FILAMENT

Filament Security Overview

Networking Devices at the Edge of Risk

Executive Summary

Filament is laying a foundation for the next industrial revolution by connecting autonomous physical objects to the Internet economy. In order to unlock the enormous value of the Internet of Things (IoT), strong privacy and security safeguards absolutely must be in place at every layer of the network, from physical hardware through communication protocols to data analytics and end-user applications. At Filament, we have invested deeply in building secure technologies that make economic exchange among autonomous devices not only a possibility but a reality.

The Internet Security Glossary (Shirey, 2007) defines security as "A system condition in which system resources are free from unauthorized access and from unauthorized or accidental change, destruction, or loss." Our intent is that Filament technologies will always be as secure as we can possibly make them; in particular, if they can withstand sophisticated, well-resourced attacks through qualities such as penetration resistance, trustworthiness, and resilience (see NIST SP 800-160), they will provide a reliable basis for mission-critical industrial infrastructure.

Core members of the Filament team previously developed and standardized Jabber/XMPP, a messaging technology that has been widely deployed in financial and military chat systems. Based on this experience, Filament has established several goals related to privacy and security:

- Hardware, firmware, and keying material must be physically protected
- Communication among devices must be private and confidential
- Devices must be able to operate without servers or cloud services
- Filament must never have access to customer data
- Access to devices must be restricted to authorized entities

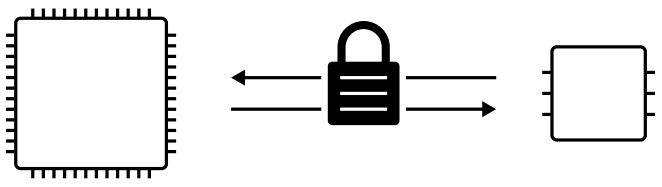
This whitepaper provides a deeper description of how we achieve these objectives. To organize the many topics involved, our storyline follows the chain of custody and deployment of a device from manufacture to active use in the field.

Security Begins With Hardware

When we set out to build secure devices for industrial IoT, off-the-shelf hardware meeting our stringent requirements did not exist. Therefore, we decided to build our own printed circuit board assembly (PCBA) using existing components.

The foundation of our hardware security strategy leverages the capabilities of a dedicated secure element that supports advanced cryptographic functions and physically protected storage of public and private keys.

At the time of manufacture, with the microcontroller unit (MCU) on the Filament Patch, this tamper-resistant crypto chip is paired through a key exchange process with the MCU on the Filament Patch to form a strong binding, establish trust that that the MCU will only talk to its paired crypto chip and vice versa. This pairing also provides a foundation for securely



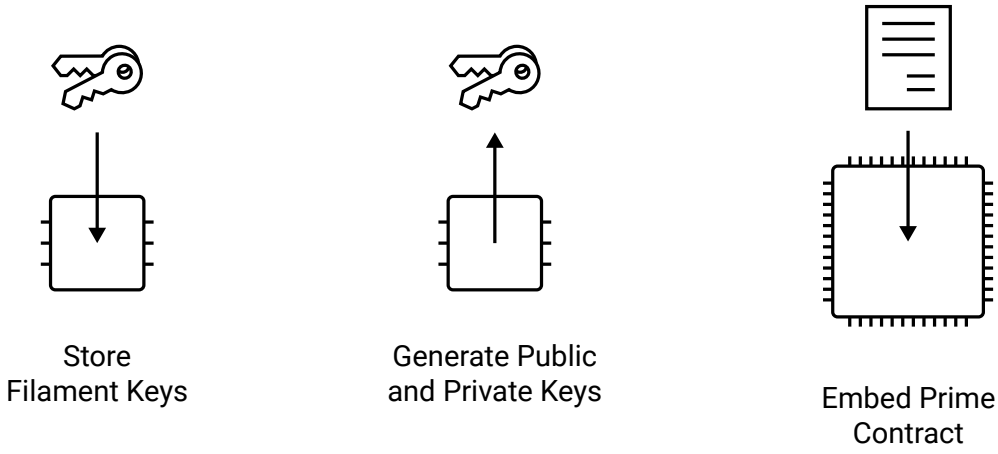
Microcontroller Crypto Chip

booting the device. (Other elements on the PCBA – such as flash memory, a long-range radio chip, and a Bluetooth Low Energy chip – are less trusted than the MCU.)

