



medcrypt

MAY 2023

DECRYPTING CRYPTOGRAPHY



INTRODUCTION: GETTING MEDICAL DEVICE CYBERSECURITY RIGHT

Implementing cybersecurity for modern medical devices requires compliance with complex regulations as well as adoption to a changing healthcare ecosystem where hospital networks are considered inherently hostile;¹ devices are increasingly integrated, and data is moving into the cloud. Getting cybersecurity right requires mature processes, careful design considerations, and finding the right balance between the desired level of security and a device's capabilities and utility. Getting cybersecurity wrong can have significant ramifications for patient safety, regulatory compliance and approval, and business and reputation.

Design, architecture, and hardware choices are examples for foundational decisions that need to be made early and need to be right - and need to be revisited on a continual basis as technologies and cyber threats change.

Security cannot be an afterthought, it needs to be designed in and requires the systematic application of good security engineering principles, lifecycle management processes, and the selection of the pertinent security technologies. For example, the following proactive security mechanisms reflect guidance provided by the FDA:

- Device data security via root-of-trust, data signing/verification, transport security and end-to-end encryption
- Security event detection and behavior monitoring integrated into the device itself
- Post-market risk and vulnerability management for all versions of devices in the field via SBOM

Cryptography

Among the many security practices and technologies, one stands out in its versatility but also its complexity, resulting design challenges and often imperfect (and hence insecure and/or uneconomical) implementation - and that is cryptography.

Cryptography is commonly used in commercial applications and standards-based implementations (e.g., full disk encryption, secure web communication via TLS). However, many more opportunities exist to use cryptography to implement or support a wide range of security functions. The use of cryptography in the design of medical devices will encounter implementation challenges (and failures), many only becoming obvious once a device is completed and enters commercial use. Finding the right approach to achieve the desired outcome, in light of common technical and use case restrictions, is not a trivial task.

¹ <https://www.meddeviceonline.com/doc/fda-releases-guidance-on-cybersecurity-in-medical-devices-0001>

GETTING CRYPTOGRAPHY RIGHT

Implementing modern cryptography requires establishing trust of cryptographic keys so to deliver on the promise of security, it is imperative to address two key dimensions:

#1

The required level of cryptographic protection as provided by a combination of the crypto algorithm and strength of the keys, including the effectiveness and security of mechanisms, protocols, and processes applied.

#2

The protection of the secret key material through appropriate practices, hardware choices, and management functions (including secure key generation, provisioning, storage, distribution, and use as well as considerations for lifecycle management of keys and certificates such as revocation and destruction).

Developing, deploying, and managing cryptography is complex and requires expertise and experience to assure an implementation that is secure and maintainable. Cryptography is all about trust and getting the PKI and root-of-trust part of it right is a non-trivial, complex task that ultimately determines the success and security of a cryptographic implementation. The following considerations address the technical and non-technical aspects of a cryptographic implementation:

Scalability	Support small to large deployments.
Key management infrastructure	Securely support desired deployment models.
Cost	Direct (e.g., hardware, certificates) and indirect (e.g., key lifecycle management and revocation, redesign of failed implementation).
Security	E.g., prevention of secret key material exposure or breach or crypto algorithm deprecation.
Crypto Agility	Foreseeing changes in technologies or security risks.
Liability	Regulatory (e.g., non-compliance) and legal (e.g., lawsuits).
Regulatory and Standards Compliance	E.g.: Regional Cybersecurity Regulations and Guidances (FDA Pre- and Postmarket, MDCG 2019-16, others as required); NIST (CSF and cryptography practices); ANSI/AAMI (TIR57, TIR97, SW96); ISO/IEC (e.g., ISO 81001-5-1, ISO 13485, ISO 14971)
Data Control and Retention	Supporting audit and forensic activities.
Aspects of Trust	<ul style="list-style-type: none">• Technical trust through a unique and verifiable identifier (e.g., certificate linked to a root of trust).• Public trust that could be impacted through security incidents (e.g., damage to reputation).

