Network Security - Week 2

Manuel E. Correia

DCC/FCUP

2025

Cryptography

The art and science of establishing secure communication over an insecure channel

Cryptography

The art and science of establishing secure communication over an insecure channel

Confidentiality

- Protect sensitive data from eavesdropping - Encryption

Cryptography

The art and science of establishing secure communication over an insecure channel

Confidentiality

- Protect sensitive data from eavesdropping - Encryption
- Requires a key to encrypt/decrypt

Cryptography

The art and science of establishing secure communication over an insecure channel

Confidentiality

- Protect sensitive data from eavesdropping - Encryption
- Requires a key to encrypt/decrypt
- The keys can be the same – symmetric cryptography
- or different – public-key cryptography

Cryptography

The art and science of establishing secure communication over an insecure channel

Confidentiality

- Protect sensitive data from eavesdropping - Encryption
- Requires a key to encrypt/decrypt
- The keys can be the same – symmetric cryptography
- or different – public-key cryptography

Integrity

- The goal can also be to detect if messages are altered

Cryptography

The art and science of establishing secure communication over an insecure channel

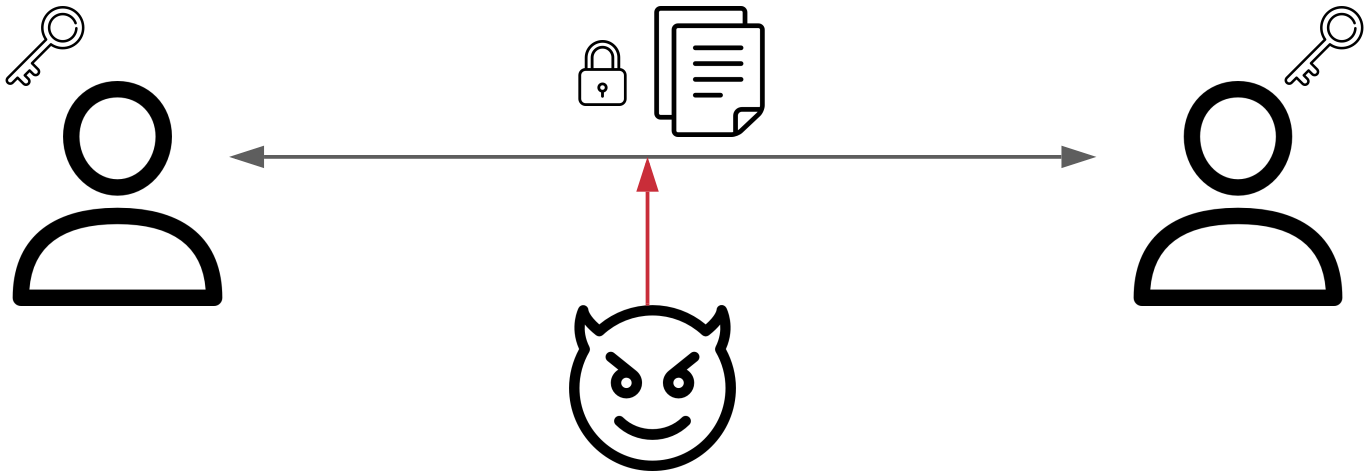
Confidentiality

- Protect sensitive data from eavesdropping - Encryption
- Requires a key to encrypt/decrypt
- The keys can be the same – symmetric cryptography
- or different – public-key cryptography

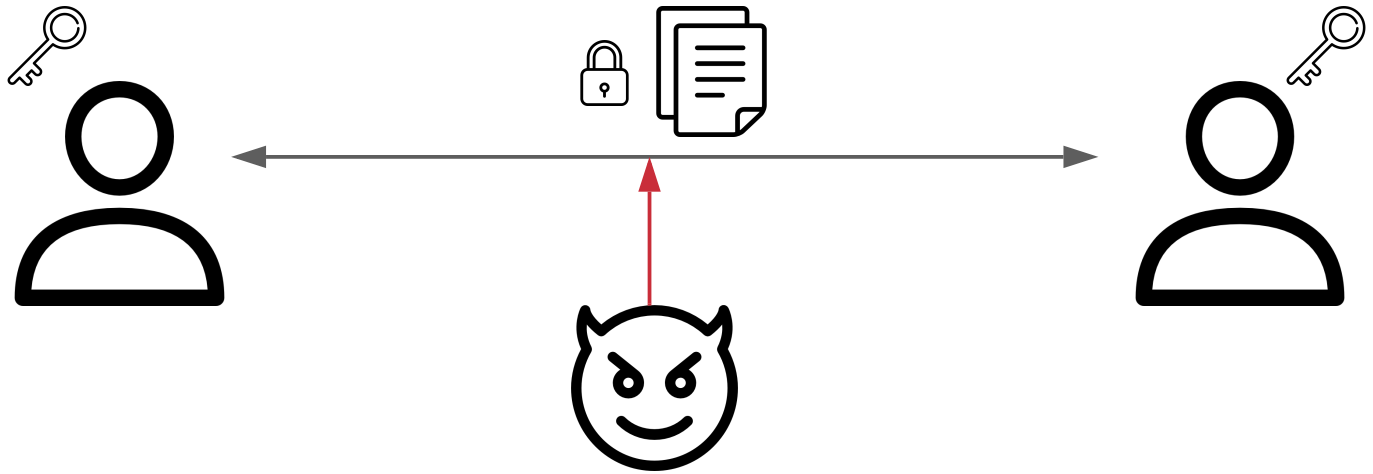
Integrity

- The goal can also be to detect if messages are altered
- For symmetric crypto, we use Message Authenticated Codes
- For public-key crypto, we use Digital Signatures

Symmetric Cryptography - P1

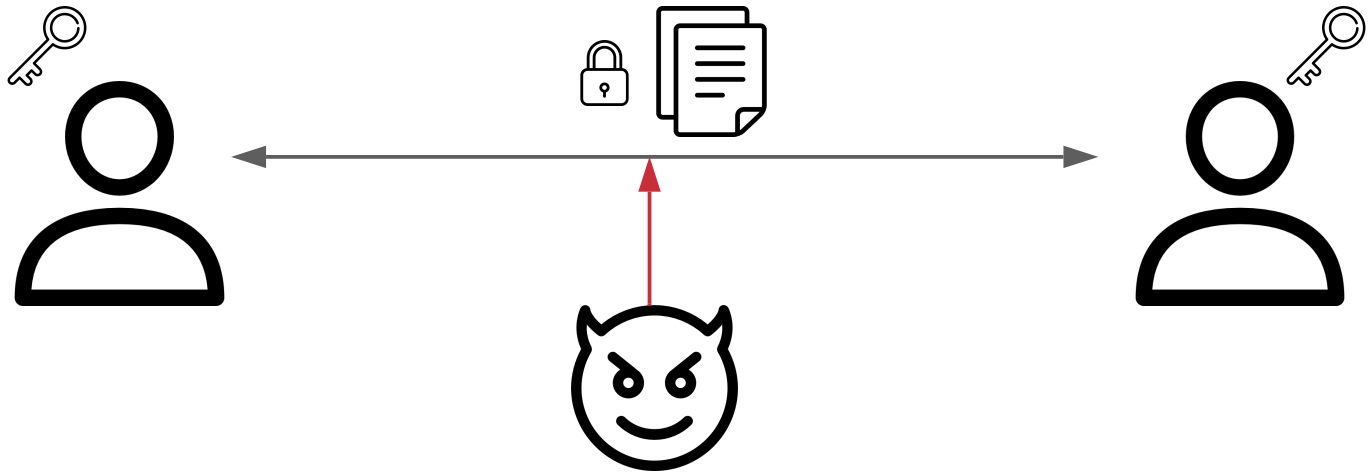


Symmetric Cryptography - P1



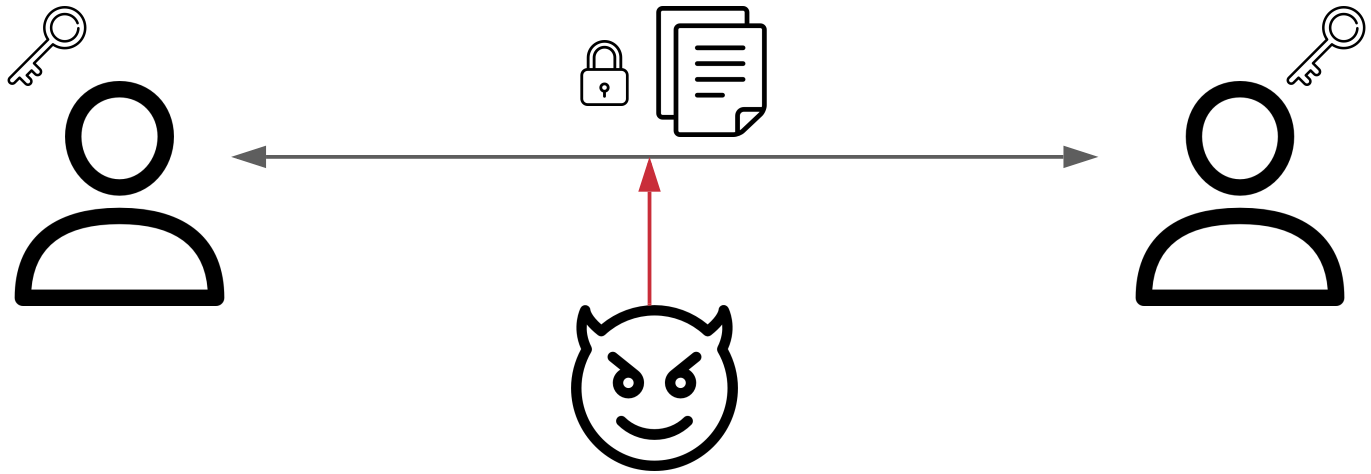
- Users know the same key (pre-shared)