In addition to access to the crypto chip, a second requirement for booting the MCU is a cryptographic check of the resident firmware to ensure that the signatures match the last known good version. This helps to avoid attacks involving a downgrade to a previous (and potentially less secure) version of the firmware, as well as attempts to manipulate the firmware stored in on-board flash memory within the MCU.

Several additional measures are in place to protect device operations.

> First, all firmware updates are signed with a protected signing key to ensure that only firmware produced by Filament can be loaded onto the device. Optionally, firmware made available for distribution to a particular customer's devices can also be co-signed by the customer for added control.
>
> Second, firmware is encrypted with another key to prevent attackers from determining the exact firmware being deployed to a device.
>
> Third, the MCU creates an encrypted file system on the flash memory chip so that sensitive data is encrypted "at rest" on the device. This information includes the device's initialization script, radio configuration, list of current communication partners, and any temporary sensor data.



Further protections include a hardware random number generator (RNG) for improved randomness in keying material, a strong key derivation function in the MCU to prevent access to or tampering with the crypto chip, called HKDF (Krawczyk and Eronen, 2010). If tampering

is attempted, the hardware sets an internal flag so the firmware is aware of the event and can reset the keys.

When each device is created on the production line, it is imprinted with public keying material that is used for signing, encryption, pairing, unlocking, and identification. In addition, the secure element self-generates private keying material that is inaccessible via API calls, even to the MCU. These operations occur while the PCBA is under the physical control of Filament, well before a device has been shipped to a customer, to prevent insecure initialization in the field.

One critically important keypair is used to create a stable and unique address for the device. This address – called a "hashname" and used within the telehash protocol for inter-device communication as described below – is a 52-character, base32 string derived from the fingerprint of the public key for the device (e.g., "eik5s2mkhgirnrk5ggf66zye4uv6zbiflhyo5bopeqgc6pjujyta"). Naturally, this long-term identity key is not directly used for encryption; instead, ephemeral key pairs are used in a Diffie-Hellman negotiation so as not to compromise forward secrecy.

Finally, Filament takes great care in how we produce the organizational keys underlying operations such as firmware signing and Blocklet smart contracts. These keys are generated using a hardware security module (HSM) with input from multiple smart cards or subkeys that are separately stored in safe deposit boxes located in different states. New signing and contracting keys can be created only if enough subkeys are presented during the signing ceremony, similar to how the keys are generated for the 13 root servers of the Domain Name System (DNS).

Device Initialization with Smart Contracts

When a customer places an order for, say, 1000 devices, we jointly define a digital smart contract to be placed on the devices, called a "prime contract." This contract, which uses our Blocklet technology and takes the form of a JSON Web Token as specified in RFC 7519 (Jones et al., 2015), is a self-enforcing digital object (Szabo, 1997) that provides a representation in software of the rights and responsibilities of both Filament and the customer in relation to these devices.

For instance, the contract might specify that Filament shall provide signed and encrypted firmware updates for the life of the contract, and that the customer shall be able to modify the behavior of the device (within certain parameters) using the on-board JavaScript virtual machine. This contract is digitally signed by both parties, placed onto the devices at the time

HEADER	Hashing Algorithm	
PAYLOAD	***RESERVED***	Issuer, Expiration, Audience, etc.
	PUBLIC	Standard Values (e.g. OpenID Connect)
	PRIVATE	Filament-specific Values
SIGNATURE	Hash of `HEADER + PAYLOAD`	

of manufacture using cryptographically authenticated tooling at the factory, and stored in an encrypted form on the device. The Blocklet prime contract forms the foundation for trust throughout the application stack on the device.

