PION: Password-based IoT Onboarding Over Named Data Networking

Davide Pesavento, Junxiao Shi, Kerry McKay, and Lotfi Benmohamed
National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA
{davide.pesavento, junxiao.shi, kerry.mckay, lotfi.benmohamed}@nist.gov

ICC 2022 - IEEE International Conference on Communications | 978-1-5386-8347-7 /22/$31.00 USG | DOI: 10.1109/ICC45855.2022.9839088

***Abstract*—While the IoT market continues to grow, securing IoT systems remains a challenge as successful cyberattacks keep escalating. Named Data Networking (NDN) offers a number of advantages over traditional IP-based communications and is considered a promising candidate to revolutionize the IoT space, thanks to its improved scalability and built-in security features. A cornerstone of any NDN IoT network is the onboarding protocol, whose main goal is to bootstrap the cryptographic keys and trust relationships necessary for a newly joining device to securely communicate with the rest of the network. Though several such protocols have been proposed, none so far combines strong security guarantees with ease of use on IoT devices that have highly constrained input/output interfaces. In this paper we introduce a novel password-based onboarding protocol to address this need. In addition to discussing its design, we produce a formalization of the protocol and verify its security properties using an automated analyzer. Finally, we present the results of benchmarking carried out on a proof-of-concept implementation that demonstrates the feasibility of our approach.**

I. Introduction

As the most prominent instance of Information Centric Networking (ICN), Named Data Networking (NDN) is an emerging network protocol with a data-centric communication architecture based on the retrieval of named and cryptographically signed content [1], [2]. The IoT market continues to grow as IoT brings numerous compelling benefits to many sectors. However, IoT systems keep experiencing attacks with significant cybersecurity impacts [3]. Using NDN for IoT networking has several advantages [4]–[6], including for device onboarding which is the subject of this paper.

Before an IoT device is allowed to join an NDN network, it must go through an *onboarding* procedure (also known as *bootstrapping* or *sign-on*) whereby a secure authentication protocol ensures that only legitimate devices can obtain the necessary credentials and can establish a local identity that is certified by the network's trust anchor. Most of the IoT solutions deployed today rely on cloud services for onboarding as well as authentication, data storage, and remote control, which comes with security and privacy risks [7]. By leveraging NDN's secure data-centric communications, we can develop IoT networking solutions that do not rely on cloud services.

In this paper we present **PION**, a password-based IoT onboarding protocol for NDN that makes use of a local certification authority to achieve fully local control of the IoT network security directly at the edge, without requiring any cloud support or even an Internet connection. The proposed protocol possesses several properties that represent a significant step forward in securing NDN IoT systems:

- We employ a password-authenticated key exchange as a building block to establish a strong symmetric key between two parties that share only a short mnemonic password. A simple password delivers a friendlier user experience and is easier to distribute using a wider range of methods, which is an important requirement for the constrained and heterogeneous platforms common in IoT.
- We build upon NDN's native security features, in particular its per-packet cryptographic signatures, naming, and digital certificates, providing a solid foundation for data-centric security independent of the actual communication channel being used to transport the packets.
- For certificate issuance, our solution leverages and extends the standard certificate management protocol NDNCERT. This simplifies the integration of the onboarding process with the various other security mechanisms and conventions used by NDN applications.
- We present a formalization of the protocol and the results of a security analysis conducted with the help of an automated software tool.
- We assess the performance of a minimal IoT testbed running a complete prototype of the protocol and demonstrate that our approach is viable on constrained devices in terms of memory usage and binary code size.

The rest of this paper is organized as follows. In Section II we review previous publications related to ICN and IoT. A few important foundational concepts are introduced in Section III, while the system model is described in Section IV. The proposed protocol is presented in Section V, followed by its security analysis (Section VI) and experimental evaluation (Section VII). Finally, Section VIII concludes the paper.

