Some thoughts on secure chip technology

Joan Daemen

[1]STMicroelectronics, Diegem, Belgium
[2]Radboud University, Nijmegen, The Netherlands

ICMC, Arlington, May 17, 2017

Outline

Outline

A word on STMicroelectronics

- Global leader in semiconductor industry
  - approximately 43,600 employees worldwide, 8,700 in R&D
  - listed on Stock Exchange of New York, Paris and Milano
- Chips for cars, home appliances, mobile, industry, IoT ...
  - sensors, including micro-electro-mechanical systems (MEMS)
  - power switching
  - imaging
  - generic microcontroller
  - secure microcontrollers

A word on STMicroelectronics (cont'd)

- Secure microcontroller division
  - ID, transportation and banking (e.g. EMV)
  - telecom: single-wire protocol SIM
  - trusted platform module (TPM)
  - Internet of Things and Smart Grid, ...
- Diegem site in Belgium
  - end-to-end solution architecture
  - chip and HSM functional specifications
  - development of chip firmware
    - final products
    - specialized libraries
  - crypto research (on the side)

A word on STMicroelectronics (cont'd)

- Secure microcontroller division
  - ID, transportation and banking (e.g. EMV)
  - telecom: single-wire protocol SIM
  - trusted platform module (TPM)
  - Internet of Things and Smart Grid, ...
- Diegem site in Belgium
  - end-to-end solution architecture
  - chip and HSM functional specifications
  - development of chip firmware
    - final products
    - specialized libraries
  - crypto research (on the side)

A word on STMicroelectronics (cont'd)

- Secure microcontroller division
  - ID, transportation and banking (e.g. EMV)
  - telecom: single-wire protocol SIM
  - trusted platform module (TPM)
  - Internet of Things and Smart Grid, ...
- Diegem site in Belgium
  - end-to-end solution architecture
  - chip and HSM functional specifications
  - development of chip firmware
    - final products
    - specialized libraries
  - crypto research (on the side)

A word on STMicroelectronics (cont'd)

- Secure microcontroller division
  - ID, transportation and banking (e.g. EMV)
  - telecom: single-wire protocol SIM
  - trusted platform module (TPM)
  - Internet of Things and Smart Grid, ...
- Diegem site in Belgium
  - end-to-end solution architecture
  - chip and HSM functional specifications
  - development of chip firmware
    - final products
    - specialized libraries
  - crypto research (on the side)

A word on STMicroelectronics (cont'd)

- Secure microcontroller division
  - ID, transportation and banking (e.g. EMV)
  - telecom: single-wire protocol SIM
  - trusted platform module (TPM)
  - Internet of Things and Smart Grid, ...
- Diegem site in Belgium
  - end-to-end solution architecture
  - chip and HSM functional specifications
  - development of chip firmware
    - final products
    - specialized libraries
  - crypto research (on the side)

Cryptographic research in ST Belgium

- Mostly symmetric crypto
- NIST public competition for AES (FIPS 197)
  - Rijndael, by Joan Daemen (Banksys) and Vincent Rijmen (COSIC)
  - submitted together with 21 competitors in 1998
  - selected as winner by NIST on October 2, 2000
- NIST public competition for SHA-3 (FIPS 202)
  - Keccak
  - by Guido Bertoni, Joan Daemen, Michaël Peeters and Gilles Van Assche, all ST
  - submitted together with 63 competing proposals in 2008
  - selected as winner by NIST on October 2, 2012
- Competitors included RSA Labs, IBM, Microsoft, Bruce Schneier, ...
- Current focus: permutation-based crypto
  - simpler and more efficient than block-cipher based

Cryptographic research in ST Belgium

- Mostly symmetric crypto
- NIST public competition for AES (FIPS 197)
  - Rijndael, by Joan Daemen (Banksys) and Vincent Rijmen (COSIC)
  - submitted together with 21 competitors in 1998
  - selected as winner by NIST on October 2, 2000
- NIST public competition for SHA-3 (FIPS 202)
  - KECCAK
  - by Guido Bertoni, Joan Daemen, Michaël Peeters and Gilles Van Assche, all ST
  - submitted together with 63 competing proposals in 2008
  - selected as winner by NIST on October 2, 2012
- Competitors included RSA Labs, IBM, Microsoft, Bruce Schneier, …
- Current focus: permutation-based crypto
  - simpler and more efficient than block-cipher based

