Post-Quantum Blockchain

Mickaël TEAUDORS, Yoann VALERI

Supervised by Ludovic PERRET and Jean-Charles FAUGERE

Challenges

1 2 3

Challenges

- Incorporate post-quantum signature algorithms in an open-source Blockchain manipulation tool : Hyperledger

1 2 3

Challenges

- Incorporate post-quantum signature algorithms in an open-source Blockchain manipulation tool : Hyperledger
- Understand the basis of the signature scheme we modified and implemented

1 2 3

Challenges

- Incorporate post-quantum signature algorithms in an open-source Blockchain manipulation tool : Hyperledger
- Understand the basis of the signature scheme we modified and implemented
- Ensure compatibility between GO and C programs

1 2 3

Challenges

- Incorporate post-quantum signature algorithms in an open-source Blockchain manipulation tool : Hyperledger
- Understand the basis of the signature scheme we modified and implemented
- Ensure compatibility between GO and C programs
- Modular integration : Ease the change of the algorithm used

Challenges

- Incorporate post-quantum signature algorithms in an open-source Blockchain manipulation tool : Hyperledger
- Understand the basis of the signature scheme we modified and implemented
- Ensure compatibility between GO and C programs
- Modular integration : Ease the change of the algorithm used
- Evaluate the impact of these algorithms on the performances

Hyperledger : an open-source Blockchain manipulation tool

IBM

HYPERLEDGER

THE LINUX FOUNDATION

Hyperledger : an open-source Blockchain manipulation tool

- Open source framework for developing Blockchain-based applications

Hyperledger : an open-source Blockchain manipulation tool

- Open source framework for developing Blockchain-based applications
- Source code is written in GO

1 2 3

Hyperledger : an open-source Blockchain manipulation tool

- Open source framework for developing Blockchain-based applications
- Source code is written in GO
- Signature algorithm implemented is Schnorr [**C. P. Schnorr.** ***Efficient Identification and Signatures for Smart Cards***] combined with Elliptic curves [**Neal Koblitz, Alfred Menezes, and Scott Vanstone.** ***The State of Elliptic Curve Cryptography***]

IBM HYPERLEDGER THE LINUX FOUNDATION

Quantum security threat

1 2 3

Quantum security threat

- Two schemes in the Blockchain :
  - Proof-Of-Work, based on the **pre-image resistance** of hash functions
  - Signature, used to authenticate transactions in Blockchains

Quantum security threat

- Two schemes in the Blockchain :
  - Proof-Of-Work, based on the **pre-image resistance** of hash functions
  - Signature, used to authenticate transactions in Blockchains

- Quantum computers allow the deployment of already-existing algorithms
  - Grover's algorithm **[Lov K. Grover. *A fast quantum mechanical algorithm for database search*]**
    - Minimal impact on the Proof Of Work process in Blockchains
    - Size of keys has to be doubled to keep the same level of security

Quantum security threat

- Two schemes in the Blockchain :
  - Proof-Of-Work, based on the **pre-image resistance** of hash functions
  - Signature, used to authenticate transactions in Blockchains

- Quantum computers allow the deployment of already-existing algorithms
  - Grover's algorithm **[Lov K. Grover. *A fast quantum mechanical algorithm for database search*]**
    - Minimal impact on the Proof Of Work process in Blockchains
    - Size of keys has to be doubled to keep the same level of security
  - Shor's algorithms **[Peter W. Shor. *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*]**
    - Huge impact on the signature process
    - Polynomial algorithm to solve the discrete logarithm problem

1 2 3

Post-quantum signature solutions

- 9 Signature algorithms currently in round 2 at the NIST competition
  - Interface has to be respected

