Towards a More Secure, Private Smart Home By Eliminating the Central Hub

Anna Róża Krzyżańska
Columbia University
New York, NY, USA
ark2219@columbia.edu

David Inyangson
Johns Hopkins University
Baltimore, MD, USA
dinyang1@jhu.edu

Tushar M. Jois
City College of New York
New York, NY, USA
tjois@ccny.cuny.edu

***Abstract*—As smart homes grow in popularity, the challenge of preserving security and privacy while managing network communications between the IoT devices involved is ever present. While current solutions include cloud controlled networks, which ultimately pose privacy risks, and physical smart home hubs, which introduce a single point of failure, we propose another option: a virtual smart home hub. In this paper we lay the groundwork for a system through which users can describe a smart home configuration with device inputs passed into a secure multiparty computation, creating a means for automation without a centralized hub.**

***Index Terms*—secure multiparty computation, Internet of Things, smart homes**

I. Introduction

Internet of Things (IoT) devices have transcended their novelty to become household staples for ordinary people [1]. At its core, the purpose of an IoT device is to connect to a network and exchange information with other devices. When there are multiple such devices with dedicated functionality interacting within a home, usually in such a way that they change states automatically and can be remotely controlled, we consider it a "smart home" [2]. The modern smart home typically has two ways of creating a network of connections between IoT devices: either by using a dedicated smart home hub, or by delegating smart home tasks to the cloud [3].

A cloud-controlled network presents a variety of security and privacy concerns. It is possible for an IoT cloud management system to execute commands without checking whether the sender has permission to make that request and likewise execute state transitions that may not be allowed [4]. Both of these aspects make the network vulnerable to remote attacks. Aside from this, there are inherent privacy concerns with using a cloud service provider. By entrusting a third party with their data, users rely on the service provider to both control who has access to it and manage their data with proper security measures. While there are laws put in place to regulate cloud service providers, data in the cloud is often stored in various physical locations, making it difficult to monitor if these policies are upheld [5]. Even more, IoT networks solely based in the cloud are not always practically feasible. A key trait of smart homes is that the devices have an internal network through which they can interact with one another [2]. However, it is possible for certain IoT devices to not have Wi-Fi access or cloud connecting capabilities [3], [4] so a cloud controlled IoT network may still require a physical hub.

978-1-6654-7345-3/22/$31.00 ©2024 IEEE

In many ways, having an IoT network managed by a physical smart home hub mitigates these risks. A physical smart home hub typically relies on protocols that do not require Wi-Fi, like Zigbee and Z-Wave, to manage an IoT network. This alone makes the traffic in such a network less prone to security risks as it requires an attacker to be in close proximity to the home [6]. Such networks are still vulnerable to attacks [7], [8], but in eliminating the need for cloud connectivity to automate device operations, the security risks of such a system are comparatively reduced. Still, having a physical hub be solely responsible for controlling an IoT network introduces new vulnerabilities: the physical hub becomes a single point of failure. If the hub is corrupted, either accidentally or maliciously, the entire IoT network is immediately at risk.

Existing literature on smart home networks often either highlights approaches for addressing security risks in current physical hub and cloud based IoT networks [3], [4], [6], [9] or else suggests ways in which the functional design of IoT networks can be improved [10], [11]. In this paper, however, instead of iterating on previously proposed network setups, we present a novel framework for a network between IoT devices that does not rely on a centralized management system. Our approach is centered on the premise that, through secure multiparty computation, each IoT device can directly communicate with other devices it depends on in the network, only accessing the data these other devices transmit in an encoded form. Thus, we present a system through which inter-device communication and traditional if-this-then-that automation can occur without a hub.

