Communications on Applied Nonlinear Analysis
ISSN: 1074-133X
Vol 32 No. 4s (2025)

Comparative Study of Functionality Isolation in Multi-Application Smart Cards

Aarya Patil[1], Pranali R Navghare[2], Geetanjali V. Kale[3], Nandnandan Patil[4], Aditya Malu[5], Charul Nampalliwar[6]

[1]Department of Computer Engineering, Pune Institute of Computer Technology, Pune, Maharashtra, India
Email: patilaarya046@gmail.com

[2]Department of Computer Engineering, Pune Institute of Computer Technology, Pune, Maharashtra, India
Email: prnavghare@pict.edu

[3]Department of Computer Engineering, Pune Institute of Computer Technology, Pune, Maharashtra, India
Email: gvkale@pict.edu

[4]Department of Computer Engineering, Pune Institute of Computer Technology, Pune, Maharashtra, India
Email: nandnandanpatil@gmail.com

[5]Department of Computer Engineering, Pune Institute of Computer Technology, Pune, Maharashtra, India
Email: adityamalu2509@gmail.com

[6]Department of Computer Engineering, Pune Institute of Computer Technology, Pune, Maharashtra, India
Email: nampalliwarcharul@gmail.com

Article History:

Received: *01-10-2024*

Revised: *29-11-2024*

Accepted: *07-12-2024*

Abstract:

A variety of domains, including finance, government, healthcare, and transportation, extensively use multi-application smart cards due to their ability to securely manage data and execute multiple applications. In such systems, the critical challenge remains in the isolation of the application for multiple effective uses so that no application gets unauthorized access to data and resources. In general, performance and implementation restrictions affect traditional isolation mechanisms such as OS partitioning, cryptographic techniques, and hardware-based TEEs. The methodology, context-aware approach for cross-resource sharing manages access dynamically based on contextual parameters such as the state of applications, security levels, or user activity without impairing the effects of security and while maintaining adequate performance efficiency. A fair overview of the existing methodologies coupled with the enforcement of dynamic access policies underlines potential in future development for multi-application smart card environments.

Keywords: multi-application smart cards, Secure resource management, Application isolation, Cross-application interaction

1. Introduction

Until the end of the 90's, it was very difficult to have more than one application running on a smart card. Multi-Application smart cards have made it possible to implement several applications and to dynamically load new ones during the card's active life. [17]

Multi-application smart cards are highly versatile cards that support a great number of functions and applications. They either use a memory chip or an ICC as a secure microcontroller, which can be used for access, payment, and identity. These cards can apply contactless operation using radio frequency

technology or through physical contact with readers. A microcontroller allows safely storing large data volumes and executing encryption, biometric authentication, and software managing applications in multiple numbers. There are three primary types of smart cards: plastic cards, SIM cards, and USB tokens. All of them follow the international standards ISO/IEC 7816 and ISO/IEC 14443. [10] [4]

There are two types: contactless multi-application smart cards and contact multi-application smart cards. Contactless multi-application smart cards make use of wireless radio frequency signals that communicate with a reader in close proximity to access multiple services without interruptions, for example, while making a payment or accessing a security service. The contact multi-application smart cards must be inserted into a reader directly to transfer data across physical contact points.[4]

One of the biggest problems with these cards, although they have many features that are really good, is maintaining a secure separation between programs. With the fact that multiple applications are stored on one card, it needs to be understood that no application can access or gain another's memory, processing power, or data. This separation should help protect private information, maintain security, and ensure the integrity of any program overall. The difficulty is in maintaining this isolation without sacrificing efficiency, since standard partitioning methods frequently don't offer sufficient security.[10]

Essentially, the problem lies in cross-resource isolation. Cross-resource isolation aims to prevent programs from accessing shared resources unknowingly or with malicious intent, hence thereby violating security. In effect, isolation guarantees security but may also limit the card's ability to make the best use of its resources, leading to probable performance issues.[11][10]

This survey paper discusses a context aware approach to dynamic resource allocation and a novel mechanism of sharing resources in a multi-application smart card. The technique optimizes both security as well as performance through the real-time distribution of contextual information on resources. This work enhances awareness regarding cross-resource sharing in multi-application smart cards by establishing an extensive review of the present methods, and by developing this new approach, it also provides potential insight into further studies and development in this realm.