Symmetric Cryptography - P1



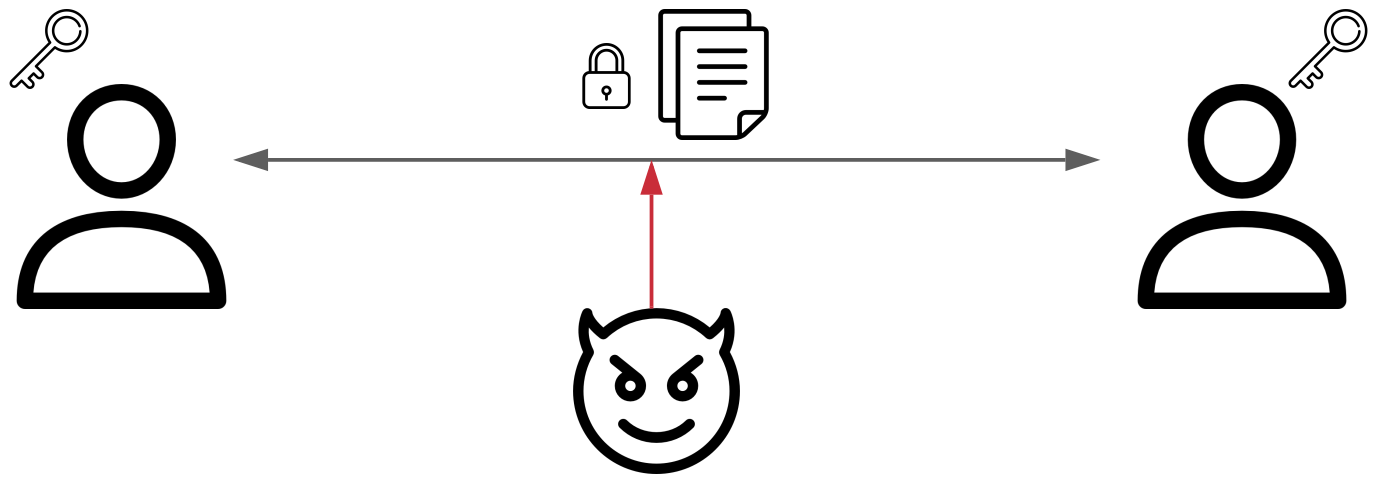
- Users know the same key (pre-shared)
- Encrypt messages passed in the channel

Symmetric Cryptography - P1



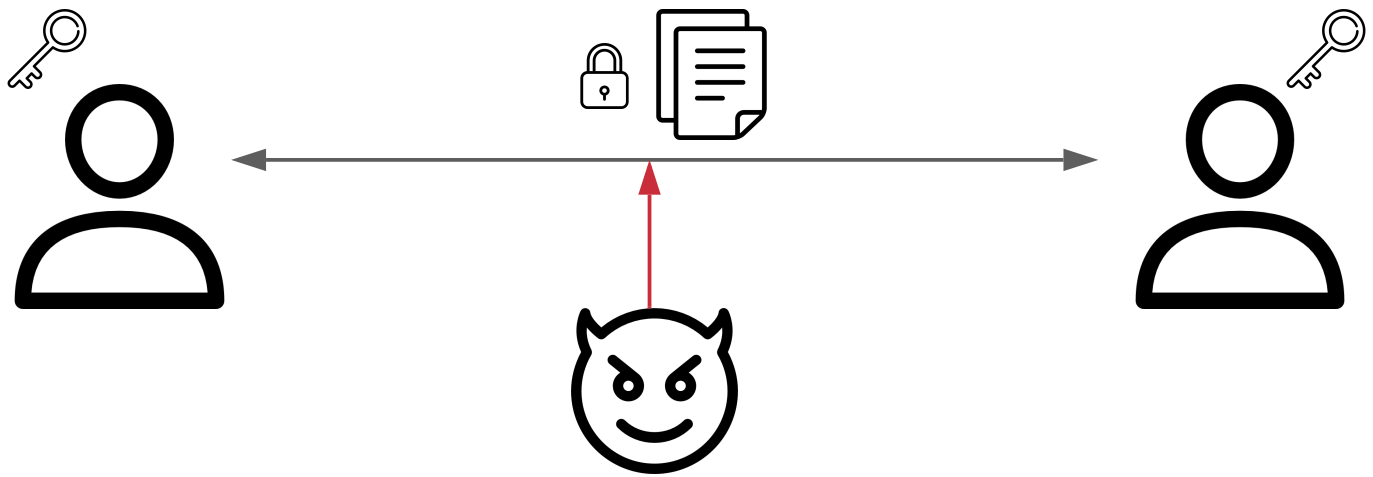
- Users know the same key (pre-shared)
- Encrypt messages passed in the channel
  - Protect message m, using key k to produce ciphertext c

Symmetric Cryptography - P1



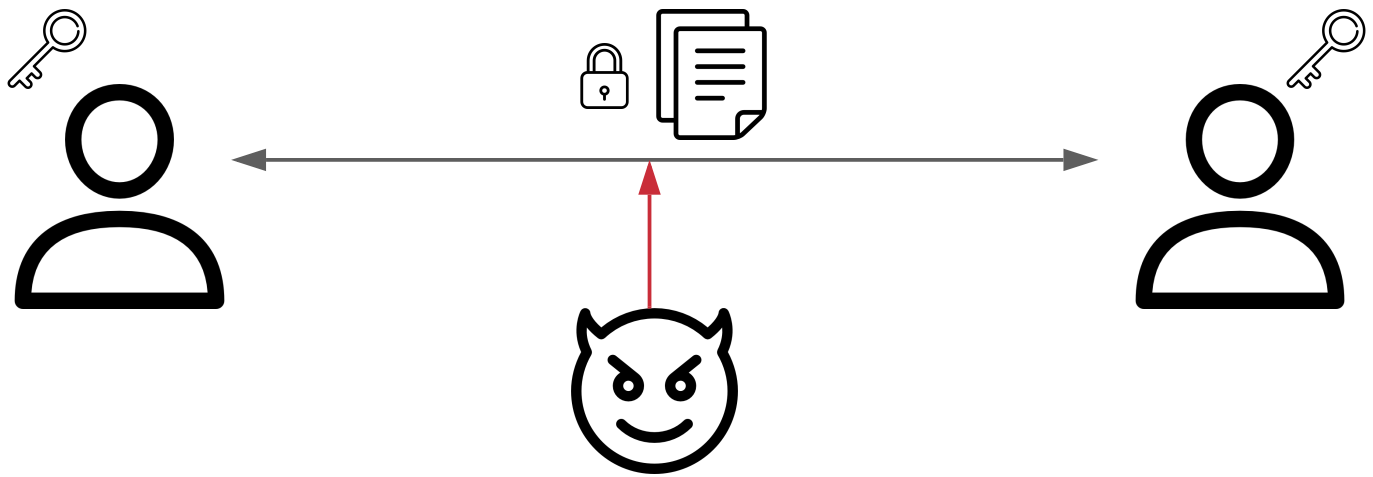
- Users know the same key (pre-shared)
- Encrypt messages passed in the channel
  - Protect message m, using key k to produce ciphertext c
  - Encryption: $c = \text{Encrypt}(k, m)$

Symmetric Cryptography - P1



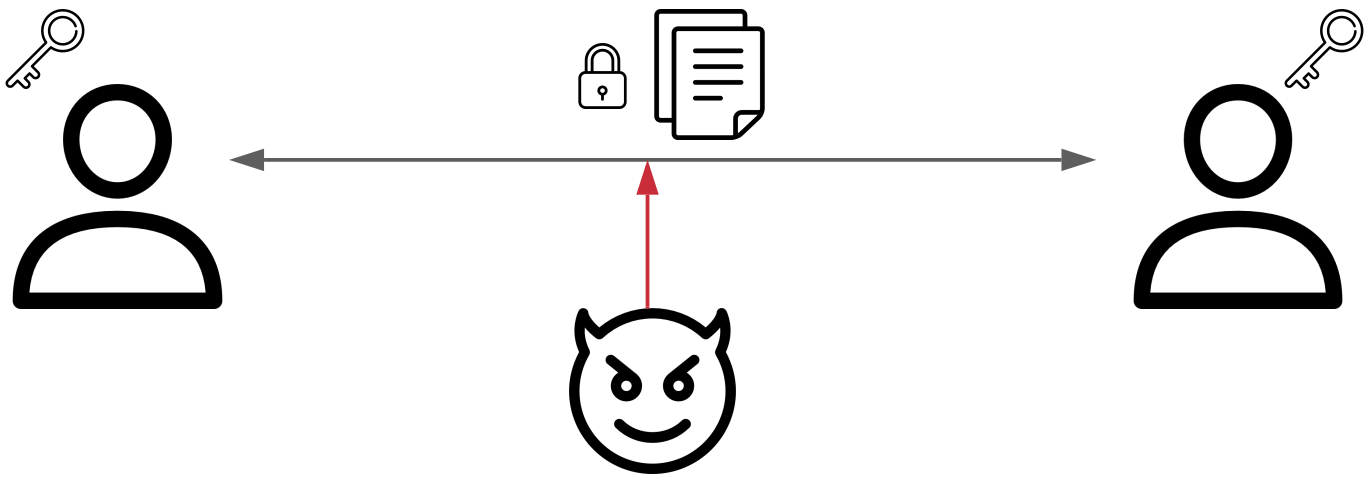
- Users know the same key (pre-shared)
- Encrypt messages passed in the channel
  - Protect message m, using key k to produce ciphertext c
  - Encryption: $c = \text{Encrypt}(k, m)$
  - Decryption: $m = \text{Decrypt}(k, c)$

