COS 433/Math 473: Cryptography

Mark Zhandry

Princeton University

Fall 2020

Announcements/Reminders

- HW1 due September 15
- PR1 due October 6

Previously on COS 433…

Perfect Security for Multiple Messages

Definition: A stateless scheme **(Enc,Dec)** has **perfect secrecy for d messages** if, for any two sequences of messages $(m_0^{(i)})_{i\in[d]}$, $(m_1^{(i)})_{i\in[d]} \in M^d$

$$\left(\mathrm{Enc}(K,\ m_0^{(i)}\ )\right)_{i\in[d]} \overset{d}{=} \left(\mathrm{Enc}(K,\ m_1^{(i)}\ )\right)_{i\in[d]}$$

Notation: $(\ f(i)\ )_{i\in[d]} = (\ f(1),\ f(2),\ \dots,\ f(d)\ )$

Randomized Encryption

Syntax:

- Key space **K** (usually $\{0,1\}^\lambda$)
- Message space **M** (usually $\{0,1\}^n$)
- Ciphertext space **C** (usually $\{0,1\}^m$)
- **Enc: K×M → C** (potentially probabilistic)
- **Dec: K×C → M** (usually deterministic)

Correctness:

- For all $k \in K$, $m \in M$,

$$\Pr[\ \mathrm{Dec}(k, \mathrm{Enc}(k,m)\ ) = m\ ] \quad = \quad 1$$

Theorem: No stateless *randomized* encryption scheme can have perfect security for multiple messages

Proof of Easy Case

Let $\mathbf{(Enc,Dec)}$ be stateless, deterministic

Let $\mathbf{m_0^{(0)} = m_0^{(1)}}$
Let $\mathbf{m_1^{(0)} \neq m_1^{(1)}}$

Consider distributions of encryptions:

- $\mathbf{(c^{(0)} , c^{(1)}) = (Enc(K, m_0^{(0)}), Enc(K, m_0^{(1)}))}$

  $\Rightarrow \mathbf{c^{(0)} = c^{(1)}}$ (by determinism)

- $\mathbf{(c^{(0)} , c^{(1)}) = (Enc(K, m_1^{(0)}), Enc(K, m_1^{(1)}))}$

  $\Rightarrow \mathbf{c^{(0)} \neq c^{(1)}}$ (by correctness)

Generalize to Randomized Encryption

Let **(Enc,Dec)** be stateless, ~~deterministic~~

Let $\mathbf{m_0^{(0)} = m_0^{(1)}}$
Let $\mathbf{m_1^{(0)} \neq m_1^{(1)}}$

Consider distributions of encryptions:

- $\mathbf{(c^{(0)} , c^{(1)}) = (Enc(K, m_0^{(0)}), Enc(K, m_0^{(1)}))}$

  $\Rightarrow$ **????**

- $\mathbf{(c^{(0)} , c^{(1)}) = (Enc(K, m_1^{(0)}), Enc(K, m_1^{(1)}))}$

  $\Rightarrow \mathbf{c^{(0)} \neq c^{(1)}}$ (by correctness)

Generalize to Randomized Encryption

$(c^{(0)} , c^{(1)}) = (\mathrm{Enc}(K, m), \mathrm{Enc}(K, m))$

$\Pr[c^{(0)} = c^{(1)}]$?

- Fix $\mathbf{k}$
- Conditioned on $\mathbf{k}$, ciphertexts $c^{(0)}$ and $c^{(1)}$ are two independent samples from same distribution $\mathrm{Enc}(k, m)$

Lemma: Given any distribution **D** over a finite set **X**, $\Pr[Y=Y' : Y \leftarrow D, Y' \leftarrow D] \geq 1/|X|$

- Therefore, $\Pr[c^{(0)} = c^{(1)}]$ is non-zero

Generalize to Randomized Encryption

Let **(Enc,Dec)** be stateless, deterministic

Let $m_0^{(0)} = m_0^{(1)}$
Let $m_1^{(0)} \neq m_1^{(1)}$

Consider distributions of encryptions:

- $(c^{(0)} , c^{(1)}) = (\mathrm{Enc}(K, m_0^{(0)}), \mathrm{Enc}(K, m_0^{(1)}))$

  $\Rightarrow \Pr[c^{(0)} = c^{(1)}] > 0$

- $(c^{(0)} , c^{(1)}) = (\mathrm{Enc}(K, m_1^{(0)}), \mathrm{Enc}(K, m_1^{(1)}))$

  $\Rightarrow \Pr[c^{(0)} = c^{(1)}] = 0$

Today: Relaxing Perfect Secrecy

What do we do now?