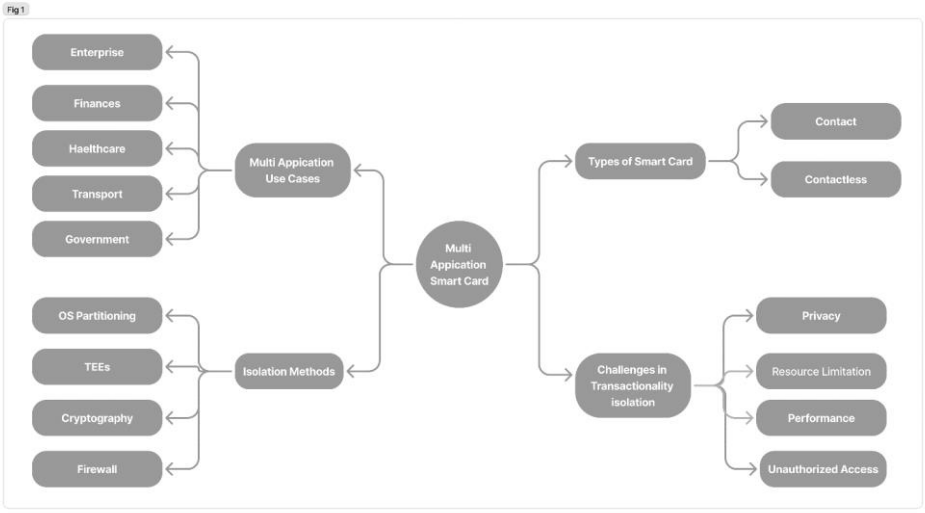

Fig. 1. High-level Overview

2. Multi-application Smart Cards

2.1. Smart Cards

Smart cards combine memory with a microprocessor, thus have found wide applicability in various sectors, including finance, telecommunication, healthcare, and transportation, for performing secure transactions and data management. COS (Card Operating System) serves as the to a smart card: the environment in which operating resources would be managed and communication protocols as well as secure data executed by encryption and authentication mechanisms. In general, the COS architecture is broken down into two layers: one is the micro-kernel layer that interfaces with hardware, and the other is the logical function layer that oversees all higher-level operations, including file handling, command processing, and security controls.[2]

2.2. Applications of Smart Cards

- Finance: Smart cards are used in the finance sector to hold data from the customers to make safe withdrawals and payments in a secure way with encryption as well as authentication to protect against fraud and unauthorized usages. [3]
- Enterprise ID: Smart cards can be part of enterprise identity management, as they store data related to a user in an embedded chip. They provide access to authorized users through mutual authentication or biometric matching, followed by advanced encryption processes in line with international standards. [4]
- Government: Smart cards are at the centre of managing government identities with secure access control and credentialing by standards like FIPS 201. They made possible interoperable systems that offer much more security to government employees and their constituents. [5]
- Health Care: Smart cards place the electronic health records in safe places to meet the HITECH meaningful use criteria. They allow secure and controlled identity verification and protection of identities during implementation to ensure privacy of patients and follows the healthcare systems.[6]
- Transportation: Smart cards are revolutionizing the transportation industry by enabling contactless fare payments and replacing transit-specific cards with prepaid or network-branded cards, reducing administrative costs and offering passengers ready, easy access to payment. [7]

2.3. Multi-application Smart Card architecture

- Card Components: There are several basic components at the architecture level: processor/controller that runs applications as well as regulates terminal communication; Program Memory (ROM) where the operating system and programs are stored; Data memory (E2PROM) where the data for an application is stored; Working Storage (RAM) used for short-term execution requirements. [1]
- Operating System: OS ensures that the applications are not intermingled and offers Memory Management through unique allocation of memory. It supports Application Downloading and Updating through secure channels, providing an Interpreted Language for running compatibility across card types.[1]
- Application Management: Application Management contains high-level Application Selection techniques. An Application Directory helps terminals decide which of their applications are most appropriate and Identical and Cooperative Applications process data in parallel or cooperative.[1]

