What's the Matter? An In-Depth Security Analysis of the Matter Protocol

Sayon Duttagupta [1‡], Arman Kolozyan [2‡], Georgio Nicolas [1‡],
Bart Preneel [1], and Dave Singelée [1]

[1] COSIC, KU Leuven firstname.lastname@esat.kuleuven.be
[2] Vrije Universiteit Brussel firstname.lastname@vub.be

Abstract. The Matter protocol has emerged as a leading standard for secure IoT interoperability, backed by major vendors such as Apple, Google, Amazon, Samsung, and others. With millions of Matter-certified devices already deployed, its security assurances are critical to the safety of global IoT ecosystems. This paper presents the first in-depth security evaluation and formal analysis of Matter's core protocols, focusing on its Passcode-Authenticated Session Establishment (PASE) and Certificate-Authenticated Session Establishment (CASE) mechanisms. While these are based on the well-studied SPAKE2+ and SIGMA respectively, Matter introduces modifications that compromise the original security guarantees. Our analysis reveals multiple cryptographic design flaws, including low-entropy passcodes, static salts, and weak PBKDF2 parameters – all of which contradict Matter's own threat model and stated security goals. We highlight cases where Matter delegates critical security decisions to vendors, rather than enforcing robust cryptographic practices in the specification, thereby making the system more fragile and susceptible to exploitation. We formally model both standard and Matter-adapted variants of these protocols in ProVerif, confirming several of Matter's security goals, but disproving others. Our findings go as far as rendering some of Matter's own mitigations insufficient, exposing *all* Matter-certified devices to threats classified as *"High Risk"* in their own documentation. As part of our study, we also discovered previously unknown vulnerabilities in Matter's public codebase, which we responsibly disclosed to the developers, leading to updates in the codebase.

1 Introduction

The Internet of Things (IoT) is transforming modern living by bringing automation and connectivity to homes and industries. However, this rapid expansion has introduced significant fragmentation: devices from different manufacturers often fail to communicate or interoperate due to incompatible proprietary protocols. To address this, the Matter protocol [2], was developed by the Connectivity Standards Alliance (Alliance), a consortium comprising prominent industry stakeholders such as Apple, Google, Amazon, Samsung, NXP, Espressif, among others.

Before Matter, IoT integration was complex and inconvenient. Users often had to manage multiple apps and ecosystems to control their devices. For instance, operating a smart light bulb from one manufacturer via a hub from another required compatibility layers or third-party integrations. Matter offers a solution to this by enabling interoperable communication and streamlined commissioning (i.e., pairing) across devices and ecosystems [14,24]. Its mission to unify the IoT ecosystem includes a strong focus on security. IoT systems have historically suffered from security threats such as eavesdropping, unauthorised access, device spoofing, and network intrusion [3,4,26,27].

The Alliance has prioritised security in Matter's design [12], asserting that its protocols and cryptographic standards offer strong protection against a range of adversarial threats. Matter proposes two core key establishment protocols as a part of its specification: PASE and CASE. PASE,

‡These authors have equally contributed to this work.

or Passcode-Authenticated Session Establishment, is used for onboarding new devices. CASE, or Certificate-Authenticated Session Establishment, is used for already commissioned devices. These protocols are based on the cryptographic protocols SPAKE2+ [32] and SIGMA [19], respectively. Both are intended to preserve the integrity and confidentiality of communication. The need for a rigorous security assessment of the Matter standard is underscored by the scale and pace of its deployment, with 8,696 unique Matter-certified devices already listed and deployed worldwide[‡] – each potentially being produced by the thousands.

In this paper, we present the first extensive security analysis and formal analysis of the cryptographic guarantees offered by the Matter protocol. Our study examines Matter's cryptographic foundations, protocol-level design, and claimed security assurances, focusing on the integration of SPAKE2+ and SIGMA within its PASE and CASE protocols. We model these protocols using the applied pi-calculus which enables automatic and reproducible proofs using ProVerif. We evaluate the overall robustness of these protocols and analyse the security consequences of their customisation within Matter. We additionally assess the "Threats and Countermeasures" section of the Matter specification and identify several cases where the stated mitigations do not adequately defend against realistic adversarial capabilities.

In particular, we uncover vulnerabilities corresponding to high-threat scenarios described in Matter's own threat model [13]. These issues may allow attackers to compromise the security of key-establishment processes to gain unauthorised access to Matter-certified devices – posing widespread security risks given the global scale of adoption. Furthermore, we suggest improvements to reinforce Matter's defences against realistic adversaries. Finally, while our primary focus is on the specification-level analysis, we also highlight important implementation vulnerabilities encountered during our analysis – specifically, cases where certified SDKs fail to adhere to the protocol's intended security guarantees. While the Matter documentation alternates between the terms *'Matter Standard'* and *'Matter Protocol'*, for consistency, we use *'Matter Protocol'* throughout this paper to refer to the overall specification. The official Matter specification is available from [13]. At the time of writing, we recommend consulting the 'Matter 1.4 Core Specification' for technical details. When we refer to Matter's threat model or to a specific threat (such as 'T102'), we reference Section 13.7 'Threats and Countermeasures' (page 1081) of the 1.4 specification.

1.1 Prior Works

Matter is widely promoted, implemented, and deployed by numerous IoT vendors worldwide. However, this broad adoption contrasts sharply with the limited scrutiny its design has received from the security and cryptographic research communities. To the best of our knowledge, only four academic studies have conducted any substantive evaluation of Matter's security foundations.

A Master's thesis [23] documents the Matter specifications and outlines potential issues and attack surfaces. It also includes Tamarin models of some protocols, though the unautomated mechanical analysis did not terminate, rendering results inconclusive. Another study [29] explores the trust establishment process between devices and controllers, revealing that while controllers verify the credentials of Matter devices, the reverse is not enforced – allowing unauthorised controllers to join the network. This is demonstrated in an attack [21], though under strong assumptions, such as users leaving QR codes on devices post-commissioning, enabling adversaries to scan and pair with them. A separate work [28] exploits vendor-side implementation flaws, gaining unauthorised access via an unenrolled management channel on commissioned devices.

[‡]Number retrieved on 09/07/2025, parsed from: `https://csa-iot.org/csa-iot_products/`

These studies address architectural concerns or vendor-specific weaknesses, but none directly analyse vulnerabilities in the protocol specification itself. In contrast, our work presents the first comprehensive, protocol-level security analysis of Matter's core design and cryptographic foundations.

A number of works in the space of formal analysis target SPAKE2+ and SIGMA. This study [20] provides an attempt at modeling algebraic properties using horn clauses and demonstrates their approach by modeling a number of protocols, including a variant of SIGMA, using ProVerif. The authors of [6] provide a formalization of SIGMA in the computational model using Coq. A cryptographic analysis of SPAKE2+ in the UC framework is found in [30].

1.2 Contributions

This work presents the first systematic security analysis of the Matter protocol, with a focus on its cryptographic foundations and the trust model embedded in key-establishment processes. We incorporate both practical experimentation and formal methods in our study. Our primary contributions are as follows:

1. **First in-depth security analysis of Matter.** We provide the first structured and rigorous security evaluation of Matter's design, focusing on its two core key establishment protocols – PASE and CASE. Our study covers the full commissioning process, from protocol-level specifications to cryptographic design choices and trust assumptions.
2. **Identification of protocol-level vulnerabilities.** We uncover some questionable design decisions related to PASE and CASE. These decisions may ultimately lead to several cryptographic and design vulnerabilities. We show how the session resumption protocol in CASE degrades the security of long-term Matter sessions. We demonstrate how certain choices enable attackers to bypass security protections in the commissioning process, thus revealing key weaknesses in Matter's key-establishment schemes.
3. **Identification of implementation-level vulnerabilities.** We identify critical deviations from the specification in an official Matter SDK, which introduce exploitable security flaws. These issues were responsibly disclosed and promptly patched by the developers.
4. **Practical validation of our attacks on Matter devices.** We present proof-of-concept attacks demonstrating the practical feasibility of brute-force and pre-computation attacks on Matter's commissioning process under its own threat model. This demonstrator, supported by benchmark measurements, underscores the real-world implications of the identified weaknesses.
5. **Analysis of Matter's Threat and Countermeasures List.** We assess the Threats and Countermeasures list provided in the Matter Core Specification, evaluating the (in)sufficiency of the proposed mitigations against identified vulnerabilities. Our analysis reveals gaps in Matter's security assumptions, showing that several listed countermeasures are insufficient in practice, and our findings directly contradict several high-assurance claims made by Matter.
6. **Automated Formal Analysis of Matter Protocols.** Using ProVerif, we conduct the first formal analysis of the security properties of PASE and CASE. We additionally provide generic models for SPAKE2+ and the SIGMA variant used by Matter (SIGMA-I). These formal models validate our findings, allow automatic reproduction of our results, and serve as a foundation for researchers who wish to carry our results further.

1.3 Responsible Disclosure

We adhered to responsible disclosure practices in line with established norms in security research. On 11 October 2024, we contacted the Connectivity Standards Alliance (Alliance), the governing body behind Matter, to disclose our findings related to multiple security vulnerabilities in the Matter protocol. The Alliance formally acknowledged receipt of our report on 15 October 2024, and we agreed on a 150-day disclosure window to provide time for internal review and remediation.

Over the following months, we engaged in several rounds of discussions with the Alliance, including detailed email exchanges and online meetings with their engineering, security and top management teams. During these sessions, we provided full technical justifications for our findings and outlined potential cryptographic improvements. Specifically, we recommended replacing PBKDF2 with memory-hard alternatives such as Argon2, enforcing unique salt generation per session, and increasing the entropy of commissioning passcodes. The Alliance confirmed the feasibility of several proposals, including the move to alphanumeric codes for passcodes and the potential adoption of stronger key derivation functions.

In addition to our protocol-level analysis, we identified critical implementation flaws in the certified open-source JavaScript Matter SDK (*matter.js*). The issues were promptly acknowledged, patched, and merged into the main codebase.

Importantly, our research is based entirely on the Matter Core Specification documents [13] (versions 1.3 and 1.4), a comprehensive ∼1100-page draft, and the official Matter SDK [2], both of which are publicly available. We did not reverse-engineer any proprietary firmware, nor did we target commercial devices in ways that would violate legal or ethical boundaries.

Our intent throughout this project has been to identify and responsibly report weaknesses that could be exploited in practice, with the ultimate goal of strengthening the security posture of Matter as a widely deployed IoT standard.

1.4 Artefacts

An artefact bundle is provided at `https://github.com/KULeuven-COSIC/whats-the-matter`. It includes source code demonstrating our attacks, benchmarks, our documented formal models, and the ProVerif outputs.

2 Foundations

2.1 Matter's Building Blocks

Matter's infrastructure comprises the following core components [13]:

Devices and Nodes: A *device* is a physical unit, such as a lightbulb or thermostat. Each device may host one or more *nodes*, which are logical, addressable entities with distinct functions. Upon commissioning, a node is assigned an Operational Node ID and runs the Matter protocol stack independently of other nodes on the same device. For instance, a smart hub (device) may contain multiple nodes, each managing separate network roles.

Fabrics: A *fabric* is a logically isolated, secure network segment in which Matter devices operate under a shared trust model. Devices can be part of multiple fabrics simultaneously, maintaining strict isolation while supporting cross-environment interoperability.

