Code-Based Cryptography

Other Cryptographic Constructions Relying on Coding Theory

Matthieu Finiasz

informatiques mathématiques Inria

Code-Based Cryptography

1. Error-Correcting Codes and Cryptography
2. McEliece Cryptosystem
3. Message Attacks (ISD)
4. Key Attacks
5. **Other Cryptographic Constructions Relying on Coding Theory**

5. Other Cryptographic Constructions Relying on Coding Theory

- **Code-Based Digital Signatures**
- The Courtois-Finiasz-Sendrier (CFS) Construction
- Attacks against the CFS Scheme
- Parallel-CFS
- Stern’s Zero-Knowledge Identification Scheme
- An Efficient Provably Secure One-Way Function
- The Fast Syndrome-Based (FSB) Hash Function

Digital Signatures

A digital signature is meant to replace a standard “paper signature” on a digital document.

Digital Signatures

A digital signature is meant to replace a standard “paper signature” on a digital document.

Only one person can create it
$\longrightarrow$ ties the signer’s identity to a document

Digital Signatures

A digital signature is meant to replace a standard “paper signature” on a digital document.

Only one person can create it
$\rightarrow$ ties the signer’s identity to a document

Everyone can verify it
$\rightarrow$ repudiation is impossible

Digital Signatures

A digital signature is meant to replace a standard “paper signature” on a digital document.

Only one person can create it
→ ties the signer’s identity to a document

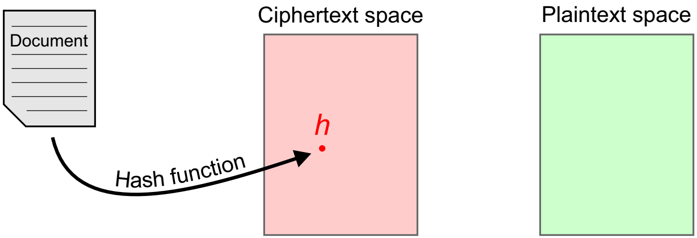
Everyone can verify it
→ repudiation is impossible

This is the “opposite” of the encryption operation in a public key scheme where:

- anyone can encrypt
- only one person can decrypt the resulting ciphertext

→ Digital signatures often use a decryption operation.

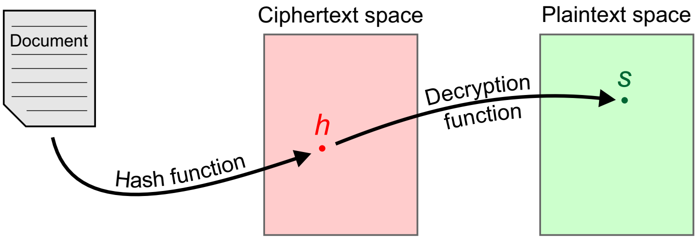
Implementing a Digital Signature



The signer hashes the document into an element of the ciphertext space using a public cryptographic hash function:

- allows to sign documents of arbitrary length
- ties the hash/ciphertext h to the document

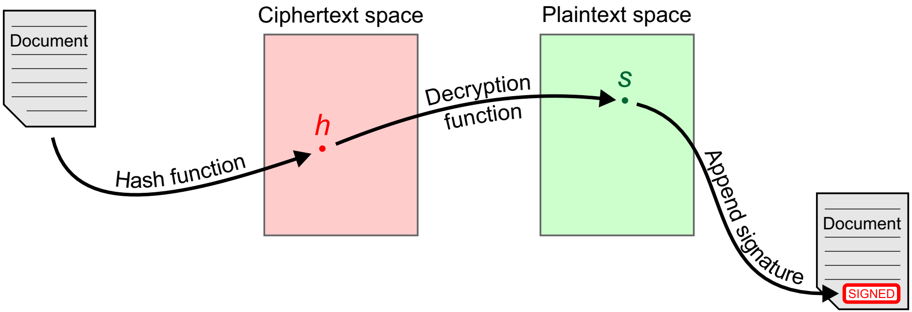
Implementing a Digital Signature



Then, the signer decrypts *h* into a plaintext *s*.

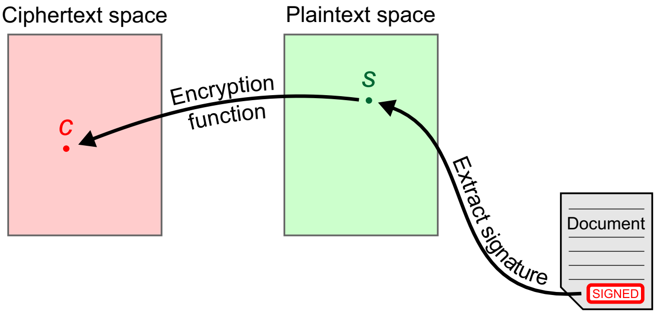
- requires the knowledge of the signer's secret key
- the plaintext *s* is the signature

Implementing a Digital Signature



The signer simply appends the signature to the document.