II. Background

A. Smart Homes

The modern smart home is built on a network of individual IoT devices that physically interact with the home. This network can include anything from smart light bulbs to security cameras, but ultimately also extends to other devices that join the household network including smart phones and routers. While perhaps not as essential to the underlying functionality of a smart home, one key smart home feature is the ability to control devices in the home remotely, with mobile apps

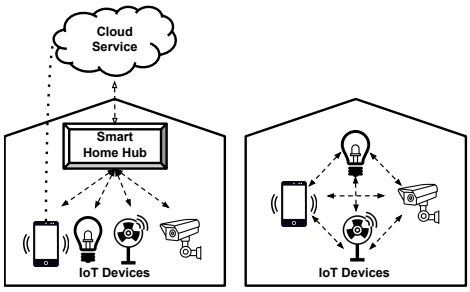

Fig. 1. On the left, we see a smart home setup wherein the cloud relies on a hub to communicate with IoT devices. On the right, we see our proposed framework in which no physical hub or cloud management service is required.

sometimes considered to be central actors in a smart home system [4]. Another defining trait is that communications between IoT devices are managed so devices can change states based on direct requests as well as changes in the environment. Controlling the execution of these state changes and ensuring the appropriate information is transmitted is an essential function of smart home hubs.

As shown in Figure 1, the actual configuration of a smart home network can potentially involve a physical hub that serves as an intermediate connection between IoT devices and the cloud, but it is also feasible for devices to directly connect to the cloud itself. In such a set-up, the physical hub does not really function as the "hub" for the IoT network, but rather sends data to the cloud which actually controls the IoT devices and automates their state changes [4]. One could also remove the involvement of the cloud service in the left half of Figure 1 and opt for an IoT network where the physical hub takes on these automation and management roles instead. Our proposal is to eliminate both entirely, as seen on the right hand side.

A smart home hub not only coordinates communication between devices, but also controls what devices are added or removed from the network, monitors these devices, and automates their operation. On a deeper level, it identifies what devices connect to it and what information each device needs from the rest of the network. In terms of automation, each IoT device outputs different data, and through a set of conditional statements, the smart home hub determines if any device connected to it should change states. It also resolves potential conflicts that occur as some rules resulting from dependencies between devices may inherently negate each other [11].

B. Secure Multiparty Computation

At the core of our framework is the notion of secure multiparty computation (MPC), through which we consider each IoT device a party in a distributed computation. MPC provides the structure by which parties can pass in inputs to some secure joint computation and receive their respective outputs without directly accessing the original inputs from the other parties. Aside from the fact that the only information each party obtains from the computation is its respective output, there are other criteria that define secure multiparty computation. A distributed computation is only considered secure if the outputs are correct and guaranteed to be received accordingly and inputs of corrupt and honest parties are selected independently. Moreover, corrupt parties should not receive outputs if honest parties do not and vice versa [12].

There are many variations of MPC, largely dependent on the nature of the adversaries involved. One type of threat model is based on semi-honest adversaries, also referred to as "honest but curious". In this setup, while the corrupt parties attempt to access private data, they do not deviate from the protocol at all. Conversely, a model based on malicious adversaries operates under the notion that corrupt parties can randomly deviate from the protocol. Similar to this are covert adversaries, which may also deviate from the protocol but only have a certain probability of being detected. If undetected, such adversaries can potentially gain access to the inputs of other parties [12].

While we use secure multiparty computation in the context of smart homes, it is also important to note that it has applications in a wide range of fields [12]. For example, MPC can be used for secure data analysis. The Boston Women's Workforce Council implemented MPC to quantify the gender pay-gap of employees without exposing their personal information [13]. It also has applications in emerging technologies as in the case of Duality, a program that uses MPC to allow data owners to give machine learning model developers the ability to train their models, with each party not knowing the details of the information they did not provide in the transaction [14].

III. System Overview

We outline our framework for running a MPC computation on an inputted smart home configuration below. All code referenced can be found here: https://github.com/spacelab-ccny/virtual-hub.