Cryptographic research in ST Belgium

- Mostly symmetric crypto
- NIST public competition for AES (FIPS 197)
  - Rijndael, by Joan Daemen (Banksys) and Vincent Rijmen (COSIC)
  - submitted together with 21 competitors in 1998
  - selected as winner by NIST on October 2, 2000
- NIST public competition for SHA-3 (FIPS 202)
  - KECCAK
  - by Guido Bertoni, Joan Daemen, Michaël Peeters and Gilles Van Assche, all ST
  - submitted together with 63 competing proposals in 2008
  - selected as winner by NIST on October 2, 2012
- Competitors included RSA Labs, IBM, Microsoft, Bruce Schneier, ...
- Current focus: permutation-based crypto
  - simpler and more efficient than block-cipher based

Cryptographic research in ST Belgium

- Mostly symmetric crypto
- NIST public competition for AES (FIPS 197)
  - Rijndael, by Joan Daemen (Banksys) and Vincent Rijmen (COSIC)
  - submitted together with 21 competitors in 1998
  - selected as winner by NIST on October 2, 2000
- NIST public competition for SHA-3 (FIPS 202)
  - KECCAK
  - by Guido Bertoni, Joan Daemen, Michaël Peeters and Gilles Van Assche, all ST
  - submitted together with 63 competing proposals in 2008
  - selected as winner by NIST on October 2, 2012
- Competitors included RSA Labs, IBM, Microsoft, Bruce Schneier, ...
- Current focus: permutation-based crypto
  - simpler and more efficient than block-cipher based

Cryptographic research in ST Belgium

- Mostly symmetric crypto
- NIST public competition for AES (FIPS 197)
  - Rijndael, by Joan Daemen (Banksys) and Vincent Rijmen (COSIC)
  - submitted together with 21 competitors in 1998
  - selected as winner by NIST on October 2, 2000
- NIST public competition for SHA-3 (FIPS 202)
  - KECCAK
  - by Guido Bertoni, Joan Daemen, Michaël Peeters and Gilles Van Assche, all ST
  - submitted together with 63 competing proposals in 2008
  - selected as winner by NIST on October 2, 2012
- Competitors included RSA Labs, IBM, Microsoft, Bruce Schneier, ...
- Current focus: permutation-based crypto
  - simpler and more efficient than block-cipher based

Outline

Secure chip technology AKA Smart Cards

- Dedicated technology focused on (cryptographic) security
- Started in the 80s, initially meant for payment
- Later also used for pay TV, access badges, ...
- Technology strongly improved over the years, challenged by
  - attackers for economic reasons
  - scrutiny by academic world and 3rd-party labs
- Modern secure chips are hardened against all thinkable threats:
  - side channel attacks: time, power, EM radiation,
  - fault attacks,
  - invasive attacks, ...
- Orders of magnitude harder to break than other platforms
- Sophisticated and dedicated hardware-software co-design
  - in ST: Rousset, France and Diegem, Belgium

Secure chip technology AKA Smart Cards

- Dedicated technology focused on (cryptographic) security
- Started in the 80s, initially meant for payment
- Later also used for pay TV, access badges, ...
- Technology strongly improved over the years, challenged by
  - attackers for economic reasons
  - scrutiny by academic world and 3rd-party labs
- Modern secure chips are hardened against all thinkable threats:
  - side channel attacks: time, power, EM radiation,
  - fault attacks,
  - invasive attacks, ...
- Orders of magnitude harder to break than other platforms
- Sophisticated and dedicated hardware-software co-design
  - in ST: Rousset, France and Diegem, Belgium

Secure chip technology AKA Smart Cards

- Dedicated technology focused on (cryptographic) security
- Started in the 80s, initially meant for payment
- Later also used for pay TV, access badges, ...
- Technology strongly improved over the years, challenged by
  - attackers for economic reasons
  - scrutiny by academic world and 3rd-party labs
- Modern secure chips are hardened against all thinkable threats:
  - side channel attacks: time, power, EM radiation,
  - fault attacks,
  - invasive attacks, ...
- Orders of magnitude harder to break than other platforms
- Sophisticated and dedicated hardware-software co-design
  - in ST: Rousset, France and Diegem, Belgium

Secure chip technology AKA Smart Cards

- Dedicated technology focused on (cryptographic) security
- Started in the 80s, initially meant for payment
- Later also used for pay TV, access badges, ...
- Technology strongly improved over the years, challenged by
  - attackers for economic reasons
  - scrutiny by academic world and 3rd-party labs
- Modern secure chips are hardened against all thinkable threats:
  - side channel attacks: time, power, EM radiation,
  - fault attacks,
  - invasive attacks, ...
- Orders of magnitude harder to break than other platforms
- Sophisticated and dedicated hardware-software co-design
  - in ST: Rousset, France and Diegem, Belgium

