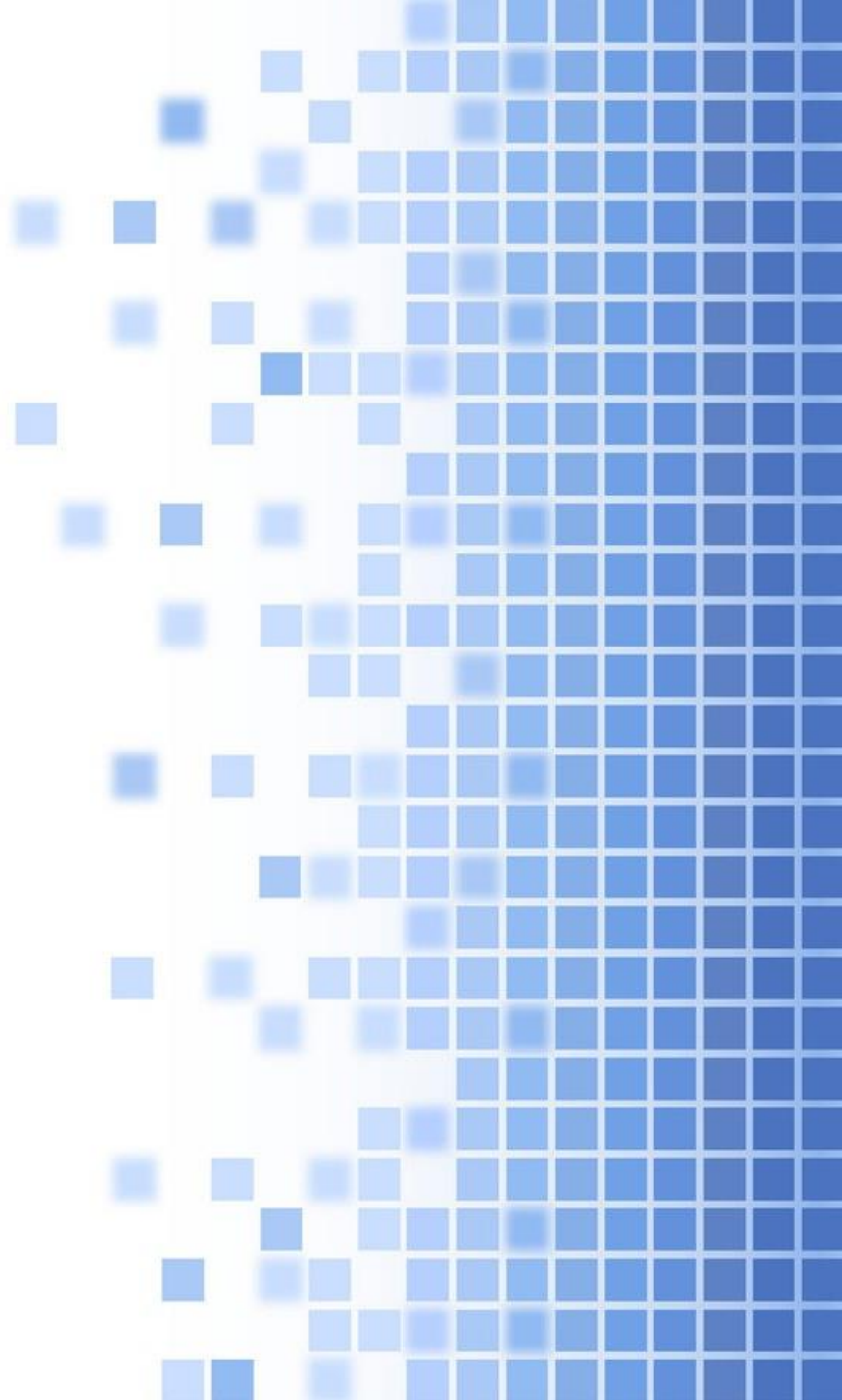
Backdoor communications using mathematics and probabilistic primality tests

Martin Grenouilloux
<martin.grenouilloux@lse.epita.fr>

Back to basics in mathematics



Prime numbers in cryptography

- A number is prime if it has no divisor except itself and 1
- They provide computational difficulty*