II. Related Work

Several existing works discuss ICN-based architectures for the Internet of Things [4]–[9] and various security aspects of NDN [10]. Among them we mention *Breaking out of the cloud* [7], which sketches out a cloud-independent approach to build IoT applications, with local trust management and rendezvous service. PION fully embraces this design direction by avoiding any reliance on remote cloud services and goes a step further in defining a detailed protocol for the specific task of onboarding IoT devices on an NDN network.

OnboardICNg [11] proposes a design based on the well-known AKEP2 authenticated key exchange, which later inspired EAP-PSK. The security of these protocols relies on the assumption that the pre-shared key (PSK) is cryptographically strong and thus must be relatively long (16 or 32 bytes). This is a fair assumption in many scenarios but may be an obstacle to wider adoption on less capable devices. Other ICN onboarding protocols [12], [13] have similar expectations.

The need for a strong PSK has several undesirable implications. Firstly, if the device does not have an easily accessible input or output interface, exchanging such a long key out-of-band is error-prone and cumbersome for the user. Secondly, if the PSK is static, e.g., it is created once when the device is manufactured and printed as a QR code on the device or its packaging, there is an increased chance that an attacker could compromise the confidentiality of the PSK and thus subvert the security of the whole system. This issue is exacerbated when we consider second-hand devices, in which case all previous owners know the device's PSK and could easily gain unauthorized access to the network of the new owner. Thirdly, if the PSK is freshly generated at each onboarding attempt, it requires a cryptographically secure pseudorandom number generator which may not always be available.

Finally, with the exception of NDNViber [14], none of the existing proposals is compatible with the standard NDNCERT certificate management protocol, making it difficult to integrate the onboarding process with other security mechanisms and conventions used by NDN applications.

III. Background

A. NDN Certificates

An NDN *certificate* [15] is a digital certificate used to bind a public key to the identity of its owner. It is an NDN Data packet that can be retrieved via Interest-Data exchange and cached anywhere in the network. The Data name is the owner's identity name followed by additional name components that identify the public key and the certificate issuer. The packet carries the public key itself and a validity period for the certificate, and is cryptographically signed by its issuer, such as a certificate authority. A certificate can be *self-signed*, i.e., signed by the private key corresponding to the public key contained in the certificate itself. This can be used either as a root trust anchor or as a certificate request.

B. The NDNCERT Protocol

The NDNCERT certificate management protocol [16] is an application-layer protocol that enables automated management of the whole NDN certificate lifecycle, from the first certificate application to its renewal and/or revocation.

A certificate application is a request-response protocol enacted between two entities: a *certificate authority* (CA) and a requester. In the first round of the protocol, called New, the requester sends its self-signed certificate to the CA and the two parties establish a symmetric session key through an ephemeral elliptic-curve Diffie-Hellman (ECDH) key agreement. This key is then used to encrypt all subsequent messages with AES-GCM. The procedure continues with one or more rounds of Challenge, in which the requester proves its identity to the CA; how this is done depends on the specific challenge type negotiated between requester and CA. Eventually, if the challenge succeeds, the CA issues (i.e., signs) the certificate.

NDNCERT has a variety of pre-defined challenges. Some of them involve an out-of-band step, such as receiving a PIN code via email. We are interested in the *proof-of-possession* challenge, which expects the requester to own a preexisting certificate issued by either the same or a different CA. In this challenge, the CA generates a random nonce and sends it to the requester, which should sign it with the key corresponding to the existing certificate and return it to the CA. If the signature is valid and the existing certificate satisfies the CA's policy, the challenge is considered successful.

C. Password-Authenticated Key Exchange

Password-Authenticated Key Exchange (PAKE) schemes, first introduced by Bellovin and Merritt [17], are a family of interactive protocols in which two or more parties authenticate each other and derive a shared cryptographic key based on their knowledge of a weaker shared password, without ever transmitting the password over the (insecure) communication channel. The idea is that an attacker cannot carry out an offline dictionary attack against the protocol, because checking the correctness of each password guess requires interacting with either one of the victims. Thus, in a PAKE it is important to limit the maximum number of authentication attempts to only a handful of tries, after which the shared password must be changed. Furthermore, relying just on a short and simple password, as opposed to a cryptographically strong key, provides a number of advantages when the password needs to be displayed to a human user, typed, or communicated out of band, which ultimately yields a friendlier user experience and a lower chance of errors.

