DOOR LOCK SECURITY BEST PRACTICES

Version 1.0

Publication Date: 10 February 2017

About HTNG

Hotel Technology Next Generation (HTNG) is a non-profit association with a mission to foster, through collaboration and partnership, the development of next-generation systems and solutions that will enable hoteliers and their technology vendors to do business globally in the 21st century. HTNG is recognized as the leading voice of the global hotel community, articulating the technology requirements of hotel companies of all sizes to the vendor community. HTNG facilitates the development of technology models for hospitality that will foster innovation, improve the guest experience, increase the effectiveness and efficiency of hotels, and create a healthy ecosystem of technology suppliers.

Table of Contents

1 DOCUMENT INFORMATION 4

1.1 DOCUMENT HISTORY 4
1.2 DOCUMENT PURPOSE 5
1.3 SCOPE 6
1.4 AUDIENCE 6

2 SECURITY CONSIDERATIONS FOR DOOR LOCK SYSTEMS 7

2.1 DOOR LOCK SOLUTION CRITERION 7
2.1.1 General System Information 7
2.1.2 Physical Lock 8
2.1.3 Keys 9
2.1.4 General Safety and Reliability 10
2.1.5 Emergency Ingress – Guest in Duress Inside Room 11
2.1.6 Emergency Egress – Guest Emergency Evacuation 11
2.1.7 Tamper Resistance 11
2.1.8 Brute Force Resistance 11
2.1.9 Maintenance and Obsolescence 12
2.1.10 Incident Investigation 12
2.1.11 Key Management 12
2.1.12 Third Party Integrations 12
2.1.13 Mutual Certification 13
2.1.14 Interface Types 13
2.1.15 Supply Chain (includes update process) 14
2.2 MOBILE KEY SOLUTION BEST PRACTICES 14
2.2.1 Objectives 14
2.2.2 Known Vulnerabilities 15
2.2.3 Architecture, Design, & Features 16
2.2.4 Third Party Integrations 18
2.2.5 Supply Chain 23
2.2.6 Mobile Key Distribution 23
2.2.7 Security 24
2.2.8 NFC Communication 27
2.2.9 Bluetooth Low Energy Communication 27
2.3 ONLINE DOOR LOCK END POINTS 27
2.3.1 Objectives 29
2.3.2 Architecture & Design 29
2.3.3 Online Network Infrastructure 29
2.3.4 Wired Infrastructure Standards 32
2.3.5 Wireless Infrastructure Standards 32
2.3.6 Wireless Architecture Standards 33
2.3.7 Third Party Integrations 34
2.4 PHYSICAL KEY SOLUTIONS 34
2.4.1 RFID vs Magnetic Key Mechanics 34
2.4.2 RFID Key Security 35
2.4.3 Authentication 35
2.4.4 Data Encryption 36
2.4.5 General Contactless Smart Access Card Requirements 38
2.4.6 Contactless Smart Access Card Options, Considerations and Recommendations 39
2.4.7 Minimally Featured Solutions 39
2.4.8 Legacy Systems 40
2.4.9 Modern Highly-Secure Basic Solutions 40
2.4.10 Modern Multi-Application Solution 40

2.4.11 Comparison Matrix *40*
2.5 THIRD PARTY INTEGRATIONS 41
2.6 GLOSSARY OF TERMS 42
2.7 IMPLEMENTATION NOTES 42

3 APPENDIX 44

3.1 THREAT MODEL 44
3.2 SECURE DESIGN PRINCIPLES 45
3.2.1 Overview *45*
3.3 ANTI-DESIGN PRINCIPLES 46
3.4 SECURITY PROGRAM 47
3.5 SYSTEM LEVEL SECURITY BEST PRACTICES 49
3.5.1 Network Security *49*
3.5.2 Application Security *50*
3.5.3 Input Validation *51*
3.5.4 Output Encoding *52*
3.5.5 Authentication and Password Management *52*
3.5.6 Session Management *54*
3.5.7 Access Control *55*
3.5.8 Cryptographic Practices *57*
3.5.9 Error Handling and Logging *57*
3.5.10 Data Protection *58*
3.5.11 Communication Security *59*
3.5.12 System Configuration *59*
3.5.13 Database Security *60*
3.5.14 File Management *61*
3.5.15 Memory Management *61*
3.5.16 General Coding Best Practices *62*
3.5.17 Update Process *63*
3.5.18 The Human Element *63*
3.5.19 Validation *64*

1 Document Information

1.1 Document History

Version	Date	Author	Comments
0.10	5 April 2016	Armand Rabinowitz, Sandy Loeffler, Grant White	Initial Best Practices outline created
0.25	13 May 2016	Workgroup	Additional Sections Added Based on Initial Outline
0.40	5 June 2016	Gary Gill	Key Card Best Practices Updates
0.45	19 July 2016	Armand Rabinowitz, Sandy Loffler, Grant White	Operational Considerations updates
0.50	20 August 2016	Workgroup	Updates and comments based on in-person meeting
0.50	25 August 2016	Mattius Wallnius	Mobile Key Security updates
0.55	10 September 2016	Eitan Avni, Christoph Zwahlen	Contactless / Key card updates
0.60	1 October 2016	Ted Harrington, Armand Rabinowitz	Mobile Key and Software Security best practices updates
0.75	4 November 2016	Workgroup Members at In-Person Meeting	Richard Moore, Cris Davidson, Gary Gill, Luis Saldana, Mike Lopes, Lisette Kaiser, Stig Lagerstedt, Philip Roosli, Michael Maxwell, Rohan Jani, Ted Harrington, Armand Rabinowitz, Patrick Dunphy edit and reorganize major document sections
0.85	5 December 2016	Armand Rabinowitz, Patrick	Hotel operational considerations, security best practices, additional reorganization of content

		Dunphy, Ted Harrington	
0.90	10 December 2016	Ted Harrington	Numerous security best practices updates
0.95	14 December 2016	Workgroup	Numerous updates and edits based on formal commentary
0.98	3 January 2017	Emily Wilson	Copy edit for publication
0.99	15 January 2017	Patrick Dunphy	Numerous minor changes in preparation for vote
1.0	20 February 2017	Workgroup	Door Lock Security Workgroup Vote – Final

1.2 Document Purpose

This document is intended to provide high level guidance for how to **securely** build and deploy electronic door lock systems, including but not limited to **mobile key** and **online locking systems** in a hospitality setting. It is designed to provide both hoteliers and vendors with a common language for how to discuss security with each other, in order to pursue the common mission of risk management.

This document is not intended to be prescriptive, nor perceived as either standards or a compliance framework.

NOTE: The best practices contained herein should be considered neither exhaustive, nor prescriptive. Every organization is unique, even those ostensibly competing and offering similar solutions to the same customer base. As every organization is unique, every organization's attack landscape will be unique, and thus every organization's defense paradigm is unique. As such, every organization needs to consider, in the context of their unique threat model, which elements of the following best practices are most applicable, which are not applicable, and which additional practices may be critical despite not being contained herein.

1.3 Scope

This document is the second deliverable published by the Door Lock Security Workgroup (DLSWG), within the trade organization Hotel Technology Next Generation (HTNG). DLSWG's first deliverable was *Threat Model: Emerging Locking Systems,* published on 31 August 2015. The best practices outlined in this document rely heavily on the guidance articulated in that document; it should be considered prerequisite reading prior to digesting this document.

1.4 Audience

The intended audience for this document encompasses a handful of parties, including:

- Hoteliers should use this document and best practices to understand the door lock system (and its components) and the security issues related to these systems.
- Locking system vendors should use this to understand the security principles of their products and services, and to ensure solutions meet the needs of the hospitality industry.

2 Security Considerations for Door Lock Systems

2.1 Door Lock Solution Criterion

In this section we deal with the attributes of the door lock itself. Room keys and back-end systems are dealt with later in the document. There are a number of factors to consider when choosing an appropriate guest room lock including:

- **Physical Security Attributes**

  The lock must be capable of securing a room properly, and it must resist a rudimental, but not sustained physical attack.

- **Hotel Operational Requirement**

  Ensure the lock solution meets your hotel's specific operational requirement and organizational strategic vision as it relates to management of guest keys, staff keys, room changes, future trends including, but not limited to, Mobile Key and other form factors.

- **Safety**

  Door lock solution must meet the local fire and life safety codes and the brand security guidelines and policies.

- **Guest Data Privacy**

  Is it a best practice to NOT put any guest data or credit card data on the card. Keep the lock system out of scope. If the door lock solution is capturing guest personal data such as Personally Identifiable Information (PII), then it is required that the solution is payment card industry – data security standard (PCI-DSS) compliant and meets other stringent data security requirements.

2.1.1 General System Information

Before handover of the door lock solution, ensure:

- Adherence to prevailing building codes and requirements issued by authorities having jurisdiction and in compliance with latest brand standards and specifications.
- It is audited (audit assessment) by a third party independent security firm recommended by the brand or organization.
- Use this best practices as a guide for the audit (audit assessment) process
- Approval by standards institute with regional authority or recognition such as:
  - UL – Underwriters' Laboratory
  - FCC – Federal Communications Commission
  - CE – Conformite Europeenne
  - ISO 14001in applicable regions
  - NFPA – National Fire Protection Agency
  - NEC – National Electric Code

Note that standards compliance is not the end of a security discussion. Secure design anti-principles at stake include:

- **Compliance**. Although commonly perceived as one, compliance is not a security measure. Compliance only works if the enemy you are trying to thwart is the compliance auditor. Against any other enemy, compliance does not effectively defend.
    - **Door locks, and access control in general are subject to numerous compliance and code issuing bodies**. Do not mistake compliance with heightened or acceptable security; adherence to a codified set of rules is a baseline, not an end-goal. A door lock system may be compliant at the factory, but non-compliant if implemented poorly. Further, compliance bodies are not capable of responding quickly to rapid developments in information security
- **Security Through Legality**. Regulation and law do not prevent an attack, nor effectively outline measures to be effective in all cases.
    - Legal regulations and law often provide remedy or justice after an attack, but do not prevent breaches nor abused access control systems.

2.1.2 Physical Lock

The following specific best practices and implementation guidelines build on the general principles detailed earlier in this whitepaper:

- Identify all third parties, vendors, or other stakeholders with access to your environment; identify, understand, and mitigate risk with each.
    - Door lock systems are a series of interconnected parts; the system itself is only as secure as the weakest element in it.
    - Vendors implement common “standards based approaches” in different ways.
- Collect and analyze detailed logs.
    - Audit logs are a necessity for access control purposes (especially for key usage).
    - Software audit logs for key issuance are also important.
- Restrict or minimize the use of removable media (such as USB drives) wherever practical.
    - Locks should be secured for only authorized individuals to access for maintenance or support purposes.
- Protect master secrets from unauthorized access.
- All random numbers, file names, GUIDs, and strings should be generated using the cryptographic module’s approved random number generator when these random values are intended to be un-guessable.
- Cryptographic modules used by the application should be compliant to FIPS 140-2 or an equivalent standard. (See http://csrc.nist.gov/groups/STM/cmvp/validation.html)
    - Not all cryptographic methods are secure – encryption by itself does not guarantee security.
- Establish and utilize a policy and process for how cryptographic keys will be managed.

Lock specific best practices include:

- The door lock must meet the minimum product life expectancy including spare parts and associated maintenance as specified by the brand or organization.
- Repurposed locks, parts or other systems need to go through a refurbishment process and be re-audited
- Ensure that the door lock solution is tamper resistant from both sides. Any exposed ports must be secured.
- Ensure the door lock meets the environmental testing requirements.
- All portable device interactions with door lock (programming, interrogation, emergency override, etc.) must be traceable (with changelog) with transaction history report.
- Ensure that either manual or automatic deadbolt mortise is provided.
- The system must be capable of overriding the deadbolt. The method used for override must be highly secure and traceable.
- Auxiliary latch/anti-pick required to prevent tampering of primary latch.
- When deadbolt is engaged, all keys must be inoperable (except emergency keys). The deadbolt must have the ability to be customized based on brand or organizational requirements.
- From a security standpoint, RFID is strongly recommended over magstripe. RFID tends to have lower maintenance and support costs, and supports future technology compatibility. Magstripe cards do not use standards based encryption, nor authentication. There is no way to prevent a magstripe card from being copied. This is a significant problem for master keys. Section 3.1.3 covers keys in more detail.
- Door locks should have all various options available to meet compliance requirements, for example ADA, as it relates to lock levers.

2.1.3 Keys

The following specific best practices and implementation guidelines build on the general principles detailed earlier in this whitepaper:

- **Reduce Asset Handling.** Do you really need to collect that personally identifiable information, just because the marketing department asked for it? If you collect fewer assets, you reduce the reasons an adversary may want to attack you.
    - Reduce the information stored on a key or lock system to reduce risk and loss in the event of a breach or attack.
- **Implement role-based access controls.** Users and applications acting on behalf of users, should be limited to only those permissions required for accessing the services/devices they need. The ability to unlock a device should not also mean the user can reset a key or lock identifier.
    - Guests should only access their guest room and common areas (lobby, parking garage, elevators, etc.).
    - Front desk staff should only be able to issue guest room keys (not master keys).

- **Implement mutual authentication.** Locks obviously need to authenticate the user trying to unlock it. However, it's just as important that the user authenticates the device they are trying to unlock so they don't inadvertently pass credentials to an attacker.
    - Guests and staff can inadvertently leak sensitive data by attempting to utilize a tampered lock system or an unapproved system.
- **Encrypt in transit.** Keys and other secrets should never be sent over a plain-text channel. Encrypt key or sensitive data at all times.
- **Consider multi-factor authentication for staff facing (or other high security) areas of the hotel.**
- **Authentication failure responses should not indicate which part of the authentication data was incorrect.** For example, instead of "Invalid username" or "Invalid password", just use "Invalid username and/or password" for both. Error responses must be truly identical in both display and source code. Reduce the amount of information an attacker can glean from your system through failure messages.