Tolerate tiny probability of distinguishing

- If $\Pr[c^{(0)} = c^{(1)}] = 2^{-128}$, in reality never going to happen

How Small Is Ok?

Practice:

- Something unlikely to happen in lifetime of data/person/civilization/universe

- Typically something like 2^{-80}, 2^{-128}, or maybe 2^{-256}
  - Being struck by lightning twice: 2^{-23}
  - Winning the lottery: 2^{-26}
  - World-ending asteroid while on this slide: 2^{-46}

How Small Is Ok?

Theory:

- Maybe things will change as technology improves
- Want a more conceptual answer
- Absolute constants unsatisfactory
- Instead, use “negligible” functions

Negligible functions

Def: A function $\mathbf{f}$ is **polynomial** if $\mathbf{f(n) = O(n^c)}$ for some constant $\mathbf{c}$

Def: A function $\mathbf{g}$ is **super-polynomial** if, for every polynomial $\mathbf{f}$, $\mathbf{f(n) = O(g(n))}$

Def: A function $\mathbf{p}$ is **inverse polynomial** if $\mathbf{1/p(n)}$ is polynomial

Def: A function $\boldsymbol{\varepsilon}$ is **negligible** if, for every inverse polynomial $\mathbf{p}$, $\boldsymbol{\varepsilon}\mathbf{(n) = O(p(n))}$

(equivalently, $\mathbf{1/}\boldsymbol{\varepsilon}$ is super-polynomial)

Examples

2^n	super-polynomial
$n^{-n/7}$	negligible
$3^{-5\log n}$	inverse polynomial ($= n^{-5\log 3}$)
$1.5^{-\sqrt[3]{n}}$	negligible
$8^{\log^3 n}$	super-polynomial ($= n^{(\log 8)(\log^2 n)}$)
$(\log n)/n$	inverse polynomial

Security Parameter **λ**

Additional input to system, dictates “security level”

Key, message, ciphertext size all **polynomial** in **λ**

Probability of adversary success is **negligible** in **λ**

Defining Encryption Again

Syntax:

- Key space $\mathbf{K_\lambda}$
- Message space $\mathbf{M_\lambda}$
- Ciphertext space $\mathbf{C_\lambda}$
- **Enc:** $\mathbf{K_\lambda \times M_\lambda \rightarrow C_\lambda}$ (potentially randomized)
- **Dec:** $\mathbf{K_\lambda \times C_\lambda \rightarrow M_\lambda}$

Correctness:

- $\mathbf{\log|K_\lambda|, \log|M_\lambda|, \log|C_\lambda|}$ polynomial in $\boldsymbol{\lambda}$
- For all $\boldsymbol{\lambda}$, $\mathbf{k \in K_\lambda,\ m \in M_\lambda,}$

$$\mathbf{Pr[Dec(k, Enc(k,m)) = m] = 1}$$

Statistical Distance

Given two distributions $\mathbf{D_1}$, $\mathbf{D_2}$ over a set $\mathbf{X}$, define

$$\Delta(D_1,D_2) = ½\Sigma_x \mid \Pr[D_1=x] - \Pr[D_2=x] \mid$$

Observations:

$$0 \leq \Delta(D_1,D_2) \leq 1$$

$$\Delta(D_1,D_2) = 0 \iff D_1 \overset{d}{=} D_2$$

$$\Delta(D_1,D_2) \leq \Delta(D_1,D_3) + \Delta(D_3,D_2)$$

(**Δ** is a metric)

Another View of Statistical Distance

Theorem: $\Delta(D_1,D_2) \geq \varepsilon$ iff $\exists$ (potentially randomized) A s.t.

$$| \Pr[A(D_1) = 1] - \Pr[A(D_2) = 1] | \geq \varepsilon$$

Terminology: for any A,

$$| \Pr[A(D_1) = 1] - \Pr[A(D_2) = 1] |$$

is called the "advantage" of A in distinguishing D_1 and D_2

Another View of Statistical Distance

Theorem: $\Delta(D_1,D_2) \geq \varepsilon$ iff $\exists$ (potentially randomized) A s.t.

$$\left| \Pr[A(D_1) = 1] - \Pr[A(D_2) = 1] \right| \geq \varepsilon$$

To lower bound Δ, just need to show adversary A with that advantage

Examples

D_1 = Uniform distribution over $\{0,1\}^n$

- $\Pr[D_1=x] = 2^{-n}$

D_2 = Uniform conditioned on even parity

- $\Pr[D_2=x] = 2^{-(n-1)}$ if x has even parity, 0 otherwise

$$\begin{aligned}\Delta(D_1,D_2) &= \tfrac{1}{2}\Sigma_{\text{even }x}\ |2^{-n} - 2^{-(n-1)}| \\ &\quad + \tfrac{1}{2}\Sigma_{\text{odd }x}\ |2^{-n} - 0| \\ &= \tfrac{1}{2}\Sigma_{\text{even }x}\ 2^{-n} + \tfrac{1}{2}\Sigma_{\text{odd }x}\ 2^{-n} \\ &= \tfrac{1}{2}\end{aligned}$$