National Institute of Standards and Technology

Post-quantum signature solutions

National Institute of Standards and Technology

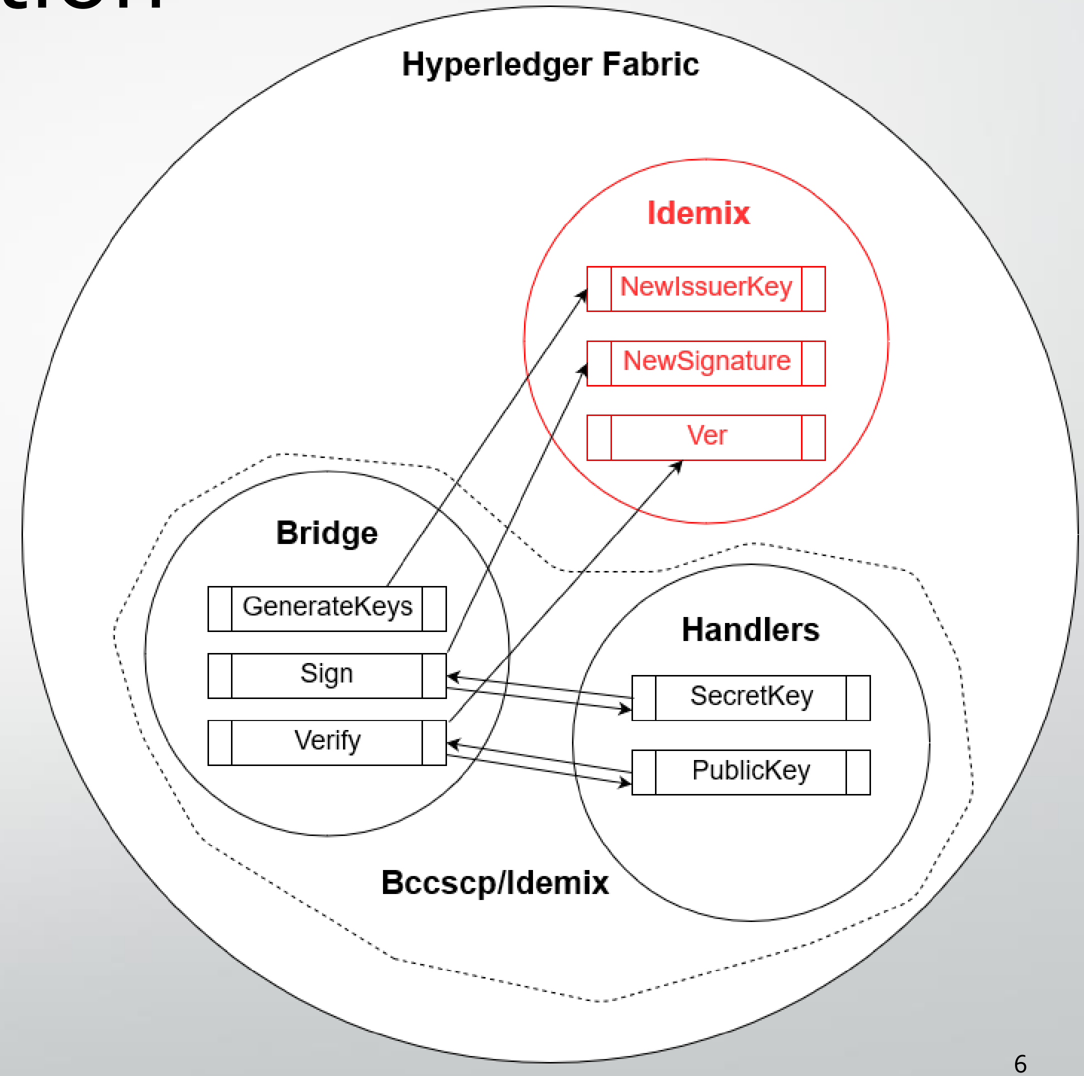
- 9 Signature algorithms currently in round 2 at the NIST competition
  - Interface has to be respected
- GeMSS **[Jean-Charles Faugère, Ludovic Perret and all. *GeMSS: A Great Multivariate Short Signature*]**
  - Security is based on multivariate polynomials equations over the finite field $\mathbb{F}_2$
  - Three security levels : 128, 192 or 256
  - Short signature length but longer keys
  - For GeMSS128 : 417,408 bytes and 14,208 bytes for public/secret keys, 48 bytes for signature

