Provable Security

Abrahim Ladha

September 28, 2020

Motivation

- Old school protocol design is like current software development
- Release it and hope its fine
- Someone found a bug?
- patch, rerelease, repeat
- This is actually really bad. We want to be able to "prove" something is security

Security

- We say something is secure if it is not insecure
- When is something insecure?
- We can mathematically model adversaries
- Different kinds, with different power
- Four kinds of proof techniques:
- Game based, Hash functions, Simulators for MPC.

Definitions

- Define an encryption scheme as a tuple of algorithms
- $(pk, sk) \leftarrow \mathcal{G}(1^\kappa)$
- $c \leftarrow E_{pk}(m)$
- $m \leftarrow D_{sk}(c)$
- Here κ is a "security parameter". A measure of hardness. In this case, the length of the keys.
- It is given in unary so the time is poly in size of the input

Definitions

- "A good disguise does not reveal the persons height".
- Given an adversary (PPTM), we want the most optimal algorithm to only be that they can guess the key
- $P[pk' \leftarrow A(pk, c)$ such that $m' = D_{pk'}(c)$ and $m' = m] = 2^{-k}$ (?)

guessing

Definitions

- "A good disguise does not reveal the persons height".
- Given an adversary (PPTM), we want the most optimal algorithm to only be that they can guess the key
- $P[pk \leftarrow A(pk, c)$ such that $m' = D_{pk'}(c)$ and $m' = m] = 2^{-k}$ (?)
- Want to not just be able to reveal whole message, but reveal any information about the message.

Indistinguishability

- We play a game
- Let the adversary choose $m_0, and m_1$
- we give them $E_{sk}(m_0), E_{sk}(m_1)$.
- If they can tell which encryption is which, then they can "distinguish" between them.
- The better they are at guessing which is which, then the better the adversaries "Advantage"

Formalization of Experiments

- Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$
- Let $Exp_{\mathcal{SE}}^{ind-cpa-b}$ be the experiment where the adversary submits ticket (M_0, M_1), we encrypt and send $E_k(M_b)$
- $Adv_{\mathcal{SE}}^{ind-cpa}(A) = |Pr[Exp_{\mathcal{SE}}^{ind-cpa-1} = 1] - Pr[Exp_{\mathcal{SE}}^{ind-cpa-0} = 1]|$
- **Define $\mathcal{SE}$ secure if** $Adv(A) < \mu(n)$
- Here, $\mu(n)$ is a negligible function, $\mu(n) < \frac{1}{poly(n)}$ where $poly(n)$ is any polynomial.
- $Adv(A) = 2 \cdot Pr[b' \leftarrow A \text{ and } b = b'] - 1$
- If they totally guess,

Formalization of Experiments

- Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$
- Let $Exp_{\mathcal{SE}}^{ind-cpa-b}$ be the experiment where the adversary submits ticket (M_0, M_1), we encrypt and send $E_k(M_b)$
- $Adv_{\mathcal{SE}}^{ind-cpa}(A) = \left| Pr[Exp_{\mathcal{SE}}^{ind-cpa-1} = 1] - Pr[Exp_{\mathcal{SE}}^{ind-cpa-0} = 1] \right|$
- **Define $\mathcal{SE}$ secure if** $Adv(A) < \mu(n)$
- Here, $\mu(n)$ is a negligible function, $\mu(n) < \frac{1}{poly(n)}$ where $poly(n)$ is any polynomial.
- $Adv(A) = 2 \cdot Pr[b' \leftarrow A \text{ and } b = b'] - 1$
- If they totally guess, $Adv(A) = 2 \cdot (\frac{1}{2}) - 1 = 0$

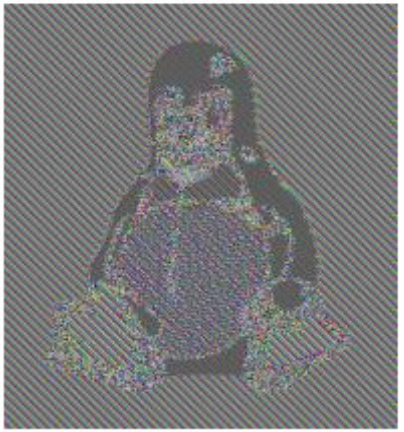
$0 \leq Adv(A) \leq 1$

ECB

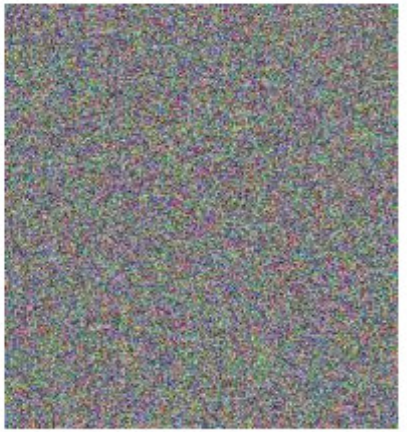
- Electronic Codebook
- Break up our message $m = m_1||m_2||...||m_n$
- $E_k(m) = E_k(m_1)||E_k(m_2)||...||E_k(m_n)$ where E is secure.
- Old world thinking. Caesar ciphers behave like this
- Horribly insecure. But why?