Examples

$D_1 =$ Uniform over $\{1,\ldots,n\}$

$D_2 =$ Uniform over $\{1,\ldots,n+1\}$

$$\Delta(D_1,D_2) = \tfrac{1}{2}\sum_{x=1}^{n} |1/n - 1/(n+1)| + \tfrac{1}{2}\,|0 - 1/(n+1)|$$

$$= \tfrac{1}{2}\sum_{x=1}^{n} 1/n(n+1) \;+\; \tfrac{1}{2}\,1/(n+1)$$

$$= \tfrac{1}{2}\,1/(n+1) + \tfrac{1}{2}\,1/(n+1) = 1/(n+1)$$

Statistical Security (Concrete)

Definition: A scheme **(Enc,Dec)** has **ε-statistical secrecy for d messages** if $\forall$ two sequences of messages $(m_0^{(i)})_{i\in[d]}$, $(m_1^{(i)})_{i\in[d]} \in M^d$

$$\Delta\left[\ \left(\mathrm{Enc}(K,\ m_0^{(i)}\ )\right)_{i\in[d]},\quad \left(\mathrm{Enc}(K,\ m_1^{(i)}\ )\right)_{i\in[d]}\ \right] < \varepsilon$$

We will call such a scheme **(d,ε)** statistically secure

Statistical Security (Asymptotic)

Definition: A scheme **(Enc,Dec)** has **statistical secrecy for d messages** if $\exists$ negligible **ε** such that $\forall$ two sequences $(m_0^{(i)})_{i\in[d]}$, $(m_1^{(i)})_{i\in[d]}$ $\in$ M_λ^d,

$$\Delta\Big[\ \big(\mathrm{Enc}(K_\lambda,\ m_0^{(i)}\ )\big)_{i\in[d]},$$

$$\big(\mathrm{Enc}(K_\lambda,\ m_1^{(i)}\ )\big)_{i\in[d]}\ \Big] < \varepsilon(\lambda)$$

We will call such a scheme **d**-time statistically secure

Stateless Encryption with Multiple Messages

Ex:

$M = C$
$K = \text{Perms}(M)$
$\text{Enc}(K, m) = K(m)$
$\text{Dec}(K, c) = K^{-1}(c)$

Q: Is this statistically secure for two messages?

Theorem: For any **ε<1**, no stateless *deterministic* encryption scheme can have **ε**-statistical security for **2** messages

(Proof basically the same as before)

Importantly: proof does **not** hold for randomized schemes for **ε>0**

Stateless Encryption with Multiple Messages

Ex:

$r \leftarrow R$

$C = M \times R$
$K = \text{Perms}(C)$
$\text{Enc}(K, m) = K(m, r)$
$\text{Dec}(K, c) = (m', r') \leftarrow K^{-1}(c)$, output m'

Q: Is this statistically secure for two messages?

Q: Is it practical?

Example

A more efficient example:

$M = \mathbb{Z}_p$ (p a prime of size 2^λ, $\lambda=128$)

$C = \mathbb{Z}_p{}^2$

$K = \mathbb{Z}_p{}^2$

Random in $\mathbb{Z}_p$

$\mathrm{Enc}(\ (a,b),\ m) = (r,\ \ (ar+b) + m\ )$

$\mathrm{Dec}(\ (a,b),\ (r,c)\ ) = c - (ar+b)$

Proof of Example

Let $\mathbf{D_b}$ be distribution of $\mathbf{(\ Enc(k,m_b^{(i)})\ )_{i\in\{0,1\}}}$
Let $\mathbf{D_b'}$ be the following:

1. Run $\mathbf{(c_0,c_1)\leftarrow Db}$
2. If $\mathbf{r_0=r_1}$, output $\perp$
3. Else output $\mathbf{(c_0,c_1)}$

Fix $\mathbf{r_0\neq r_1,\ m_0,m_1,c_0,c_1}$

$$\Pr_{(a,b)}[ar_0+b+m_0=c_0,\ ar_1+b+m_1=c_1] = 1/p^2$$

So $\mathbf{D_0' \overset{d}{=} D_1'}$ $\mathbf{(\ \Delta(D_0',\ D_1') = 0\ )}$

The Symbol ⊥ (“bot”)

