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

Stern's Zero-Knowledge Identification Scheme

Identification Scheme

Allows a prover to prove his identity to a verifier.

Zero-Knowledge Protocol

Interactive protocol where one proves the knowledge of something, without revealing any information about it.

Stern's Zero-Knowledge Identification Scheme

Identification Scheme

Allows a prover to prove his identity to a verifier.

Zero-Knowledge Protocol

Interactive protocol where one proves the knowledge of something, without revealing any information about it.

Stern's Scheme, invented in 1993:

- its security relies on the Syndrome Decoding problem
- it uses a random binary matrix
  → no need to hide a trap
- like other identification schemes, it can be converted into a signature scheme

Stern's Zero-Knowledge Identification Scheme

System parameters:

- A public $n \times r$ binary matrix H, a weight w

Stern's Zero-Knowledge Identification Scheme

System parameters:

- A public $n \times r$ binary matrix H, a weight w

Key generation:

- Each user picks a secret binary vector e of length n and Hamming weight w
- He computes $s = H \times e$ and publishes it

Stern's Zero-Knowledge Identification Scheme

System parameters:

- A public $n \times r$ binary matrix H, a weight w

Key generation:

- Each user picks a secret binary vector e of length n and Hamming weight w
- He computes $s = H \times e$ and publishes it

Identification protocol:

- The verifier knows s
- The prover has to prove he knows e such that $s = H \times e$

  → without revealing any information about e

Stern's Zero-Knowledge Identification Scheme

Prover **Verifier**

Pick: $y \in \mathbb{F}_2^n, \sigma$ perm. of $[1, n]$

Stern's Zero-Knowledge Identification Scheme

Prover		Verifier
Pick: $y \in \mathbb{F}_2^n$, σ perm. of $[1, n]$		
Compute: $c_0 = \text{Hash}(\sigma \| H \times y)$		
$c_1 = \text{Hash}(\sigma(y))$		
$c_2 = \text{Hash}(\sigma(y \oplus e))$	$\xrightarrow{c_0, c_1, c_2}$	Store the commitments

Stern's Zero-Knowledge Identification Scheme

Prover		**Verifier**
Pick: $y \in \mathbb{F}_2^n, \sigma$ perm. of $[1, n]$		
Compute: $c_0 = \text{Hash}(\sigma \vert\vert H \times y)$		
$c_1 = \text{Hash}(\sigma(y))$		
$c_2 = \text{Hash}(\sigma(y \oplus e))$	$\xrightarrow{c_0, c_1, c_2}$	Store the commitments
	$\xleftarrow{b}$	Pick: $b \in \{0, 1, 2\}$

Stern's Zero-Knowledge Identification Scheme

Prover		Verifier
Pick: $y \in F_2^n, \sigma$ perm. of $[1, n]$		
Compute: $c_0 = \text{Hash}(\sigma \| H \times y)$		
$c_1 = \text{Hash}(\sigma(y))$		
$c_2 = \text{Hash}(\sigma(y \oplus e))$	$\xrightarrow{c_0, c_1, c_2}$	Store the commitments
	$\xleftarrow{b}$	Pick: $b \in \{0, 1, 2\}$
If $b = 0$ reveal info for c_1 and c_2	$\xrightarrow{\sigma(y), \sigma(e)}$	Compute:
		$c_1' = \text{Hash}(\sigma(y))$
		$c_2' = \text{Hash}(\sigma(y) \oplus \sigma(e))$
		Accept if: $c_1' = c_1$ and $c_2' = c_2$

Stern's Zero-Knowledge Identification Scheme

Prover | **Verifier**

Pick: $y \in \mathbb{F}_2^n, \sigma$ perm. of $[1, n]$

Compute: $c_0 = \text{Hash}(\sigma || H \times y)$

$c_1 = \text{Hash}(\sigma(y))$

$c_2 = \text{Hash}(\sigma(y \oplus e))$ $\xrightarrow{c_0, c_1, c_2}$ Store the commitments

$\xleftarrow{b}$ Pick: $b \in \{0, 1, 2\}$

If $b = 1$ reveal info for c_0 and c_2 $\xrightarrow{y \oplus e, \sigma}$ Compute:

$c_0' = \text{Hash}(\sigma || (H \times (y \oplus e)) \oplus s)$

$c_2' = \text{Hash}(\sigma(y \oplus e))$

Accept if: $c_0' = c_0$ and $c_2' = c_2$

Stern's Zero-Knowledge Identification Scheme

Prover		Verifier
Pick: $y \in F_2^n, \sigma$ perm. of $[1, n]$		
Compute: $c_0 = \text{Hash}(\sigma \| H \times y)$		
$c_1 = \text{Hash}(\sigma(y))$		
$c_2 = \text{Hash}(\sigma(y \oplus e))$	$\xrightarrow{c_0, c_1, c_2}$	Store the commitments
	$\xleftarrow{b}$	Pick: $b \in \{0, 1, 2\}$
If $b = 2$ reveal info for c_0 and c_1	$\xrightarrow{y, \sigma}$	Compute:
		$c_0' = \text{Hash}(\sigma \| H \times y)$
		$c_1' = \text{Hash}(\sigma(y))$
		Accept if: $c_0' = c_0$ and $c_1' = c_1$

