Public-key signature scheme with reduced hardware trust

Albert Lu and Andrew Carratu

Mentors:
Jules Drean and Sacha Servan-Schreiber

Table of Contents

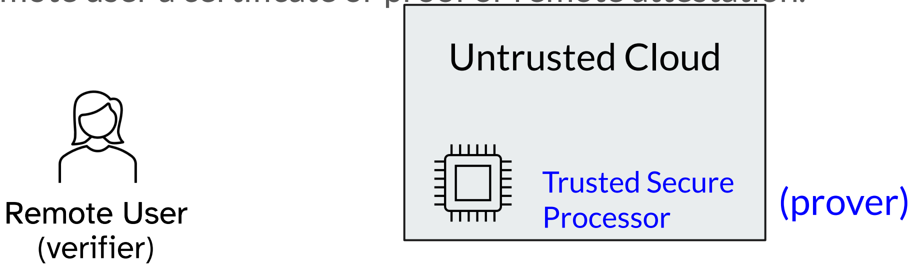
Remote Attestation

Setup:

A remote user wants to perform some sensitive computation on an untrusted computer in the cloud.

More Specifically:

- A "verifier" wants to verify that a "prover" is not compromised i.e. doesn't contains malicious code.
- The untrusted device sends the remote user a certificate or proof or remote attestation.



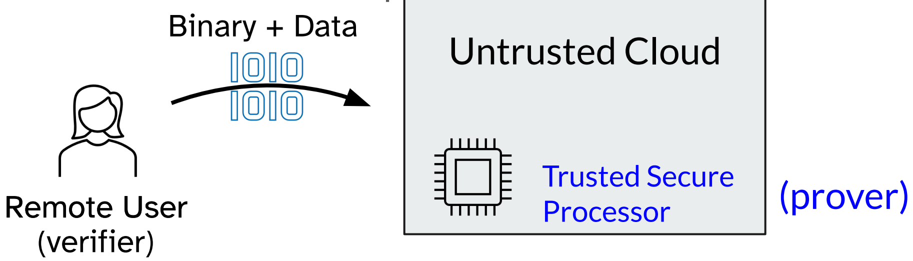
Remote Attestation

Setup:

A remote user wants to perform some sensitive computation on an untrusted computer in the cloud.

More Specifically:

- A “verifier” wants to verify that a “prover” is not compromised i.e. doesn't contains malicious code.
- The untrusted device sends the remote user a certificate or proof or remote attestation.



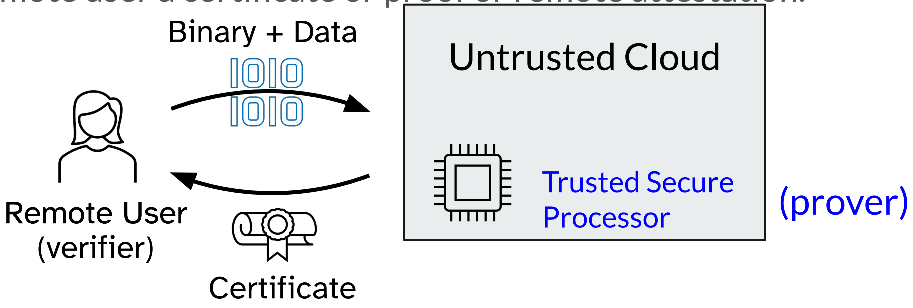
Remote Attestation

Setup:

A remote user wants to perform some sensitive computation on an untrusted computer in the cloud.

More Specifically:

- A "verifier" wants to verify that a "prover" is not compromised i.e. doesn't contains malicious code.
- The untrusted device sends the remote user a certificate or proof or remote attestation.



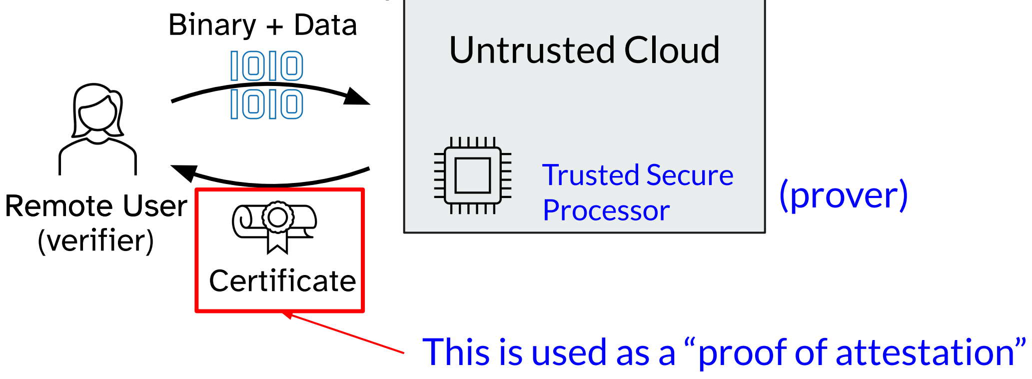
Remote Attestation

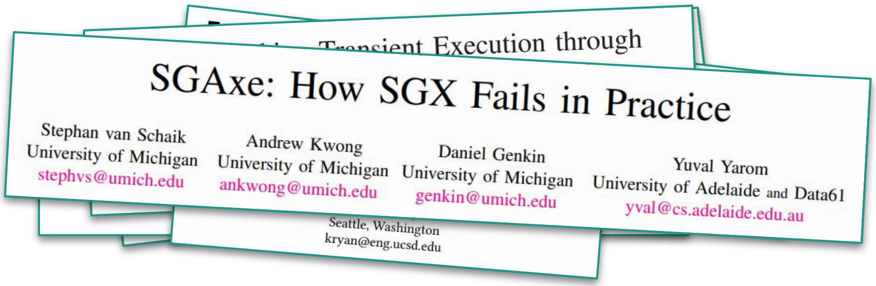
Setup:

A remote user wants to perform some sensitive computation on an untrusted computer in the cloud.

More Specifically:

- A “verifier” wants to verify that a “prover” is not compromised i.e. doesn't contains malicious code.
- The untrusted device sends the remote user a certificate or proof or remote attestation.




SGAxe: How SGX Fails in Practice

Stephan van Schaik
University of Michigan
stephvs@umich.edu

Andrew Kwong
University of Michigan
ankwong@umich.edu

Daniel Genkin
University of Michigan
genkin@umich.edu

Yuval Yarom
University of Adelaide and Data61
yval@cs.adelaide.edu.au

Seattle, Washington
kryan@eng.ucsd.edu

- In recent years, the security of remote attestation schemes has been compromised.
- Most attacks target the hardware (microarchitectural side channels and transient execution attacks).
- These attacks steal the secret key used to **sign** the certificate.