All basic cryptographic operations – encryption, signing, authentication, and key exchange – rely on mechanisms and processes that ensure that these secrets remain secret. If, for example, a key becomes known to an attacker, then the attacker gains the same privileges as the legitimate party. If the key is a signing key, then it can be used to sign any message, transaction, or document as the legitimate signer. If the key is a decryption key, then it can decipher everything protected by that key. And if the key is used for authentication of a person or device, then it can be used to impersonate that person or device at will. With the attacker using those secrets in the same way as the legitimate user, these attacks are difficult to detect. Correctly implementing all aspects of cryptography is a prerequisite for meeting the desired security goals and to achieve the required level of assurance.

Consequently, any design must not only consider the appropriate algorithm and key strength but also its implementation (i.e., how secret keys are protected while they are managed and stored).

In that context, the main cryptographic components are:

Algorithm	The mathematical formula that, in combination with a cryptographic key, is used to alter data to support the desired security function (e.g., encryption, decryption, signing, authentication). Different algorithms provide differing levels of cryptographic protection but also may use different levels of system resources (memory, CPU, battery) and differing times to execute. There are fundamentally two different types of algorithms, symmetric (using the same key for encryption and decryption) and asymmetric (using a public key for encryption and a private key for decryption). Within these two groups, varying algorithms are available to support the respective security function at the right level. Recommendations for various use cases are readily available. ^{2 3 4}
Algorithm Implementation	The actual software code that performs the cryptographic calculation and functions, ideally supported by hardware accelerators. Common open-source libraries exist, e.g., OpenSSL ⁵ or WolfSSL. ⁶ As software implementations of crypto algorithms is complex and requires great care to not result in insufficient security, it is generally not advised to write your own implementation of the crypto code. ⁷
Algorithm Lifecycle	Any good crypto implementation should allow for update of the cryptographic code (e.g., to remediate vulnerabilities) as well as, longer term, update of the algorithm to account for its to-be-expected future deprecation.
Cryptographic Key(s)	A string of data (numbers and/or letters or their binary representation) that, together with the data to be protected, is processed through a cryptographic algorithm. With modern cryptographic algorithms, the key is the protected secret, rather than the algorithm. Symmetric encryption uses the same key for encryption and decryption, which makes it more challenging to protect as it is a shared secret. Asymmetric encryption uses a public-private key pair that is unique to each communicating party and with the private key residing with the decrypting entity, an approach that makes it easier to prevent its exposure.
Key Protection	With the key (i.e., the private key for asymmetric encryption, the secret key for symmetric encryption) being the critical secret, its protection at rest (e.g., on a device) and in transit (e.g., during provisioning or update) is essential and will require careful design, hardware, and process decisions.
Key Lifecycle Management	Depending on use case, desired level of security, and economic and practical limitations, keys should be updated on a more or less frequent but regular basis. ⁸

² General NIST resources on cryptography: <https://www.nist.gov/cryptography>

³ Lightweight cryptography: <https://csrc.nist.gov/projects/lightweight-cryptography>

⁴ Postquantum cryptography: <https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022>

⁵ <https://www.openssl.org/>

⁶ <https://www.wolfssl.com/>

⁷ <https://resources.infosecinstitute.com/topic/the-dangers-of-rolling-your-own-encryption/>