Key specific best practices:

- The key should not store PII or PCI data of the guest (or staff).
- Encrypt or hash keys with a tested, public and peer-reviewed authentication protocol and encryption mechanism as described in Section 4.4.2.1.
- Protect keys from short and long distance remote cloning.
- A new guest key upon usage must invalidate all previous guest keys.
- Invalidate a lost staff key by rekeying all staff keys sharing the same group of staff keys or preferably the individual.
- Invalidate Installation or Construction Keys provided by the vendor post installation.
- Locks and encoders from a vendor may contain vendor proprietary authentication or encryption keys (not access key cards), but there must be an option to replace encryption or authentication keys with hotel or brand specific keys.
- Manual key overrides introduce additional complexity and risk, but some local regulations (or owner/operator standards) require it. If manual key overrides are needed, auditing and logging may be necessary to mitigate risk.

2.1.4 General Safety and Reliability

- The lock should have been subject to fire testing and approved for use with a fire rated door i.e. fitting requirements should not hinder the fire resistance capability of the guest room door.
- The lock should be equipped with static electricity protection (ESD) to minize the chance of guests receiving static shocks and malfunction of lock components.
- Some local regulations require recertification after door lock hardware has been changed.

2.1.5 Emergency Ingress – Guest in Duress Inside Room

- The lock should have an emergency override system to provide ingress in the event that the door has been locked using the secondary latch bolt mechanism from the inside.
- If the lock should include a mechanical key override, the key blanks for this cylinder should be restricted and not commercially available.
- The cylinder should ideally be re-keyed without removing from the door if a master key is lost. The use of any mechanical override key should be recorded as an event in the electronic audit trail.

2.1.6 Emergency Egress – Guest Emergency Evacuation

- For emergency exiting, there should be a simultaneous retraction of the deadbolt and any other secondary mortise bolts with a single turning action of the handle.

2.1.7 Tamper Resistance

- The lock should be tamper resistant with no accessible parts from the corridor side when the door is closed – this prevents dismantling and unauthorized manipulation.
- If the inner working of the lock is exposed, no cutting or short-circuiting of any or all the wires should allow disengagement of the locking mortise.
- The lock should be immune to the effects of electrical or magnetic pulses, strong magnetic fields and electromagnetic discharge, in that, if they are applied to the lock it maintains its status (i.e. remains locked).
- Protection against electrical, magnetic pulse, magnetic fields and electromagnetic discharge must be provided.
- UL 294: make sure to see Section 3.8 for existing standards in referenced documents.
    a. May need to differentiate between guest room locks vs. back of house locks
- The battery pack must be accessible for maintenance.
    a. If the battery is on the front of the door (public access), this exposes a risk of vandalism or loss of battery (batteries also tend to last longer on the interior of the door).

2.1.8 Brute Force Resistance

- The lock should have a locking mortise of a minimum length of 20mm to allow for door frame tolerances.
- There should be an auto-dead latch fitted, this dead locks the primary mortise when the door is closed, preventing plastic carding attacks. Locks should be fitted correctly to allow for door frame tolerances to ensure the auto-dead latches engage properly.
- As part of the latch set, there should be a guest-operated turn-piece inside the guestroom operating a secondary mortise bolt. When this turn-piece is engaged, the lock should be programmed so that all keys, except any emergency keys, are rendered inoperable. Where

locks have a separate 'do not disturb' function rather than a secondary mortise; the lock should be programmed to make keys inoperable in the same manner as above.

2.1.9 Maintenance and Obsolescence

- An acceptable warranty should be made available given the life expectancy of the product.
- Products should have an acceptable service-life (support), where updates and replacement parts are readily available. You don't want to find yourself with an unsupported system in years to come.
- The lock should have a low-battery indicator. This prevents battery failure, keeps rooms operational and avoids poor guest experiences.
    - Ideally the lock should use standard, readily available batteries without the need for proprietary battery packs or sets. The batteries should be easily replaceable without requiring the need to remove trims or escutcheons.
    - Battery changes and internal clock updates should be easily managed by hotel staff. Battery changes should take no longer than 10 minutes.

2.1.10 Incident Investigation

- The lock should have an audit Interrogation feature that captures at least the last 1000 room entries and details, including date, time and Key ID. Larger possible audit entries may require online systems. Individual products should be investigated for audit details.
- If portable programming devices, or other non-key/card devices are used to access, program, or otherwise control or interact with locks, users should authenticate with a unique ID and/or password that is auditable to track their usage.

2.1.11 Key Management

- Lock systems should have multiple customizable access levels to provide guests and staff with access to parts of the hotel according to their access rights.
- The system should be able to place time and zone restrictions on the use of each key so that guests and staff are only allowed access to certain parts of the hotel at certain times of day.
- The system should be capable of programming duplicate keys for a room, but each duplicate key should have a unique ID.
- The system should void all previous rental keys at each new guest occupancy (not masters). This prevents a guest from re-entering a room after someone else has taken up occupancy.

2.1.12 Third Party Integrations

Many third party systems are commonly integrated with door lock systems for various purposes. The most common systems include:

- Property Management Systems (PMS)
- Parking Management Systems

Optional integration systems include:

- Energy Management Systems (HVAC and Lighting Control)
- Point of Sale Systems (POS)
- Time and Attendance
- CCTV, Security
- Building Management
- BOH Access Control
- Data Warehouse

2.1.13 Mutual Certification

All interfaces between two systems should be certified by both vendors responsible for supporting and maintaining the interface. The Certification requirements should specify the regular interval upon which the system vendor and third party will make necessary updates to the system to ensure security and compatibility. There should also be an agreement on how to handle feature improvements.

2.1.14 Interface Types

There are three common types of interfaces used to integrate lock systems with third party systems.

Legacy

- Serial (RS-485, RS-232) interfaces are simple
- Best suited when the integrated systems occupy the same physical location and can be connected through a single cable
- Must be no sensitivity of data or risk of data interception because data is transmitted unencrypted between communication points
- Limited ability needed to monitor health and security of the interface

Network

- TCP/IP interfaces leverage local Internet Protocol (IP) network communication
- Suitable for systems that occupy the same local network address space, typically connected to the same network switch or behind the same network router or firewall
- Data is usually sent unencrypted and relies on the security profile of the local network and policy requirements for devices to join the network and intercept data transmissions
- Advanced features available to monitor health of the network

- Limited ability to confirm identity of communication partner, opening the possibility of man-in-the-middle attacks or other devices sending privileged commands or instructions to a system

Web

- Web Service interfaces are the most versatile
- Systems can be located anywhere as long as they have internet access
- Communication can be optionally encrypted using many available open-source technologies based on standards
- Multitude of methods available to validate communication pairs to ensure authenticity of data and authority of commands

Based on the advantages described above, Web-Service Interfaces should be required for all lock systems. Legacy and Network TCP/IP interfaces should be phased out and disabled entirely to reduce attack surfaces unless tight controls are in place to secure the IP network.

2.1.15 Supply Chain (includes update process)

Door Lock Systems (including applications and component manufacturers) are expected to support their products for a reasonable period of time (including software updates).

2.2 Mobile Key Solution Best Practices

Mobile Key solutions are a recent innovation in door lock security. Mobile Keys allow guests/staff the ability to utilize smart devices to access rooms instead of using traditional magstripe and RFID technology. Mobile solutions allow for additional guest convenience, in addition to enhanced security functionality such as new or more granular audit features, remote revocation of credentials, and integration with hotel loyalty applications. Along with this new ability, it introduces new attack surfaces that must be considered.

2.2.1 Objectives

The objective of these best practices are to outline general considerations for mobile key solutions that can be used when evaluating competing solutions to ensure they meet your requirements for security, operations, third party integrations, or functional requirements for guest experience.

In addition to sections following, general considerations for a mobile key solution include:

- **In Coverage/Out of Coverage.** Mobile keys shall operate with periodic network coverage. It's reasonable to need network coverage periodically to download or retrieve a mobile key, but once loaded the mobile key should be able to operate with or without network coverage, for example in areas where Wi-Fi coverage is spotty or non-existent.
    - **Trust Reluctance.** Assume all trusted parties (including users, trusted employees, integrated third parties, etc.) are or can become malicious; architect defenses accordingly.
    - **Economy of Mechanism.** Simplicity is the friend of security. From a security perspective, more simple systems are easier to build, validate and maintain.

    - **Complete Mediation.** Every access to every resource and/or asset must be validated for authorization.
- **Online/Offline Systems.** Mobile keys should operate with either type of system whether the locks are online or offline. Online locks are available from a number of vendors. Normally this is used as a maintenance tool to collect statistics and to allow or block access to locks. Most of these locks are online via wired connections. Offline locks do not have any connection to the lock management system. Offline locks are usually lower cost than online locks and have longer battery life. An offline lock will read a card and decode the card data before opening the lock.
    - **Reduce Asset Handling.** Do you really need to collect that personally identifiable information, just because the marketing department asked for it? If you collect fewer assets, you reduce the reasons an adversary may want to attack you.
- **Coexistence of Mobile Keys and Traditional Keys.** Not all guests will adopt mobile keys initially. Operations may be limited for various reasons. For example, not everyone may have mobile devices. Mobile Key solutions shall support and coexist with all physical card use cases, including the ability to use a mixture of physical card interactions (i.e. previous guest, next guest, cancel cards, blocking cards, etc.) with mobile keys.
- **User Intent.** Since mobile key solutions are RF based, there is a need to correctly interpret each user's intent in various situations. For example, if a user is inside the room, near the room or near multiple rooms. Mobile key solutions shall not cause false intent and open doors when someone doesn't want them to.
    - **Psychological Acceptability.** If security becomes too intrusive for a user to effectively perform his or her role, the user will circumvent the security controls. Psychological Acceptability balances security and convenience. After a certain degree of inconvenience, security will actually be undermined by the user.
- **Feature Considerations**. Mobile Key solutions from different vendors will have differentiating features, for example, hands-free operation, user experience (NFC wake up vs. BLE only*)*, battery life and audit functionality. While these features provide for differentiated solutions, all of them should still follow these best practices where applicable.
    - **Secure the Weakest Link.** Just as defenders calculate return on resource investment, so do adversaries. Modern adversaries choose the weakest link in the security chain, as the easiest path to system compromise.
    - **Build Security In.** It is both more effective and less expensive to build security into each stage of the development process, rather than considering it at the end.

2.2.2 Known Vulnerabilities

The following vulnerabilities were current as of the time of this writing, but new vulnerabilities will arrive every day.

The following article from Hackaday notes that many ubiquitous Bluetooth locks are easy to hack and states you need to make sure the locks have AES or another encryption. This article also states that many of the Bluetooth locks entering the market don't have the security precautions they should.

- http://hackaday.com/2016/08/08/the-terrible-security-of-bluetooth-locks/

Here is a collection of some published vulnerabilities that are good reference points for developers of mobile key solutions. These examples may not exactly fit the problem set of any given app developer, but should instead be seen as a method to provide context and inspiration for how to learn from mistakes that occurred elsewhere.

- http://blog.trendmicro.com/let-get-door-remote-root-vulnerability-hid-door-controllers/
- http://blog.perfectlylogical.com/post/2015/03/29/Making-Smart-Locks-Smarter
- http://ns.umich.edu/new/multimedia/videos/23748-hacking-into-homes-smart-home-security-flaws-found-in-popular-e-alarms/
- http://www.digitaltrends.com/home/bluetooth-smart-locks-easily-hackable/
- https://github.com/merculite/BLE-Security/

2.2.3 Architecture, Design, & Features

The main components of a mobile key solution include:

- Credential Service: *On Premise, Cloud based, or Hybrid*
- Mobile Platform: *Full app, or mobile key library for integration with other apps*
- Lock Access Controller: *Full lock solution, or mobile key module for integration by lock vendors*

The following figure shows these basic conceptual components to a mobile key solution:

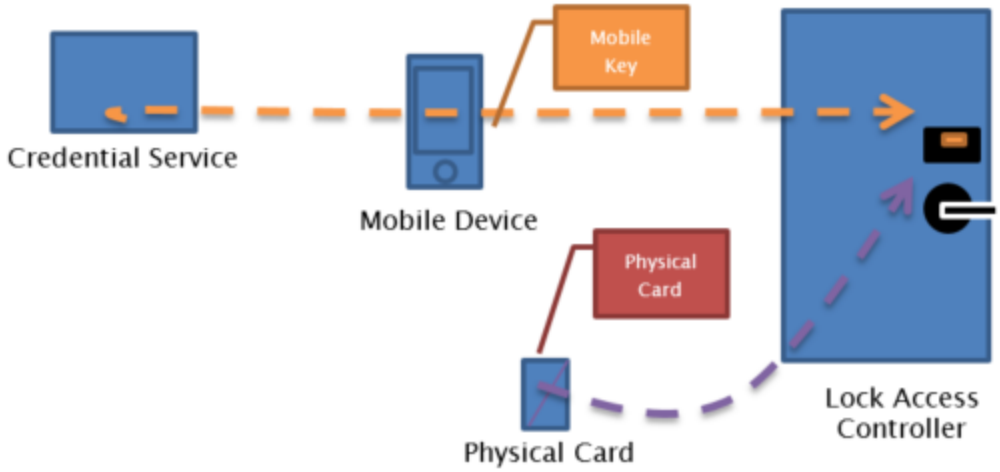


Figure 1 Credential Delivery Methods for Key Systems

While the details and specifics vary, mobile key solutions operate by distributing a credential from a credential service to a mobile device. This sends the credential to a lock after a guest signal's their intent to open a door and the credential is then validated before opening the lock. These systems coexist with existing Lock Vendor systems encoding traditional physical cards that a guest can also use to open their room door.

The following design principles apply to this system:

- **Least Privilege.** Allow a user only the absolute minimum access required in order to successfully perform his or her function, and nothing more.
- **Privilege Separation.** Divide privileges so that a user must have multiple privileges in order to perform a larger scale compromise. This requirement of additional privileges reduces risk.
- **Least Common Mechanism.** Shared resources introduce shared compromise, and as such, wherever possible, an organization should reduce or eliminate shared attack surfaces.
- **Fail Secure.** Systems fail, and those building systems should plan for failure. In the event of failure, the system should default to a secure state.
- **Secure the Weakest Link.** Just as defenders calculate return on resource investment, so do adversaries. Modern adversaries choose the weakest link in the security chain, as the easiest path to system compromise.