Symmetric Cryptography - P1



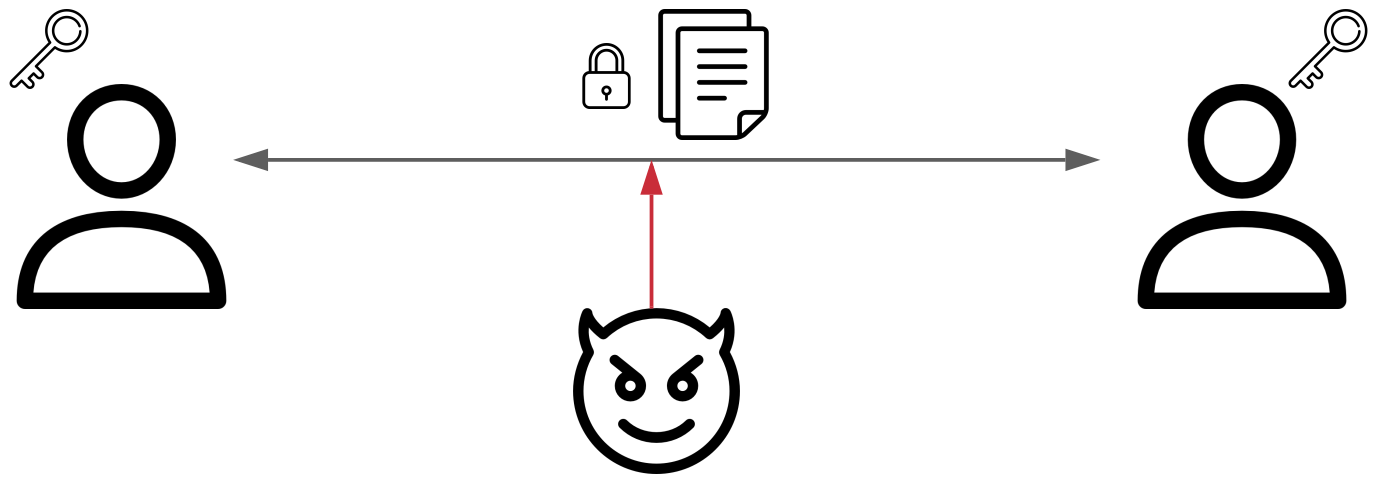
- Users know the same key (pre-shared)
- Encrypt messages passed in the channel
  - Protect message m, using key k to produce ciphertext c
  - Encryption: $c = \text{Encrypt}(k, m)$
  - Decryption: $m = \text{Decrypt}(k, c)$
- Protect the integrity of messages in the channel

Symmetric Cryptography - P1



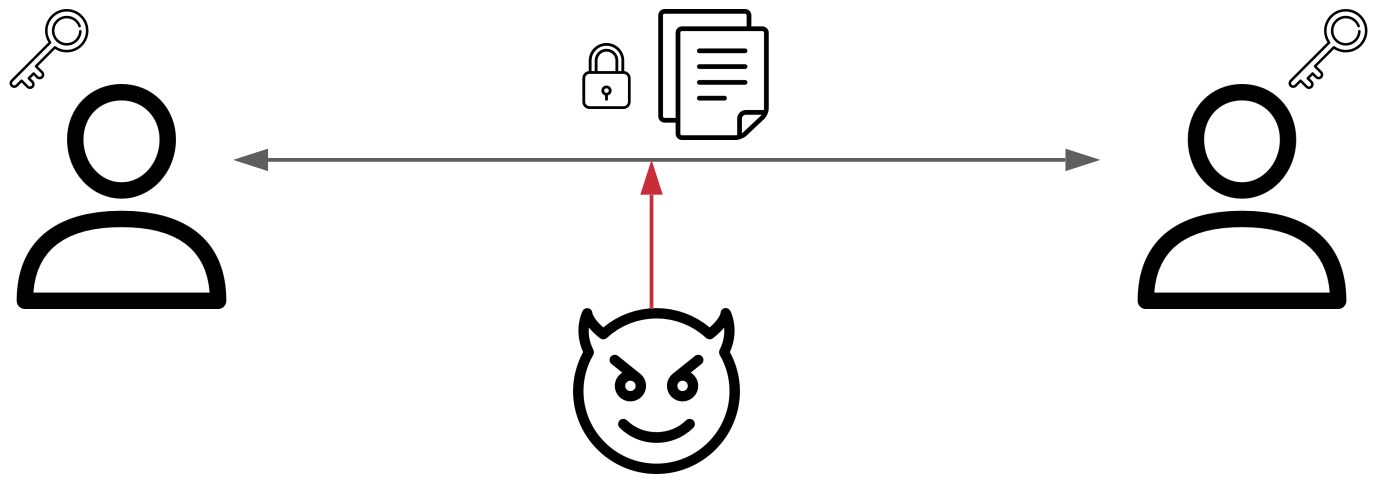
- Users know the same key (pre-shared)
- Encrypt messages passed in the channel
  - Protect message m, using key k to produce ciphertext c
  - Encryption: $c = \text{Encrypt}(k, m)$
  - Decryption: $m = \text{Decrypt}(k, c)$
- Protect the integrity of messages in the channel
  - Protect message m, using key k to produce MAC t

Symmetric Cryptography - P1



- Users know the same key (pre-shared)
- Encrypt messages passed in the channel
  - Protect message m, using key k to produce ciphertext c
  - Encryption: $c = \text{Encrypt}(k, m)$
  - Decryption: $m = \text{Decrypt}(k, c)$
- Protect the integrity of messages in the channel
  - Protect message m, using key k to produce MAC t
  - Authentication: $t = \text{MAC}(k, m)$

Symmetric Cryptography - P1



- Users know the same key (pre-shared)
- Encrypt messages passed in the channel
  - Protect message m, using key k to produce ciphertext c
  - Encryption: $c = \text{Encrypt}(k, m)$
  - Decryption: $m = \text{Decrypt}(k, c)$
- Protect the integrity of messages in the channel
  - Protect message m, using key k to produce MAC t
  - Authentication: $t = \text{MAC}(k, m)$
  - Verification: $T/F = \text{Verify}(k, m, t)$

Symmetric Cryptography - P2

Primitives

- Symmetric encryption

Symmetric Cryptography - P2

Primitives

- Symmetric encryption
  - Confidentiality

Symmetric Cryptography - P2

Primitives

- Symmetric encryption
  - Confidentiality
  - AES-CBC, AES-CTR, RC4

Symmetric Cryptography - P2

Primitives

- Symmetric encryption
  - Confidentiality
  - AES-CBC, AES-CTR, RC4
- Message authentication codes

Symmetric Cryptography - P2

Primitives

- Symmetric encryption
  - Confidentiality
  - AES-CBC, AES-CTR, RC4
- Message authentication codes
  - Integrity

Symmetric Cryptography - P2

Primitives

- Symmetric encryption
  - Confidentiality
  - AES-CBC, AES-CTR, RC4
- Message authentication codes
  - Integrity
  - HMAC, CMAC

Symmetric Cryptography - P2

Primitives

- Symmetric encryption
  - Confidentiality
  - AES-CBC, AES-CTR, RC4
- Message authentication codes
  - Integrity
  - HMAC, CMAC
- Authenticated Encryption with Associated Data (AEAD)

Symmetric Cryptography - P2

Primitives

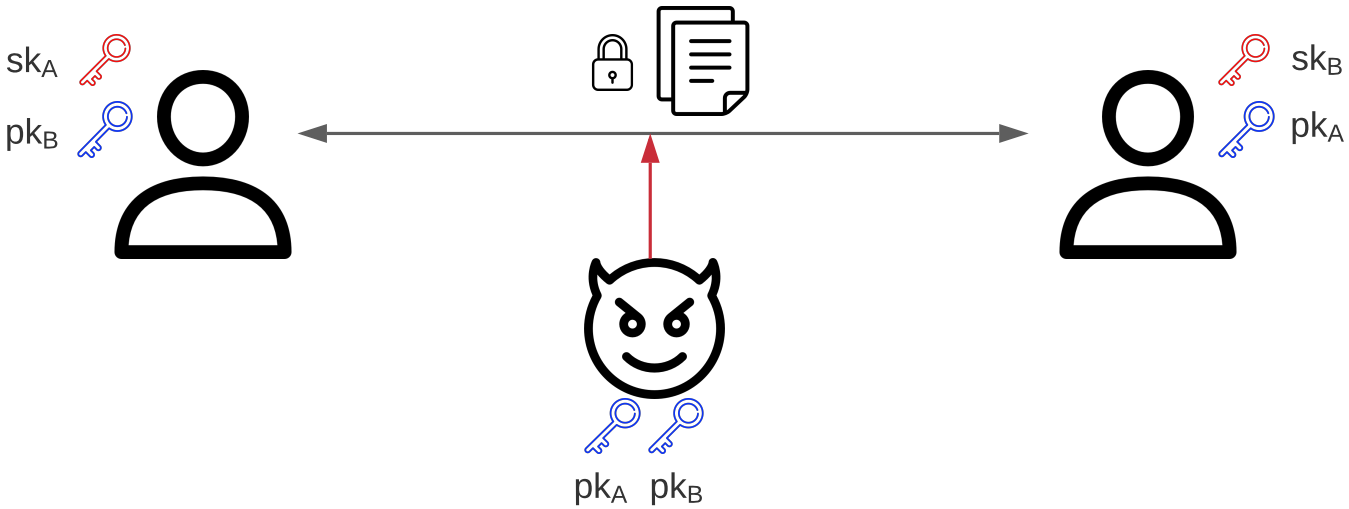
- Symmetric encryption
  - Confidentiality
  - AES-CBC, AES-CTR, RC4
- Message authentication codes
  - Integrity
  - HMAC, CMAC
- Authenticated Encryption with Associated Data (AEAD)
  - Confidentiality *and* Integrity

Symmetric Cryptography - P2

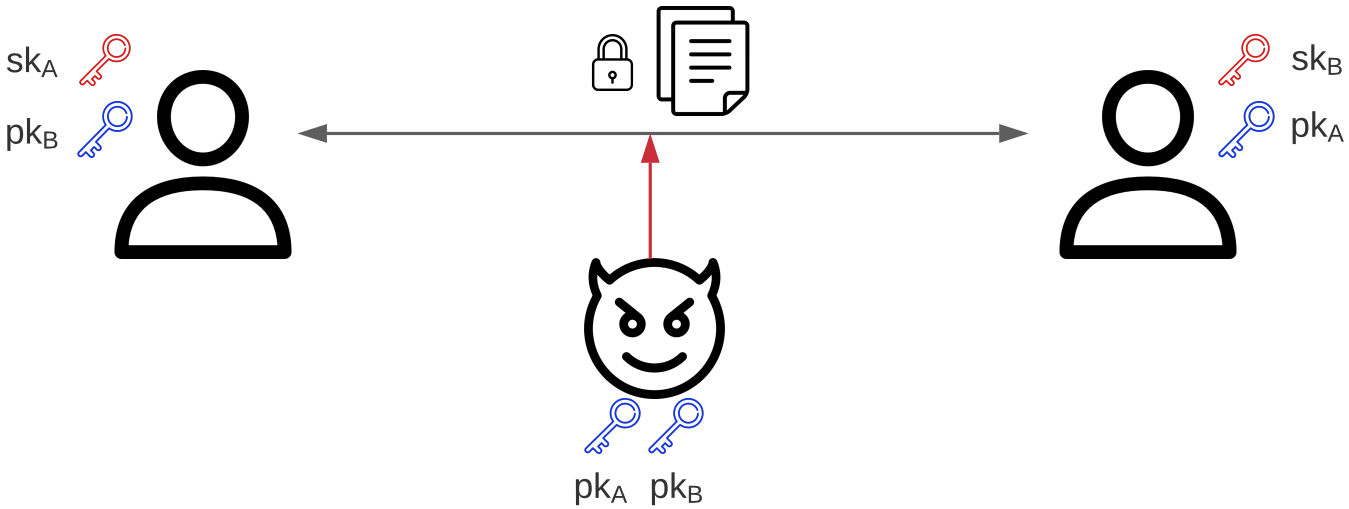
Primitives

- Symmetric encryption
  - Confidentiality
  - AES-CBC, AES-CTR, RC4
- Message authentication codes
  - Integrity
  - HMAC, CMAC
- Authenticated Encryption with Associated Data (AEAD)
  - Confidentiality *and* Integrity
  - AES-GCM, Poly-ChaCha