Several PAKEs have been proposed over the years. We chose to use SPAKE2 [18], [19] in PION for its simplicity, efficiency, and because it has been proved secure [20].

IV. System Model and Assumptions

A. System Model

We consider a home or small office environment where networked devices communicate over WiFi (IEEE 802.11). We assume that the WiFi network is protected by WPA2-Personal, WPA3-Personal, or any other similar method that requires a shared password for authentication. At the network layer, NDN may be used directly over layer 2 or as an overlay over IP. The NDN network has a local *certificate authority* (CA) which, for simplicity, also acts as the trust anchor.

To avoid imposing additional requirements on the user interaction capabilities of the certificate authority software, which may be running on a headless machine such as the network's router or gateway, we introduce a second logical entity: the *authenticator* (H). The authenticator is the main interface through which the user authorizes new devices, for instance it could be part of a network management app installed on the user's smartphone. It is assumed that H has already established a mutual trust relationship with the CA prior to the execution of the onboarding protocol.

An IoT device D that wishes to join the network must learn the network credentials (WiFi password) before it can connect, and must obtain a local identity that is certified by the CA before it can start interacting with other entities on the network. D may be a newly purchased device, or it may be a device that has been factory-reset after being acquired from a previous owner or moved from a different network. We expect that D may have very limited computational, memory, and storage resources, while CA and H are essentially unconstrained.

The objective of our onboarding protocol is twofold: D should securely learn the network credentials and the identity of the trust anchor; at the same time, it should obtain an NDN certificate for its public key signed by the CA. Moreover, the protocol must guarantee that no entity other than D can obtain a valid certificate from the CA and that the network credentials are never disclosed to any unauthorized party.

B. Threat Model

We consider a threat model where the attacker is bound by the same computational and memory limits of the underlying cryptographic primitives. We assume that the attacker is able to eavesdrop on all network traffic, can inject forged, altered, or replayed packets, and can delay or prevent reception of packets sent by other network nodes. However, the attacker may not break or subvert the security of cryptographic primitives that are considered secure, such as ECDSA, HMAC, AES, SHA-2, etc., as long as the assumptions of those primitives are met.

Because this model gives the attacker control over the channel, it is possible for the attacker to prevent an honest device from completing the onboarding successfully. This is a property of the model and channel, not a failure of the protocol. The adversary has successfully broken the protocol if they can fraudulently obtain a valid certificate from the CA.

V. The PION Protocol

The proposed PION onboarding protocol consists of five steps, depicted in Figure 1 and described in the subsections below. The authenticator participates only in the first three steps of the protocol when the direct connection is active. Conversely, the CA takes part only in the last phase when the device connects to the home network. No communication between authenticator and CA needs to occur at any point. This was a deliberate design choice that allows the authenticator to onboard multiple devices one after the other without reconnecting to the home network, thereby enabling a more streamlined experience even with a large number of devices.

A. Direct Connection

The authenticator H establishes a "direct" connection to the device D. This can be achieved in various ways depending on the communication technologies supported by both parties. For instance, D could set up a WiFi access point with a well-known SSID and H could connect to it, or they could use

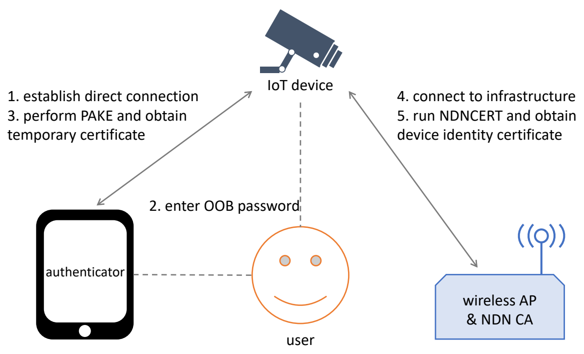
IoT device
1. establish direct connection
3. perform PAKE and obtain temporary certificate
4. connect to infrastructure
5. run NDNCERT and obtain device identity certificate
2. enter OOB password
authenticator
user
wireless AP & NDN CA

Fig. 1. Overview of PION protocol interactions.