A. Threat Model

Our framework operates under the assumption that after an initial trusted party (e.g. the user setting up the initial smart home configuration files on each device) defines the parameters of the network and gives each device access to the MPC computation file, none of the devices are considered to be "trusted". Instead we employ a semi-honest threat model in which we consider each party to have the potential to maliciously try to retrieve data from other parties while still adhering to the defined protocol. We expect a corrupt party to try to obtain information about other parties' inputs through data leakage in the distributed computation, but likewise assume adversaries to be passive, meaning they cannot alter transmitted data. In this scenario we assume a stronger model, like malicious adversaries, is unnecessary because we believe the user could potentially detect misbehavior out-of-band.

Additionally, we make the assumption that the parties cannot be fully trusted on the basis that IoT devices themselves present potential security and privacy threats. As presented in Ren et al.'s study on 81 different IoT devices, 72 of the 81 devices sent data to external parties besides their manufacturers. While a significant portion of those parties were meant

Fig. 2. For initial set-up, we pass an input XML file into a program that processes the tags and generates a file to run the computation.

for computing support, certain devices also sent information to third party advertising and analytic companies [1]. We thus consider any external communication that occurs between an IoT device and the internet to be a vulnerability.

B. Framework Outline

To address security and privacy concerns while avoiding the use of unnecessary additional hardware, we suggest the notion of a virtual smart home hub. In our proposed set-up, each device is able to communicate directly with other devices in the home and automate its own tasks accordingly without a designated hub managing these interactions. Through the use of MPC, an individual IoT device can input data to the network without gaining access to the information other devices are inputting and receive a signal indicating whether or not it should change states based on a secure computation that occurs. In our case, this computation is a series of "if" statements (commonly used in IoT automations [11]) to compare the current states of each device to certain thresholds and see if device states must be altered accordingly.

We begin with a smart home configuration file; in our case we use an XML file where a user can define various elements with various attributes corresponding to the appropriate headers (e.g. `<deviceName> fan <\deviceName>`). This file will be filled in by an initial trusted party: the owner of the house setting up the network on their phone, for example. It does not include inputs of the current state of each device, but rather is meant to provide information about what devices are connecting to the network, how many inputs they will pass into our computation, and how these devices depend on one another. For instance, this file might describe that there is both a fan and a thermostat in the network, with the fan inputting if it is on or off and the thermostat inputting the room's temperature. It would also contain the relationship between these devices without including any actual numerical values. In this example, it would describe that if the thermostat indicates a certain temperature is exceeded, the fan should change states. The fan would later input what state it should change to and the temperature threshold value needed for it to change states. As seen in Figure 2, once the XML file is filled out, it is processed by a Python script that generates another Python file with the corresponding comparisons necessary to automate the states of devices in the network.

As shown in Figure 3, along with the initial configuration file, we also assume the user creates individual configuration files that correspond to each device and contain information about the device state. These individual configuration files have the same structure as the initial configuration file, but

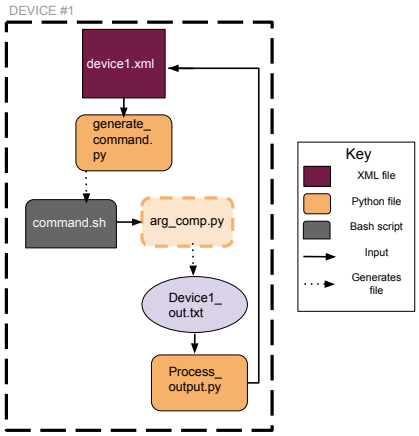

Fig. 3. Using the Python script generated from the initial configuration file, each device processes its own configuration file and runs the computation, updating its configuration file accordingly.

each device's individual configuration file contains the numerical values relevant to that device. For example, the fan's configuration file would include the temperature threshold values required for it to turn on but not the temperature threshold values required for the blinds to close. These files are then processed by a script which runs locally on the device ("generate_command.py" in Figure 3) that extracts the numerical values and writes them as command line arguments for the computation file executable.

Once the arguments are passed into the computation (through "command.sh" in Figure 3) and it is executed, each device receives one output per state change it might incur if certain conditions are met. These outputs each either correspond to the existing value of one of the input states (meaning no change is necessary) or they reflect the value one of the input states should change to. A script is run to process these results and update values accordingly.