Represents an abort/reject/bad outcome

Augments whatever set we are talking about

- Ex: support of $\mathbf{D_b = C^2}$, so support of $\mathbf{D_b' = C^2} \cup \mathbf{\{\perp\}}$

Proof of Example

Lemma: $\Delta(D_1,D_2) \leq \frac{1}{2}\Pr[\text{bad}|D_1] + \frac{1}{2}\Pr[\text{bad}|D_2] + \Delta(D_1' , D_2')$

Where:

- "**bad**" is some event
- $\Pr[\text{bad}|D_b]$ is probability "**bad**" when sampling from D_b
- D_b' is D_b, except outputs $\perp$ on "**bad**"

Proof of Lemma

$$\Delta(D_1,D_2) = \tfrac{1}{2}\Sigma_x|\ \Pr[D_1=x] - \Pr[D_2=x]\ |$$

$$= \tfrac{1}{2}\Sigma_{x:bad}|\ \Pr[D_1=x\ ] - \Pr[D_2=x\ ]\ |$$
$$+ \tfrac{1}{2}\Sigma_{x:good}|\ \Pr[D_1=x\ ] - \Pr[D_2=x\ ]\ |$$

$$\leq \tfrac{1}{2}\Sigma_{x:bad}|\ \Pr[D_1=x\ ]\ | + \tfrac{1}{2}\Sigma_{x:bad}|\ \Pr[D_2=x\ ]\ |$$
$$+ \tfrac{1}{2}\Sigma_{x:good}|\ \Pr[D_1=x] - \Pr[D_2=x]\ |$$

$$= \tfrac{1}{2}\Sigma_{x:bad}|\ \Pr[D_1=x\ ]\ | + \tfrac{1}{2}\Sigma_{x:bad}|\ \Pr[D_2=x\ ]\ |$$
$$+ \tfrac{1}{2}\Sigma_x|\ \Pr[D_1'=x] - \Pr[D_2'=x]\ |$$

$$= \tfrac{1}{2}\ \Pr[bad|D_1] + \tfrac{1}{2}\ \Pr[bad|D_2] + \Delta(D_1',D_2')$$

Proof of Example

Let $\mathbf{D_b}$ be distribution of $\mathbf{(\ \ Enc(k,m_b^{(i)})\ \ )_{i\in\{0,1\}}}$
Let **bad** be when $\mathbf{r_0=r_1}$
Let $\mathbf{D_b}$' be the following:

1. Run $\mathbf{(c_0,c_1)\leftarrow Db}$
2. If **bad**, output ⊥
3. Else output $\mathbf{(c_0,c_1)}$

$\mathbf{Pr[bad|D_b] = 1/p}$
$\mathbf{\Delta(D_0', D_1') = 0}$

Therefore, $\mathbf{\Delta(D_0, D_1) \leq 1/p \approx 2^{-\lambda}}$

Summary so Far

Stateless encryption for multiple messages ✓

But, key length grows with number of messages ✘

And, key length grows with length of message ✘

Limits of Statistical Security

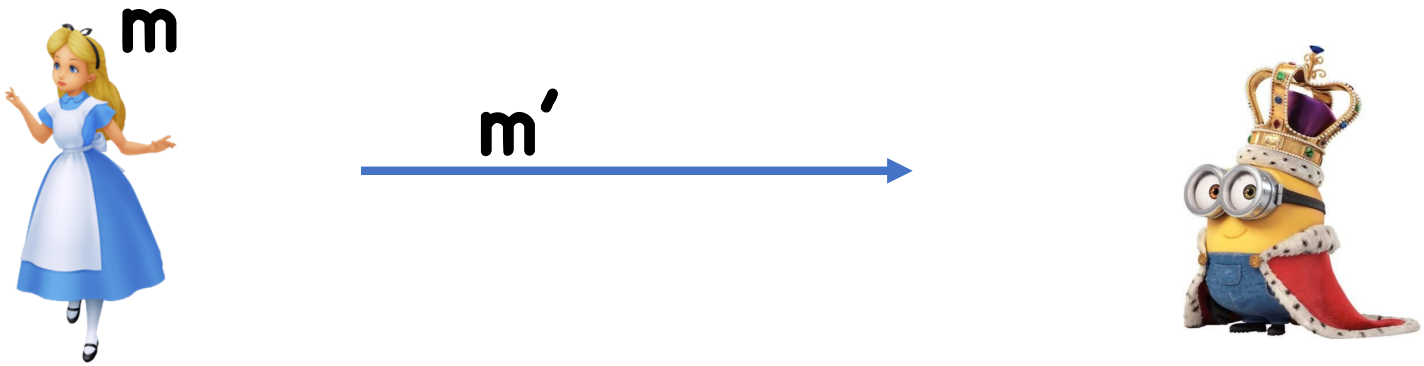
Theorem: Suppose **(Enc,Dec)** has plaintext space $\mathbf{M} = \{0,1\}^n$ and key space $\mathbf{K} = \{0,1\}^t$. Moreover, assume it is **(d, 0.4999)**-secure. Then:

$$t \geq d\ n$$

In other words, the key must be at least as long as the total length of all messages encrypted

