Extension and Integration of an Abstract Interface to Cryptography Providers

Steve Wagner

EI-3nat7

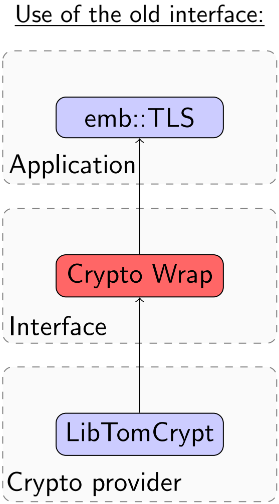
Prof. Dr. Sikora
Dipl.-Phys. Andreas Walz

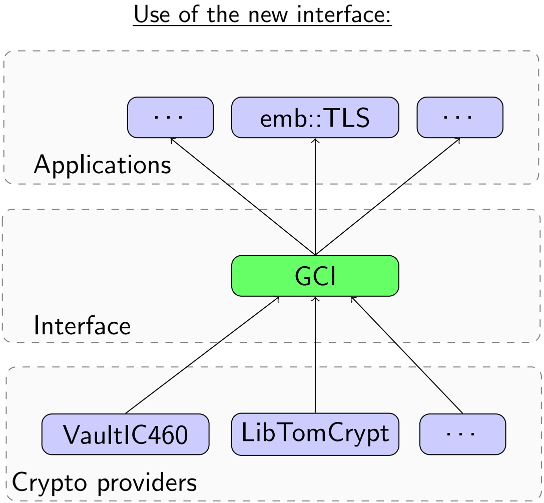
Laboratory Embedded Systems and Communication Electronics (ivESK)
Offenburg University of Applied Science

Table of contents

1. Goal of the project
2. Interface
3. Implementation
4. Results
5. Conclusion

Goal of the project





Requirements

Old cryptographic interface:

- Cannot be use in other application without changing some functions
- No other library can be use without rewriting the interface
- To old regarding the evolution of the cryptograhy

New cryptographc interface GCI:

- Possibility to use other cryptographic libraries
- Possibility to use it in hardware-coded-based cryptographic modules
- Possibility to easily add new cryptographic algorithms

Scheduling of the project

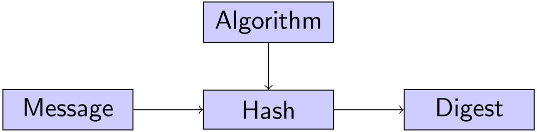
1. Acquisition of the basic cryptographic algorithms
2. Acquisition of TLS's princip and the implementation emb::TLS
3. Understanding the design of old cryptographic interface (Crypto Wrap)
4. Analysis of the cryptographic requirements imposed by TLS
5. Integration of the new interface in the application emb::TLS
6. Implementation of the provider LibTomCrypt

Interface

Interface in 5 main cryptography parts:

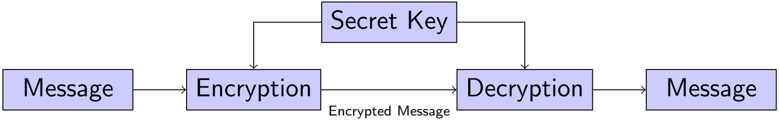
- Hash
- Symmetric cipher
- Asymmetric cipher
- Signature
- Diffie-Hellman

Interface - Hash



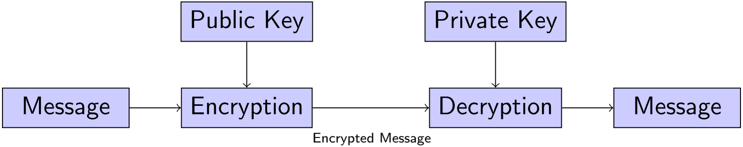
- quick to compute for any message
- infeasible to modify a message without changing the hash
- infeasible to find two different message with the same hash

Interface - Sym. Cipher



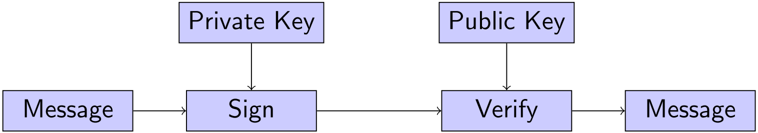
- Same key uses for Encryption and Decryption
- Allows privacy of data (nobody can understand the encrypted message only he has the secret key for decrypting it)