From this postulate comes asymmetric encryption

- RSA
- Diffie-Hellman
- …

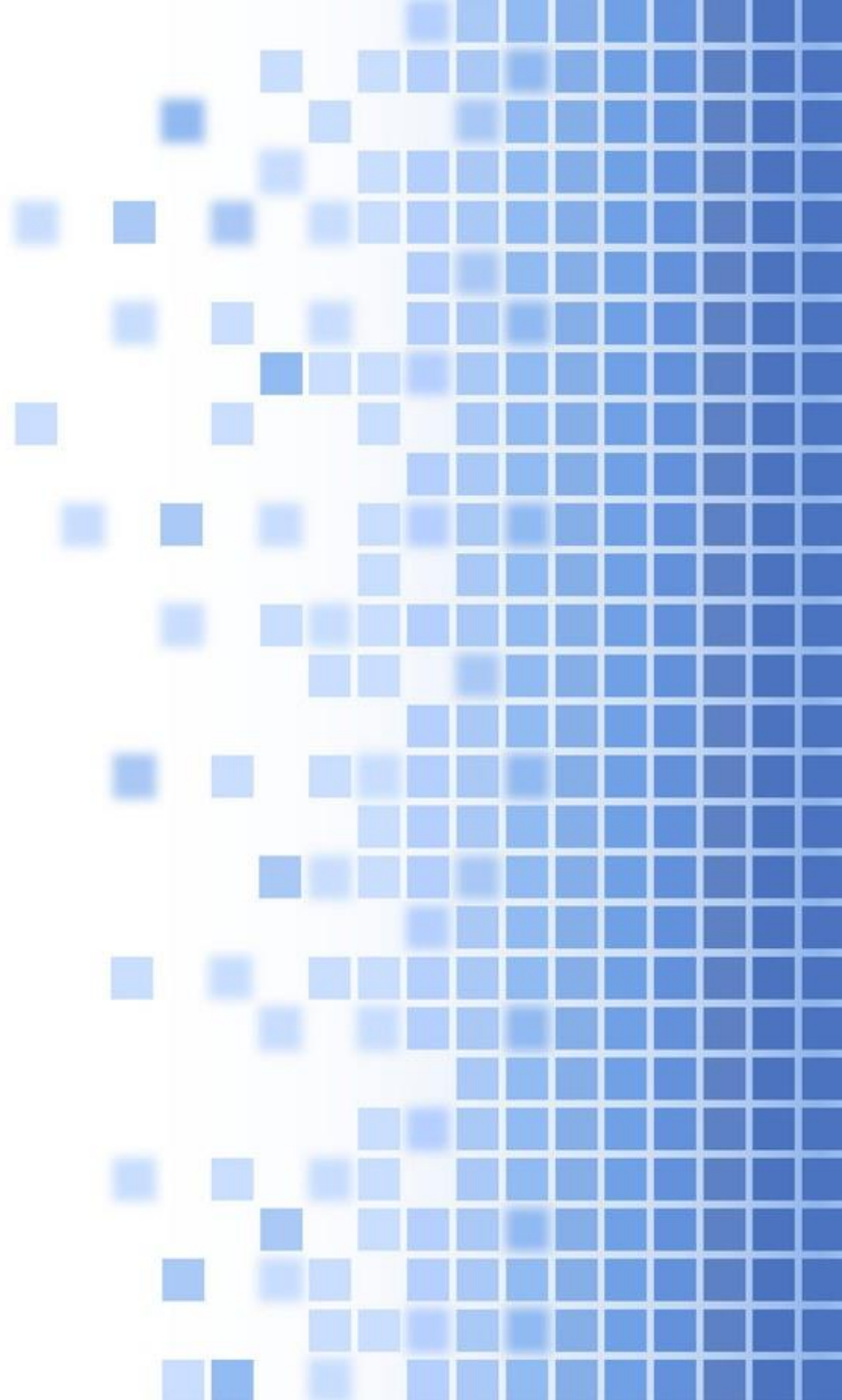
And its guaranteed security !

