Fingerprinting and Primality

Josep Díaz   Maria J. Serna   Conrado Martínez
U. Politècnica de Catalunya

RA-MIRI 2023–2024

Fingerprinting technique

Freivalds's algorithm is an example of the algorithmic fingerprinting technique, we do not want to compute, but just to check.

We want to compare two items, A_1 and A_2, instead of comparing them directly, we compute random fingerprints $\phi(A_1)$ and $\phi(A_2)$ and compare these.
We seek a fingerprint function $\phi()$ with the following properties:

- If $A_1 = A_2$ then $\mathbb{P}[\phi(A_1) = \phi(A_2)] = 1$.
- If $A_1 \neq A_2$ then $\mathbb{P}[\phi(A_1) = \phi(A_2)] \leqslant c$ for some $c \leqslant 1/2$ or $\mathbb{P}[\phi(A_1) = \phi(A_2)] \to 0$ whp.
- It is a lot more efficient to compute and compare $\phi(A_1)$ and $\phi(A_2)$, than computing and comparing A_1 and A_2.

Notice that for Freivalds's algorithm, if A is $n \times n$ matrix, then $\phi(A) = A \cdot \vec{r}$, for a random n-dimensional Boolean vector $\vec{r}$.

Database consistency

From MR 7.4

Alice and Bob are in different continents. Each has a copy of a huge database with N bits. Alice maintain its large N-bit database $X = \{x_{N-1}, \ldots, x_0\}$ of information, while Bob maintains a second copy $Y = \{y_{N-1}, \ldots, y_0\}$ of the same database.

Periodically they want to check consistency of their copies, i.e., to check that both are the same.

Alice could send X to Bob, and he could compare it to Y. But this requires transmission of N bits, which is costly and error-prone.

Instead, suppose Alice first computes a much smaller fingerprint $\phi(X)$ and sends this to Bob. He then computes $\phi(Y)$ and compares it with $\phi(X)$. If the fingerprints are equal, he announces that the copies are identical.

What kind of fingerprint function should we use here?
How many bits do we need to send?

Which is the error in the fingerprint test?

Database consistency

From MR 7.4

Alice and Bob are in different continents. Each has a copy of a huge database with N bits. Alice maintain its large N-bit database $X = \{x_{N-1}, \dots, x_0\}$ of information, while Bob maintains a second copy $Y = \{y_{N-1}, \dots, y_0\}$ of the same database.

Periodically they want to check consistency of their copies, i.e., to check that both are the same.

Alice could send X to Bob, and he could compare it to Y. But this requires transmission of N bits, which is costly and error-prone.

Instead, suppose Alice first computes a much smaller fingerprint $\phi(X)$ and sends this to Bob. He then computes $\phi(Y)$ and compares it with $\phi(X)$. If the fingerprints are equal, he announces that the copies are identical.

What kind of fingerprint function should we use here?

How many bits do we need to send?

Which is the error in the fingerprint test?

Database consistency

From MR 7.4

Alice and Bob are in different continents. Each has a copy of a huge database with N bits. Alice maintain its large N-bit database $X = \{x_{N-1}, \ldots, x_0\}$ of information, while Bob maintains a second copy $Y = \{y_{N-1}, \ldots, y_0\}$ of the same database.

Periodically they want to check consistency of their copies, i.e., to check that both are the same.

Alice could send X to Bob, and he could compare it to Y. But this requires transmission of N bits, which is costly and error-prone.

Instead, suppose Alice first computes a much smaller fingerprint $\phi(X)$ and sends this to Bob. He then computes $\phi(Y)$ and compares it with $\phi(X)$. If the fingerprints are equal, he announces that the copies are identical.

What kind of fingerprint function should we use here?
How many bits do we need to send?
Which is the error in the fingerprint test?

Database consistency

From MR 7.4

Alice and Bob are in different continents. Each has a copy of a huge database with N bits. Alice maintain its large N-bit database $X = \{x_{N-1}, \ldots, x_0\}$ of information, while Bob maintains a second copy $Y = \{y_{N-1}, \ldots, y_0\}$ of the same database.

Periodically they want to check consistency of their copies, i.e., to check that both are the same.

Alice could send X to Bob, and he could compare it to Y. But this requires transmission of N bits, which is costly and error-prone.

Instead, suppose Alice first computes a much smaller fingerprint $\phi(X)$ and sends this to Bob. He then computes $\phi(Y)$ and compares it with $\phi(X)$. If the fingerprints are equal, he announces that the copies are identical.