Interface - Asym. Cipher



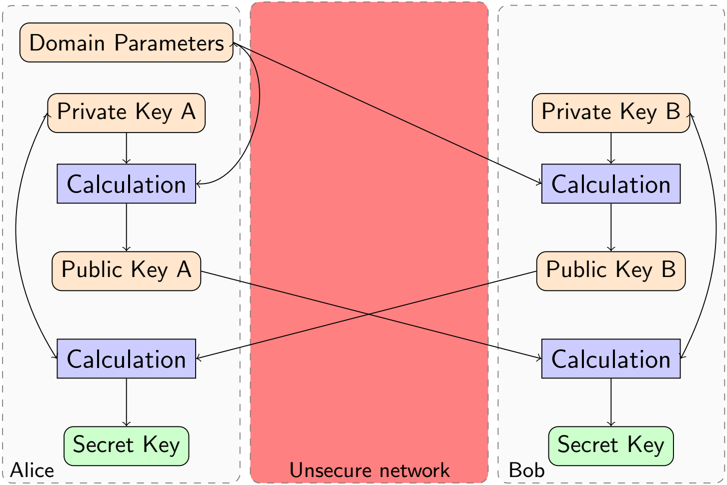
- Public and private key created by one person
- Public key sended to everyone who wants to get a communication
- Private key stay by who has created the key pair
- Public key allows the Encryption of the message
- Private key only can decrypt the message
- Allows privacy (only this one who has the private key can decypt the messages)
- Integraty (Sure that nobody can decrypt the messages)

Interface - Signature



- Private key uses to sign the message
- Public to verify the signature
- Allows no-repudiation (means that we are sure who sended this message)

Interface - Diffie-Hellman



Interface - Diffie-Hellman

- Domain parameters created by one person (Alice)
- Private key created by each one and not sended
- Public key computed with the domain parameters and the private key
- Secret key compute with the own private key and the public key of the other person
- Same secret key in each side

- Start with asymmetric keys
- Finish with symmetric key

Interface - Contexts

Princip

Old cryptographic interface (Crypto Wrap):

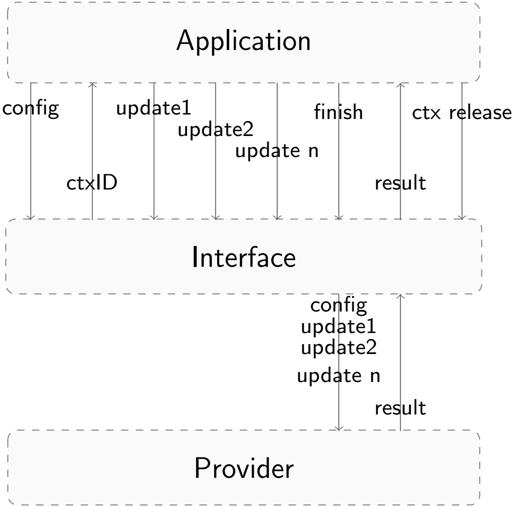
- Several functions with only one parameter of difference
- Several times the same parameters added for doing the same thing

New cryptographic interface (GCI):

- Use of context to save one time the parameters
- Give an identifiant (ID) back with where are the parameters saved
- Use of the ID to update the datas and get the result
- Release the context (the parameters in the same time) to free memory

Interface - Contexts

Example of use:



Princip

Old cryptographic interface (Crypto Wrap):

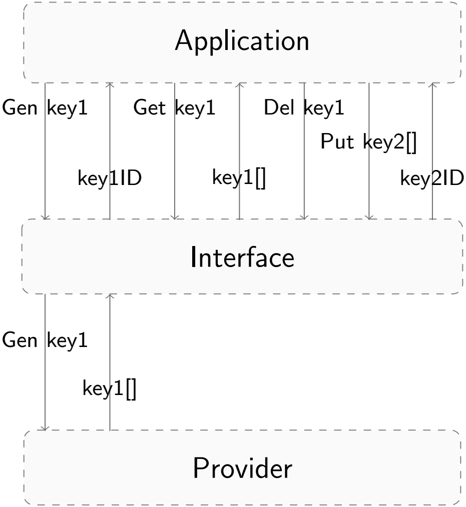
- Do not use memory to save several times the same key