Post-quantum signature solutions

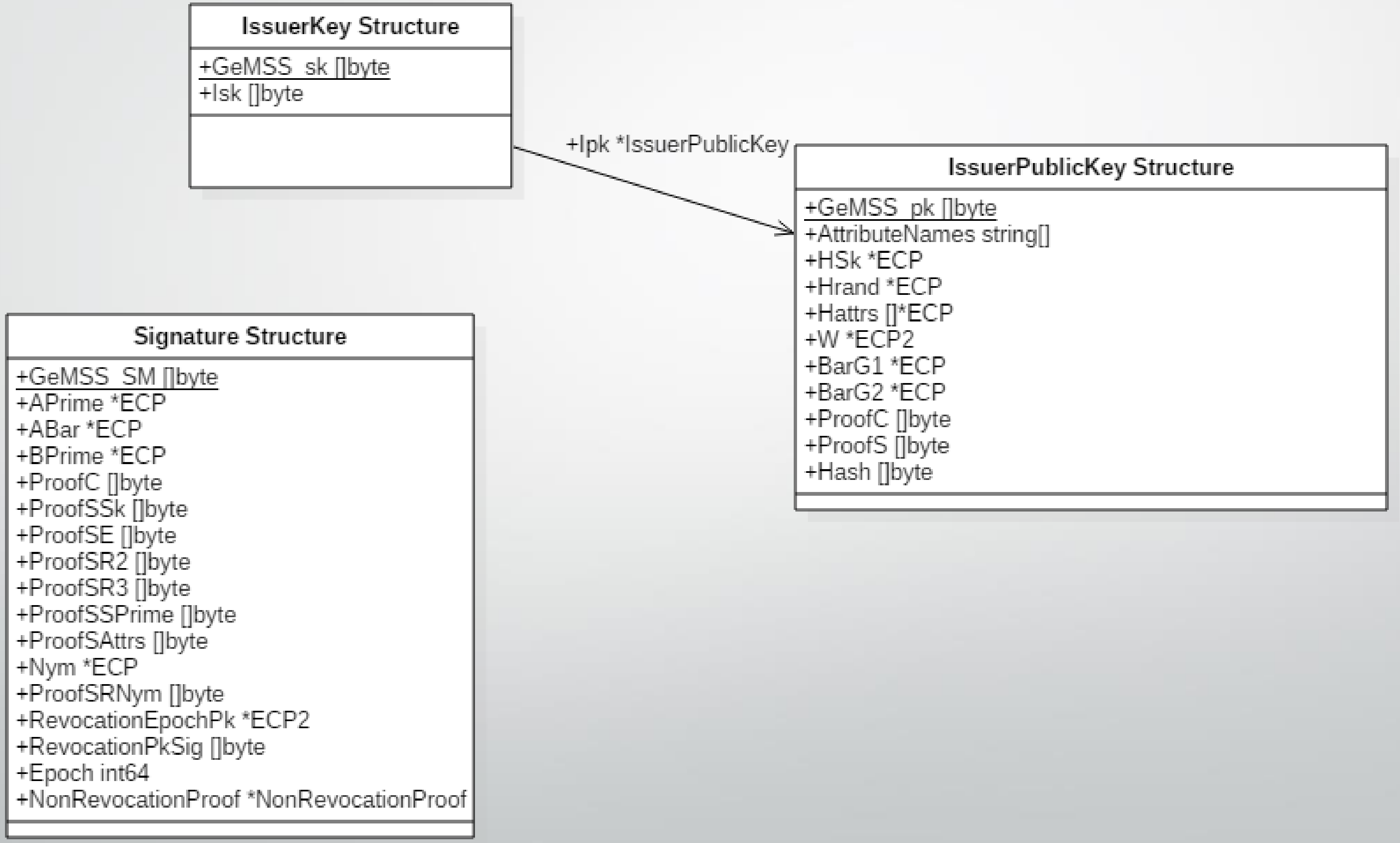
- 9 Signature algorithms currently in round 2 at the NIST competition
  - Interface has to be respected
- GeMSS **[Jean-Charles Faugère, Ludovic Perret and all. *GeMSS: A Great Multivariate Short Signature*]**
  - Security is based on multivariate polynomials equations over the finite field $\mathbb{F}_2$
  - Three security levels : 128, 192 or 256
  - Short signature length but longer keys
  - For GeMSS128 : 417,408 bytes and 14,208 bytes for public/secret keys, 48 bytes for signature
- MQDSS **[Ming-Shing Chen, Andreas Hulsing and all. *From 5-pass MQ-based identification to MQ-based signatures*]**
  - MQ problem **[Christopher Wolf and Bart Preneel. *Taxonomy of Public Key Schemes Based on the Problem of Multivariate Quadratic Equations*]** : multivariate polynomial equations over a finite field $\mathbb{F}_q$ with q a prime
  - Short keys but longer signature
  - 62 bytes and 32 bytes for public/secret keys, 32,882 bytes for signature