What kind of fingerprint function should we use here?
How many bits do we need to send?

Which is the error in the fingerprint test?

Review of Algebra (i)

Given $a, b, n \in \mathbb{Z}$, a congruent with b modulo n ($a \equiv b \pmod{n}$) if $n|(a-b)$ (n divides $(a-b)$).

1. $a \mod n = b \Rightarrow a \equiv b \pmod{n}$.
2. $(a+b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n$.
3. $(a \cdot b) \bmod n = ((a \bmod n) \cdot (b \bmod n)) \bmod n$.
4. $a + (b+c) \equiv (a+b) + c \pmod{n}$ (associativity)
5. $ab \equiv ba \pmod{n}$ (commutativity)
6. $a(b+c) \equiv ab + ac \pmod{n}$ (distributivity)

n partitions $\mathbb{Z}$ in n equivalence classes: $\mathbb{Z}_n = \{0, 1 \ldots, n-1\}$. For any $m \in \mathbb{Z}$, $m \mod n \in \mathbb{Z}_n$.

Define $\mathbb{Z}_n^+ = \{1 \ldots, n-1\}$. $(\mathbb{Z}_n, +_n, \cdot_n)$ form a commutative ring.

Review of Algebra (ii)

Theorem (Prime number Theorem)

Let $n \in \mathbb{Z}$ and let $\pi(n)$ be the number of primes $\leqslant n$, then

$$\pi(n) \sim \frac{n}{\ln n}, \text{ as } n \to \infty.$$

The frequency of primes slowly decays as the integers increase in length.

For ex. if $n = 10^4$, $\pi(n) = 1929$ and $\frac{n}{\ln n} = 1086$,
while, if $n = 10^7$, $\pi(n) = 664579$ and $\frac{n}{\ln n} = 620420$.

Review of Algebra (iii)

Lemma

If $n \in \mathbb{Z}$ *has* N*-bits, then* $n \leqslant 2^N$*, and at most* N *different primes can divide* n

Proof

As prime numbers are $\geqslant$ 2, the number of distinct primes that divide n is $\leqslant$ N, because if we multiply together more than N numbers that are at least 2, then we would get a number greater than 2^N $\square$

Example

For ex. if $n = 33$, $(33_2 = 100001)$, so $N = 6$ and $2^6 = 64$. Besides, $\pi(33) = 11$ of which only 2 of them divide 33 $(2 < 6)$

Review of Algebra (iv)

Corollary

Let p_i *be the* i*-th. prime number, then the value of* $p_i \sim i \ln i$

Example

For ex. if $i = 1000$, then $p_i \sim 1000 \ln(1000) = 6907$ and the exact value is $p_{1000} = 7919$

Solution to the database consistency problem

If Alice (A) has X and Bob (B) has Y, they use the following algorithm to check they are the same:

- See the data as N-bit integers: $\mathbf{x} = \sum_{i=0}^{N-1} x_i 2^i$ and $\mathbf{y} = \sum_{i=0}^{N-1} y_i 2^i$.
- A chooses u.a.r. a prime $p \in [2, 3, 5, \ldots, m]$, for suitable $m = cN \ln N$. (The number of primes in 2^N is N)
- A computes $\phi(\mathbf{x}) = \mathbf{x} \bmod p$ and sends the result together with the value p to B.
- B computes $\phi(\mathbf{y}) = \mathbf{y} \bmod p$ and compares with the quantity he got from A.
- If $\phi(\mathbf{x}) \neq \phi(\mathbf{y})$ for sure $X \neq Y$, but it is possible $\phi(\mathbf{x}) = \phi(\mathbf{y})$ and $X \neq Y$. (This happens if $\mathbf{x} \bmod p = \mathbf{y} \bmod p$, with $\mathbf{x} \neq \mathbf{y}$).

Bounding the probability of error

By the Prime Number Theorem $\pi(m) \sim \frac{m}{\ln m}$, so as we see below, we need to take $m = cN \ln N$, for constant $c > 1$.