Hardware Vulnerabilities and Side Channels

- Systems are **not secure** if an attacker can steal secret keys.
- The hardware resources (processors, memory etc...) are **shared between several programs**.
- One program might be able to exploit shared resources to spy on another and **steal secret keys**.

These are called side channels:

Real life example: **When you watch a movie on your computer and it freezes...**

... you can guess someone else in the house is using the internet connection!

Similarly, an attacker program can observe the ressources it shares with a victim and infer secrets!

The introduction of the Spectre (transient-execution attack) make these attacks even worse!

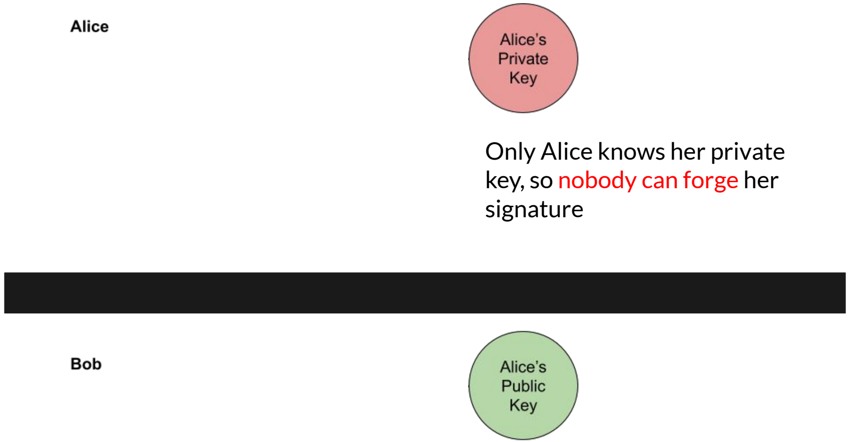
Conclusion: We need to change our trust assumptions on the hardware.

Digital Signatures

- Family of cryptographic algorithms used to prove the authenticity of a message.
- Some schemes use a key pair with a private key (to sign) and a public key (to verify the signature).

Digital Signatures

- Family of cryptographic algorithms used to prove the authenticity of a message.
- Some schemes use a key pair with a private key (to sign) and a public key (to verify the signature).

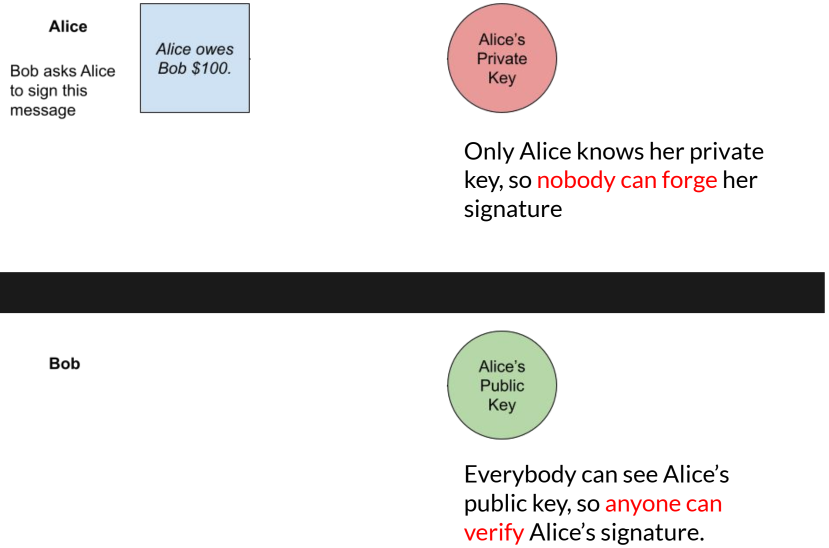


Only Alice knows her private key, so nobody can forge her signature

Everybody can see Alice's public key, so anyone can verify Alice's signature.

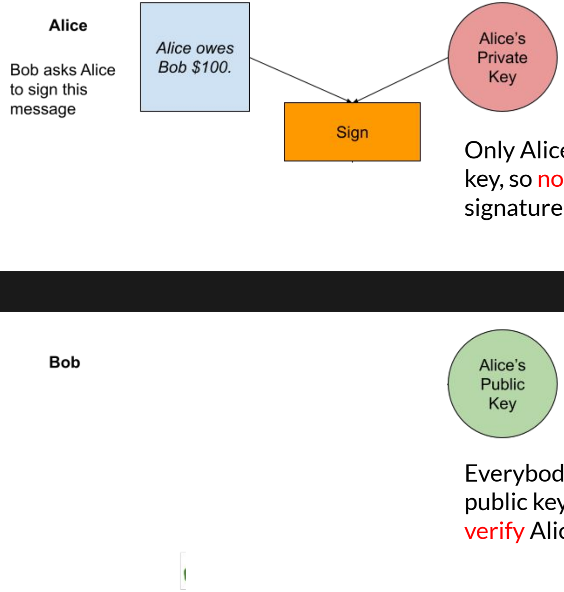
Digital Signatures

- Family of cryptographic algorithms used to prove the authenticity of a message.
- Some schemes use a key pair with a private key (to sign) and a public key (to verify the signature).



Digital Signatures

- Family of cryptographic algorithms used to prove the authenticity of a message.
- Some schemes use a key pair with a private key (to sign) and a public key (to verify the signature).

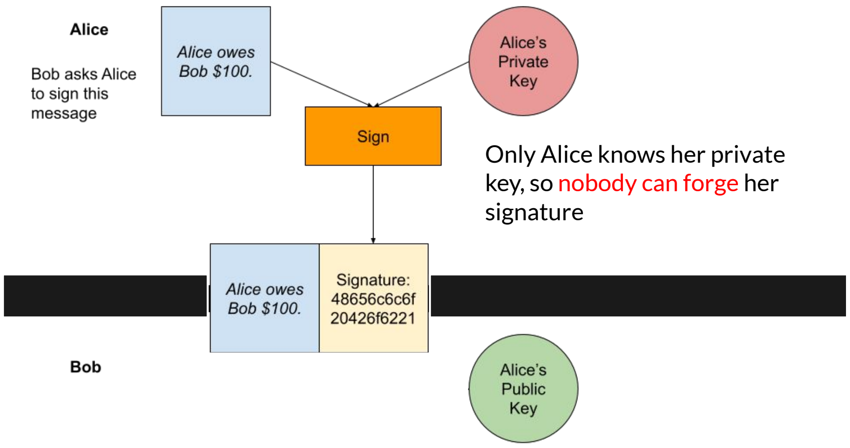


Only Alice knows her private key, so nobody can forge her signature

Everybody can see Alice's public key, so anyone can verify Alice's signature.