New cryptographic interface (GCI):

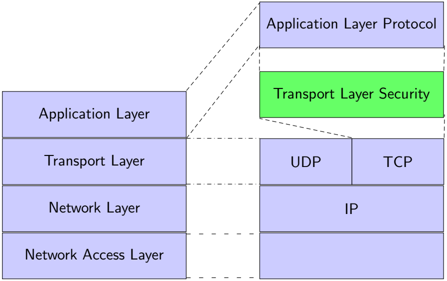
- Use of key management to save keys and become an identifiant (ID) of where they are saved
- Key could be get by passing the ID
- Release the key saved to free memory

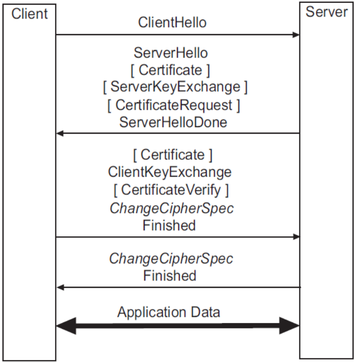
Interface - Key Management

Example of use:



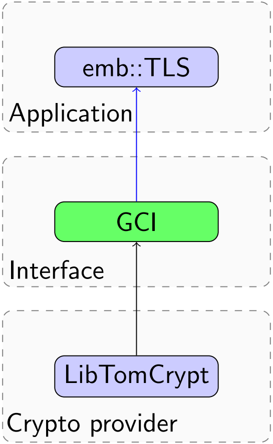
emb::TLS

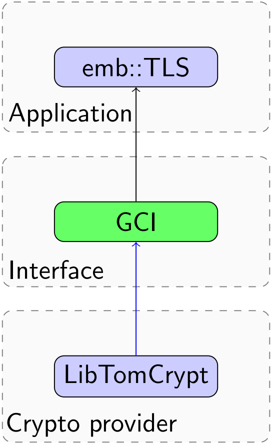




- emb::TLS as client with OpenSSL as server
- emb::TLS as server with Curve as client

Implementation



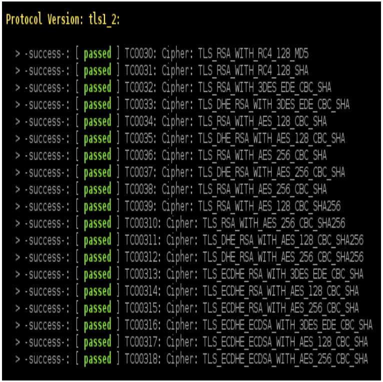


- Function from Cryto Wrap changed with this of GCI

- Implementation of the provider LibTomCrypt

Results - emb::TLS Client

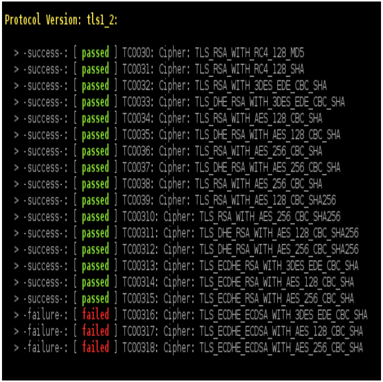
with old interface (Crypto Wrap):



```
Protocol Version: tls1_2:

 > -success-: [ passed ] TC0030: Cipher: TLS_RSA_WITH_RC4_128_MD5
 > -success-: [ passed ] TC0031: Cipher: TLS_RSA_WITH_RC4_128_SHA
 > -success-: [ passed ] TC0032: Cipher: TLS_RSA_WITH_3DES_EDE_CBC_SHA
 > -success-: [ passed ] TC0033: Cipher: TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
 > -success-: [ passed ] TC0034: Cipher: TLS_RSA_WITH_AES_128_CBC_SHA
 > -success-: [ passed ] TC0035: Cipher: TLS_DHE_RSA_WITH_AES_128_CBC_SHA
 > -success-: [ passed ] TC0036: Cipher: TLS_RSA_WITH_AES_256_CBC_SHA
 > -success-: [ passed ] TC0037: Cipher: TLS_DHE_RSA_WITH_AES_256_CBC_SHA
 > -success-: [ passed ] TC0038: Cipher: TLS_RSA_WITH_AES_256_CBC_SHA
 > -success-: [ passed ] TC0039: Cipher: TLS_RSA_WITH_AES_128_CBC_SHA256
 > -success-: [ passed ] TC00310: Cipher: TLS_RSA_WITH_AES_256_CBC_SHA256
 > -success-: [ passed ] TC00311: Cipher: TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
 > -success-: [ passed ] TC00312: Cipher: TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
 > -success-: [ passed ] TC00313: Cipher: TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
 > -success-: [ passed ] TC00314: Cipher: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
 > -success-: [ passed ] TC00315: Cipher: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
 > -success-: [ passed ] TC00316: Cipher: TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
 > -success-: [ passed ] TC00317: Cipher: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
 > -success-: [ passed ] TC00318: Cipher: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
```