As described above, our design operates under the assumption that each device in the network has the capability to do the necessary computations independently. We consider the logistics of integrating our framework with the software of an IoT device to be outside the scope of this project. We similarly will not be taking into account the process of how these IoT devices might connect to the internet without a designated centralized hub in our design, but instead operate under the assumption that this functionality could be implemented at a later stage, for instance by adding a router as an IoT device to the network, as future work.

C. Security Analysis

Our Python code uses the MPyC package, which allows multiple parties to connect and provide inputs to a computation, but encodes these inputs in such a way that, while each party has access to all the inputs to run the computation, none of the parties knows what these values actually are. In particular, MPyC implements MPC through Shamir's threshold secret sharing scheme and securely operates when

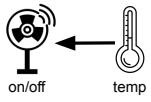

Fig. 4. Input Config. 1. The fan changes states based on the temperature read by the thermostat.

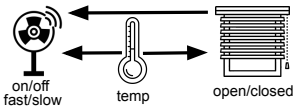

Fig. 5. Input Config. 2. The fan changes states based on input from the blinds and the thermostat while the blinds only depend on the thermostat.

the adversaries are passive and in control of less than 50% of the parties [15]. This mitigates the risks discussed in our threat model, ensuring secure computation if less than half of the parties are corrupted.

Because the initial configuration file uploaded to each device contains no numerical information and the individual configuration files only contain numerical values for their respective devices, we further address the risk of a potentially corrupt device gaining any information about the state of devices in the rest of the network. In this way, apart from the initial setup, the network can operate under the assumption that parties may be corrupted and still automate processes securely.

IV. Evaluation

We simulate our framework using two Raspberry Pi 4B devices and a Raspberry Pi 3 to reflect the limited computing capabilities an IoT device might have. Due to budget and resource constraints, rather than implement our virtual hub using actual IoT devices, we seek instead to establish feasibility on IoT-class hardware. In our testing, device number 3 was always simulated using the Raspberry Pi 3.

To test our setup, we use various initial configuration files meant to reflect realistic device setups one might have in a smart home. We ultimately ran the computation 100 times on 4 different smart home configurations (see Figures 4 to 7) with varying degrees of dependency between the devices. Each time before the computation is re-run, we process the configuration files and alter input values with a 33% chance to reflect how, in a real smart home, devices can be manually adjusted and environmental factors, like room temperature, may also change between computations. This probability value was arbitrarily selected to ensure that a fair portion of the runs involves some sort of state change. As the computation is automated and device configuration files update, we are able to better simulate the dynamic nature of a smart home IoT network.

Using MPyC, we establish a number of bidirectional TCP/IP connections in our local network that directly corresponds to the number of devices. According to the documentation, for m devices involved in the computation, this result is exactly $\binom{m}{2}$ [15]. So for 3 devices, we make a total of 3 TCP connections.

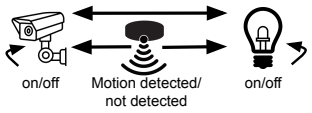

Fig. 6. Input Config. 3. The light turns on when motion is detected, and the camera turns on if both motion is detected and the light is on. If motion is no longer detected, and both the light and camera are on, the camera turns off. If the light is on but no motion is detected and the camera is off, it turns off.

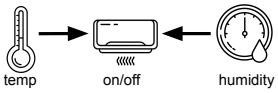

Fig. 7. Input Config. 4. The air conditioner depends on input from both the thermostat and the hygrometer.

We measured statistics only for the process in which the actual multiparty computation occurs. For each device involved in a given computation we measured the runtime of the process, bytes sent in the transaction, as well as the memory metrics of Resident Set Size (RSS) and the Unique Set Size (USS). The purpose of including both was to highlight the distinction between memory that is both used by the process and shared between processes (as captured by RSS) and memory that is solely tied to the running process that would be freed if it was terminated (as captured by USS) [16].

