Fundamentals of Cryptography

LEC01

1. Talking about security:

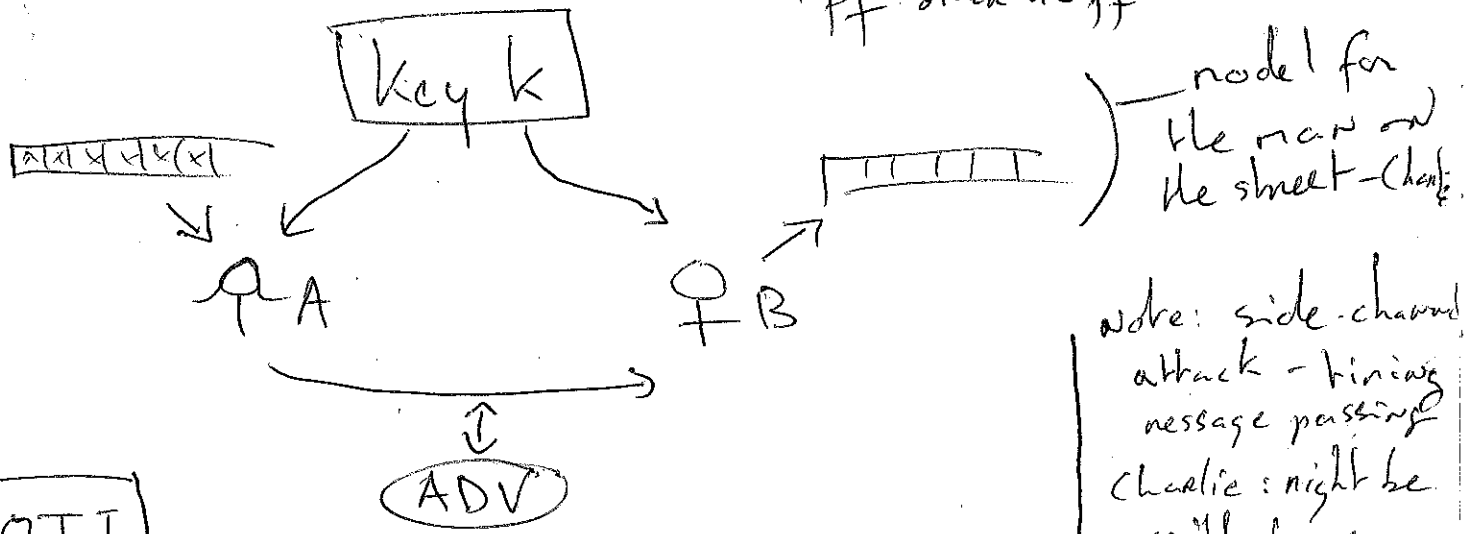
ECE → programmers are bad

CS ← right approach

MAT → like NUMBER theory (explore) small in depth)

Adversary: • privacy: ADV can't learn anything about messages
• integrity: ADV can't make B output smthg wrong

Theorem: Secure sessions are possible iff pseudo-gens exist
iff one-way f° exist
iff other stuff



note: side-channel attack - timing message passing
Charlie: might be possible to use analogue signal

TMOTI

"the magic of the internet" → works without an adversary

2. Perfect Privacy

ADV with unlimited computational power

cryptology
- Cryptography → make
- cryptoanalysis → break

One-time pad

implicit: ADV does not know key but knows cryptosys

$k = k_1 \dots k_n$

$m = m_1 \dots m_n$

A → B

$e_1 = m_1 \oplus k_1$, $e_2 = \dots$

B decrypts

$m_1 \leftarrow e_1 \oplus k_1$

$m_2 \leftarrow e_2 \oplus k_2$

We have $\overbrace{ENC(M,K)}^{l_3 \text{ bits}}$ ($\uparrow l_1$ $\uparrow l_2$) → $\overbrace{DEC(E,K)}^{l_1 \text{ bits}}$ ($\uparrow l_3$ $\uparrow l_2$)

Def 1 For $M \in \{0,1\}^{l_1}$, define distribution D_M on l_3-bit strings output $ENC(M,k)$ for a randomly chosen K — then, for all $M, M' \in \{0,1\}^{l_1}$, $D_M = D_{M'}$

