $n^{-1} = p - n$. Now, considering two participants A and B with the shared secret $s \in G(\mathrm{N}/p\mathrm{N}, +)$ and A active sending the number $n = 42 \in G$, the global sum is calculated as follows:

$$\begin{aligned} (Local_A + Local_B) &\mod p \\ = ((s+n) + (s^{-1}+) &\mod p \\ = (s + s^{-1} + 0 + n) &\mod p \\ = n & \end{aligned} \tag{4}$$

Reservation Phase.

The reservation is done with the reservation map technique described in [20], based on superposed sending (see section 2) and with the same abelic group described above. Each client who wants to send messages, randomly chooses positions in the reservation vector and indicates this by setting the specific positions to 1. All other positions are set to 0. Nodes that don't want to send messages, send an empty reservation vector with zeros in all positions. Afterwards, every node applies its shared secrets to all slots. Each client broadcasts this vector through the network and receives the reservation vectors from all other clients. After every client has received all reservation vectors, it sums the vectors component wise together. Afterwards, positions in the vector with value 0 mean: no reservation, positions with value 1 mean: successful reservation of exactly one node and positions with value > 1 indicate a collision. The reservation vector consists of a single magic byte at the beginning indicating that this is a reservation vector and four 8 Byte slots containing the reservation number.

Transmission Phase.
In the transmission phase each, every node prepares its vectors: 0s and the applied shared secrets in the passive sending slots and n and the applied shared secrets for the active sending slot (if any slot was reserved successfully) with n being the payload encoded as a group number. The transmission vector is of equal size as the reservation vector with a different magic byte at the beginning. The nodes send their transmission vector and receive all vectors from the others, combine the vectors and are finally able to see the payloads in each slot for each message round.

If a node wants to leave the network, it is important to have an signaling mechanisms such the other nodes are informed and the computation of the global sum doesn't rely on the local result from the leaving node. Such an "Exit phase" is currently not implemented in the prototype.

3.7 Implementation details

The implementation is done on a Zolertia RE-Mote [23], a wireless sensor device equipped with the CC2538, a ARM Cortex M3 SOC from Texas Instruments with 32MHz, 512KB flash and 32KB RAM [19]. The CC2538 chip further incorporates a hardware acceleration unit for SHA2 hashing, AES encryption and Elliptic Curve Cryptography. The RE-Mote has two radio interfaces: the CC2538 is capable of running on the 2.4GHz IEEE 802.15.4 band and a additional TI CC1200 which is running on 868 or 915MHz for long-range distances [6]
The Contiki operating system (OS) [7] is used as the fundament for this implementation. Contiki is open source, written in platform-independent `C` and in active development. Contiki is providing a rich set of standard protocols (CoAP, UDP, TCP, IPv6 etc.) while still being as lightweight as possible and therefor an ideal basis for Internet of Things applications.
The pseudo-random number generator that is used for generating the shared secrets for each message round is currently a hardware-accelerated implementation of the *SHA256* function where only the first 7 Bytes are extracted (to fit into the abelic group).

3.8 Evaluation

The main contribution of this paper is to show that building a basic DC-net in an IoT environment is actual possible. However, for a broad evaluation, the remaining pieces such as the key exchange, the entry phase and the exit phase have to be implemented and the implementation has to be tested on more than two devices.

The evaluation in this section therefore focuses on the overhead of the implementation for the individual /textitRE-Mote devices. Whilst the implementation is not yet optimized on efficiency, a look on the code size overhead and the required timings gives an indicator for costs of an DC-net on constrained devices.

The size of the implementation is done through measuring the difference between an plain Contiki firmware image and the DC-net implementation plus the Contiki image. The size was measured with the `size` tool after using the `strip` tool, both provided by the gcc-arm toolchain. The results are listed in table 1.

Table 1: Code size overhead in Byte of the prototype implementation. Measured with the `size` tool provided by the gcc-arm toolchain.

firmware	text	data	bss	total
Contiki	19993	228	5457	25678
Contiki + DC-net	23296	252	5675	29223
overhead	3303	24	218	3545

The network communication was analyzed with the *sensniff* tool provided by Contiki, which was flashed on a third *RE-Mote*. The sniffed packets got serialized read in by the *sensniff* [7] Python script and then encoded in a `.pcpap` file. This method allowed to directly sniff the `802.15.4` packets sent through the air, rather than using a *802.15.4* border router.

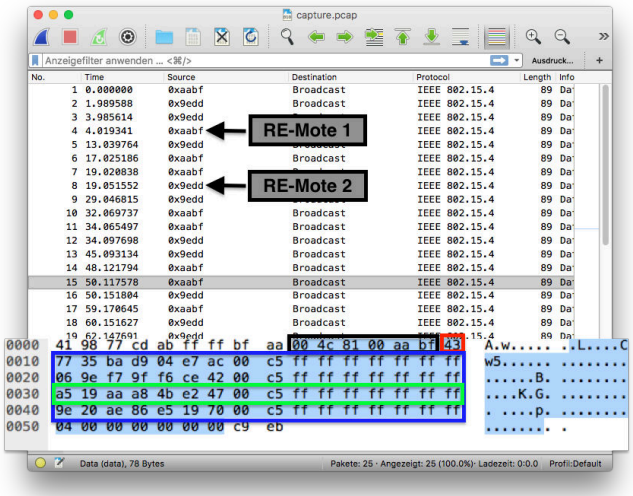

Figure 5: Analyzing the captured traffic between two *RE-Motes* with Wireshark. Red is the magic byte (here: reservation), blue are the four vectors, green is the vector in the third slot).

A sample capture is depicted in image 5 [8].