Public-Key Cryptography - P1

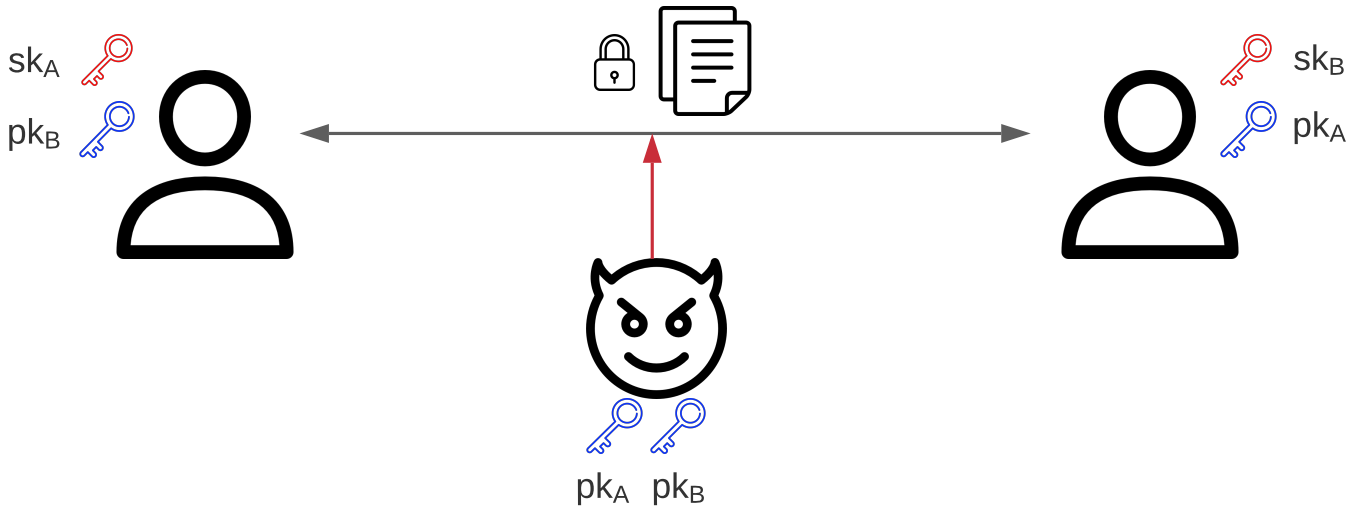


Public-Key Cryptography - P1



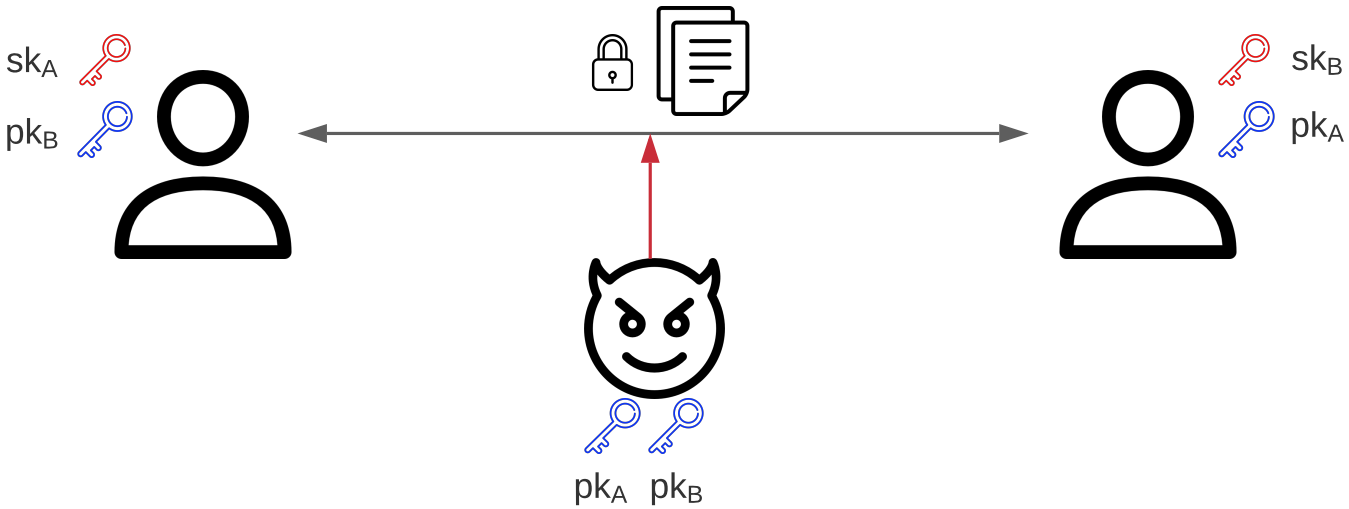
- Users work with different keys

Public-Key Cryptography - P1



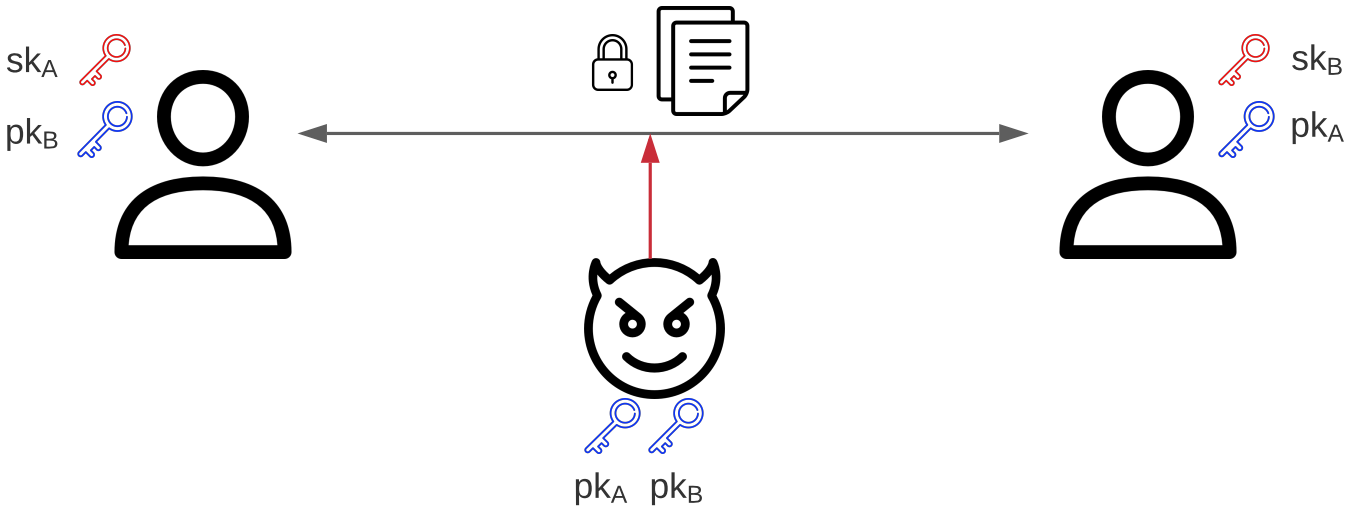
- Users work with different keys
- Encrypt messages passed in the channel

Public-Key Cryptography - P1



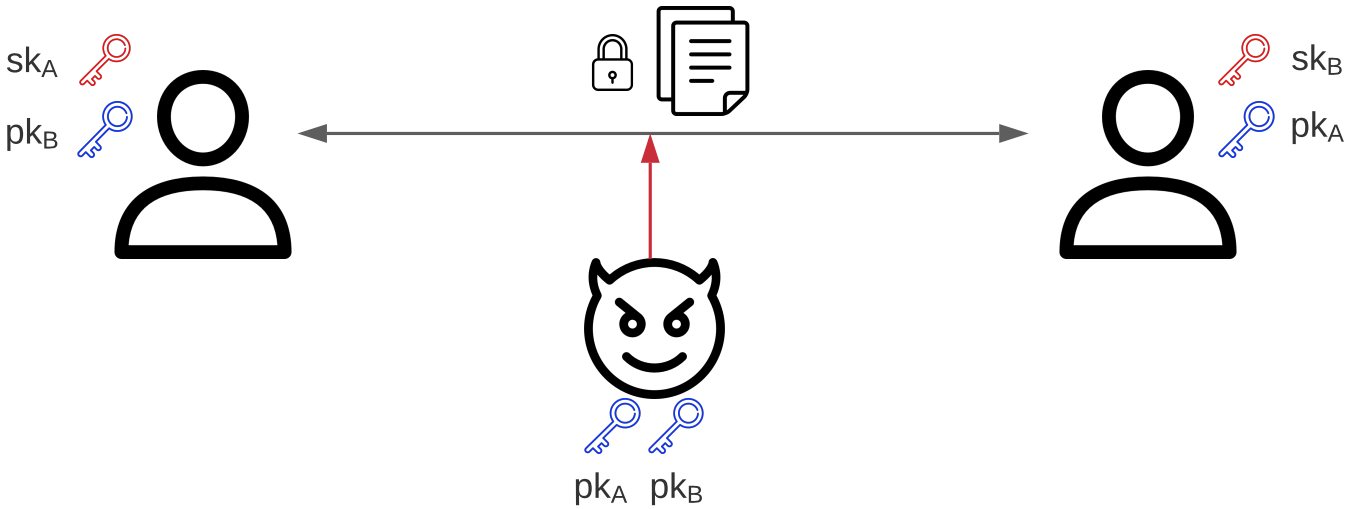
- Users work with different keys
- Encrypt messages passed in the channel
  - Protect message *m*, using key *pk* to produce ciphertext *c*