Notation: $M \xleftarrow{\$} \{0,1\}^n$ means randomly

in math defs are artificial, in CS, we follow our intuition to def

Charlie it is hard to define notions in crypto — e.g. for every $f: \{0,1\}^{l_3} \to \{0,1\}^{l_1}$, f^n does badly^n

Def 3 Given $M \in \{0,1\}^{l_1}$ chosen randomly and some for k in $\{0,1\}^{l_2}$. the probability of $f(ENC(M,K)) = M$ is 2^{-l_1}

Def 2 ("for every 2 messages, can't tell which one was encrypted")

$\forall f: \{0,1\}^{l_3} \to \{0,1\}^{l_1}$, $\forall M_0, M_1 \in \{0,1\}^{l_1}$

$b \xleftarrow{\$} \{0,1\}$, $k \xleftarrow{\$} \{0,1\}^{l_2}$: $P(f(ENC(M_b,k)) = b) = \frac{1}{2}$

Theorem 1

Defs 1-2-3 are ⇔

Theorem 2

One-time pad satisfies perfect privacy

Theorem 3

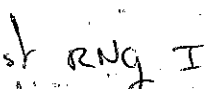
If (ENC, DEC) satisfies perfect privacy (& DEC properly decrypts)
then $l_2 \geq l_1$

Charlie: "it's funny how in movies they show decryption as bits & pieces slowly appearing and progressively leading to the message"

3. Imperfect Privacy

Idea: Use a pseudo random number generator:
$G: \{0,1\}^n \to \{0,1\}^{n^3}$
and let both parties use $G(k)$ as a one-time pad.

How to parse pseudo random number generator
→ generator of strings (is archaic)

(Charlie: best RNG I can think of is  → ⊗
but to get an exact one, we need quantum mechanics

Def. A # gen is a polytime comp func°:
$G: \{0,1\}^* \to \{0,1\}^*$, s.t. $|G(s)| = l(|s|)$
for some f° $l: \{0,1\}^*$ one-one

Vague Def: A # gen G is pseudo random if polytime adversary cannot do "significantly" well in distinguishing PRN from RAN key — the world!

(Charlie: just stating this def was a huge contribution from TCS to
→ see notes [missed lecture 2].

LEC03

Def: G is pseudo random against multiple sampling iff $\forall D$,
(where D is a probabilistic polytime algorithm s.t. input is $\alpha \in \{0,1\}^{n-l(n)}$ for some n and D's output is accept/reject)
$\forall c$ and sufficiently large n: $|P_D(n) - R_D(n)| \leq \frac{1}{n^c}$

where: $\cdot P_D(n)$ = prob that if $(s_i)_{i \le n}$ are $\overset{iid}{\text{R.V.}}$ from $\{0,1\}^n$ and D is run on $|G(s_1)G(s_2)\ldots G(s_n)|$ then D accepts.

$\cdot R_D(n)$ = prob that if α is R.V. from $\{0,1\}^{n-\ell(n)}$ and D is run on α, then D accepts.

Thm G is pseudo random iff G is p.r. against multiple sampling

thought experiment: "Do this m times $0 \le i \le m$:
- choose first i p.rand. and last $m-i$ rand.
- choose $(s_i)_{i \le n}$ from $\{0,1\}^n$
- ——— $r_{i+1}, \ldots r_m$ from $\{0,1\}^{\ell(n)}$ } randomly

and run D on $G(s_1)G(s_2)\ldots G(s_n)r_{i+1}\ldots r_m$"

Charlie: & ↦ 0 0

- Let q_i = prob D accepts
- Let $q_0 = R_D(n)$ & $q_m = P_D(n)$ & $P_D(n) - R_D(n) > \varepsilon_n$, $q_m - q_0 > \varepsilon(n)$

$= (q_m - q_{m-1}) + (q_{m-1} - q_{m-2}) \ldots$

- Let i be s.t. $q_{i+1} - q_i > \frac{\varepsilon(n)}{n}$
- Define adv. D' for G (w/ single sampling) as follows: choose $(s_i)_{1 \le i}$ from $\{0,1\}^n$, $r_{i+2}\ldots r_m \in \{0,1\}^{\ell(n)}$
  → Run D on $G(s_1)\ldots \alpha\, r_{i+2}\ldots r_m$ & accept iff D accepts.

Thus $P_{D'}(n) = q_{i+1}$; $R_{D'}(n) = q_i$

$\Rightarrow P_{D'}(n) - R_{D'}(n) > \frac{\varepsilon(n)}{n} = \frac{1}{n^{c+1}}$

Nonuniform setting: i is built into circuit

[Charlie: law of large nbrs you learned in stats is nice & elegant but useless. Doesn't tell you how quickly it converges.]