Fundamental Theorem of Arithmetic

- Every integer can be represented as a product of prime numbers
- This product is unique
- $120 = 2^3 * 3 * 5$

Finding such a product is considered **hard**

Of primality testing



Of primality testing – two ways to do it

Deterministic	Probabilistic
• inefficient • certainty of output • AKS test runs in $O(\log(n)^6)$	• highly efficient • approach a probability of 1 • used everywhere • Fermat, Miller-rabin, Baillie-PSW…

Fermat primality test

Based on Fermat's little theorem
→ for every prime p we have
$a^{p-1} \equiv 1\ [p]$ $a^{p} \equiv a\ [p]$

a is said to be a witness / non-witness of p's compositeness

The probability depends then on the number of witnesses tested.

Miller-Rabin primality test

Assume there only are only trivial roots to

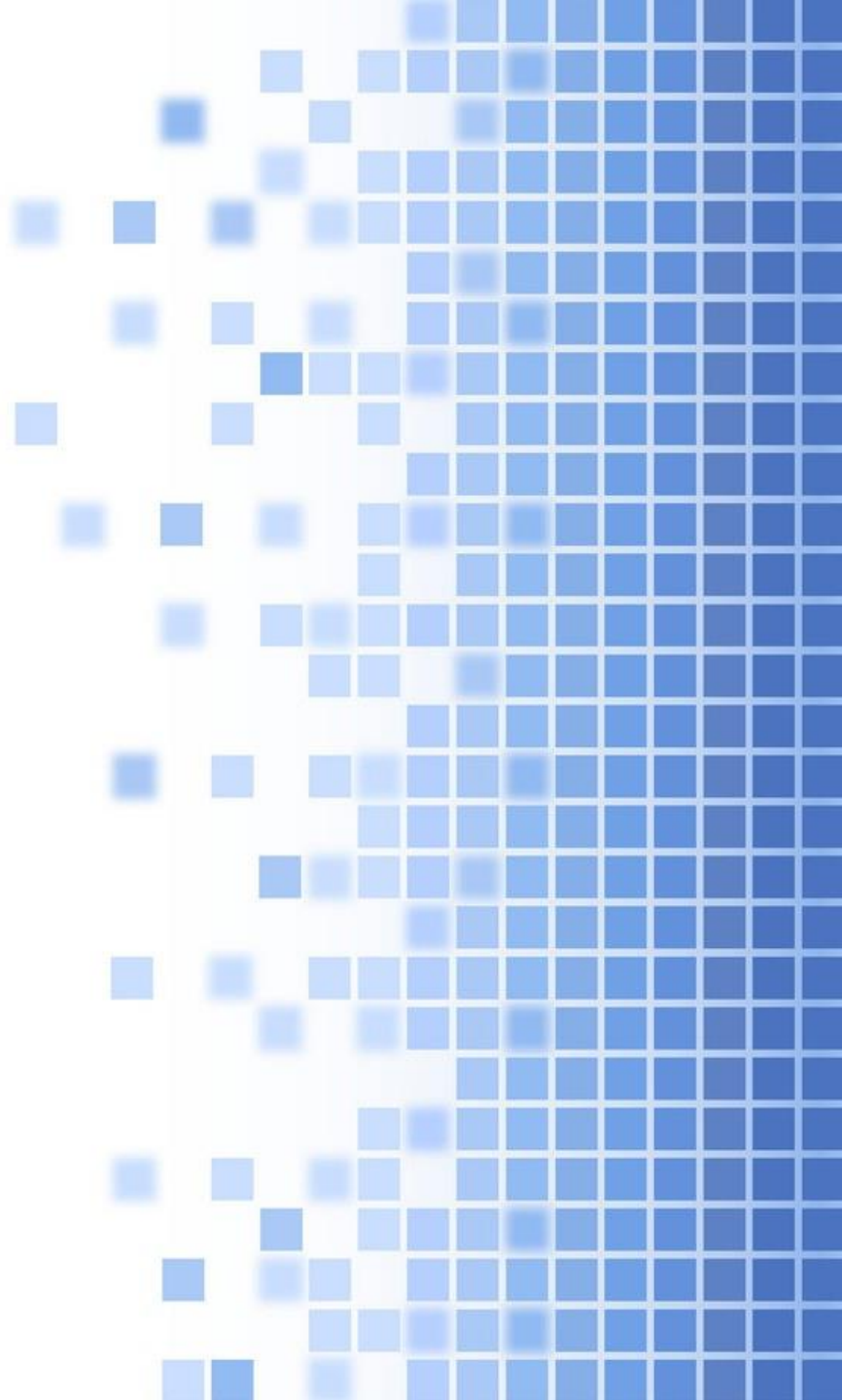
$$a^k \equiv 1\,[n]$$

$\rightarrow n = 2^e d + 1$ $\rightarrow a^d \equiv 1\,[n]$

No composite can pass the Miller-Rabin test with a probability $> (1/4)^t$

2^{-80} for cryptographical purposes ($t > 40$)

Carmichael numbers



Carmichael numbers – properties

A positive integer *n* is said to be a Carmichael number if and only if

- n is square-free
- ∀ prime divisor p of n → p-1 | n-1
- $a^{n-1} \equiv 1\ [n]$ $a^{n} \equiv a\ [n]$

All Carmichael numbers have at least 3 prime factors

Forging Carmichael numbers

Dubner's method

$T \leftarrow$ Odd fixed with many factors
$A \leftarrow$ Odd fixed
$C \leftarrow$ Odd that varies
while P and Q are not prime **do**
    $M \leftarrow \frac{(TC-1)^A}{4}$
    $P \leftarrow 6 * M + 1$
    $Q \leftarrow 12 * M + 1$
end while

for all factors f of 12*M + 4 **do**
    $R \leftarrow 6 * M * f + 1$
    if R is prime **then return** $P * Q * R$
    end if