A. Results

As seen in Table 1, we observe that all of our simulations resulted in under 1MB of data being transmitted by each device. Thus, under our framework, the network does not experience significant overhead in terms of bandwidth consumption. We note that, by design, even if the device's conditions to change states are not met, the device's current state value is output for that threshold comparison. So, in a given configuration, each device consistently sends out a fixed number of bytes

The duration of the computation is also relatively low, with times always under 0.6 seconds on average. This suggests that in a real-world setting IoT devices under this framework could transmit data between themselves and determine whether or not to change states with relatively low latency. As such, the latencies of our results would likely be negligible from the user's perspective. However, this may change in configurations with additional devices and more complex dependencies.

We also note that the RAM usage, both RSS and USS, is fairly low, with RSS peaking around 38MB for all of our simulations. For reference, a Nest Thermostat has 256 MB of memory [17] and some IoT devices have far more, like the Samsung Smart Fridge which has 2.5 GB of RAM [18], both of which greatly exceed the maximum RAM usage of our computation. Thus, it is reasonable to assume our secure multiparty computation would be able to run on existing IoT devices. Ultimately, this supports the notion that our proposed framework in which IoT devices individually run the computations necessary for secure multiparty computation

TABLE I
SIMULATION RESULTS FOR DIFFERENT SMART HOME CONFIGURATIONS.

Config.	Device	Avg. Time (s)	Avg. RSS (MB)	Avg. USS (MB)	Bytes Sent
1	1	0.15±0.135	37.44±.011	26.86±.003	108
	2	0.03±0.037	37.44±.003	27.23±.003	18
2	1	0.42±0.133	37.55±.044	26.93±.003	4812
	2	0.42±0.135	37.37±.024	27.02±.003	4578
	3	0.41±0.129	24.81±.024	19.40±.003	4560
3	1	0.54±0.184	37.60±.002	27.01±.003	8646
	2	0.54±0.18	27.09±.027	27.09±.003	8898
	3	0.54±0.172	24.88±.029	19.46±.004	8898
4	1	0.47±0.175	37.48±.053	26.91±.003	4758
	2	0.47±0.175	37.24±.005	27.00±.003	4470
	3	0.45±0.167	24.78±.016	19.38±.003	4524

to allow them to automate state changes is not necessarily restricted by computing abilities of the devices. This suggests that we could potentially extend this system for a real-world IoT network without a hub.

V. FUTURE WORK

Since we did not have access to a full test bed of devices, we believe future work can build upon our setup by increasing the amount of devices involved in testing the framework. As seen in our results, we expect that increasing the number of devices and dependencies involved may increase both the latency and bytes sent per device in the computation. Still, more testing is required to determine exactly how this might scale as the network grows and if there is some threshold at which adding more devices and more complex dependencies under this framework becomes detrimental to overall operation. Moreover, we believe it would be beneficial to study the human interaction aspect of our model, such as whether or not users would see an increase in latency between state changes of devices in a negative light if they know the framework as a whole mitigates security threats. Future iterations could also extend our framework to incorporate other features of a smart home hub, like device identity management [4].

VI. CONCLUSION

Through this paper we suggest a framework in which one could develop a network between IoT devices that would not require a centralized hub to manage them. In introducing a system through which secure multiparty computation can be implemented to mitigate security risks while still allowing state changes to be automated, we have shown that theoretically, a decentralized IoT network is realizable. Moreover, our results highlight that our framework has the potential to be implemented on actual IoT devices without incurring unreasonable runtimes and RAM usage. We expect future work can build on the structure we introduced, developing a virtual hub network where individual IoT devices not only automate their own state changes but also jointly determine whether to add or remove devices and monitor traffic between devices.

ACKNOWLEDGMENTS

The authors would like to thank the CyberNYC REU program and Prof. Rosario Gennaro for the opportunity to pursue this research. The authors are funded by the Algorand Foundation and by the NSF under award 1955172. The views and conclusions contained herein are those of the authors and should not be interpreted as representing those of the sponsors. Any mention of specific companies or products does not imply any endorsement by the authors or the sponsors.