Secure chip technology AKA Smart Cards

- Dedicated technology focused on (cryptographic) security
- Started in the 80s, initially meant for payment
- Later also used for pay TV, access badges, ...
- Technology strongly improved over the years, challenged by
  - attackers for economic reasons
  - scrutiny by academic world and 3rd-party labs
- Modern secure chips are hardened against all thinkable threats:
  - side channel attacks: time, power, EM radiation,
  - fault attacks,
  - invasive attacks, ...
- Orders of magnitude harder to break than other platforms
- Sophisticated and dedicated hardware-software co-design
  - in ST: Rousset, France and Diegem, Belgium

Current trend: away from secure chips

- Most platforms have a secure chip on-board
  - SIM or secure element (SE) on smartphone
  - TPM in PC and laptop
- Still, cryptography seems to move to main CPU
  - secret keys protected by using white-box crypto
  - modules such as so-called Trusted Execution Environment (TEE)
- Smart card roll out is slow
  - e.g. payments on internet
- Why is that the case?
  - it is hard to get the keys of an application in the SE
  - business reason: stakeholders are fierce competitors
  - technical reason: systems and protocols are overly complex

Current trend: away from secure chips

- Most platforms have a secure chip on-board
  - SIM or secure element (SE) on smartphone
  - TPM in PC and laptop
- Still, cryptography seems to move to main CPU
  - secret keys protected by using white-box crypto
  - modules such as so-called Trusted Execution Environment (TEE)
- Smart card roll out is slow
  - e.g. payments on internet
- Why is that the case?
  - it is hard to get the keys of an application in the SE
  - business reason: stakeholders are fierce competitors
  - technical reason: systems and protocols are overly complex

Current trend: away from secure chips

- Most platforms have a secure chip on-board
  - SIM or secure element (SE) on smartphone
  - TPM in PC and laptop
- Still, cryptography seems to move to main CPU
  - secret keys protected by using white-box crypto
  - modules such as so-called Trusted Execution Environment (TEE)
- Smart card roll out is slow
  - e.g. payments on internet
- Why is that the case?
  - it is hard to get the keys of an application in the SE
  - business reason: stakeholders are fierce competitors
  - technical reason: systems and protocols are overly complex

Current trend: away from secure chips

- Most platforms have a secure chip on-board
  - SIM or secure element (SE) on smartphone
  - TPM in PC and laptop
- Still, cryptography seems to move to main CPU
  - secret keys protected by using white-box crypto
  - modules such as so-called Trusted Execution Environment (TEE)
- Smart card roll out is slow
  - e.g. payments on internet
- Why is that the case?
  - it is hard to get the keys of an application in the SE
  - business reason: stakeholders are fierce competitors
  - technical reason: systems and protocols are overly complex

Current trend: away from secure chips

- Most platforms have a secure chip on-board
  - SIM or secure element (SE) on smartphone
  - TPM in PC and laptop
- Still, cryptography seems to move to main CPU
  - secret keys protected by using white-box crypto
  - modules such as so-called Trusted Execution Environment (TEE)
- Smart card roll out is slow
  - e.g. payments on internet
- Why is that the case?
  - it is hard to get the keys of an application in the SE
  - business reason: stakeholders are fierce competitors
  - technical reason: systems and protocols are overly complex

Outline

Remember Security Assurance (CC Flavor)

- When we consider deploying a security product ...
- we want to know whether we will actually have security
- First: what is the security we want?
- Description of security goals: Security Target
  - clear and unambiguous description
  - must clearly specify the attacker model
  - often scope is limited
- Product that implements functionality: Target of Evaluation (TOE)

Remember Security Assurance (CC Flavor)

- When we consider deploying a security product ...
- we want to know whether we will actually have security
- First: what is the security we want?
- Description of security goals: Security Target
  - clear and unambiguous description
  - must clearly specify the attacker model
  - often scope is limited
- Product that implements functionality: Target of Evaluation (TOE)

Remember Security Assurance (CC Flavor)

- When we consider deploying a security product ...
- we want to know whether we will actually have security
- First: what is the security we want?
- Description of security goals: Security Target
  - clear and unambiguous description
  - must clearly specify the attacker model
  - often scope is limited
- Product that implements functionality: Target of Evaluation (TOE)

Verification of the target

- Verify whether target functionally implements requirements:
  - requires documentation at multiple abstraction levels
  - architecture documents
  - documented code
  - traceability across all levels
- Verify whether TOE resists attacks
  - evaluation by independent third party with expertise
  - white-box: access to all documentation and sources
- Tracing production and engineering process
  - to detect e.g., Trojans or backdoors
- Tough and expensive job!