Controllers and Administrators: *Controllers* manage nodes, coordinate communication, and perform device operations within a fabric. *Administrators* define access control policies and

act as the root of trust, issuing certificates to devices. Upon joining a fabric, a device receives an operational certificate and stores the Administrator's root certificate. A key Matter feature is *multi-admin*. This feature allows devices to be a part of multiple fabrics, so that controllers from different ecosystems (e.g., Apple HomePod and Amazon Echo) can control the same device.

Commissioners: *Commissioners* onboard new devices into a fabric, verify credentials, and issue operational certificates. A commissioner may also act as a controller or administrator. During commissioning, the commissioner is implicitly granted administrative rights over the device.

2.2 Cryptographic Primitives in Matter

Matter uses a suite of cryptographic primitives aligned with NIST guidelines to guarantee security. These include random number generators such as AES-CTR-DRBG, SHA-256 and PBKDF2. We explore these primitives further in Appendix A. Additionally, Matter adapted protocols like SPAKE2+ [32] and SIGMA [19] for its key-establishment protocols PASE and CASE. A thorough analysis is provided in Sections 4 and 5.

2.3 Commissioning in Matter

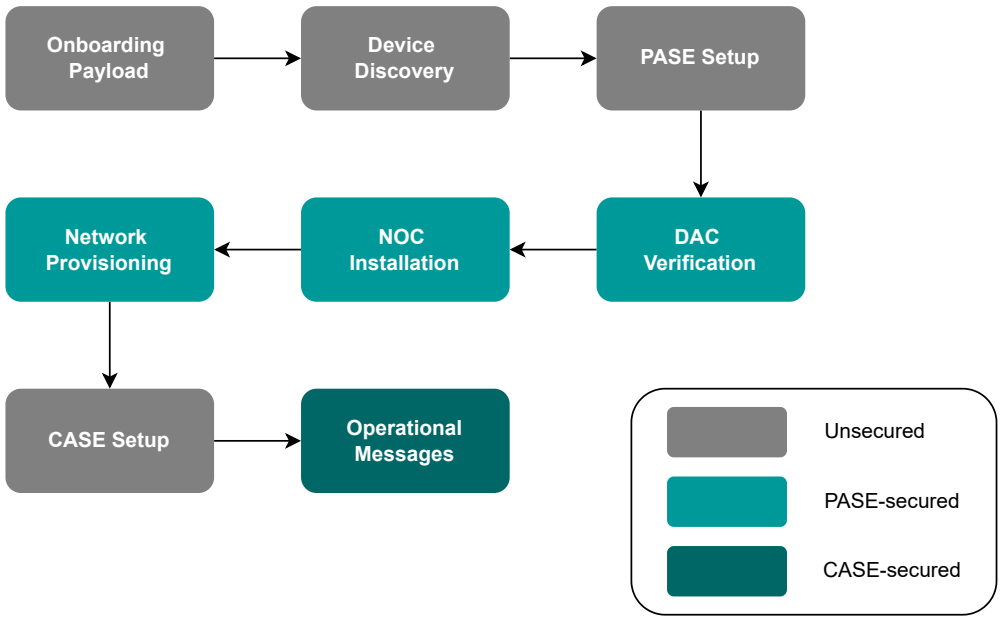


Fig. 1: Overview of the Matter commissioning process, which advances through escalating security levels: initial unsecured communication for PASE setup, PASE-secured exchange for DAC and NOC provisioning, and CASE-secured communication for operational use.

The commissioning process in Matter is a multi-phase, secure procedure for onboarding new devices (commissionees) into a fabric. As shown in Fig. 1, the process comprises the following steps:

Onboarding Payload: Stored on the commissionee and shared out-of-band with the commissioner (e.g., via QR code, NFC, or manual entry), it includes the device version, Vendor ID, Product ID, and an 8-digit setup passcode.

Passcode: The passcode acts as a shared secret for mutual authentication during commissioning. The commissioner proves knowledge of the passcode, while the commissionee proves possession of cryptographic values derived from it. Matter supports static and dynamic passcodes – the latter are session-specific and generated via a cryptographically secure pseudo-random number generator. However, since most IoT devices lack screens or speakers, static passcodes are almost always used.

Device Discovery and Initialisation: This process begins with the commissionee announcing its readiness to set up secure session. This is done typically using Bluetooth Low Energy (BLE) or IP-based discovery. A scanning commissioner can then detect the device, initiate communication, and confirm its readiness for secure onboarding.

PASE Session Establishment: Using the shared passcode, the commissioner initiates the PASE protocol to create a secure communication channel. Keys derived during this exchange protect all messages until we run the CASE protocol.

Timer Activation: Upon completing PASE, the commissionee arms the fail-safe timer, a security feature that limits the commissioning window to 60 seconds.

Attestation and Validation: The commissioner authenticates the device by verifying its Device Attestation Certificate (DAC) and Product Attestation Intermediate (PAI) certificate, both issued by trusted certificate authorities. This step ensures that only certified, legitimate devices can join the fabric, protecting against compromised or unauthorised devices.

Operational Certificate Assignment: After successful attestation, the device generates an operational key pair and sends a Certificate Signing Request (CSR) to the commissioner, who forwards it to the root certificate authority to obtain a Node Operational Certificate (NOC). The NOC, returned to the device, acts as its cryptographic identity within the fabric and is essential for communicating with other devices in the network. More on NOCs in Appendix C.

Network Provisioning: The commissioner configures the device's network parameters (e.g. IP addresses, Wi-Fi credentials) and instructs it to join the target network.

CASE Session Establishment: With the NOC installed, the commissioner and commissionee use their operational certificates to establish a CASE channel. This ensures mutual authentication and secures all subsequent communication within the fabric through cryptographic guarantees of confidentiality and integrity. Once complete, the device is fully integrated, the fail-safe timer is disarmed, and it can now securely interact with other Matter-enabled nodes.

Notably, a device's passcode is deactivated after successful commissioning (unless a factory reset is performed). The initial commissioner deletes the verification data, preventing reuse of the original QR code. Any future commissioning must use a new pairing code generated by the existing controller – displayed on-screen or provided via audio prompt, depending on the device's capabilities.

2.4 PKI and Access Control in Matter

Matter's Public Key Infrastructure (PKI) underpins secure device authentication and communication by managing device identities through digital certificates. This infrastructure ensures trusted interactions between devices within the same fabric. In parallel, Matter employs an access control framework to regulate device authorisation. Each device holds an Access Control List (ACL) that

defines which entities are allowed to access specific functions. These topics are discussed further in Appendices B and F.

3 Threat Model and Security Objectives

Matter aims to establish secure communication channels that ensure confidentiality, authentication, and protection against unauthorised access or impersonation. These guarantees rely on the secrecy of setup passcodes and derived keys during commissioning.

In this work, we study the ability of these key-establishment protocols to create secure channels that guarantee Matter's stated goals. For that, we focus on threats to the device pairing process and overall communication security by considering an adversarial model. Our threat model strictly follows the adversarial capabilities defined in the Matter specification and its "Threats and Countermeasures" list. In particular, we assume that protocol participants follow either the specification or one of the officially certified SDK implementations. This enables us to evaluate security both in theory and in practice, by aligning our model with the behaviour expected in real-world deployments.

As a result, our threat model considers protocol participants and external network adversaries. The participants can be honest, honest-but-curious, or malicious. For each protocol execution, both honest and honest-but-curious participants follow the protocol as specified without deviation. The honest-but-curious participant may at any point attempt to derive additional knowledge from these honest executions. In contrast, a malicious participant may decide when to follow the protocol execution and when to deviate by sending malicious messages in an attempt to gain some advantage.

Our external network adversary, whether active or passive, may intercept and record protocol execution transcripts, as well as any ciphertexts transmitted after a protocol's execution. In an attempt to gain an advantage, the passive network adversary may only manipulate these transcripts post-protocol-execution and will never inject or interfere with any message over the network. In contrast, an active adversary may inject messages or drop them before they reach their intended recipient.

These capabilities allow the network adversary to attempt various attacks based on observed data without requiring direct physical access to the device. Thus, we exclude the external attacker's capability of physically tampering with or extracting keys or data from devices.

We later show scenarios where, with these assumptions, an attacker may carry out attacks to recover the protocol's setup passcodes with any off-the-shelf computer. Such attacks can be trivially parallelised if the attacker has access to GPUs or cloud infrastructure to obtain the results instantly.

4 SPAKE2+/PASE Analysis

The initial session establishment in Matter happens via the Passcode-Authenticated Session Establishment (PASE) protocol. PASE is based on SPAKE2+, a Password-Authenticated Key Exchange (PAKE) protocol used to derive a strong shared secret between two parties from a low-entropy password [32]. Since SPAKE2+ is an *augmented* PAKE protocol, only one party needs to have knowledge of the password. In the case of Matter, the two parties are a commissioner and commissionee.

In the case of static passcodes, which applies to the overwhelming majority of currently deployed Matter devices, this 8-digit setup passcode is generated externally during the manufacturing process.

In most cases, a QR code encoding the passcode and additional metadata is printed and provided with the device. The device itself does not store the passcode but retains only key material derived from it, which is used during PASE.

During commissioning, the commissioner obtains the passcode it needs to initiate PASE out-of-band, typically by scanning the QR code.

Although Matter's PASE is based on the second draft of SPAKE2+ [31], its implementation introduces several deviations from the original protocol. In the following subsections, we first outline the SPAKE2+ protocol, then examine its integration within Matter. Finally, we identify key differences between the two and explain how these deviations weaken the security of the protocol in Matter's context.

4.1 SPAKE2+

SPAKE2+ is an augmented PAKE protocol designed for secure password-based authentication [32]. It enables two parties, the *prover* and the *verifier*, to establish a strong shared secret based on a password without revealing the password itself. The augmented nature of SPAKE2+ means that only one party, the prover, directly uses the password. The verifier, on the other hand, only has access to a record derived from the password, created during an initial offline registration phase. Since Matter's implementation is based on the second draft of this protocol, we discuss the protocol design as specified in that draft.

The protocol relies on several key elements. We provide a brief summary:

Group G Cyclic group in which the Computational Diffie-Hellman (CDH) problem is hard. G has order $p*h$, where p is a large prime and h is the cofactor.

Generator P Generator of the subgroup of order p, used to generate group elements during the key exchange process.

Identities The identities of the prover, A, and the verifier, B, are used in the key derivation process to bind the derived keys to the participants.

SPAKE2+ consists of two main phases: an offline registration phase and an online authentication phase. During the offline phase, the prover derives, from the password and the identities, an integer value via a PBKDF[§]. This integer is then split into two halves, *w0s* and *w1s*, which are each reduced modulo p, resulting in values *w0* and *w1*.

The prover shares *w0* and $L = w1*P$ with the verifier. It is important to note that the verifier does not possess, and therefore does not store, the value *w1* in its "registration record." The prover computes,

$$w0s \parallel w1s = PBKDF(pw \parallel A \parallel B, salt, iteration\_count)$$
$$w0 = w0s \bmod p$$
$$w1 = w1s \bmod p$$

Next, in the online phase, two public group elements of unknown discrete logarithm M, $N \in$ the prime-order subgroup of G are generated. These elements may be pre-computed and re-used. The specification of SPAKE2+ suggests pre-computed values, which Matter adopts in its implementation.

