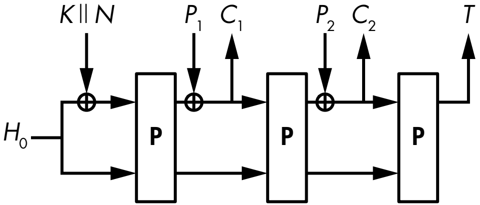


CS 553
CRYPTOGRAPHY

Lecture 22
Authenticated Encryption
$+$
Computationally Hard Problems

Instructor
Dr. Dhiman Saha

One Unified Primitive

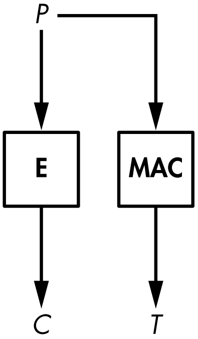
Two Goals – Privacy + Authenticity

- Relatively New Area
- Recently concluded CAESAR competition

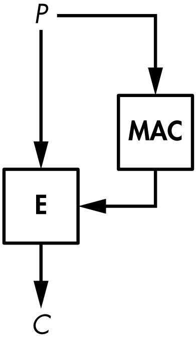
CAESAR

Competition for **A**uthenticated **E**ncryption: **S**ecurity, **A**pplicability, and **R**obustness

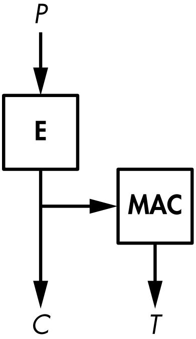
AE using MACs



Encrypt-and-MAC



MAC-then-encrypt



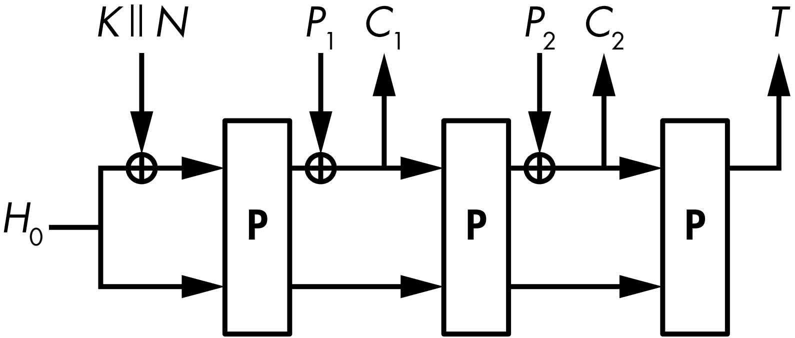
Encrypt-then-MAC

Which combination is most secure?

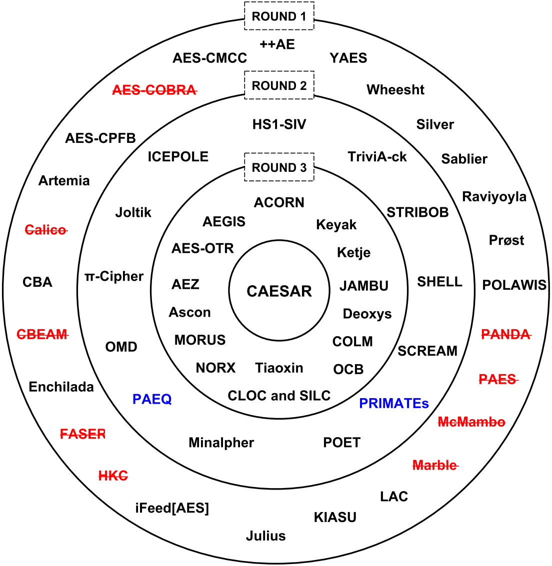
- E&M
- MtE
- EtM

Additional Features

- AEAD - AE with Associated Data
- Nonce based AE
- RUP - Release of Unverified Plaintexts
- And many more