One further piece of information included in the smart contract is a customer-specific community name that is used for bootstrapping private communication on the customer's radio network. For convenience, we can think of this community name as similar to a Wi-Fi hotspot name. Similar to the way in which a community name is used to bootstrap security on encrypted Wi-Fi systems such as Wi-Fi Protected Access (WPA), on a Filament network the name is used as input to HKDF so that all of the devices within the customer's private community can algorithmically derive the same shared secret, cipher key, and radio settings (such as channels and listening times). This information is used only for discovery of other devices and during the process of joining the network; when a new device is deployed at the customer's location (say, a factory or a construction site) it immediately has shared context with the existing devices and can join the network if it was initialized with those permissions.

Unlike Wi-Fi systems, which only provide network-layer security, Filament devices also enforce session-based security for inter-device communication.

Specifically, the process of establishing communication with any given device on the local radio network requires cryptographic handshaking in accordance with the telehash protocol as described below. This handshake includes both the device's public identity key and information from the device's prime contract, proving a chain of trust back to Filament and the particular customer, similar to how trusted authorization works on the Public Key

Infrastructure (PKI) in Secure Sockets Layer (SSL) and Transport Layer Security (TLS). In this way, a newly deployed device must prove to its peers that it is allowed to be on the network. Once each peer-to-peer handshake completes, all further communication between two parties takes the form of a private and encrypted telehash link.

The foregoing workflow is used for private, customer-specific communities. Naturally, it might be desirable to also run public communities, for example in a smart city deployment. Even though a device might simultaneously be able to communicate with both a private community and a public community, it needs a way to determine the trust relationships it has with a device requesting communication over the public community. This is where blockchain-based systems such as Blockstack, Namecoin, and Blockname come in. These technologies can provide global, decentralized registries of "things" like device identities and public keys.

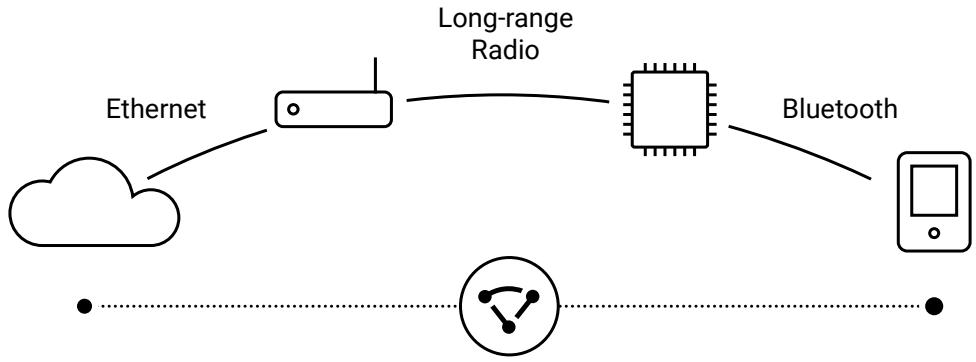
Unlike centralized technologies such as the Domain Name System (DNS) in which addresses are assigned in a hierarchical context (device@host, with the host obtaining its identity through registration of a DNS domain name), a blockchain-based approach is more agile because it enables devices to register directly. This obviates the need for accounts at centralized host servers, which are often subject to external cracking (e.g., dictionary attacks) and internal leaks (e.g., theft of password databases). In addition, the decentralized nature of these technologies reduces the threat of denial of service (DoS) attacks of the kind that have been launched against the DNS.

A further security property is that each registered identity can be associated with the device's public key, thus enabling more secure bootstrapping of communication and greater trust in the overall network. Although Filament has not yet deployed any public networks, we are actively monitoring blockchain-based discovery technologies so that we are ready to use them when appropriate.