- Security Features: In the architecture, sophisticated features of security are present, such as key management for safe storage of cryptographic keys and authentication mechanisms that make the card authenticate users actively against possible risks such as cloning. [1]
- Terminal Interaction: Terminal Interaction is a very critical component in card-to-terminal communication. It supports multiple protocols for communication such as ISO 7816 and EMV. As it forms the fallback mechanism in case it goes offline; solid connections become much more crucial in real-time transactions.[1]
- User Interface: With a well-designed User Interface interaction across various applications, just like payments or even access to such can be created, and chances for the user increase through a feedback mechanism that shows the result of a transaction.[1]
- Collaboration Mechanisms: It allows cards to be shared between organizations while enforcing data separation, an important requirement for co-branding applications, such as a bank-transit partnership, requiring stronger security requirements. [1]
- Deployment Considerations: They are primarily based on market segmentation with functionalities being built according to user demographics. Clearly meeting regulatory requirements regarding security and data protection is important to establish trust and encourage adoption.[1]

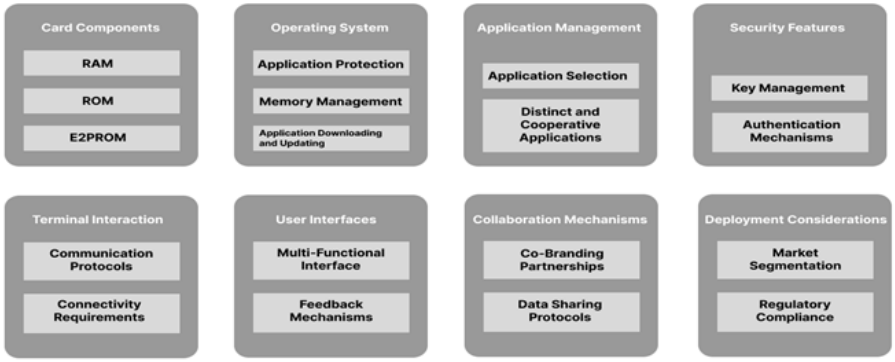

Fig. 2. Multi-Application smart card Architecture

3. Functionality Isolation

Smart cards have different new, multi-application storage requirements and need to guarantee secure co-existence of many applets residing in their NVM (Non-Volatile Memory). Strong isolation between the applets needs to be ensured so that no malicious interference or unauthorized access takes place. Smart cards have been used historically only for a single application, but with the advent of JVM (Java Virtual Machine) and GlobalPlatform specifications, it is now possible to run several applets concurrently on the same card. This capability introduces new security challenges, such as secure bytecode interpretation and resource partitioning. If not managed properly, a malicious applet might, through one of several vulnerabilities, gain access to or manipulate data in the ownership of some other applet. These concerns are addressed through a variety of isolation mechanisms: from partitioning an operating system to cryptographic techniques, TEEs, and mechanism firewalls, enforcing a strict separation of functionalities without compromising the security of sensitive operations.[16]

3.1. Operating System Partitioning

One of the defence mechanisms used by smart cards for cross-resource sharing systems is partitioning, which is based on the foundation of an architectural OS. In the JavaCard, multiple Java applets run on a single card, with the JVM managing them, enforcing isolation by putting every applet in a different sandbox.[11] However, a vulnerability within the JVM or OS-level components may still be exploitable by malicious applets to bypass those means of isolation. Still, it is widely in use because of its simplicity in implementation and management.[13]

Given a logical address L, segment number S, and offset O, the physical address P is calculated using the segment table. Let ST[S] be the starting address and Length[S] be the segment length. Then:

- $P = ST[S] + O$ if $O < Length[S]$
- $\geq Length[S] \Rightarrow$ Cross-Border Violation Detected.

3.2. Cryptographic Techniques

Cryptography is another technology for application separation on multi-application smart cards. Applications will, therefore, have unique cryptographic keys, so even if there is a breach in physical access to the card, the cryptographic data of other applications remains secure. However, cryptographic operations especially key generation, management, and exchange are significant consumers of smart card's scarce computing power.[9] In this scenario, side-channel attacks could also attack the encryption process itself.[12]

Diffie-Hellman key exchange based on base g and prime number p. Public key of the smart card is denoted as C_{pub}, private key as C_{priv}; similarly, the public key for the host is H_{pub} while its private key is H_{priv}. Here, using shared secret between the smart card and host establishes the session key $K_{session}$. Let M be the message exchanged by the smart card and the host, and $E_K(M)$ denote the encryption of M using the secret key K. Define Cert(X) as the certificate of public key X, and Valid(Cert(X)) is a function that returns true if the certificate Cert(X) is valid. Hash output of X is Hash(X).