Bluetooth, Zigbee, etc. This connection may even traverse multiple layer-2 hops as long as it appears as a single hop at the NDN layer. Since the protocol makes no assumptions on the security of this connection, the intermediate nodes do not need to be trusted.

B. OOB Step

The user ensures that D and H agree on an *out-of-band* (OOB) password. The OOB password guarantees the mutual authentication of D and H, and prevents impersonation and man-in-the-middle attacks on the protocol. A major advantage of PION is that the password can be short and have low complexity, thanks to the use of a PAKE (see Section III-C). Typically, 20 bits to 30 bits of entropy are considered sufficient, equivalent to a 7-digit PIN or a 5-character alphanumeric string randomly sampled from a uniform distribution.

PION does not mandate how the OOB password should be shared between D and H. In practice, this step is highly dependent on the hardware capabilities of the individual devices. We envision that the authenticator software will support multiple OOB methods and will pick the most appropriate one based on the specific device being onboarded. For instance [21], a connected light bulb could transmit the password by blinking in specific patterns and the authenticator could detect these patterns through the smartphone's camera. Reversing the roles is also possible, e.g., an electronic door lock with only a keypad may not have a way to produce an output. In that case, the password must be generated and displayed by the authenticator and then entered on the keypad by the user.

As pointed out in Section III-C, it is critically important to stop brute-force attacks by limiting the number of password guesses that an attacker is allowed to make. Consequently, PION requires the OOB password to be regenerated and reshared after three failed onboarding attempts.

C. PAKE Step

This step of the protocol takes place between D and H after they have established a direct connection, and consists of six messages (three round-trips). In the first two messages, SPAKE2 [19] is employed to derive a strong shared key from the low-entropy OOB password previously exchanged. Next, an explicit key confirmation round provides assurance that

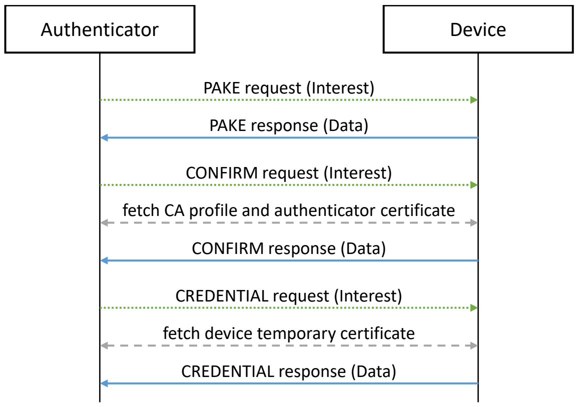
Authenticator
Device
PAKE request (Interest)
PAKE response (Data)
CONFIRM request (Interest)
fetch CA profile and authenticator certificate
CONFIRM response (Data)
CREDENTIAL request (Interest)
fetch device temporary certificate
CREDENTIAL response (Data)

Fig. 2. Sequence diagram of the PAKE step.

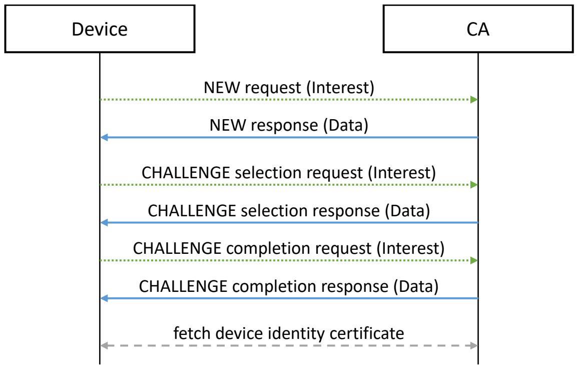
Device
CA
NEW request (Interest)
NEW response (Data)
CHALLENGE selection request (Interest)
CHALLENGE selection response (Data)
CHALLENGE completion request (Interest)
CHALLENGE completion response (Data)
fetch device identity certificate

Fig. 3. Sequence diagram of the NDNCERT step.

the two parties have derived the same key and are therefore authenticated to each other. Finally, the shared key is used to encrypt and authenticate the last round, during which D obtains a temporary NDN certificate (signed by H) and the infrastructure network credentials necessary for the next step. In case this step fails, the OOB password cannot be reused, but a new one must be generated and exchanged.