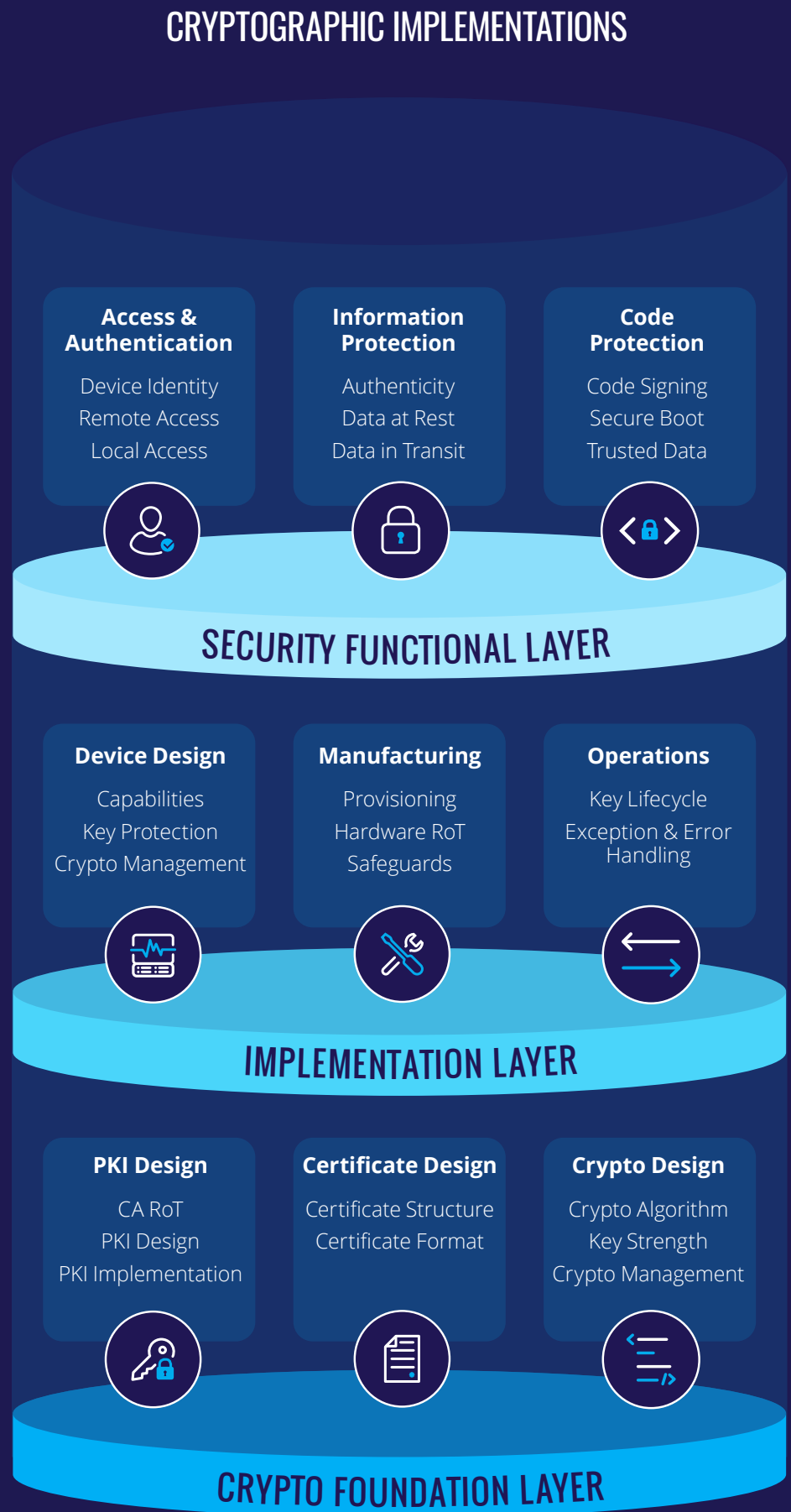
⁸ <https://csrc.nist.gov/Projects/key-management>

Certificate	Contains a public key and additional metadata, e.g., issuer or expiration date. Certificates should be signed by a trusted certificate authority using the CA's private key to lay the foundation for trust in the cryptographic process and form the baseline for subsequent layers in the overall PKI hierarchy.
Certificate Provisioning	The process of providing an initial identity to a device (usually during production) and signing it under an issuing CA that is signed by the root of trust prior to first operation in the field.
Public Key Infrastructure (PKI)	Public key infrastructure (PKI) - collective term for everything required to establish and manage public key encryption, including: <ul style="list-style-type: none"> • Certificate Authority (CA) that stores, issues and signs the digital certificates; • Registration Authority (RA) which verifies the identity of entities requesting their digital certificates to be stored at the CA; • A central directory—i.e., a secure location in which keys are stored and indexed; • A certificate management system managing e.g., access to stored certificates or the delivery of the certificates to be issued; • A certificate policy stating the PKI's requirements concerning its procedures. Its purpose is to allow outsiders to analyze the PKI's trustworthiness.
Root of Trust (RoT)	The foundational component to which all keys relate to for the purpose of confirming validity. Common examples are a hardware RoT within a device to establish the foundation on which all secure operations rely on, or a CA RoT as the ultimate trusted root certificate within a public key infrastructure (PKI). With the RoT being the foundation for the overall trust structure, explicitly protecting it with utmost adherence to best practices is of critical importance and requires great care in design.