- $C_{pub} = g^{Cpriv} \bmod p$, $H_{pub} = g^{Hpriv} \bmod p$
- If $Valid(Cert(H_{pub})) \Rightarrow$ Proceed; otherwise, terminate the session.
- $S_C = H_{pub}{}^{Cpriv} \bmod p$, $S_H = C_{pub}{}^{Hpriv} \bmod p$, If $S_C = S_H \Rightarrow K_{session} = Hash(S_C)$
- If $E_{Hpub}(K_{session}) \leftrightarrow E_{Cpub}(K_{session})$, proceed; otherwise, terminate.
- If $M_{encrypted} = E_{Ksession}(M)$, then $M = D_{Ksession}(M_{encrypted})$.

3.3. Trusted Execution Environments (TEEs)

TEEs refer to some kind of protected space within a processor that offers an isolated environment where the applications running in it cannot be accessed or tampered with by any other application on the same card. Hardware isolation assures a TEE that even an attack targeting the full stack, even the most sophisticated one, will not breach the security of an application. This is the foundation of the GlobalPlatform Card specification that uses the Security Domains combined with TEEs to realize strong isolation between applications, especially those considered sensitive like payment and identity management systems.[14]

- $\forall x \in$ Processes, (trusted(x) $\rightarrow$ isolate(x, untrusted processes))
- $\forall y \in$ Data, access(y) $\leftrightarrow$ credentials(y) $\geq$ required level

- $\forall z \in Data, encrypted(z) \land hash(z) = hash\ stored$
- $\forall m \in Data, share(m) \rightarrow (encrypt(m) \land authenticate(receiver))$

3.4. Firewall Mechanisms

Firewall isolation on smart cards provides robust access control that delimits very clear limits in ensuring the applications work without interferences or leakage of data. It uses user-centric policies, where one can dynamically change permissions and indicate which application must be able to communicate or even access specific data. The system further secures attempted unauthorized access by logging attempts and even providing real-time threat detection through analysis patterns of interaction. Besides this, encrypted channels protect the data flow of communication when data must be transferred from or to the computer of a client.[8]

Access Control Logic:

- $Access = (ClientAID \in ACL) \land (Credentials_{Client} = Valid)$
- $Access = (ResourceLevel \leq AccessLevel_{Client})$
- $Access = (ResourceID \notin -F_{Exception})$
- $Access = (CurrentTime < ExpiryTime)$
- $DelegatedAccess = (DelegationRequest_{Client} = Authorized)$

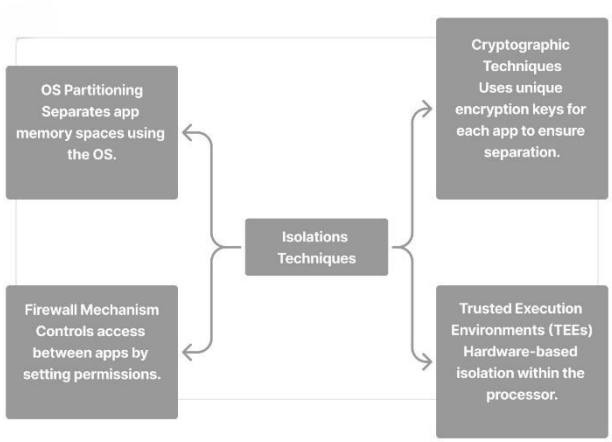

Fig. 3. Isolation Techniques

Criteria	Operating System Partitioning	Cryptographic Techniques	Trusted Execution Environments (TEEs)	Firewall Mechanism
Working	Separates applications by giving each app its own memory space, managed by the OS (e.g., JavaCard).[11]	Uses unique encryption keys for each app so that one app can't access another's data.[9]	Creates a secure area within the hardware where sensitive operations run isolated from others.[14]	Controls access to shared resources by allowing users to set permissions for app interactions.[8]