Digital Signatures

- Family of cryptographic algorithms used to prove the authenticity of a message.
- Some schemes use a key pair with a private key (to sign) and a public key (to verify the signature).

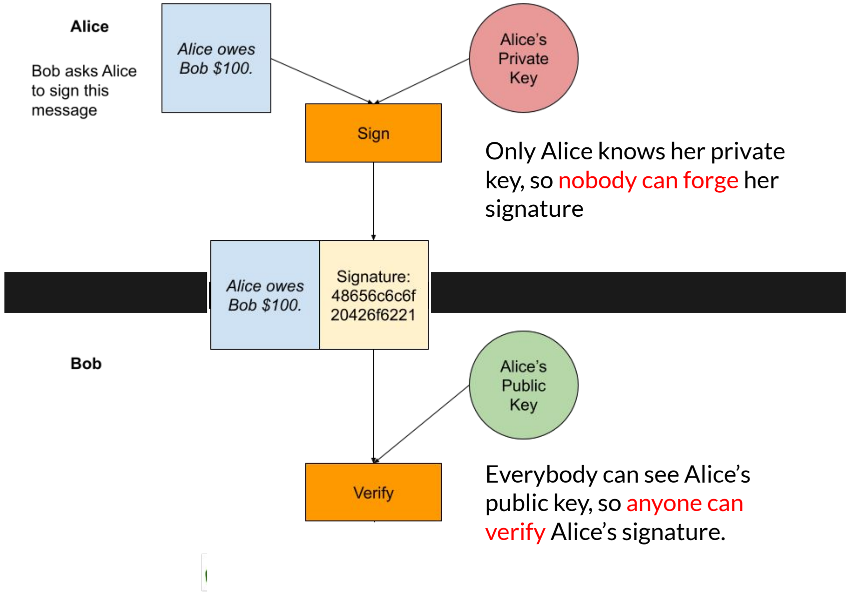


Only Alice knows her private key, so nobody can forge her signature

Everybody can see Alice's public key, so anyone can verify Alice's signature.

Digital Signatures

- Family of cryptographic algorithms used to prove the authenticity of a message.
- Some schemes use a key pair with a private key (to sign) and a public key (to verify the signature).

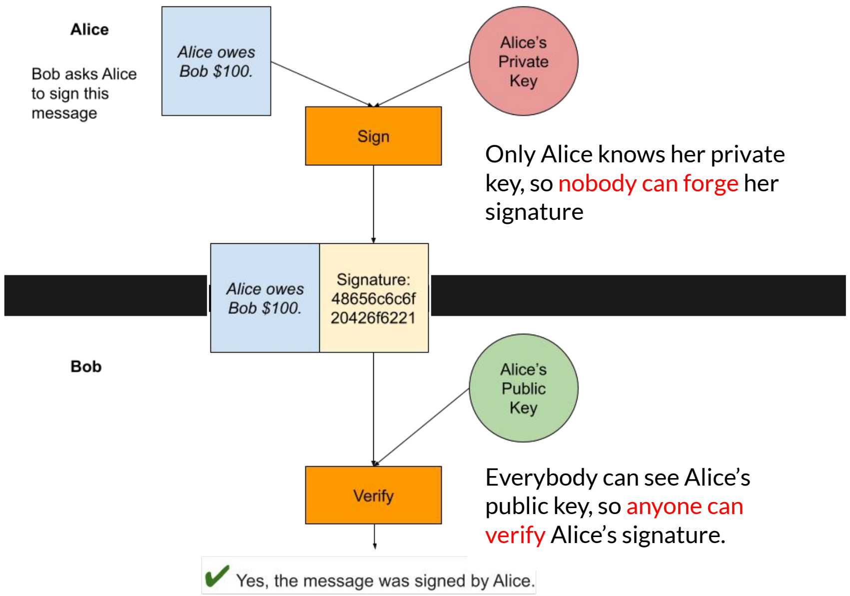


Only Alice knows her private key, so nobody can forge her signature

Everybody can see Alice's public key, so anyone can verify Alice's signature.

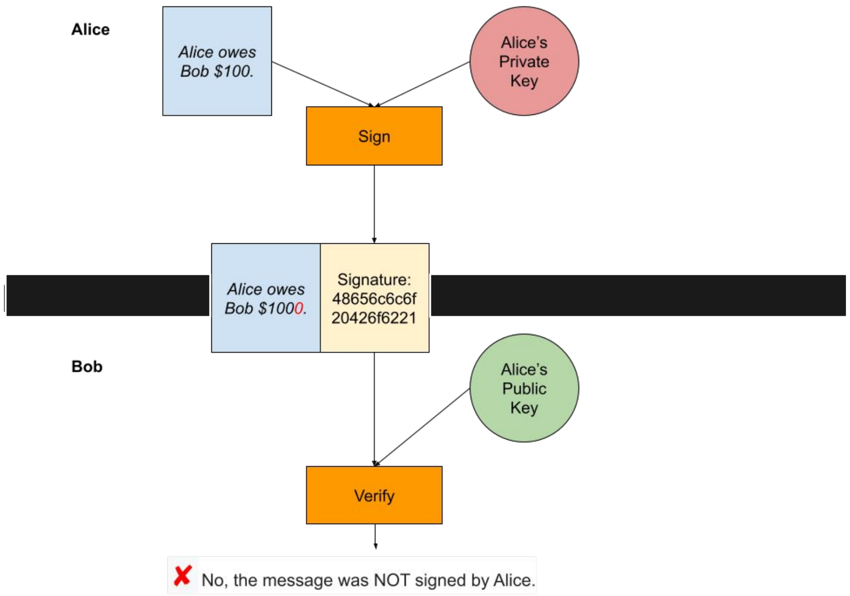
Digital Signatures

- Family of cryptographic algorithms used to prove the authenticity of a message.
- Some schemes use a key pair with a private key (to sign) and a public key (to verify the signature).