The guest experience of using a Mobile Key solution is impacted by the specific features and design choices made by the Mobile Key technology provider. For example:

- Some solutions require a phone to be placed next to a lock in order to wake-up the lock and activate Bluetooth. This can unify the user experience between BLE and NFC and can be natural for guests that are familiar with RFID card transactions.
- Some solutions allow a guest to open their door from a distance by pushing a virtual 'unlock' button on the phone interface to open the door. This provides for explicit user intent and is natural for people who use smart phones.
- Some solutions are hands-free where the phone can remain in the Guest's pocket, providing a similar experience to keyless entry in the automotive industry.

The following secure design principle applies:

- **Psychological Acceptability.** If security becomes too intrusive for a user to effectively perform his or her role, the user will circumvent the security controls. Psychological Acceptability balances security and convenience, as after a certain degree of inconvenience, security will actually be undermined by the user.

Some specific questions to ask to best match a mobile key solution to your operational requirements include:

- What is the expected battery life for a lock?
- What is the guest experience for opening the lock with a Mobile Key versus a traditional plastic key?
- Does the guest experience for opening a lock match your brand's target demographic or meet the experience needs for your brand?
- Does the solution work with multiple lock vendors or just one lock vendor?
- Does the mobile device need to have network coverage in order to open the door?
    - If the solution does provide for out-of-network use, for how long can the mobile device be offline and still open a door?
- Does the mobile key solution provide capability for you to integrate your own mobile application to a mobile key library or framework so that guests can use your loyalty application to open their door?
- Do you need an out-of-the box solution where the mobile app is already provided by the Mobile Key vendor or another third party?
    - Some solutions utilize a Hotel Loyalty Application for guest interaction and the Hotel Loyalty Application integrates a mobile key library. This provides a familiar and unified interface to the guest for their hotel stay.

2.2.4 Third Party Integrations

Mobile key solutions should provide third party integration points. Regardless of the integration technology used, solutions should utilize the industry agreed upon standard (most secure). While best practices for each use case or integration interface are noted, not all hoteliers or hotel properties will require all of these integration points.

The following secure design principles apply to each use case and best practice:

- **Complete Mediation**. Every access to every resource and/or asset must be validated for authorization.
- **Least Common Mechanism**. Shared resources introduced shared compromise, and wherever possible, an organization should reduce or eliminate shared attack surfaces.
- **Fail Secure**. Systems fail, and those building systems should plan for failure. In the event of failure, the system should default to a secure state.
- **Privilege Separation**. Divide privileges so that a user must have multiple privileges in order to perform a larger scale compromise. This requirement of additional privileges reduces risk.
- **Implement Role-based Access Controls.** Users and applications acting on behalf of users, should be limited to only those permissions required for accessing the services/devices they need. The ability to unlock a device should not mean the user can reset a key or lock identifier.
- **Implement Mutual Authentication.** Locks obviously need to authenticate the user trying to unlock it. However, it's just as important that the user authenticates the device they are trying to unlock so they don't inadvertently pass credentials to an attacker.
- **Expire Session Keys.** Devices should not use static keys for unlocking. While they may have a unique shared secret used for authenticating with a key server, unique session tokens should be used for unlocking devices, and these tokens should expire in a short amount of time.
- Establish and utilize standard, tested, authentication services whenever possible.
- Use a centralized implementation for all authentication controls, including libraries that call external authentication services.

Integration Use Case	Best Practices
Track 2 – Parking Garage	For physical card locking systems, hotel guests carry only one physical card to open both their guest room and in some cases, to access parking garages. In magnetic stripe card systems, one track of magnetic data (e.g. Track 1) is used to encode the lock vendor key string for the guest and another track (e.g. Track 2) is used to encode access control data for parking. There are various methods used for encoding both tracks: 1. The Lock Vendor Encoder can encode both tracks at the request of a PMS interface where the PMS 'checks-in' a guest and also provides Track 2 data to encode on the card in addition to the key string data. The Track 2 data was first retrieved from a parking vendor system by the

	PMS before being supplied to the lock vendor system to encode a physical card. 2. Or, a separate system and encoder is used for lock vendor key string data from another system and encoder that encodes the Track 2 parking data onto the same card. The card essentially gets encoded twice. Mobile Key systems shall provide functionality to enable third party parking systems to interact with the mobile device to authenticate the guest for parking. Best practices include: • Turnkey solutions that actuate parking gates directly with a mobile key enabled reader and optionally interact with physical guest cards. • Or, integration solutions where parking vendors can integrate mobile key components to provide compatible solutions.
Track 2 – Payment	For physical card systems, Track 2 has also been used for payment, where an account identifier or credit card number is encoded to Track 2. Mobile Key systems may provide functionality to enable third party payment systems to interact with the mobile device to authenticate the guest for payment. Best practices include: • Utilizing existing payment systems (for example Apple Pay) for payment instead of the mobile key system.
Elevator Control	For physical card systems there are provisions in the key string data to encode access rights to elevators. There is sufficient flexibility in these systems to define specific floors or groups of floors, or specific elevators. Mobile Key systems shall provide functionality to enable access controls for elevators. Best practices include:

	• Turnkey mobile key enabled readers that replace existing elevator readers and operate both by card and mobile key
Multiple lock vendors provide compatibility to the same Mobile Key System	Mobile Key systems may provide for multiple lock vendors to integrate so that only one mobile key system can open locks from multiple vendors. Historically, many lock vendors utilize the same card technologies, including mag cards, ISO-14443 variants, and the like. Historically this has been done by using common components, for example, using RFID reader chips that are ISO-14443 compatible (such as NXP or MIFARE). Best practices include: • Lock vendors can integrate a Mobile Key credential module with Hardware & Software development kit that can be provided to lock vendors for integration • Module has clearly defined interfaces, proven integrations, and supporting documentation • Vendor has available development & QA environments for integration and testing • Vendor or supplier to vendor has demonstrated capability for module development, manufacturing and supply • Mobile Key credential module is supported by a roadmap of future modules that continue to enable Mobile keys of the future • Mobile key credential module can be firmware upgraded

The above integration use cases are supported by the following interfaces and best practices. Not all interfaces are required in each integration use case noted above:

Interface	Type	Used By / Purpose	Best Practices
Legacy Serial PMS Interface on Lock Vendor Key Server	RS232 – Serial Data	• PMS checks in a guest	• PMS Server and Key Server are located in hotel office space that is locked and not accessible to guests • PMS Server and Key Server are located close together • Serial Cable is securely attached at each end

			• Identify an upgrade Roadmap for the PMS and Lock Vendor Key Server that obsoletes this interface
Lock Vendor Key Server API	Web Service REST/JSON	• PMS checks in a guest	• Restful APIs (Currently accepted best practice) • Support Extensibility o Versioning • Strong Authentication • On Premise or Cloud hosted
Mobile Key Credential Service API	Web Service REST/JSON	• PMS Issues a Mobile Key to a Mobile Device	• Restful APIs (Currently accepted best practice) • Support Extensibility o Versioning • Strong Authentication
Mobile Key App API	Software Library	• Hotel Loyalty App integrates with Credential Mobile Library	• Software Library compiled with Hotel Loyalty App • Encapsulates Mobile Key functionality for credential download and delivery to Locks • API enables Hotel Loyalty app experience to 1) Present nearby locks to Guest, 2) Signal intent to open lock, and 3) Provide actionable definitive feedback on success or failure of credential delivery to lock and whether lock can be opened • Encrypts data at rest • No hard-coded encryption keys • Operation logging
Mobile Key Credential Module Interface	Hardware Interface Serial Data	• Credential module adds Mobile key capability to a Lock	• Reliable soldered interface • Located in secure part of lock (not easily accessible to guest) • Definitive feedback from lock to module for the result of an operation (i.e. lock can be opened, lock cannot be opened because the Privacy Bolt is

		• Module used by a Parking reader • Module used by an Elevator Controller	closed, lock cannot be opened – this is not your room, etc.)

2.2.5 Supply Chain

Mobile key solutions require that physical components (i.e. Door Locks, readers, etc.) are supplied to hotels that are compatible with mobile devices. Traditional supply chain expectations and best practices apply.

Best practices for mobile key supply chain include:

- Ensuring vendors and suppliers of vendors control of the trust chain for delivering mobile key components and devices. The risk is that an attacker could provide forged or Trojan devices that are compromised before they are even installed.
- Provisioning of encryption keys needs to be managed and well documented. A best practice is to have this process reviewed by third party security experts.

The following secure design principles apply:

- **Open Design**. The integrity of system security should not rely on secrecy.
- **Build Security In.** It is both more effective and less expensive to build security into each stage of the development process, rather than considering it at the end.
- **Reassessment Iteration**. Security is an ongoing process that should be visited at very frequent, regular intervals.

2.2.6 Mobile Key Distribution

Mobile Key Distribution involves the whole chain of delivering access rights to a mobile device so, for example, a guest can open their guestroom door. Mobile key distribution can be done in many ways but there are a couple of best practices that always should be followed, in addition to the security best practices described in Section 3.2.7.

Always consider:

- **Scalability** – Mobile Key systems should support credential distribution growth over time for many guests, many properties, and many rooms as they are added. Scalability includes capability to add additional servers to handle load as well as future proof sizing of identifiers and data elements.

- **Availability** – Best practices include redundancy, always on database configurations, patching without outage, fail-over systems, and disaster recovery systems. An industry best practice is to provide 3 '9s' for 99.9% availability for Mobile Key credential services.
- **Maintainability** – Best practices are to build the system with state of the art software languages and technologies and to continue to update the system over time.
- **Extensibility** –Best practices are to maintain backwards compatibility on releases while adding forward functionality that can be utilized to extend features and enhance the system. Always adhere to industry best practices for interfaces including object orientation. Avoid technologies that obscure the way the solution is implemented.
- **Supportability** –Best practices are to provide visibility to auditable logging information including error conditions and log records for supporting and troubleshooting guest mobile key problems.
- **Measurability** –Best practices include tracking metrics pertaining to the guest experience (for example, time to open lock) and for lock performance (for example, current battery level).
- **Accessibility** –Best practices include white listing the mobile key credential service on the hotel guest Wi-Fi network so it can be freely reached on the hotel Wi-Fi network.
- **Usability** –Best practices are to keep things simple, uncluttered, and intuitive to use. User convenience is the number one download criteria apart from entertainment for mobile applications.

2.2.7 Security

When looking at security of mobile key solutions, industry security best practices always apply including, for example, SDLC (System Development Life Cycle – Planning, Analysis, Design, Implementation, Maintenance) best practices, OWASP (Open Web Application Security Project – is an organization that provides unbiased and practical, cost-effective information about computer and Internet applications) principles, and so on. Follow these guidelines:

- Design for security
- Look at the whole chain
- Find where to store sensitive information
- Determine how this sensitive information is communicated

The weakest link in the chain is usually the link that requires human interaction. Among these are logins in different forms as well as the potential for social engineering people when human interaction is required. Another weak link in mobile keys involves/includes

older technologies and legacy interfaces, such as the serial PMS interface on lock vendor key servers.

Security best practices include:

- **Auditable security:** Always ask for separate, third party reviewed security reports that include testing as well as secure code scans and reviews. It's not uncommon for these reports to initially have numerous findings, but Mobile Key vendors should continually find and fix these issues to maintain a level where no High or Medium risk issues are known, Lows are on a roadmap to fix, and any product updates are additionally reviewed for security findings. Mobile Key vendors should be committed to delivering and supporting a secure product.
    - **Audit Frequency:** A best practice is for a Mobile Key system to be reviewed annually or on major releases.
- **Secure credentials:** Mobile key solutions shall utilize secure encryption technology to deliver a secure credential. A credential is a packet of data (think of it like a document) that communicates information and can be verified. Parts of the credential may be kept private by encrypting it. The credential may have a signature that can be verified to ensure it is trustworthy. There are different types of encryption architecture that can be used to create a credential. For example:
    - **Symmetric Key Architecture:** In this architecture, the encryption key used to make the credential is the same as the key used to verify the credential. Best Practices:
        - Utilize a different encryption key for every property so that if a key is compromised, only this one hotel is compromised. The best practice is to utilize a unique key for every access point.
        - Utilize different keys for different use cases; for example, one key is used for guest credentials, another for staff credentials, and so on.
        - Provide for key rolling or changing
    - **Asymmetric Key Architecture:** In this architecture, the encryption key used to make the credential is a private key that is only known by one party; for example, the Credential Service. The verification key is a public key that can be known by anyone. Best Practices include:
        - Utilizing different public/private key pairs for different hotels
        - Providing methods for rolling, revoking or changing public/private key pairs

- **Secure protocols:** Always use secure, standard protocols (for example, HTTPS – TLS 1.2 or higher currently) as a base communication protocol where possible.
- **Protect legacy technologies from public network access:**
    - RS 232 communication
    - Open TCP/IP or UDP/IP ports
    - Non-encrypted protocols
    - There are numerous ways to protect legacy technologies as long as the chosen way is reviewed properly.
- **Minimize access points to the system:**
    - Clear segmentation of access points
    - Make sure that each access points has proper access rules applied
    - Do not allow new access points without review
    - Avoid generic access tools such as remote desktop access or at least make sure they are properly contained.
- **Do not hard-code username/passwords:**
    - Password policies need to be enforceable.
    - All tools involved with Mobile Key should enforce a password policy.
- **All pieces of the solution shall authenticate each other**. For example: locks need to authenticate the user trying to unlock it. However, it's just as important the user authenticates the device they are trying to unlock so they don't inadvertently pass credentials to an attacker.
- **Role-based access controls:** Users, and applications acting on behalf of users, should be limited to only those permissions required for accessing the services/devices they need. The ability to unlock a device should not mean the user can reset a key or lock identifier.
- **Expiring session keys:** Devices should not use static keys for unlocking. While they may have a unique shared secret used for authenticating with a key server, unique session tokens should be used for unlocking devices and should expire in a short amount of time.
- **Data Encryption:** Recommend all data encryption is AES 128 or stronger.
    - **Transit encryption:** Keys and other secrets should never be sent over a plain-text channel.
    - **Data encryption at rest:** Data stored on mobile devices should be encrypted.
- **Authentication Options:** Mobile keys offer the ability to add layers of security that may not exist in traditional key card scenarios. For example: requiring the device to be unlocked can add a second authentication layer.