Proof Idea: Compression

Use an encryption protocol to build a compression protocol



m

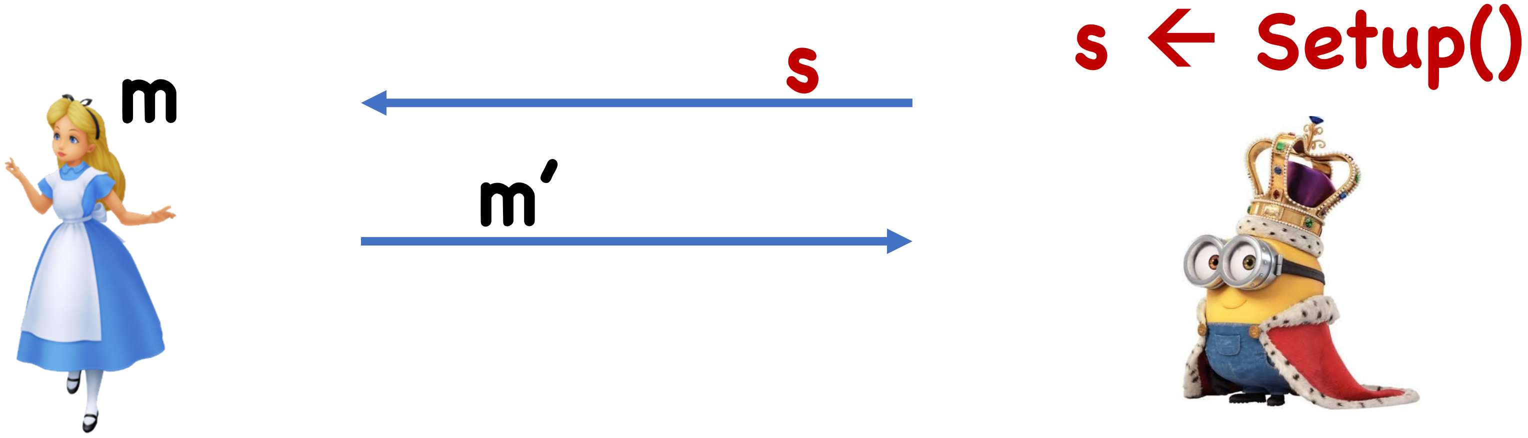
m'

$m' \leftarrow \text{Comp}(m)$

$m \leftarrow \text{Decomp}(m')$

Goal: $|m'| < |m|$

For Now: Easier Goal



m

s ← Setup()

s

m′

m′ ← Comp(s,m)

m ← Decomp(s,m′)

Goal: $|m'| < |m|$

The Protocol

Let $\mathbf{m_0}$ be some arbitrary message in $\mathbf{M}$

Setup():

- Choose random $\mathbf{k_0 \leftarrow K}$
- **Let** $\mathbf{c_1 \leftarrow Enc(k_0,m_0), \ldots, c_d \leftarrow Enc(k_0,m_0)}$
- **Output** $\mathbf{(c_1,\ldots,c_d)}$

In $\mathbf{M^d}$

Comp($\mathbf{(c_1,\ldots,c_d), (m_1,\ldots,m_d)}$ **):**

- Find $\mathbf{k,r_1,\ldots,r_d}$ such that $\mathbf{c_i=Enc(k,m_i;\ r_i)}\ \ \forall \mathbf{i}$
- If no such values exist, abort
- Output $\mathbf{k}$

The Protocol

Let $\mathbf{m_0}$ be some message in $\mathbf{M}$

In $\mathbf{M}^d$

$\mathbf{Comp(\ (c_1,\ldots,c_d),\ (m_1,\ldots,m_d)\ ):}$
- Find $\mathbf{k,r_1,\ldots,r_d}$ such that $\mathbf{c_i{=}Enc(k,m_i;\ r_i)}\ \forall\,\mathbf{i}$
- If no such values exist, abort
- Output $\mathbf{k}$

$\mathbf{Decomp((c_1,\ldots,c_d),\ k\ ):}$
- Compute $\mathbf{m_i = Dec(k,c_i)}$
- Output $\mathbf{(m_1,\ldots,m_d)}$