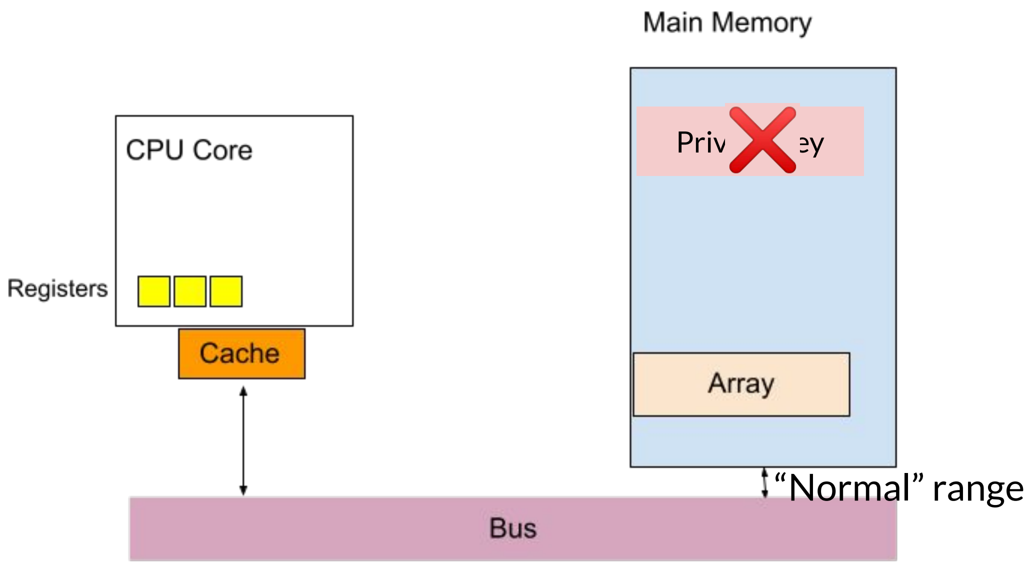
Digital Signatures: Forgery Detection

- What if Bob modified the message?



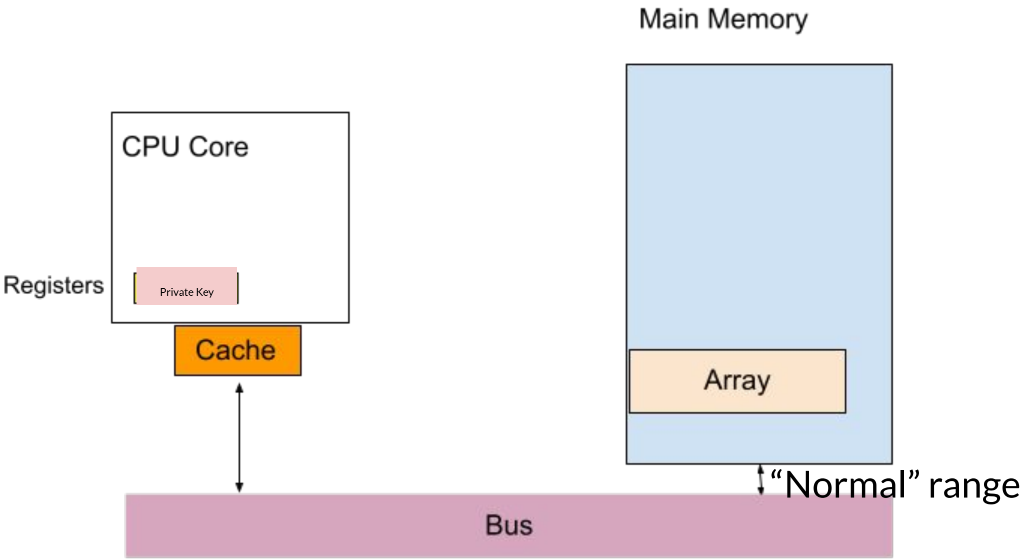
How to make digital signatures with minimal trust?

- Contribution I: Limit shared hardware resources
- Contribution II: Keep all secrets in the CPU registers



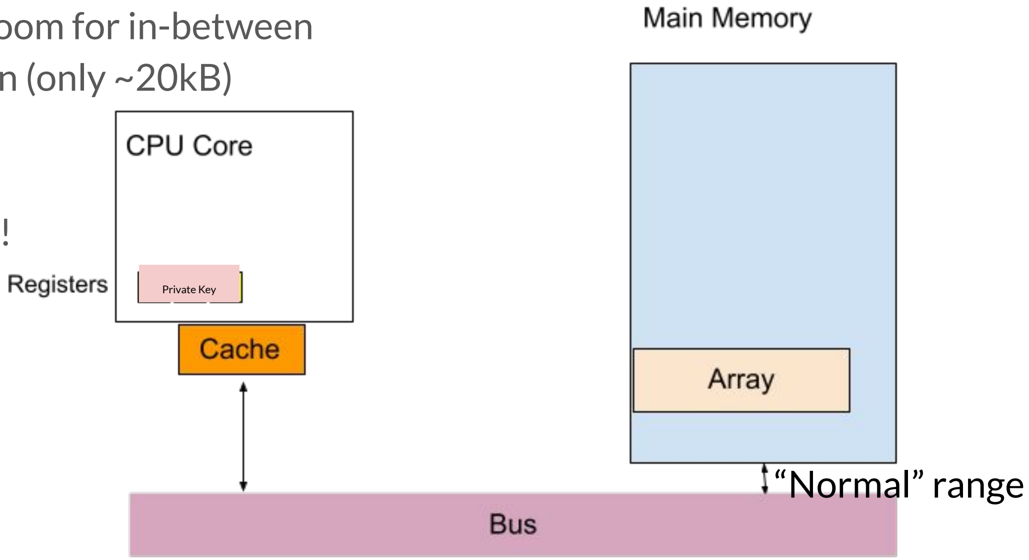
How to make digital signatures with minimal trust?

- Contribution I: Limit shared hardware resources
- Contribution II: Keep all secrets in the CPU registers



How to make digital signatures with minimal trust?

- Contribution I: Limit shared hardware resources
- Contribution II: Keep all secrets in the CPU registers
- Challenges
  - Very little room for in-between computation (only ~20kB)
- We need a lightweight signature scheme!



