Selected Areas in Cryptology
Cryptanalysis
Week 1

Marc Stevens

stevens@cwi.nl

https://homepages.cwi.nl/~stevens/mastermath/2021/

Some preliminaries

Cost of algorithms

Time complexity

= runtime

= number of unit operations
(unit: e.g. bit operation, cpu instruction, function call)

Memory complexity

= amount of unit storage (unit: e.g. bit, byte, block)

Asymptotic complexity functions

Parameter n (bitlength of the input / security parameter)

Definitions:

We write $f(n) = O(g(n))$ if
$|f(n)| \leq M\ g(n)$ for all $n \geq n_0$ (for some M, n_0)
(also the called order of the function,
only the asymptotically fastest growing term is relevant)

$$poly(n) := \{f(n) \colon \mathbb{R} \to \mathbb{R} \mid f(n) = O\big(n^d\big), d \in \mathbb{N}\}$$

(set of all functions that are asymptotically bounded by some polynomial)

$$f, g \in poly(n) \quad \Rightarrow \quad f + g,\ f \cdot g,\ f \circ g \in poly(n)$$

Probabilistic algorithms $A(x)$

Uses random coins, non-deterministic

For fixed input, output has probability distribution

PPT := Probabilistic Polynomial-Time

Notation

$x \xleftarrow{r} \mathcal{X}$ randomly sample from $\mathcal{X}$

assume uniform distribution if $\mathcal{X}$ is set

$\Pr[event]$ = probability event happens

$E[X]$ = the expected value for random variable X

Cryptographic constructions must asymptotically be

efficient: construction is PPT

secure: attacks should not be PPT

then for any desired gap factor G (e.g. $G = 2^{128}$)
there exists a n_0 such that for all $n \geq n_0$
cost of attack $\geq G \times$ cost of construction

Success probability for attacks A

= probability algorithm outputs correct solution $y \in Sol(x)$

$$p_{succ}^{A}(x) \coloneqq \Pr[y \leftarrow A(x) \wedge y \in Sol(x)]$$

Negligible success probability:

$$negl(n) \coloneqq \{f : \mathbb{R} \rightarrow \mathbb{R} \mid \forall d \in \mathbb{N} : \lim_{n \rightarrow \infty} f(n) \cdot n^{d} = 0\}$$

negligible functions go to 0 very quickly,
even when multiplied by any arbitrary polynomial function

E.g.: key guessing attack

- Simply try R random secret keys of n bits
- Finds correct key with probability $R \cdot 2^{-n}$
- Should be negligible
  - In concrete sense:
    so unlikely one can disregard this attack
  - As in asymptotic sense:
    $R \cdot 2^{-n} \in negl(n)$ if $R \in poly(n)$

One-Time Pad

Symmetric Encryption Schemes

Send message secretly from sender to receiver
Using pre-shared secret key K (unknown to adversary)

Sender encrypts plaintext P to ciphertext C

- Keyspace $\mathcal{K}$, plaintext space $\mathcal{P}$, ciphertext space $\mathcal{C}$
- Function $E_K: \mathcal{P} \to \mathcal{C}$ for $\mathrm{K} \in \mathcal{K}$
- $C = E_K(P)$

Receiver uses corresponding decryption to obtain plaintext P

- $P = D_K(C)$
- $D_K: \mathcal{C} \to \mathcal{P}$

Goals:

- Correctness: $D_K(E_K(P)) = P$ for all K, P
- Secrecy: without key K
  "no information is learned from C about message
  (formalization comes later)

One-Time Pad (OTP)

For any $l \in \mathbb{N}$:

$$\mathcal{K}_l = \mathcal{P}_l = \mathcal{C}_l = \{0,1\}^l \approx \mathbb{F}_2^l$$

$$E_K(P) := P \oplus K$$

$$D_K(C) := C \oplus K$$

$0 \oplus 0 = 1 \oplus 1 = 0$
$1 \oplus 0 = 0 \oplus 1 = 1$
Addition in $\mathbb{F}_2$

Requires K uniformly random selected

Key, and Plain- and ciphertext have equal length

Only encryption method providing perfect secrecy

no statistical correlation between cipher- and plaintext if key is unknown

$\Rightarrow$ no information can be learned

even with ∞ computing power

$$\Pr_{K}[C = P \oplus K] = \Pr_{K}[K = P \oplus C] = 2^{-l}$$

Given C, every plaintext is equally likely

Given P, every ciphertext is equally likely

One-Time Pad issues

Perfect secrecy, but …

Perfect secrecy is broken if

- Key K is not kept secret
- K was not selected uniformly at random from $\mathcal{K}_l$
- Key K is reused for two messages
  attacker learns: $C_1 \oplus C_2 = P_1 \oplus P_2$

Also malleable!