The first version of PION defines a single ciphersuite, consisting of ECDSA signatures with NIST P-256 as the elliptic curve, SHA-256 as the hash function, HKDF-SHA-256 as the key derivation function, HMAC-SHA-256 as the message authentication code, and AES-GCM-128 as the symmetric cipher. More ciphersuites can easily be added later with no changes to the protocol flow.

D. Infrastructure Connection

At this point, D can tear down the direct link to H and connect to the infrastructure network (e.g., home WiFi) using the credentials acquired in the previous step.

E. NDNCERT Step

Once connected to the infrastructure network, D initiates an NDNCERT certificate request with the CA and selects the *proof-of-possession* challenge to prove its identity, as explained in Section III-B and illustrated in Figure 3. The preexisting certificate used for the challenge is the temporary certificate issued to D by H during the PAKE step. In addition to the standard NDNCERT checks, the CA also verifies that the temporary certificate was indeed issued by a trusted authenticator by looking at the name in the `KeyLocator` field of the certificate. If the challenge is successful, the CA issues a new certificate to D, which indicates that the device is now trusted, and the onboarding protocol is complete.

VI. Security Analysis

The security of PION was analyzed both manually and using the *Cryptographic Protocol Shapes Analyzer* (CPSA) automated tool[1] that searches for valid execution over strand spaces [22]. A *strand* is a sequence of messages sent and messages received by a party, and each role in the protocol is defined by a strand. Multiple strands can be bundled into a single execution of the protocol, e.g., an authenticator strand and a device strand would be needed to show correct execution of the PAKE step of our protocol. CPSA uses the strands defined in the model to search the strand space for complete executions of the protocol. An execution that cannot be generalized is called a *shape*. An analyst can study the shapes found by the tool and determine if something unexpected occurs. All of the shapes that were identified during the search over our models[2] showed expected executions of the protocol.

A. Protocol Modeling

Because CPSA performs a search over the strand space, it will be able to fully examine a simpler model that has a small strand space. As the number of messages and variables in a model increases, the search becomes infeasible. Therefore, it is important to restrict the search to match protocol assumptions. Some aspects of the protocol were not obvious to model in CPSA, or resulted in intractably large search spaces. As detailed below, sometimes our model diverged from the specification in order to improve the search, e.g., by reducing the number of variables. We emphasize that the models still maintain the dependencies and spirit of our protocol.

Modeling the PAKE step: There is no straightforward way in CPSA to express the computation of the SPAKE2 public values sent by D and H. Thus, we treat them as a secret encoding of the public ECDH values using the shared secret w, derived from the OOB password, as the key. Then we can verify that w is not leaked.

The derivations of the SPAKE2 keys were originally modeled using the `fn-of` (function of) relations in CPSA. However, there were many cases in the results where strands shared a key, but did not have the same values for the variables used to derive it. This greatly increased the search space without yielding any new information. So, instead of representing the SPAKE2 keys as functions, we simply use the shared key from the Diffie-Hellman agreement in the search. If we can show that w and the private exponents are not leaked, it follows that the derived keys would have been computed correctly.

The search was further constrained by ensuring that shared secrets, private keys, and transcript are never transmitted.

[1]`https://github.com/mitre/cpsa`
[2]CPSA models and resulting shapes have been omitted due to lack of space.

Moreover, ECDH parameters are fresh and it is assumed that device and authenticator are honest parties.