Examples	JavaCard OS, GlobalPlatform OS.[11]	Apps communicating with servers through encrypted channels.[9]	ARM TrustZone, Intel SGX.[14]	User-centric firewalls in multi-application smart card systems.[8]
Security	It offers good isolation if the OS and JVM are secure, but vulnerable to OS bugs.[13]	Provides strong security through encryption, ensuring apps can't "see" each other's data.[12]	Very secure since it's hardware-based, making software attacks almost impossible.[14]	Enhances isolation by preventing unauthorized access and managing app permissions dynamically.[8]
Performance Impact	Normally it performs well, but too many apps may slow things down.[11]	Can slow down the system because encryption takes a lot of computing power.[9]	May need extra hardware, which can slow things down but is generally optimized for high security.[14]	Minimal impact on performance if configured efficiently, but poor configuration can lead to bottlenecks.[8]
Scalability	Works well with lots of applications; ideal for high-volume environments.[11]	Works but can become sluggish with many apps due to increased encryption overhead.[9]	Less suitable for cheap cards, but great for high-end applications where security is paramount.[14]	Scales well with user demands; policies can adapt to manage increased application load.[8]
Pros	Easy to manage and implement across many apps.[11] Logical isolation is simple to enforce.[11]	Offers very strong encryption-based security.[9] Ensures secure communication between apps.[9]	Hardware-level isolation is highly secure.[14] Effective against software-based attacks.[14]	Empowers users with control over app interactions.[8] Dynamic access policies enhance security.[8]
Cons	Relies heavily on OS security, and if that fails, all apps are at risk.[11]	Computationally demanding, which can slow things down.[9] Vulnerable to side-channel attacks.[9]	High cost and complexity in implementation.[14] Needs specialized hardware, which isn't always practical.[14]	Security risks from misconfigurations; effectiveness relies on proper user management.[8]

Table 1. Comparison between Isolation Methods

4. Context Aware Systems

Context aware systems adapt dynamically according to the behavior of the environment. They use contextual information, such as location or time, user activity, environmental conditions, to provide certain services or responses. They can sense, process, and adapt to changes in context in real-time. Thus, they deliver personal experiences and improve decision-making. The complexity and ambiguity associated with the description of context in different scenarios call for a structured methodology for designing such systems. This includes three stages: (1) Getting insight into context. (2) Determining what components are needed to sense and adapt to context. (3) Determining the rules for how the system should adapt in different situations.[15]

5. Proposed methodology for Isolation

We propose a mechanism for Safe, Cross-function Resource Sharing in a Multi-Application Smart Card with applications kept isolated from each other: The development of the next steps makes clear proposition regarding such a mechanism. From context-aware access policies to dynamic management, this approach will control whatever applications' resource access might be made to others based on its role, current state and the entire security context. Hence, this framework denies unauthorized interactions while being able to provide fine-grained access control for cooperation in the isolated environments of applications.

- Identify Contextual Parameters

In our framework proposal, the step starts by identifying the contextual parameters that are relevant to access requests coming from applications. Key parameters include application state (active, idle, or blocked), time of access (with specific time constraints on permissions), location of the device (either inside a secure facility or outside) user interaction state (at present, whether a legitimate user is actively using the card or is unattended) current security threat level (which shows anomalies that have been detected or attacks that are ongoing) and application roles and identities, (Issuer, Acquirer, or Service Provider roles).

- Define Context-Based Access Policies

It then defines applications that have access to certain resources subject to conditions, based on context-based access policies. For example, the following policy might be stated: "If the application is active, and security level is low, it is acceptable to read shared files." Another may demand: "If the card is in a high-security place, then it is all right to share cryptographic keys between applications." To represent these and similar rules, we recommend using a policy specification language such as XACML or ad-hoc JSON/XML dialects.

- Implement a Context-Aware Access Manager (CAAM)

We propose designing a Context-Aware Access Manager (CAAM) as a central entity within the smart card environment. The CAAM will continuously monitor and collect contextual parameters from various sources, utilizing a Context Aggregator Module to synthesize these inputs into a coherent context profile. It will then match the current context against predefined policies and employ decision trees or rule-based engines to determine access permissions. Upon reaching a decision, the CAAM

will enforce it through a Policy Enforcement Point (PEP), directly controlling access to the requested resources.