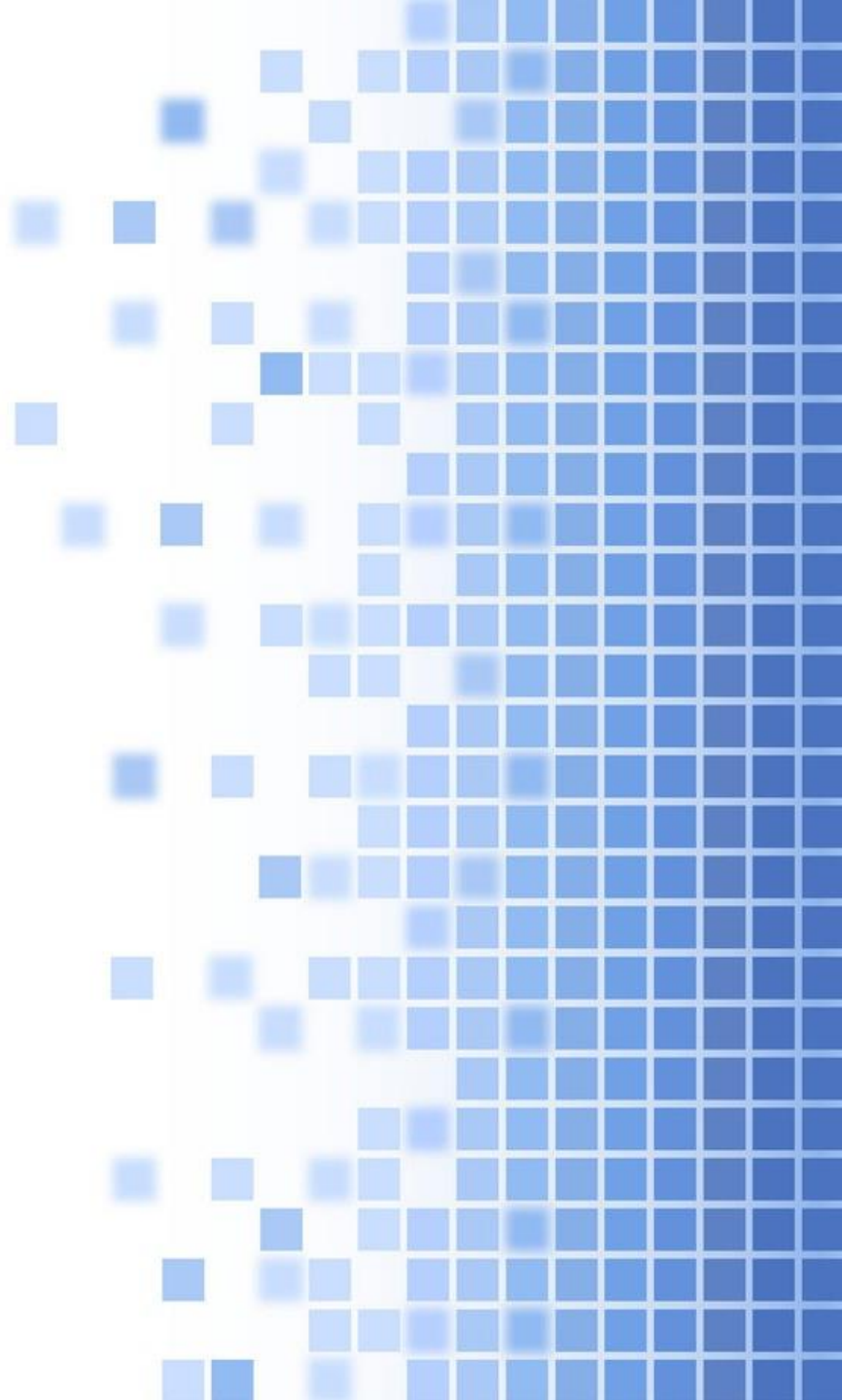
Forging Carmichael pseudoprimes

François Arnault's method

Generates a pseudoprime to a set of prime bases $\{a_1, a_2, ..., a_t\}$ with the form $n = p_1 p_2 ... p_h$

- Legendre Symbol and Gauss's law on quadratic residues
- $p_i = k_i (p_1 - 1)$

Cool bro, now what ?