[§]In reality, the length of each item is also incorporated into the PBKDF input, but we have omitted these details to simplify the explanation.

Messages are then exchanged to compute the shared secret, as illustrated in Fig. 2. The steps are detailed below:

1. The prover initiates the protocol by:
   (a) Generating a random integer x from [0, p-1].
   (b) Computing the ephemeral public value pA, with $pA = xP + w0*M$.
   (c) Sending pA to the verifier.
2. The verifier, upon receiving pA, performs the following:
   (a) Checks if pA is a member of the large prime-order subgroup of G. If not, the verifier aborts.
   (b) Generates a random integer y from [0, p-1].
   (c) Computes the ephemeral public share pB, with $pB = yP + w0*N$.
   (d) Sends pB to the prover.
   (e) Computes the common values:
   $Z = h*y*(pA - w0*M)$
   $V = h*y*L$.
3. The prover, upon receiving pB, does the following:
   (a) Checks pB for group membership.
   (b) Computes the common values:
   $Z = h*x*(pB - w0*N)$
   $V = h*w1*(pB - w0*N)$.

After public value exchange and computation of the shared values, both parties derive a shared secret Ke from the protocol transcript TT, defined as follows[¶]:

$$\begin{aligned} TT = {} & len(A) \parallel A \parallel len(B) \parallel B \parallel len(M) \parallel M \\ & \parallel len(N) \parallel N \parallel len(pA) \parallel pA \parallel len(pB) \parallel pB \\ & \parallel len(Z) \parallel Z \parallel len(V) \parallel V \parallel len(w0) \parallel w0 \end{aligned}$$

The secrets Ka and Ke are derived as follows:

$$Ka \parallel Ke = Hash(TT)$$

Each key is half the length of the hash output. From Ka, confirmation keys KcA and KcB are derived using a KDF:

$$KcA \parallel KcB = KDF(nil, Ka, \text{"ConfirmationKeys"})$$

The confirmation keys are used to generate and verify key confirmation messages, ensuring both parties agree on the derived secrets before proceeding. Specifically, the prover sends $cA = MAC(KcA, pB)$ and the verifier sends $cB = MAC(KcB, pA)$[‖]. Both sides then check whether the expected MAC value matches the received one. This ensures both parties derived the same shared secret, as the MAC binds confirmation to the exchanged ephemeral values (pA and pB).

Once keys are confirmed, Ke becomes the authenticated shared secret for all future encryption. It is important to note that Ka and its derived keys serve no purpose beyond key confirmation and should be securely erased from memory.

[¶]The SPAKE2+ specification incorporates a *Context* component (such as the application name or version) into the calculation of TT. We omit this detail for clarity.

[‖]In the SPAKE2+ specification, a fixed string may be used instead of pB and pA.

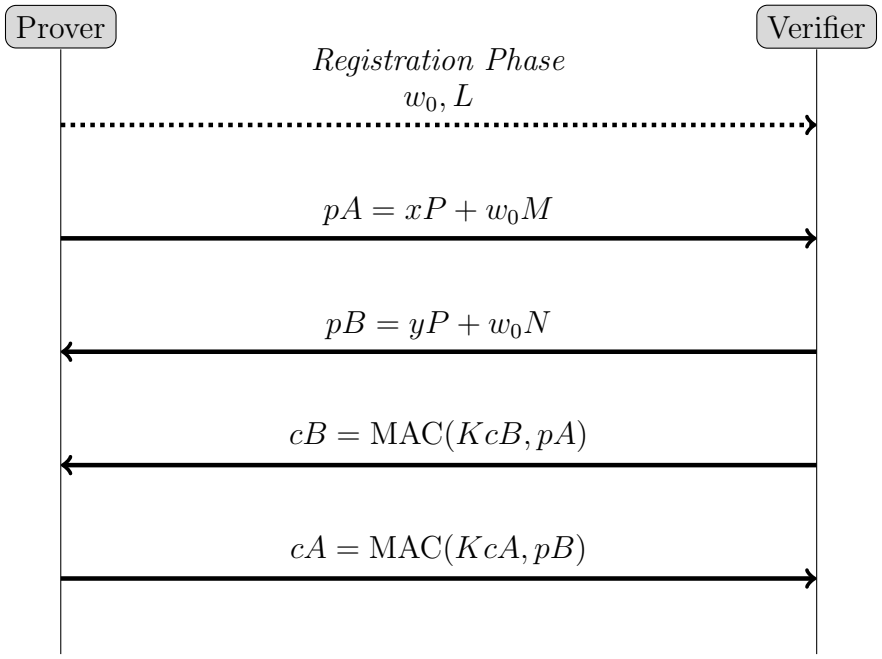


Fig. 2: Message sequence chart of the SPAKE2+ protocol. The offline registration phase, during which the necessary common values (such as *w0* and L) are generated and shared, is represented with dotted lines.

4.2 PASE

In the Matter protocol, PASE is implemented based on the SPAKE2+ protocol [31]. The prover here is the commissioner, and the verifier is the commissionee. It is assumed that the commissioner has obtained the passcode through some out-of-band method (such as scanning a QR code), and that the commissionee has the corresponding *(w0, L)* pair pre-installed on the device prior to protocol initiation.

The flow of Matter's PASE is summarised below and illustrated in Fig. 3.

1. **PBKDF Parameters Exchange**
   (a) The commissioner sends a *PBKDFParamRequest* message to request PBKDF parameters from the commissionee.
   (b) The commissionee responds with a *PBKDFParamResponse* message containing those parameters.
2. **SPAKE2+ Messages** This step mirrors the original SPAKE2+ message flow, with protocol-specific deviations discussed in the next subsection.
3. **Key Derivation** Both parties derive the session key from the shared secret computed during the key exchange.

4.3 Differences between SPAKE2+ and PASE

There are several significant differences between Matter's SPAKE2+ implementation and the official SPAKE2+ specification. Firstly, Matter is based on the second draft of the SPAKE2+ protocol, rather than its final published version. The second draft contains different terminology as well as variations in key confirmation and derivation processes. These are further detailed in Appendix D. This subsection outlines all the differences we have identified between the official SPAKE2+ specification and Matter's implementation.

PBKDF Parameters Sharing Matter introduces additional messages to exchange PBKDF parameters. This ensures that both parties agree on the same inputs for key derivation, which is critical to producing consistent and secure keys.

Session ID and Random Number Each message in PASE includes a session ID and a random number. The session ID uniquely identifies the session, while the random number provides resistance to replay attacks.

Lack of Group Membership Checks The SPAKE2+ specification mandates checking that all elements are part of the designated group to prevent subgroup confinement attacks. These attacks occur when an adversary forces a key exchange into a smaller subgroup of the elliptic curve group, potentially leaking partial private key information [22], thereby weakening the protocol's security. We examined both the Matter specification and its publicly available SDK, but found no indication that these checks are performed. We recommend explicitly including such checks, as required by the SPAKE2+ specification. Upon contacting the Alliance, they acknowledged this omission and stated that they would investigate further.

Final Keys From the shared secret *Ke*, Matter derives two session keys: *I2RKey*, used by the initiator to encrypt and authenticate messages (and by the responder to decrypt and verify them), and *R2IKey* for the reverse direction.

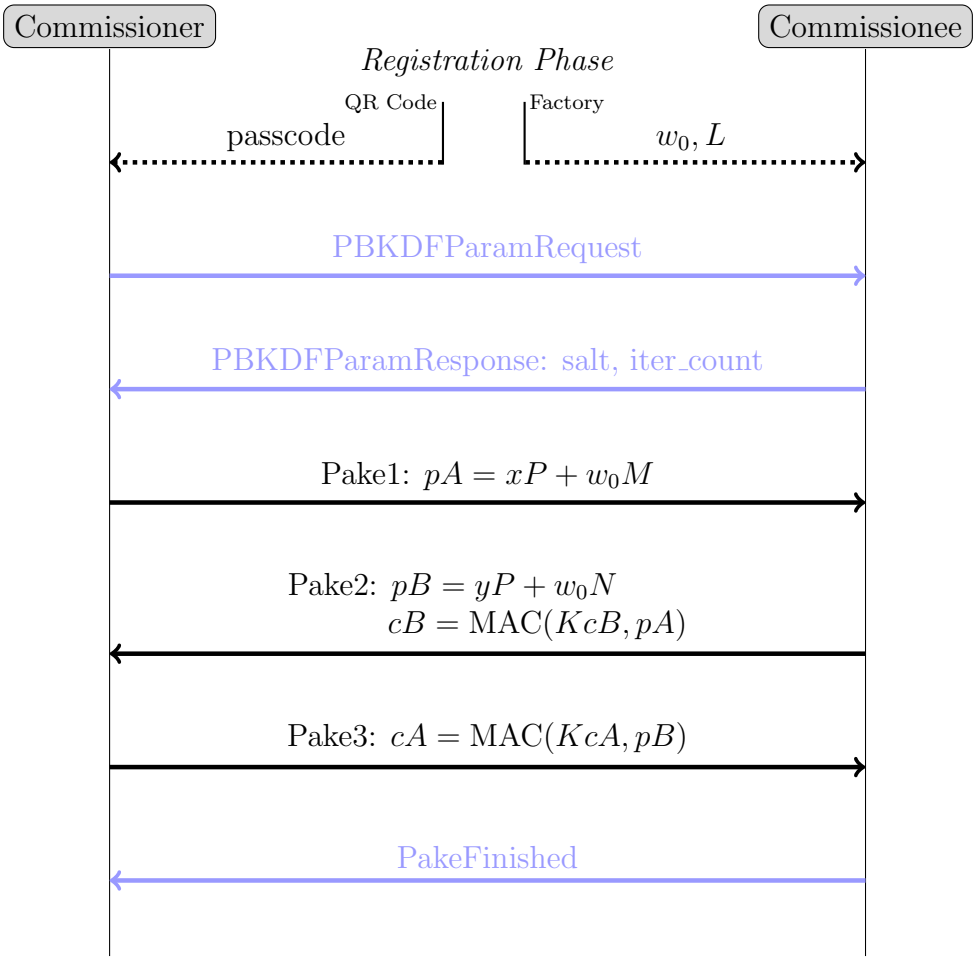


Fig. 3: Message sequence chart of Matter's PASE. The offline registration process is represented with dotted lines. Messages related to the PBKDF parameters and the final status report are highlighted in purple, as they are not part of the original SPAKE2+ specification. Matter combines pB and cB into a single message, which, as noted in the SPAKE2+ specification, does not introduce security concerns.

4.4 Security of PASE

Matter recommends using a number of iterations for PBKDF between 1,000 and 100,000. This range is significantly lower than the currently recommended minimum of 600,000 iterations for *PBKDF2-HMAC-SHA256* [1]. Fewer iterations reduce the computational effort required for brute-force attacks, weakening the key derivation process.