4. CONCLUSION AND FUTURE WORK

In this paper, the application of the DC-net protocol on constrained devices in a Internet of Things environment was examined. DC-net is a powerful communication protocol that allows for unobservable communication in decentral network. There is a need for such privacy enhancing technologies, especially in the Internet of Things, where a plethora

[6] maximum range up to 20km, see `https://github.com/Zolertia/Resources/wiki/RE-Mote` (last visited 12/17/2016)

[7] `https://github.com/g-oikonomou/sensniff.git`

[8] Note that the group number's modulus are currently carried out with the packet. Since they are all the same, they will be refactored out in a new version.

of sensors are collecting data all the time and everywhere. Encryption alone doesn't prevent attackers from collecting meta-data and communication patterns that can reveal privacy related informations. Current technologies like TOR offer a much lower level of anonymity, compared to the DC-net and are not intended for the application on low-level hardware like microcontrollers. Hence, this paper brings the DC-net protocol onto a constrained, microcontroller-based device, the *RE-Mote*. Together with some of the improvements that have been made on the original DC-net since its presentation, a proof-of-concept implementation was developed and tested. Whilst this proof-of-concept implementation is not a ready-for-production solution, it shows, that various problems from working in a constrained environment can be mitigated and some characteristics, like the cheap, physical broadcast over the air can be exploited. There is a lot of room for future work, such as the optimization of the different communication parameters, like slot size and message size, the timing of the broadcasts and the implementation of a reliable, but still lightweight approach for detecting disrupters and attackers. Privacy enhancing technologies like the DC-net are worth to be considered and definitely needed for the Internet of Things and the implied dangers for the privacy of everyone.

5. REFERENCES

[1] J. Bauer, R. C. Staudemeyer, H. C. Pöhls, and A. Fragkiadakis. *ECDSA on Things: IoT Integrity Protection in Practise*, pages 3–17. Springer International Publishing, Cham, 2016.

[2] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker. Low-resource routing attacks against tor. In *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society*, WPES '07, pages 11–20, New York, NY, USA, 2007. ACM.

[3] J. Bos and B. den Boer. *Detection of Disrupters in the DC Protocol*, pages 320–327. Springer Berlin Heidelberg, Berlin, Heidelberg, 1990.

[4] D. L. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[5] H. Corrigan-Gibbs and B. Ford. Dissent: Accountable anonymous group messaging. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 340–350, New York, NY, USA, 2010. ACM.

[6] H. Corrigan-Gibbs, D. I. Wolinsky, and B. Ford. Proactively accountable anonymous messaging in verdict. In *Proceedings of the 22Nd USENIX Conference on Security*, SEC'13, pages 147–162, Berkeley, CA, USA, 2013. USENIX Association.

[7] A. Dunkels, B. Grönvall, and T. Voigt. Contiki – a lightweight and flexible operating system for tiny networked sensors. In *29th ann. IEEE int. conf. on Local Computer Networks (LCN'04)*, pages 455–462, 2004.

[8] J. Feigenbaum and B. Ford. Seeking anonymity in an internet panopticon. *Commun. ACM*, 58(10):58–69, Sept. 2015.

[9] S. Goel, M. Robson, M. Polte, and E. Sirer. Herbivore: A scalable and efficient protocol for anonymous communication. Technical report, Cornell University Computing and Information Science, 2003.

[10] P. Golle and A. Juels. Dining Cryptographers revisited. In *Proc. of Advances in Cryptology (EUROCRYPT '04)*, volume 2729, pages 456–473, 2004.

[11] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. RFC4944 – Transmission of IPv6 Packets over IEEE 802.15.4 networks. {RFC} 4944, sep 2007.

[12] S. J. Murdoch and G. Danezis. Low-cost traffic analysis of tor. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, SP '05, pages 183–195, Washington, DC, USA, 2005. IEEE Computer Society.

[13] M. Mössinger, B. Petschkuhn, J. Bauer, R. C. Staudemeyer, M. Wójcik, and H. C. Pöhls. Towards quantifying the cost of a secure iot: Overhead and energy consumption of ecc signatures on an arm-based device. In *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–6, June 2016.

[14] J. Olsson. 6LoWPAN demystified. Texas Instruments, 2014.

[15] E. Rescorla and N. Modadugu. Datagram transport layer security version 1.2. RFC 6347, RFC Editor, January 2012. `http://www.rfc-editor.org/rfc/rfc6347.txt`.

[16] Z. Shelby, K. Hartke, and C. Bormann. RFC7252 – The Constrained Application Protocol (CoAP). {RFC} 7252, jun 2014.

[17] E. G. Sirer, S. Goel, M. Robson, and D. Engin. Eluding carnivores: File sharing with strong anonymity. In *Proceedings of the 11th workshop on ACM SIGOPS European workshop: beyond the PC - EW11*, page 19, 2004.

[18] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *SIGCOMM Comput. Commun. Rev.*, 31(4):149–160, Aug. 2001.

[19] Texas Instruments. CC2538 – A Powerful System-On-Chip for 2.4-GHz IEEE 802.15.4 and ZigBee Applications, 2016.

[20] M. Waidner and B. Pfitzmann. The dining cryptographers in the disco: Unconditional sender and recipient untraceability with computationally secure serviceability. *Proc. of the Workshop on the Theory and Application of Cryptographic Techniques on Advances in Cryptology (EUROCRYPT '89)*, 89:690, 1990.

[21] M. Waidner and J. Vandewalle. *Unconditional Sender and Recipient Untraceability in Spite of Active Attacks*, pages 302–319. Springer Berlin Heidelberg, Berlin, Heidelberg, 1990.

[22] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson. Dissent in numbers: Making strong anonymity scale. In *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation*, OSDI'12, pages 179–192, Berkeley, CA, USA, 2012. USENIX Association.

[23] Zolertia. Re-MOTE, 2015.