Tux

Original



Encrypted using ECB mode



Modes other than ECB result in pseudo-randomness

ECB is not secure

- Recall for ECB, $E_k(m_0||m_1) = E_k(m_0)||E_k(m_1)$
- Let (M_0, M_1) submitted by A be $(m_0||m_0, m_0||m_1)$
- A receives $E_k(m_b) = m_2||m_3$.
- if $m_2 = m_3$, output guess $b = 0$
- else output $b = 1$
- $Adv_{\mathcal{SE}}^{ind-cpa}(A) = Pr[Exp_{\mathcal{SE}}^{ind-cpa-1} = 1] - Pr[Exp_{\mathcal{SE}}^{ind-cpa-0} = 1]$
- $Adv_{\mathcal{SE}}^{ind-cpa}(A) = 1 - 0 \rightarrow 1$

Triple Elgamal, Moscow Election

- Choose three groups $G_1, G_1, G_3 = \langle g_1 \rangle, \langle g_2 \rangle, \langle g_3 \rangle$
- Gen(1^κ): (sk_1,sk_2,sk_3,pk_1,pk_2,pk_3)
- $(x_1, x_2, x_3, g_1^{x_1}, g_2^{x_2}, g_3^{x_3})$
- $(a_1, b_1) = \mathrm{Enc}_{sk_1, g_1}(m) = (g_1^r, g_1^{x_1 r} m)$
- $(a_2, b_2) = \mathrm{Enc}_{sk_1, g_1}(a_1)$
- $(a_3, b_3) = \mathrm{Enc}_{sk_1, g_1}(a_2)$
- MultEnc(m) $= (b_1, b_2, a_3, b_3)$
- Decryption is done in reverse

Triple Elgamal, Moscow Election

- Choose three groups $G_1, G_1, G_3 = \langle g_1 \rangle, \langle g_2 \rangle, \langle g_3 \rangle$
- Gen(1^κ): (sk_1,sk_2,sk_3,pk_1,pk_2,pk_3)
- $(x_1, x_2, x_3, g_1^{x_1}, g_2^{x_2}, g_3^{x_3})$
- $(a_1, b_1) = \mathrm{Enc}_{sk_1,g_1}(m) = (g_1^r, g_1^{x_1 r} m)$
- $(a_2, b_2) = \mathrm{Enc}_{sk_1,g_1}(a_1)$
- $(a_3, b_3) = \mathrm{Enc}_{sk_1,g_1}(a_2)$
- MultEnc(m) $= (b_1, b_2, a_3, b_3)$
- Decryption is done in reverse
- Guessing the key size should be $2^{3\kappa}$, but instead its $3 \cdot 2^\kappa$

IND-CCA2

- Indistinguishability under chosen *ciphertext* attacks
- Lets take the everything from IND-CPA
- The only thing we will change, we allow the adversary to make a polynomial number of queries to an encryption oracle
- The adversary can submit M to the oracle, who will return $E_k(M)$.

Deterministic Protocols

- Let $\mathcal{SE}$ is deterministic,
- A submits ticket (M_0, M_1)
- A send M_0 to its oracle and receives $c = E_k(M_0)$
- A receives $E_k(M_b)$
- if $E_k(M_b) = c \quad (= E_k(M_0))$, return $b = 0$, else $b = 1$

Deterministic Protocols

- Let $\mathcal{SE}$ is deterministic,
- A submits ticket (M_0, M_1)
- A send M_0 to its oracle and receives $c = E_k(M_0)$
- A receives $E_k(M_b)$
- if $E_k(M_b) = c \quad (= E_k(M_0))$, return $b = 0$, else $b = 1$
- IRL RSA encryption is done not as M^e, but something equivalent to $(M||r)^e$ for $r \xleftarrow{\$} \mathbb{Z}_a$
- Decrypt as $((M||r)^e)^d = M||r$ then shave off r.

A good definition?