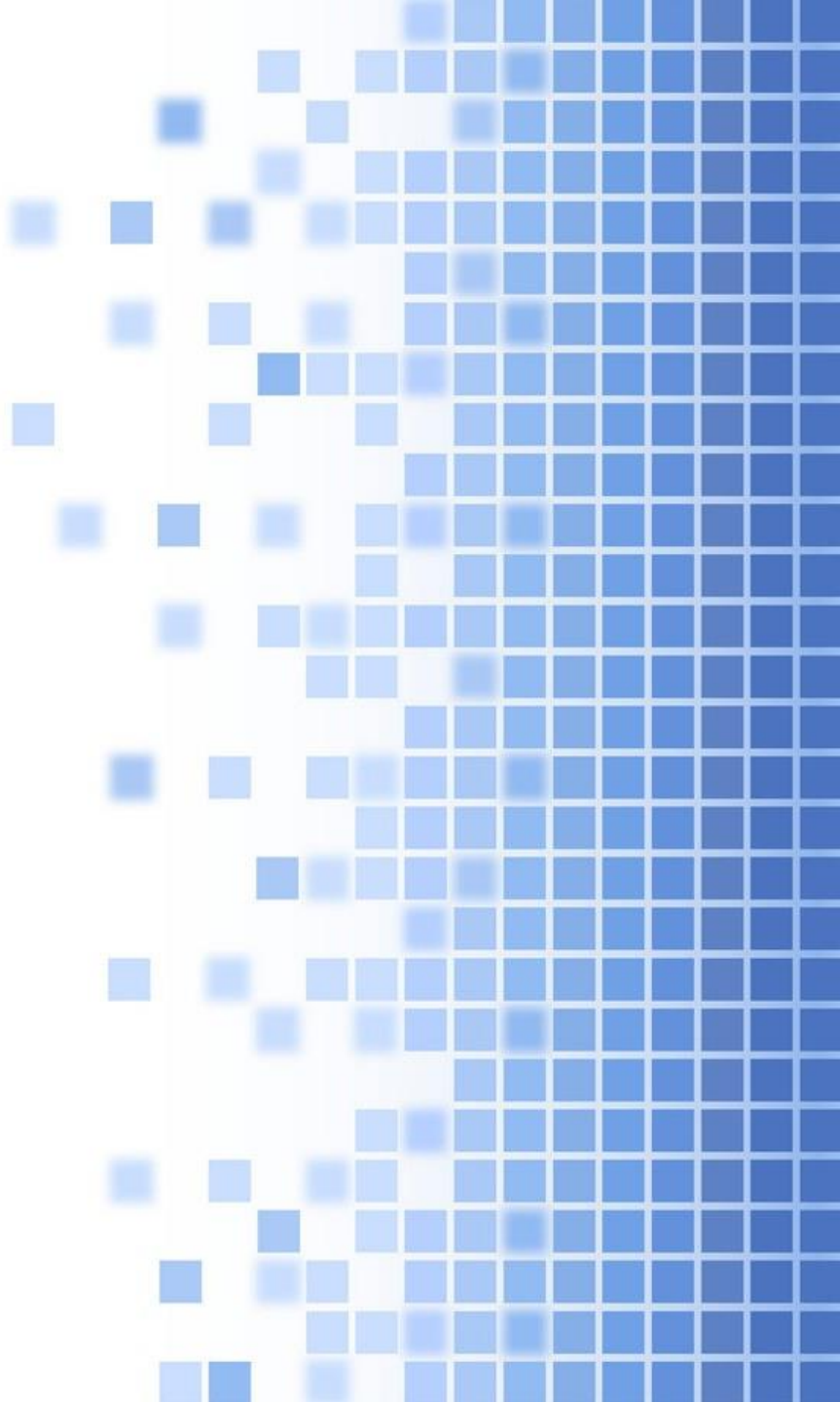
You've got prime numbers that are not

- Multi prime RSA
- Composite DH modulus
- …

Every prime-based cryptosystem

Impacted libraries

- OpenSSL
- GNU GMP
- Libgcrypt
- Apple Corecrypto
- …



Impacted libraries – libgcrypt

Source code for prime checking in libgcrypt v 1.8.2

```
/* Note: 2 is not included because it can be tested more easily by
   looking at bit 0. The last entry in this list is marked by a zero */
static ushort small_prime_numbers[] = {
    3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43,
    47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101,
        ...
    4903, 4909, 4919, 4931, 4933, 4937, 4943, 4951,
    4957, 4967, 4969, 4973, 4987, 4993, 4999,
    0
};
```

Impacted libraries – libgcrypt

Source code for prime checking in libgcrypt v 1.8.2

```
/* A quick Fermat test. */
{
  gcry_mpi_t result = mpi_alloc_like( prime );
  gcry_mpi_t pminus1 = mpi_alloc_like( prime );
  mpi_sub_ui( pminus1, prime, 1);
  mpi_powm( result, val_2, pminus1, prime );
  mpi_free( pminus1 );
  if ( mpi_cmp_ui( result, 1 ) )
    {
      /* Is composite. */
      mpi_free( result );
      progress('.');
      return 0;
    }
  mpi_free( result );
}
```

Fermat primality test

Impacted libraries – libgcrypt

Source code for prime checking in libgcrypt v 1.8.2

```
if (!cb_func || cb_func (cb_arg, GCRY_PRIME_CHECK_AT_MAYBE_PRIME, prime))
  {
    /* Perform stronger tests. */
    if ( is_prime( prime, rm_rounds, &count ) )
      {
        if (!cb_func
            || cb_func (cb_arg, GCRY_PRIME_CHECK_AT_GOT_PRIME, prime))
          return 1; /* Probably a prime. */
      }
  }
```