Verification of the target

- Verify whether target functionally implements requirements:
  - requires documentation at multiple abstraction levels
  - architecture documents
  - documented code
  - traceability across all levels
- Verify whether TOE resists attacks
  - evaluation by independent third party with expertise
  - white-box: access to all documentation and sources
- Tracing production and engineering process
  - to detect e.g., Trojans or backdoors
- Tough and expensive job!

Verification of the target

- Verify whether target functionally implements requirements:
  - requires documentation at multiple abstraction levels
  - architecture documents
  - documented code
  - traceability across all levels
- Verify whether TOE resists attacks
  - evaluation by independent third party with expertise
  - white-box: access to all documentation and sources
- Tracing production and engineering process
  - to detect e.g., Trojans or backdoors
- Tough and expensive job!

Verification of the target

- Verify whether target functionally implements requirements:
  - requires documentation at multiple abstraction levels
  - architecture documents
  - documented code
  - traceability across all levels
- Verify whether TOE resists attacks
  - evaluation by independent third party with expertise
  - white-box: access to all documentation and sources
- Tracing production and engineering process
  - to detect e.g., Trojans or backdoors
- Tough and expensive job!

How to obtain good security assurance?

- Appropriateness of Security Target
  - does it address the real concerns?
- Complexity and quality: the simpler the better
  - Security Target shall be simple
  - TOE shall have simple architecture and interface
  - modularity helps
- Quantity: the smaller the better
  - amount of documentation
  - # levels of abstraction
  - # functions
  - # lines of code

How to obtain good security assurance?

- Appropriateness of Security Target
  - does it address the real concerns?
- Complexity and quality: the simpler the better
  - Security Target shall be simple
  - TOE shall have simple architecture and interface
  - modularity helps
- Quantity: the smaller the better
  - amount of documentation
  - # levels of abstraction
  - # functions
  - # lines of code

How to obtain good security assurance?

- Appropriateness of Security Target
  - does it address the real concerns?
- Complexity and quality: the simpler the better
  - Security Target shall be simple
  - TOE shall have simple architecture and interface
  - modularity helps
- Quantity: the smaller the better
  - amount of documentation
  - # levels of abstraction
  - # functions
  - # lines of code

How to obtain good security assurance? (cont’d)

- Volatility:
  - the more stable the better
  - firmware/software update is a liability
- Good understanding of attack surface
  - physical: side channel, faults, ...
  - logical: API attacks
  - industrialization: key management and handling
  - ...
- actual security assurance and CC EAL are different things

How to obtain good security assurance? (cont'd)

- Volatility:
  - the more stable the better
  - firmware/software update is a liability
- Good understanding of attack surface
  - physical: side channel, faults, ...
  - logical: API attacks
  - industrialization: key management and handling
  - ...
- actual security assurance and CC EAL are different things

How to obtain good security assurance? (cont'd)

- Volatility:
  - the more stable the better
  - firmware/software update is a liability
- Good understanding of attack surface
  - physical: side channel, faults, ...
  - logical: API attacks
  - industrialization: key management and handling
  - ...
- actual security assurance and CC EAL are different things

Factors that inhibit security assurance

- Bad specifications and standards
  - mixing up requirements and mechanisms
  - long and/or complex documents
  - absence of finite state machines
  - specification of one side of the protocol only
  - frequent updates and *enhancements*
- Platforms with rich functionality
  - complex processor architecture
  - undocumented features, e.g. for updating firmware
- Software with rich functionality
  - regular updates
  - heterogeneous

Factors that inhibit security assurance

- Bad specifications and standards
  - mixing up requirements and mechanisms
  - long and/or complex documents
  - absence of finite state machines
  - specification of one side of the protocol only
  - frequent updates and *enhancements*
- Platforms with rich functionality
  - complex processor architecture
  - undocumented features, e.g. for updating firmware
- Software with rich functionality
  - regular updates
  - heterogeneous

Factors that inhibit security assurance

- Bad specifications and standards
  - mixing up requirements and mechanisms
  - long and/or complex documents
  - absence of finite state machines
  - specification of one side of the protocol only
  - frequent updates and *enhancements*
- Platforms with rich functionality
  - complex processor architecture
  - undocumented features, e.g. for updating firmware
- Software with rich functionality
  - regular updates
  - heterogeneous

Security assurance of secure chips

- Product with security assurance as core business
- Architecture:
  - limited functionality
  - simple interface
- Design and development:
  - hardware: CPU, memory, crypto accelerators
  - software: more than just functionally correct

Security assurance of secure chips

- Product with security assurance as core business
- Architecture:
  - limited functionality
  - simple interface
- Design and development:
  - hardware: CPU, memory, crypto accelerators
  - software: more than just functionally correct