1. Sender encrypts $P =$ "I owe you 10$"
2. Attacker intercepts $C = K \oplus P$

   Let $D = \text{"I owe you 10\$"} \oplus \text{"I owe you 5k\$"}$
   $= \text{"\_\_\_\_\_\_\_\_10\_"} \oplus \text{"\_\_\_\_\_\_\_\_5k\_"}$

   Attacker doesn't even need to know the actual text, only the position and value of the change
3. Attacker sends $C' = C \oplus D$ to receiver
4. Receiver obtains C' and decrypts:
   $P' = K \oplus C' = P \oplus D = \text{"I owe you 5k\$"}$

Stream Ciphers

Stream ciphers

Operates very similar to One-Time Pad:

simply XOR message with long key stream

$$C := P \oplus \widetilde{K}$$

Except: long key stream $\widetilde{K}$ is generated from <u>short pre-shared key K</u>

Short key size: k bits, $\mathcal{K} = \{0,1\}^k$

Large internal state: s bits, $S_i \in \{0,1\}^s$

Arbitrary long plain- and ciphertext: $\mathcal{C} = \mathcal{P} = \{0,1\}^*$

1. Initialization:
   $$S_0 := Init(K)$$
2. Step update for bit i
   $$(S_{i+1}, O_i) := Update(S_i)$$
3. Encrypt bit i
   $$C_i := P_i \oplus O_i$$
4. Repeat 2-3 for $i = 0, \ldots |P| - 1$

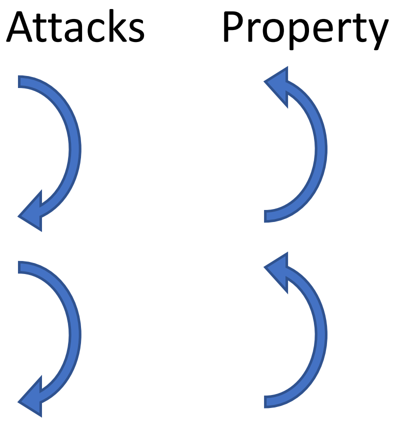
Asynchronous version (to recover from missed C_j)

$$(S_{i+1}, O_i) := Update(S_i, C_{i-1}, \ldots, C_{i-l})$$

No perfect secrecy

- Only 2^k possible long key streams
- For each plaintext, only 2^k possible ciphertexts
- For each ciphertext, only 2^k possible plaintexts
- For very long ciphertexts (with `meaningful` plaintext)
  - It's theoretical possible to try every key
  - $2^k - 1$ attempts should lead to `garbage`
  - The right key should lead to a `meaningful` plaintext
- Hence, security must be computational

Computational Security Properties for Stream Ciphers:



- Key recovery hardness
  Given long keystream $\widetilde{K}$ hard to recover key K
- State recovery hardness
  Given long keystream $\widetilde{K}$ hard to recover a state S_i
- Indistinguishability
  Attacker cannot distinguish between
  - a random long keystream $\widetilde{K}$ and
  - a uniformly random bitstring of same length

Indistinguishability

Security parameter: key size k (and state size: l)

Attacker: distinguisher algorithm A

- Gets access to $\mathcal{O}$:
  - Either, stream cipher oracle: $\mathcal{O}_{sc}$
    - Selects $K \xleftarrow{r} \{0,1\}^k$ uniformly at random
    - Outputs O_i on i-th query
  - Or, random oracle: $\mathcal{O}_{ur}$
    - Outputs random $O_i \xleftarrow{r} \{0,1\}$ on i-th query
  - Does not know which of the two it gets access to!
- Returns either 0 or 1

  think of the output as it's guess: stream cipher or random

Stream cipher is $(d(k), c(k), \epsilon(k))$-indistinguishable

if: $\left|\Pr[A^{\mathcal{O}_{sc}} = 1] - \Pr[A^{\mathcal{O}_{ur}} = 1]\right| \leq \epsilon(k)$

for all distinguishers A that:

- read at most $d(k)$ bits from oracle
- perform at most $c(k)$ operations

Concrete security	*Asymptotic security*	*Information-theoretic security*
$d(k), c(k) \coloneqq 2^{\min(k,l)}$ $\epsilon(k) \coloneqq 2^{-128}$	$d(k), c(k) \coloneqq poly(k)$ $\epsilon(k) \coloneqq negl(k)$	Impossible: $c(k) \coloneqq \infty$

Generic attacks

Generic Key recovery attack given target keystream $\tilde{T}$

1. Walk over the search space $K \in \{0,1\}^k$
2. Generate keystream $\widetilde{K}$ with same length as $\tilde{T}$
3. If $\widetilde{K} = \tilde{T}$ then return K
4. if no such K then return $\perp$

Time complexity: $O\left(2^k\right)$

Generic State recovery attack given target keystream $\tilde{T}$

1. Walk over the search space $S_0 \in \{0,1\}^l$
2. Generate keystream $\widetilde{K}$ with same length as $\tilde{T}$
3. If $\widetilde{K} = \tilde{T}$ then return S_0
4. If no such S_0 then return $\perp$

Time complexity: $O\left(2^l\right)$

Generic distinguishing attack

From generic key recovery attack or state recovery attack:

Return 0 if recovery attack returns $\perp$, and 1 otherwise

Time complexity: $O(2^{\min(k,l)})$

Stream Cipher Malleability

Again Stream Ciphers only provide secrecy

As ciphertexts can be precisely modified:

1. Sender encrypts $P =$ “I owe you 10$”
2. Attacker intercepts $C = K \oplus P$
   Let $D =$ "________10_" $\oplus$ "________5k_“
3. Attacker sends $C' = C \oplus D$ to receiver
4. Receiver obtains C' and decrypts:
   $P' = K \oplus C' = P \oplus D = \text{"I owe you 5k\$"}$

Key reuse

Key K must not be reused for two messages
otherwise attacker learns: $C_1 \oplus C_2 = P_1 \oplus P_2$

Common trick:

- Split k bit key into k_1 bits secret key and k_2 bits (public) nonce
- Pre share single secret key $K_1 \in \{0,1\}^{k_1}$
- Reuse K_1 by choosing different (public) values $K_2 \in \{0,1\}^{k_2}$
  - E.g.: maintain counter K_2 with $k_2 \geq 32$
  - Or choose random K_2, then $k_2 \geq 128$

Block Ciphers

- Block ciphers work differently from the one-time pad and stream ciphers
  - Only encrypts fixed-size blocks as a whole (not per bit)
  - Let security parameter n
  - Key space $\mathcal{K}(n)$ and block space $\mathcal{M}(n)$ (e.g., $\{0,1\}^n$)
  - $Enc: \mathcal{K}(n) \times \mathcal{M}(n) \rightarrow \mathcal{M}(n)$ such that
    - $Enc_K: \mathcal{M}(n) \rightarrow \mathcal{M}(n)$ is a permutation for all $K \in \mathcal{K}(n)$
    - $Dec_K \coloneqq Enc_K^{-1}$ is efficiently computable

  - Note: n is typically omitted: $\mathcal{K}, \mathcal{M}$

Generic attacks

Generic key recovery attack model

No relevant key stream to provide to attacker

Instead list of plaintext + ciphertext pairs: $(P_1, C_1), (P_2, C_2), \ldots$

Which pairs?

- Known plaintext attack: random plaintexts
- Chosen plaintext attack: attacker may choose P_i
- …

Generic key recovery attack

1. Query l pairs $(P_1, C_1), \ldots, (P_l, C_l)$
2. Walk over search space $K \in \mathcal{K}$
3. If $C_i = Enc_K(P_i)$ for $i = 1, \ldots, l$ then return K
4. Otherwise, if no such K, return $\perp$

Complexity: $O(|\mathcal{K}|)$

Note: Even if there are l pairs to check in total
The first is very likely to fail and the key candidate c
so most times we only have to check 1 pair

Generic 1-out-of-L key recovery attack

Assume L users with different keys $K_1, \ldots, K_L$

Attacker succeeds if it finds 1 key

Generic attack

1. Chooses l plaintexts $P_1, \ldots, P_l$
2. Queries encryptions for each user:

   $C_{i,j} = Enc_{K_j}(P_i)$ for $i = 1, \ldots, l$ and $j = 1, \ldots, L$
3. Walks over search space $K \in \mathcal{K}$
4. Compute $\tilde{C}_1 = Enc_K(P_1)$
5. For j such that $C_{1,j} = \tilde{C}_1$ do
6. If $C_{i,j} = Enc_K(P_i)$ for $i = 2, \ldots, l$ then return K
7. Otherwise, return $\perp$

Every key guess has success probability $L/|\mathcal{K}|$

Complexity: $O(|\mathcal{K}|/L)$

Speed up by factor L!