Hyperledger organisation

- *Idemix* : Implementation of the cryptographic functions
- *Handlers* : functions used to make the conversion between types
- *Bridge :* Converts parameters before a cryptographic function call



Integration : Structures



Integration : Functions

1 2 3

1 2 3

Integration : Functions

- Use of a static library to give access to GeMSS functions

Integration : Functions

- Use of a static library to give access to GeMSS functions
- Memory was the main problem

Integration : Functions

- Use of a static library to give access to GeMSS functions
- Memory was the main problem
- Memory allocation is done in GO before calling C functions
  - CGO gives access to standard C functions such as malloc, free ...

Integration : Functions

- Use of a static library to give access to GeMSS functions
- Memory was the main problem
- Memory allocation is done in GO before calling C functions
  - CGO gives access to standard C functions such as malloc, free …
- Prototypes of C functions must be declared in a header file

Integration : Functions

- Use of a static library to give access to GeMSS functions
- Memory was the main problem
- Memory allocation is done in GO before calling C functions
  - CGO gives access to standard C functions such as malloc, free …
- Prototypes of C functions must be declared in a header file
- Definition of a macro to choose the algorithm
  - GeMSS 128, 192 or 256
  - MQDSS

Integration : Functions

- Use of a static library to give access to GeMSS functions
- Memory was the main problem
- Memory allocation is done in GO before calling C functions
  - CGO gives access to standard C functions such as malloc, free …
- Prototypes of C functions must be declared in a header file
- Definition of a macro to choose the algorithm
  - GeMSS 128, 192 or 256
  - MQDSS
- Wrapper functions to perform correct-sized malloc and appropriate function calls

1 2 3

Integration : Functions

- Use of a static library to give access to GeMSS functions
- Memory was the main problem
- Memory allocation is done in GO before calling C functions
  - CGO gives access to standard C functions such as malloc, free …
- Prototypes of C functions must be declared in a header file
- Definition of a macro to choose the algorithm
  - GeMSS 128, 192 or 256
  - MQDSS
- Wrapper functions to perform correct-sized malloc and appropriate function calls
- Swap between post-quantum signature algorithms by changing one macro

1 2 3

Hybrid-cryptography model

$$s = S_1(H(m), sk_1) \,||\, S_2(H(m), sk_2)$$

- S_1 and S_2 : Signature algorithms
- sk_1, sk_2 : Secret keys
- $||$: Concatenation operator
- m : message to sign
- H : Hash function

1 2 3

Hybrid-cryptography model

$$s = S_1(H(m), sk_1\ ) \,||\, S_2(H(m), sk_2)$$

- S_1 and S_2 : Signature algorithms
- sk_1, sk_2 : Secret keys
- | | : Concatenation operator
- m : message to sign
- H : Hash function