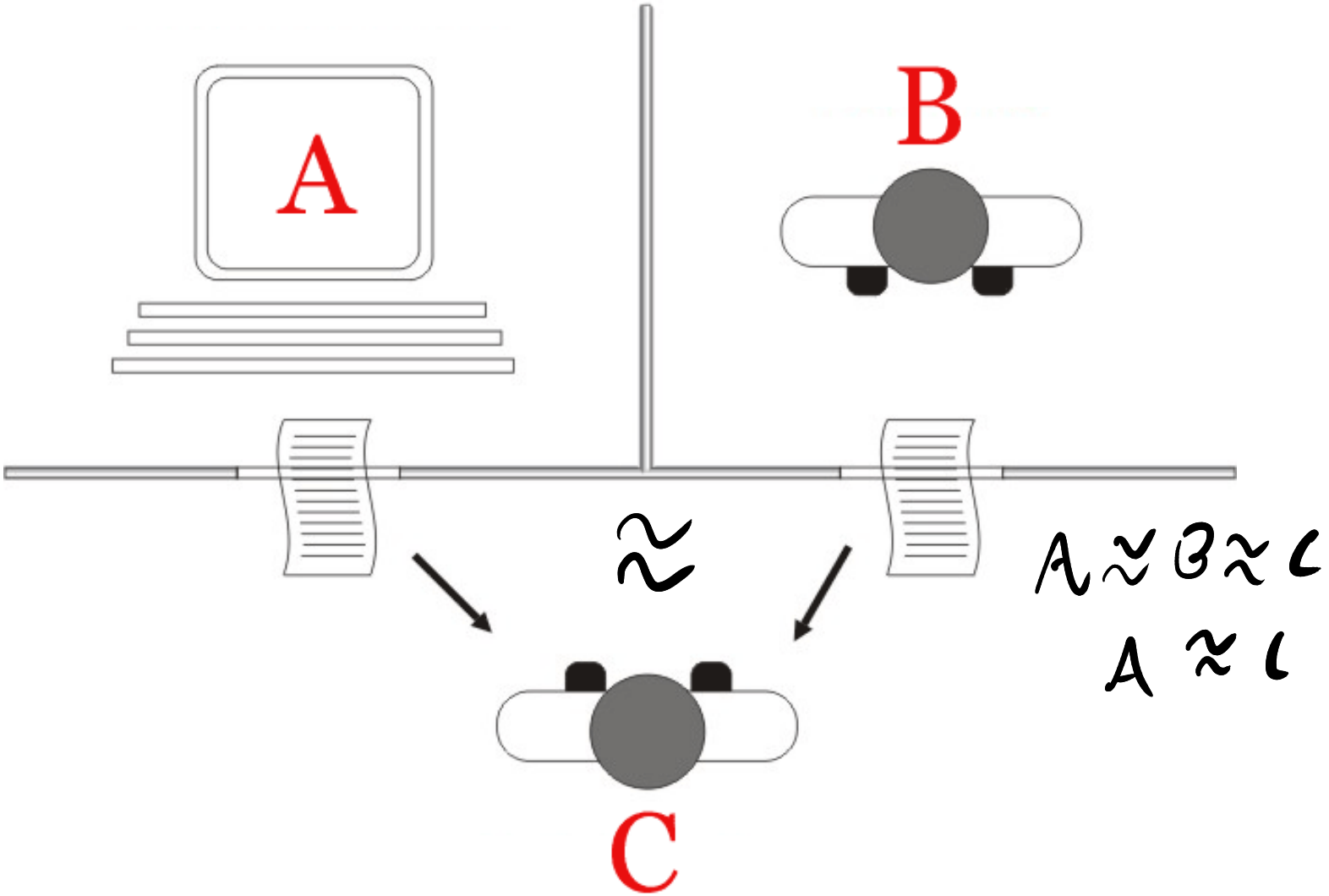
$m_0 = 0^1$

$m_1 = 0^{100}$

- Does it capture a real world model of security?
- Almost a philosophical question, much debate
- Inability to decrypt is not evidence of security
- Many alternative definitions. IND-CCA, IND-CCA3
- IND-CCA2 $\Longrightarrow$ IND-CCA $\Longrightarrow$ IND-CPA
- Some definitions can even be contradictory!