↳ Chernoff Bounds: if we try to estimate q by doing this experiment $(\frac{1}{\varepsilon^2})m$ times, then w/ prob $\ge 1 - \frac{1}{2^m}$, our estimate will be within ε of q.

R! how much polling is sufficient to get good bounds!

Uniform setting: choose i rand. from $\{0..m-1\}$

Def: G is unpredictable (iff "you are going to look at the first N bits and cannot predict the next one well")

Thm G is p. rand iff G is unpredictable —

(1) $\Rightarrow$ (easy): look at how many bits you predict correctly the next bit of K (supposedly p. rand) vs how many bits it would take you iff K was truly rand

Def: adversary $\{(A_i, c_i), 6..\}$ poly size family, A_m has $i_m - 1$ inputs and outputs $1 \le i_m \le l(m)$

Def: $\text{pred}_A(n)$ defined as follows:

- given $S \xleftarrow{\$} \{0,1\}^n$, A_m gives first $i_m - 1$ bits of $G(s)$
- $\Rightarrow \text{pred}_A(n) \leftarrow$ prob A_m outputs i_m^{th} bit of $G(s)$ —

Def: G is unpredictable iff $\forall$ adversary, $\text{pred}_A(n) \le \frac{1}{2} + \frac{1}{n^c}$ for every c & sufficiently large n —

Charlie: this is also called unpredictability from the left. Is it obvious that it is $\Leftrightarrow$ to unpred. from right? It turns out that yes, but I find it extremely ~~unpredictable~~ not intuitive —

(2) $\Leftarrow$ Say $D = [D_n]$ breaks the p. randomness of G. Say $P_D(n) - R_D(n) > \frac{1}{n^c}$

For $0 \le i \le l(n)$, define p_i:

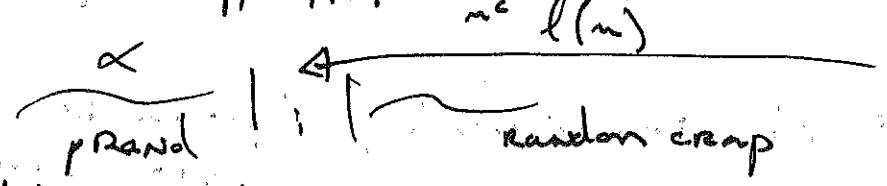
- experiment:
  - $S \xleftarrow{\$} \{0,1\}^n$, say $G(s) = b_1 b_2 \dots b_{l(n)}$
  - choose rand. bits $a_{i+1} \dots a_{l(n)}$
  - Run D on $b_1 \dots b_i a_{i+1} \dots a_{l(n)}$
  - $+p_i \leftarrow$ prob. D accepts —

$\Rightarrow P_0 = R_D(n)$; $P_{l(n)} = P_D(n)$

App to know how much stuff is left in your system —

I can't remember if I took my meds last night. feeling ~~violent~~

Let i be s.t. $p_i - p_{i-1} > \frac{1}{n^c} \frac{1}{\ell(n)}$



So let $I_n = i$. Let A_n be as follows: A_n gives $i-1$ bit string α → Run D on $\alpha b b_1 \ldots b_{\ell(n)}$ for $b_1 \ldots b_{\ell(n)}$. If D accepts, output b, else output $\bar{b} = 1-b$

Then $pred_A(n) = prob(b = b_i \;\&\; D_n \text{ accepts}) + prob(b = \bar{b_i} \;\&\; D_n \text{ rejects})$

$= \frac{1}{2} prob(D_n \text{ accepts} \mid b = b_i) + \frac{1}{2} prob(D_n \text{ rejects} \mid b = \bar{b_i})$ (p_i)

$= \frac{1}{2} + (p_i - p_{i-1}) \geq \frac{1}{2} + \frac{\varepsilon(n)}{\ell(n)}$ — $(1-p_{i-1}) - \frac{1}{2}(1-p_i)$

Charlie: if you pick a side and I flip a coin, even if you choose the same side, the prob is still 1/2

$prob(D_n \text{ rejects}) = 1 - p_{i-1}$: justify ←

$= \frac{1}{2} prob(D_n \text{ rejects} \mid b = \bar{b_i}) + \frac{1}{2} prob(D_n \text{ rejects} \mid b = b_i)$

$1 - p_i$ — prob D_n accepts!