Public-Key Cryptography - P1



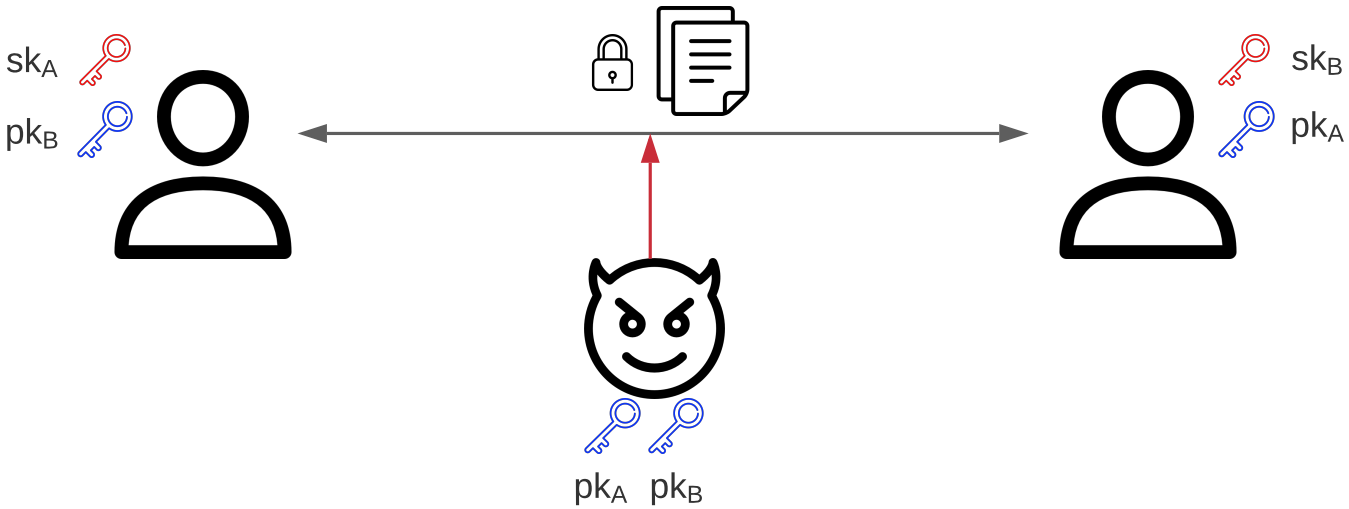
- Users work with different keys
- Encrypt messages passed in the channel
  - Protect message m, using key pk to produce ciphertext c
  - Encryption: $c = \text{Encrypt}(pk, m)$

Public-Key Cryptography - P1



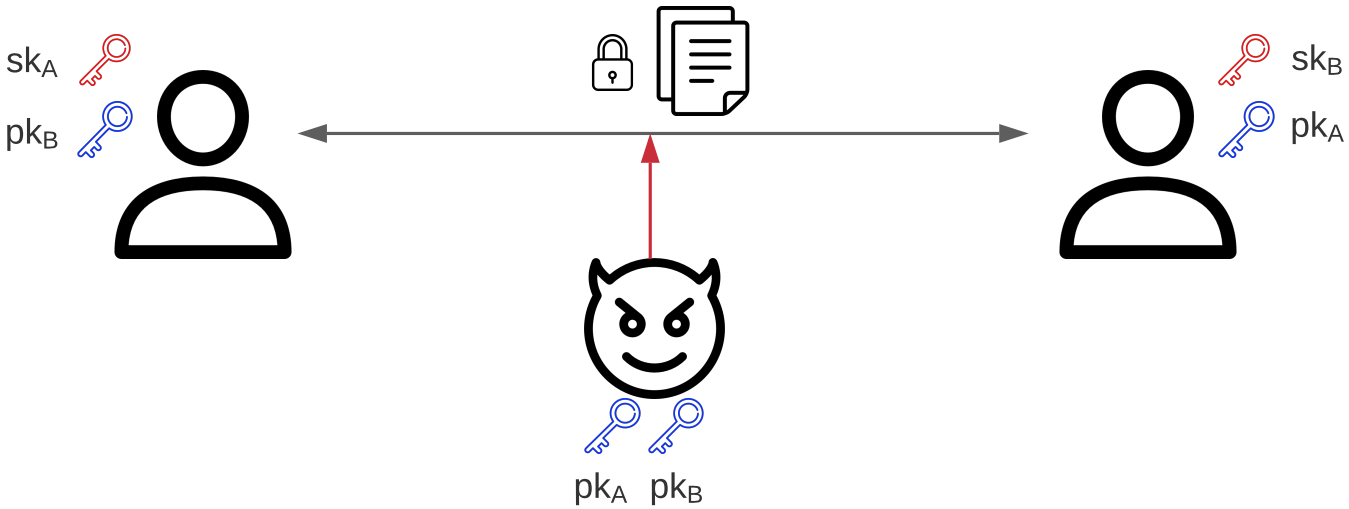
- Users work with different keys
- Encrypt messages passed in the channel
  - Protect message m, using key pk to produce ciphertext c
  - Encryption: $c = \text{Encrypt}(pk, m)$
  - Decryption: $m = \text{Decrypt}(sk, c)$

Public-Key Cryptography - P1



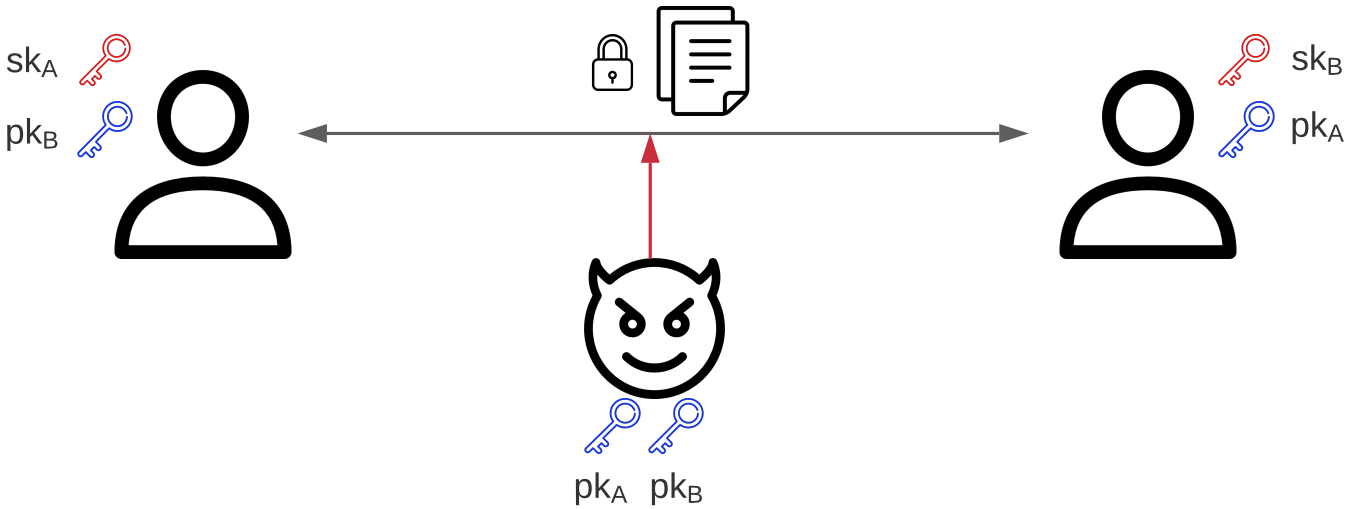
- Users work with different keys
- Encrypt messages passed in the channel
  - Protect message m, using key pk to produce ciphertext c
  - Encryption: $c = \text{Encrypt}(pk, m)$
  - Decryption: $m = \text{Decrypt}(sk, c)$
- Protect the integrity of messages in the channel

Public-Key Cryptography - P1



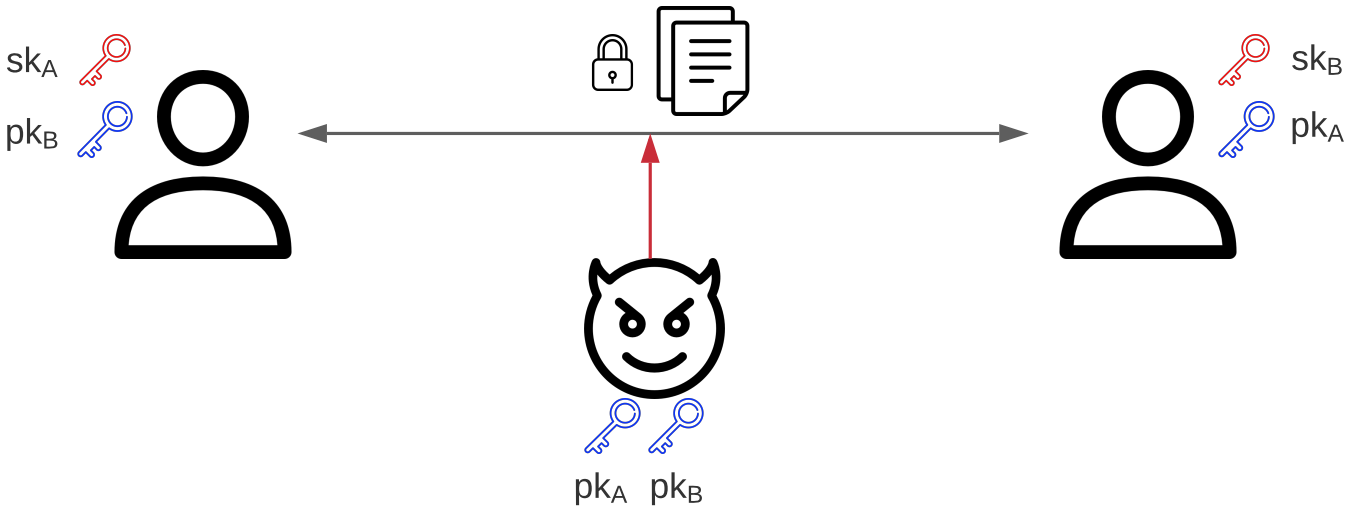
- Users work with different keys
- Encrypt messages passed in the channel
  - Protect message m, using key pk to produce ciphertext c
  - Encryption: $c = \text{Encrypt}(pk, m)$
  - Decryption: $m = \text{Decrypt}(sk, c)$
- Protect the integrity of messages in the channel
  - Protect message m, using key sk to produce signature t