Modeling NDNCERT with proof-of-possession: The modeling of this phase was fairly straightforward. The search was again constrained by ensuring that shared secrets and private keys are never transmitted. ECDH parameters and the random challenge are fresh and it is assumed that the CA is honest.

VII. Implementation and Evaluation

We wrote a prototype implementation[3] of PION and evaluated it on a small testbed consisting of a Raspberry Pi 3B and an ESP32 development board[4]. The device firmware and the authenticator are implemented in C++, with Mbed TLS 2.16 as crypto library. Where feasible, we used constant-time elliptic curve operations to mitigate potential timing attacks. The certificate authority was separately implemented in Node.js.

To demonstrate that PION is transport-agnostic, our implementation provides a selection of transport technologies. For the direct connection, the device may utilize either WiFi or Bluetooth Low Energy (BLE). Since the BLE connection has a 512-byte MTU, which is smaller than some of the onboarding packets, we also enabled NDNLPv2 fragmentation for the BLE transport. During the infrastructure connection step, the device acts as a WiFi client (station) and connects to the access point in WPA2 mode. This phase may employ either a UDP/IPv4 transport or an Ethernet transport, that transmits NDN packets directly inside layer-2 frames without any IP encapsulation.

We repeated the experiment 25 times for each of the 4 combinations of transport types: WiFi+UDP, WiFi+Ethernet, BLE+UDP, BLE+Ethernet. At the beginning of each run, the device firmware running on the ESP32 generates a fresh OOB password. Then, on the Raspberry Pi, we start the CA and the authenticator, we enter the password on the latter, and let the three parties execute the protocol. The device eventually reports whether it has successfully obtained a certificate, as well as statistics regarding packet timing, packet sizes, and heap memory usage.

A. Experimental Results

We now present the results of our evaluation on the prototype implementation of PION. In particular, we focus on quantifying the hardware resources required on the device to execute the protocol; we also look at the size of all transmitted packets and the time spent processing each received message.

Firmware size: The firmware size, measured with the `size` command, is plotted in Figure 4. The figure shows that the vast majority of the firmware image is taken up by the basic OS functions and layer-2 networking libraries. PION uses between 40 kB and 60 kB of space while the NDN stack needs 100 kB to 160 kB, which is between 2.6x and 4x smaller than the WiFi code and over 6x smaller than the BLE code. We can

[3] `https://github.com/usnistgov/PION` (commit `03e81c5`)

[4] Disclaimer: Any mention of commercial products or reference to commercial organizations is for information only; it does not imply recommendation or endorsement by NIST, nor does it imply that the products mentioned are necessarily the best available for the purpose.

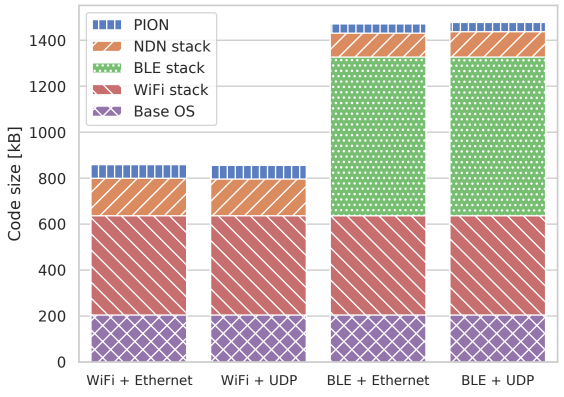

Fig. 4. ESP32 firmware size.

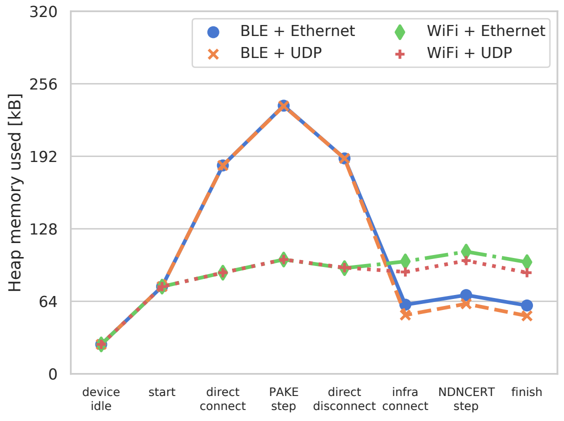

Fig. 5. Heap memory used on ESP32.

conclude that NDN and PION have a very minor impact on the footprint of the device firmware, therefore code size should not be regarded as an obstacle to their adoption.