Security assurance of secure chips

- Product with security assurance as core business
- Architecture:
  - limited functionality
  - simple interface
- Design and development:
  - hardware: CPU, memory, crypto accelerators
  - software: more than just functionally correct

Outline

Focus on the key availability problem

- Implicit architecture
  - applicative functionality: on the main CPU
    - Ticketing, access to services, display and keyboard
  - cryptographic functionality: in a secure element
    - encryption and/or authentication
    - electronic signature
    - key establishment
    - transaction counters and key ratification
    - possibly data management: electronic value, logs, ...
- The problem we address:
  - getting secret keys from application provider to SE

Focus on the key availability problem

- Implicit architecture
  - applicative functionality: on the main CPU
    - Ticketing, access to services, display and keyboard
  - cryptographic functionality: in a secure element
    - encryption and/or authentication
    - electronic signature
    - key establishment
    - transaction counters and key ratification
    - possibly data management: electronic value, logs, ...
- The problem we address:
  - getting secret keys from application provider to SE

Focus on the key availability problem

- Implicit architecture
  - applicative functionality: on the main CPU
    - Ticketing, access to services, display and keyboard
  - cryptographic functionality: in a secure element
    - encryption and/or authentication
    - electronic signature
    - key establishment
    - transaction counters and key ratification
    - possibly data management: electronic value, logs, ...
- The problem we address:
  - getting secret keys from application provider to SE

Focus on the key availability problem

- Implicit architecture
  - applicative functionality: on the main CPU
    - Ticketing, access to services, display and keyboard
  - cryptographic functionality: in a secure element
    - encryption and/or authentication
    - electronic signature
    - key establishment
    - transaction counters and key ratification
    - possibly data management: electronic value, logs, ...
- The problem we address:
  - getting secret keys from application provider to SE

Focus on the key availability problem

- Implicit architecture
  - applicative functionality: on the main CPU
    - Ticketing, access to services, display and keyboard
  - cryptographic functionality: in a secure element
    - encryption and/or authentication
    - electronic signature
    - key establishment
    - transaction counters and key ratification
    - possibly data management: electronic value, logs, ...
- The problem we address:
  - getting secret keys from application provider to SE

Focus on the key availability problem

- Implicit architecture
  - applicative functionality: on the main CPU
    - Ticketing, access to services, display and keyboard
  - cryptographic functionality: in a secure element
    - encryption and/or authentication
    - electronic signature
    - key establishment
    - transaction counters and key ratification
    - possibly data management: electronic value, logs, ...
- The problem we address:
  - getting secret keys from application provider to SE

Focus on the key availability problem

- Implicit architecture
  - applicative functionality: on the main CPU
    - Ticketing, access to services, display and keyboard
  - cryptographic functionality: in a secure element
    - encryption and/or authentication
    - electronic signature
    - key establishment
    - transaction counters and key ratification
    - possibly data management: electronic value, logs, ...
- The problem we address:
  - getting secret keys from application provider to SE

A hypothetical scenario

- Bob will be traveling to Paris and will
  - use the Metro there
  - visit some museums, ...
- Paris Metro and museums support smartphone app for access
- Bob downloads the app on his phone, including tickets
  - app comes from some app provider
- The secret keys for the smartphone app end up in the SE
  - keys owned by PariMetro Co.
  - SE in smartphone controlled by ...Phone Co.
- Challenge: getting keys from PariMetro Co. HSM to SE

A hypothetical scenario

- Bob will be traveling to Paris and will
  - use the Metro there
  - visit some museums, ...
- Paris Metro and museums support smartphone app for access
- Bob downloads the app on his phone, including tickets
  - app comes from some app provider
- The secret keys for the smartphone app end up in the SE
  - keys owned by PariMetro Co.
  - SE in smartphone controlled by ...Phone Co.
- Challenge: getting keys from PariMetro Co. HSM to SE

A hypothetical scenario

- Bob will be traveling to Paris and will
  - use the Metro there
  - visit some museums, ...
- Paris Metro and museums support smartphone app for access
- Bob downloads the app on his phone, including tickets
  - app comes from some app provider
- The secret keys for the smartphone app end up in the SE
  - keys owned by PariMetro Co.
  - SE in smartphone controlled by ...Phone Co.
- Challenge: getting keys from PariMetro Co. HSM to SE

A hypothetical scenario

- Bob will be traveling to Paris and will
  - use the Metro there
  - visit some museums, ...
- Paris Metro and museums support smartphone app for access
- Bob downloads the app on his phone, including tickets
  - app comes from some app provider
- The secret keys for the smartphone app end up in the SE
  - keys owned by PariMetro Co.
  - SE in smartphone controlled by ...Phone Co.
- Challenge: getting keys from PariMetro Co. HSM to SE

Architecture in the SE: GP/Javacard

- Different applications called *agents*
  - each of a given type with specific functionality
  - e.g. EMV payment, Mifare, ...
  - with its own keys and data
  - platform controls interaction between agents
  - interface with commands/responses (historically ISO 7816)
- Each agent has an owner
  - there can be agents of multiple owners on the same SE
  - each owner has agent on SE to manage his agents: *security domain* (SD)
  - cryptographic *secure channel* between owner HSM and SD
  - SD and central HSM share unique secret key for that purpose
- transfer of keys from HSM to agent on SE with secure channel!