with new interface (GCI):



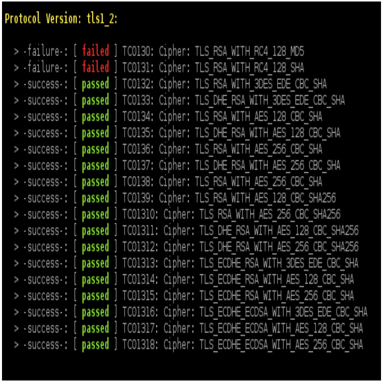
```
Protocol Version: tls1_2:

 > -success-: [ passed ] TC0030: Cipher: TLS_RSA_WITH_RC4_128_MD5
 > -success-: [ passed ] TC0031: Cipher: TLS_RSA_WITH_RC4_128_SHA
 > -success-: [ passed ] TC0032: Cipher: TLS_RSA_WITH_3DES_EDE_CBC_SHA
 > -success-: [ passed ] TC0033: Cipher: TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
 > -success-: [ passed ] TC0034: Cipher: TLS_RSA_WITH_AES_128_CBC_SHA
 > -success-: [ passed ] TC0035: Cipher: TLS_DHE_RSA_WITH_AES_128_CBC_SHA
 > -success-: [ passed ] TC0036: Cipher: TLS_RSA_WITH_AES_256_CBC_SHA
 > -success-: [ passed ] TC0037: Cipher: TLS_DHE_RSA_WITH_AES_256_CBC_SHA
 > -success-: [ passed ] TC0038: Cipher: TLS_RSA_WITH_AES_256_CBC_SHA
 > -success-: [ passed ] TC0039: Cipher: TLS_RSA_WITH_AES_128_CBC_SHA256
 > -success-: [ passed ] TC00310: Cipher: TLS_RSA_WITH_AES_256_CBC_SHA256
 > -success-: [ passed ] TC00311: Cipher: TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
 > -success-: [ passed ] TC00312: Cipher: TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
 > -success-: [ passed ] TC00313: Cipher: TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
 > -success-: [ passed ] TC00314: Cipher: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
 > -success-: [ passed ] TC00315: Cipher: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
 > -failure-: [ failed ] TC00316: Cipher: TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
 > -failure-: [ failed ] TC00317: Cipher: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
 > -failure-: [ failed ] TC00318: Cipher: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
```