Corollary G is unpredict from left $\iff$ G is unpredict from right.

Charlie: "We are building a huge beautiful edifice out of stuff that might not even exist at all. I find that disturbing... If P=NP, then none of this crap holds."

Thm P=NP ⇒ ∄ p.r. number generator

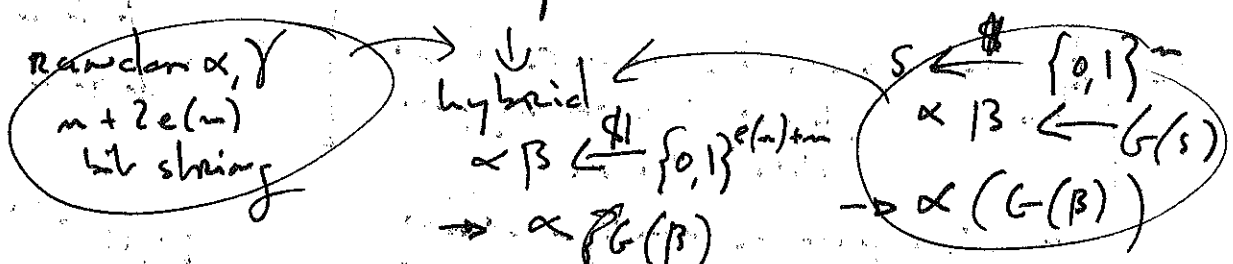
Thm Let $G'(s) = G(s)\,G(s)$, G p.r.n.g. Then $G'(s)$ is not p.r. (doesn't matter if p.r.ng. ∃)

Thm Let $G'(bs) = b\,G(s)$, G p.r.n.g. Then if ∃ p.r.n.g. then:

- ∃ G', $G'(s)$ is p.r.ng
- ∃ G, $G'(s)$ is not p.r.n.g.

NIST std: All applications of G, p.r.n.g. must have same seed length
$\Rightarrow$ $G'(s) = G(G(s))$ is impossible (given $|G(s)| > |s|$)

example



Let G be a n.g., $l(n) = n+1$. Let $f(n) = n^d$ (prob. wlog)
Say $l(n) = e(n) + n$, $e(n) \geq 1$

extra bit

For every $i \in \mathbb{N}$, $s \in \{0,1\}^n$, let $\begin{cases} G_0(s) = s & \text{(empty string)} \\ G_{i+1}(s) = \alpha G_i(\beta) \end{cases}$

NEVER say something about cause you TA when students are around!

(where $G(s) = \alpha\beta$, $|\alpha| = e(n)$, $|\beta| = n$, $l(n) = l(n) - e(n)$)

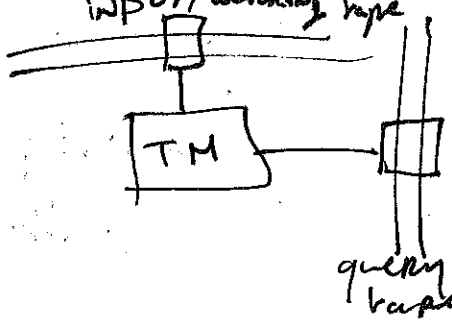
Let $G'(s) = G_{l(n)}(s)$

Thm G p.r.n.g. $\Rightarrow$ G' p.r.n.g.

$G : \{0,1\}^n \longrightarrow \{0,1\}^{2^n \times n}$ — how is it possible? in polytime — $\Rightarrow$ can compute any bit in polytime

$\rightarrow$ we want an expo output
$\rightarrow$ can generate an exponential strings?
$\times$ $\rightarrow$ generate a function generating strings

Def A function generator F associates w/ every $s \in \{0,1\}^n$, a func $F_s : \{0,1\}^n \rightarrow \{0,1\}^n$.

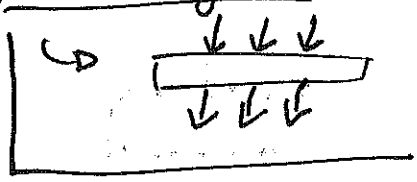


Def ~~F is PRF~~ uniform adversary: a prob. polytime TM with a special query tape (w/ function)

Charlie: "invent a new syntax!!!" $\rightarrow$ ok

Def: Oracle tape: query something not present.