Permutation Based AE


ROUND 1
++AE
AES-CMCC
YAES
~~AES-COBRA~~
Wheesht
AES-CPFB
Silver
Sablier
Artemia
Raviyoyla
~~Calico~~
Prøst
CBA
POLAWIS
~~CBEAM~~
~~PANDA~~
Enchilada
~~PAES~~
~~FASER~~
~~McMambo~~
~~HKC~~
~~Marble~~
iFeed[AES]
LAC
KIASU
Julius
ROUND 2
HS1-SIV
ICEPOLE
TriviA-ck
Joltik
STRIBOB
π-Cipher
SHELL
OMD
SCREAM
PAEQ
PRIMATEs
Minalpher
POET
ROUND 3
ACORN
AEGIS
Keyak
AES-OTR
Ketje
AEZ
JAMBU
Ascon
Deoxys
MORUS
COLM
NORX
Tiaoxin
OCB
CLOC and SILC
CAESAR

Finalists

- ACORN
- AEGIS
- Ascon
- COLM (Two Indian Designers)
- Deoxys-II
- MORUS
- OCB

Computational Hardness

The property of computational problems for which there is **no algorithm** that will run in a **reasonable** amount of time

- Also called **intractable** problems
- Often **practically** impossible to solve

Equivalence of Computing Models

Computational hardness is **independent** of the type of computing device used.

- An exception is **quantum** computers

Equivalence of Computing Models

Computational hardness is **independent** of the type of computing device used.

- An exception is **quantum** computers

Computational Complexity

The approximate number of operations done by an algorithm as a function of its input size.

- The size is counted in **bits** or in the **number** of elements taken as input.

Linear Search $O(n)$

```
search(x, array, n):
    for i from 1 to n {
        if (array[i] == x) {
            return i;
        }
    }
    return 0;
}
```

Linear Vs Exponential

A complexity **linear** in n is considered **fast**, as opposed to complexities **exponential** in n.

- Recall Sorting by comparison
- $O(n \log n)$
- sometimes called **linearithmic** complexity
- Slower than 'linear'
- But still practical

Recall Brute-force Search

What about its complexity for key-size n?

Linear Vs Exponential

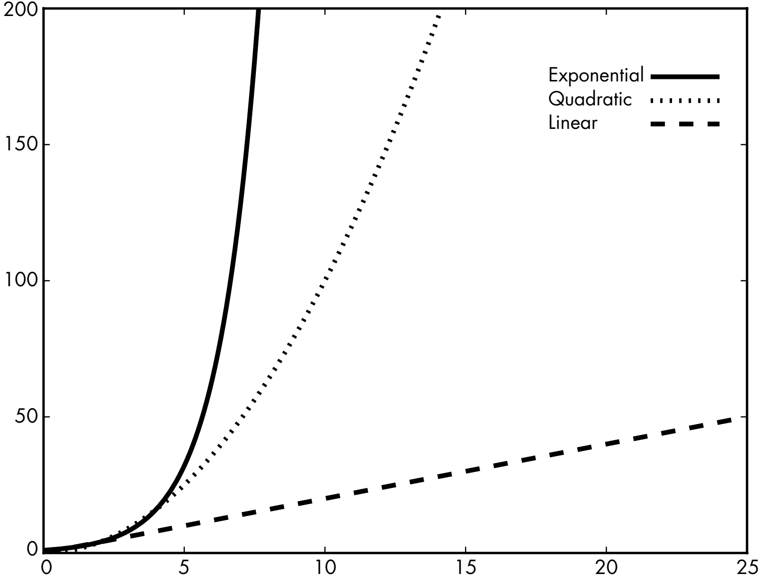
A complexity **linear** in n is considered **fast**, as opposed to complexities **exponential** in n.