REFERENCES

[1] J. Ren, D. J. Dubois, D. Choffnes, A. M. Mandalari, R. Kolcun, and H. Haddadi, "Information exposure from consumer iot devices: A multidimensional, network-informed measurement approach," in *Proceedings of the Internet Measurement Conference*, ser. IMC '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 267–279. [Online]. Available: https://doi.org/10.1145/3355369.3355577

[2] X. Ye and J. Huang, "A framework for cloud-based smart home," *International Conference on Computer Science and Network Technology*, 2011.

[3] E. Zeng, S. Mare, and F. Roesner, "End user security and privacy concerns with smart homes," in *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*. Santa Clara, CA: USENIX Association, Jul. 2017, pp. 65–80. [Online]. Available: https://www.usenix.org/conference/soups2017/technical-sessions/presentation/zeng

[4] W. Zhou, Y. Jia, Y. Yao, L. Zhu, L. Guan, Y. Mao, P. Liu, and Y. Zhang, "Discovering and understanding the security hazards in the interactions between IoT devices, mobile apps, and clouds on smart home platforms," in *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 1133–1150. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/zhou

[5] P. Angin, B. Bhargava, R. Ranchal, N. Singh, M. Linderman, L. B. Othmane, and L. Lilien, "An entity-centric approach for privacy and identity management in cloud computing," in *2010 29th IEEE Symposium on Reliable Distributed Systems*, 2010, pp. 177–183.

[6] D. Geneiatakis, I. Kounelis, R. Neisse, I. Nai-Fovino, G. Steri, and G. Baldini, "Security and privacy issues for an iot based smart home," in *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2017, pp. 1292–1297.

[7] S. Khanji, F. Iqbal, and P. Hung, "Zigbee security vulnerabilities: Exploration and evaluating," in *2019 10th International Conference on Information and Communication Systems (ICICS)*, 2019, pp. 52–57.

[8] O. Olawumi, K. Haataja, M. Asikainen, N. Vidgren, and P. Toivanen, "Three practical attacks against zigbee security: Attack scenario definitions, practical experiments, countermeasures, and lessons learned," in *2014 14th International Conference on Hybrid Intelligent Systems*, 2014, pp. 199–206.

[9] D. Geneiatakis, I. Kounelis, R. Neisse, I. Nai-Fovino, G. Steri, and G. Baldini, "Security and privacy issues for an iot based smart home," *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2017.

[10] A. Brown, R. Mortier, and T. Rodden, "Multinet: Reducing interaction overhead in domestic wireless networks," 2013.

[11] H. Lin, C. Li, J. Lang, Z. Wang, L. Fan, and C. Duan, "Cp-iot: A cross-platform monitoring system for smart home," 2024.

[12] Y. Lindell, "Secure multiparty computation," *Commun. ACM*, vol. 64, no. 1, p. 86–96, dec 2020. [Online]. Available: https://doi.org/10.1145/3387108

[13] A. Lapets, F. Jansen, K. D. Albab, R. Issa, L. Qin, M. Varia, and A. Bestavros, "Accessible privacy-preserving web-based data analysis for assessing and addressing economic inequalities," in *Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies*, ser. COMPASS '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3209811.3212701

[14] "Duality," https://duality.cloud, 2024.

[15] "Mpyc demos," 2018-2024. [Online]. Available: https://mpyc.readthedocs.io/en/latest/mpyc.html

[16] G. Rodola, "Real process memory and environ in python," 2016.

[17] Google, "Nest thermostat technical specifications," 2024. [Online]. Available: https://support.google.com/googlenest/answer/9230098?hl=en

[18] Samsung, 2024. [Online]. Available: https://www.samsung.com/us/home-appliances/refrigerators/4-door-flex/29-cu--ft--smart-4-door-flex--refrigerator-with-family-hub--and-beverage-center-in-black-stainless-steel-rf29a9771sg-aa/