Def: Non uniform adversary:
D_m is a circuit w/ 0,1 input & function gates



Let $R_{D_m}(n)$ = prob D_m accepts when f is chosen randomly from $\{0,1\}^n \to \{0,1\}^n$

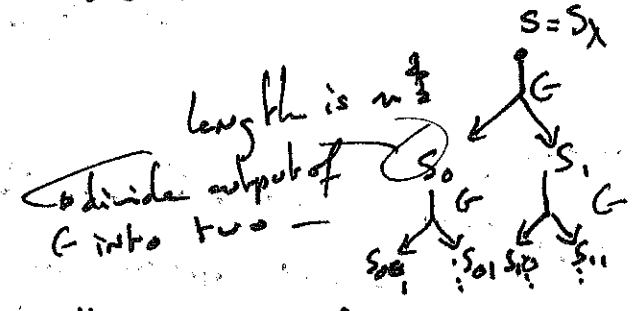
$P_{D_m}(n)$ = prob D_m accepts when run on exp in $\{0,1\}^n$ & $f \in (\{0,1\}^n)^{\{0,1\}^n}$

Def: $\forall D$ poly size/poly time, $\forall c$, for suff. large n, $|P_D(n) - R_D(n)| \leq \frac{1}{n^c}$

F is p.R. iff

Thm $\exists$ p.R.N.g. iff $\exists$ p.R.f.g.

proof. ($\Leftarrow$) define $G(s) = F_s(\bar{0})\,F_s(\bar{1})$ — strings corresponding to index 0
then show if can break $G(s) \Rightarrow$ can break $F_s(s)$ —
($\Rightarrow$) Let G, a p.R.N.g., $l(n) = 2n$ —



$s_\lambda \leftarrow s$
$s_{\sigma 0} =$ left half of $G(s_\sigma)$
$s_{\sigma 1} =$ right

Why is this secure?
→ say we can break F. The i^{th} hybrid: level i chosen randomly & tree constructed downwards —

NIST (National Institute of Standards & Technology)
└ AES (Advanced Encryption System)

$f_{gen} \text{in} AES_k : \{0,1\}^{128} \to \{0,1\}^{128}$
128 bit key

Note: 2 more versions:
$AES_k^{192} : \{0,1\}^{128} \to \{0,1\}^{128}$ (192 bits)
$AES_k^{256} : \{0,1\}^{128} \to \{0,1\}^{128}$ (256 bits)

- Given a p.r.f.g. F, we want F' s.t.

$$F'_k : \{0,1\}^n \longrightarrow \{0,1\}^{2n}$$

→ i.e. we want $AES'_k : \{0,1\}^{128} \longrightarrow \{0,1\}^{256}$

Robert: try $F'_k(x) = F_k(x) F_k(F_k(x))$

Sean: if that was secure, then I am about $F'_k(x) = F_k(x)F_k(x)$

Charlie: proof that both are insecure:

→ Adv. given access to $f : \{0,1\}^n \to \{0,1\}^{2n}$
→ choose x.
→ $\alpha\beta \leftarrow f(x)$, $\gamma\delta \leftarrow f(\alpha)$ (w/ $|\alpha| = |\beta|$)
→ accept iff $\delta = \beta$.

try $F'_k(\alpha\beta) = F_k(\alpha\beta)F_k(\beta\alpha)$

Charlie: ... what if $\alpha = \beta$? ... I give up;

$$F'_{k_1,k_2}(x) = F'_{k_1}(x)\, F'_{k_2}(x)$$

Robert: but we want a n bit key - how do you get $2n$?

Charlie: ok. $F''_k(x) = F'_{G(x)}(x)$

where G is a p.r.n.g obtained by:

$$G(x) = F_x(\bar{0})\, F_x(\bar{1})$$

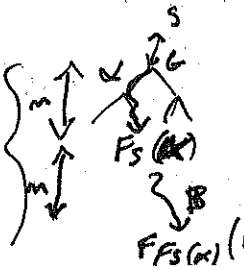
Now: $F''_k(x) = F_{F_k(\bar{0})}(x)\, F_{F_k(\bar{1})}(x)$

- Now we want: $F'_k : \{\{0,1\}^{2n} \to \{0,1\}^n$

Charlie: you don't get points for creativity, it just has to work.