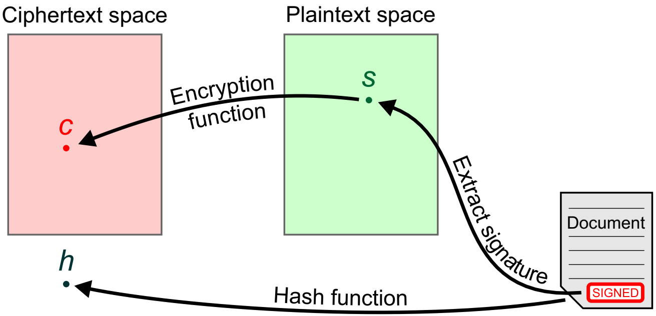
Implementing a Digital Signature



The verifier starts by extracting the signature *s* and re-encrypting it into a ciphertext *c*:

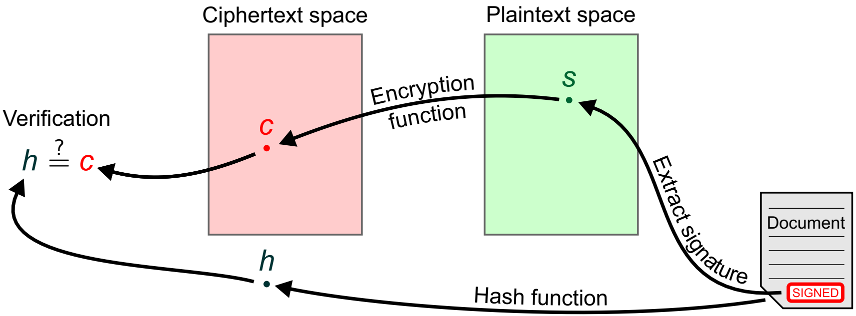
- only the public key of the signer is needed here
- encryption must be deterministic

Implementing a Digital Signature



The verifier also re-computes the hash h of the document.

Implementing a Digital Signature



The signature is considered valid if both the hash h and the ciphertext c are equal.

McEliece or Niederreiter?

The McEliece scheme:

- converts the plaintext into a message and encodes it
- adds some random errors to it

Problems:

- encryption is not deterministic
- similar ciphertexts can correspond to the same plaintext

McEliece or Niederreiter?

The McEliece scheme:

- converts the plaintext into a message and encodes it
- adds some random errors to it

Problems:

- encryption is not deterministic
- similar ciphertexts can correspond to the same plaintext

The Niederreiter scheme:

- embeds the plaintext into an error pattern
- computes its syndrome

$\rightarrow$ encryption is deterministic

The Niederreiter scheme is used!

The Problem with Code-Based Cryptosystems

The hash-and-sign method we presented works well with most public key cryptosystems, but:

- it requires to know the ciphertext space
- it must be possible to hash onto the ciphertext space

The Problem with Code-Based Cryptosystems

The hash-and-sign method we presented works well with most public key cryptosystems, but:

- it requires to know the ciphertext space
- it must be possible to hash onto the ciphertext space

For example, for RSA signatures:

- the ciphertext space is $[1, N-1]$
- any integer in this interval is a valid ciphertext

  $\rightarrow$ one simply needs to hash onto a uniformly distributed integer range

The Problem with Code-Based Cryptosystems

The hash-and-sign method we presented works well with most public key cryptosystems, but:

- it requires to know the ciphertext space
- it must be possible to hash onto the ciphertext space

However, for cryptosystem like those of McEliece and Niederreiter:

- we know a space containing the ciphertexts:
  words of length n, syndromes of length $n - k$

The Problem with Code-Based Cryptosystems

The hash-and-sign method we presented works well with most public key cryptosystems, but:

- it requires to know the ciphertext space
- it must be possible to hash onto the ciphertext space

However, for cryptosystem like those of McEliece and Niederreiter:

- we know a space containing the ciphertexts:
  words of length n, syndromes of length $n-k$
- the exact image of the encryption function is unknown:
  the set of decodable words/syndromes
  $\rightarrow$ deciding if a word/syndrome is decodable is hard

The Problem with Code-Based Cryptosystems

The hash-and-sign method we presented works well with most public key cryptosystems, but:

- it requires to know the ciphertext space
- it must be possible to hash onto the ciphertext space

However, for cryptosystem like those of McEliece and Niederreiter:

- we know a space containing the ciphertexts:
  words of length n, syndromes of length $n-k$
- the exact image of the encryption function is unknown:
  the set of decodable words/syndromes
  → deciding if a word/syndrome is decodable is hard
- hashing onto valid ciphertexts is impossible

5. Other Cryptographic Constructions Relying on Coding Theory

- Code-Based Digital Signatures
- **The Courtois-Finiasz-Sendrier (CFS) Construction**
- Attacks against the CFS Scheme
- Parallel-CFS
- Stern's Zero-Knowledge Identification Scheme
- An Efficient Provably Secure One-Way Function
- The Fast Syndrome-Based (FSB) Hash Function