Attacks with precomputation

There are attacks that cost $O(|\mathcal{K}|)$ or more in total

But < $O(|K|)$ per problem instance

Two phases:

- An offline part that performs at least $O(|\mathcal{K}|)$ operations
- An online part that attacks each of the L keys independently

An extreme example, codebook dictionary:

Offline:
1. choose block B
2. create hash table with $(Enc_K(B), K)$ entries

Complexity: Time O($|\mathcal{K}|$), Memory: O($|\mathcal{K}|$)

Online:
1. for each secret key K_i to be attacked
2. Query $C = Enc_{K_i}(B)$
3. Find table entry (C, K_i)

Complexity: Time $O(1)$, Memory: $O(|\mathcal{K}|)$

Non-uniform attacks:

- Another view on an attack with pre-computation
- Make pre-computed data part of online attack algorithm
- Online algorithm now only has total cost $< O(|\mathcal{K}|)$
- Such algorithms called non-uniform

Hellman's Time-Memory trade off attack

An attack that uses more time, but less memory

Idea:

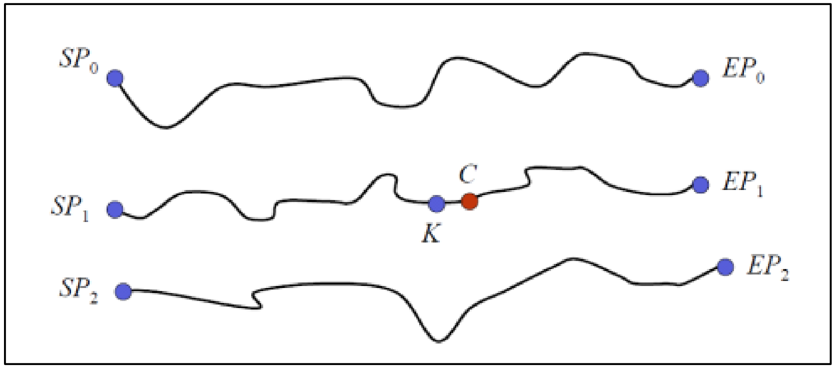
Use fixed block B and map $\phi: \mathcal{C} \to \mathcal{K}$

Iterative function $F: \mathcal{K} \to \mathcal{K}$ 'walks' through key space

$F(K_i) = K_{i+1}$, where $C = Enc_{K_i}(B)$, $K_{i+1} = \phi(C)$

Offline: store many long walks covering key space

Only store begin and endpoints $(SP_j, EP_j = F^t(SP_j))$



Online: query $C_0 = Enc_K(B)$, $K_0 = \phi(C_0)$

Note that: $F(K) = K_0$ by definition

Walk from K_0 until say endpoint EP_1 is found

Find secret K by walking from SP_1

Hellman's Time-Memory trade off attack

Setup Details:

$F: \mathcal{K} \rightarrow \mathcal{K}$, where $F = \phi \circ E$

$E: \mathcal{K} \rightarrow \mathcal{M}, \qquad E(K) \coloneqq Enc_K(B)$

$\phi: \mathcal{M} \rightarrow \mathcal{K}$ needs to be surjective

If $|\mathcal{M}| \geq |\mathcal{K}|$ then easy, otherwise impossible

When $|\mathcal{M}| < |\mathcal{K}|$

- Use multiple blocks $B_1, B_2, \dots, B_l$ such that $|\mathcal{M}|^l \geq |\mathcal{K}|$
- $E: \mathcal{K} \rightarrow \mathcal{M}^l, \quad E(K) \coloneqq \left(Enc_K(B_1), \dots, Enc_K(B_l)\right)$
- And surjective map $\phi: \mathcal{M}^l \rightarrow \mathcal{K}$

Hellman's Time-Memory trade off attack

Simplified version

Attack parameters

Number of walks: m
Length of each walk: t

Offline attack:

1. Choose $SP_1, \dots, SP_m$ uniformly at random from $\mathcal{K}$
2. Compute $EP_i = F^t(SP_i)$ for $i = 1, \dots, m$
3. Store (EP_i, SP_i) in hash table / sorted table