Digital Signature: Lamport Signature Scheme

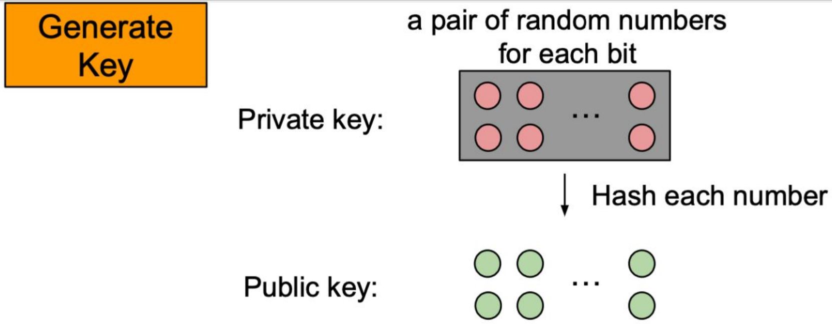
First Public Key Digital Signature Algorithm!

For each bit of the message to sign (256 bits):

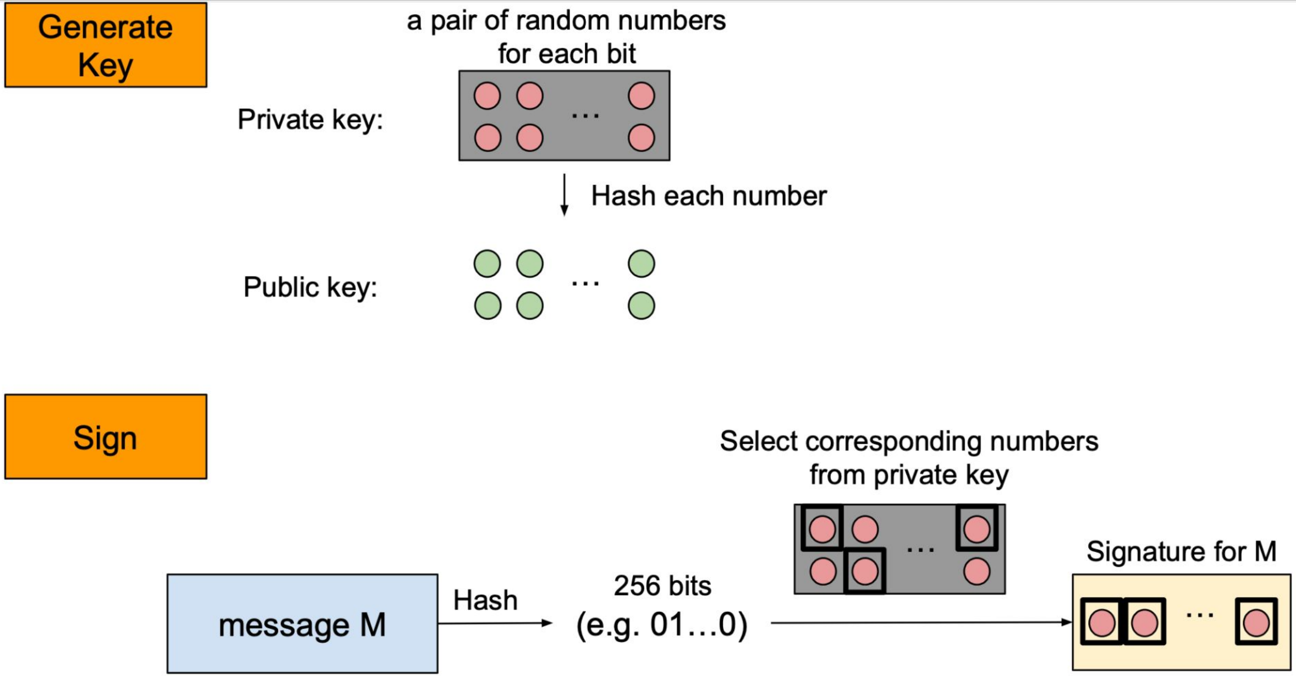
We generate 2 random 128-bit number, one to encode 0 and one to encode 1.
0: 53285a2d862e7d9b13bbf416bb4a09e3 →
1:
These are one element of the private key.

We can generate an element of the public key by hashing.
0: H(53285a2d862e7d9b13bbf416bb4a09e3) = H( )
= c21c9b4aa082bdace250f85db5b6e1b8db1f0262cc5afe8dbb6b4d9e989e8758 →
1: H( ) =

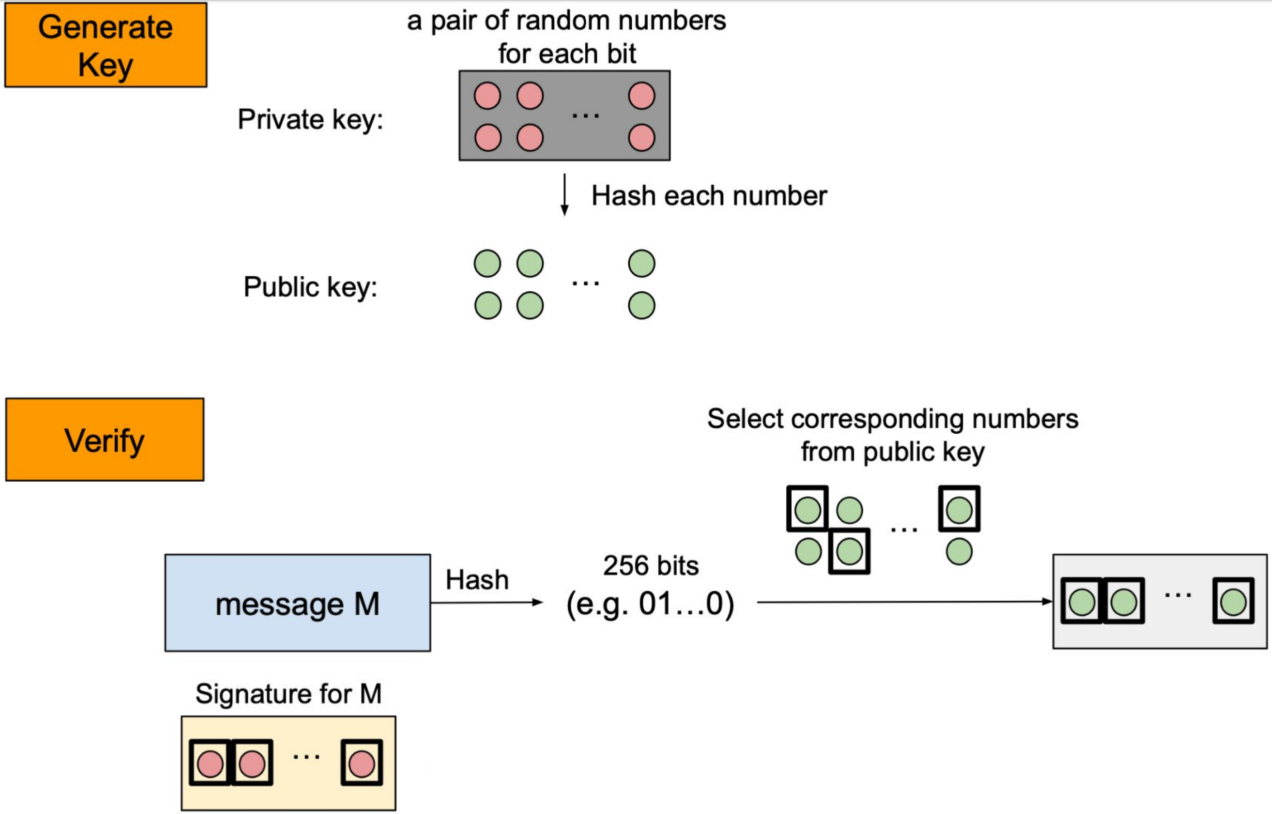
Digital Signature: Lamport Signature Scheme