- Mobile Key Vulnerabilities that should be addressed:
  - **Man in the Middle Attacks**. Prevent the ability to capture, re-use, or alter mobile key data.
  - **Protection against replay attacks**. Solutions should prevent these attacks. Once a session token has been used to unlock a device (see above), that token should no longer be accepted. Expiring session keys and protection from replay attacks are available in standard protocols (i.e. Kerberos v5 and OAuth v2).
- **Security Patching:** Mobile Key vendors should have process to address and deliver security patches and updates.
- **Mobile key solutions should not utilize homegrown encryption algorithms**.

2.2.8 NFC Communication

Near Field Communication (NFC) is a method of using a mobile phone to communicate to a RFID reader. There are 3 modes: Reader/Writer, Card Emulation, and Peer-peer. The advantage of using Card Emulation mode is that a Mobile Key system can be introduced and operated without requiring an upgrade to locking devices. Some mobile phones can successfully emulate RFID cards using NFC Card Emulation mode, and some cannot. For example: Apple does not currently support NFC communication. Android phone's newer versions are compatible with both ISO 14443 A/B and FeliCa aka JIS X 6319-4 standards.

For best practices related to RFID and particularly to RFID security, see the section 3.1.2 of this document.

2.2.9 Bluetooth Low Energy Communication

Bluetooth Low Energy, here called BLE, is a form of Bluetooth with characteristics that makes it suitable for low power battery solutions. The advantage of using Bluetooth is that it is supported by all current popular mobile phones. For example, Apple phones are popular in North America and Android phones are popular in Europe and other parts of the world.

For mobile keys, the communication between smartphone and the lock unit should not rely entirely upon native BLE security. BLE security is extensively covered in the Security description on the website of Bluetooth SIG.

Mobile key solutions should address the security items detailed above, on top of BLE security.

2.3 Online Door Lock End Points

The principal function of a door lock is to provide secure access to guestrooms and other areas in hotels. Access is granted through the presentation of credentials. For normal operation, the

locks do not have to be connected to a network. For enhanced features and capabilities, door lock end-points can be connected to a network.

The main incentive to bring door locks online is to allow guest keys (credentials) to be revoked or extended at will. This lets guest keys get revoked during an early check-out or extended to avoid interaction between the guest and the front desk. Also, this allows staff keys to be revoked system-wide without having to visit each lock by a member of the security team Secondary benefits are centralized event logging and monitoring. Also, such systems have the ability of automatically detecting and reporting abnormal situations, such as a person who attempts to gain access with an invalid key (wandering intruder). Further, a networked lock can be integrated with other networked systems, such as a room automation (energy management) system.

With online systems, it is possible to remotely configure or upload the firmware to the door lock (OTA, Over-The-Air upload). These features have a high-impact potential for cyber security weaknesses and deserve a strong focus during the threat analysis. It is encouraged for additional defensive measures to be considered, such as separate authentication layers for configuration instructions or secure boot features for OTA.

Secure design principles at stake include:

- **Fail Secure**. Systems fail, and those building systems should plan for failure. In the event of failure, the system should default to a secure state.
- **Secure the Weakest Link**. Just as defenders calculate return on resource investment, so do adversaries. Modern adversaries choose the weakest link in the security chain as the easiest path to system compromise.
- **Economy of Mechanism**. Simplicity is the friend of security. From a security perspective, more simple systems are easier to build, validate and maintain.
- **Complete Mediation**. Every access to every resource and/or asset must be validated for authorization.
- **Least Privilege**. Allow a user only the absolute minimum access required in order to successfully perform his or her function, and nothing more.
- **Privilege Separation**. Divide privileges so that a user must have multiple privileges in order to perform a larger scale compromise. This requirement of additional privileges reduces risk.
- **Defense in Depth**. Adopt the assumption that a compromise has already occurred and architect defenses in a way to make broader scale compromise difficult. Start by identifying the most valuable assets and then build layers of protection emanating outwards.

Unlike a commercial online access control system where locking mechanisms are merely readers and where all access decisions are made by a central service, a hotel access control system typically requires to guarantee the access function (even under the condition where the network is inoperable). In the case of a network outage in a commercial application, the system

would typically fail into the "locked" mode (it will be acceptable that there will be no access of facilities until the network is re-established). Such a failure mode is unacceptable because no reasonably staffed hotel can handle the resulting guest complaints and assistance in accessing each guest room with emergency keys. Unlike in commercial access control systems, this functional difference hardens the hotel application against network outages and, in particular, DoS (Denial-of-Service) attacks.

2.3.1 Objectives

The objectives when evaluating the benefits of an online lock system are:

- Valuation of the additional features that an online system brings. Will the increased system functionality outweigh the increased complexity?
- Identification of the possible failure mode and potential remedies.
- Identification of the additional cyber security threats introduced by adding a network transponder to the locks.
- Identification of unintended consequences, such as increased battery usage by the locks that would require a more frequent battery replacement.

2.3.2 Architecture & Design

The following are general practices for an online lock system:

- Create and manage access controls
- Actively manage proper configuration of firewalls and other perimeter defenses
- Implement strong authentication techniques for all remote access as well as for any internal access to high value assets
- Harden core operating systems and major applications against internal attack
- Separate internal users from guest users
- Physically secure your access points
- Limit Wi-Fi range to property confines
- Search for rogue access points
- Quarantine legacy systems to low privilege segments
- Utilize multi-factor authentication
- Run continuous automated scan tools
- Perform ongoing, periodic, thorough, manual, security assessments
- Wandering intruder feature must be available to disable the key card after a configurable number of rogue attempts

2.3.3 Online Network Infrastructure

In most instances, the locks will utilize a wireless network transponder to facilitate the online functionality. Key consideration for this transponder:

- Battery efficiency
- Widely followed radio standard to allow integration with several different third-party systems.
- Good co-existence with other radio standards, especially for minimizing impact of cross-talk with the credential reader (key reader such as RFID, BLE, etc.)

The locks' radio transponder is mainly used for multiple logical network applications. In a first instance, the lock will communicate on a peer-to-peer basis with the access control server. Typically, the door lock and the access control server are provided and managed by the same vendor company. A second logical network application links the lock with third party applications, such as a room automation or an energy management system. For common areas, this secondary network can also enhance functionalities of elevator and garage access points.

Secure design principles at stake include:

- **Open Design**. The integrity of system security should not rely on secrecy.
- **Economy of Mechanism**. Simplicity is the friend of security. From a security perspective, more simple systems are easier to build, validate and maintain.
- **Least Common Mechanism**. Shared resources introduced shared compromise. Wherever possible, an organization should reduce or eliminate shared attack surfaces.
- **Reassessment Iteration**. Security is an ongoing process that should be visited at very frequent, regular intervals.

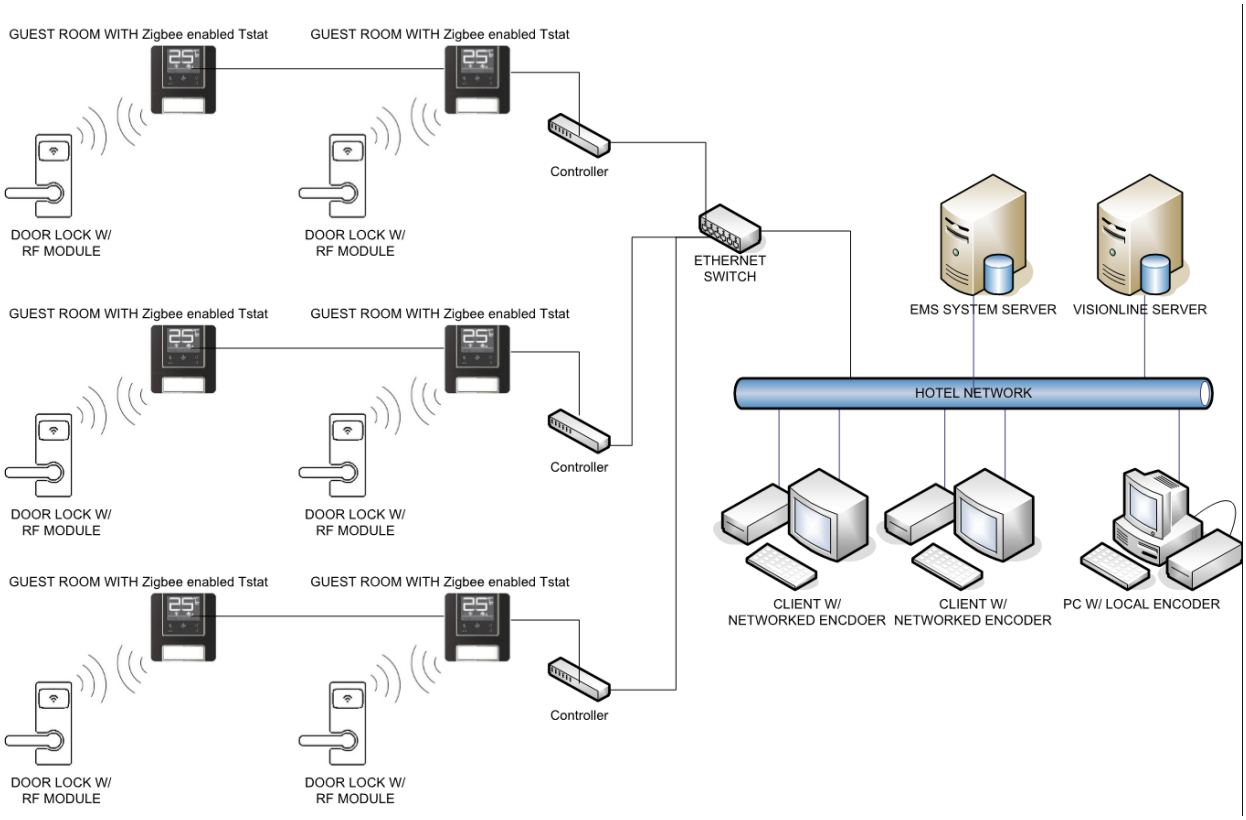


Figure 2 Example Online Door Lock System

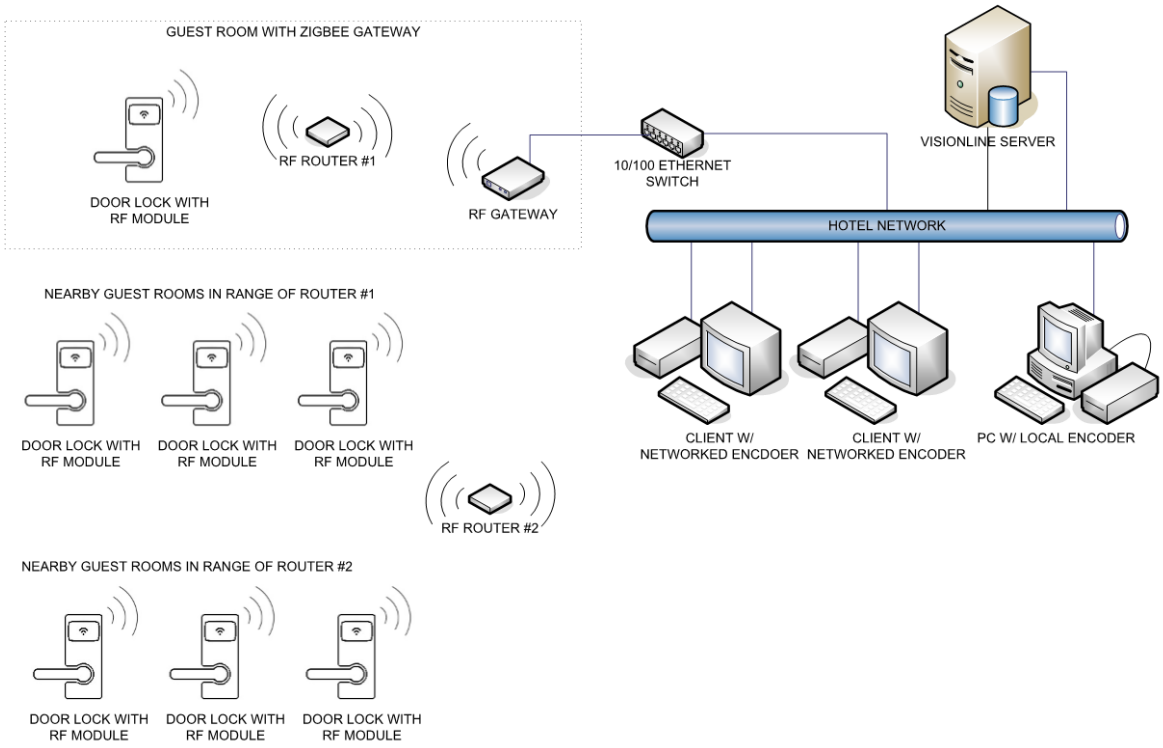


Figure 3 Example Door Lock System Using Wireless Connectivity

If a lock is not only communicating with its access server, but also with an auxiliary system such as a room automation system, it is a good practice for the security configuration (keys) with the auxiliary system to not get shared with the access control application. Further, the door lock's API with the auxiliary system should be carefully protected against any functionality that could impact the access control decision.

Secure design principles at stake:

- **Secure the Weakest Link**. Just as defenders calculate return on resource investment, so do adversaries. Modern adversaries choose the weakest link in the security chain as the easiest path to system compromise.
- **Least Common Mechanism**. Shared resources introduced shared compromise. Wherever possible, an organization should reduce or eliminate shared attack surfaces.

Bringing door locks online can be based on a single vendor solution where all networking equipment will be provided from a single source. In other instances, the lock can ride an existing third party network. Independent of the chosen approach, these networks transport sensitive information for keeping guests safe, avoid loss of revenue and provide access to unaltered information. It is therefore mandatory that the date needs to be secured against cyber security threats.

In either case, the buyer of a networked solution needs to clarify who owns and maintains the network infrastructure.

Online lock systems are often also integrated with external systems that reside behind the access control server (PMS, elevator, garage, external access control system, theme park, ski lift, pool, etc.). These external applications provide yet another aspect that needs to be reviewed from a cyber security perspective.

2.3.4 Wired Infrastructure Standards

Door lock systems make use of several different standards. Often, the applied technology is chosen based on what is available on premise or what a third-party integration partner can support.

