Statistical and Computational Security

Cryptography and Protocols

Andrei Bulatov

Statistical Security

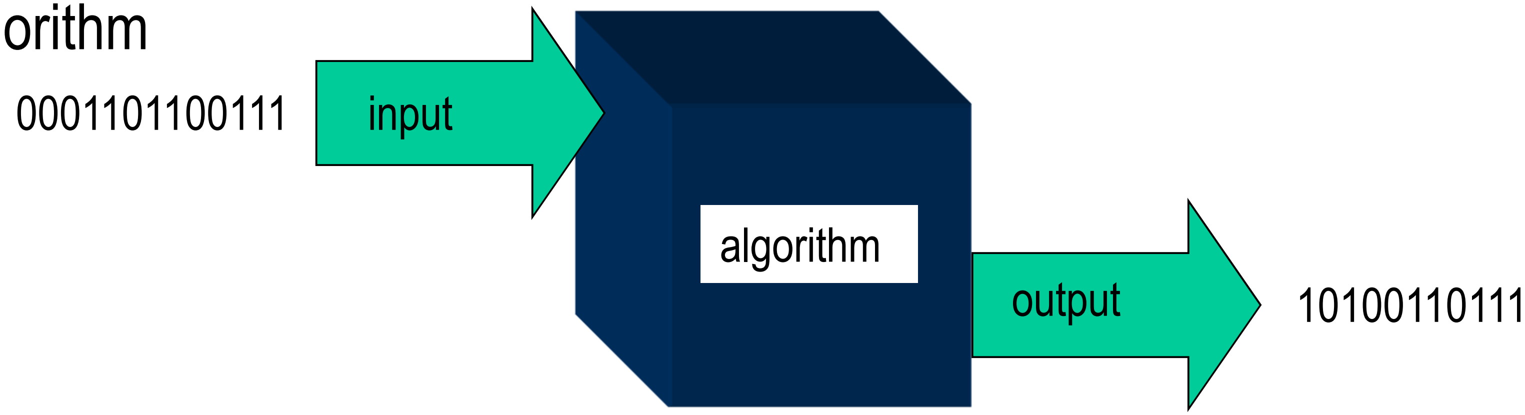
- Let $\mathcal{X}$ and $\mathcal{Y}$ be two distributions over $\{0,1\}^m$ The statistical distance between $\mathcal{X}$ and $\mathcal{Y}$, denoted $\Delta(\mathcal{X},\mathcal{Y})$ is

$$\max_{T\subseteq\{0,1\}^m} |\Pr[\mathcal{X}\in T]-\Pr[\mathcal{Y}\in T]|$$

- A symmetric encryption scheme is said to be ε-statistically secure, if for any two plaintexts P_1, P_2 distributions $E_k(P_1), E_k(P_2)$ are ε-equivalent

Algorithms

- Algorithm



- Algorithm performs a sequence of `elementary steps' that can be:
  - arithmetic operations
  - bit operations
  - Turing machine moves
  - ………. (but not quantum computing!!)
- We allow probabilistic algorithms, that is flipping coins is permitted

Complexity

- The time complexity of algorithm A is function $f(n)$ that is equal to the number of elementary steps required to process the most difficult input of length n
- We do not distinguish algorithms of complexity $2n^2$ and $100000n^2$
- A computational problem has time complexity at most $f(n)$ if there is an algorithm that solves the problem and has complexity $O(f(n))$

  - problem solvable in linear time: there is an algorithm that on input of length n performs at most Cn steps

  - problem solvable in quadratic time: there is an algorithm that on input of length n performs at most Cn^2 steps
- Polynomial time solvable problems: O(polynomial); P, BPP

Complexity (cntd)

- Polynomial time solvable problems:

  There is a polynomial p(n) such that the problem is solvable in time O(p(n)
- P - class of problems solvable in poly time by a deterministic algorithm
- BPP - class of problems solvable in poly time by a probabilistic algorithm
- An algorithm is superpolynomial if its time complexity f(n) is not in O(p(n)) for any polynomial p(n)
- A function ε: N $\to$ [0,1] is polynomially bounded if $\varepsilon(n) \geq \frac{1}{p(n)}$ for some polynomial p(n)

Computational Security

- Let (K,E,D) be a SES that uses n-bit keys to encrypt m(n)-bit messages. It is computationally secure if for any polynomial time algorithm Eve: $\{0,1\}^* \to \{0,1\}$, any polynomially bounded ε: $\{0,1\}^* \to [0,1]$, n, and $P_1, P_2 \in \{0,1\}^{m(n)}$

$$|\Pr[\mathrm{Eve}(E_{U_n}(P_1)) = 1] - \Pr[\mathrm{Eve}(E_{U_n}(P_2)) = 1]| < \varepsilon(n)$$

- **Conjecture**.

  A computationally secure SES exists for $m(n) = n^{100}$ (may be even for $m(n) = 2^{0.9n}$)

Computational Indistinguishability: Difficulties

- It is useful to define computational security in a similar way as statistical one: define distance or equivalence of distributions and then say that $E_{U_n}(P_1)$ and $E_{U_n}(P_1)$ are `equivalent'. However, there are problems
- For computational definitions we need algorithms, not events

  Solution: Instead of saying $X \in S$ we use the characteristic function f of S. So we say f(X) = 1 instead.