- Add a layer of post-quantum security rather than completely changing it

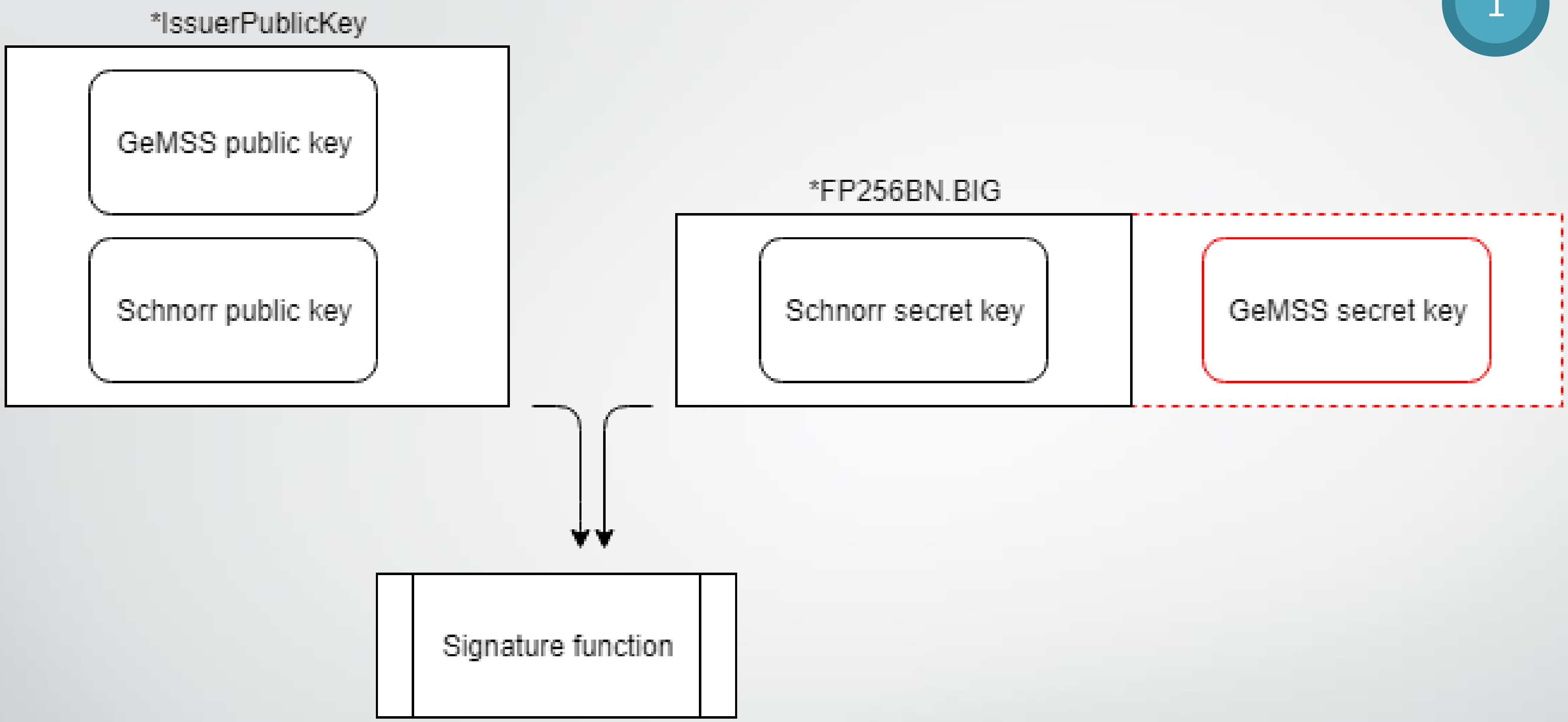
1 2 3

Hybrid-cryptography model

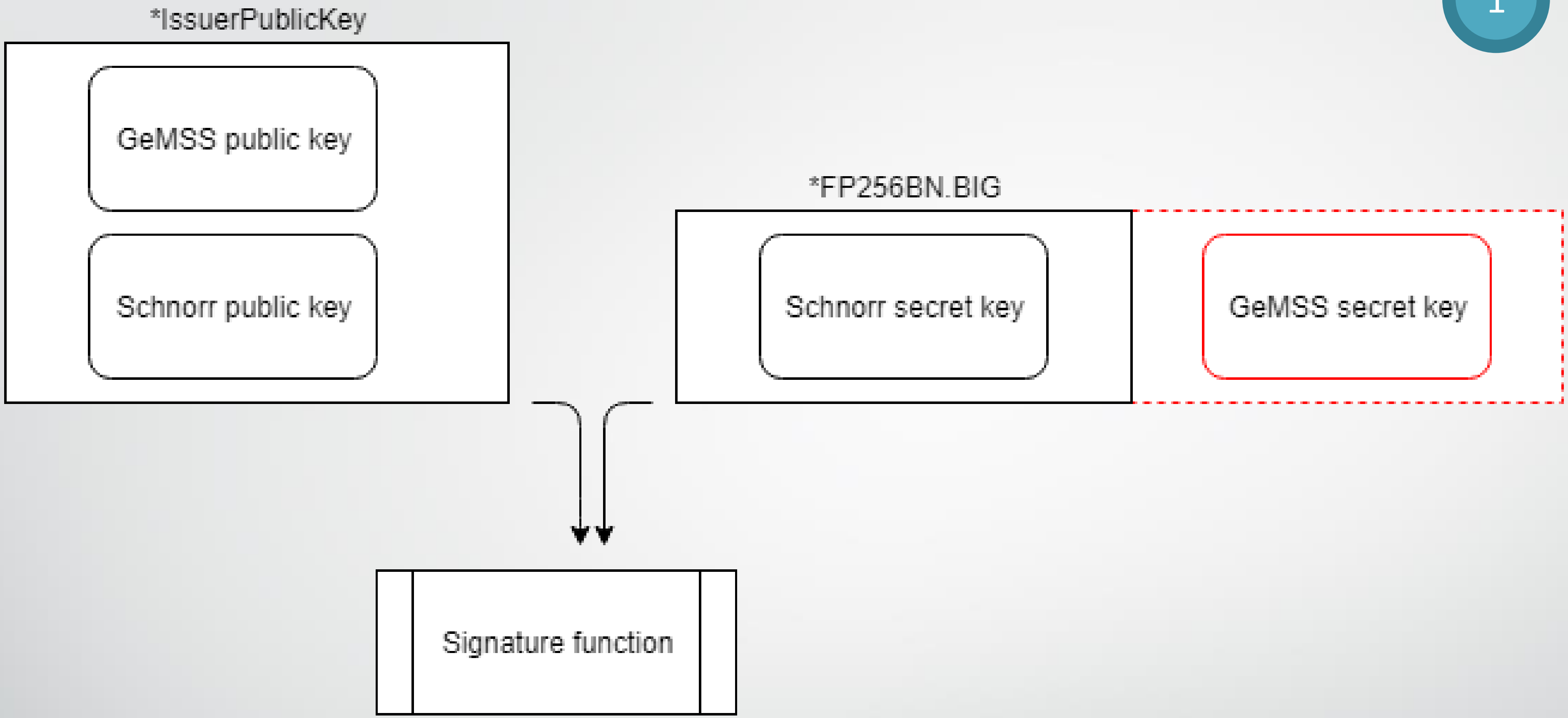
$$s = S_1(H(m), sk_1\ ) \,||\, S_2(H(m), sk_2)$$

- S_1 and S_2 : Signature algorithms
- sk_1, sk_2 : Secret keys
- | | : Concatenation operator
- m : message to sign
- H : Hash function

- Add a layer of post-quantum security rather than completely changing it
- Improve the security without changing Hyperledger's prototypes



- Signature uses a big integer *FP256BN.BIG* instead of the secret key



- Signature uses a big integer *FP256BN.BIG* instead of the secret key
- Format memory to add GeMSS's secret key after the *FP256BN.BIG*

Performances tests

	Hyperledger	GeMSS (128)	MQDSS (120)	GeMSS (192)	GeMSS (256)
Keys generation	0.002 ms	13 ms	1 ms	60 ms	191 ms
Signature	31 ms	220 ms	33 ms*	560 ms	905 ms
Signature verification	60 ms	0.05 ms	22 ms	0.2 ms	0.5 ms

	Hyperledger + GeMSS 128	Hyperledger + MQDSS	Hyperledger + GeMSS 192	Hyperledger + GeMSS 256
Keys generation	14 ms	1.1 ms	66 ms	207 ms
Signature	227 ms	61 ms*	619 ms	1180 ms
Signature verification	60 ms	85 ms	62 ms	62 ms

Measures on *Intel(R) Core(TM) i5-7600K CPU @ 3.80GHz*

** Time taken over a few runs*

1 2 3

	GeMSS 128	MQDSS	GeMSS 192	GeMSS 256
Keys generation	434 KB	0.288 KB	1351 KB	3137 KB
Signature	0.232 KB	49.288 KB	0.264 KB	0.288 KB
Signature verification	96 B	96 B	96 B	96 B