ACHIEVING SECURITY THROUGH CRYPTOGRAPHY

Many security functions can be realized through cryptographic approaches and, if implemented correctly, can deliver the desired level of security, within the given restraints, that meets business goals and can be maintained over the lifetime of the device. Most fundamentally, cryptography establishes trust and therefore can be used to ensure information confidentiality, integrity, and authenticity. Depending on device capabilities and platform, standards-based commercial solutions are available, but often design of a custom solution is required that supports the desired functionality under the given restraints, e.g., hardware, memory, or battery limitations.



SECURITY FUNCTIONS PROVIDED BY CRYPTOGRAPHY

An overview of cryptographic security functions is provided in the top layer of the diagram (page 7).

ACCESS AND AUTHENTICATION	Of devices and device ecosystem:
Device Identity	Establish a unique and cryptographically secure identity of a device that can be reliably verified. May include binding between cryptographic identifiers and other identifiers, e.g., serial number or barcode.
Remote Access	A combination of identity verification (of device and user) and secure transmission so to prevent security compromise through a remote connection.
Local Access	A combination of identity verification (of device and user) and secure transmission so to prevent security compromise through a local connection.
INFORMATION PROTECTION	As the most basic and common application of security to ensure C-I-A of information and its metadata:
Authenticity	The ability to verify the origin of information to a level sufficient to establish trust and have confidence in its validity and legitimacy.
Data at Rest	Ensure that data stored on a device or within a device ecosystem is protected against compromise and unauthorized access or modification.
Data in Transit	Ensure that data exchanged between the device and the device ecosystem is protected against compromise and unauthorized access or modification.
CODE PROTECTION⁹	Is an extended use case of Information Protection (with the software code being the protected information) to ensure that a device's code can not be compromised:
Code Signing	The ability of a device to verify that to be installed software / firmware code or updates / patches are legitimate and come from a trusted source, i.e. is authentic and has not been altered, i.e., preservation of integrity can be demonstrated.
Secure Boot	The ability of a device to verify that its boot image can be trusted and the boot process can be initiated.
Trusted Data	The ability to verify trust in sensitive / critical data (e.g., configuration files) so it can be installed or used during device operation.

In summary, depending on the cryptographic function to be implemented, the actual design and implementation requires careful consideration of the device's use case, ecosystem, and security needs. Further, it also requires a future-proof cryptographic infrastructure. Designing cryptographic functions requires careful considerations and often is a tradeoff between the level of security desired vs. the device's capabilities within its use environment.

⁹ <https://csrc.nist.gov/publications/detail/white-paper/2018/01/26/security-considerations-for-code-signing/final>

CRYPTOGRAPHY IN DEVICE DESIGN

Implementing cryptography architecture and design must account for production, maintenance, and usability requirements. Making the right crypto design decisions is crucial not only to achieve the desired level of security but also make sure that the cryptographic function can be technically and economically supported in the future.

During the design phase, engineers are often confronted with finding a balance between device hardware and desired cryptography, e.g., select the right hardware to support the identified crypto algorithm, or to find the best possible crypto implementation for a given hardware. As depicted in the middle layer of the diagram, the following needs to be considered:

DEVICE DESIGN	Decisions and considerations impacting device design:
Capabilities	Of the selected hardware platform and chip set. Modern chip sets include hardware support for cryptographic functions (e.g., accelerators) as well as dedicated memory areas for the secure storage of cryptographic secrets (keys and certificates). However, traditionally very low resourced hardware, as we may find in older devices, may be limited in their capability and features but with modern hardware platforms this is less of a concern.
Key Protection	On the device in a secure memory area as well as supporting software functions for secret material and memory management.
Crypto Management	Management functions to be supported by the device, including generation and validation of keys as well as updates of keys as part of their lifecycle management or even update of the actual crypto implementation.
MANUFACTURING	Functions to ensure efficient and secure management of crypto functions during device manufacturing process, including support for unique scenarios like third party manufacturing:
Provisioning	Of keys or key pairs in such a way that secret material (symmetric keys or asymmetric private keys) is not exposed or could be compromised during the production process.
Hardware Root of Trust (RoT)	To establish the foundation on which all secure operations rely on.
Safeguards	Of cryptographic secrets and processes through secure supporting infrastructure and secure processes.
OPERATIONS	Functions to be designed into the device:
Key Lifecycle	Device-side support of cryptographic key management functions such as revocation or updating.
Exception and Error Handling	Is a set of device- and use case-specific functions that define how exceptions (e.g., device being off-line) or errors (e.g., failure of cryptographic key verification) are handled and logged.