Furthermore, the passcode is an 8-digit random number, resulting in only 10^8 (100 million) possible combinations. While we show in Appendix G that this entropy is adequate to withstand brute-force attacks under ideal conditions, it remains a concern when enforcement mechanisms are missing or bypassed. We recommend increasing the passcode entropy to reduce dependence on rate-limiting and strengthen resilience against wrongly certified implementations. For instance, Matter attempts to mitigate brute-force attacks through countermeasure *CM100*, which states: "Device exits commissioning mode after 20 failed commissioning attempts." This mitigation is cited explicitly for threats *T101* and *T112*, which concern online brute-force attacks on the setup passcode. However, we discovered a critical security vulnerability in the official JavaScript SDK of the Matter codebase[**]. This SDK is explicitly marked as compliant with Matter version 1.4 in the Matter Handbook[††], and has successfully passed Matter certification. It is also part of the official Project CHIP repository[‡‡]. In our experiments, we found that the 20-attempt limit is not effectively enforced [§§] . Although the SDK logs the message "Maximum number of PASE pairing errors reached, canceling commissioning" once the threshold is reached, it continues to accept and process new pairing requests. The internal error counter even continues incrementing beyond the threshold (e.g., 21/20, 22/20, etc.), effectively nullifying the intended protection mechanism. Moreover, Matter's recommended 60-second commissioning window was also not enforced in the SDK, allowing an attacker to terminate failed attempts even earlier (e.g., without having to wait for the final status report from the device). We responsibly disclosed these vulnerabilities to the developers, who took immediate action to patch it.

Another critical vulnerability relates to handling concurrent PASE session requests. The SDK did not reject new PASE session attempts during an ongoing session establishment [¶¶] . This violated the specification, which states: "When a Commissioner is either in the process of establishing a PASE session with the Commissionee or has successfully established a session, the Commissionee shall not accept any more requests for new PASE sessions until session establishment fails or the successfully established PASE session is terminated on the commissioning channel." Failure to enforce this requirement introduces a serious security vulnerability. It enables an attacker to passively wait for a legitimate user to initiate commissioning on the device, then interrupt the process and take over the session with its own controller – without any form of physical access to the device. This directly undermines Matter's threat model, particularly Threat *T15*: "Commission an uncommissioned Node without physical access to Device," and its countermeasure *CM3*, which mandates "Commissioning is started with some form of physical user interaction (e.g. power cycle or button press)." We also disclosed this issue to the developers, who received the report and patched the SDK accordingly.

[**] `https://github.com/project-chip/matter.js/`
[††] `https://handbook.buildwithmatter.com/development/`
[‡‡] `https://github.com/project-chip`
[§§] `https://github.com/project-chip/matter.js/issues/1894`
[¶¶] `https://github.com/project-chip/matter.js/issues/1896`

As discussed in Sect. 2.3, most devices use a static passcode, meaning a single PASE verifier *(w0, L)* is used. This implies a static salt, retrievable by sending a *PBKDFParamRequest* message to the commissionee. This raises concerns about the suitability and mismatch of SPAKE2+ in Matter. SPAKE2+ was designed for a multi-client, single-server model where the server is protected against password file compromises [10], whereas in Matter, a single password is shared by all users.

It is also unclear why the commissioner is required to apply a PBKDF to the passcode. An attacker with the passcode can apply the same PBKDF to derive *w0* and *w1*. Why does Matter not include *w0* and *w1* directly in the QR code? One rationale could be to keep the manual passcode entry short when no QR is available. This interpretation is supported by the Matter specification: "For example, a device may not require the *Crypto_PBKDF()* primitive, as values based on this operation could in some instances be pre-computed and stored during the manufacturing process of the device."

In Sect. 6, we present attacks that exploit the weaknesses discussed in this section. Our proposed recommendations, along with the responses we received from the Matter team, are covered in Sect. 9.

5 SIGMA/CASE Analysis

After successful commissioning, the Certificate-Authenticated Session Establishment (CASE) protocol is used to provide mutual authentication between peer nodes and to generate cryptographic keys securing subsequent communications within the session.

The CASE protocol in Matter is based on SIGMA[***], a family of key-exchange protocols designed for secure, authenticated Diffie-Hellman key exchange using a combination of digital signatures and message authentication codes (MACs). SIGMA stands for "SIGn-and-MAc", reflecting its use of both cryptographic primitives to achieve its security goals. In Matter, the protocol is used to:

1. Generate a shared secret by exchanging ephemeral elliptic curve public keys.
2. Provide identities by exchanging the NOC certificates.
3. Prove possession of the NOC private key by signing both the ephemeral keys and the NOC certificate.

The SIGMA protocols are notable for balancing strong security guarantees with practical requirements, including optional identity protection and a minimal number of protocol rounds. They form the cryptographic backbone of Internet Key Exchange (IKE), the standardised key-exchange protocol for IPsec [19].

5.1 SIGMA Protocols

The SIGMA family of key-exchange protocols is designed to provide certificate-based security [19]. It supports scenarios where identity protection may or may not be required. SIGMA protocols achieve this using a combination of Diffie-Hellman exchanges authenticated by digital signatures and MACs. Several variants exist, each suited to different threat models. The basic SIGMA protocol

[***] While the Matter specification refers to its protocol as mirroring *the* SIGMA protocol, this is technically incorrect, as SIGMA is a family of protocols [19]. Throughout this paper, when we refer to *SIGMA*, we specifically mean the variant employed by Matter. The details of this variant are discussed further in the paper.

does not offer identity protection, while SIGMA-I and SIGMA-R provide identity protection for the initiator and responder, respectively.

We focus on the SIGMA-I variant, which is particularly well-suited for scenarios requiring protection of the initiator's identity against active attackers, and the responder's identity from passive attackers. For example, when a client (initiator) connects to a server (responder), it may be more important to conceal the user's identity than that of the server.

Fig. 4 illustrates the protocol flow of SIGMA-I. In this figure:

- g^x and g^y are the Diffie-Hellman public values.
- $\text{Sig}_B(g^x, g^y)$ and $\text{Sig}_A(g^y, g^x)$ are digital signatures on the exchanged values.
- $\text{MAC}_{K_m}(\text{B})$ and $\text{MAC}_{K_m}(\text{A})$ are the MACs computed over the sender's identity using a MAC key K_m derived from g^{xy}.
- $\{...\}_{K_e}$ denotes encryption using a key K_e, also derived from g^{xy}.

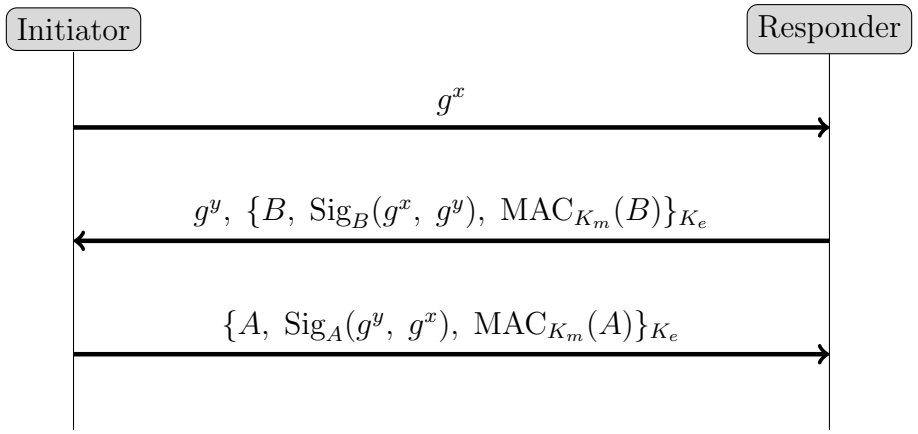


Fig. 4: Message sequence chart of the SIGMA-I protocol. The initiator here is the commissioner, whose identity is protected from active attackers.

The main components of the protocol are the following:

Diffie-Hellman Exchange: Each party generates an ephemeral Diffie-Hellman key pair and exchanges the public values, g^x and g^y. This ensures perfect forward secrecy.
Signatures and MACs: Both parties sign the exchanged Diffie-Hellman values to ensure integrity and authenticity. Including the peer's DH value in the signed message also prevents replay attacks [19]. MACs are computed over each party's identity using a key derived from the shared secret g^{xy}. These MACs serve as "proof of possession" of the session key and bind each identity to the key, preventing identity misbinding attacks – where an attacker relays messages between honest parties to make them believe they are talking to each other.
Encryption for Identity Protection: To provide identity protection, the identities and signatures are encrypted using a key derived from the shared Diffie-Hellman secret g^{xy}.

5.2 CASE

Once the PASE protocol is completed, a Node Operational Certificate (NOC) is installed on the commissionee (see Appendix C). Using the installed NOC, Matter employs CASE to enable nodes within a fabric to mutually authenticate and encrypt all subsequent messages. Although not explicitly stated in the Matter specifications, CASE is based on the SIGMA-I variant of the SIGMA protocols. In particular, the commissioner only reveals its identity to the commissionee after verifying the latter's identity. The message flow in CASE closely follows SIGMA-I, with minor differences – such as an additional status report message at the end. These are discussed in the next subsection.

Fig. 5 illustrates the protocol flow of the Certificate Authenticated Session Establishment (CASE) handshake. In this figure:

- g^x and g^y are the ephemeral Diffie-Hellman public keys.
- $\mathrm{Sig}_R(g^x, g^y, \mathrm{NOC}_R)$ and $\mathrm{Sig}_I(g^y, g^x, \mathrm{NOC}_I)$ are digital signatures over the handshake transcript and the sender's operational certificate.
- $\{\cdots\}_{S_{2K}}$ and $\{\cdots\}_{S_{3K}}$ denote authenticated encryption with keys derived from the shared secret.
- $\mathrm{Tag}_{S_{2K}}$ and $\mathrm{Tag}_{S_{3K}}$ are AEAD authentication tags appended to the encrypted blobs.

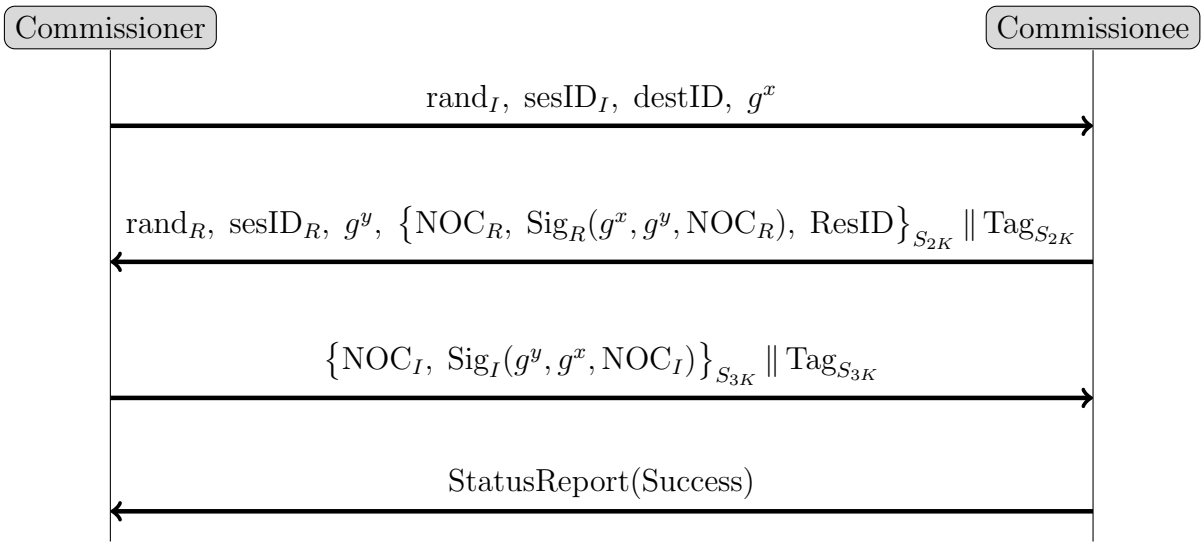


Fig. 5: Message sequence chart of the CASE protocol.