Analysis of Protocol

If **Comp** succeeds, **Decomp** must succeed by correctness

- Since $\mathbf{c_i{=}Enc(k,m_i;\ r_i)}$, $\mathbf{Dec(k,c_i)}$ must give $\mathbf{m_i}$

Therefore, must figure out when **Comp** succeeds

Claim: For any sequence of messages $\mathbf{m_1,\ldots,m_d}$, **Comp** succeeds with probability at least $\mathbf{1-\varepsilon}$

(Probability over the randomness used by **Setup()**)

Claim: For any sequence of messages $m_1,\ldots,m_d$, **Comp** succeeds with probability at least $1-\varepsilon$

Proof:

- Suppose **Comp** succeeds with probability $1-p$ for messages $m_1,\ldots,m_d$
- Let $A(c_1,\ldots,c_d)$ be the algorithm that runs $\mathrm{Comp}((c_1,\ldots,c_d),\ (m_1,\ldots,m_d))$ and outputs **1** if **Comp** succeeds

- If $c_i = \mathrm{Enc}(k_0,m_i)$, then $\Pr[A(c_1,\ldots,c_d)=1] = 1$
- If $c_i = \mathrm{Enc}(k_0,m_0)$, then $\Pr[A(c_1,\ldots,c_d)=1] = 1-p$

- By (d,ε)-statistical security of **Enc**, p must be $\leq\varepsilon$

Claim: For any sequence of messages $\mathbf{m_1,\ldots,m_d}$, **Comp** succeeds with probability at least $\mathbf{1-\varepsilon}$

Claim: For **a random** sequence of messages $\mathbf{m_1,\ldots,m_d}$, **Comp** succeeds with prob at least $\mathbf{1-\varepsilon}$

(Probability over the randomness used by **Setup()** and the random choices of $\mathbf{m_1,\ldots,m_d}$)

Next step: Removing Setup

We know:

$$\Pr\left[\text{Comp succeeds}: \begin{array}{l}(c_1,\ldots,c_d)\leftarrow\text{Setup}(),\\ m_i\leftarrow M\end{array}\right] \geq 1-\varepsilon$$

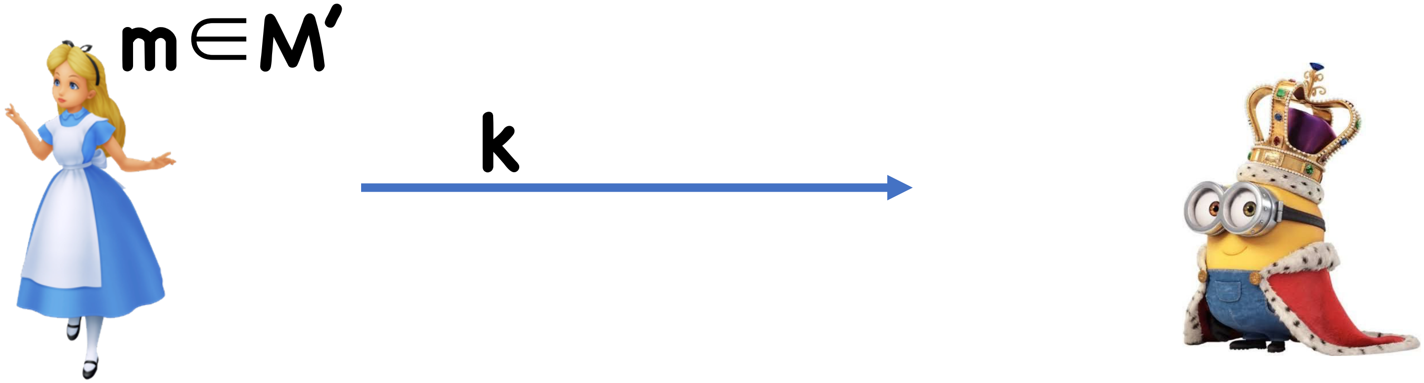
$\Longrightarrow$ there must exist *some* $(c_1^*,\ldots,c_d^*)$ such that