Always-On Encryption

Telehash is the protocol we use for encrypted communication. It represents a further evolution of our technical team's original goals with Jabber/XMPP technologies. Among the benefits of telehash are a serverless architecture with strong security that cannot be disabled. Because there are no cleartext communication modes, downgrade attacks that force devices back to an unencrypted state are not possible. In addition, forward secrecy is always enabled and frequent re-keying is enforced, which means that even if a permanent identity key is compromised, past session keys (and the data communicated using those keys) are still protected. Furthermore, telehash uses modern cryptographic primitives of the

kind that will be included in the forthcoming version of Transport Layer Security (TLS 1.3), including block ciphers such as AES-GCM, stream ciphers such as Salsa and ChaCha, stream authenticators such as Poly1305, hashing algorithms such as SHA-256, and signing and encryption methods such as Elliptic Curve Cryptography (currently with the P-256 curve and a hardware RNG to help mitigate any potential weaknesses).



End-to-end Encryption via Telehash

One reason we prefer to use telehash is that it is transport-agnostic. This enables us to encrypt all communications per-hop and end-to-end, even if the end-to-end media path flows over a variety of transports. As an example, depending on network topologies and application requirements, data from an edge device might go over some combination of long-range radio, TCP/IP via a passthrough gateway to an on-premise server or a third-party cloud service, BLE to a smartphone or tablet in the field, or even USB serial connections to a local laptop for administrative access. All of these methods use the same encryption methods, thus reducing the attack surface and ensuring data confidentiality across all communication modalities.

Since Filament networks provide end-to-end encryption of data from the edge device all the way to a decryption endpoint of the customer's choosing, Filament cannot read any "data in motion" generated by the customer's application. Furthermore, we do not run any data storage systems on the other side of the decryption endpoint, since we integrate with the customer's preferred on-premise system or cloud provider; thus we cannot read "data at rest" either.

This is by design. We do not treat customer data as a revenue opportunity, but as inherently private information.

Radio Networking

Although end-to-end encryption addresses a wide range of privacy and security threats, plenty of potential attacks remain. Because Filament's current hardware uses long-range radio as its underlying transport mechanism, we have developed methods to prevent eavesdroppers from learning about communication patterns among our devices based on metadata that could otherwise be discovered over the air.

Many long-range radio systems for IoT use a technology called LoRaWAN, developed by the LoRa Alliance. Although Filament uses the same radio chip as many devices that support the LoRa standard, we do so to create ad-hoc networks instead of the centralized wireless architectures in LoRaWAN. The PHY-layer encryption from LoRaWAN is not applicable in this decentralized approach, so we have defined a MAC-layer protocol extension to telehash – called TMesh – for enhanced privacy. By using consistent frame sizes as well as seemingly random switching of radio channels and communication timing (which are known only to the devices within the private radio community, as previously described), we are able to effectively hide metadata about communication among the devices on the network.

Peer-to-peer networking technologies can be challenging to operationalize. We work around some of the traditional problems by automatically tuning the radio usage so that each device communicates with only a small number of physical or virtual neighbors instead of the entire network. Depending on the network topology that results from these real-time adjustments, as well as the number of devices on the network and their power state (e.g., on battery vs. A/C power), intermediate nodes might act as data relays or ad-hoc "routers" for messages sent between any two devices. To make this pattern private, packets are onion-routed and relays cannot decrypt destination payloads – they peel off the header information for one hop and route the resulting packet to the next peer. To make this pattern scalable, devices do not keep a large routing table of full media paths in memory. Instead, they know about their neighbors and address their packets accordingly, or in some circumstances maintain an end-to-end session with the destination hashname if the media path is stable enough for the life of the session.