Session Resumption Matter also employs a mechanism called *session resumption*, allowing rapid re-establishment of previously secured sessions without requiring a full CASE execution. In resumed sessions, signature creation and verification are omitted. This significantly reduces computational overhead – especially important for resource-constrained IoT devices.

During the initial CASE session, a resumption ID is generated and exchanged between the initiator and the responder. This ID, along with keys derived during session setup, is stored locally by both parties. To resume a session, the initiator sends a resumption request along with the resumption ID. The responder checks whether this ID matches a stored session; if so, both parties use the previously derived keys to quickly re-establish a secure channel using a pre-negotiated shared secret.

5.3 Differences between SIGMA-I and CASE

A key difference in Matter's implementation is that the MAC does not solely cover identity information (e.g., certificates), but also includes additional data such as the ephemeral public keys and the signature itself. Effectively, the MAC is computed over the entire signed message, resulting in a structure resembling $\mathrm{MAC}_{K_m}(\mathrm{Sig}_B(g^x, g^y, B))$. In addition, the responder sends a resumption ID, which allows both parties to resume the session later without repeating the full handshake. The initial message also includes a destination identifier, which tells the responder which fabric the initiator has selected.

We also observed the following additional differences between Matter's implementation and SIGMA-I:

1. **Random Number Generation:**
   - **SIGMA:** The original SIGMA protocol does not involve nonce generation by the initiator or responder.
   - **Matter:** Both the initiator and responder generate random values (*initiatorRandom* and *responderRandom*) to ensure session uniqueness.
2. **Session ID Generation:**
   - **SIGMA:** Session IDs are not explicitly generated or exchanged.
   - **Matter:** Each side generates a session ID (*initiatorSessionId* and *responderSessionId*), which facilitates identification and management of individual sessions.
3. **S2K and S3K Key Derivation:**
   - **SIGMA:** No specific keys are derived for encrypting individual messages.
   - **Matter:** S2K and S3K keys are derived to encrypt the second and third messages, respectively.
4. **MACing and Identity Verification:**
   - **SIGMA:** Uses separate keys derived from the shared Diffie-Hellman secret for MAC and encryption.
   - **Matter:** Employs *Authenticated Encryption with Associated Data (AEAD)* to simultaneously provide confidentiality and integrity, rather than using separate primitives.

5.4 Security of CASE

As discussed in the previous subsection, Matter's implementation MACs not only the identity but also the ephemeral public keys and the signature. The security implications of this design are not explicitly addressed in the original SIGMA paper. The paper states that the MAC moved under the signature may cover just the identity of the sender or the whole signed information. For example, in B's message, the pair $(\mathrm{Sig}_B(g^x, g^y), \mathrm{MAC}_{K_m}(B))$ is replaced with either $\mathrm{Sig}_B(g^x, g^y, \mathrm{MAC}_{K_m}(B))$ or $\mathrm{Sig}_B(\mathrm{MAC}_{K_m}(g^x, g^y, B))$. This method saves space for an extra MAC outside the signature, and

the verification of the MAC is merged with that of the signature. In either case, as long as the MAC covers the identity of the signer, the same security of the SIGMA protocols is maintained [19].

It is not mentioned in specification why Matter chose to implement SIGMA-I instead of SIGMA-R. We believe that this choice is most likely based on the following reasoning: a controller or commissioner typically holds more privileges within the network than a commissionee. Compromising the controller can lead to more significant security breaches, given its role in managing and controlling various devices within the Matter network. That is why protecting the identity of the controller can have more benefits. However, the Matter specification highlights a significant concern: "Since there are no trust mechanisms employed for commissioners advertising themselves, commissionees may provide commissioner selection choices to the User that are from malicious entities masquerading as commissioners". This means that a malicious commissioner could potentially access the identities of all devices through active attacks, since the identities of the commissionees are not protected by the SIGMA-I variant.

Finally, our formal analysis results in Sect. 8 reveal a major concern: the session resumption mechanism significantly downgrades the security of the session compared to a standard CASE execution. The specific reasons for this are detailed in that section.

6 Attacks on Matter's Commissioning Protocols

We present a set of attacks on the commissioning process in the Matter protocol, each targeting specific weaknesses in its cryptographic design, parameter choices, and enforcement of trust assumptions. While some attacks may be challenging to mount in practice, all of them expose critical flaws, either in the protocol specification or in officially certified implementations, that contradict Matter's stated security objectives. One such example is a brute-force attack on the passcode, which becomes more practical considering the SDK vulnerabilities discovered in Sect. 4.4. We provide supporting benchmarks for this attack in Appendix G, and describe the rest of the attacks in more depth in this section.

6.1 Pre-Computation Attack

The pre-computation attack targets the cryptographic parameters used in Matter's PASE session. Specifically, the attacker leverages the fact that the salt and certain cryptographic values (e.g., *w0* and L) are hardcoded in the device. By pre-computing a large table of possible passcodes and their corresponding cryptographic outputs offline, the attacker can efficiently recover the correct passcode during the online phase of the attack.

The pre-computation phase involves computing the possible outputs for every potential passcode. Once the table is constructed, we can look up a specific *w0* value to find the corresponding passcode.

Obtaining *w0* is feasible through a post-compromise scenario. If an initially honest commissioner later turns malicious, it can exploit its access to *w0* obtained during the initial commissioning. Importantly, we do not make a strong assumption that the attacker has access to the actual passcode. We only assume that it has access to *w0*.

Matter claims in its high-severity threat *T102* that such a pre-computation attack, where an attacker has access to *w0* and L, is not possible. We demonstrate that it is, even under a weaker assumption: the attack can be executed using only *w0*. This highlights that Matter's existing countermeasures are insufficient to prevent these types of attacks.

Assumptions

- The attacker has access to *w0*.
- The salt used in the PBKDF is known to the attacker. This is not a hard assumption at all since the PBKDF parameters (salt and number of iterations) can be obtained by the attacker via a simple, unencrypted request to the device.

Exploited Vulnerability The use of a hardcoded salt in the PBKDF function. Because the salt is static, the exact same *w0* value from the initial commissioning can be reused to recover the passcode for another commissioning attempt. This enables an attacker to leverage a pre-computed table, as the cryptographic parameters remain unchanged.

Performance We conducted benchmarks for PBKDF2-HMAC-SHA256 (which produces the pair $(w0, w1)$ in PASE) to evaluate the computation cost of the attack. Our results confirm that, while building the rainbow table can take several hours, the lookup performance is indeed efficient enough to enable such an attack in practice. All measurements were taken on a MacBook Air with the following specifications: *Apple M2 Chip, 16 GB RAM, macOS Sequoia 15.3.2*. Every data point in the measurements is the median of 100 consecutive evaluations, reported together with its 95% confidence interval.

Table 1: Time to derive a single (w_0, w_1) pair for various PBKDF2 iteration counts.

Iterations	**Median time**	**95 % CI**
1 000	361.34 µs	[361.14 µs, 361.58 µs]
10 000	3.60 ms	[3.60 ms, 3.61 ms]
100 000	35.95 ms	[35.91 ms, 35.99 ms]
600 000	214.24 ms	[213.77 ms, 214.69 ms]
1 000 000	353.72 ms	[353.02 ms, 354.44 ms]

For an eight-digit passcode space of size $N = 99\,999\,987$ (excluding the invalid passcodes), we obtain the total table build times T_1 on a single core listed in Table 2.

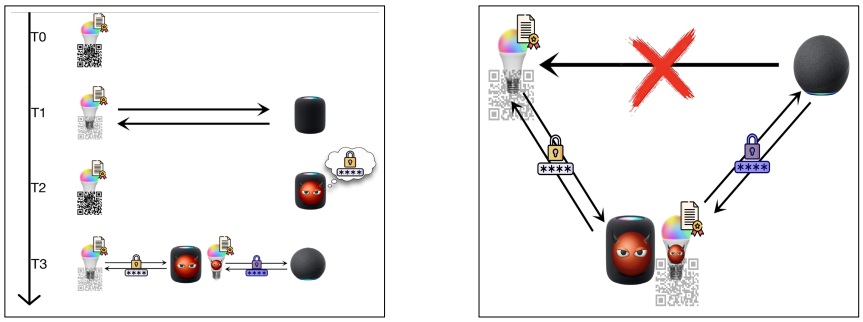
To illustrate how little clock time an attacker needs with modest parallelism, we also report $T_{128} = \frac{T_1}{128}$. This assumes ideal linear scaling across 128 identical CPU cores, which is readily available on cloud servers. The numbers show that the minimum iteration count allowed by Matter (1,000) yields a full rainbow table in about ten hours on one core, or just five minutes on 128 cores. Even the maximum permitted iteration count by Matter (100,000) can be pre-computed in eight hours on such a multi-core setup.

While generating the table may require significant time, once built, the lookups are fast. We measured the time required to look up a w_0 value in the pre-computed rainbow table. Using a naive linear search implementation, the median lookup time was approximately 33s with a 95% confidence interval of $[3s, 42s]$ This represents a worst-case scenario without any optimisations. With standard techniques such as binary search or hash maps – and on a more powerful CPU – lookup times could be reduced to the order of milliseconds.

Table 2: Estimated offline time needed to pre-compute a rainbow table for the 8-digit pass-code space at different PBKDF2 iteration counts. T_1 is the wall-clock time on a single CPU core; T_{128} denotes scaling across 128 cores.

Iterations	T_1	$T_{128} = T_1/128$
1,000	10 h	5 min
10,000	100 h	47 min
100,000	42 d	8 h
600,000	251 d	47 h
1,000,000	414 d	3 d

Outcome If the pre-computation attack is successful, the attacker can recover the passcode by quickly matching the observed cryptographic output with the values in the pre-computed table. This gives the attacker full control over the device, including accessing sensitive device data as described in Matter's threat list. As we have highlighted in this subsection, this attack is feasible given Matter's threat model, and protections need to be implemented to mitigate it, as we discuss in Sect. 9.



(a) (b)

Fig. 6: (a) Overview of the Controller-in-the-Middle Attack, and (b) Detailed Message Relay in the attack. A faded QR code indicates that the QR code is inactive since the device is already commissioned. The locks are depicted in two different colours to represent the use of distinct passcodes.

6.2 Controller-in-the-Middle Attack

We designed a Controller-in-the-Middle (CitM) attack to demonstrate an exploit targeting the commissioning process of devices in Matter. The attack takes advantage of the fact that, after obtaining the passcode, a malicious commissioner can intercept and manipulate the commissioning process to insert themselves between the device and a legitimate controller. The attack is depicted in Fig. 6. The exact steps of the attack are described below.

- **T0**: A lightbulb with a QR code to be commissioned.
- **T1**: An honest commissioner retrieves the passcode.