$$\Pr[\text{Comp succeeds}: m_i\leftarrow M] \geq 1-\varepsilon$$

Fix $(c_1^*,\ldots,c_d^*)$, define: $M' = \{(m_1,\ldots,m_d)$: Comp succeeds$\}$

- Note that $|M'| \geq (1-\varepsilon)\ |M|^d$

The Protocol



Find $k,r_1,\dots,r_d$ such that
$c_i^*=\mathrm{Enc}(k,m_{i;}\ r_i)\ \forall i$

For each i,

Let $m_i\leftarrow\mathrm{Dec}(k,c_i^*)$

Output $(m_1,\dots,m_d)$

By previous analysis,

- Alice always successfully compresses
- Bob always successfully decompresses

Final Touches

Can compress messages in $\mathbf{M'}$ into keys in $\mathbf{K}$

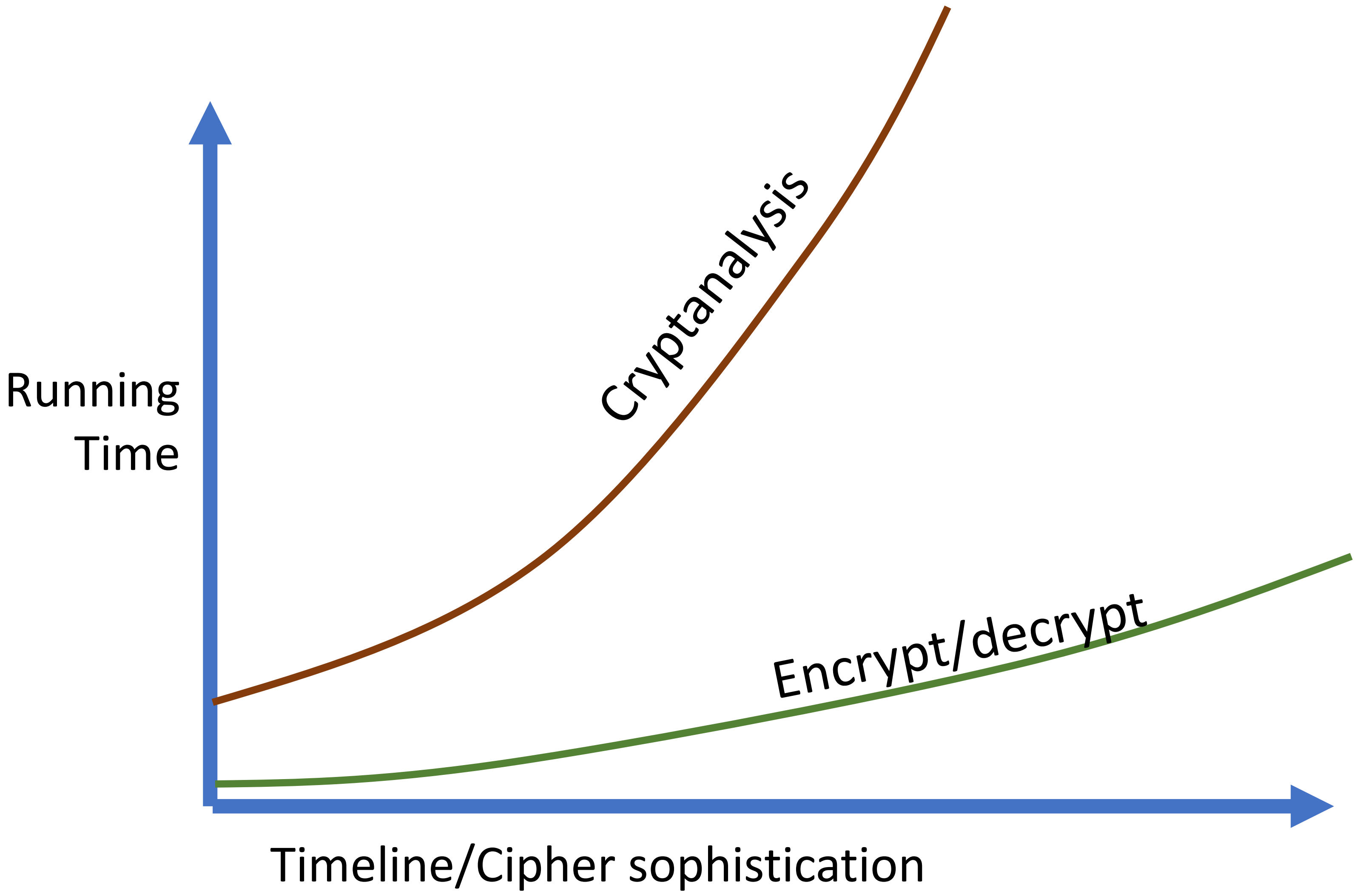
Therefore, it must be that $|\mathbf{M'}| \leq |\mathbf{K}|$

Meaning
$$\begin{aligned} t &= \log |K| \\ &\geq \log |M'| \\ &\geq \log [\ (1-\varepsilon)\ |M|^d\ ] \\ &= d \log |M| + \log [1-\varepsilon] \\ &= dn + \log [1-\varepsilon] \\ &\geq dn \ \text{(as long as } \varepsilon<1/2\text{)} \end{aligned}$$

Takeaway

If you don’t want to physically exchange keys frequently, you cannot obtain statistical security

So, now what?