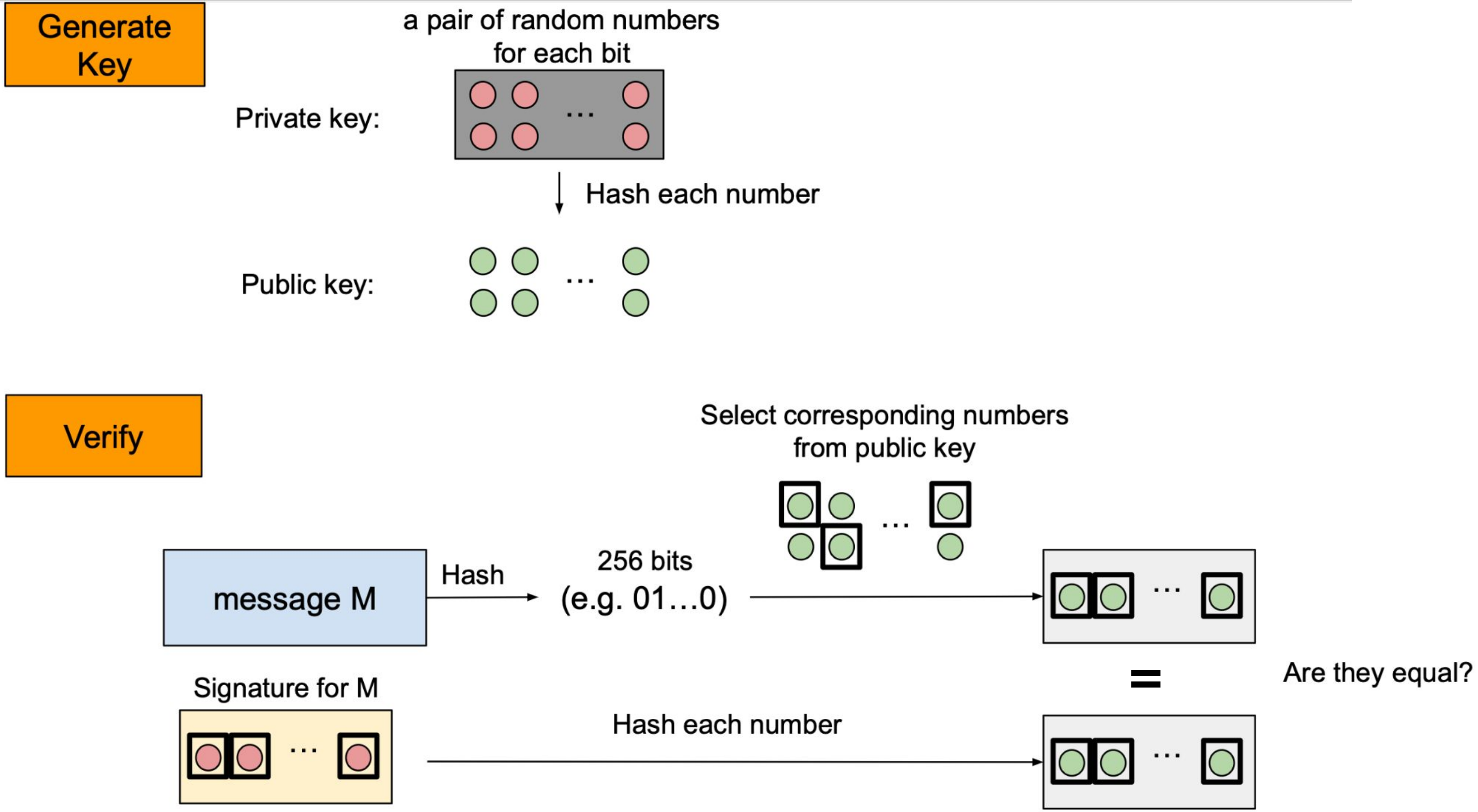
Digital Signature: Lamport Signature Scheme



Digital Signature: Lamport Signature Scheme



Digital Signature: Lamport Signature Scheme



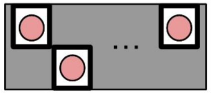
Limitations of Lamport

One Time Usage: **a private key may only be used once**!!

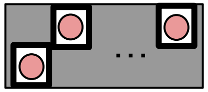
Each signature reveal part of the key ->

an attacker could sign new unseen messages by reconstructing the key!

Msg 1



Msg 2

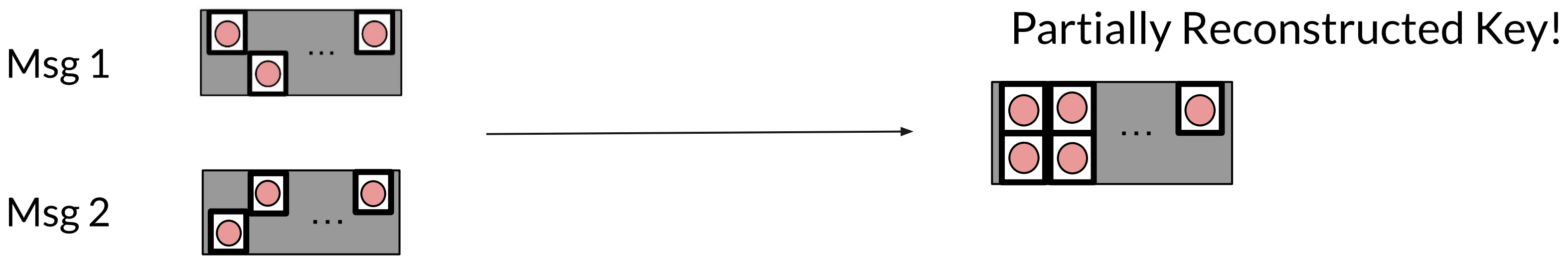


Limitations of Lamport

One Time Usage: **a private key may only be used once**!!