– **T2**: The commissioner turns malicious but retains knowledge of the valid passcode. This step is critical as the attacker now has the necessary credentials to hijack future sessions.
– **T3**: When a new, honest commissioner tries to commission the device, the malicious commissioner intercepts the communication. The attacker acts as the actual Matter device with a fake certificate and completes the commissioning process with the new honest commissioner. At the same time, the attacker commissions with the real device using the valid passcode.

This process allows the malicious actor to control communication between the real device and the new commissioner, as well as insert themselves into the middle of any sensitive data exchanges.

Assumptions

– The attacker has prior knowledge of the passcode. If the attacker only possesses *w0*, then the attack described in Sect. 6.1 can be leveraged as a sub-attack to recover the passcode.
– The attacker can impersonate the real device by using a fake certificate, which must be generated by an entity trusted by Matter's root. This attack remains possible without user notification until the root certificate of that entity is marked as untrusted and the user receives the revocation update. Even in scenarios where a fake certificate cannot be obtained, the attack is still feasible because, as outlined in the specification, the commissioning process can proceed even if device attestation fails.

Exploited Vulnerability The protocol lacks a mechanism to verify the authenticity of the commissioner, allowing an attacker to insert themselves into the communication without detection.

Performance This attack is highly effective once the attacker has obtained the passcode. Once the attacker is in the middle, they can relay messages and data without disrupting the normal operation of the device.

Outcome A successful Controller-in-the-Middle attack allows the malicious actor to intercept and manipulate all communication between the device and the controller. The attacker can:

– Eavesdrop on sensitive information, such as network credentials or commands intended for the device.
– Relay commands from the fake device to the real device, maintaining control without detection.

This compromises user privacy and security, as the attacker gains unauthorised access to potentially sensitive data being transmitted, as shown in Fig. 6b.

7 Implementation - Reconstructing the Attacks

Our implementation of the brute-force and pre-computation attacks utilises the official `Matter SDK`[†††] repository. This SDK allows developers to commission Matter controllers and devices across various platforms (e.g., Linux and Android) using example applications provided in the repository.

[†††] `https://github.com/project-chip/connectedhomeip`

We opted to work with the "lighting-app"[‡‡‡] example, which simulates a Matter-enabled lighting device.

The attacks were developed and tested on a MacBook Air with the following specifications: *Apple M2 Chip, 16 GB RAM, macOS Sequoia 15.3.2*. Our implementation comprised three main scripts to demonstrate the feasibility of various attack vectors:

1. **Brute Force Attack** At a high level, our brute-force attack interacts with this example device by repeatedly initiating commissioning attempts using different passcodes (from 1 to 99,999,998 and excluding any invalid passcodes). As prescribed by the Matter specification, a device should reject further commissioning attempts after 20 incorrect passcode trials. To proceed with additional passcode attempts, we programmatically re-enable commissioning mode on the device after reaching the attempt limit. This process is then repeated for the next batch of passcodes until a success status report is returned.
2. **Pre-Computation Attack**
   (a) **Table Builder** This script constructs a pre-computed table of possible passcodes, alongside their corresponding *w0* and *w1* values, after obtaining the PBKDF parameters.
   (b) **Attack Simulation** This attack simulates a scenario where the attacker has access to specific cryptographic values (such as *w0*) and leverages the pre-computed table to quickly retrieve the passcode.

To validate our findings concerning salt reuse and PBKDF2 iteration counts, we conducted experiments on two Matter-certified devices: the TP-Link Tapo P100M (smart plug) and the Nous P3 (smart bulb). For each device type, we acquired multiple units to verify consistency in cryptographic parameters across different production batches. Our results indicate that both manufacturers use static but device-specific salts. However, the iteration count remained consistent across all tested units from the same manufacturer. Notably, Nous P3 devices used the minimum permissible iteration count of 1,000, while TP-Link Tapo P100M devices employed a higher but still relatively low count of 15,000. These findings underline the variability in cryptographic strength across devices and highlight the need for stricter enforcement of parameter recommendations.

8 Formal Analysis

We conduct a formal symbolic analysis of the protocols at hand using the symbolic verification tool *ProVerif* under the Dolev-Yao model, assuming primitives as perfect black boxes.

ProVerif accepts as input a protocol model written in the applied pi-calculus. In such a model, we specify the algorithms that each participant executes as *processes*. We can have an unbounded number of such processes be executed in parallel. Each process may communicate with other processes by posting a message on *channels* or by reading messages posted by other processes on them. These channels emulate real-world communication channels.

ProVerif is an automated analysis tool, which means that we can provide it with a formal model of our protocol, some queries, and then ask it to prove or disprove our queries. This analysis can operate under either a passive or an active attacker. We refer the curious reader to the *ProVerif* manual [8] which serves as an excellent reference to learn about the tool.

[‡‡‡] `https://github.com/project-chip/connectedhomeip/tree/master/examples/lighting-app`

8.1 Formal Analysis of Matter

In this work, we model and verify PASE and CASE, as well as the protocols they are based on: SPAKE2+ and SIGMA-I respectively. Our analysis assumes the default active attacker defined by *ProVerif*. The models and results can be found in our artefact package 1.4.

Unfortunately, Matter does not use standard formal definitions for cryptographic security. Using their protocols, they aim to establish fresh confidential keys for each execution in order to transmit messages that are confidential and authenticated post-execution. Our annotated models provide an in-depth line-by-line explanation of the protocol execution and the results. We use this section to summarize our findings.

SPAKE2+ and PASE We acknowledge that SPAKE2+ and PASE rely heavily on elliptic curve group arithmetic, and that our symbolic modelling of these operations does not provide a result as robust as a proof in the computational model. Nonetheless, for this scenario, we started by analysing SPAKE2+ on its own, then PASE. The analysis of these two protocols yielded the same security properties. Thus, our analysis confirms that the protocols satisfy the following properties:

- Authentication from the commissioner to the commissionee as an entity which knows the setup passcode.
- Confidentiality of keys and subsequently encrypted messages.
- Forward Secrecy in case the passcode is compromised.

One minor issue arises when the salt is fixed as discussed in 2.3 and 4.4. This allowed *ProVerif* to re-use the values which we bootstrap the PASE sessions with. However, the security of the protocol under the symbolic model is not affected by this issue. Choosing a dynamic salt prevents *ProVerif* from being able to crack the secret passcode. Either way, if a password were to leak after a session's establishment, an active attacker would not be able to decrypt any session message.

SIGMA-I and CASE *ProVerif* was unable to find any attacks when it came to CASE. It did satisfy Matter's confidentiality and mutual authentication requirements. The protocol also provides mutual identity hiding for both initiator and responder.

We attempted to stretch the potential of CASE as much as possible and came to the conclusion that an attacker would only be able to impact the protocol if we expand its capabilities. More precisely, peer identities would only leak if the attacker gains access to an Identity Protection Key (IPK) and if one of the ephemeral keys of the parties is revealed to the attacker. For the attacker to have access to the IPK, they would either need to already be a member of the same fabric as the target or receive valid real-time keys from a fabric insider. We also find that SIGMA-I provides the same properties as CASE, except for the mutual identity hiding property. In SIGMA-I, the identity of the initiator is hidden while that of the responder is not against an active attacker.

CASE with Resumption As described in Section 5.2, Matter specifies an extension to CASE which allows parties that already established a CASE session to efficiently re-negotiate fresh session keys without the need for digital signatures. Unfortunately, our findings reveal that a session established through this resumption protocol is much less secure than a session established through a standard CASE execution. The IPK in this case no longer acts as shield around the session keys. This means that the attacker, instead of having to obtain this value through membership in the

fabric or by proxy, only needs to obtain one of the ephemeral keys. Thankfully, a compromised CASE with Resumption session still does not leak the identities which are securely shared during the preceding CASE session. We would expect a resumption protocol to not degrade security, even through stretched assumptions, which justifies our concern in this situation.

Results So what do these results mean? Recall that a PASE session precedes a CASE session during the commissioning process. We would expect that a compromised PASE session would ultimately lead to a compromised CASE session. However, that is not always the case thanks to Matter's DAC C check safeguard. It would require a rogue certificate issuer for our controller in the middle attack to take place, otherwise executing this attack would lead to a warning on the end-user's side.

Aside from the resumption protocol, Matter's protocols are relatively well designed and our formal analysis supports the properties targeted by the authors. However, we believe there is still room for improvement.

9 Discussion and Mitigation

The security issues we identified in the Matter protocol expose systemic weaknesses with potentially wide-reaching consequences for the Matter ecosystem. With over 8,000 unique certified devices, these vulnerabilities may affect all currently deployed Matter products.

As detailed in Sect. 4.4, a weakness lies in Matter's use of PBKDF2 in the PASE protocol due to the low iteration count. Rather than relying solely on iterations, we recommend adopting a memory-hard password hashing derivation function such as Argon2 [7]. Argon2 is specifically designed to make brute-force attacks costly by introducing memory requirements that hinder parallelisation and GPU acceleration [9]. In our benchmarks, we evaluated Argon2id using the recommended secure parameter set with the smallest memory footprint (7 MB). Despite these secure settings, Argon2id completed a derivation in approximately 11.56 ms, compared to 214.24 ms for PBKDF2 with the recommended 600,000 iterations. This makes Argon2id roughly 18 times faster, while providing significantly better protection against pre-computation attempts due to its memory-hard design. The primary trade-off lies not in performance, but in memory footprint. Devices with as little as 32 KB of RAM are noted to support Matter. The capability of such devices to run as a commissioner is questionable. Nonetheless, if a commissioner has at least 7 MB of usable RAM, there is little justification for preferring PBKDF2 over Argon2id. Exact benchmark results for Argon2id are provided in Appendix H.

Another weakness stems from Matter's use of a 27-bit numeric setup passcode. While devices embed this passcode in a QR code, users may also enter it manually via an 11-digit "QR Key." A simple change to an 11-character alphanumeric format would raise the entropy to approximately 57.5 bits, substantially improving security with minimal impact on usability. The design features a numeric-only design to preserve usability across regions – particularly to avoid confusion in locales that do not use Latin alphabets, despite the fact that a larger alphanumeric character-set would offer stronger protection.

While the use of public salts is standard in cryptographic protocols, they *must* be uniquely generated per session to prevent pre-computation attacks. Matter does not enforce this, weakening its resilience. Moreover, fresh salt generation is computationally feasible: Matter devices already perform intensive operations such as elliptic-curve multiplications. It was initially suggested that enforcing dynamic salts would require changes to QR code generation, but this is not the case: the

QR code and passcode remain unchanged, with only the derived keys being affected. Ultimately, the design prioritises "efficiency" over stronger security guarantees due to concerns about computational burden on resource-constrained devices.

Matter's use of SPAKE2+ within PASE raises further concerns, as the protocol is inherently susceptible to offline pre-computation attacks. Additionally, if an adversary were to recover the discrete logs of the public parameters M or N, the security of both SPAKE2+ and PASE would collapse entirely. We recommend replacing SPAKE2+ with OPAQUE [17], a protocol explicitly designed to resist such attacks and provide stronger assurances during device pairing. Alternatively, Matter could avoid PAKEs altogether by adopting a simpler key establishment protocol, such as Noise's IXpsk2 variant [18, 25].