Computational Indistinguishability

- For two probability ensembles $\{X(a,n)\}_{n>0}, \{Y(a,n)\}_{n>0}$ indexed by strings a and integers n, we say that X is computationally indistinguishable from Y $(X \approx Y) \iff$ For all Distinguishers D:
  $|Pr[D(X(a,n)) = 1] - Pr[D(Y(a,n)) = 1]| < \mu(n)$
- $\mu(n)$ is a negligible function if $\mu(n) < 1/poly(n)$
- D must also be polytime? ~~and non-uniform~~


A
B
C
≈
A ≈ B ≈ C
A ≈ C

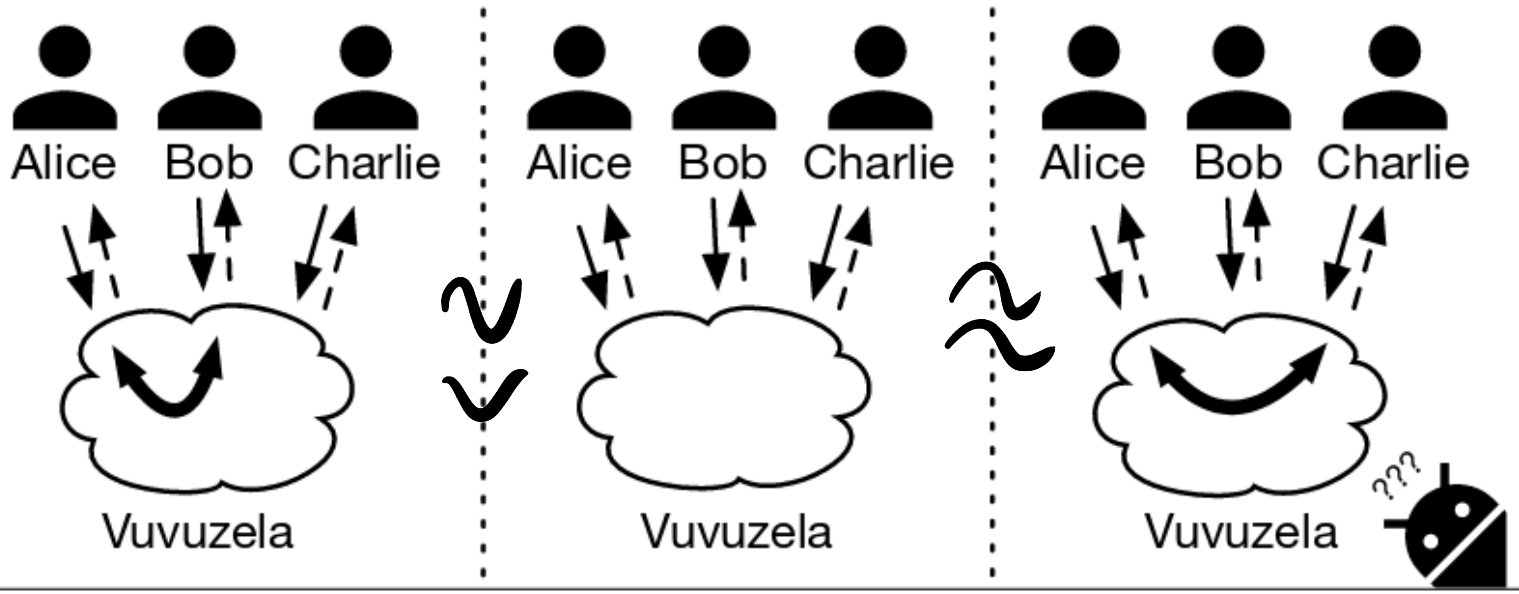
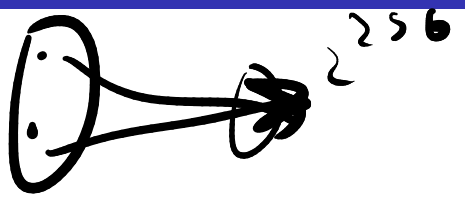


Figure 2: Vuvuzela's security goal. An adversary must not be able to distinguish between various possible worlds. In one world, Alice is communicating through Vuvuzela with Bob. In another, she is connected but not exchanging messages with other users. In a third, she is communicating with Charlie. Vuvuzela gives Alice differential privacy: any event observed by the adversary has roughly equal probability in all worlds.

