KU LEUVEN

Ramstake

April 2018

Alan Szepieniec alan.szepieniec@esat.kuleuven.be

KU Leuven, ESAT/COSIC

Original Goal

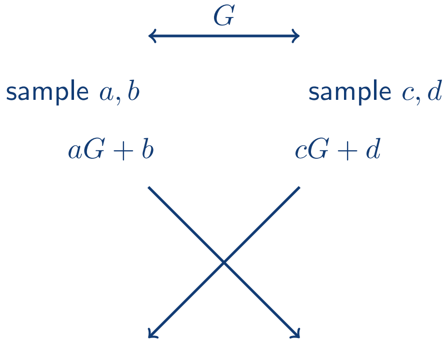
$$\overset{G}{\longleftrightarrow}$$

sample a, b sample c, d

$aG + b$ $cG + d$

$$K_A = a(cG + d) \quad \approx \quad c(aG + d) = K_B$$

Original Goal



$$K_A = a(cG + d) \quad \approx \quad c(aG + d) = K_B$$

$$\boxed{\sqrt{q} \ll \{a, b, c, d\} \ll q}$$

Original AJPS

Available versions in chronological order

20170530:001542 (posted 30-May-2017 00:15:42 UTC)
A New Public-Key Cryptosystem via Mersenne Numbers
Divesh Aggarwal and Antoine Joux and Anupam Prakash and Miklos Santa

20170530:071730 (posted 30-May-2017 07:17:30 UTC)
A New Public-Key Cryptosystem via Mersenne Numbers
Divesh Aggarwal and Antoine Joux and Anupam Prakash and Miklos Santha

20170530:072202 (posted 30-May-2017 07:22:02 UTC)
A New Public-Key Cryptosystem via Mersenne Numbers
Divesh Aggarwal and Antoine Joux and Anupam Prakash and Miklos Santha

20171206:004144 (posted 06-Dec-2017 00:41:44 UTC)
A New Public-Key Cryptosystem via Mersenne Numbers
Divesh Aggarwal and Antoine Joux and Anupam Prakash and Miklos Santha

20171206:004924 (posted 06-Dec-2017 00:49:24 UTC)
A New Public-Key Cryptosystem via Mersenne Numbers
Divesh Aggarwal and Antoine Joux and Anupam Prakash and Miklos Santha

Retrieve selected version | Go to latest version

[Cryptology ePrint archive]

Original AJPS

- Mersenne prime modulus: $q = 2^n - 1$ with n prime

Original AJPS

- Mersenne prime modulus: $q = 2^n - 1$ with n prime
  - easy arithmetic
  - somewhat Hamming weight preserving

Original AJPS

- Mersenne prime modulus: $q = 2^n - 1$ with n prime
  - easy arithmetic
  - somewhat Hamming weight preserving

- KeyGen:
  - sample f, g of low Hamming weight
  - public key: $f/g \bmod q$
  - secret key: g

Original AJPS

- Mersenne prime modulus: $q = 2^n - 1$ with n prime
  - easy arithmetic
  - somewhat Hamming weight preserving

- KeyGen:
  - sample f, g of low Hamming weight
  - public key: $f/g \bmod q$
  - secret key: g

- Encrypt($m \in \{0, 1\}$):
  - sample a, b of low Hamming weight
  - ciphertext: $(-1)^m \times (\mathsf{pk} \times a + b)$

Original AJPS

- Mersenne prime modulus: $q = 2^n - 1$ with n prime
  - easy arithmetic
  - somewhat Hamming weight preserving

- KeyGen:
  - sample f, g of low Hamming weight
  - public key: $f/g \bmod q$
  - secret key: g

- Encrypt($m \in \{0, 1\}$):
  - sample a, b of low Hamming weight
  - ciphertext: $(-1)^m \times (\mathsf{pk} \times a + b)$

- Decrypt(c):
  - $d \leftarrow cg$
  - return $\begin{cases} 1 & \text{if } d \text{ has high Hamming weight} \\ 0 & \text{if } d \text{ has low Hamming weight} \end{cases}$

Short-and-Sparse

Short-and-Sparse

PDF of a: 

Short-and-Sparse

PDF of a:

PDF of b:

Short-and-Sparse

PDF of a:

PDF of b:

PDF of $a + b$:

Short-and-Sparse

PDF of a:

PDF of b:

PDF of $a + b$:

PDF of $a \times b$:

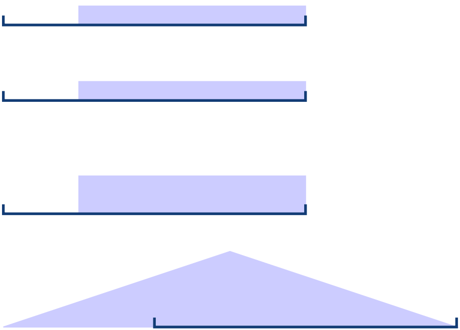
Short-and-Sparse

PDF of a:

PDF of b:

PDF of $a + b$:

PDF of $a \times b$:



Short-and-Sparse

PDF of a:

PDF of b:

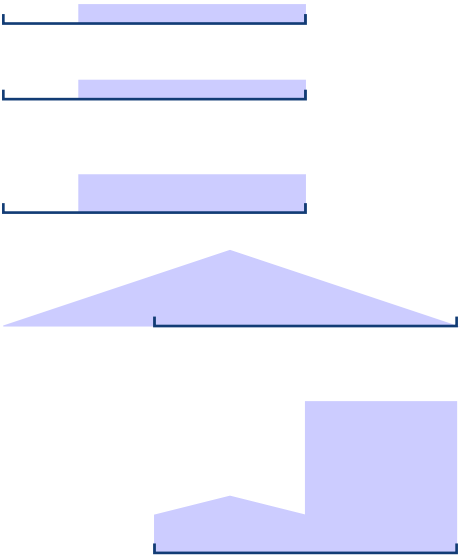
PDF of $a + b$:

PDF of $a \times b$:

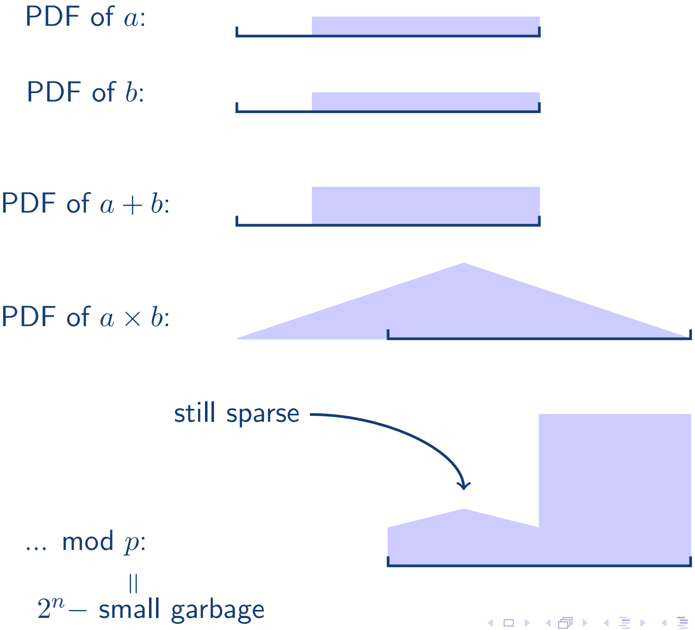
... mod p:

||

$2^n -$ small garbage



Short-and-Sparse



Ramstake Key Agreement

- ~~short-and~~ sparse integers
- ~~pseudo-~~Mersenne prime modulus
- Hamming weight metric

$$G$$

sample a, b sample c, d

$aG + b$ $cG + d$

$$K_A = a(cG + d) \quad \approx \quad c(aG + d) = K_B$$

Ramstake KEM

- KeyGen:
  - sample seed for G
  - sample a, b
  - public key: (seed for G, $aG + b$)
  - secret key: (a ~~, b~~)

Ramstake KEM

- KeyGen:
  - sample seed for G
  - sample a, b
  - public key: (seed for G, $aG + b$)
  - secret key: (a , ~~b~~)

- DetEncaps(pk; s):
  - sample c, d
  - $S_B \leftarrow c(aG + b) + d$
  - ciphertext: $(cG + d$, $\mathcal{E}(s) \oplus \llcorner S_B \lrcorner$, $\mathsf{H}(s))$
  - key: $K = \mathsf{H}(\mathsf{pk} \|$ ~~coins~~ $s)$