Public-Key Cryptography - P1



- Users work with different keys
- Encrypt messages passed in the channel
  - Protect message m, using key pk to produce ciphertext c
  - Encryption: $c = \text{Encrypt}(pk, m)$
  - Decryption: $m = \text{Decrypt}(sk, c)$
- Protect the integrity of messages in the channel
  - Protect message m, using key sk to produce signature t
  - Authentication: $t = \text{Sign}(sk, m)$

Public-Key Cryptography - P1



- Users work with different keys
- Encrypt messages passed in the channel
  - Protect message m, using key pk to produce ciphertext c
  - Encryption: $c = \text{Encrypt}(pk, m)$
  - Decryption: $m = \text{Decrypt}(sk, c)$
- Protect the integrity of messages in the channel
  - Protect message m, using key sk to produce signature t
  - Authentication: $t = \text{Sign}(sk, m)$
  - Verification: $T/F = \text{Verify}(pk, m, t)$

Public-Key Cryptography - P2

Primitives

- Encryption

Public-Key Cryptography - P2

Primitives

- Encryption
  - Confidentiality, Integrity

Public-Key Cryptography - P2

Primitives

- Encryption
  - Confidentiality, Integrity
  - RSA-OAEP

Public-Key Cryptography - P2

Primitives

- Encryption
  - Confidentiality, Integrity
  - RSA-OAEP
- Digital Signatures

Public-Key Cryptography - P2

Primitives

- Encryption
  - Confidentiality, Integrity
  - RSA-OAEP
- Digital Signatures
  - Integrity, Non-repudiation

Public-Key Cryptography - P2

Primitives

- Encryption
  - Confidentiality, Integrity
  - RSA-OAEP
- Digital Signatures
  - Integrity, Non-repudiation
  - Schnorr

Public-Key Cryptography - P2

Primitives

- Encryption
  - Confidentiality, Integrity
  - RSA-OAEP
- Digital Signatures
  - Integrity, Non-repudiation
  - Schnorr
- Key exchange protocols

Public-Key Cryptography - P2

Primitives

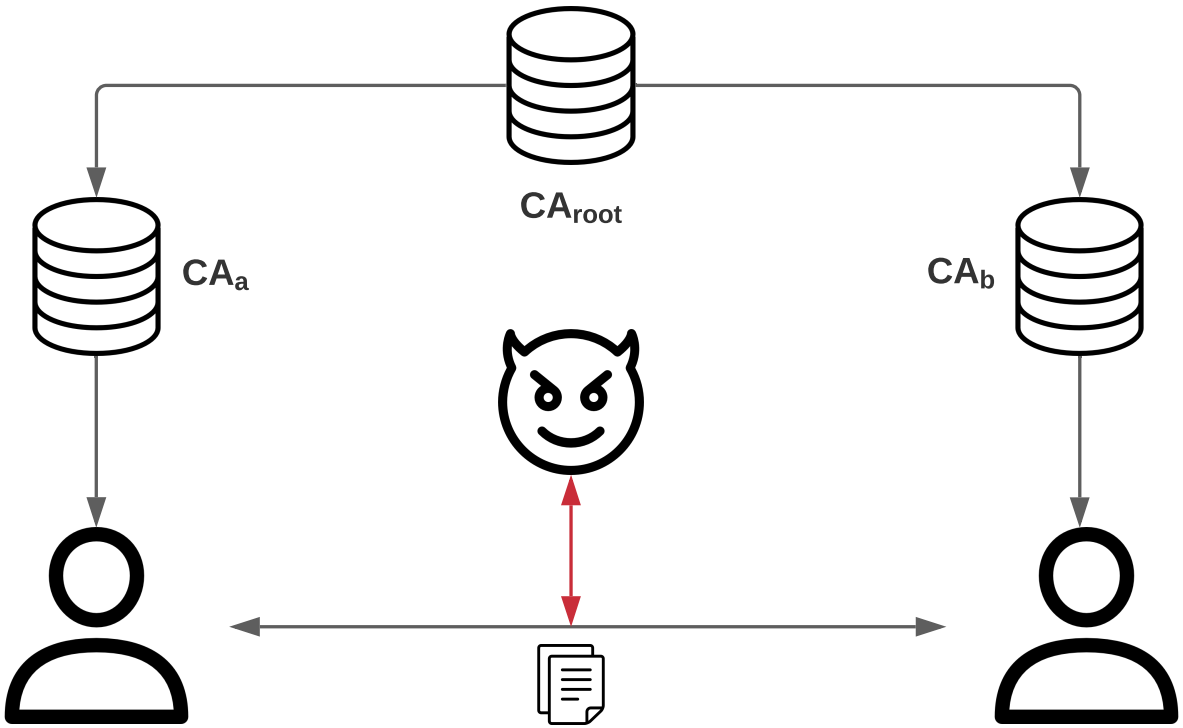
- Encryption
  - Confidentiality, Integrity
  - RSA-OAEP
- Digital Signatures
  - Integrity, Non-repudiation
  - Schnorr
- Key exchange protocols
  - Exchange symmetric key

Public-Key Cryptography - P2

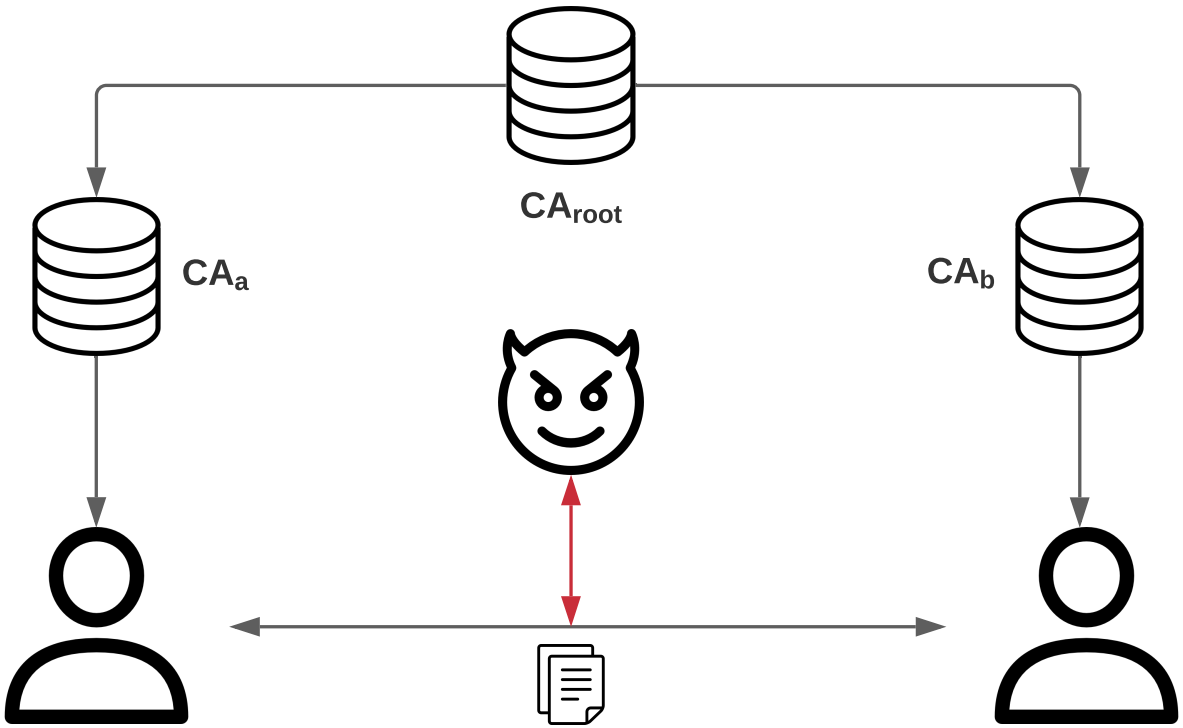
Primitives

- Encryption
  - Confidentiality, Integrity
  - RSA-OAEP
- Digital Signatures
  - Integrity, Non-repudiation
  - Schnorr
- Key exchange protocols
  - Exchange symmetric key
  - Diffie-Hellman