Architecture in the SE: GP/Javacard

- Different applications called *agents*
  - each of a given type with specific functionality
  - e.g. EMV payment, Mifare, ...
  - with its own keys and data
  - platform controls interaction between agents
  - interface with commands/responses (historically ISO 7816)
- Each agent has an owner
  - there can be agents of multiple owners on the same SE
  - each owner has agent on SE to manage his agents: *security domain* (SD)
  - cryptographic *secure channel* between owner HSM and SD
  - SD and central HSM share unique secret key for that purpose
- transfer of keys from HSM to agent on SE with secure channel!

Architecture in the SE: GP/Javacard

- Different applications called *agents*
  - each of a given type with specific functionality
  - e.g. EMV payment, Mifare, ...
  - with its own keys and data
  - platform controls interaction between agents
  - interface with commands/responses (historically ISO 7816)
- Each agent has an owner
  - there can be agents of multiple owners on the same SE
  - each owner has agent on SE to manage his agents: *security domain* (SD)
  - cryptographic *secure channel* between owner HSM and SD
  - SD and central HSM share unique secret key for that purpose
- transfer of keys from HSM to agent on SE with secure channel!

Architecture in the SE: GP/Javacard

- Different applications called *agents*
  - each of a given type with specific functionality
  - e.g. EMV payment, Mifare, ...
  - with its own keys and data
  - platform controls interaction between agents
  - interface with commands/responses (historically ISO 7816)
- Each agent has an owner
  - there can be agents of multiple owners on the same SE
  - each owner has agent on SE to manage his agents: *security domain* (SD)
  - cryptographic *secure channel* between owner HSM and SD
  - SD and central HSM share unique secret key for that purpose
- transfer of keys from HSM to agent on SE with secure channel!

Architecture in the SE: GP/Javacard

- Different applications called *agents*
  - each of a given type with specific functionality
  - e.g. EMV payment, Mifare, ...
  - with its own keys and data
  - platform controls interaction between agents
  - interface with commands/responses (historically ISO 7816)
- Each agent has an owner
  - there can be agents of multiple owners on the same SE
  - each owner has agent on SE to manage his agents: *security domain* (SD)
  - cryptographic *secure channel* between owner HSM and SD
  - SD and central HSM share unique secret key for that purpose
- transfer of keys from HSM to agent on SE with secure channel!

Architecture in the SE: GP/Javacard

- Different applications called *agents*
  - each of a given type with specific functionality
  - e.g. EMV payment, Mifare, ...
  - with its own keys and data
  - platform controls interaction between agents
  - interface with commands/responses (historically ISO 7816)
- Each agent has an owner
  - there can be agents of multiple owners on the same SE
  - each owner has agent on SE to manage his agents: *security domain* (SD)
  - cryptographic *secure channel* between owner HSM and SD
  - SD and central HSM share unique secret key for that purpose
- transfer of keys from HSM to agent on SE with secure channel!

Architecture in the SE: GP/Javacard

- Different applications called *agents*
  - each of a given type with specific functionality
  - e.g. EMV payment, Mifare, ...
  - with its own keys and data
  - platform controls interaction between agents
  - interface with commands/responses (historically ISO 7816)
- Each agent has an owner
  - there can be agents of multiple owners on the same SE
  - each owner has agent on SE to manage his agents: *security domain* (SD)
  - cryptographic *secure channel* between owner HSM and SD
  - SD and central HSM share unique secret key for that purpose
- transfer of keys from HSM to agent on SE with secure channel!

Architecture in the SE: GP/Javacard

- Different applications called *agents*
  - each of a given type with specific functionality
  - e.g. EMV payment, Mifare, ...
  - with its own keys and data
  - platform controls interaction between agents
  - interface with commands/responses (historically ISO 7816)
- Each agent has an owner
  - there can be agents of multiple owners on the same SE
  - each owner has agent on SE to manage his agents: *security domain* (SD)
  - cryptographic *secure channel* between owner HSM and SD
  - SD and central HSM share unique secret key for that purpose
- transfer of keys from HSM to agent on SE with secure channel!