- Implement Adaptable Policies with Real-Time Context Updates

We shall recommend mechanisms of adaptation when policy conditions in the running context change. For instance, if the threat level has mounted from low to high, then the system should automatically revoke those permissions deemed non-essential. A Dynamic Policy Manager can so facilitate smooth updates of policy without interrupting operations already in motion; security remains responsive and effective in countering shifting threats.

6. Use cases

6.1. Law Enforcement Access to Purchase History

In a criminal investigation, law enforcement officers may use a smart card with controlled access to retrieve relevant data from a suspect's transaction history, such as purchases related to weapons or suspicious items.

- Contextual Parameters: Officers could be granted access based on the severity of the case, the user's involvement in the investigation, or legal permissions (e.g., a court order).
- Location-Based Access: Such access could be restricted to specific locations, like police stations or investigative offices, ensuring the data is accessed in a secure environment.
- Time-Based Restrictions: Access to the transaction history may be granted only for a limited time during the investigation and revoked once no longer needed.
- Security Threat Level: If a potential threat (e.g., the individual is linked to a high-risk activity like terrorism), the system could escalate access to include more detailed purchasing patterns, allowing a quicker response.

6.2. Educational Institution Access and Privileges

A smart card issued to students at a university could manage access to facilities (libraries, labs) and digital resources (learning materials) based on the student's status, schedule, and location.

- Contextual Parameters: Students can access certain lab facilities or study materials only when enrolled in relevant courses and during assigned times.
- Location-Based Access: Access to facilities such as libraries or dormitories could be restricted based on whether the student is physically present on campus.
- Application State: A student might only be allowed to borrow equipment or check out books related to their current coursework, while other non-relevant resources remain inaccessible.
- Security Threat Level: If the student is flagged for academic misconduct or other violations, the card could limit privileges such as accessing study materials, attending certain classes, or borrowing resources.

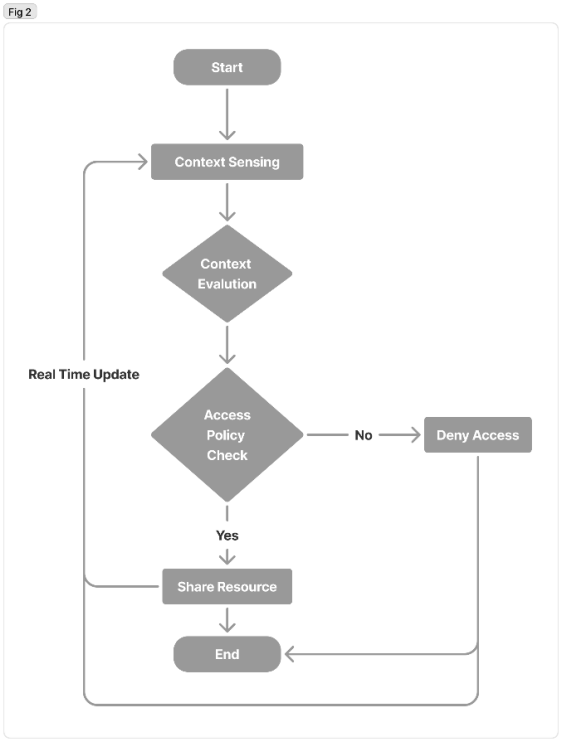

Fig. 4. Context aware Resource sharing

7. Conclusion

In this paper, an issue of isolation across different resources in multi-application smart cards has been addressed with a mechanism that adjusts access control policies dynamically with respect to real-time contextual information. Solutions such as OS partitioning, cryptographic techniques, and TEEs have been implemented but are generally weak in one aspect-strong in the other; either scalability is limited or complex in implementation. A new approach derives security requirements based on contextual data about application state, user interaction, and levels of security threats to allow for secure yet flexible resources sharing. Such a guarantee yields an optimal security framework with no loss of performance. Future work can be in the investigation of how machine learning models can promote stronger adaptability in context aware policies.