Memory consumption for one function call with a 40 bytes message

1 2 3

	GeMSS 128	MQDSS	GeMSS 192	GeMSS 256
Keys generation	434 KB	0.288 KB	1351 KB	3137 KB
Signature	0.232 KB	49.288 KB	0.264 KB	0.288 KB
Signature verification	96 B	96 B	96 B	96 B

Memory consumption for one function call with a 40 bytes message

- MQDSS is post-quantum resilient and faster than Hyperledger

1 2 3

	GeMSS 128	MQDSS	GeMSS 192	GeMSS 256
Keys generation	434 KB	0.288 KB	1351 KB	3137 KB
Signature	0.232 KB	49.288 KB	0.264 KB	0.288 KB
Signature verification	96 B	96 B	96 B	96 B

Memory consumption for one function call with a 40 bytes message

- MQDSS is post-quantum resilient and faster than Hyperledger
- Signatures are stored in blocks so their sizes have to be taken into account

1 2 3

	GeMSS 128	MQDSS	GeMSS 192	GeMSS 256
Keys generation	434 KB	0.288 KB	1351 KB	3137 KB
Signature	0.232 KB	49.288 KB	0.264 KB	0.288 KB
Signature verification	96 B	96 B	96 B	96 B

Memory consumption for one function call with a 40 bytes message

- MQDSS is post-quantum resilient and faster than Hyperledger
- Signatures are stored in blocks so their sizes have to be taken into account
- GeMSS adapted for Blockchains : Large quantity of verifications

	GeMSS 128	MQDSS	GeMSS 192	GeMSS 256
Keys generation	434 KB	0.288 KB	1351 KB	3137 KB
Signature	0.232 KB	49.288 KB	0.264 KB	0.288 KB
Signature verification	96 B	96 B	96 B	96 B

Memory consumption for one function call with a 40 bytes message

- MQDSS is post-quantum resilient and faster than Hyperledger
- Signatures are stored in blocks so their sizes have to be taken into account
- GeMSS adapted for Blockchains : Large quantity of verifications
- Memory allocation for the verification is the same and depends on the message's size

Remaining tasks

- Finalize the implementation
  - Adapt the hybrid-cryptography model to the application
- Test a real use case
  - Test over a broad network
  - Perform multiple transactions
- Incorporate other post-quantum signature algorithms

Conclusion

1 2 3

Conclusion

- Understand how CGO works

Conclusion

- Understand how CGO works
- Implement our functions to call GeMSS functions

Conclusion

- Understand how CGO works
- Implement our functions to call GeMSS functions
- Modify structures to take into account those new functions and fields

Conclusion

- Understand how CGO works
- Implement our functions to call GeMSS functions
- Modify structures to take into account those new functions and fields
- The drawbacks on performances are small

Conclusion

- Understand how CGO works
- Implement our functions to call GeMSS functions
- Modify structures to take into account those new functions and fields
- The drawbacks on performances are small
- Changing to post-quantum signature algorithm is already possible and usable even in current applications

References

- **[1]** Christopher Wolf and Bart Preneel. *Taxonomy of Public Key Schemes Based on the Problem of Multivariate Quadratic Equations*. 2005.
- **[2]** Lov K. Grover. *A fast quantum mechanical algorithm for database search*. May 29, 1996.
- **[3]** Peter W. Shor. *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer.* Oct. 1997
- **[4]** C. P. Schnorr. *Efficient Identification and Signatures for Smart Cards*. 1990
- **[5]** Antoine Casanova, Jean-Charles Faugère, Gilles Macario-Rat, Jacques Patarin, Ludovic Perret, and Jocelyn Ryckeghem. *GeMSS: A Great Multivariate Short Signature.* 2017
- **[6]** Ming-Shing Chen, Andreas Hulsing, Joost Rijneveld, Simona Samardjiska, and Peter Schwabe. *From 5-pass MQ-based identification to MQ-based signatures.* 2016
- **[7]** Neal Koblitz, Alfred Menezes, and Scott Vanstone. *The State of Elliptic Curve Cryptography*