- **Ethernet (802.3) and PoE.**
- **Fiber**
- **RS485, RS232**

Because no site is alike, it is quite common that door lock vendors can offer various options, gateways and bridges to achieve a certain functional requirement. Because of this great variance in system topologies, it is recommended that the network traffic is end-to-end secured so systems do not have to rely solely on the network security of each link. While link security is a desirable additional layer of protection, it would not be wise to rely on the assumption that every link is sufficiently secured to protect against eavesdropping or packet insertion.

Secure design principles at stake include:

- **Open Design**. The integrity of system security should not rely on secrecy.
- **Economy of Mechanism**. Simplicity is the friend of security. From a security perspective, more simple systems are easier to build, validate and maintain.

2.3.5 Wireless Infrastructure Standards

Online door lock systems built with the following wireless standards:

- ZigBee (various versions and implementations), 6LoWPAN
- 802.15.4 (Data link layer protocol of above)
- Z-Wave
- BLE
- Proprietary formats (e.g. infrared)
- Wi-Fi
- LoRaWAN

Key considerations for selecting the wireless technology include:

- Impact on battery life
- Real-time capability to send a command to the lock

- Distance between the lock and the corresponding closest network transponder
- Number of network standards on premise
- Cost to maintain the different network technologies on premise
- Availability of the needed bandwidth

As of November 2016, there are no high-level (API) protocol standards to facilitate an off-the-shelf interoperability between a lock and a third-party system or to allow substitution of a lock from one vendor to a different vendor in the same system. This fact happens to be valid for wired network standards (think BACnet) as well. The closest to a high-level standard for this industry is the "HTNG Intelligent Guestroom".

Secure design principles at stake include:

- **Open Design**. The integrity of system security should not rely on secrecy.
- **Economy of Mechanism**. Simplicity is the friend of security. From a security perspective, more simple systems are easier to build, validate and maintain.

2.3.6 Wireless Architecture Standards

Online door lock systems are built with two principal connection topologies:

- **Point-to-point**: The door lock communicates with a designated proxy device. The proxy device provides a store-forward (mailbox) capability to buffer messages toward the lock. Locks periodically query their respective proxy device to acquire the buffered messages. This polling interval typically defines the response time (real-time) capabilities of the system. The faster the lock polls, the quicker the response time of the lock, but the higher the drain on the battery.
- **Mesh network**: The lock communicates through multiple (or alternate) devices to a proxy device. In a wireless mesh network (e.g. ZigBee), the proxy functionality is typically implemented in the mesh gateway (PAN coordinator).
- **Powered door locks**: In the case where continuous power is available for the lock, it can act as a full-function wireless device that has on-demand availability. These systems have the fastest response time because they do not depend on a polling mechanism. If the principal lock mechanism is mounted in the door, the solution can be relatively costly because electrical power would need to be brought to the door blade. This solution is available without a significant cost penalty if the main lock controller is in the form of a wall reader that can operate a strike. Wall reader versus door mounted door locks have regional preferences and code implications (e.g. Middle East vs. North America).

Data security on these networks can be achieved on the application layer and/or on the network layer. When locks are made interoperable with third party systems, attention needs to be paid to shared network keys. Sharing keys with additional partners increases the potential to leak the keys to potential intruders.

Secure design principles include:

- **Build Security In.** It is both more effective and less expensive to build security into each stage of the development process, rather than considering it at the end.
- **Reassessment Iteration**. Security is an ongoing process that should be visited at very frequent, regular intervals.
- **Fail Secure**. Systems fail, and those building systems should plan for failure. In the event of failure, the system should default to a secure state.

2.3.7 Third Party Integrations

This chapter does not only pertain to online lock systems but to non-networked lock systems alike. In all cases it will be required to authenticate a communication partner and to carefully vet that the operation requested is in fact allowed. For example, if a third-party system requests to create a key for a room, the access control system has no ability to test that the request has contextual validity (the true purpose or reason to create the key). It is therefore impossible to defend against attacks that enter through authorized APIs. However, a recommended practice is that such APIs are fully and securely logged for providing accountability and assistance in resolving security related inquiries.

2.4 Physical Key Solutions

Contactless key cards offer benefits in the guest access experience, security, and robustness over magnetic stripe cards.

- **Superior robustness**: Contactless cards require no physical contact with the room door lock and therefore do not wear out from use. They are also not affected by magnetic fields which can erase magnetic stripe cards, rendering them inoperative.
- **Reduced maintenance**: Contactless door readers do not require periodic cleaning for proper operation.
- **Faster transaction and action times**: Contactless cards can be programmed and open doors more quickly than magnetic stripe cards since the cards do not have to be inserted for use.
- **Higher security**: Contactless cards offer a range of security features not available with magnetic stripe cards.
- **Broader application use and future proofing**: Some contactless card technologies enable additional features for a hotel branded card such as loyalty, micropayment and mobile device interaction.

2.4.1 RFID vs Magnetic Key Mechanics

The primary architectural difference between a magnetic stripe card and a RFID card is how data is written on and read from the data storage area of the card.

A magnetic stripe contains 2 to 4 tracks of data that can be read by any compatible reader that comes IN PHYSICAL CONTACT with the magnetic stripe.

A RFID card stores data on an Integrated Circuit (IC) which is designed to transmit the stored data to any compatible reader coming within the required PROXIMITY to the card.

The ability for a RFID reader to receive the data from a RFID IC without physical contact increases the security attack surface. Therefore, to protect the data stored on a RFID IC, the IC must have a method to verify the identity of the RFID reader requesting the data.

Secure design principles at stake:

- **Build Security In.** It is both more effective and less expensive to build security into each stage of the development process, rather than considering it at the end.
- **Secure the Weakest Link**. Just as defenders calculate return on resource investment, so do adversaries. Modern adversaries choose the weakest link in the security chain as the easiest path to system compromise.
- **Defense in Depth**. Adopt the assumption that a compromise has already occurred and architect defenses in a way to make broader scale compromise difficult. Start by identifying the most valuable assets and build layers of protection emanating outwards from there.

2.4.2 RFID Key Security

RFID communication security consists of two elements: Authentication and Data Encryption.

- **Authentication**: The process of verifying the authenticity of two communication partners to each other. Therefore, a RFID IC must be designed to use a method of authentication to prevent an unauthorized reader from obtaining the data stored on the RFID IC. This process is intended to prevent a forged card from exchanging data with a reader.
- **Data Encryption**: Once both parties have established that they are authentic, data is then passed between them. This data exchange can be encrypted to prevent eavesdropping.

2.4.3 Authentication

There are three basic architectures for authentication:

1) Password systems
2) Challenge-response authentication
3) Customized and zero-knowledge authentication

Password systems are a weak level of security and zero-knowledge require inefficient methods that involve strong mathematical problems that are challenging to implement. Challenge-response are broadly used because they are strong and efficient.

Asymmetric Authentication requires the communication pairs to each exchange public keys. These public keys are often verified by a third party (online) to confirm identity and exchange a unique private key to use for the communication session.

- Communication point *A* sends a random one-time-use value (nonce) to point *B* using *B*'s public key.

B decrypts the nonce using its own private key (required to decrypt the message) and sends the nonce value back to point *A* (sometimes encrypted using *A*'s public key) for verification of the value.

Asymmetric key techniques typically require more processing than symmetric approaches which can result in longer transaction times and more expensive card solutions. This method is most common in software and internet communication due to the availability of CAs and the processing required to complete the process. RSA is an example of asymmetric key authentication.

Symmetric authentication leverages a shared secret key. Authentication is accomplished by verifying the possession of the secret key without actually transmitting it.

One challenge with symmetric keys is the distribution and management of these private keys. Every update to the key has to be communicated to all participating communication partners securely. In hotel lock systems, the communication partners are all RFID encoders and door locks equipped to read RFID. Compromising of only one device holding the secret/private key compromises the whole system. Symmetric key systems have the advantage of typically being more simple to implement and require less processing which can result in quicker transaction times and lower card costs.

Symmetric authentication systems are best suited for closed systems such as hotel door lock systems, where the private key and subsequent updates come from a central system or supplier. However, the challenge in hotels is that there is rarely, if ever, an opportune time to update the secret key. This timing is difficult because hundreds to thousands of guest room and staff keys may need to remain valid before and after the time the secret key is changed during the update process. This challenge can be mitigated by building key update capabilities into readers and encoders.

2.4.4 Data Encryption

Once the card has been authenticated with the reader or door lock, data is passed between the two parties. This data may be either transmitted clear (unencrypted) or encrypted. If the data being passed has independent value or privacy requirements, encryption of this data is required.

Two common approaches for RFID data encryption:

- Proprietary encryption: The use of propriety encryption algorithms protect the data that is passed after the authentication stage.
- Open standard encryption: The use of open standard encryption algorithms such as Triple DES or AES are independently evaluated.

The above describes why it is a best practice for RFID-based lock systems to use an authentication mechanism with a symmetric challenge-response. For this system to maintain a robust security profile, there must be a method for updating the private keys between all communication partners (i.e. RFID encoders and RFID door lock readers).

Currently, there is not a process in place to update the symmetric keys.

There are two typical scenarios:

1. The lock vendor is using its own set of keys and is not exposed to any other vendor. (This typically requires having the end user acquiring the credentials already secured with those symmetric keys.)
2. The lock vendor has a unique set of keys assigned to the end user.

Updating the symmetric keys requires updating the keys at the DB level and from there to the different readers. This process could be challenging for an operational hotel because there could be a gap in between the credentials being encoded with the new set of symmetric keys and the readers.

In the case of online-wired readers, the process of updating the symmetric is very simple. In the case of stand-alone readers, the lock vendor will need to figure out a process to transmit the keys from the DB into the lock reader.

This process will apply for Mifare plus, Desfire, Desfire Ev1 and Mifare Ultra light. However, different lock and reader vendors has some limitation or different processes to manage the keys.

This process will potentially increase the security level of each site if the keys are exposed.

Questions to consider:

- How often will the keys be required to be updated?
- What is the vendor internal procedure in case of PCI bridge?
- What process to automatically update the keys can be formed? (The goal would be for the system to be capable of generating a new set of keys and automatically updating those keys on every reader.)

Secure design principles at stake include:

- **Reassessment Iteration**. Security is an ongoing process that should be visited at very frequent, regular intervals.
- **Trust Reluctance**. Assume all trusted parties (including users, trusted employees, integrated third parties, etc.) are or can become malicious; architect defenses accordingly.
- **Deferral of Risk**. In many industries, vendors often look to their customers to articulate requirements. Conversely, the customer tends to rely on the vendor to sell them solutions that are secure. Outlining requirements is a valid method to deliver business benefit, but not necessarily a valid way to ensure proper system security. By relying on others to dictate security requirements, and not driving it as a mission-critical business agenda, the vendor is deferring risk to the unwitting customer, who may not know the gaps in the vendor systems. Another instance of this condition is heavily reliant on compliance, standards, and/or automated scanning, rather than threat modeling and security assessment.

- **Security Through Obscurity**. An adversary not knowing where to find an asset or how a system operates does not inherently protect the asset (the inverse of Open Design).
- **Compliance**. Although commonly perceived as one, compliance is not a security measure. Compliance only works if the enemy you are trying to thwart is the auditor.

2.4.5 General Contactless Smart Access Card Requirements

Contactless smart card interface	ISO 14443-4 A/B Usage of accepted and standardized interfaces ensures compatibility with future products and a high availability of compatible components.
Reading distance	For maximum convenience and usability cards shall offer 2-3 cm reading range.
System compliance	Certified card performance Independent card performance certification is recommended to ensure quality and reliability. This also ensures highest performance and interoperability of cards throughout different infrastructure components. Leading test facilities: • UL Transaction Security (www.ul-ts.com) • Arsenal Testhouse (www.arsenal-testhouse.com)
Security communication protocols	-Mutual authentication: Protects system against copying credentials. - Encrypted communication: Protects privacy of data stored on card.
Physical card reliability	Conformance with: • ISO 7810 • ISO 7816
Product certificates	Cards conformance with: Conflict Minerals Report (CMRT) Hazardous substances (ROHS) Restricted Chemicals (REACH)
Manufacturing environment	Manufacturer certified: Quality Management Systems (ISO 9001) Environmental Management System (ISO 14001)

Secure design principles at stake:

- **Security Through Legality**. Regulation and law do not prevent an attack, nor do they effectively outline measures to be effective in all cases.
- **Compliance**. Although commonly perceived as one, compliance is not a security measure. Compliance only works if the enemy you are trying to thwart is the compliance auditor. Against any other enemy, compliance does not effectively defend.
- **Fail Secure**. Systems fail, and those building systems should plan for failure. In the event of failure, the system should default to a secure state.
- **Secure the Weakest Link**. Just as defenders calculate return on resource investment, so do adversaries. Modern adversaries choose the weakest link in the security chain, as the easiest path to system compromise.
- **Build Security In.** It is both more effective and less expensive to build security into each stage of the development process, rather than considering it at the end.
- **Reassessment Iteration**. Security is an ongoing process that should be visited at very frequent, regular intervals.

2.4.6 Contactless Smart Access Card Options, Considerations and Recommendations

Contactless smart cards first entered the market in the early 1990's. This first generation of smart cards offered linear memory structures typically secured by proprietary cryptographic algorithms and protocols for secure authentication and data protection.
Over time, security vulnerabilities in this first generation of cards have been exposed which has driven fundamental security improvements in the design and certification of subsequent offerings. This next generation of smart card offerings are designed, validated, and certified to third party standards with an independent evaluation process, resulting in significantly more secure products. These improvements in design, technology, and certification include:

- The use of the open standard encryptions are evaluated and monitored by independent testing, academic, and government organizations with Triple DES and AES being examples of the current state of the art.
- Incorporation of hardware and software features protect against a range of direct and indirect (side channel) attacks and cloning efforts.
- Independent third party functional and security certification such as that provided by Common Criteria.

In addition to security improvements, new features and capabilities are offered by some smart cards to simplify multi-application support, advanced applications or applets, and compatibility with mobile devices.

2.4.7 Minimally Featured Solutions

Minimal requirements for contactless cards used in hospitality use cases:

- Solution offers a very basic level of security through an optional mutual authentication
- Products offer minimal security by design
- Products are primarily recommended for limited use applications where neither security nor privacy are main concerns
- Products offer the convenience of contactless technology at the lowest price point

Simple NFC cards or MIFARE Ultralight are examples of this type of solution.