Stern's Zero-Knowledge Identification Scheme

Prover		**Verifier**
Pick: $y \in \mathbb{F}_2^n$, σ perm. of $[1, n]$		
Compute: $c_0 = \text{Hash}(\sigma \| H \times y)$		
$c_1 = \text{Hash}(\sigma(y))$		
$c_2 = \text{Hash}(\sigma(y \oplus e))$	$\xrightarrow{c_0, c_1, c_2}$	Store the commitments
	$\xleftarrow{b}$	Pick: $b \in \{0, 1, 2\}$
If $b = 2$ reveal info for c_0 and c_1	$\xrightarrow{y, \sigma}$	Compute:
		$c_0' = \text{Hash}(\sigma \| H \times y)$
		$c_1' = \text{Hash}(\sigma(y))$
		Accept if: $c_0' = c_0$ and $c_1' = c_1$

In all three cases, the verifier can verify 2 out of the 3 commitments.

Verification of the Zero-Knowledge Property

When running the protocol, the verifier learns:

- the values of the 3 commitments

  → assuming the hash function is secure, these do not leak any information

Verification of the Zero-Knowledge Property

When running the protocol, the verifier learns:

- the values of the 3 commitments
  → assuming the hash function is secure, these do not leak any information
- depending on the choice of b, one of the following pairs of values:
  - $\sigma(y)$ and $\sigma(e)$
  - $y \oplus e$ and σ
  - y and σ

- y is random, so $\sigma(y)$ gives no information
- $\sigma(e)$ discloses the weight of e, which is always w

Verification of the Zero-Knowledge Property

When running the protocol, the verifier learns:

- the values of the 3 commitments
  → assuming the hash function is secure, these do not leak any information
- depending on the choice of b, one of the following pairs of values:
  - $\sigma(y)$ and $\sigma(e)$
  - $y \oplus e$ and σ
  - y and σ

- y is random, so $y \oplus e$ gives no information
- σ is random and gives no information

Verification of the Zero-Knowledge Property

When running the protocol, the verifier learns:

- the values of the 3 commitments
  → assuming the hash function is secure, these do not leak any information
- depending on the choice of b, one of the following pairs of values:
  - $\sigma(y)$ and $\sigma(e)$
  - $y \oplus e$ and σ
  - y and σ

- y is random and gives no information
- σ is random and gives no information

Security of the Protocol

Again, there are two ways to attack this protocol.

Recovery of the secret:

- similar to decoding attacks on McEliece or signature forgery in CFS
- requires to solve an instance of syndrome decoding

  → a truly random instance, with no trap: both H and e are random

Security of the Protocol

Again, there are two ways to attack this protocol.

Recovery of the secret:

- similar to decoding attacks on McEliece or signature forgery in CFS
- requires to solve an instance of syndrome decoding
  → a truly random instance, with no trap: both H and e are random

Impersonation attacks:

- an attacker executes the protocol with a verifier
  → tries to give answers the verifier will accept
- impossible to give commitments that can be opened for all 3 values of b

Security of the Protocol

Again, there are two ways to attack this protocol.

Recovery of the secret:

- similar to decoding attacks on McEliece or signature forgery in CFS
- requires to solve an instance of syndrome decoding
  → a truly random instance, with no trap: both H and e are random

Impersonation attacks:

- an attacker executes the protocol with a verifier
  → tries to give answers the verifier will accept
- impossible to give commitments that can be opened for all 3 values of b

Without the knowledge of the secret e, the probability of success is at most $\frac{2}{3}$.

Impersonation Attack

An attacker can achieve a probability of impersonation of $\frac{2}{3}$ by choosing any of these 3 constructions:

Choice 1:

- Pick y, σ, and e' of weight w
- Send: $c_0 = \text{Hash}(\sigma || H \times y)$, $c_1 = \text{Hash}(\sigma(y))$, $c_2 = \text{Hash}(\sigma(y \oplus e'))$

If $b = 0$, verify c_1 and c_2	If $b = 1$, verify c_0 and c_2	If $b = 2$, verify c_0 and c_1
Send $\sigma(y)$ and $\sigma(e')$	Problem!	Send y and σ

Impersonation Attack

An attacker can achieve a probability of impersonation of $\frac{2}{3}$ by choosing any of these 3 constructions:

Choice 1:

- Pick y, σ, and e' of weight w
- Send: $c_0 = \text{Hash}(\sigma || H \times y)$, $c_1 = \text{Hash}(\sigma(y))$, $c_2 = \text{Hash}(\sigma(y \oplus e'))$

Choice 2:

- Pick $y \oplus e'$, σ, and e' of weight w
- Send: $c_0 = \text{Hash}(\sigma || H \times (y \oplus e') \oplus s)$, $c_1 = \text{Hash}(\sigma(y))$, $c_2 = \text{Hash}(\sigma(y \oplus e'))$

If $b = 0$, verify c_1 and c_2	If $b = 1$, verify c_0 and c_2	If $b = 2$, verify c_0 and c_1
Send $\sigma(y)$ and $\sigma(e')$	Send $y \oplus e'$ and σ	Problem!

Impersonation Attack

An attacker can achieve a probability of impersonation of $\frac{2}{3}$ by choosing any of these 3 constructions:

Choice 1:

- Pick y, σ, and e' of weight w
- Send: $c_0 = \text{Hash}(\sigma || H \times y)$, $c_1 = \text{Hash}(\sigma(y))$, $c_2 = \text{Hash}(\sigma(y \oplus e'))$

Choice 2:

- Pick $y \oplus e'$, σ, and e' of weight w
- Send: $c_0 = \text{Hash}(\sigma || H \times (y \oplus e') \oplus s)$, $c_1 = \text{Hash}(\sigma(y))$, $c_2 = \text{Hash}(\sigma(y \oplus e'))$

Choice 3:

- Pick y, σ, and e' of heavy weight, such that $H \times e' = s$
- Send: $c_0 = \text{Hash}(\sigma || H \times y)$, $c_1 = \text{Hash}(\sigma(y))$, $c_2 = \text{Hash}(\sigma(y \oplus e'))$

If $b = 0$, verify c_1 and c_2

$\sigma(e')$ is too heavy!

If $b = 1$, verify c_0 and c_2

Send $y \oplus e'$ and σ

If $b = 2$, verify c_0 and c_1

Send y and σ

Reaching a High Security Level

A probability of impersonation of $\frac{2}{3}$ is too high :)

The protocol can simply be iterated:

- run the protocol ℓ times
- if any of the ℓ proofs fails, abort
- if all ℓ iterations can be verified, authentication is successful

→ the final probability of impersonation is $\left(\frac{2}{3}\right)^{\ell}$

52 iterations give a probability of less than 1 in a billion.
137 iterations give a probability of 2^{-80}.

→ around 3 000 bits are exchanged at each iteration.

Conversion to a Signature Scheme

The Fiat-Shamir transform can turn any ZK identification scheme into a signature scheme.

- choose the document D to sign

Conversion to a Signature Scheme

The Fiat-Shamir transform can turn any ZK identification scheme into a signature scheme.

- choose the document D to sign
- compute the commitments for ℓ iterations of the protocol
  → note T the “transcript” containing these ℓ triples (c_0, c_1, c_2)

Conversion to a Signature Scheme

The Fiat-Shamir transform can turn any ZK identification scheme into a signature scheme.

- choose the document D to sign
- compute the commitments for ℓ iterations of the protocol
  $\rightarrow$ note T the "transcript" containing these ℓ triples (c_0, c_1, c_2)
- compute $h = \text{Hash}(D || T)$

Conversion to a Signature Scheme

The Fiat-Shamir transform can turn any ZK identification scheme into a signature scheme.

- choose the document D to sign
- compute the commitments for ℓ iterations of the protocol
  → note T the "transcript" containing these ℓ triples (c_0, c_1, c_2)
- compute $h = \text{Hash}(D||T)$
- use the bits of h to obtain ℓ values of b, tied to D and T

Conversion to a Signature Scheme

The Fiat-Shamir transform can turn any ZK identification scheme into a signature scheme.

- choose the document D to sign
- compute the commitments for ℓ iterations of the protocol
  → note T the "transcript" containing these ℓ triples (c_0, c_1, c_2)
- compute $h = \mathrm{Hash}(D || T)$
- use the bits of h to obtain ℓ values of b, tied to D and T
- open the commitments corresponding to these b
  → note S the "transcript" containing the opening values

Conversion to a Signature Scheme

The Fiat-Shamir transform can turn any ZK identification scheme into a signature scheme.

- choose the document D to sign
- compute the commitments for ℓ iterations of the protocol
  → note T the "transcript" containing these ℓ triples (c_0, c_1, c_2)
- compute $h = \text{Hash}(D||T)$
- use the bits of h to obtain ℓ values of b, tied to D and T
- open the commitments corresponding to these b
  → note S the "transcript" containing the opening values
- the signature of D is the full transcript $T||S$

The security of the signature is $\left(\frac{2}{3}\right)^{\ell}$
The size of the signature is the full transcript size
→ 50 kB for a security of 2^{80}

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