Architecture in the SE: GP/Javacard

- Different applications called *agents*
  - each of a given type with specific functionality
  - e.g. EMV payment, Mifare, ...
  - with its own keys and data
  - platform controls interaction between agents
  - interface with commands/responses (historically ISO 7816)
- Each agent has an owner
  - there can be agents of multiple owners on the same SE
  - each owner has agent on SE to manage his agents: *security domain* (SD)
  - cryptographic *secure channel* between owner HSM and SD
  - SD and central HSM share unique secret key for that purpose
- transfer of keys from HSM to agent on SE with secure channel!

Outline of protocol (key transfer part)

- we assume the app provider has an SD on the secure element
- application keys travel from PariMetro HSM to SE in two hops
  - from PariMetro HSM to app provider HSM
  - from app provider HSM to SE
- during transport, keys are in a secure *key envelope*
  - a key consists of a value, a unique identifier and attributes
  - all wrapped: enciphered and authenticated
  - key to perform the wrapping is called a KEK

Outline of protocol (key transfer part)

- we assume the app provider has an SD on the secure element
- application keys travel from PariMetro HSM to SE in two hops
  - from PariMetro HSM to app provider HSM
  - from app provider HSM to SE
- during transport, keys are in a secure *key envelope*
  - a key consists of a value, a unique identifier and attributes
  - all wrapped: enciphered and authenticated
  - key to perform the wrapping is called a KEK

Outline of protocol (key transfer part)

- we assume the app provider has an SD on the secure element
- application keys travel from PariMetro HSM to SE in two hops
  - from PariMetro HSM to app provider HSM
  - from app provider HSM to SE
- during transport, keys are in a secure *key envelope*
  - a key consists of a value, a unique identifier and attributes
  - all wrapped: enciphered and authenticated
  - key to perform the wrapping is called a KEK

Outline of protocol (key transfer part)

- we assume the app provider has an SD on the secure element
- application keys travel from PariMetro HSM to SE in two hops
  - from PariMetro HSM to app provider HSM
  - from app provider HSM to SE
- during transport, keys are in a secure *key envelope*
  - a key consists of a value, a unique identifier and attributes
  - all wrapped: enciphered and authenticated
  - key to perform the wrapping is called a KEK

Outline of protocol (key transfer part)

- we assume the app provider has an SD on the secure element
- application keys travel from PariMetro HSM to SE in two hops
  - from PariMetro HSM to app provider HSM
  - from app provider HSM to SE
- during transport, keys are in a secure *key envelope*
  - a key consists of a value, a unique identifier and attributes
  - all wrapped: enciphered and authenticated
  - key to perform the wrapping is called a KEK

Outline of protocol (cont'd)

- Hop from ParìMetro HSM to app provider HSM
  - requires shared KEK between them
  - master KEK can be established with a PKI
  - certificates imply having passed an audit
  - unique KEK per key envelope can be derived from master KEK
- app provider HSM
  - unwraps key envelope to cleartext
  - wraps cleartext to key envelope meant for SE
- SD in SE unwraps key envelope and passes key to agent
  - which agent is determined by key identifier

Outline of protocol (cont'd)

- Hop from ParìMetro HSM to app provider HSM
  - requires shared KEK between them
  - master KEK can be established with a PKI
  - certificates imply having passed an audit
  - unique KEK per key envelope can be derived from master KEK
- app provider HSM
  - unwraps key envelope to cleartext
  - wraps cleartext to key envelope meant for SE
- SD in SE unwraps key envelope and passes key to agent
  - which agent is determined by key identifier

Outline of protocol (cont'd)

- Hop from ParіMetro HSM to app provider HSM
  - requires shared KEK between them
  - master KEK can be established with a PKI
  - certificates imply having passed an audit
  - unique KEK per key envelope can be derived from master KEK
- app provider HSM
  - unwraps key envelope to cleartext
  - wraps cleartext to key envelope meant for SE
- SD in SE unwraps key envelope and passes key to agent
  - which agent is determined by key identifier

Features of the protocol

- Main feature: never cleartext keys outside HSM or SE
- Scope for security assurance:
  - module in HSM for key envelope generation
  - module in HSM for key envelope unwrap-wrap
  - module in SE for key envelope unwrap
- Same protocol to transfer app provider SD key to SE
  - from app provider HSM to Phone Co. HSM
  - from Phone Co. HSM to SE

Features of the protocol

- Main feature: never cleartext keys outside HSM or SE
- Scope for security assurance:
  - module in HSM for key envelope generation
  - module in HSM for key envelope unwrap-wrap
  - module in SE for key envelope unwrap
- Same protocol to transfer app provider SD key to SE
  - from app provider HSM to Phone Co. HSM
  - from Phone Co. HSM to SE