2.4.8 Legacy Systems

Requirements for contactless cards used in legacy system focus on current use cases and deployed technology. Many of these solutions are based on out of date security design practices and cryptographic algorithms with established vulnerabilities. These offerings typically offer limited security at a moderate price point. An example of this card type is MIFARE Classic.

2.4.9 Modern Highly-Secure Basic Solutions

These cards are typically offered as cost and functional equivalents to legacy systems, but have significantly stronger security capabilities. These include the latest encryption algorithms and security protocols created to leverage the latest design practices and technologies with extensive third party validation. They offer dramatically improved security at a similar price point to many legacy systems. An example of this card type is MIFARE Plus.

2.4.10 Modern Multi-Application Solution

A solution with state-of-the-art security level offering maximal security and technology features for the integration of third party applications such as Smart Mobility, micropayment or event ticketing, onto the hotel card. Further, this solution is well suited for combination with brands' loyalty program and are often designed with mobile device compatibility in mind. These products typically feature larger memory sizes, a broad range new features and higher costs. An example of this card type is MIFARE DESFire.

2.4.11 Comparison Matrix

	Minimally Featured	Legacy	Modern Highly-Secure Basic	Modern Multi-Application
Interface	ISO-14443-3 A/B	ISO-14443-3,4 A/B	ISO-14443-3,4 A/B	ISO-14443-4 A/B
System Compliance Certification	-	Recommended	Recommended	Recommended
Security protocols	Optional mutual authentication, password	Mutual Authentication & Data Encryption	Mutual Authentication & Data Encryption	Mutual Authentication & Data Encryption
Cryptographic algorithms	Optional DES, TDES, AES	Legacy algorithms (eg.	TDES, AES	TDES, AES

		Crypto 1 for MIFARE® Classic)		
Privacy Protection	–	Data Encryption	Data Encryption, Random ID	Data Encryption, Random ID
Advanced Security Features	Optional originality check	Optional originality check, proximity check	Originality check, proximity check	Originality check, proximity check
Security Certification	–	–	Common Criteria EAL4+	Common Criteria EAL 4+
Multi-application support	No	Optional	Optional	Designed for multi-application use
Standard communication protocol support	–	–	Optional ISO 7816-4 support	ISO 7816-4 support
Memory Size	>40 Bytes	>256 Bytes	>256 Bytes	>2k Bytes
Physical Card Reliability	Optional ISO 7810, 7816	ISO 7810, 7816	ISO 7810, 7816	ISO 7810, 7816
Relative Cost	*	**	**	***
Relative Security	*	**	****	****

2.5 Third Party Integrations

Enable integrated systems to read data from keys.

- The more basic integration has the different systems using different memory sectors within the chip. Each vendor will secure each application with its own set of keys. This requires obviously having the credentials being encoded and protected by the different applications that are making use of the chip.
- “In the case that lock vendors write additional information such as the guest ID or folio number, third party systems such as POS are able to obtain the data.”If lock vendors encrypt the data using its own encryption algorithms, it is required to use a lock vendor reader to capture the data and output on a standard interface (wiegand, clock data, OMRON, etc.). This is preferred because the keys are not exposed and shared between different vendors.

Data storage on the key credential and how it is read back to a third party system

- In some scenarios there is no choice other than sharing keys. In this case, integrate the lock system with tablets or kiosk systems. The third party reader (different than the reader provided by the lock vendor) will be encoding the credentials so it will need the keys to have access to write the information inside the lock vendor memory.
  It is important defining a good practice process to ensure the sharing of keys between the different vendors involved is properly managed. As an example, the electronic copies of the keys should be avoided.

- Address best practices for both magnetic and RFID
  - Consider multiple variants of RFID, including but not limited to Mifare Ultralight.
  - For security, management and functionality purposes, it is recommended to use for staff high encryption and memory size. Additional memory sizes may enable:
    - Integration with other systems (T&A, AC, POS, etc.)
    - Complex access plan (in case of data on card solutions)

  - In the case of a guest, of course the high level of security is required, however the size of the chip will depend on the level of integrations required. For that reason, Mifare Ultralight C could be an option but in many occasions it will be required to acquire a bigger chip size as Mifare plus.

2.6 Glossary of Terms

For the purpose of this document the following terms have been defined as follows:

Term	Definition
Common Criteria	An international set of guidelines and specifications developed for evaluating information security products.
Contactless Smart Card	A contactless credential, typically embedded into an integrated circuit that communicates via radio waves.
Mutual Authentication	Two parties authenticate each other at the same time.
ISO Standards	14443 (a/b), 7810, 7816, 7816-4 ISO standards that apply to various parts of a door lock or key system (or integrated circuit)
JIS Standards	Integrated circuit standards (x 6319)

2.7 Implementation Notes

For additional recommended reading on many of the aspects of this best practices document, the working group recommends the following resources:

Secure SDLC:

Software Security: Building Security In, by Gary McGraw

The Security Development Lifecycle, by Michael Howard, Steve Lipner

Secure software design and implementation:

High-Assurance Design, by Cliff Berg

Security Patterns, by Markus Schumacher, et al

Building Secure Software: How to Avoid Security Problems the Right Way, by John Viega, Gary McGraw

Coding bugs and flaws:

24 deadly sins of software security, by Michael Howard, David LeBlanc, John Viega

Writing Secure Code, by Michael Howard, David LeBlanc

iOS:

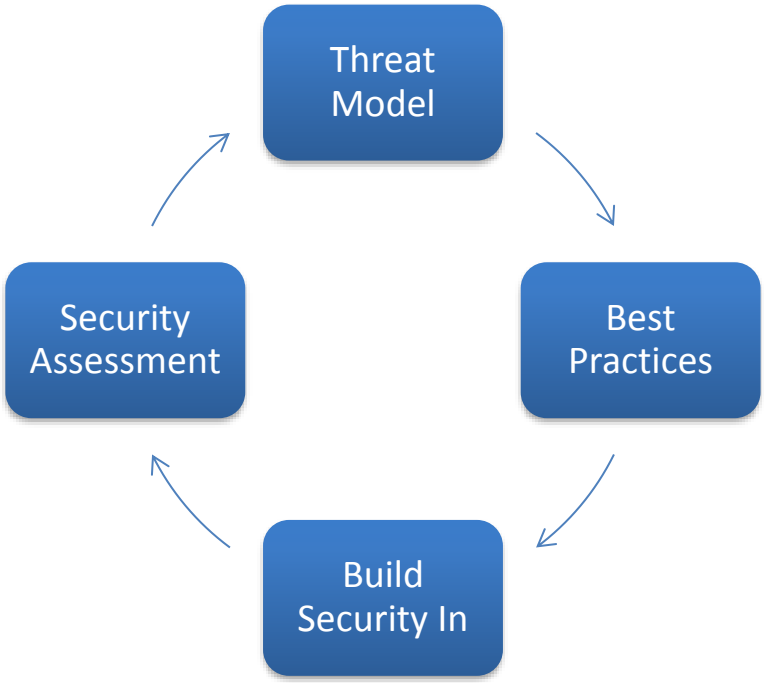
iOS Application Security: The Definitive Guide for Hackers and Developers, by David Thiel

C/C++:

Secure Programming Cookbook for C and C++: Recipes for Cryptography, Authentication, Input Validation & More, by John Viega, Matt Messier

3 Appendix

Once the requisite best practices have been considered, and subsequent development has attempted to account for building in the appropriate security controls, every organization needs to purse an effective validation mechanism to investigate for security vulnerabilities and remediate them.



3.1 Threat Model

Threat modeling is an exercise through which an organization identifies:

- The **assets** it wishes to protect
- The **adversaries** it wishes to defend against
- The collection of **attack surfaces** against which adversaries launch malicious campaigns in pursuit of assets
- The ways in which the system is **abused**, when campaigns are launched against those attack surfaces

In the context of hospitality locking systems, considerations for **assets** and **adversaries** should be given to the elements outlined in the document *Threat Model: Emerging Locking Systems*, dated 31 August 2015, published by the Door Lock Security Workgroup of Hotel Technology Next Generation.

The aforementioned threat model document outlines some abstracted elements pertaining to **attack surfaces**; this is generalized and must be adapted by vendors and/or hoteliers to the

unique collection of attack surfaces that will be relevant. From there, consideration can be given to **abuse cases** against those attack surfaces, which could be manipulated in ways to arrive at system compromise.

NOTE 1: The aforementioned threat model is for reference purposes only, intended as a foundational guide. All vendors developing online locking systems or mobile key solutions, and/or all hoteliers procuring/deploying online locking systems or mobile key solutions must adapt the foundational threat model and customize to the unique circumstances of the system(s) in question.

NOTE 2: The existence alone of a threat model is insufficient to ensure product security. Vendors and/or hoteliers must perform effective validation that the customized threat model has been (a) adequately designed, (b) properly implemented, (c) considered in such a way that the system has been thoroughly investigated for vulnerabilities and reassessed at a frequent cadence not to exceed biannually, and (d) adapted in such a way to account for mitigating controls to resolve security vulnerabilities.

3.2 Secure Design Principles

3.2.1 Overview

First and foremost, developers of emerging locking systems need to adhere to secure design principles when developing solutions.

A principle is a fundamental truth; a secure design principle is a principle upon which systems are built in order to be resilient against attack. Secure design principles are well established in the academic and research communities, yet many businesses have difficulty implementing these principles successfully, as evidenced by the widespread, devastating security breaches that continue to plague businesses today. Proper implementation of secure design principles, taken in context with business objectives and constraints, significantly reduces vulnerability and mitigates risk.

Secure Design Principles – These are universally accepted approaches to secure system design, and should be pursued.

- **Least Privilege**. Allow a user only the absolute minimum access required in order to successfully perform his or her function, and nothing more.
- **Privilege Separation**. Divide privileges so that a user must have multiple privileges in order to perform a larger scale compromise. This requirement of additional privileges reduces risk.
- **Defense in Depth**. Adopt the assumption that a compromise has already occurred, and architect defenses in a way to make broader scale compromise difficult. Start by identifying the most valuable assets and build layers of protection emanating outwards from there.

- **Trust Reluctance**. Assume all trusted parties (including users, trusted employees, integrated third parties, etc.) are or can become malicious; architect defenses accordingly.
- **Open Design**. The integrity of system security should not rely on secrecy.
- **Economy of Mechanism**. Simplicity is the friend of security. From a security perspective, more simple systems are easier to build, validate and maintain.
- **Complete Mediation**. Every access to every resource and/or asset must be validated for authorization.
- **Least Common Mechanism**. Shared resources introduced shared compromise, and wherever possible, an organization should reduce or eliminate shared attack surfaces.
- **Psychological Acceptability**. If security becomes too intrusive for a user to effectively perform his or her role, the user will circumvent the security controls. Psychological Acceptability balances security and convenience. After a certain degree of inconvenience, security will actually be undermined by the user.
- **Fail Secure**. Systems fail, and those building systems should plan for failure. In the event of failure, the system should default to a secure state.
- **Secure the Weakest Link**. Just as defenders calculate return on resource investment, so do adversaries. Modern adversaries choose the weakest link in the security chain, as the easiest path to system compromise.
- **Reduce Asset Handling**. Do you really need to collect that personally identifiable information, just because the marketing department asked for it? If you collect fewer assets, you reduce the reasons an adversary may want to attack you.
- **Build Security In.** It is both more effective and less expensive to build security into each stage of the development process, rather than considering it at the end.
- **Reassessment Iteration**. Security is an ongoing process that should be visited at very frequent, regular intervals.

3.3 Anti-Design Principles

These are approaches that are commonly misunderstood as valid security measures. They either do not improve system security or in fact reduce it. These should be avoided:

- **Compliance**. Although commonly perceived as one, compliance is not a security measure. Compliance only works if the enemy you are trying to thwart is the compliance auditor. Against any other enemy, compliance does not effectively defend.
- **Complexity**. The inverse to Economy of Mechanism, complexity is the enemy of security. Additional complexity introduces additional bugs, vulnerabilities and attack surfaces.
- **Security Through Obscurity**. An adversary not knowing where to find an asset or how a system operates does not inherently protect the asset (the inverse of Open Design).
- **Security Through Legality**. Regulation and law do not prevent an attack, nor effectively outline measures to be effective in all cases.
- **Deferral of Risk**. In many industries, vendors often look to their customers to articulate requirements. Conversely, the customer tends to rely on the vendor to sell them

solutions that are secure. Outlining requirements is a valid method to deliver business benefit, but not necessarily a valid way to ensure proper system security. By relying on others to dictate security requirements, and not driving it as a mission-critical business agenda, the vendor is deferring risk to the unwitting customer, who may not know the gaps in the vendor systems. Another instance of this condition is heavily reliant on compliance, standards, and/or automated scanning, rather than threat modeling and security assessment.

3.4 Security Program

Security is not a collection of unrelated activities; rather it is business critical discipline, requiring investment of resources, strategic planning, and iterative development. The most effective way to build security into a solution is by establishing a security program at the organization overall. Needs will vary from organization to organization, but the following are some of the most common elements of an effective security program:

- Obtain executive buy-in; make security a board-level issue
- Separate IT from Security
- Empower the CISO
- Build a threat model
    - Assets
    - Adversaries
    - Attack Surfaces
    - Abuse Cases
- Establish a risk management process
- Create and take inventory of information assets
    - Hardware
    - Applications developed in-house
    - Applications developed by third parties
    - Databases
    - Miscellaneous other, such as sites, shared folders, etc.
- Develop a security policy
- Implement security controls
- Manage vendors / third parties / suppliers / other trusted parties
- Create a disaster recovery plan
- Create an incident response plan
- Create a change management procedure
- Create a culture of security
- Conduct ongoing training
    - Cater to different audiences, including: executives, end users, developers, technical security staff and others.
    - Deliver training that drives behavior change, rather than just “checks a box”. Online, short-form training is usually ineffective in changing behavior.