Public-key Infrastructure

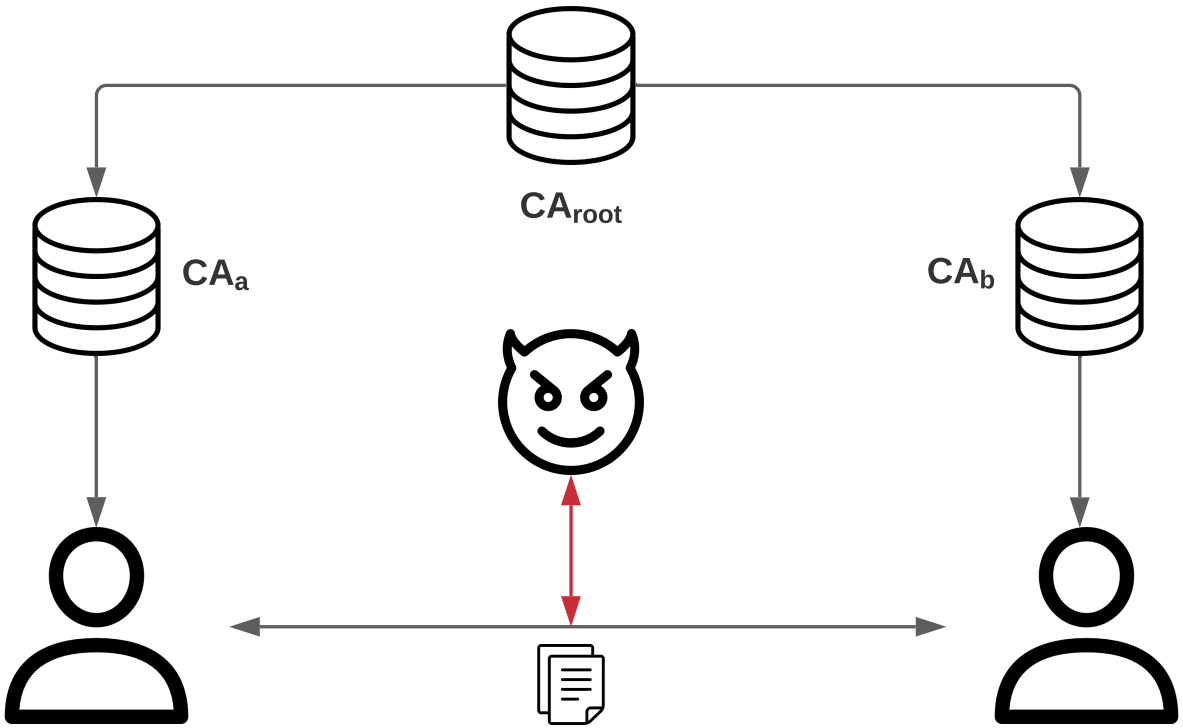


Public-key Infrastructure



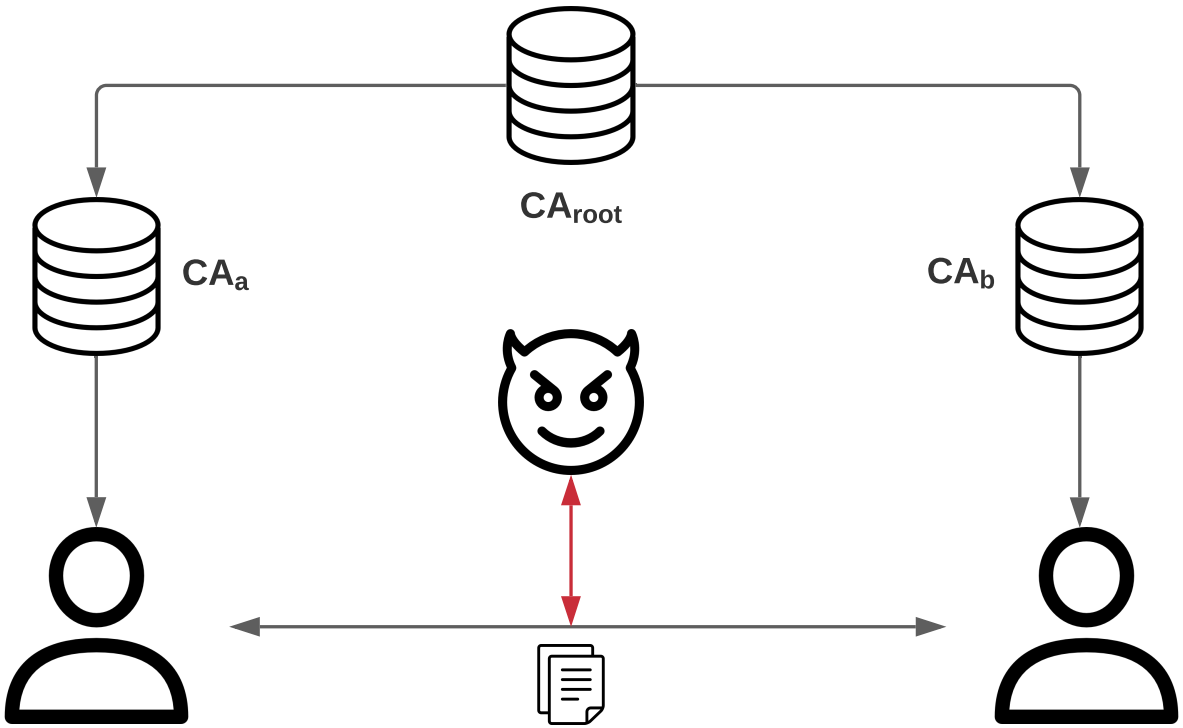
- A and B trust CA_{root}

Public-key Infrastructure



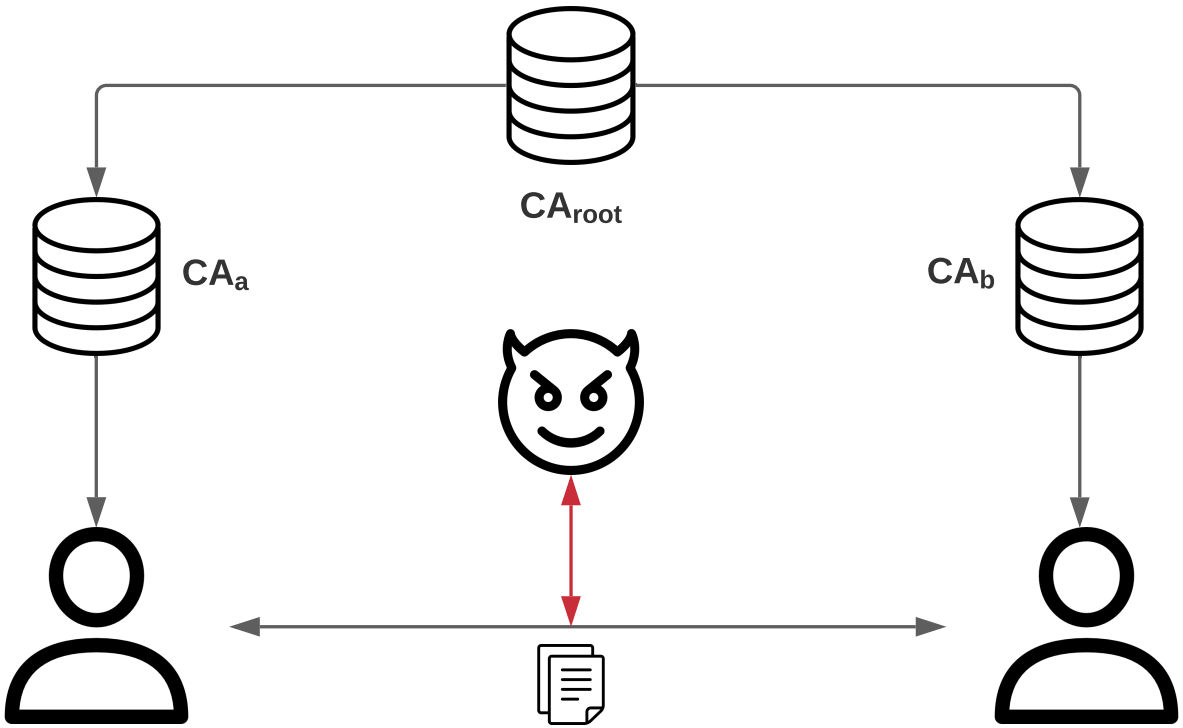
- A and B trust CA_{root}
- They might not trust CA_A or CA_B

Public-key Infrastructure



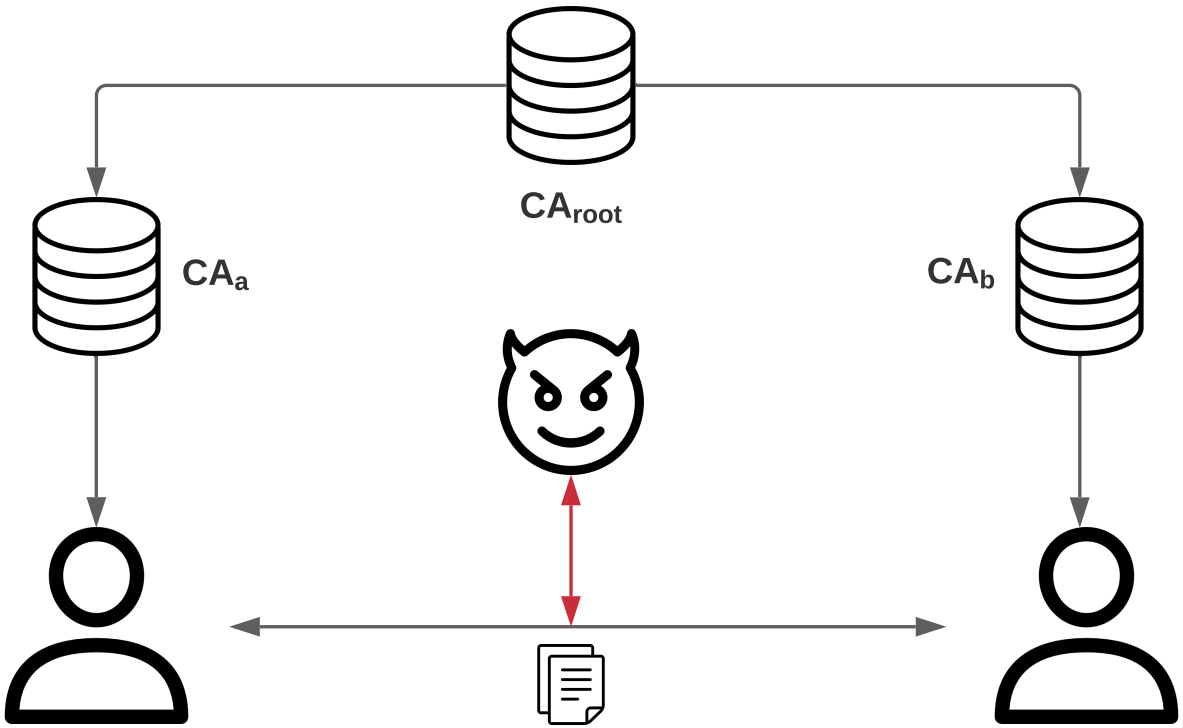
- A and B trust CA_{root}
- They might not trust CA_A or CA_B
- Trust hierarchy