Device memory consumption: In Figure 5 we show the heap memory utilization at various stages of the protocol execution, roughly corresponding to the steps outlined in Section V. The "device idle" state is added for reference and represents the baseline memory consumption before starting the onboarding. We can observe that the largest chunk of memory is allocated by the BLE stack during the direct connection phase and is then released when the connection is torn down. Up to the "infra connect" stage, there is no difference between the Ethernet and UDP scenarios because those transports come into play only with the infrastructure connection step.

Packet size and processing time: Throughout the execution of the experiments, we recorded the size of each packet and the time at which it was sent/received by the device. From these timestamps we can calculate how long it takes for the device to process one received packet and prepare the next one. The results are reported in Table I, where each row roughly corresponds to one of the substeps depicted in Figures 2 and 3.

The packet size metric incorporates all the NDN and NDNLPv2 headers in addition to the PION protocol data, but excludes UDP and Ethernet headers. As expected, the

TABLE I
PACKET SIZE AND PROCESSING TIME FOR EACH MESSAGE EXCHANGE.

PAKE STEP	Size (B)		Time (ms)	
	WiFi	BLE	WiFi	BLE
PAKE	504	504	1235.20	2081.08
CONFIRM	2053	2125	1971.08	4074.56
CREDENTIAL	1103	1103	327.52	903.04
Total	3660	3732	3533.80	7058.68
NDNCERT STEP	Size (B)		Time (ms)	
			Ethernet	UDP
NEW	978		1298.66	1269.68
CHALLENGE	1770		2029.68	1939.18
FETCH CERT	479		54.54	19.32
Total	3227		3382.88	3228.18

number of bytes transmitted is unaffected by the transport technology, except in the CONFIRM substep. This substep includes two Interest-Data exchanges to fetch the CA profile and the authenticator's certificate. Both are too large for the MTU of the BLE connection and thus require fragmentation, which adds 72 bytes of overhead. WiFi has a much larger MTU, so all other transport types are not affected by this.

The computation cost of PION is largely dominated by the elliptic curve functions, but also includes packet encoding/decoding and low-level send/receive operations. We can notice that sending packets over BLE is much slower than over WiFi, which is to be expected. Overall, a complete execution of the PION protocol takes about 6.7 seconds on this particular IoT device, assuming WiFi and UDP, which is more than acceptable given that onboarding is performed very infrequently. We stress that this is a proof of concept and has not been optimized yet. We anticipate that more efficient implementations will substantially reduce the runtime.

VIII. CONCLUSION

This paper describes PION, a protocol for the secure onboarding of IoT devices joining an NDN network. Thanks to its novel design, PION greatly simplifies the out-of-band authentication step, reducing the hardware requirements for the device and at the same time improving the user experience and providing strong security guarantees. Formal analysis showed no unexpected executions where an attacker can subvert the integrity of the protocol. Furthermore, an unoptimized prototype implementation demonstrated reasonable performance on a constrained device, completing a full onboarding procedure in 6.7 seconds and transmitting less than 7 kB of data.

Additional enhancements are planned as future work, such as native support for multi-hop forwarding schemes used in NDN mesh networks. We are also exploring ways to reduce the number of round-trips without compromising the security.

REFERENCES

[1] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, 2009, pp. 1–12.

[3] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, and A. Zanella, "Iot: Internet of threats? a survey of practical security vulnerabilities in real iot devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8182–8201, 2019.