- Following recommendations as outlined by the SANS Institute whitepaper *Implementing an Effective IT Security Program*, which include:
  - Periodically Assess Risk
  - Document an entity-wide security program plan
  - Establish a security management structure and clearly assign security responsibilities
  - Implement effective security-related personnel policies
  - Monitor the security program's effectiveness and make changes as necessary
- Proactively hunt for security flaws, before the adversaries find them
- Conduct security assessments, utilizing neutral external third party security experts
- Build a foundational program, such as outlined in this DEF CON talk, and as shown by this pyramid hierarchy:

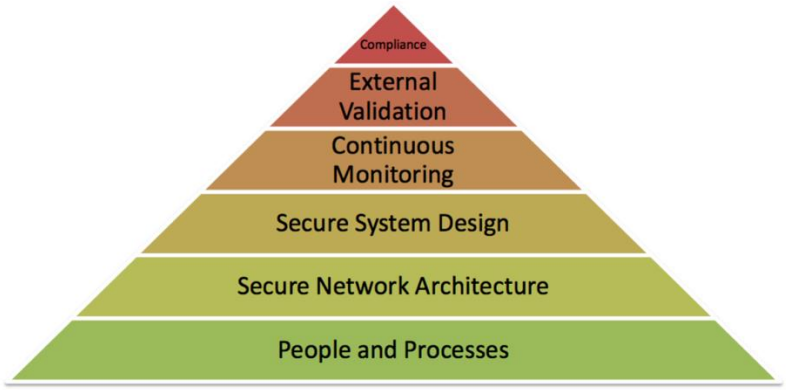


Figure 4 Foundations of Security

- Avoid traps!
  - **Compliance**. Compliance is not synonymous with security; avoid a compliance-focused attempt to arrive at security. Good security results in compliance, but compliance does not result in security.
  - **Silver bullets.** Despite common claims from security product vendors, no single product can deliver the entire solution. Products play an integral role in the effectiveness of an overall security program but do not deliver the entire resolution. Avoid reliance solely on tools to deliver the security program.
  - **Return on Investment (ROI).** It can sometimes be less obvious how to quantify the return on security investment, so some organizations elect to reduce costs associated with it. Security should be perceived as a business enabler, rather than a cost to be minimized. In order to build an effective security program, avoid thinking about it in terms of activities that eat profit margin, and instead be able to quantify and articulate the business advantages of investing in security.

3.5 System Level Security Best Practices

3.5.1 Network Security

Hoteliers deploy solutions at their property and corporate environments; vendors develop and protect intellectual property at their own corporate environments. Both stakeholders need robust network security in order to adequately provide a stable, secure environment to deploy locking systems. Needs will vary from organization to organization, but the following are some of the most common elements of an effective network security posture:

- Build a threat model
- Identify all third parties, vendors, or other stakeholders with access to your environment; identify, understand, and mitigate risk with each
- Inventory your environment, understand configurations, analyze
- Create and manage access controls
- Actively manage proper configuration of firewalls and other perimeter defenses
- Implement strong authentication techniques for all remote access as well as for any internal access to high value assets
- Harden core operating systems and major applications against internal attack
- Separate internal users from guest users
- Collect and analyze detailed logs
- Define an update policy and maintain security patches
- Monitor user activity for anomalies that could be indicators of compromise
- Physically secure your access points
- Limit Wi-Fi range to property confines
- Search for rogue access points
- Create a data breach response plan
- Create a BYOD policy
- Quarantine legacy systems to low privilege segments
- Wherever possible, encrypt data, both while at rest and in transit.
- Where appropriate, implement and keep up to date tools such as data loss prevention, intrusion detection, intrusion prevention, end point security, email filters, antimalware, antivirus, etc.
- Understand that tools require people to run them, and training to make the people effective. Invest only in tools that are effective for the needs of your organization and make sure you can invest the time and resources to have qualified people trained and running these tools.
- Restrict or minimize the use of removable media (such as USB drives) wherever practical.
- Remove anything not compliant with your organization's security policy
- Utilize multi-factor authentication
- Run continuous automated scan tools
- Perform ongoing, periodic, thorough, manual, security assessments

3.5.2 Application Security

Building and deploying applications that are resilient against attack hinge upon the effectiveness of security baked into the solution design. Needs will vary from organization to organization, but the following are some of the most common elements of an effective application security approach:

- **Build a threat model.**
- **Implement role-based access controls.** Users, and applications acting on behalf of users, should be limited to only those permissions required for accessing the services/devices they need. The ability to unlock a device should not mean the user can reset a key or lock identifier.
- **Implement mutual authentication.** Locks obviously need to authenticate the user trying to unlock it. However, it's just as important that the user authenticates the device they are trying to unlock so they don't inadvertently pass credentials to an attacker.
- **Expire session keys.** Devices should not use static keys for unlocking. While they may have a unique shared secret used for authenticating with a key server, unique session tokens should be used for unlocking devices, and these tokens should expire in a short amount of time.
- **Protect against replay attacks.** Once a session token has been used to unlock a device (see above), that token should no longer be accepted. Expiring session keys and protection from replay attacks are available in standard protocols such as Kerberos v5 and OAuth v2.
- **Encrypt in transit.** Keys and other secrets should never be sent over a plain-text channel.
- **Implement multi-factor authentication.** Higher consideration should be given to protocols that offer multi-factor authentication for increased security, as some smart locks currently offer.
- **Monitor applications with access to data.**
- **Create specific access controls.**
- **Avoid custom encryption.**
- **Use and validate certificates wherever possible.**
- **Perform ongoing, periodic, thorough, manual, security assessments.**
- **Be aware of common security flaws.** These can be identified by industry resources such as the Open Web Application Security Project (OWASP) Top Ten, which include:
    - A1 Injection
    - A2 Broken Authentication and Session Management
    - A3 Cross-Site Scripting (XSS)
    - A4 Insecure Direct Object References
    - A5 Security Misconfiguration

  - A6 Sensitive Data Exposure
  - A7 Missing Function Level Access Control
  - A8 Cross-Site Request Forgery (CSRF)
  - A9 Using Components with Known Vulnerabilities
  - A10 Unvalidated Redirects and Forwards
- Adhere to recommendations outlined by industry bodies, such as the OWASP Secure Coding Practices.

3.5.3 Input Validation

- Conduct all data validation on a trusted system (e.g. the server).
- Identify all data sources and classify them into trusted and untrusted. Validate all data from untrusted sources (e.g. Databases, file streams, etc.).
- There should be a centralized input validation routine for the application.
- Specify proper character sets, such as UTF-8, for all sources of input.
- Encode data to a common character set before validating (*Canonicalize*).
- All validation failures should result in input rejection.
- Determine if the system supports UTF-8 extended character sets and if so, validate after UTF-8 decoding is completed.
- Validate all client provided data before processing, including all parameters, URLs and HTTP header content (e.g. Cookie names and values). Be sure to include automated post backs from JavaScript, Flash or other embedded code.
- Verify that header values in both requests and responses contain only ASCII characters.
- Validate data from redirects. (An attacker may submit malicious content directly to the target of the redirect, thus circumventing application logic and any validation performed before the redirect.)
- Validate for expected data types.
- Validate data range and length.
- Validate all input against a "white" list of allowed characters whenever possible.
- If any potentially *hazardous characters* must be allowed as input, be sure that you implement additional controls like output encoding, secure task specific APIs and account for the utilization of that data throughout the application. Examples of common hazardous characters include:

  < > " ' % ( ) & + \ \' \"
- If your standard validation routine cannot address the following inputs, then they should be checked discretely.
  - Check for null bytes (%00)
  - Check for new line characters (%0d, %0a, \r, \n)

- Check for “dot-dot-slash" (../ or ..\) path alterations characters. In cases where UTF-8 extended character set encoding is supported, address alternate representation like: %c0%ae%c0%ae/

(Utilize *canonicalization* to address double encoding or other forms of obfuscation attacks)

3.5.4 Output Encoding

- Conduct all encoding on a trusted system (e.g. the server).
- Utilize a standard, tested routine for each type of outbound encoding.
- *Contextually output encode* all data returned to the client that originated outside the application's *trust boundary*. *HTML entity encoding* is one example, but does not work in all cases.
- Encode all characters unless they are known to be safe for the intended interpreter.
- Contextually *sanitize* all output of un-trusted data to queries for SQL, XML, and LDAP.
- *Sanitize* all output of un-trusted data to operating system commands.

3.5.5 Authentication and Password Management

- Require authentication for all pages and resources, except those specifically intended to be public.
- All authentication controls must be enforced on a trusted system (e.g. the server).
- Establish and utilize standard, tested, authentication services whenever possible.
- Use a centralized implementation for all authentication controls, including libraries that call external authentication services.
- Segregate authentication logic from the resource being requested and use redirection to and from the centralized authentication control.
- All authentication controls should fail securely.
- All administrative and account management functions must be at least as secure as the primary authentication mechanism.
- If your application manages a credential store, it should ensure that only cryptographically strong one-way salted hashes of passwords are stored and that the table/file that stores the passwords and keys is write-able only by the application. Do not use the MD5 algorithm if it can be avoided.
- Password hashing must be implemented on a trusted system (e.g. the server).
- Validate the authentication data only on completion of all data input, especially for *sequential authentication* implementations.

- Authentication failure responses should not indicate which part of the authentication data was incorrect. For example, instead of "Invalid username" or "Invalid password", just use "Invalid username and/or password" for both. Error responses must be truly identical in both display and source code.
- Utilize authentication for connections to external systems that involve sensitive information or functions.
- Authentication credentials for accessing services external to the application should be encrypted and stored in a protected location on a trusted system (e.g. the server). The source code is NOT a secure location.
- Use only HTTP POST requests to transmit authentication credentials.
- Only send non-temporary passwords over an encrypted connection or as encrypted data, such as in an encrypted email. Temporary passwords associated with email resets may be an exception.
- Enforce password complexity requirements established by policy or regulation. Authentication credentials should be sufficient to withstand attacks that are typical of the threats in the deployed environment (e.g. requiring the use of alphabetic as well as numeric and/or special characters).
- Enforce password length requirements established by policy or regulation. Eight characters is commonly used, but 16 is better. Also consider the use of multi-word password phrases.
- Password entry should be obscured on the user's screen (e.g. on web forms use the input type "password").
- Enforce account disabling after an established number of invalid login attempts (five attempts is common). The account must be disabled for a period of time sufficient to discourage brute force guessing of credentials, but not so long as to allow for a denial-of-service attack to be performed.
- Password reset and changing operations require the same level of controls as account creation and authentication.
- Password reset questions should support sufficiently random answers (e.g. "favorite book" is a bad question because “The Bible” is a very common answer).
- If using email based resets, only send email to a pre-registered address with a temporary link/password.
- Temporary passwords and links should have a short expiration time.
- Enforce the changing of temporary passwords on the next use.
- Notify users when a password reset occurs.
- Prevent password re-use.
- Passwords should be at least one day old before they can be changed, to prevent attacks on password re-use.

- Enforce password changes based on requirements established in policy or regulation. The time between resets must be administratively controlled; critical systems may require more frequent changes.
- Disable "remember me" functionality for password fields.
- The last use (successful or unsuccessful) of a user account should be reported to the user at their next successful login.
- Implement monitoring to identify attacks against multiple user accounts, utilizing the same password. This attack pattern is used to bypass standard lockouts when user IDs can be harvested or guessed.
- Change all vendor-supplied default passwords and user IDs or disable the associated accounts.
- Re-authenticate users prior to performing critical operations.
- Use *Multi-Factor Authentication* for highly sensitive or high value transactional accounts.
- If using third party code for authentication, inspect the code carefully to ensure it is not affected by any malicious code.

3.5.6 Session Management

- Use the server or framework's session management controls. The application should only recognize these session identifiers as valid.
- Session identifier creation must always be done on a trusted system (e.g. the server).
- Session management controls should use well vetted algorithms that ensure sufficiently random session identifiers.
- Set the domain and path for cookies containing authenticated session identifiers to an appropriately restricted value for the site.
- Logout functionality should fully terminate the associated session or connection.
- Logout functionality should be available from all pages protected by authorization.
- Establish a session inactivity timeout that is as short as possible, based on balancing risk and business functional requirements. In most cases it should be no more than several hours.
- Disallow persistent logins and enforce periodic session terminations, even when the session is active, especially for applications supporting rich network connections or connections to critical systems. Termination times should support business requirements and the user should receive sufficient notification to mitigate negative impacts.

- If a session was established before login, close that session and establish a new session after a successful login.
- Generate a new session identifier on any re-authentication.
- Do not allow concurrent logins with the same user ID.
- Do not expose session identifiers in URLs, error messages or logs. Session identifiers should only be located in the HTTP cookie header. For example, do not pass session identifiers as GET parameters.
- Protect server side session data from unauthorized access, from other users of the server, by implementing appropriate access controls on the server.
- Generate a new session identifier and deactivate the old one periodically. This can mitigate certain session hijacking scenarios where the original identifier was compromised.
- Generate a new session identifier if the connection security changes from HTTP to HTTPS, as can occur during authentication. Within an application, it is recommended to consistently utilize HTTPS rather than switching between HTTP to HTTPS.
- Supplement standard session management for sensitive server-side operations, such as account management, by utilizing per-session strong random tokens or parameters. This method can be used to prevent *Cross Site Request Forgery* attacks.
- Supplement standard session management for highly sensitive or critical operations by utilizing per-request, as opposed to per-session, strong random tokens or parameters.
- Set the "secure" attribute for cookies transmitted over an TLS connection.
- Set cookies with the HttpOnly attribute, unless you specifically require client-side scripts within your application to read or set a cookie's value.

3.5.7 Access Control

- Use only trusted system objects (e.g. server side session objects) for making access authorization decisions.
- Use a single site-wide component to check access authorization. This includes libraries that call external authorization services.
- Access controls should fail securely.
- Deny all access if the application cannot access its security configuration information.
- Enforce authorization controls on every request, including those made by server side scripts, "includes" and requests from rich client-side technologies like AJAX and Flash.
- Segregate privileged logic from other application code.