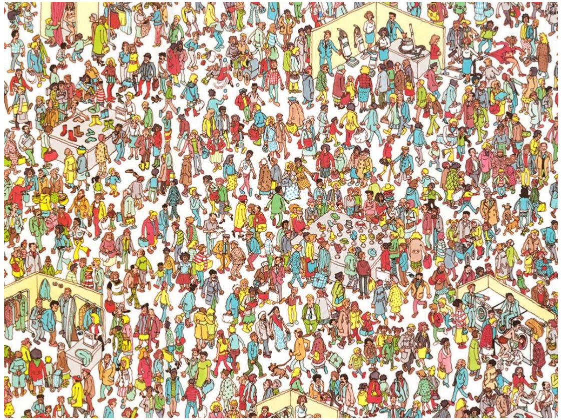
Each signature reveal part of the key ->

an attacker could sign new unseen messages by reconstructing the key!

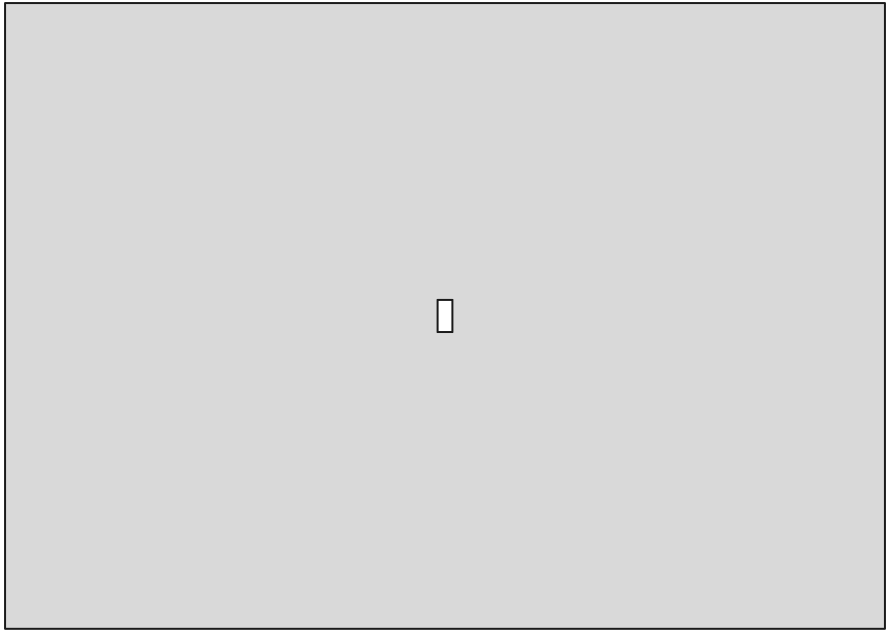


Zero Knowledge Proof-of-Knowledge

- Can we "sign" a message without revealing any of the private key values?
- Prove that we know the value of a secret "s" without revealing the secret.
- Example: Where's Waldo?

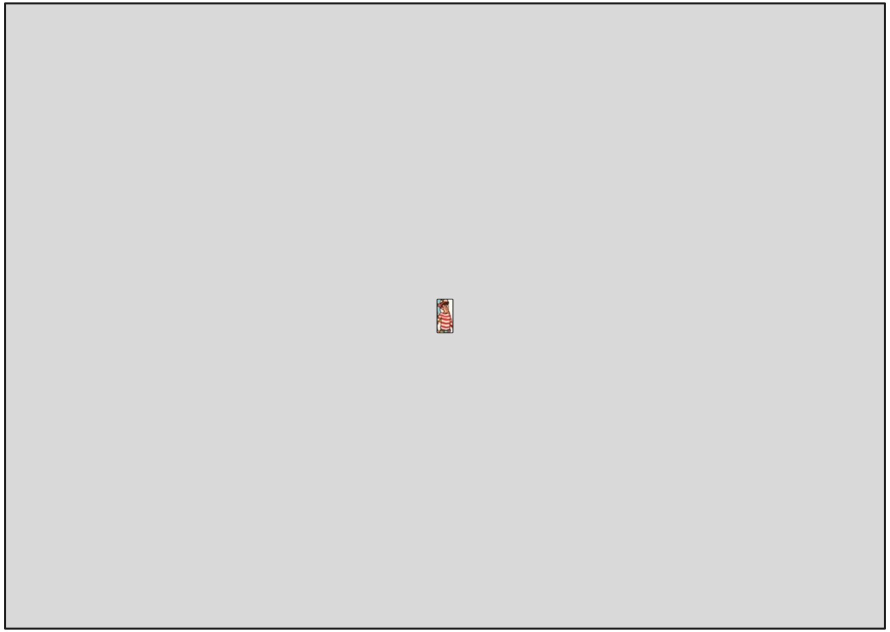


We have a blank canvas with a hole.



We have a blank canvas with a hole.