CRYPTOGRAPHIC FOUNDATION AND INFRASTRUCTURE

No matter what the cryptographic function, security use case, or device design, a certain set of supporting cryptographic functions is required and design decisions need to be made accordingly.

PKI DESIGN	Defines the infrastructure and design decisions to support and manage asymmetric cryptography.
CA Root of Trust (RoT)	Defines the ultimate trusted root certificate provided by a trusted Certificate Authority.
PKI Design	Defines the structure and layers of certificates that are used across device types but also for different crypto functions within a given device. Good crypto design provides functional separation between different operations that require different levels of security. E.g., when implementing code signing, software should be signed with a different key than the one used to facilitate transport layer security.
PKI Implementation	Ability to ensure secure provisioning of devices in production. The infrastructure used to manage the provisioning process for a device should be equally cryptographically protected.
CERTIFICATE DESIGN	Defines the structural capabilities of a certificate, including:
Certificate Structure	Including organizational and functional mapping to enable maximum security and maximum flexibility on how certificates get managed during their lifecycle.
Certificate Format	That include, in addition to the cryptographic key, information pertaining to the management of the key, including issuer or expiration date. Other, use case specific information, can be included as well.
CRYPTO DESIGN	Scoping to determine:
Crypto Algorithm	<p>As determined by use case, hardware capabilities, and desired level of security. The most fundamental decisions being between the use of symmetric (less resource intensive and faster but challenging to ensure security) or asymmetric (more resource intensive and slower but delivering several practical and security advantages) encryption. Although asymmetric cryptography provides a clearer path to stronger trust, symmetric may be advised in some resource restrained situations, but requires careful consideration (e.g., protection of the shared symmetric key). Using a combination of asymmetric and symmetric (e.g., use of ephemeral keys) can be a viable solution.</p> <p>Within each main category a number of different crypto algorithms and implementations are available and during the design process the advantages and long-term maintainability need to be carefully considered.</p>
Key Strength	Is another factor that determines the strength of the selected cryptography. In general, the longer the key the more difficult it becomes to break. Specifics depend on the chosen algorithm.
Crypto Management	External functions and infrastructure required to manage cryptographically protected devices, including key revocation and lifecycle management or updates of a given cryptographic implementation.

COMMON MISTAKES FOUND IN CRYPTOGRAPHIC IMPLEMENTATIONS

Certain design compromises should be avoided if at all possible, or if hardware or use case leave no other choice, any accepted compromise or risk needs to be the result of careful deliberation rather than indiscriminately accepting compromise without understanding its implications or evaluation of alternatives.



Examples for common mistakes in cryptographic design and implementation include:

- Design or resource constraints may lead to the decision to use symmetric keys without assessing the use of alternatives, e.g., an approach that would use a public/private key pair to generate a shared secret that supports individual sessions using symmetric encryption.
- Not accounting for the risks resulting from the need to protect secret materials (e.g., when using symmetric cryptography).
- Accommodation of perceived operational priorities (e.g., sharing of secret keys material across organizational entities).
- Use of a single key to support multiple security functions with different levels of risk, criticality, or lifecycle needs.
- Use of the same key across multiple devices to address economical or logistical constraints, leading to the risk of larger quantities of devices being exposed if a single key is cracked or compromised.
- Failure to manage key lifecycle and provide management features to support secure key generation, provisioning, storage, distribution, use, and destruction.
- Failure to anticipate need for crypto algorithm and library updates to account for to-be-expected algorithm depreciation.
- Use of cryptographic concepts and architectures from the non-device space, e.g., web browser-server communication or general IT use cases that don't account for device unique requirements.

CRYPTOGRAPHY IN RESOURCE-CONSTRAINED DEVICES

Many medical devices need to be designed with serious resource-constraints in mind, i.e., based on their use case they need to meet design requirements that limit physical size, need to operate on limited battery capacity, and/or need to guarantee longevity of the device. Examples for such devices are implantables, e.g., pacemakers or neurostimulators, or patient-worn and operated drug delivery systems like insulin pumps and pens. Based on their clinical use case, these devices need to be small and lightweight and need to operate reliably for many years as, for example, replacement may require surgical procedures.