Impacted libraries - libgcrypt

Source code for prime checking in libgcrypt v 1.8.2

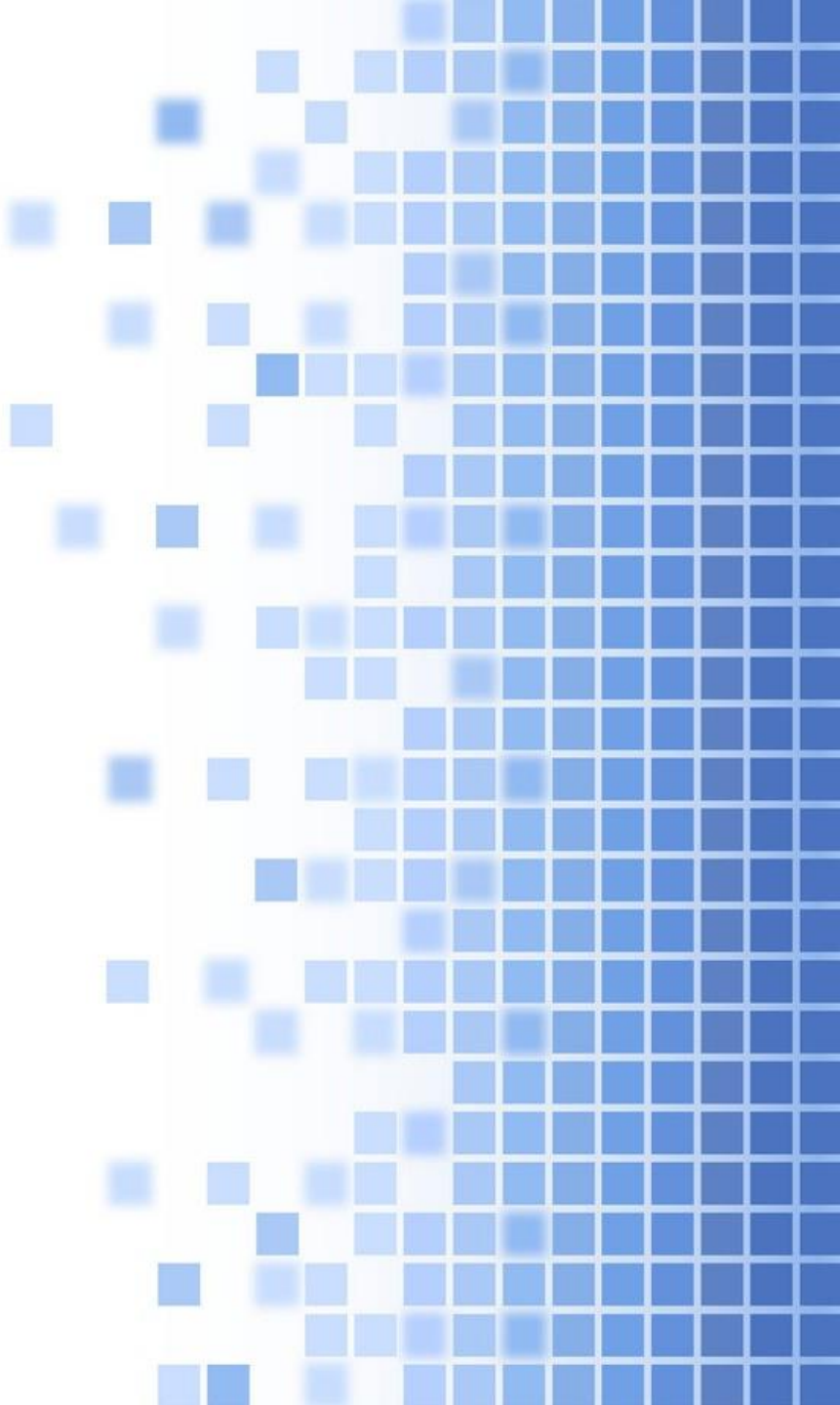
```
if (steps < 5) /* Make sure that we do at least 5 rounds. */
  steps = 5;
```

- probability that p is not prime is $(1/4)^5 = 1 / 1024$
- A cryptographically acceptable probability is 2^{-80}

Resistant libraries – Baillie-PSW

- Probabilistic primality test
- Combine Lucas AND Miller-Rabin
- No known pseudoprimes

- Java 10+ (≥ 100 bits)
- Crypto++
- Golang crypto
- …



Any questions ?

Martin Grenouilloux
<martin.grenouilloux@lse.epita.fr>

Primality testing under adversarial conditions
(Martin R. Albrecht, Jake Massimo, Kenneth G. Paterson, and Juraj Somorovsky)

A new method for producing large Carmichael numbers
(H. Dubner)