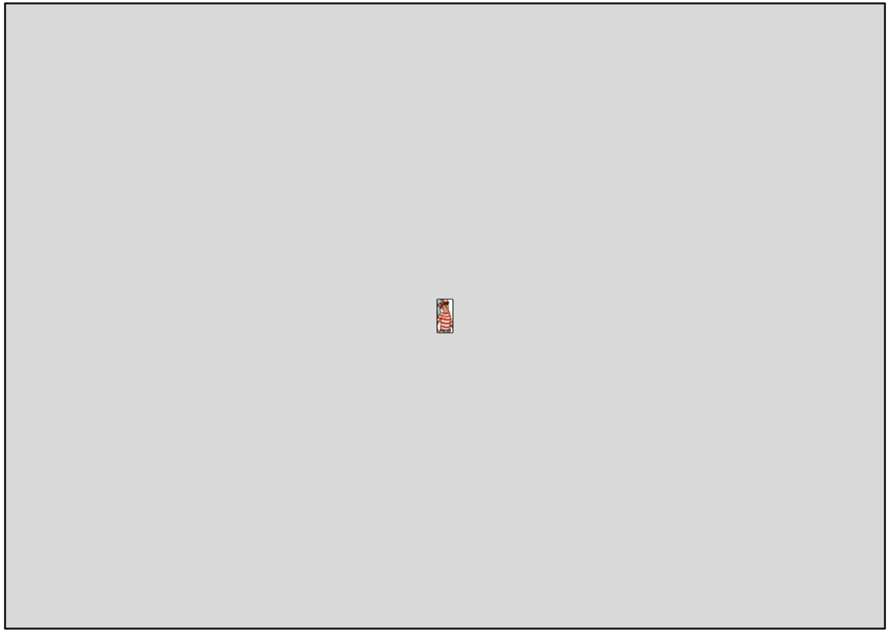
We position the picture behind the canvas so Waldo can be seen through the hole!



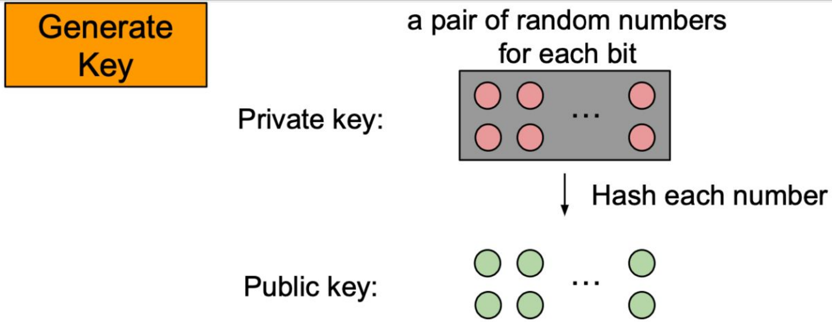
We have a blank canvas with a hole.

We position the picture behind the canvas so Waldo can be seen through the hole!

Someone can verify that we know where Waldo is, but we are not revealing Waldo’s exact location.

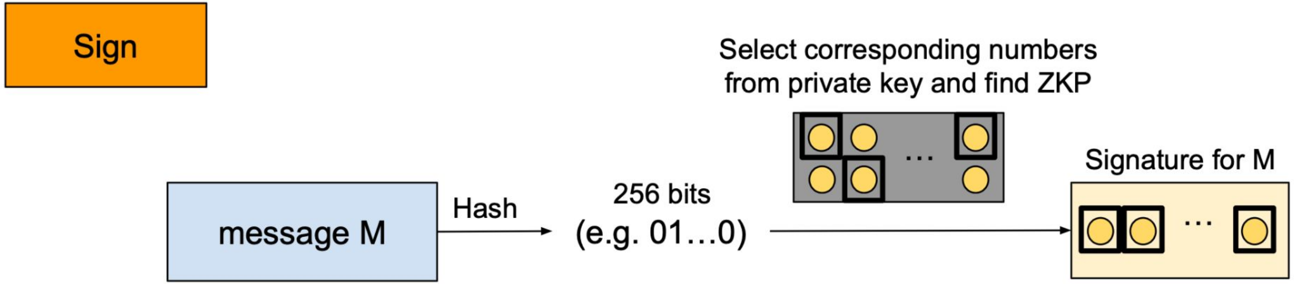


Digital Signature: Lamport Signature Scheme + Zero Knowledge



For each bit of the message, we want to prove we know such that H( ) = , but without revealing the value of .

We can use zero-knowledge proof of that! (represented with)

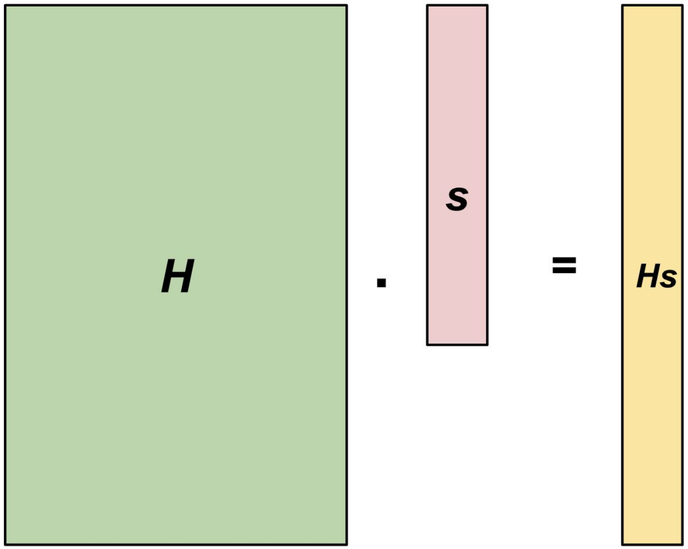


Assumptions we are considering for Zero-Knowledge Proof Scheme

- Discrete Logarithm & Schnorr
- Rabin one-way-function & square root modulo N
- Dual of Learning Parity with Noise (dual-LPN) & Stern ZKP

Dual of Learning Parity with Noise (dual-LPN)

- Assumption that given (H, Hs), it is “hard” to find s, where
  - H is an (n x m) bit matrix
  - s is a m-length random bit vector with hamming weight m/10 (sparse)



Stern's ZKP

- Prover picks y, a m-length random bit vector, and a permutation σ of size m
  - Commitment 1: σ || Hy
  - Commitment 2: σ (y)
  - Commitment 3: σ(y $\oplus$ s)
- Verifier picks a random bit b in {0, 1, 2}, and Prover opens the commitments as follows:
  - If b = 0, it opens commitment to σ (y) by giving (y and σ)
  - If b = 1, it opens (y $\oplus$ s)
  - If b = 2, it opens σ(y) and σ(s)
- Verifier verifies that
  - If b = 0, it verifies commitments (1), (2)
  - If b = 1, it verifies (1), (3) and that H*(y $\oplus$ s) $\oplus$ H*(s) = H(y)
  - If b = 2, it verifies (2), (3) and that σ(s) has correct hamming weight

Next Steps

- Designing our signature scheme
- Implementing the signature scheme
- Performance evaluation if the signature scheme is fast enough

Acknowledgements

Our mentors

Jules Drean

Sacha Servan-Schreiber

MIT PRIMES organizers for making this possible!

Thank you!