Hash Functions

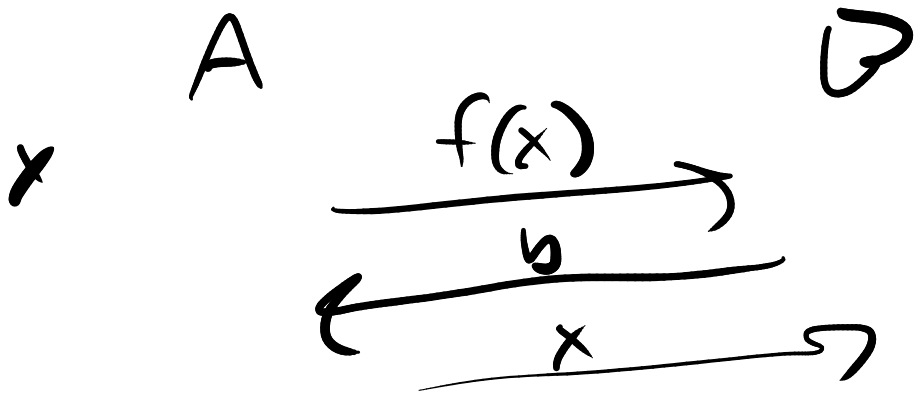
- Hash functions are one-way functions that always output a string of fixed length
- $H : \{0,1\}^* \rightarrow \{0,1\}^\lambda$
- Useful for file verification
- Collect a hash of your file, upload it somewhere, hash it there
- If hashes match, files match.

Hash Functions



- Should be unique
- a collision is a pair x, y such that $H(x) = H(y)$ and $x \neq y$.
- Given H is one-to-one, and the domain $>$ co-domain, collisions are guaranteed to exist. (Why?)
- 2^{80} considered good enough.
- Last cryptanalysis of MD5 was 2^{18}.
- Good hash functions have butterfly effect
- Changing one bit should propogate through whole calculation, whole hash changes

Coin flip over a telephone



- Alice chooses $x \xleftarrow{\$} \mathbb{Z}_n$
- sends $y = f(x)$ to Bob
- He guesses if x is even or odd, sends his guess to Alice
- Alice says yes/no, sends him x.
- Bob verifies $f(x) = y$
- If Bob's guess was correct, its heads
- else tails

One way functions

- Easy to compute $f(x)$ given x
- very difficult to compute x given $f(x)$.
- Don't even know if they actually exist
- Existence of a one-way function $\implies P \neq NP$
- $f : \{0,1\}^* \to \{0,1\}^*$ computed in polytime
- $\forall F$ randomized algorithms: ✗ uniform
- $Pr[f(F(f(x))) = f(x)] < \mu(x)$
- μ is a negligible function if $|\mu(x)| < 1/poly(x)$ for all polynomials

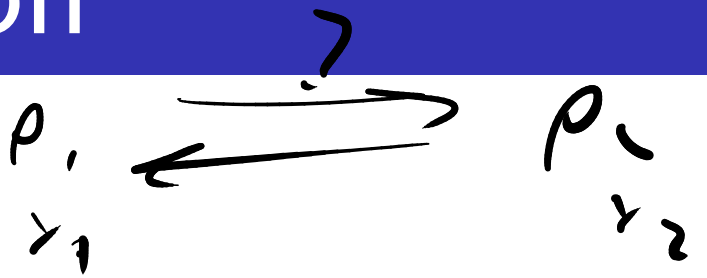
Collision resistance

- We say H is collision resistant if finding a collision pair is computationally infeasible
- Formally, We give an adversary A some key $k \xleftarrow{\$} \{0,1\}^k$
- A outputs ticket (x, y)
- $Adv_H^{cr}(A) = Pr[H_k(x) = H_k(y)]$
- Modern hash functions have the collision resistance proven to some other hard problem.
- If you can show some hash function is not collision resistant, then you can factor integers or perform discrete log.