We notice a pattern in Matter's design with a tendency to ship one set of cryptographic algorithms that must work even for the least constrained devices. We believe in a more crypto-agile approach that maximizes security regardless of device constraints. This is highlighted by the issue we found with CASE Resumption. This protocol is motivated as such in the spec:"resumption does not require expensive signature creation and verification, which significantly reduces the computation time". We recommend dropping the resumption protocol until a better alternative is proposed.

10 Conclusion

Throughout this study, we sought to evaluate Matter against the high standards it set through its ambitious goals as a secure and interoperable IoT protocol. As such, we presented the first extensive security evaluation of the Matter protocol. We highlighted the core cryptographic components of the standard and uncovered vulnerabilities that targeted all Matter devices.

We demonstrated multiple vectors that exploit these weaknesses: an attack on the CASE Resumption protocol, a brute-force attack, a pre-computation attack, and a controller-in-the-middle attack. Though not all are immediately practical, poor implementations can make them viable – even under Matter's own threat model. These attacks may grant adversaries full control over devices and unrestricted access to sensitive data.

We also examined Matter's Threats and Countermeasures list and found several countermeasures insufficient. To mitigate these issues, we proposed protocol-level improvements that enhance security without sacrificing scalability. Finally, we contributed the first formal verification of the security properties of Matter's key establishment protocols and the protocols from the literature on which Matter's designs are based.

Our findings reveal flaws that can be resolved by revisiting design decisions at the specification level, rather than relying on vendors to implement mitigations correctly. This would ensure stronger protections across all Matter devices without compromising the scalability or usability that Matter aims to deliver. Delegating critical security decisions, such as the validation of group membership, to implementers leads to inconsistent security across the ecosystem. We hope this work serves as a first step towards a more security-conscious evolution of the Matter protocol.

Acknowledgements

We thank the members of the Connectivity Standards Alliance for their time, responsiveness, feedback and contributions throughout the disclosure procedure as well as for their review of this document.

This work was supported in part by the Flemish Government through the Cybersecurity Research Program with grant number VOEWICS02, CyberSecurity Research Flanders with reference number VR20192203, and by the European Union's Horizon Research and Innovation program under grant agreement No. 101119747 (TELEMETRY). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of Cybersecurity Research Flanders or the European Union.

References

1. https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html#pbkdf2
2. Alliance, C.S.: Matter. https://github.com/project-chip/connectedhomeip
3. Alrawi, O., Lever, C., Antonakakis, M., Monrose, F.: Sok: Security evaluation of home-based iot deployments. In: 2019 IEEE Symposium on Security and Privacy (S&P) (2019)
4. Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J.A., Invernizzi, L., Kallitsis, M., Kumar, D., Lever, C., Ma, Z., Mason, J., Menscher, D., Seaman, C., Sullivan, N., Thomas, K., Zhou, Y.: Understanding the mirai botnet. In: 26th USENIX Security Symposium (USENIX Security 17) (2017)
5. Barker, E., Kelsey, J.: Recommendation for random number generation using deterministic random bit generators. Special Publication 800-90A Revision 1, National Institute of Standards and Technology (2015), https://doi.org/10.6028/NIST.SP.800-90Ar1
6. Barthe, G., Hedin, D., Zanella-Béguelin, S., Grégoire, B., Heraud, S.: A Machine-Checked Formalization of Sigma-Protocols. In: 23rd IEEE Computer Security Foundations Symposium, CSF 2010. pp. 246–260. 23rd IEEE Computer Security Foundations Symposium CSF 2010, IEEE, Edinburgh, Sweden (Jul 2010), https://inria.hal.science/inria-00552886
7. Biryukov, A., Dinu, D., Khovratovich, D.: Argon2: New generation of memory-hard functions for password hashing and other applications. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 292–302 (2016)
8. Blanchet, B., Smyth, B., Cheval, V., Sylvestre, M.: Proverif 2.05: Automatic cryptographic protocol verifier, user manual and tutorial. https://bblanche.gitlabpages.inria.fr/proverif/manual.pdf (2023)
9. Blocki, J., Harsha, B., Zhou, S.: On the economics of offline password cracking. In: 2018 IEEE Symposium on Security and Privacy (S&P) (2018)
10. Cash, D., Kiltz, E., Shoup, V.: The twin diffie-hellman problem and applications. In: Smart, N. (ed.) Advances in Cryptology – EUROCRYPT 2008 (2008)
11. Chen, L., Moody, D., Peralta, R., Sudharshan, R., Regenscheid, A.: Recommendation for discrete logarithm-based cryptography: Elliptic curve domain parameters. Special Publication (Draft) 800-186, National Institute of Standards and Technology (2019), https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-186-draft.pdf
12. Connectivity Standards Alliance (CSA): Matter Security and Privacy Fundamentals (2022), https://csa-iot.org/wp-content/uploads/2022/03/Matter_Security_and_Privacy_WP_March-2022.pdf, available online
13. Connectivity Standards Alliance (CSA): Matter Core Specification Version 1.3 (2024), https://csa-iot.org/developer-resource/specifications-download-request/, available online
14. DeLoatch, P.: Matter's mission to unify smart homes from apple, amazon, and google. The Verge (2021), https://www.theverge.com/22787729/matter-smart-home-standard-apple-amazon-google
15. Dworkin, M.: Recommendation for block cipher modes of operation: The ccm mode for authentication and confidentiality. Special Publication 800-38C, National Institute of Standards and Technology (2004), https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf
16. Elaine Barker, W.C.B.: Recommendation for password-based key derivation. Special Publication 800-132, National Institute of Standards and Technology (2010), https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf
17. Jarecki, S., Krawczyk, H., Xu, J.: Opaque: An asymmetric pake protocol secure against pre-computation attacks. In: Advances in Cryptology – EUROCRYPT 2018. pp. 456–486 (2018)
18. Kobeissi, N., Nicolas, G., Bhargavan, K.: Noise explorer: Fully automated modeling and verification for arbitrary noise protocols. In: 2019 IEEE European Symposium on Security and Privacy (EuroS&P) (2019)
19. Krawczyk, H.: Sigma: The 'sign-and-mac' approach to authenticated diffie-hellman and its use in the ike protocols. In: Advances in Cryptology - CRYPTO 2003. pp. 400–425 (2003)

20. Küsters, R., Truderung, T.: Using proverif to analyze protocols with diffie-hellman exponentiation. In: 2009 22nd IEEE Computer Security Foundations Symposium. pp. 157–171 (2009)
21. Liao, S., Yan, J., Cheng, L.: Wip: Hidden hub eavesdropping attack in matter-enabled smart home systems. In: Workshop on Security and Privacy in Standardized IoT (SDIoTSec) (2024)
22. Lim, C.H., Lee, P.J.: A key recovery attack on discrete log-based schemes using a prime order subgroup. In: Kaliski, B.S. (ed.) Advances in Cryptology — CRYPTO '97 (1997)
23. Loos, M.: Security analysis of the matter protocol. Master Thesis (2023)
24. Monckton, P.: Matter smart home standard explained: Here's why google, apple, and amazon are backing it. Tom's Guide (2022), https://s.42l.fr/Matter
25. Perrin, T.: The noise protocol framework. https://noiseprotocol.org/noise.pdf (2015)
26. Ronen, E., Shamir, A.: Extended functionality attacks on iot devices: The case of smart lights. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P) (2016)
27. Ronen, E., Shamir, A., Weingarten, A.O., O'Flynn, C.: Iot goes nuclear: Creating a zigbee chain reaction. In: 2017 IEEE Symposium on Security and Privacy (S&P) (2017)
28. Shafqat, N., Ranganathan, A.: Seamlessly insecure: Uncovering outsider access risks in aidot-controlled matter devices. In: 2024 IEEE Security and Privacy Workshops (SPW) (2024)
29. Shashwat, K., Hahn, F., Ou, X., Singhal, A.: Security analysis of trust on the controller in the matter protocol specification. In: 2023 IEEE Conference on Communications and Network Security (CNS). pp. 1–6 (2023)
30. Shoup, V.: Security analysis of SPAKE2+. Cryptology ePrint Archive, Paper 2020/313 (2020), https://eprint.iacr.org/2020/313
31. Taubert, T., Wood, C.A.: SPAKE2+, an Augmented PAKE. Internet-Draft draft-bar-cfrg-spake2plus-02, Internet Engineering Task Force (Dec 2020), https://datatracker.ietf.org/doc/draft-bar-cfrg-spake2plus/02/, work in Progress
32. Taubert, T., Wood, C.A.: SPAKE2+, an Augmented Password-Authenticated Key Exchange (PAKE) Protocol. RFC 9383 (Sep 2023), https://www.rfc-editor.org/info/rfc9383
33. Turan, M.S., Barker, E., Kelsey, J., McKay, K., Baish, M., Boyle, M.: Recommendation for the entropy sources used for random bit generation. Special Publication 800-90B, National Institute of Standards and Technology (2018), https://doi.org/10.6028/NIST.SP.800-90B

A Cryptographic Primitives in Matter

Matter utilises a broad set of cryptographic primitives to secure communication, ensure authentication, and maintain data integrity. It mostly follows the NIST guidelines, outlined in [5,11,15,16,33].

- **Deterministic Random Number Generators**
  - **CTR DRBG**: Uses AES in Counter mode for deterministic pseudorandom bit generation.
  - **Hash DRBG**: Employs SHA-256 and SHA-512 for deterministic randomness generation.
  - **HMAC DRBG**: Utilises HMAC with SHA-256 and SHA-512 for pseudorandom bit generation.
- **True Random Number Generators**
  - Provides non-deterministic entropy for key generation and nonce creation, often sourced from hardware-based randomness, if the device supports it.
- **Hash Function**
  - **SHA-256**: The primary hash function in Matter, used for data integrity and message authentication.
- **Message Authentication Code**
  - **HMAC-SHA-256**: Used for message authentication, ensuring data integrity and authenticity.
- **Public Key Cryptography**
  - **ECDSA (Elliptic Curve Digital Signature Algorithm)**: Provides digital signatures on the NIST P-256 curve (secp256r1) for data authenticity and non-repudiation.

  - **ECDH (Elliptic Curve Diffie-Hellman)**: Enables secure key exchange on the NIST P-256 curve, allowing devices to derive shared secrets.
- **Symmetric Encryption**
  - **AES-CCM (Counter with CBC-MAC)**: Provides authenticated encryption, ensuring confidentiality, integrity, and authenticity of data in transit.
- **Key Derivation Functions**
  - **HKDF (HMAC-based Key Derivation Function)**: Based on HMAC-SHA-256, derives cryptographic keys through extraction and expansion.
  - **PBKDF2 (Password-Based Key Derivation Function 2)**: Uses HMAC-SHA-256 with a public salt and configurable iteration count to generate keys from passcodes, primarily during commissioning.

B Matter's Public Key Infrastructure (PKI)

Matter's Public Key Infrastructure (PKI) framework is designed to ensure secure device identity verification, establish trust relationships, and maintain confidentiality within the IoT ecosystem. It is key to securely onboard devices, verify their identities, and perform access control.

B.1 PKI Hierarchy: Root CAs, ICAs, and DACs

Root CAs: The Root CA is the highest level in the PKI hierarchy and acts as the ultimate trust anchor. It issues certificates to ICAs, and each Root CA is generally maintained by a trusted, central entity overseeing the Matter PKI system.