Features of the protocol

- Main feature: never cleartext keys outside HSM or SE
- Scope for security assurance:
  - module in HSM for key envelope generation
  - module in HSM for key envelope unwrap-wrap
  - module in SE for key envelope unwrap
- Same protocol to transfer app provider SD key to SE
  - from app provider HSM to Phone Co. HSM
  - from Phone Co. HSM to SE

Standard key envelope

- for HSM-HSM and HSM-SE interoperability
  - encoding of payload:
    - key values
    - key identifiers, including owner ID
    - key attributes (limited)
  - identification of KEK
    - hierarchy: master, base, session
    - identifiers
    - derivation function
  - key wrap algorithm: preferably a single one
- Preferably same for HSM-HSM and HSM-SE
- HSM-SE key transfer partially specified in GP but outdated
- This can be done in a very simple specification

Standard key envelope

- for HSM-HSM and HSM-SE interoperability
  - encoding of payload:
    - key values
    - key identifiers, including owner ID
    - key attributes (limited)
  - identification of KEK
    - hierarchy: master, base, session
    - identifiers
    - derivation function
  - key wrap algorithm: preferably a single one
- Preferably same for HSM-HSM and HSM-SE
- HSM-SE key transfer partially specified in GP but outdated
- This can be done in a very simple specification

Standard key envelope

- for HSM-HSM and HSM-SE interoperability
  - encoding of payload:
    - key values
    - key identifiers, including owner ID
    - key attributes (limited)
  - identification of KEK
    - hierarchy: master, base, session
    - identifiers
    - derivation function
  - key wrap algorithm: preferably a single one
- Preferably same for HSM-HSM and HSM-SE
- HSM-SE key transfer partially specified in GP but outdated
- This can be done in a very simple specification

Standard key envelope

- for HSM-HSM and HSM-SE interoperability
  - encoding of payload:
    - key values
    - key identifiers, including owner ID
    - key attributes (limited)
  - identification of KEK
    - hierarchy: master, base, session
    - identifiers
    - derivation function
  - key wrap algorithm: preferably a single one
- Preferably same for HSM-HSM and HSM-SE
- HSM-SE key transfer partially specified in GP but outdated
- This can be done in a very simple specification

Standard key envelope

- for HSM-HSM and HSM-SE interoperability
  - encoding of payload:
    - key values
    - key identifiers, including owner ID
    - key attributes (limited)
  - identification of KEK
    - hierarchy: master, base, session
    - identifiers
    - derivation function
  - key wrap algorithm: preferably a single one
- Preferably same for HSM-HSM and HSM-SE
- HSM-SE key transfer partially specified in GP but outdated
- This can be done in a very simple specification

Standard key envelope

- for HSM-HSM and HSM-SE interoperability
  - encoding of payload:
    - key values
    - key identifiers, including owner ID
    - key attributes (limited)
  - identification of KEK
    - hierarchy: master, base, session
    - identifiers
    - derivation function
  - key wrap algorithm: preferably a single one
- Preferably same for HSM-HSM and HSM-SE
- HSM-SE key transfer partially specified in GP but outdated
- This can be done in a very simple specification

Standard key envelope

- for HSM-HSM and HSM-SE interoperability
  - encoding of payload:
    - key values
    - key identifiers, including owner ID
    - key attributes (limited)
  - identification of KEK
    - hierarchy: master, base, session
    - identifiers
    - derivation function
  - key wrap algorithm: preferably a single one
- Preferably same for HSM-HSM and HSM-SE
- HSM-SE key transfer partially specified in GP but outdated
- This can be done in a very simple specification

Outline

Conclusions

- Tension between
  - Technology evolves constantly with many innovations
  - Security assurance requires clarity and stability
- Principles of sound key distribution do not evolve
  - technology does: secure chips getting better and better
- If there is trust, secure key transport to SE is feasible
- A standard key envelope can help in this

Thanks for your attention!

Conclusions

- Tension between
  - Technology evolves constantly with many innovations
  - Security assurance requires clarity and stability
- Principles of sound key distribution do not evolve
  - technology does: secure chips getting better and better
- If there is trust, secure key transport to SE is feasible
- A standard key envelope can help in this

Thanks for your attention!

Conclusions

- Tension between
  - Technology evolves constantly with many innovations
  - Security assurance requires clarity and stability
- Principles of sound key distribution do not evolve
  - technology does: secure chips getting better and better
- If there is trust, secure key transport to SE is feasible
- A standard key envelope can help in this

Thanks for your attention!