Zoo

- https://www.cs.ucdavis.edu/ rogaway/papers/relates.pdf

Multiparty computation

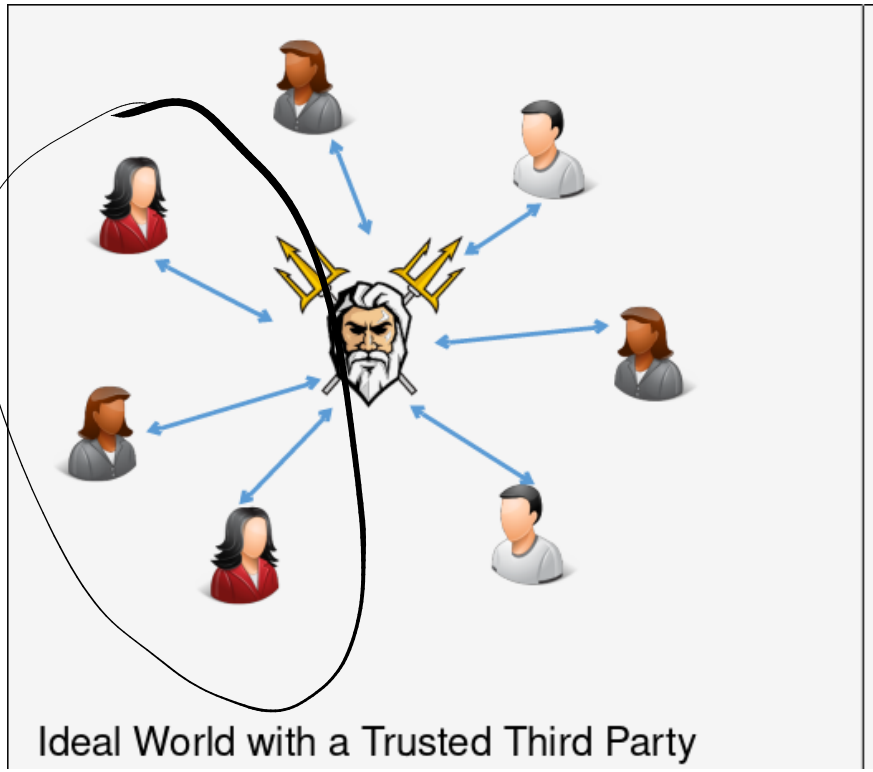


- Suppose we have n players, Each p_i has input x_i.
- Want to compute $F(x_1, ..., x_n)$ such that each player learns the output, and nothing else.
- Suppose $F(x_1, x_2) = x_1 + x_2$, then p_i, knowing $x_1, and (x_1 + x_2)$ can always learn x_2.
- But this is fine.

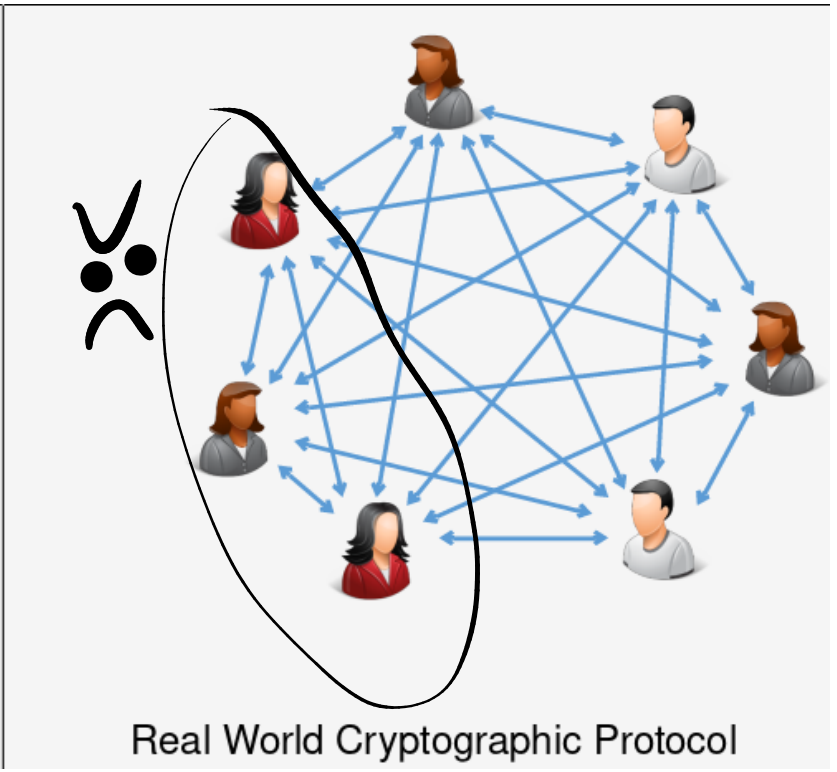
A first definition

- We desire certain properties, like input privacy, correctness, independence of inputs, guaranteed output delivery, fairness.
- Suppose F is supposed to be an auction, p_1 encrypts their bid as $g^{x_1} \pmod{m}$.
- p_1 can bid one higher by $gE(x_1) = g(g^{x_1}) = g^{x_1+1} = E(x_1 + 1)$