These use case constraints often lead to hardware choices with severe limitations, e.g., low-power microcontrollers, limits in memory size, or low-capacity batteries. Also, they may result in design and implementation decisions such as lower clock frequency or optimizing power consumption through sleep mode.

Adding cryptography to such a device creates a number of challenges and may lead to trade-off decisions between desired level of security (e.g., crypto strength) and device properties (e.g., processor capacity and speed). Examples for conflicts are:

- The need for additional code and compute cycles to support a crypto function vs. the resulting timing and energy needs.
- The need for higher speed and capability MCU to support crypto functions vs. the additional power and space such a chip would require.
- The need for more memory to support crypto code and storage of crypto keys vs. the additional space and power required by the added memory.
- The avoidance of sign/verify operations to facilitate trust operations as these tend to be much more computationally expensive and less likely to be supported by hardware acceleration in smaller devices.
- The reduction of low-energy sleep cycles to support crypto functions vs. the increase in battery use and reduction in device life.

Implementing crypto is computationally expensive for constrained devices and in many cases this requires a careful tradeoff between desired level of security (as provided by the crypto function) and device design and functional constraints - and sometimes this decision will result in the implementation of a lower level of security. This is understood by regulators and MDMs are not expected to implement the highest level of security at any cost and to the detriment of the device's safety and effectiveness. However, such risk-tradeoffs should not be made lightly, require careful analysis, may lead to the use of



compensating controls, and should be documented for the user and operator as well as to withstand regulatory scrutiny.

Resource constraints may lead to improper or hasty design decisions that can lead to new security vulnerabilities, as security researchers have demonstrated. E.g., implantable devices can be overwhelmed with a DDoS-type attack that would prematurely deplete its battery. Or, a manufacturer may decide to implement symmetric encryption (as it is less computational intensive) with the shared secret key for each device stored in a cloud database that, if compromised, would compromise each and all deployed devices.

Often, design alternatives exist and can be implemented to deliver the right level of security under the given design constraints, e.g., instead of using symmetric keys the use of an ephemeral key to establish shared secrets to support a secure communication session may be a viable approach. With the right expertise it can be ensured that such trade-off decisions are reduced to a minimum, if required, still provide for the highest level of security possible under the given circumstances.

CONCLUSION

After reviewing the above, the obvious question is “what now”? As an actionable, leading practices approach to “getting security right” it is generally recommended to:

- Start by generating a threat model.
- Identify the risks that are best mitigated by cryptography (see “Security Functions Provided by Cryptography”).
- Identify the constraints your device has that may impact your crypto design decisions (see “Cryptography in Device Design”).
- Test the highest-level crypto implementation your hardware can support, and ensure that your performance criteria are met (as discussed in “Cryptography for Resource-Constrained Devices”).
- Design your manufacturing and lifecycle processes in a way that properly accommodates key generation and management (see “Cryptographic Foundation and Infrastructure”).
- Ensure that trust / PKI / root-of-trust aspect are properly considered in the overall cryptographic architecture and implementation.
- Certificates and certificate-chains provide the level of trust for the keys so to ensure a trusted base for all of the cryptographic security.
- Finally, document your design, and the ways in which it controls for the risks you identified in your threat model.

Cryptography is a powerful methodology that can be used to realize many important security functions. However, crypto design requires careful consideration and experience to ensure that the chosen implementation is technically sound, delivers the desired level of security, is future-proof, and meets the goals of the business.

It is critical that MDMs (Medical Device Manufacturers) properly apply authentication, authorization, encryption, and signing services while also maintaining the availability and functionality of the devices and of the data that is associated with. A PKI-based cryptographic solution ensures strong identity and message integrity for data exchange between applications, users, devices, and systems.

MedCrypt provides proactive security for healthcare technology. From top device manufacturers to startups, we work with medical device manufacturers of all sizes to help secure their products, ensuring devices are secure by design. After a successful Series B fundraising round in 2022, MedCrypt has raised more than \$36 million in funding to date with participation from Johnson & Johnson Innovations, Intuitive Ventures, and Dexcom Ventures. The company is based in San Diego, California.

For further details, please visit and contact:

Website: www.medcrypt.com

Email: info@medcrypt.com