References

[1] Hendry, M. (2007). Multos. In: Multi-Application Smart Cards: Technology and Applications (pp. 90-97). Cambridge, UK: Cambridge University Press.
[2] Chen Yuqiang, Guo Jianlan, Hu Xuanzi and Liu Liang, "Design and implementation of Smart Card COS," 2010 International Conference on Computer Application and System Modeling (ICCASM 2010), Taiyuan, 2010, pp. V13-398-V13-402, doi: 10.1109/ICCASM.2010.5622819.
[3] Secure Technology Alliance. (2019). Biometric Payment Cards (Version 1.0). March 2019.
[4] Smart Card Alliance Physical Access Council. (March 2011). Smart Cards and Biometrics (Publication No. PAC-11002).
[5] Smart Card Alliance Physical Access Council & Identity Council. (February 2011). Personal Identity Verification Interoperability (PIV-I) for Non-Federal Issuers: Trusted Identities for Citizens across States, Counties, Cities and Businesses (Publication No. PAC-11001).

[6] Smart Card Alliance Healthcare Council. (February 2011; minor update July 2019). Getting to Meaningful Use and Beyond: How Smart Card Technology Can Support Meaningful Use of Electronic Health Records (Publication No. HC-11001).
[7] Smart Card Alliance Transportation Council. (March 2010). A Guide to Prepaid Cards for Transit Agencies (Publication No. TC-10002).
[8] Akram, Raja Naeem & Markantonakis, Konstantinos & Mayes, Keith. (2010). Firewall Mechanism in a User Centric Smart Card Ownership Model. 118-132. 10.1007/978-3-642-12510-2_9.
[9] Markantonakis, Konstantinos & Mayes, Keith. (2005). A Secure Channel Protocol for Multi-Application Smart Cards Based on Public Key Cryptography. International Federation for Information Processing Digital Library; Communications and Multimedia Security;. 10.1007/0-387-24486-7_6.
[10] Bakdi, I., Domingo-Ferrer, J., Posegga, J., & Schreckling, D. (2006). Towards a secure and practical multifunctional smart card. In Smart Card Research and Advanced Applications (pp. 16-31). Springer Berlin Heidelberg. DOI: 10.1007/11733447_2
[11] Xiaohui Cheng and Mengmeng Dou, "The research of dynamic multi-application smart card segment storage management model," 2012 IEEE International Conference on Computer Science and Automation Engineering, Beijing, China, 2012, pp. 267-270, doi: 10.1109/ICSESS.2012.6269457.
[12] D. Naccache and D. M'Raihi, "Cryptographic smart cards," in IEEE Micro, vol. 16, no. 3, pp. 14-24, June 1996, doi: 10.1109/40.502402.
[13] Anderson, R., & Kuhn, M. (1998). Low cost attacks on tamper resistant devices. In B. Christianson, B. Crispo, M. Lomas, & M. Roe (Eds.), Security Protocols (Security Protocols 1997, Lecture Notes in Computer Science, vol. 1361, pp. 1-13). Springer, Berlin, Heidelberg. DOI: 10.1007/BFb0028165
[14] Anciaux, N., Bonnet, P., Bouganim, L., Nguyen, B., Sandu Popa, I., & Pucheral, P. (2012). Trusted Cells: A Sea Change for Personal Data Services. IT University Technical Report Series TR2012-158. IT University of Copenhagen. ISBN 978-87-7949-263-9. ISSN 1600-6100.
[15] Sélinde van Engelenburg, Marijn Janssen, Bram Klievink. (2019). Designing context-aware systems: A method for understanding and analysing context in practice. Journal of Logical and Algebraic Methods in Programming, 103, 79-104. ISSN 2352-2208. DOI: 10.1016/j.jlamp.2018.11.003
[16] Markantonakis, K., Mayes, K., Tunstall, M., Sauveron, D., Piper, F. (2007). Smart Card Security. In: Nedjah, N., Abraham, A., Mourelle, L.d.M. (eds) Computational Intelligence in Information Assurance and Security. Studies in Computational Intelligence, vol 57. Springer, Berlin, Heidelberg. Doi: 10.1007/978-3-540-71078-3_8
[17] Damien Sauveron, Multiapplication smart card: Towards an open smart card?, Information Security Technical Report, Volume 14, Issue 2, 2009, Pages 70-78, ISSN 1363-4127, doi: 10.1016/j.istr.2009.06.007.