$a,\ a+b \rightarrow b$



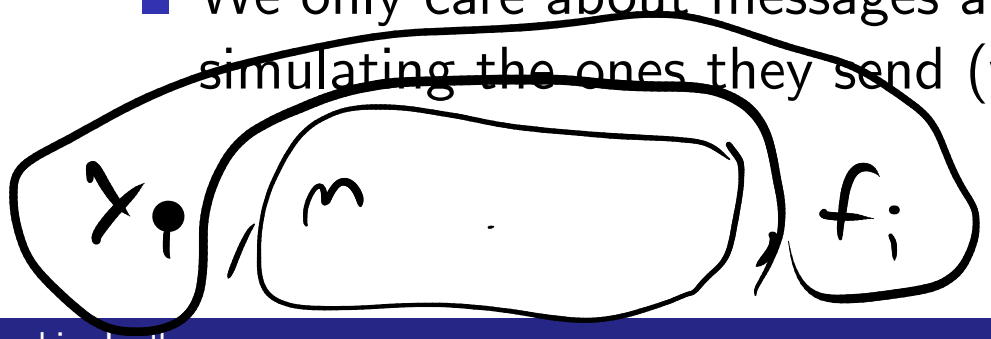
Ideal World with a Trusted Third Party



Real World Cryptographic Protocol

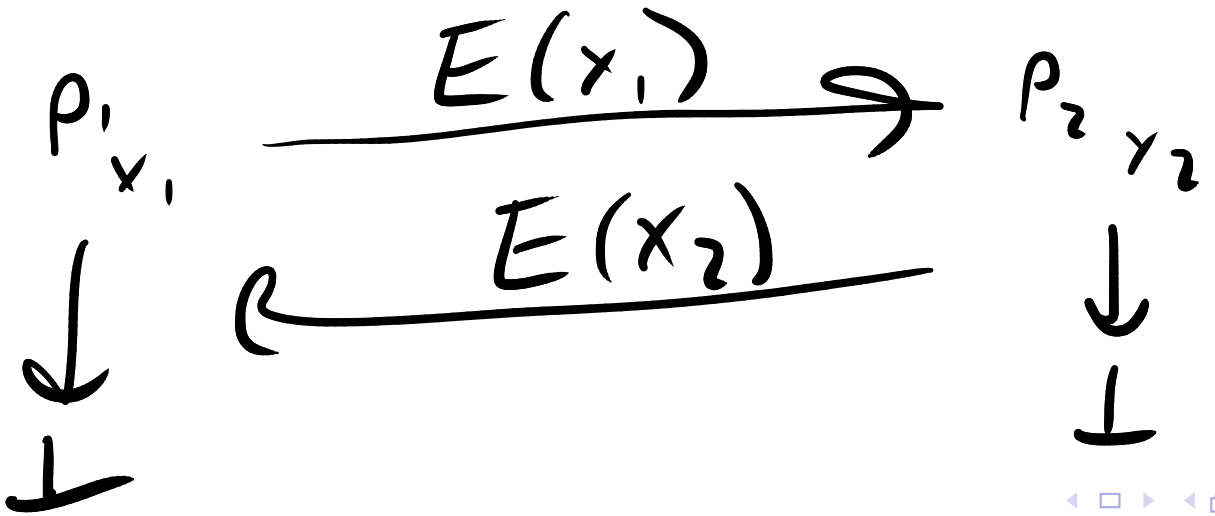
Real-Ideal Paradigm

- We define a protocol to be **secure** if for any subset of adversaries attacking the real world, they can achieve the same by attacking the ideal world.
- Prove it by writing simulators.
- A player sends and receives messages. $\{x_i, m_i^1, ...m_i^n, f_i\}$
- A simulator is a PPTM $Sim(x_i, f_i)$ and outputs a string indistinguishable from a players view.
- We only care about messages a player receives, and not about simulating the ones they send (why?)



Small example

- Dummy protocol. p_1 encrypts x_1, sends it to p_2. Then p_2 does the same. Then they both output $\perp$
- For p_1, view is $VIEW_1^{\pi} = \{x_1, E(x_2), \perp\}$.
- $Sim_1(x_1, \perp)$ must output an indistinguishable string.



Small example

- Dummy protocol. p_1 encrypts x_1, sends it to p_2. Then p_2 does the same. Then they both output $\perp$
- For p_1, view is $VIEW_1^{\pi} = \{x_1, E(x_2), \perp\}$.
- $Sim_1(x_1, \perp)$ must output an indistinguishable string.
- Consider $\{x_1, E(0), \perp\}$. Since $E(0) \approx E(x_2)$, then
- $\{x_1, E(0), \perp\} \approx \{x_1, E(x_2), \perp\}$

$x_1, E(0), \perp$

62 60

P

A

m_0, m_1 ←

Q?