- His works:

$$F'_k(\alpha\beta) = F_{F_k(\alpha)}(\beta)$$



Charlie: in Goldreich's book, he don't even cover encryption.

in [GGM] terms ↪ (MD construction)

Another (less obvious to prove) method using cipher-block chaining

$F'_k(x\beta) = F_k(F_k(x) \oplus \beta)$ — hard part of proof is to show this looks rand. when f is rand.

+ We never want $F_k^* : \{0,1\}^* \to \{0,1\}^n$ ☹

→ How do we get $F'_k : (\{0,1\}^m)^* \to \{0,1\}^n$ from p.r. F_k : $\{0,1\}^n \to \{0,1\}^n$

+ attempt 1 $F'_k(\lambda) = k$ (empty string), $F'_k(x\beta) = F_{F'_k(x)}(\beta)$ (x: m bits)

Thm 1 F'_k is not p.r.

Thm 2 F'_k is p.r. w.r.t. adversaries restricted for every input f, to make all queries to f of same length.

+ attempt 2 For $x \in (\{0,1\}^n)^m$, let $l_x = \overline{m}$ (the n-bit string for m)

Then let: $F''_k(x) = F'_{F_k(l_x)}(x) = F'(l_x\, x)$

note: $F'(x\, l_x)$ is not p.r.

MAC: Message Authentication Code

Easy fact: f'gen with unbounded inputs that is p.r. can be used as a MAC. A MAC can be used as a shared private key signature scheme — $MAC_k : \{0,1\}^* \to \{0,1\}^n$

Security property def: A poly time Adv querying MAC_k on some inputs, "cannot" compute $MAC_k(x)$ for some new x —

A construction that doesn't work: $F'_k(x) = F_k(x \oplus k)$ where F_k is p.r. { the p.r. of F^* does not carry over 'security' to F'

Thm: IF p.r. generators exist, then there is a p.r. f.g. F s.t. F' is not secure —

proof: We construct a p.r. F s.t. for all k, $F_k(k) = \bar{0}$
So $F_k'(0) = \bar{0}$ for all k —
Say we start with p.r.f.g. H.
Define $F_k(x) = \begin{cases} H_k & \text{if } x \neq k \\ \bar{0} & \text{if } x = k \end{cases}$

Then F is p.r.

~~attempt~~ 2 Using "appropriate" hash family H.

$F'_{k_1,k_2}(x) = f_{k_2}(H_{k_1}(x))$ → $H_{k_1} : \{0,1\}^* \to \{0,1\}^{\ell}$
"hash down" or "hash up"

→ what properties for H?

→ property for hash f:

- "privately collision resistant": "an adv. cannot find (without knowing k) $x_1, x_2 \in \{0,1\}^*$, $x_1 \neq x_2$ s.t. $H_{k_1}(x_1) = H_{k_1}(x_2)$"
- strong unif. def: "$\forall c, d, \forall^{\text{prob}}_{k} k \in \{0,1\}^m, \forall x_1, x_2 \in \{0,1\}^{\leq m^c}, x_1 \neq x_2$, $H_k(x_1) = H_k(x_2) \leq \frac{1}{m^d}$ for suff. large m —"

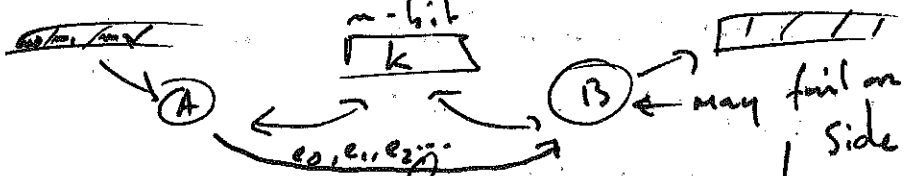
→ Thm: H w/ such properties exist. e.g. $H_k(x) = x \bmod k$
→ $H_k(x_1) = H_k(x_2)$ iff k divides $x_1 - x_2$ —

→ cannot find a # it's likely to divide except if it is very large (e.g. product of all n-bit integers) —

→ actually doesn't work: $0 \& 00$ (strings differ from integers be of leading 0s.) use this

$H_k(x) = 1x \bmod k$ —

Back to secure sessions.
(notice that the hard part won't be to prove things are secure but to define it)



Adv. (has complete control over channel)

B may fail or abort (≡ sus. integrity failure Adv.)

Side-channel info: e.g. how much time goes into 'computing packages', delays, etc.