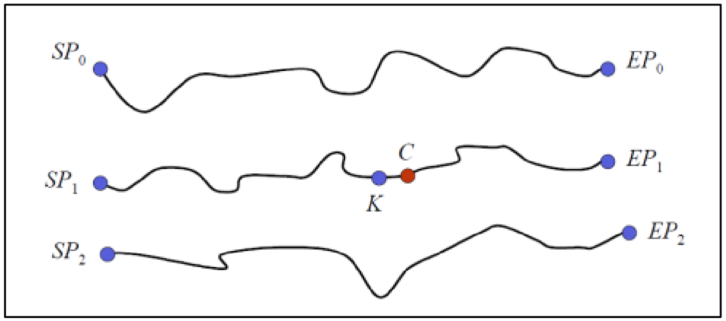
Online attack:

1. Given $C_0 = Enc_K(B)$ for some unknown key K
2. Let $P_0 = \phi(C_0)$
3. For $i = 0, \dots, t-1$
4.   If $P_i = EP_j$ for some j then
5.   Let $\widetilde{K} \coloneqq F^{t-i-1}(SP_j)$
6.   If $Enc_{\widetilde{K}}(B) = C_0$ then return $\widetilde{K}$
7.   Compute $P_{i+1} \coloneqq F(P_i)$
8. Otherwise, return $\perp$

Hellman’s Time-Memory trade off attack

Simplified version analysis

Ideally, use $m \cdot t = |\mathcal{K}|$ and hope to cover entire space

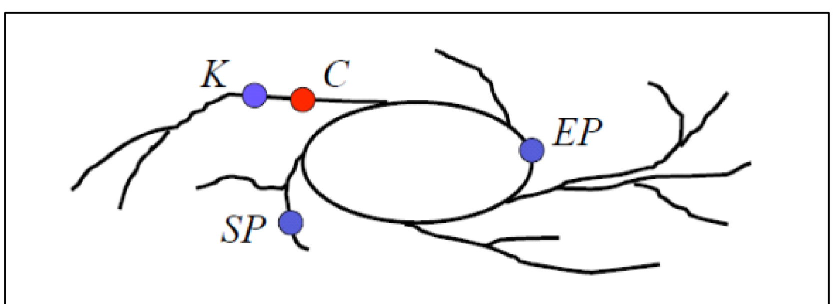


Ideal situation

However, F behaves as a random function

Many collisions $F(x) = F(y)$ exist and merges walks

- Substantial part of space is never reached
- Creates false alarms:
  - K does not actually lie on walk from SP_i,
    but on walk from another SP with same EP_i



Expected situa
random functi

Hellman’s Time-Memory trade off attack

Simplified version analysis

Collisions start to occur when $m \cdot t \approx \sqrt{|\mathcal{K}|}$

Due to the birthday paradox (covered later)

The expected number of collisions grows roughly quadratic in $m \cdot t$

False alarms analysis:

- Walk from P_0 has t points
- There are at most $m \cdot t$ points covered by the table
- Each pair has probability $1/|\mathcal{K}|$ to collide and cause false alarm (i.e. without P_0 actually being on the walk)
- Expected number of false alarms: $E[Z] \leq m \cdot t^2/|\mathcal{K}|$
- Expected costs of false alarm: $t \cdot E[Z] \leq m \cdot t^3/|\mathcal{K}|$

Success only if target K is covered (part of a walk from a SP_i)

Hellman: when $m \cdot t^2 = |\mathcal{K}|$, success probability is $\approx 0.80mt/|\mathcal{K}|$

Hellman's Time-Memory trade off attack

Improved version

Use r independent tables with different $\phi_1, \ldots, \phi_r$

Even if the same key is covered in different tables
then different ϕ_i imply different walks instead of merging walks

Hellman proposed $m = t = r = \sqrt[3]{|\mathcal{K}|}$

Individual table: success probability $\approx 0.80 \,/\, \sqrt[3]{|\mathcal{K}|}$

Total success probability ≈ 0.8

Offline time complexity: $O(mtr) = O(|\mathcal{K}|)$

Offline memory complexity: $O(rm) = O(|\mathcal{K}|^{2/3})$

Online complexity:

$$O(rt + rmt^3/|\mathcal{K}|) = O\left(|\mathcal{K}|^{2/3} + |\mathcal{K}|^{2/3}\right) = O(|\mathcal{K}|^{2/3})$$