- Recall Sorting by comparison
- $O(n \log n)$
- sometimes called **linearithmic** complexity
- Slower than 'linear'
- But still practical

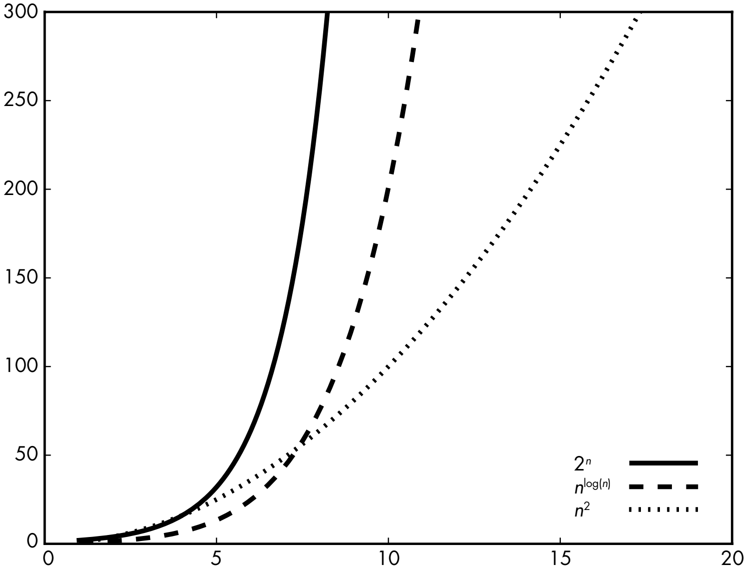
Recall Brute-force Search

What about its complexity for key-size n?

Growth of Functions



Polynomial Vs Super-polynomial



Time Complexity

TIME$(f(n))$

- **TIME**(n^2)
  - All computational problems solvable in time $O(n^2)$
- **TIME**(2^n)
  - Class of problems solvable in time $O(2^n)$

Any problem in the class **TIME**(n^2) also belongs to the class **TIME**(n^3)

Time Complexity

TIME$(f(n))$

- **TIME**(n^2)
  - All computational problems solvable in time $O(n^2)$
- **TIME**(2^n)
  - Class of problems solvable in time $O(2^n)$

Any problem in the class **TIME**(n^2) also belongs to the class **TIME**(n^3)

P **TIME**(n^k)

The union of all classes of problems, **TIME**(n^k), where k is a constant, is called **P**, which stands for **polynomial** time.

How much memory an algorithm uses.

- Note: A single memory access is usually orders of magnitudes **slower** than a basic arithmetic operation in a CPU.

SPACE$(f(n))$

The class of problems solvable using $f(n)$ bits of memory.

PSPACE

The union of all **SPACE**(n^k) problems is called **PSPACE**

How much memory an algorithm uses.

- Note: A single memory access is usually orders of magnitudes **slower** than a basic arithmetic operation in a CPU.

SPACE$(f(n))$

The class of problems solvable using $f(n)$ bits of memory.

PSPACE

The union of all **SPACE**(n^k) problems is called **PSPACE**

Note

A polynomial amount of memory doesn't necessarily imply that an algorithm is practical.

- **TIME**$(f(n))$ is included in **SPACE**$(f(n))$ How?
  - Any problem solvable in time $f(n)$ needs at **most** $f(n)$ memory

Implication

P is a subset of **PSPACE**

P Vs **PSPACE**

Note

A polynomial amount of memory doesn't necessarily imply that an algorithm is practical.

- **TIME**$(f(n))$ is included in **SPACE**$(f(n))$ How?
  - Any problem solvable in time $f(n)$ needs at **most** $f(n)$ memory

Implication

P is a subset of **PSPACE**

The Class **NP**

- NP $\rightarrow$ ~~Non-polynomial~~
- NP $\rightarrow$ Nondeterministic Polynomial Time

NP is the class of problems for which a solution can be **verified** in **polynomial time**, even though the solution may be **hard** to find

- NP $\rightarrow$ ~~Non-polynomial~~
- NP $\rightarrow$ Nondeterministic Polynomial Time

NP is the class of problems for which a solution can be **verified** in **polynomial time**, even though the solution may be **hard** to find

- NP → ~~Non-polynomial~~
- NP → Nondeterministic Polynomial Time

NP is the class of problems for which a solution can be **verified** in **polynomial time**, even though the solution may be **hard** to find

- The problem of recovering a secret key with a **known plaintext** is in **NP**
- What about the case when only ciphertext is known?
- Belongs in **P** or **NP**?

KPA Vs COA

- The problem of recovering a secret key with a **known plaintext** is in **NP**
- What about the case when only ciphertext is known?
- Belongs in **P** or **NP**?