Intermediate CAs (ICAs): ICAs are subordinate to the Root CA and are authorised to issue DACs to devices. This intermediary layer increases security by isolating the issuance of device-level certificates from the Root CA, with ICAs often managed by the device manufacturers or third-party PKI providers.

Device Attestation Certificates (DACs): DACs are unique to each device and verify its identity and compliance with the Matter standards. During manufacturing, a device is provisioned with a DAC signed by an ICA, ensuring each device's unique identity is authenticated and verified within the Matter ecosystem.

B.2 Authentication and Device Attestation

Matter uses DACs for device attestation during the commissioning process. Attestation allows the Commissioner to verify a device's legitimacy before it allows it to join a fabric. Attestation takes place as follows:

1. Upon request, the device presents its DAC to the Commissioner.
2. The Commissioner validates the DAC against the issuing ICA, tracing the certificate chain up to the Root CA.
3. Once validated, the Commissioner recognises the device as authorised and compliant, allowing it to proceed with the onboarding process.

Attestation through PKI ensures that only devices with verified identities and certificates can enter a Matter fabric, protecting the fabric's security integrity.

B.3 Operational Certificate Assignment and the Role of DACs

Once a device's DAC is verified, the Commissioner issues a Node Operational Certificate (NOC), establishing the device's identity within the fabric. The DAC serves as a pre-requisite for obtaining the NOC, verifying that the device has passed initial authentication. The NOC provides each device with a cryptographic identity for secure fabric operations, with the PKI system maintaining robust identity management. In the context of a NOC, we distinguish between two types of authorities: an optional Intermediate Certificate Authority (ICA) and a Root Certificate Authority (RCA), where the RCA serves as the root of trust. It is important to note that Product Attestation Authorities/Intermediates (PAA/PAI) are entirely unrelated to ICA/RCA. Indeed, there are two distinct certificate chains in Matter: The PAA/PAI chain authenticates the device's identity, independent of any fabric, while the (ICA/RCA) signs the device's operational credentials specific to a particular fabric.

B.4 Certificate Lifecycle Management and Revocation

Matter's PKI system incorporates lifecycle management and revocation mechanisms. Devices periodically check certificate validity, ensuring whether certificates are valid or expired/revoked. Certificate Revocation Lists (CRLs) are employed to manage revoked certificates, ensuring that the fabric retains only trusted devices.

C Node Operational Certificate (NOC)

The Node Operational Certificate (NOC) is a core component of Matter's identity and access control infrastructure. Each commissioned device receives a unique NOC, which enables secure interactions within the fabric by providing a cryptographically authenticated device identity.

C.1 NOC Issuance Process

The commissioner obtains the NOC after successful device attestation. Key steps in the NOC issuance include:

1. After DAC validation, the device generates a keypair and submits a Certificate Signing Request (CSR) to the commissioner requesting a NOC.
2. The commissioner relays this request to the fabric's trusted CA.
3. If the request is valid, the CA generates and signs a NOC, so that the device's cryptographic identity is tied to the fabric.
4. The NOC is relayed back to the device by the commissioner, where it is securely stored, providing the basis for mutual authentication with other devices.

C.2 Role of NOCs in Secure Communication

NOCs are the core component of the Certificate-Authenticated Session Establishment (CASE) protocol. Pairs of devices in a fabric use this protocol to establish secure channels among them. Consequently, all communications in a fabric would benefit from the confidentiality and mutual authentication properties provided by this protocol.

D Shared Key Derivation Differences

In the SPAKE2+ RFC document, the key derivation process is as follows:

```
(1) K_main = Hash(TT)
(2) K_shared = KDF(nil, K_main, "SharedKey")
(3) K_confirmP ‖ K_confirmV =
     KDF(nil, K_main, "ConfirmationKey")
```

Here a key K_{main} is derived from the hash of the transcript TT as an intermediate step. Using K_{main}, we make two distinct KDF calls to derive the shared secret K_{shared}, and the confirmations keys $K_{confirmP}, K_{confirmV}$.

The key difference from second draft version is that the shared key K_e is one half of the hash of the transcript TT. The method from the final SPAKE2+ specification is thus potentially more secure as it passes the entire hash into a key derivation function to create the shared value. This might be helpful in completely decoupling the shared secret from the protocol transcript. We are uncertain if sticking to the old design poses any significant problems.

E Further details about message encryption using PASE & CASE

Matter utilises two primary protocols for establishing secure communication channels: PASE and CASE. PASE is used between a non-commissioned device and a commissioner, with the goal of establishing a secure channel to ultimately install a NOC on that device. CASE requires a NOC to be executed, and is therefore only usable after a device has obtained this certificate. Its goal is to establish secure mutually authenticated channels between pairs of devices in a fabric.

E.1 Session Context and Roles

Matter employs a "secure session context", which holds the following PASE/CASE-related data about each session on a Matter device:

Session Type Indicates whether the session was established using PASE or CASE.
Session Role Specifies whether the node is the session's initiator or the responder.
Local Session Identifier A unique identifier used to look-up encryption keys and authenticate incoming messages. This identifier is used to map from an incoming message's Session ID field to the session context data.
Peer Session Identifier Assigned by the peer during session establishment, used in the Session ID field of every outgoing message, so that the peer can use it as the Local Session Identifier.
I2RKey Encrypts and decrypts data sent from the initiator to the responder.
R2IKey Encrypts and decrypts data sent from the responder to the initiator.
SharedSecret The secret that is computed during the CASE protocol execution. This value is used for CASE session resumption.

E.2 Encryption Key Derivation and Message Construction

During PASE or CASE, the derived shared secret is used to derive a pair of symmetric encryption keys (*I2RKey* and *R2IKey*) based on session roles. Message counters are instantiated with a random

number between 1 and 2^{28} for each session, which helps with masking the duration for which a session has been open. The message counters are incremented with every transmission to allow peers to detect replay attacks, and are reset at a threshold or when a session is resumed. Session-specific parameters, including message counters and a session identifier, help ensure integrity and confidentiality. The secure channel established by the session context uses these parameters throughout the communication.

F Access Control in Matter

Matter utilises access control policies to ensure that only authorised nodes can interact with functionalities exposed by a node's endpoint clusters. By default, no implicit access is provided, and permissions must be explicitly granted using Access Control Entries (ACEs). Each ACE is defined by specific access control policies and is scoped to the associated fabric within a node's Access Control List (ACL).

F.1 Authentication Modes

Matter's access control leverages different authentication modes to verify the identities of subjects initiating actions on target nodes:

- **PASE**: Used during commissioning.
- **CASE**: Used in operational phases, authenticated via a certificate (NOC).
- **Group**: Utilised in operational phases, authenticated via an Operational Group Key.

F.2 Access Control Entries (ACEs)

Each ACE defines specific permissions assigned to subjects for given targets, containing fields like:

- **FabricIndex**: Links the entry to the associated fabric.
- **Privilege**: Specifies the level of access, such as View, Operate, Manage, or Administer.
- **Authentication Mode**: Indicates the applicable authentication method.
- **Subjects**: Lists applicable subjects for the ACE.
- **Targets**: Identifies clusters or endpoints the entry applies to.

F.3 Wildcards

Wildcards in ACEs allow a single entry to grant privileges to multiple subjects or targets. An empty `Subjects` list (`[]`) denotes that all subjects using the specified authentication mode are granted the privilege, while an empty `Targets` list applies the privilege to all clusters on all endpoints.

F.4 Bootstrapping

During commissioning, administrative privileges are automatically granted to the commissioner, allowing it to initialise the ACL on the node. Typically, this privilege-transfer enables the commissioner (or an administrator node assigned by it) to configure and manage access control policies.

F.5 Security Considerations

Administrators must exercise caution when defining and distributing ACLs to prevent unintentional access. It is recommended to not incrementally update ACEs, as this may inadvertently introduce wildcard entries with overly broad permissions. Specifically, ACEs that use wildcards in combination with the Group Authentication Mode may grant privileges to numerous senders, increasing the potential attack surface.

The use of wildcards should be carefully monitored. We would recommend that Matter reduces the risk of confusion by employing a clear unique symbol to represent wildcard entries.

G Brute-Force Attack Feasibility Analysis

To assess the practicality of the brute-force attack, we measured the time required to attempt 20 incorrect passcodes, the maximum allowed before the session is terminated, and the time required to restart the commissioning process. Our measurements were conducted using the official Matter SDK on a MacBook Air with the following specifications: *Apple M2 Chip, 16 GB RAM, macOS Sequoia 15.3.2*.

The median time to perform 20 pairing attempts was 19.85 seconds, with a 95% confidence interval of ± 0.12 seconds. Restarting the commissioning session had a median time of 5.01 seconds, with a confidence interval of ± 0.00 seconds. We consider two different scenarios.

In the first scenario, we assume that the attacker is forced to restart commissioning after 20 attempts. The number of required restart cycles is:

$$\left\lceil \frac{99{,}999{,}987}{20} \right\rceil = 5{,}000{,}000$$

Each cycle (20 attempts + one restart) takes:

$$19.85 \text{ s} + 5.01 \text{ s} = 24.86 \text{ seconds}$$

The total time to exhaust the passcode space is:

$$5{,}000{,}000 \times 24.86 \text{ s} = 124{,}300{,}000 \text{ s} \approx 3.94 \text{ years}$$

If the 20-attempt limit can be bypassed (which, based on our findings, appears to be within practical reach), the attacker can send all pairing attempts in a single uninterrupted session. The average time per attempt becomes:

$$\frac{19.85 \text{ s}}{20} = 0.9925 \text{ s}$$

Hence, the total time required is:

$$99{,}999{,}987 \times 0.9925 \text{ s} = 99{,}249{,}988 \text{ s} \approx 3.15 \text{ years}$$

H Argon2id Benchmark Results

To evaluate the practical performance of Argon2id, we benchmarked it using the recommended secure parameter set with the smallest memory footprint (7 MB). In this configuration, deriving a single key takes

$$11.56\,\text{ms} \quad (95\,\%\,\text{CI} : [11.55\,\text{ms},\ 11.58\,\text{ms}]).$$

Given the eight-digit passcode space of size $N = 99,999,987$, the time to pre-compute a complete rainbow table using Argon2id on a single CPU core is:

$$T_1 = 11.56\,\text{ms} \times 99,999,987 \approx 1,155,998\,\text{s} \approx 13.5\,\text{days}.$$

Assuming ideal linear scaling across 128 identical cores, this time reduces to:

$$T_{128} = \frac{T_1}{128} \approx 2.4\,\text{hours}.$$

This calculation shows that, despite Argon2id's memory-hard design, pre-computation attacks remain feasible in the static-salt scenario. Argon2id is significantly faster than PBKDF2 at 600,000 iterations (approximately 18 times faster), which lowers the computational barrier for generating a full rainbow table when salts are fixed. However, the primary defense offered by Argon2id is not computation time but its memory requirements. With 7 MB of RAM required per derivation, attempting to compute all $99,999,987$ passcode hashes in parallel would demand approximately 700 TB of RAM. Such requirements are beyond practical reach for an attacker.

This highlights the importance of using dynamic salts, as they force the attacker to redo the expensive memory-hard computation for every individual guess. In such a setting, the memory cost of Argon2id effectively prevents pre-computation attacks.