[4] W. Shang, A. Bannis, T. Liang, Z. Wang, Y. Yu, A. Afanasyev, J. Thompson, J. Burke, B. Zhang, and L. Zhang, "Named data networking of things," in *2016 IEEE first international conference on internet-of-things design and implementation (IoTDI)*. IEEE, 2016, pp. 117–128.

[5] Z. Zhang, E. Lu, Y. Li, L. Zhang, T. Yu, D. Pesavento, J. Shi, and L. Benmohamed, "Ndnot: A framework for named data network of things," in *Proceedings of the 5th ACM Conference on Information-Centric Networking*, 2018, pp. 200–201.

[6] M. Frey, C. Gündoğan, P. Kietzmann, M. Lenders, H. Petersen, T. C. Schmidt, F. Juraschek, and M. Wählisch, "Security for the industrial iot: the case for information-centric networking," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*. IEEE, 2019, pp. 424–429.

[7] W. Shang, Z. Wang, A. Afanasyev, J. Burke, and L. Zhang, "Breaking out of the cloud: Local trust management and rendezvous in named data networking of things," in *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*, 2017, pp. 3–13.

[8] E. Baccelli, C. Mehlis, O. Hahm, T. C. Schmidt, and M. Wählisch, "Information centric networking in the iot: Experiments with ndn in the wild," in *Proceedings of the 1st ACM Conference on Information-Centric Networking*, 2014, pp. 77–86.

[9] M. Amadeo, C. Campolo, A. Iera, and A. Molinaro, "Information centric networking in iot scenarios: The case of a smart home," in *2015 IEEE international conference on communications (ICC)*. IEEE, 2015, pp. 648–653.

[10] Z. Zhang, Y. Yu, H. Zhang, E. Newberry, S. Mastorakis, Y. Li, A. Afanasyev, and L. Zhang, "An overview of security support in named data networking," *IEEE Communications Magazine*, vol. 56, no. 11, pp. 62–68, 2018.

[11] A. Compagno, M. Conti, and R. Droms, "Onboardicng: a secure protocol for on-boarding iot devices in icn," in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, 2016, pp. 166–175.

[12] T. Mick, R. Tourani, and S. Misra, "Laser: Lightweight authentication and secured routing for ndn iot in smart cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 755–764, 2017.

[13] Y. Li, Z. Zhang, X. Wang, E. Lu, D. Zhang, and L. Zhang, "A secure sign-on protocol for smart homes over named data networking," *IEEE Communications Magazine*, vol. 57, no. 7, pp. 62–68, 2019.

[14] S. K. Ramani, P. Podder, and A. Afanasyev, "Ndnviber: Vibration-assisted automated bootstrapping of iot devices," in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2020, pp. 1–6.

[15] NDN Project. NDN certificate format version 2.0. [Online]. Available: https://named-data.net/doc/ndn-cxx/0.8.0/specs/certificate.html

[16] ——. NDNCERT protocol version 0.3. [Online]. Available: https://github.com/named-data/ndncert/wiki/NDNCERT-Protocol-0.3/69d841e20515a5f7e8e5452e8366225e55bf2f86

[17] S. M. Bellovin and M. Merritt, "Encrypted key exchange: Password-based protocols secure against dictionary attacks," 1992.

[18] M. Abdalla and D. Pointcheval, "Simple password-based encrypted key exchange protocols," in *Cryptographers' track at the RSA conference*. Springer, 2005, pp. 191–208.

[19] W. Ladd and B. Kaduk, "SPAKE2, a PAKE," Internet Engineering Task Force, Internet-Draft draft-irtf-cfrg-spake2-26, Feb. 2022, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-spake2-26

[20] M. Abdalla and M. Barbosa, "Perfect forward security of SPAKE2." *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 1194, 2019.

[21] J. Suomalainen, "Smartphone assisted security pairings for the internet of things," in *2014 4th International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems (VITAE)*. IEEE, 2014, pp. 1–5.

[22] F. J. T. Fábrega, J. C. Herzog, and J. D. Guttman, "Strand spaces: Proving security protocols correct," *Journal of computer security*, vol. 7, no. 2/3, pp. 191–230, 1999.