Running Time
Cryptanalysis
Encrypt/decrypt
Timeline/Cipher sophistication

Computational Security

We are ok if adversary takes a really long time

Only considered attack for adversaries that don’t take too long

How Long Is Ok?

Practice:

- Lifetime of data/person/civilization/universe

- Typically something like 2^{80}, 2^{128}, or maybe 2^{256}
  - Lifetime of universe in nanoseconds: 2^{58}
  - Number of atoms in known universe: 2^{265}

How Long Is Ok?

Theory:

- Maybe things will change as technology improves
- Want a more conceptual answer
- Absolute constants unsatisfactory
- Instead, consider an attack if time bounded by polynomial function

Brute Force Attacks

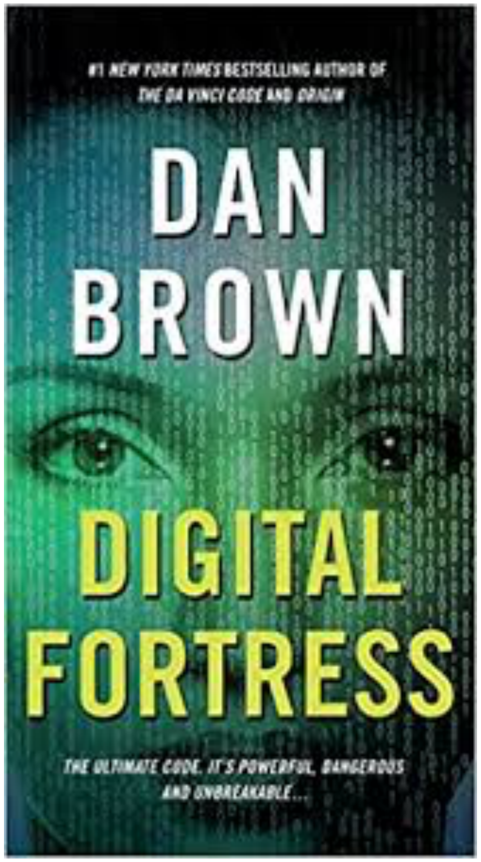
Simply try every key until find right one

If keys have length **λ**, **2^λ** is upper bound on attack

Not always applicable – requires being able to test when guess was correct

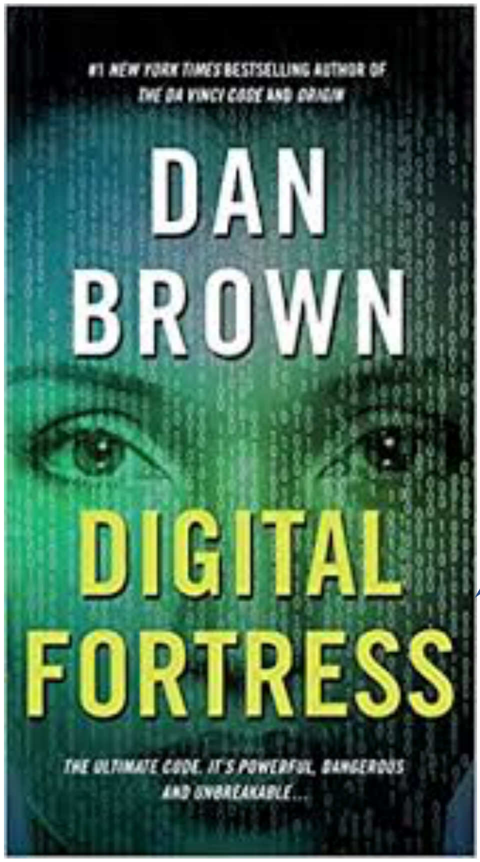
- Always applicable when **$|\text{key}| \leq |\text{message}|$**

Holiwudd Criptoe!


#1 NEW YORK TIMES BESTSELLING AUTHOR OF
THE DA VINCI CODE AND ORIGIN
DAN
BROWN
DIGITAL
FORTRESS
THE ULTIMATE CODE. IT'S POWERFUL, DANGEROUS
AND UNBREAKABLE...

[TRANSLTR]’s three million processors would all work in parallel ... trying every new permutation as they went

Holiwudd Criptoe!


#1 NEW YORK TIMES BESTSELLING AUTHOR OF
THE DA VINCI CODE AND ORIGIN
DAN BROWN
DIGITAL FORTRESS
THE ULTIMATE CODE. IT'S POWERFUL, DANGEROUS
AND UNBREAKABLE...

“What’s the longest you’ve ever seen TRANSLTR take to break a code?”

“About an hour, but it had a ridiculously long key—ten thousand bits”

Reminders

- HW1 due September 15
- PR1 due October 6