  Distance between distributions can then be defined as

  $$\max_f |\Pr[f(X)=1]-\Pr[f(Y)=1]|$$

  over all `easily' computable functions f
- Computational complexity does not make sense for fixed distributions.

  Solution: Use collections or sequences of random variables

Computational Indistinguishability: Definition

- Let T(n) and ε(n) be functions on natural numbers. Collections of random variables $\{X_n\}$ and $\{Y_n\}$ such that $X_n, Y_n \in \{0,1\}^n$ are said to be computationally (T,ε)-indistinguishable , if for any probabilistic algorithm Alg with time complexity at most T(n)

$$|\Pr[\mathrm{Alg}(X_n) = 1] - \Pr[\mathrm{Alg}(Y_n) = 1]| \le \varepsilon(n)$$

  Denoted $\{X_n\} \approx_{T,\varepsilon} \{Y_n\}$

- For example:

  Let (K,E,D) be a SES that uses n-bit keys to encript m(n)-bit messages. It is computationally secure if for any $P_1, P_2 \in \{0,1\}^{m(n)}$ distributions $E_{U_n}(P_1)$ and $E_{U_n}(P_1)$ are (T,ε)-indistinguishable for any polynomial T and any polynomially bounded ε

Pseudo Random Generators

Cryptography and Protocols

Andrei Bulatov

Pseudorandom Generators

- Let T(n), ε(n) be functions. A collection $\{X_n\}$ of random variables with $X_n \in \{0,1\}^n$ is called (T,ε)-pseudorandom if $\{X_n\} \approx_{T,\varepsilon} \{U_n\}$

- A collection of functions $g_n : \{0,1\}^n \to \{0,1\}^{m(n)}$ is called a (T,ε)-pseudorandom generator if $\{g_n(U_n)\}$ is (T,ε)-pseudorandom

Good Pseudorandom Generators

- $m(n) > n$ Otherwise it is trivial and useless
  $m(n) - n$ is the stretch of a PRG
- A functions T is called superpolynomial if for any polynomial $p(n)$, $p \in o(T)$
- A pair of functions (T,ε) is superpolynomial if T is superpolynomial and $\varepsilon(n) = 1/f(n)$ where f is superpolynomial
- A PRG should be (T,ε)-pseudorandom for some superpolynomial pair (T,ε)
- g_n must be efficiently computable
- **PRG Axiom**: A good PRG exists
- (see Goldreich) There is a $(2^{n/8}, 2^{-n/8})$ -pseudorandom generator

PRGs and Statistical Security

- **Lemma**.

  If m(n) > n then for any collection of functions $\{g_n\}$ we have
  $\Delta(g_n(U_n), U_m) \geq 1/2$

- **Proof**.

  Let S_n be the image of $g_n(\{0,1\}^n)$. Clearly $|S_n| \leq 2^n \leq 2^{m(n)-1}$

  Thus $\Pr[g_n(U_n) \in S_n] = 1$ while $\Pr[U_{m(n)} \in S_n] \leq 1/2$

Candidate PRGs: Blum – Blum - Shub

- This is a PRG that given an input of length 2n produces a string of bits of length m, where m is as big as we want

- Input: an n-bit integer N and integer X, $1 \leq X \leq N$

```
num_outputted = 0;
while num_outputted < m:
        X := X*X mod N
        num_outputted := num_outputted + 1
        output (least-significant-bit(X) )
endwhile
```

Blum – Blum – Shub is Good

- **Theorem**.

  The BBS PRG is (T,ε)-pseudorandom for some superpolynomial pair (T,ε) if the assumption below is true

- **Assumption**.

  There is a superpolynomial pair (T,ε) such that for any probabilistic algorithm Alg with time complexity less than $T(n)$ the following holds

  Pr[Alg finds factorization of a random n-bit integer] $< \varepsilon(n)$

Candidate PRGs: RC4

- RC4 stands for Ron's Cipher no. 4
- Widely used: SSL (and then TSL), SSH, WEP, WPA (IEEE 802.11), BitTorrent protocol encryption, Microsoft Point-to-Point Encryption,
- A byte is a number from {0,…,255}
- Input: a permutation S: {0,…,255} $\rightarrow$ {0,…,255}

```
i := 0 j := 0
while num_outputted < m :
    i := (i + 1) mod 256 j := (j + S[i]) mod 256
    swap(S[i],S[j])
    output (S[(S[i] + S[j]) mod 256] )
endwhile
```

Candidate PRGs: RC4 (cntd)

- RC4 given an input of length 2048 produces an output of length m, which is as big as we want
- If 2048 is too much, there is another algorithm – KSE, the Key Scheduling Algorithm – that uses an input of length $40 \le n \le 128$ to generate S
- Input: a key k of length n, $40 \le n \le 128$

```
for i from 0 to 255     S[i] := i     endfor
j := 0
for i from 0 to 255
    j := (j + S[i] + k[i mod n]) mod 256
    swap(S[i],S[j])
endfor
```