We want to bound the probability that $\mathbf{x} \neq \mathbf{y}$ but $\phi(\mathbf{x}) = \phi(\mathbf{y})$, i.e.,

$$\begin{aligned}
\mathbb{P}[\mathbf{x} \bmod p = \mathbf{y} \quad \bmod p \mid \mathbf{x} \neq \mathbf{y}] &= \mathbb{P}[p \text{ divides } |\mathbf{x} - \mathbf{y}|] \\
&= \frac{\#\text{ of primes dividing } |\mathbf{x} - \mathbf{y}|}{\#\text{ of primes} \leqslant m} \\
&\leqslant \frac{N}{m/\ln m} = \frac{N \ln m}{cN \ln N} = \frac{\ln m}{c \ln N} \\
&= \frac{\ln(cN \ln N)}{c \ln N} = \frac{\ln N + \ln(c \ln N)}{c \ln N} \\
&= \frac{1}{c} + \frac{\ln(c \ln N)}{c \ln N} = \frac{1}{c} + o(1)
\end{aligned}$$

Lemma: Taking $c = 1/\epsilon$ for a chosen $0 < \epsilon < 1$, the algorithm achieves an error probability of $\leqslant \epsilon$.

Choosing a large $m \Rightarrow$, i.e. a large c, we have a larger selection for p, so it is less likely that p divides $|\mathbf{x} - \mathbf{y}|$.

Communication bits

Lemma

The fingerprint algorithm to check the consistency of two databases with N bits uses $\mathcal{O}(\lg N)$ bits of communication.

Proof

A sends to B p and $\mathbf{x} \bmod p$, both are $\leqslant m$.
Since $m = cN \ln N$, then m requires $\lg(cN \ln N) = \lg N + \lg(c \ln N) \sim \mathcal{O}(lgN)$ bits, so the number of transmitted bits is $\mathcal{O}(\lg N)$. $\square$

We proved that by using a more efficient representation of the data (modular), the randomized fingerprinting algorithm gives an exponential decrease in the amount of communication at a small cost in correctness.

How to pick a random prime number

Problem: Given an integer N we want to pick a random prime $p \in [2, \ldots, 2^N - 1]$.

Recall: if n has N bits $\Rightarrow n \leqslant 2^N - 1$ and $N \geqslant \lg n$.

Assume we have an efficient algorithm **Prime?** which tell us if an integer is a prime, or not.
Define the set $P = \{p \mid 1 < p \leqslant 2^N - 1 \text{ and } p \text{ is prime}\}$.
We want to pick u.a.r. $p \in P$ (i.e., with probability $\frac{1}{|P|}$)

```
procedure PICKPRIME(p)
    for i := 0 to t do
        p := RAND(2^N − 1)
        if PRIME?(p) then
            return p
        end if
    end for
end procedure
```

t will be fixed later
First analyze one iteration of the algorithm
After we analyze the probability of error after amplifying t times.

Analysis of the algorithm

Let A be the event that a random generated N-bit integer is a prime in P:

$$\mathbb{P}[A] = \frac{|P|}{2^N} = \frac{(2^N / \ln 2^N)}{2^N} = \frac{1}{N \ln 2} = \frac{1.442}{N}.$$

Example

If $N = 2000$ then $\mathbb{P}[A] = 0.000721$, therefore the probability of failing is $\mathbb{P}\left[\bar{A}\right] = 0.999271$. Quite high!

Taking into consideration the t-amplification,

$$\mathbb{P}[\text{Failure after } t \text{ repetitions}] = \left(1 - \frac{1.442}{N}\right)^t \leqslant e^{-\frac{1.442t}{N}},$$

so taking $t = 10N$ suffices to make small the probability of failure.

Analysis of the algorithm: Numerical example

Example

If $N = 2000$ taking $t = 10N = 20000$ yields $\mathbb{P}[\text{Failure}] = 0.00004539$ and $\mathbb{P}[\text{Success}] = 0.999955$. If $t = N = 2000$, $\mathbb{P}[\text{Success}] = 0.76425$.

In practice, most of the algorithms to generate a large prime, follow the previous scheme (see for ex. https://asecuritysite.com/encryption/random3)

The Primality problem

From Cormen et al., 31.8 (3rd edition)
INPUT: $n \in \mathbb{N}$. QUESTION: Is n prime?

Naïve algorithm:

```
procedure PRIME?(n)
    for a ∈ {2, 3, ..., √n} do
        if n mod a = 0 then
            return false ▷ n is composite
        end if
    end for
    return true
end procedure
```

Recall that in arithmetic complexity, for large n ($n = 2^{2024}$), the input size is the number of bits N to express n
i.e., $n = 2^N$ and $N = \lg n$

Complexity of the algorithm: $T(N) = \mathcal{O}(2^{N/2}N^2)$ Too slow!