$b \xleftarrow{\$} \mathbb{Z}_2$

$E(M_b)$ →

b'

$P[b = b'] = \frac{1}{2}$

2^{-K}

$\mu(n)\, n^k$ still negl

$m_0 \| m_0 \,,\; m_0 \| m_1$ ←

$E_K(m_b)$ →

$E_K(m)$ $(b_1, b_2, a_3, b_3, .)$

$[K_1, K_2, K_3]$

$D_{K_3}(a_3, b_3) \to a_2$

$D_{K_2}(\quad, b_2) \to a_1$

$D_{K_1}(\quad, b_1) = m$

$m_2 \| m_3$

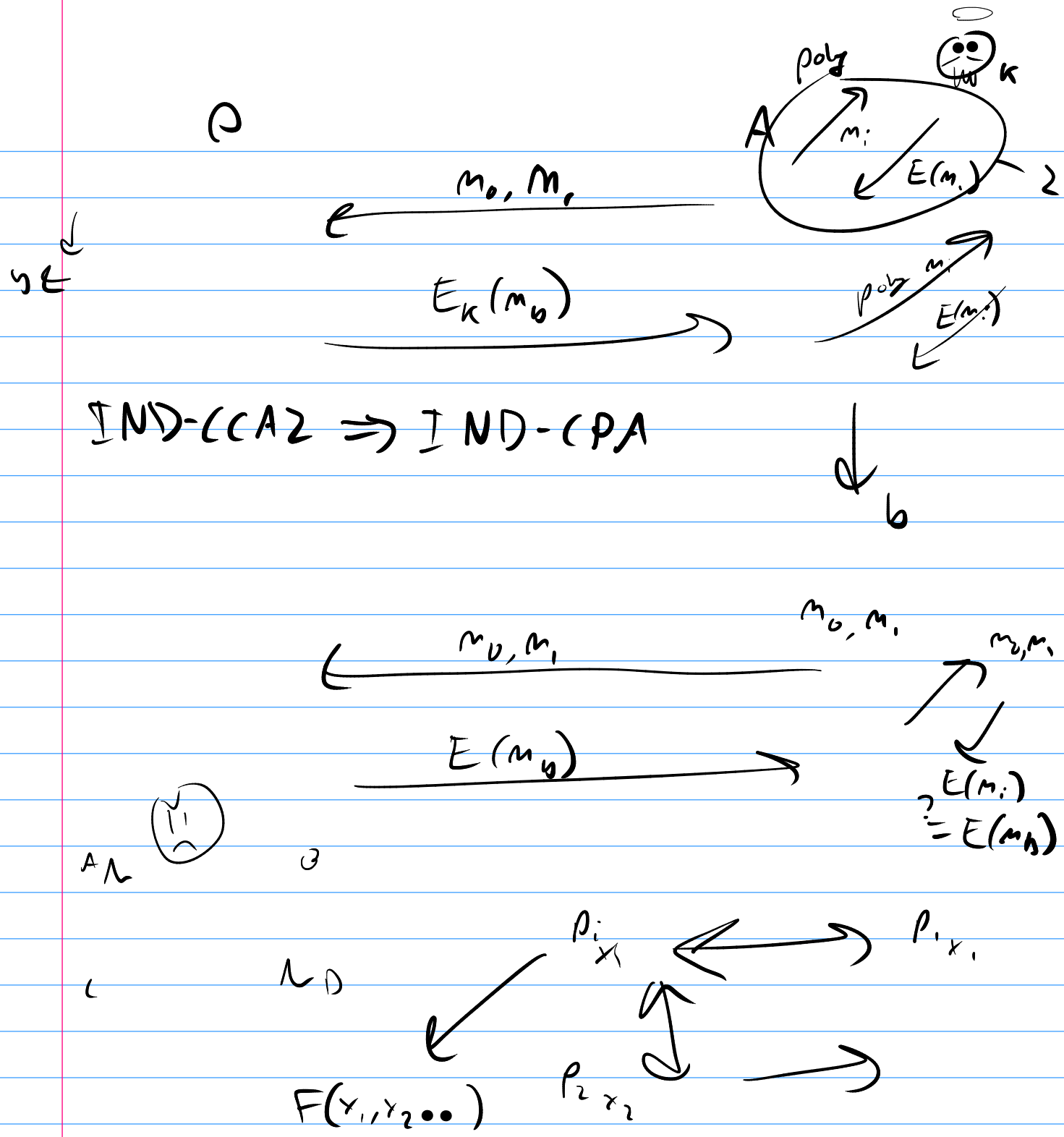
if $b = 0$ $m_2 = m_3$

$b = 1$ $m_2 \neq m_3$

$|K| = 3|K_1|$

32^{K_1}

$2^{3|K_1|}$


poly
K
A
m_i
$E(m_i)$
2
m_0, m_1
$E_K(m_b)$
poly
m_i
$E(m_i)$
IND-CCA2 ⇒ IND-CPA
b
m_0, m_1
m_0, m_1
m_2, m_3
$E(m_b)$
$E(m_i)$
$= E(m_3)$
A
$P_{i\,x_1}$
$P_{1\,x_1}$
$P_{2\,x_2}$
$F(x_1, x_2 \ldots)$