- All cipher suites work
- Only ECDSA which isn't implemented doesn't work yet

Results - emb::TLS Server

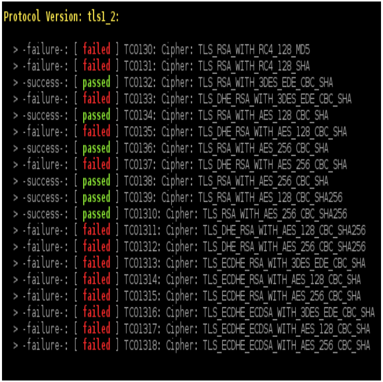
with old interface (Crypto Wrap):



```
Protocol Version: tls1_2:

 > -failure-: [ failed ] TC0130: Cipher: TLS_RSA_WITH_RC4_128_MD5
 > -failure-: [ failed ] TC0131: Cipher: TLS_RSA_WITH_RC4_128_SHA
 > -success-: [ passed ] TC0132: Cipher: TLS_RSA_WITH_3DES_EDE_CBC_SHA
 > -success-: [ passed ] TC0133: Cipher: TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
 > -success-: [ passed ] TC0134: Cipher: TLS_RSA_WITH_AES_128_CBC_SHA
 > -success-: [ passed ] TC0135: Cipher: TLS_DHE_RSA_WITH_AES_128_CBC_SHA
 > -success-: [ passed ] TC0136: Cipher: TLS_RSA_WITH_AES_256_CBC_SHA
 > -success-: [ passed ] TC0137: Cipher: TLS_DHE_RSA_WITH_AES_256_CBC_SHA
 > -success-: [ passed ] TC0138: Cipher: TLS_RSA_WITH_AES_256_CBC_SHA
 > -success-: [ passed ] TC0139: Cipher: TLS_RSA_WITH_AES_128_CBC_SHA256
 > -success-: [ passed ] TC01310: Cipher: TLS_RSA_WITH_AES_256_CBC_SHA256
 > -success-: [ passed ] TC01311: Cipher: TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
 > -success-: [ passed ] TC01312: Cipher: TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
 > -success-: [ passed ] TC01313: Cipher: TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
 > -success-: [ passed ] TC01314: Cipher: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
 > -success-: [ passed ] TC01315: Cipher: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
 > -success-: [ passed ] TC01316: Cipher: TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
 > -success-: [ passed ] TC01317: Cipher: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
 > -success-: [ passed ] TC01318: Cipher: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
```

with new interface (GCI):



```
Protocol Version: tls1_2:

 > -failure-: [ failed ] TC0130: Cipher: TLS_RSA_WITH_RC4_128_MD5
 > -failure-: [ failed ] TC0131: Cipher: TLS_RSA_WITH_RC4_128_SHA
 > -success-: [ passed ] TC0132: Cipher: TLS_RSA_WITH_3DES_EDE_CBC_SHA
 > -failure-: [ failed ] TC0133: Cipher: TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
 > -success-: [ passed ] TC0134: Cipher: TLS_RSA_WITH_AES_128_CBC_SHA
 > -failure-: [ failed ] TC0135: Cipher: TLS_DHE_RSA_WITH_AES_128_CBC_SHA
 > -success-: [ passed ] TC0136: Cipher: TLS_RSA_WITH_AES_256_CBC_SHA
 > -failure-: [ failed ] TC0137: Cipher: TLS_DHE_RSA_WITH_AES_256_CBC_SHA
 > -success-: [ passed ] TC0138: Cipher: TLS_RSA_WITH_AES_256_CBC_SHA
 > -success-: [ passed ] TC0139: Cipher: TLS_RSA_WITH_AES_128_CBC_SHA256
 > -success-: [ passed ] TC01310: Cipher: TLS_RSA_WITH_AES_256_CBC_SHA256
 > -failure-: [ failed ] TC01311: Cipher: TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
 > -failure-: [ failed ] TC01312: Cipher: TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
 > -failure-: [ failed ] TC01313: Cipher: TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
 > -failure-: [ failed ] TC01314: Cipher: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
 > -failure-: [ failed ] TC01315: Cipher: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
 > -failure-: [ failed ] TC01316: Cipher: TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
 > -failure-: [ failed ] TC01317: Cipher: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
 > -failure-: [ failed ] TC01318: Cipher: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
```

- Diffie-Hellman doesn't work yet (problem of implementation in emb::TLS)
- Elliptic Curve Diffie-Hellman doesn't work too
- ECDSA isn't implemented yet

Conclusion

What work:

- The new interface is in emb::TLS implemented
- The provider is in the interface implemented
- Client cipher suites work

What doesn't still work:

- Diffie-Hellman and Elliptic Curve Diffie-Hellman doesn't work in the server case
- ECDSA isn't implemented

Conclusion

TODOs:

1. Implementation of the rest of the server part
2. Add ECDSA in server and client
3. Write the documentation of the interface

Thanks for your attention

Questions?