- Encaps(pk):
  - sample seed s
  - return DetEncaps(pk, s)

Ramstake KEM

- KeyGen:
  - sample seed for G
  - sample a, b
  - public key: (seed for G, $aG + b$)
  - secret key: (a ~~, b~~)
- DetEncaps(pk; s):
  - sample c, d
  - $S_B \leftarrow c(aG + b) + d$
  - ciphertext: ($cG + d$, $\mathcal{E}(s) \oplus \llcorner S_B \lrcorner$, $\mathsf{H}(s)$)
  - key: $K = \mathsf{H}(\mathsf{pk} \|$ ~~coins~~ $s)$
- Encaps(pk):
  - sample seed s
  - return DetEncaps(pk, s)
- Decaps(sk, ctxt):
  - $S_A \leftarrow a(cG + d)$
  - $s \leftarrow \mathcal{D}(\mathcal{E}(s) \oplus \llcorner S_B \lrcorner \oplus \llcorner S_A \lrcorner)$
  - ctxt', $K \leftarrow$ DetEncaps(pk; s)
  - if ctxt' $\neq$ ctxt, return $\perp$
  - else return K

Ramstake KEM

- KeyGen:
  - sample seed for G
  - sample a, b
  - public key: (seed for G, $aG + b$)
  - secret key: (a ~~, b~~)
- DetEncaps(pk; s):
  - sample c, d
  - $S_B \leftarrow c(aG + b) + d$
  - ciphertext: $(cG + d,$ $\mathcal{E}(s) \oplus \lfloor S_B \rceil$, $\mathsf{H}(s))$
  - key: $K = \mathsf{H}(\text{pk}\|$ ~~coins~~ $s)$
- Encaps(pk):
  - sample seed s
  - return DetEncaps(pk, s)
- Decaps(sk, ctxt):
  - $S_A \leftarrow a(cG + d)$
  - $s \leftarrow \mathcal{D}(\mathcal{E}(s) \oplus \lfloor S_B \rceil \oplus \lfloor S_A \rceil)$
  - ctxt', $K \leftarrow$ DetEncaps(pk; s)
  - if ctxt' $\neq$ ctxt, return $\perp$
  - else return K
- $\mathcal{E}, \mathcal{D}$ — error-correcting code
  - Reed-Solomon over GF(256) with design distance 223
  - capable of correcting 111 bit-errors
  - repeated ν times
  - used for functionality not security

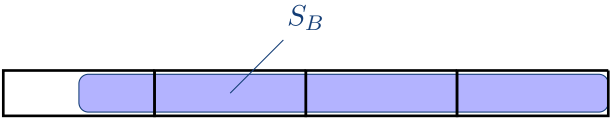
Ramstake KEM

- KeyGen:
  - sample seed for G
  - sample a, b
  - public key: (seed for G, $aG + b$)
  - secret key: $(a$, ~~b~~$)$
- DetEncaps(pk; s):
  - sample c, d
  - $S_B \leftarrow c(aG + b) + d$
  - ciphertext: $(cG + d,$ $\mathcal{E}(s) \oplus \llcorner S_B \lrcorner$, $\mathsf{H}(s))$
  - key: $K = \mathsf{H}($pk$\|$ ~~coins~~ $s)$
- Encaps(pk):
  - sample seed s
  - return DetEncaps(pk, s)
- Decaps(sk, ctxt):
  - $S_A \leftarrow a(cG + d)$
  - $s \leftarrow \mathcal{D}(\mathcal{E}(s) \oplus \llcorner S_B \lrcorner \oplus \llcorner S_A \lrcorner)$
  - ctxt', $K \leftarrow$ DetEncaps(pk; s)
  - if ctxt' $\neq$ ctxt, return $\perp$
  - else return K
- $\mathcal{E}, \mathcal{D}$ — error-correcting code
  - Reed-Solomon over GF(256) with design distance 223
  - capable of correcting 111 bit-errors
  - repeated ν times
  - used for functionality not security

    $\Rightarrow$ Ramstake is **not** a code-based cryptosystem

Pitfall

- Pitfall[1]

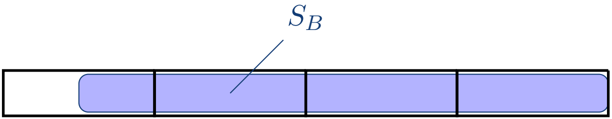


  - don't use most significant byte/chunk!