Goals: { Integrity, Privacy } ← relies on'

→ e.g. $e_i = m_i \oplus F_k(\bar{i})$ and decode the reverse way
→ integrity? no
→ privacy? yes

→ e.g. $e_i = m_i F_k(m_i)$
→ integ? no → priv? no

→ e.g. $e_i = m_i F_k(i m_i)$ — just MAC m_i w/ its position i to avoid Adv.s the possibility of noticing repeated messages
→ integ? yes → priv? no

→ e.g. $e_i = \underbrace{m_i \oplus F_{k_1}(\bar{i})}_{\alpha} \underbrace{F_{k_2}(\bar{i}\alpha)}$ — MAC it with position for integrity
gives privacy → integ? yes priv? yes

"encrypt for privacy, then MAC it for integrity"
⊕

→ And decode as follows: given $e_i = \alpha\beta$
if $\beta \neq F_{k_2}(\bar{i}\alpha)$ then abort.
else output $m_i' = \alpha \oplus F_{k_1}(\bar{i})$

Def: Session protocol

+ $z(m) = m_i$
$z'(m) = e_i$
$z''(m)$ = history of A & B (fixed length so computn don't become longer (longer) and
+ In the absence of Adv. everything works (A wants to send m_i ⇒ B outputs m_i)
[correctness property]

Def: Strong Adv.

. works in poly time, is probabilistic (now uniform)

strong adv. must be online: chooses m_0, sees e_0 ... chooses m_{n^d}, sees e_{n^d}

. interacts with A, B and randomly chosen fixed k

→ $q(n)$ = prob that for some $i \le n^d$, $m_i \neq m_i'$ & $m_i' \neq$ FAIL

B decodes: sees e_i and outputs m_i'

Def: Integrity: $\forall$ Adv, suff. large n, $\forall c$: $q_{Adv}(n) \le \frac{1}{n^c}$

Def: Privacy: what does Adv. know? can't know all the message

→ choose the entire message except 1 bit. "choose"

(Charlie: "in practice, we can observe that B FAILS, info can be used")

→ Adv. must be able to see when & if B FAILS

ADV redux

claim w/ integr. this means privacy

- . poly time + chooses i.
- . chooses m_i, sees e_i except for 1 bit: $b \in \{0,1\}$ chosen randomly
- . chooses $\alpha = e_0 e_1 e_2 \cdots$
- . learns if/when B fails while receiving α

Def: Security = Privacy + Integrity

concatenation (aka ||)

e.g. secure system: $e_i = \underbrace{m_i \oplus F_{k_1}(i)}_{r}, F_{k_2}(i, \underbrace{m_i \oplus F_k(i)}_{r})$

Def: Let $F_k : \{0,1\}^m \to \{0,1\}^m$, a f^m gen.

F_k is a permutation gen. iff $\begin{cases} \forall k, F_k \text{ is 1-1 / onto} \\ \exists \text{ poly time alg that given } k \text{ \&} \\ y \in \{0,1\}^m, \text{ computes } F_k^{-1}(y) \end{cases}$

Def: F_k, perm. gen, is strongly p.r. if: w/ poly time ADV^D w/ oracles f, f^{-1}

then: $\forall D, \forall c$, for suff. large n, $|R_D(n) - P_D(n)| \le \frac{1}{n^c}$

where: $\begin{cases} R_D(n): \text{prob } D \text{ accepts given } f, f^{-1} \text{ for a rand perm gen} \\ P_D(n): \text{---------- } F_k, F_k^{-1} \text{ for a rand } k \in \{0,1\}^m \end{cases}$

Fact: AES is a perm. gen & it is believed to be strong p.R.

e.g. secure protocol

. Let F be str. p.r. s.t. $F_k : \{0,1\}^{2m} \to \{0,1\}^{2m}$ (m bits)

. $e_i \leftarrow F_k(i, m_i)$ (i: m bits, m_i: m bits)

. Bob e_i' : . $(j,u) \leftarrow F_k^{-1}(e_i')$, if $j \neq i$: ABORT
. else output piece u

claim: This is secure! (really need str p.r. though)
(if F str p.r.)