Access Control

In addition to the community bootstrapping functions described above, Blocklet smart contracts also provide cryptographically strong methods for authentication, authorization, and access control. When IoT devices are initialized with contractual capabilities via Blocklet, they can use those contracts as the basis for verifying the identity of other entities (e.g., other

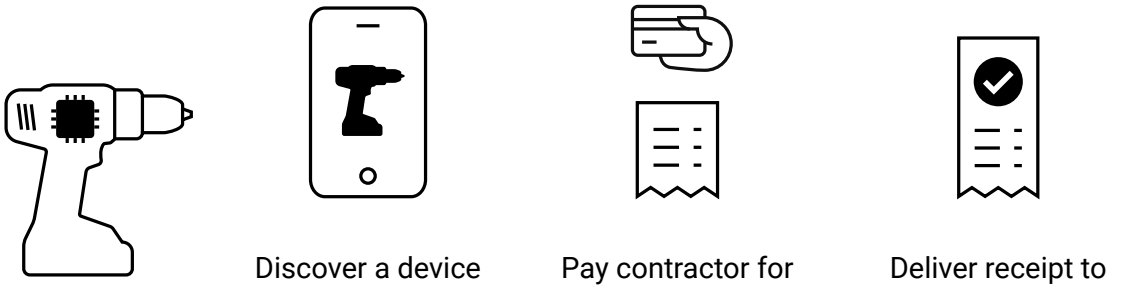
IoT devices, or management apps on tablets and smartphones). This is accomplished by ensuring that the other entity presents cryptographic credentials that can be chained back to a common root. In addition, these contracts can enable fine-grained control over data reporting, network connectivity, and management operations through separately permissioned trust branches for different internal departments (e.g., line workers vs. IT) and different external contractors or vendors.

All of these techniques mitigate against common attacks on IoT deployments because the devices involved do not allow access based on username/password combinations, which are often easily guessed or which even ship with known defaults and are never changed in the field. Such a vulnerability led to the Mirai botnet in 2016, but would be extremely difficult with cryptographically strong credentials. By using Blocklet to deploy self-executing, self-enforcing smart contracts on IoT devices, Filament devices can make access decisions autonomously at the edge of the network without an always-on connection to a centralized policy server.

One special usage of this approach on Filament devices involves control over the on-board scripting environment, which takes the form of a JavaScript virtual machine (VM). The only entities that are allowed to upload scripts to the device are those which present Blocklet credentials matching the permissions model established in the prime contract and any subsidiary contracts. In addition, because the VM operates as an event-driven computing platform, the MCU can enforce additional checks to ensure that the provided script does not compromise the underlying functionality and service expectations for the device (e.g., battery life requirements that are part of the prime contract). By using Blocklet throughout the entire firmware, software, and application stack on the device, Filament is able to fully leverage the strong binding between the physical and digital worlds that is founded in the tamper-resistant crypto chip and secure enclave.

Contractuality and Microtransactions

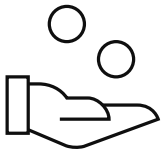
A fundamental goal of our technology stack is connecting devices not just to the Internet, but to the Internet *economy* (see our white paper "Foundations for the Next Economic Revolution: Distributed Exchange and the Internet of Things"). The intent is for devices to be autonomous economic actors - so that, for instance, a car could itself pay for parking, a toothbrush could itself order replacement heads, or a temporary worker could pay for access to a tool for each use. Another example is directly selling temporary access to a network deployed along a highway, power line, railroad, or pipeline.



These ad-hoc interactions are made possible by the Blocklet capabilities of the device itself, rooted in the prime contract that is placed onto the device at the time of manufacture. Consider a prototypical microtransaction, such as selling network access by the minute. In order to reduce the amount of friction (e.g., transaction costs) involved in a series of such transactions, the two parties might branch off a payment blockchain (such as bitcoin) and establish a shared escrow or "tab" against which the buyer can charge smaller microtransactions. Such an escrow could be established using a blockchain multi-signature feature along with a lock time, so that the funds are not paid out until both parties agree that value has been exchanged in accordance with the underlying contract. By employing a shared escrow, the seller also limits the risk of double spending by the buyer (assuming eventual connectivity to sync with the relevant distributed ledger).