[1] credit: Jacob Alperin-Sheriff
[2] credit: Gustavo Banegas

Pitfall

- Pitfall[1]



  - don't use most significant byte/chunk!

- Sage fix:[2]

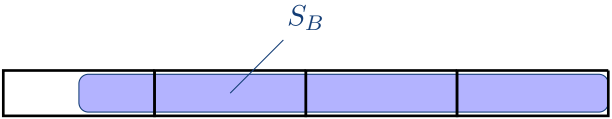
```
def export(a):
        return "".join(reversed([a[i:i+2] for
                     i in range(0, len(a), 2)]))
```

  and use `export(string)` instead

[1] credit: Jacob Alperin-Sheriff
[2] credit: Gustavo Banegas

Pitfall

- Pitfall[1]



  - don't use most significant byte/chunk!

- Sage fix:[2]

```
def export(a):
        return "".join(reversed([a[i:i+2] for
                     i in range(0, len(a), 2)]))
```

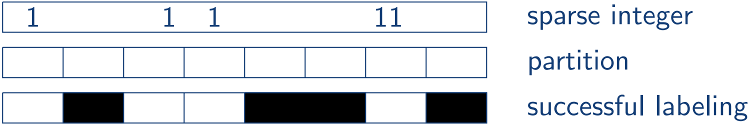
  and use `export(string)` instead

- specs and C implementations are correct

[1] credit: Jacob Alperin-Sheriff
[2] credit: Gustavo Banegas

Slice-and-Dice Attack[3]

- find (a, b) from $(G, aG + b)$
  - slice and dice
  - label randomly
  - pray
  - run LLL

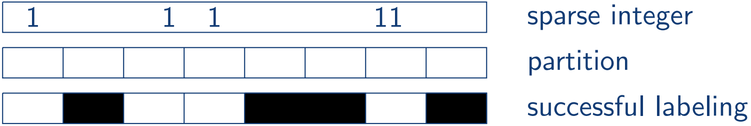


- bottleneck: guessing labels

[3] Beunardeau *et al.* "On the Hardness of the Mersenne Low Hamming Ratio Assumption"

Slice-and-Dice Attack[3]

- find (a, b) from $(G, aG + b)$
  - slice and dice
  - label randomly
  - pray
  - run LLL

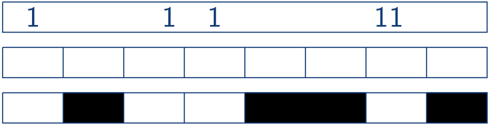


- bottleneck: guessing labels
- **not** lattice reduction

[3] Beunardeau *et al.* “On the Hardness of the Mersenne Low Hamming Ratio Assumption”

Slice-and-Dice Attack[3]

- find (a, b) from $(G, aG + b)$
  - slice and dice
  - label randomly
  - pray
  - run LLL



sparse integer

partition

successful labeling

- bottleneck: guessing labels
- **not** lattice reduction

  $\Rightarrow$ Ramstake is **not** a lattice-based cryptosystem

[3] Beunardeau *et al.* "On the Hardness of the Mersenne Low Hamming Ratio Assumption"

Comparison: Mersenne-756839

Ramstake

- pk $=$ (seed, $aG + b$)
- sk $= (a, b)$
- RS ECC $+$ repetition
- IND-CCA
  - error prob.: $\leq 2^{-64}$
- ctxt' $\stackrel{?}{=}$ ctxt
  - ctxt contains H(s)
- really simple

Mersenne-756839

- pk $= (R, fR + g)$
- sk $=$ seed
- bit-by-bit repetition
- IND-CCA
  - error prob.: $\leq 2^{-239}$
- ctxt' $\stackrel{?}{=}$ ctxt
  - no H(s)
- even simpler

Comparison: Other KEMs

Ramstake + Mersenne-756839

- $|\text{ctxt}| \approx |\text{pk}| \sim 100$ kB
- hard problem:
  - sparse integers
- simple

Other KEMs

- $|\text{ctxt}| \approx |\text{pk}| \sim 1$ kB
- hard problem:
  - lattices
  - coding theory
  - supersingular isogeny
- less simple