Public-key Infrastructure



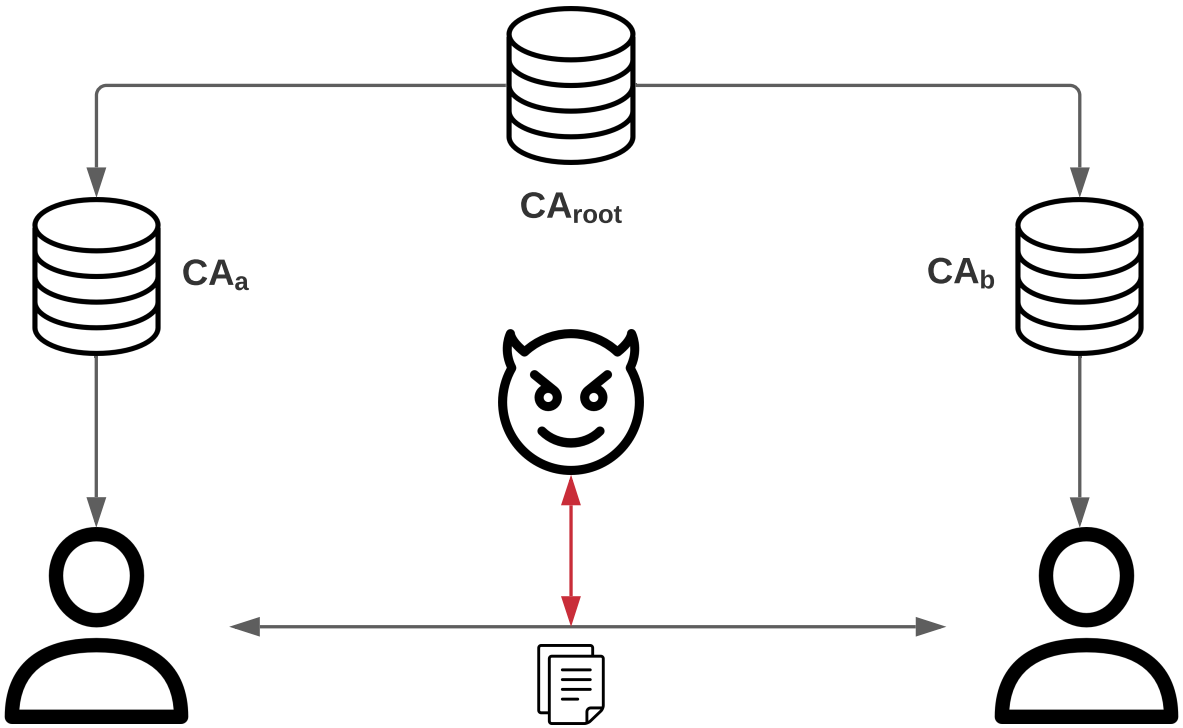
- A and B trust CA_{root}
- They might not trust CA_A or CA_B
- Trust hierarchy
  - Root certifies other CAs

Public-key Infrastructure



- A and B trust CA_{root}
- They might not trust CA_A or CA_B
- Trust hierarchy
  - Root certifies other CAs
  - Sub-CAs certify public keys

Public-key Infrastructure



- A and B trust CA_{root}
- They might not trust CA_A or CA_B
- Trust hierarchy
  - Root certifies other CAs
  - Sub-CAs certify public keys
  - Alice and Bob exchange certificates

Trusted Computing Base

Bottom-line: We have to assume something!

Trusted Computing Base

Bottom-line: We have to assume something!

Trusted Computing Base (TCB)

Trusted Computing Base

Bottom-line: We have to assume something!

Trusted Computing Base (TCB)

- Any security system has it

Trusted Computing Base

Bottom-line: We have to assume something!

Trusted Computing Base (TCB)

- Any security system has it
- Components we will have to *assume* work as expected

Trusted Computing Base

Bottom-line: We have to assume something!

Trusted Computing Base (TCB)

- Any security system has it
- Components we will have to *assume* work as expected
- Can have multiple concrete definitions

Trusted Computing Base

Bottom-line: We have to assume something!

Trusted Computing Base (TCB)

- Any security system has it
- Components we will have to *assume* work as expected
- Can have multiple concrete definitions
- Does not mean trust is unwarranted

Trusted Computing Base

Bottom-line: We have to assume something!

Trusted Computing Base (TCB)

- Any security system has it
- Components we will have to *assume* work as expected
- Can have multiple concrete definitions
- Does not mean trust is unwarranted
  - Cryptographic coprocessors

Trusted Computing Base

Bottom-line: We have to assume something!

Trusted Computing Base (TCB)

- Any security system has it
- Components we will have to *assume* work as expected
- Can have multiple concrete definitions
- Does not mean trust is unwarranted
  - Cryptographic coprocessors
  - Tamper-resistant

Trusted Computing Base

Bottom-line: We have to assume something!

Trusted Computing Base (TCB)

- Any security system has it
- Components we will have to *assume* work as expected
- Can have multiple concrete definitions
- Does not mean trust is unwarranted
  - Cryptographic coprocessors
  - Tamper-resistant
  - Standard-compliant APIs

Trusted Computing Base

Bottom-line: We have to assume something!

Trusted Computing Base (TCB)

- Any security system has it
- Components we will have to *assume* work as expected
- Can have multiple concrete definitions
- Does not mean trust is unwarranted
  - Cryptographic coprocessors
  - Tamper-resistant
  - Standard-compliant APIs
- Trusted hardware not covered, but important to acknowledge!

Trust Domains

A trust domain is:

A security context within which participants agree on:

- Identity proofing rules (how identities are established and validated)

Trust Domains

A trust domain is:

A security context within which participants agree on:

- Identity proofing rules (how identities are established and validated)
- Credential issuance and lifecycle management

Trust Domains

A trust domain is:

A security context within which participants agree on:

- Identity proofing rules (how identities are established and validated)
- Credential issuance and lifecycle management
- Cryptographic trust anchors (e.g., root certificates, key hierarchies)

Trust Domains

A trust domain is:

A security context within which participants agree on:

- Identity proofing rules (how identities are established and validated)
- Credential issuance and lifecycle management
- Cryptographic trust anchors (e.g., root certificates, key hierarchies)
- Policies and assurance levels for authentication, authorization, and data protection

Trust Domains

A trust domain is:

A security context within which participants agree on:

- Identity proofing rules (how identities are established and validated)
- Credential issuance and lifecycle management
- Cryptographic trust anchors (e.g., root certificates, key hierarchies)
- Policies and assurance levels for authentication, authorization, and data protection

- It ensures that all parties can rely on the validity and integrity of transactions carried out within that boundary.
- Trust domains can be hierarchical (e.g., PKI), federated (e.g., SAML/OIDC federations), or cross-organizational (e.g., interoperability frameworks).

Access Control

Authentication

Determining whether a user should be allowed access to a system

Access Control

Authentication

Determining whether a user should be allowed access to a system

- Local machines - password, smartcards, biometrics
  - Something you know / have / are

Access Control

Authentication

Determining whether a user should be allowed access to a system

- Local machines - password, smartcards, biometrics
  - Something you know / have / are
- **Through network - security protocols**

Access Control

Authentication

Determining whether a user should be allowed access to a system

- Local machines - password, smartcards, biometrics
  - Something you know / have / are
- **Through network - security protocols**

Authorization

Deciding what actions a user can perform in the system

Access Control

Authentication

Determining whether a user should be allowed access to a system

- Local machines - password, smartcards, biometrics
  - Something you know / have / are
- **Through network - security protocols**

Authorization

Deciding what actions a user can perform in the system

- Fine-grained set of restrictions to system resources

Access Control

Authentication

Determining whether a user should be allowed access to a system

- Local machines - password, smartcards, biometrics
  - Something you know / have / are
- **Through network - security protocols**

Authorization

Deciding what actions a user can perform in the system

- Fine-grained set of restrictions to system resources
  - Check system characteristics
  - Read/write data
  - Alter configurations

Protocols

- Human protocols - Rules followed in human interactions
  - *E.g.* Asking a question in class

Protocols

- Human protocols - Rules followed in human interactions
  - *E.g.* Asking a question in class
- Networking protocols - Rules followed in networked communication systems
  - *E.g.* HTTP, FTP, etc.

Protocols

- Human protocols - Rules followed in human interactions
  - *E.g.* Asking a question in class
- Networking protocols - Rules followed in networked communication systems
  - *E.g.* HTTP, FTP, etc.
- Security Protocol - the (communication) rules followed in a security application
  - *E.g.* SSL, IPSec, SSH, Kerberos, etc.

Protocol Correctness/Security

Protocol flaws range from **obvious** to extremely **subtle**

Protocol Correctness/Security

Protocol flaws range from **obvious** to extremely **subtle**

- Error in checking string length values

Protocol Correctness/Security

Protocol flaws range from **obvious** to extremely **subtle**

- Error in checking string length values
- Configuration allows for outdated crypto

Protocol Correctness/Security

Protocol flaws range from **obvious** to extremely **subtle**

- Error in checking string length values
- Configuration allows for outdated crypto
- Statistically skewed randomness

Protocol Correctness/Security

Protocol flaws range from **obvious** to extremely **subtle**

- Error in checking string length values
- Configuration allows for outdated crypto
- Statistically skewed randomness

Many protocols fall victim

- WEP for wireless networks

Protocol Correctness/Security

Protocol flaws range from **obvious** to extremely **subtle**

- Error in checking string length values
- Configuration allows for outdated crypto
- Statistically skewed randomness

Many protocols fall victim

- WEP for wireless networks
- Heartbleed SSL security flaw (2014)

Protocol Correctness/Security

Protocol flaws range from **obvious** to extremely **subtle**

- Error in checking string length values
- Configuration allows for outdated crypto
- Statistically skewed randomness

Many protocols fall victim

- WEP for wireless networks
- Heartbleed SSL security flaw (2014)

Designing *trustworthy* protocols is a challenging task...

Ideal Security Protocols

So what makes a protocol *trustworthy*?

Ideal Security Protocols

So what makes a protocol *trustworthy*?

- Satisfy security requirements
  - But they have to be precise

Ideal Security Protocols

So what makes a protocol *trustworthy*?

- Satisfy security requirements
  - But they have to be precise
- Efficient
  - Minimize computational resources
  - Minimize bandwidth usage and delays

Ideal Security Protocols

So what makes a protocol *trustworthy*?

- Satisfy security requirements
  - But they have to be precise
- Efficient
  - Minimize computational resources
  - Minimize bandwidth usage and delays
- Robust
  - Works when an attacker attempts to break it
  - Works when the environment is unstable(-ish)

Ideal Security Protocols

So what makes a protocol *trustworthy*?

- Satisfy security requirements
  - But they have to be precise
- Efficient
  - Minimize computational resources
  - Minimize bandwidth usage and delays
- Robust
  - Works when an attacker attempts to break it
  - Works when the environment is unstable(-ish)
- Easy to implement and operate

Ideal Security Protocols

So what makes a protocol *trustworthy*?

- Satisfy security requirements
  - But they have to be precise
- Efficient
  - Minimize computational resources
  - Minimize bandwidth usage and delays
- Robust
  - Works when an attacker attempts to break it
  - Works when the environment is unstable(-ish)
- Easy to implement and operate
- Flexible in configuration and deployment