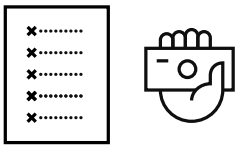
One common misunderstanding is that IoT devices might need to complete the resource-intensive task of calculating blocks in order to securely engage in blockchain interactions. Thankfully that is not true, as it is sufficient to keep the equivalent of a bitcoin wallet on the device. However, micropayment channels do require both parties to re-sign each transaction – an operation that can still be too intensive for highly constrained computing platforms.

Filament devices work around this problem by using a less-involved hashing technique to validate changes to the microtransaction tab between both parties. This technique builds on the Blocklet foundation by employing JSON Web Tokens along with zero-knowledge proofs to establish a set of shared secrets (SHA-256 hashes) to represent a series of transactions. As transactions are consummated and verified by both parties, the parties can clear payment up to the "high-water mark" at any time (e.g., if one of the devices goes offline or if the parties agree that the series of transactions is complete). And, of course, all of these interactions occur over encrypted links between devices to protect the sensitive information involved.



Buyer places money in escrow, and seller creates a “tab” of tokens

Buyer spends tokens in local microtransactions, and payments are recorded in the tab



When contract conditions are met, money is released from escrow to the seller

The Security Process

In this whitepaper we have described some of the key measures we take to ensure the privacy and security of device connectivity and contractuality. However, we recognize that security is a process, that new attacks emerge constantly, that industry best practices continue to evolve, and that it is difficult to be completely objective about the security of your own products and protocols. As part of our strong commitment to privacy and security throughout the product development lifecycle, we are committed to ongoing penetration testing by highly reputable external firms. We are also likely to invest in standardization and associated security review of protocols such as telehash, as well as in certification of our products under UL's Cybersecurity Assurance Program (UL 2900).

Additional documents and future versions of this whitepaper will provide even more in-depth information about our approach to privacy and security. In the meantime, we welcome communication regarding the security characteristics of our products, including notification of vulnerabilities, via security@filament.com (see https://filament.com/security/ for details).

References

Cooper, A., H. Tschofenig, B. Aboba, J. Peterson, J. Morris, M. Hansen, and R. Smith. 2013. RFC 6973: Privacy Considerations for Internet Protocols. https://www.rfc-editor.org/rfc/rfc6973.txt

Dierks, T. and E. Rescorla. 2008. RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2. https://www.rfc-editor.org/rfc/rfc5246.txt

Krawczyk, H. and P. Eronen. 2010. RFC 5869: HMAC-based Extract-and-Expand Key Derivation Function (HKDF). https://www.rfc-editor.org/rfc/rfc5869.txt

Jones, M., J. Bradley, and N. Sakimura. 2015. RFC 7515: JSON Web Signature (JWS). https://www.rfc-editor.org/rfc/rfc7515.txt

Jones, M. and J. Hildebrand. 2015. RFC 7516: JSON Web Encryption (JWE). https://www.rfc-editor.org/rfc/rfc7516.txt

Jones, M., J. Bradley, and N. Sakimura. 2015. RFC 7519: JSON Web Token (JWT). https://www.rfc-editor.org/rfc/rfc7519.txt

Ross, R, J.C. Oren, and M. McEvilley. 2014. NIST SP 800-160: Systems Security Engineering: An Integrated Approach to Building Trustworthy Resilient Systems. (Initial public draft.) http://csrc.nist.gov/publications/drafts/800-160/sp800_160_draft.pdf

Shirey, R. 2007. RFC 4949: Internet Security Glossary, Version 2. https://www.rfc-editor.org/rfc/rfc4949.txt

Szabo, N. 1997. "Formalizing and Securing Relationships on Public Networks." http://firstmonday.org/ojs/index.php/fm/article/view/548/469