Randomized algorithms for Primality Testing

Theorem (Fermat's Little Th.,XVII)

If n is prime, then for all $a \in \mathbb{Z}_n^+$, $a^{n-1} \equiv 1 \pmod{n}$.

Fermat only works in one direction. There exist composite integers n s.t. for all a, $a^{n-1} \equiv 1 \pmod{n}$ such that $\gcd(a, n) = 1$. These composite numbers are known as Carmichael numbers.

For example $561 = 3 \times 11 \times 17$ is the smallest Carmichael number. The next two are 1105 and 1729.

Carmichael numbers are very rare.

$C(x) =$ # of Carmichael numbers $\leqslant x$

k	8	9	10	11	12	13	14
$C(10^k)$	255	646	1547	3605	8241	19279	44706

k	15	16	17	18	19	20	21
$C(10^k)$	105212	246683	585355	1401644	3381806	8220777	20138200

Test of pseudo-primality

Assuming the non-existence of Carmichael numbers:

```
procedure PRIME?(n)
    a := RAND(1, n − 1)
    if  a^(n−1) ≡ 1 (mod n) then
        return  "prime"
    else
        return  composite
    end if
end procedure
```

Complexity: $\mathcal{O}(N^3)$.

Test of pseudo-primality: Error probability

Theorem

Assume n is not a Carmichael number. If the algorithm says composite then n is composite.
If the algorithm says prime then the answer might be correct because n is prime or it might be wrong: n can be composite and still $a^{n-1} \equiv 1 \pmod{n}$; but this happens with probability $\leqslant 1/2$.

Test of pseudo-primality: Error probability

Proof

Suppose n is composite but not a Carmichael number.
Then the set

$$F_n := \{a \mid 1 \leqslant a < n \wedge a^{n-1} \equiv 1 \pmod{n}\}$$

must be a proper subset of $\mathbb{Z}_n^+ = \{a \mid 1 \leqslant a < n\}$, because there must be at least one $a \in \mathbb{Z}_n^+$ such that $a^{n-1} \not\equiv 1 \pmod{n}$—because n is composite and it is not a Carmichael number.
But $(F_n, \cdot)$ is then a proper subgroup of Z_n^+, and this implies that $|F_n|$ must divide $|Z_n^+| = n - 1$. Since $|F_n| < |Z_n^+|$ we must have $|F_n| \leqslant |Z_n^+|/2$, therefore the probability that we choose an integer from F_n will be at most 1/2.
□

Test of pseudo-primality: Error probability

The previous algorithm has one-side error, therefore amplifying t times the algorithm, the probability of error goes down to $\leqslant 1/2^t$. The complexity is $\mathcal{O}(tN^3)$.

```
procedure REPEATED-FERMAT(n, t)
    for i := 1 to t do
        a := RAND(1, n − 1)
        if a^(n−1) ≢ 1 (mod n) then
            return composite
        end if
    end for
    return "prime"
end procedure
```

Taking into consideration the Carmichel numbers

- If equation $x^2 \equiv 1 \pmod{n}$ has exactly solutions $x = \pm 1$ that implies n is prime.
- If there is another solution different than ± 1, then n can not be prime.
- To see if n is prime: Randomly choose an integer $a < n$, if $a^2 \equiv 1 \pmod{n}$, then a is a non-trivial root of $1 \mod n$, so n is not prime. Such an a is denoted a witness to the compositeness of n. Otherwise, n may be a prime.

Based on the observation above G. Miller (1976) and later M. Rabin (1980) gave an algorithm which is very similar to the pseudoprimality test in previous slides; however, it will detect if n is a Carmichael number and report *composite* in that case. Miller-Rabin's algorithm is also a Montecarlo one-side error algorithm and the probability of error be reduced to less than 2^{-t} as usual.

Deciding primality

- For a long time it was open to prove that primality is in P. In 2006, Agrawal, Kayal and Saxena gave a deterministic polynomial time algorithm for Primality.
- If $n \leqslant 2^{N}$ the best implementation for the AKS algorithm is $\tilde{\mathcal{O}}(N^{6}) = \mathcal{O}(N^{6} \lg N)$.
- AKS has terrible running time, and it is not clear that it can be improved in the near future.
- Miller-Rabin's algorithm is the basis for existing efficient algorithms.
- However, the Fermat pseudo-primality test can also work fairly nicely; for example, if we are dealing with $N = 9$, the probability of hitting a Carmichel number is 0.000000255, so we can take this little risk —and avoid the somewhat costly and cumbersome tests needed to deal with Carmichael numbers.