- Restrict access to files or other resources, including those outside the application's direct control, to only authorized users.
- Restrict access to protected URLs to only authorized users.
- Restrict access to protected functions to only authorized users.
- Restrict direct object references to only authorized users.
- Restrict access to services to only authorized users.
- Restrict access to application data to only authorized users.
- Restrict access to user and data attributes and policy information used by access controls.
- Restrict access security-relevant configuration information to only authorized users.
- Server side implementation and presentation layer representations of access control rules must match.
- If state data must be stored on the client, use encryption and integrity checking on the server side to catch state tampering.
- Enforce application logic flows to comply with business rules.
- Limit the number of transactions a single user or device can perform in a given period of time. The transactions/time should be above the actual business requirement, but low enough to deter automated attacks.
- Use the "referrer" header as a supplemental check only, it should never be the sole authorization check, as it is can be spoofed.
- If long authenticated sessions are allowed, periodically re-validate a user's authorization to ensure that their privileges have not changed and if they have, log the user out and force them to re-authenticate.
- Implement account auditing and enforce the disabling of unused accounts (e.g. after no more than 30 days from the expiration of an account's password).
- The application must support disabling of accounts and terminating sessions when authorization ceases (e.g. changes to role, employment status, business process, etc.).
- Service accounts or accounts supporting connections to or from external systems should have the least privilege possible.
- Create an Access Control Policy to document an application's business rules, data types and access authorization criteria and/or processes so access can be properly provisioned and controlled. This includes identifying access requirements for both the data and system resources.

3.5.8 Cryptographic Practices

- All cryptographic functions used to protect secrets from the application user must be implemented on a trusted system (e.g. the server).
- Protect master secrets from unauthorized access.
- Cryptographic modules should fail securely.
- All random numbers, file names, GUIDs and strings should be generated using the cryptographic module's approved random number generator when these values are intended to be un-guessable.
- Cryptographic modules used by the application should be compliant to FIPS 140-2 or an equivalent standard. See http://csrc.nist.gov/groups/STM/cmvp/validation.html
- Establish and utilize a policy and process for how cryptographic keys will be managed.

3.5.9 Error Handling and Logging

- Do not disclose sensitive information in error responses, including system details, session identifiers or account information.
- Use error handlers that do not display debugging or stack trace information.
- Implement generic error messages and use custom error pages.
- The application should handle application errors and not rely on the server configuration.
- Properly free allocated memory when error conditions occur.
- Error handling logic associated with security controls should deny access by default.
- All logging controls should be implemented on a trusted system (e.g. the server).
- Logging controls should support both success and failure of specified security events.
- Ensure logs contain important *log event data.*
- Ensure log entries including un-trusted data will not execute as code in the intended log viewing interface or software.
- Restrict access to logs to only authorized individuals.
- Utilize a master routine for all logging operations.
- Do not store sensitive information in logs, including unnecessary system details, session identifiers or passwords.
- Ensure that a mechanism exists to conduct log analysis.
- Log all:
    - Input validation failures
    - Authentication attempts, especially failures
    - Access control failures

- Log all input validation failures.
- Log all authentication attempts, especially failures.
- Log all access control failures.
- Log all apparent tampering events, including unexpected changes to state data.
- Log attempts to connect with invalid or expired session tokens.
- Log all system exceptions.
- Log all administrative functions, including changes to the security configuration settings.
- Log all backend TLS connection failures.
- Log cryptographic module failures.
- Use a cryptographic hash function to validate log entry integrity.

3.5.10 Data Protection

- Implement least privilege, restrict users to only the functionality, data and system information that is required to perform their tasks.
- Protect all cached or temporary copies of sensitive data stored on the server from unauthorized access and purge those temporary working files as soon as they are no longer required.
- Encrypt highly sensitive stored information, like authentication verification data, even on the server side. Always use well vetted algorithms, see Section 7.2.6 for additional guidance.
- Protect server-side source-code from being downloaded by a user.
- Do not store passwords, connection strings or other sensitive information in clear text or in any non-cryptographically secure manner on the client side. This includes embedding in insecure formats such as MS viewstate, Adobe flash or compiled code.
- Remove comments in user accessible production code that may reveal backend system or other sensitive information.
- Remove unnecessary application and system documentation as this can reveal useful information to attackers.
- Do not include sensitive information in HTTP GET request parameters.
- Disable auto complete features on forms expected to contain sensitive information, including authentication.
- Disable client side caching on pages containing sensitive information. Cache-Control: no-store, may be used in conjunction with the HTTP header control "Pragma: no-cache", which is less effective, but is HTTP/1.0 backward compatible.
- The application should support the removal of sensitive data when that data is no longer required (e.g. personal information or certain financial data).

- Implement appropriate access controls for sensitive data stored on the server. This includes cached data, temporary files and data that should be accessible only by specific system users.

3.5.11 Communication Security

- Implement encryption for the transmission of all sensitive information. This should include TLS for protecting the connection and may be supplemented by discrete encryption of sensitive files or non-HTTP based connections.
- TLS certificates should be valid and have the correct domain name, and be installed with intermediate certificates when required.
- Failed TLS connections should not fall back to an insecure connection.
- Utilize TLS connections for all content requiring authenticated access and for all other sensitive information.
- Utilize TLS for connections to external systems that involve sensitive information or functions.
- Utilize a single standard TLS implementation that is configured appropriately.
- Specify character encodings for all connections.
- Filter parameters containing sensitive information from the HTTP referrer, when linking to external sites.

3.5.12 System Configuration

- Ensure servers, frameworks and system components are running the latest approved version.
- Ensure servers, frameworks and system components have all patches issued for the version in use.
- Turn off directory listings.
- Restrict the web server, process and service accounts to the least privileges possible.
- When exceptions occur, fail securely
- Remove all unnecessary functionality and files.
- Remove test code or any functionality not intended for production prior to deployment.
- Prevent disclosure of your directory structure in the robots.txt file by placing directories not intended for public indexing into an isolated parent directory. Next, "Disallow" that entire parent directory in the robots.txt file rather than disallowing each individual directory.
- Define which HTTP methods, Get or Post, the application will support and whether it will be handled differently in different pages in the application.

- Disable unnecessary HTTP methods such as WebDAV extensions. If an extended HTTP method that supports file handling is required, utilize a well-vetted authentication mechanism.
- If the web server handles both HTTP 1.0 and 1.1, ensure that both are configured in a similar manor or insure that you understand any difference that may exist (e.g. handling of extended HTTP methods).
- Remove unnecessary information from HTTP response headers related to the OS, web-server version and application frameworks.
- The security configuration store for the application should be able to be output in human readable form to support auditing.
- Implement an asset management system and register system components and software in it.
- Isolate development environments from the production network and provide access only to authorized development and test groups. Development environments are often configured less securely than production environments. Attackers may use this difference to discover shared weaknesses or as an avenue for exploitation.
- Implement a software change control system to manage and record changes to the code both in development and production.

3.5.13 Database Security

- Use strongly typed *parameterized queries.*
- Utilize input validation and output encoding and be sure to address metacharacters. If these fail, do not run the database command.
- Ensure that variables are strongly typed.
- The application should use the lowest possible level of privilege when accessing the database.
- Use secure credentials for database access.
- Connection strings should not be hard coded within the application. Connection strings should be stored in a separate configuration file on a trusted system and they should be encrypted.
- Use stored procedures to abstract data access and allow for the removal of permissions to the base tables in the database.
- Close the connection as soon as possible.
- Remove or change all default database administrative passwords. Utilize strong passwords/phrases or implement multi-factor authentication.
- Turn off all unnecessary database functionality (e.g. unnecessary stored procedures or services, utility packages, install only the minimum set of features and options required (surface area reduction)).
- Remove unnecessary default vendor content (e.g. sample schemas).

- Disable any default accounts that are not required to support business requirements.
- The application should connect to the database with different credentials for every trust distinction (e.g. user, read-only user, guest, administrators).

3.5.14 File Management

- Do not pass user supplied data directly to any dynamic include function.
- Require authentication before allowing a file to be uploaded.
- Limit the type of files that can be uploaded to only those types that are needed for business purposes.
- Validate uploaded files are the expected type by checking file headers. Checking for file type by extension alone is not sufficient.
- Do not save files in the same web context as the application. Files should either go to the content server or in the database.
- Prevent or restrict the uploading of any file that may be interpreted by the web server.
- Turn off execution privileges on file upload directories.
- Implement safe uploading in UNIX by mounting the targeted file directory as a logical drive using the associated path or the chrooted environment.
- When referencing existing files, use a white list of allowed file names and types. Validate the value of the parameter being passed and if it does not match one of the expected values, either reject it or use a hard coded default file value for the content instead.
- Do not pass user supplied data into a dynamic redirect. If this must be allowed, then the redirect should accept only validated, relative path URLs.
- Do not pass directory or file paths, use index values mapped to pre-defined list of paths.
- Never send the absolute file path to the client.
- Ensure application files and resources are read-only.
- Scan user uploaded files for viruses and malware.

3.5.15 Memory Management

- Utilize input and output control for un-trusted data.
- Double check that the buffer is as large as specified.
- When using functions that accept a number of bytes to copy, such as strncpy(), be aware that if the destination buffer size is equal to the source buffer size, it may not NULL-terminate the string.
- If calling the function in a loop, check buffer boundaries and make sure there is no danger of writing past the allocated space.

- Truncate all input strings to a reasonable length before passing them to the copy and concatenation functions.
- Specifically close resources, don't rely on garbage collection (e.g. connection objects, file handles, etc.).
- Use non-executable stacks when available.
- Avoid the use of known vulnerable functions (e.g. printf, strcat, strcpy etc.).
- Properly free allocated memory upon the completion of functions and at all exit points.

3.5.16 General Coding Best Practices

- Use tested and approved managed code rather than creating new unmanaged code for common tasks.
- Utilize task specific built-in APIs to conduct operating system tasks. Do not allow the application to issue commands directly to the Operating System, especially through the use of application initiated command shells.
- Use checksums or hashes to verify the integrity of interpreted code, libraries, executables, and configuration files.
- Utilize locking to prevent multiple simultaneous requests or use a synchronization mechanism to prevent race conditions.
- Protect shared variables and resources from inappropriate concurrent access.
- Explicitly initialize all your variables and other data stores, either during declaration or just before the first usage.
- In cases where the application must run with elevated privileges, raise privileges as late as possible and drop them as soon as possible.
- Avoid calculation errors by understanding your programming language's underlying representation and how it interacts with numeric calculation. Pay close attention to byte size discrepancies, precision, signed/unsigned distinctions, truncation, conversion and casting between types, "not-a-number" calculations and how your language handles numbers that are too large or too small for its underlying representation.
- Do not pass user supplied data to any dynamic execution function.
- Restrict users from generating new code or altering existing code.
- Review all secondary applications, third party code and libraries to determine business necessity and validate safe functionality, as these can introduce new vulnerabilities.
- Implement safe updating. If the application will utilize automatic updates, then use cryptographic signatures for your code and ensure your download clients verify those signatures. Use encrypted channels to transfer the code from the host server.

3.5.17 Update Process

Technology iterates; disruptors innovate; adversaries evolve. Solutions deployed today will need to be updated in the near future to account for evolving attack scenarios. Needs will vary from organization to organization, but the following are some of the most common elements of an effective update process:

1. **Inventory**. Establish and maintain an understanding of all deployed production systems, including the various data about those systems (such as quantity, physical locations, operating systems, versions, etc.).
2. **Qualify risk**. Map listing of known vulnerabilities to inventory of production systems; measure severity criticality per vulnerability. Consider the impact a compromise would have: articulate business impact multiplied by likelihood.
3. **Quantify action thresholds**. Determine what type of issue or period of time warrants an update (i.e. mandatory forced update, optional periodic update, critical security update, etc.). Develop a standard operating procedure to follow when remediation is necessary.
4. **Updates as a feature**. Build into your system a feature for update or patching, where possible. If updating is not possible, and replacement is the only solution for remediating vulnerabilities, communicate that to the customer. Articulate the standard operating procedure and associated timelines for performing updates when required.

3.5.18 The Human Element

Humans are always the weakest link in the security chain, as humans are susceptible to social engineering and other wetware attacks, they often lack awareness about effective security measures and tend to default to trust. As such, organizations should take proactive measures in order to combat the human element. Needs will vary from organization to organization, but the following are some of the most common elements of an effective approach to attacks against the human element:

- Promote a culture of security:
    - **Obtain executive buy-in**. Security starts from the CEO and then flows down to the rest of the organization, not pushed upwards from a security technician. Ensure that security is seen as the business-critical discipline that it is; the success of the program depends on it.
    - **Security as a business enable**. Align security with the overall objectives of the business and how security can help improve pursuit of those objectives. Security should be seen as a way to enhance the development of the business, rather than a cost to be minimized.
    - **Evangelize**. Security is everyone's job, not just the security or IT teams.
    - **Convey the *Why***. People need a reason to do something, and need to believe in that reason. Help employees understand why security in their job function is important to their professional and personal lives.

- **Reward excellence**. Recognize those individuals or teams who are contributing positive accomplishments to the security program. Make the program attractive to others to emulate.
- **Make it fun**. The success to create a culture of security is getting people to want to do it. Find out what your people enjoy and tap into that.
- **Cross the aisle**. Galvanize different groups, especially those not traditional in the security organization, to become allies and advocates in the pursuit of the security mission. Momentum builds when it is across lines of group division.
- **Create, and empower, a security leader**. Security should not be a subset of someone's job, it should be the entirely of that leader's job. And that leader needs peer level status with the executive leaders of the organization
- **Train everyone**. Security is everyone's job, so this should not be limited to just those in the security or IT teams. This would include executives, group leaders, end users, developers, technical staff, non-technical staff, and pretty much everyone else across the organization.
- **Solicit feedback**. Find out especially what your people think is annoying. Keep an open dialogue with the entire organization about the security program. This will not only help identify breakdowns, it will also foster participation, raise awareness, reinforce the significance, identify areas to focus on for training, and reveal pain points that might be causing people to circumvent security controls.
- **Start now**. Not next quarter, not next fiscal year, not when budget becomes available. Now.

- Train end users, executives, developers, and security staff
- Implement awareness program
- Beware of social engineering
- Define and publish policies

3.5.19 Validation

"Trust, but verify."

This encapsulates the fundamentally most critical aspect of effective security: Even when security has been prioritized and adequately resourced, vulnerabilities can still be unwittingly introduced. There is a fundamental difference between *building* a solution and *breaking* a solution. In that vein, every security program needs an effective **validation mechanism** in order to investigate systems for **unintended security vulnerabilities**.

Vetting a security partner can be challenging in an increasingly confusing security marketplace where many offerings sound similar and a company's effectiveness or skill level may not be clear. When selecting a security expert to partner with, attempt to understand the following attributes:

- Approach & Methodology

- Pedigree
- Domain expertise
- Scope of services rendered