Thm (very hard to prove): $\exists$ p.r. gen $\Rightarrow \exists$ str. p.r. perm. gen.

e.g. bad example w/ CBC (cypher block chaining)

$e_{-1} \leftarrow F_k(\bar{0})$

$e_i \leftarrow F_k(e_{i-1} \oplus m_i)$ | Does not satisfy privacy:

proof: . choose m_0, see e_0.
. choose $m_1 = e_0$, then $e_1 = F_k(\bar{0}) = e_{-1}$ ($e_0 \oplus e_0 = \bar{0}$)
. choose $m_2 = m_0$, then $e_2 = F_k(e_1 \oplus m_0) = e_0$
. if $e_2 = e_0$, guess $b = 0$ —

Perm gen construc°

DES ~ 1985
AES ~ 2000

Start w/ crappy (but fast) function gen F_k (m bits) $\{0,1\}^m \to \{0,1\}^m$

Let $F'_{k_1 \ldots k_R} = F_{k_1} \circ \ldots \circ F_{k_R}$. Let $F''_k = F'_{ks(k)}$ ('key sheduler')

$ks(k) = k_1 k_2 \ldots k_R$

→ DES: F_k (48 bits): $\{0,1\}^{64} \to \{0,1\}^{64}$

"Feistal permuta" ($F_k(LR) = R, L \oplus g(k,R)$; L, R: 32 bits; g: 32 bits) — crappy, but randomish fixed func°

Charlie: "how many samples to crack this? to get measurable probability of selecting 2 identical elements from D, $|D| = d$, need $\sqrt{d}$ samples. So it's easy to crack DES (only 4 billion samples)"

→ AES_k: $F_k : \{0,1\}^{128} \to \{0,1\}^{128}$ (128 bits)

{ 128 bits w/ rounds
192 — —
256 — —

→ construction: given F, p.r. $F_k \{0,1\}^m \to \{0,1\}^m$

• Do 3 rounds of Feistel to compute $H_{k_1 k_2 k_3} : \{0,1\}^{2m} \to \{0,1\}^{2m}$

$$\begin{cases} L_1 = R_0 \,, & L_0 \oplus F_{k_1}(R_0) = R_1 \\ L_2 = R_1 \,, & L_1 \oplus F_{k_2}(R_1) = R_2 \\ L_3 = R_2 \,, & L_2 \oplus F_{k_3}(R_2) = R_3 \end{cases}$$

Charlie: "there's some great property that holds after 16 rounds of Feistel"

→ $H_{k_1 k_2 k_2}(L_0 R_0) = (L_3 R_3)$ is a p.r.p.g.

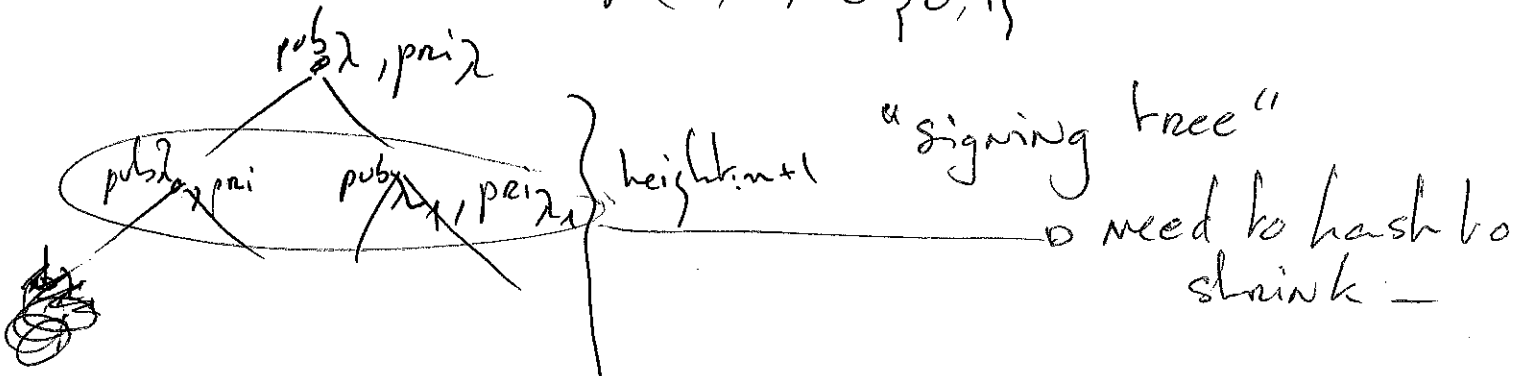
Missed lecture → see notes

Last Time

→ showed a scheme $\mathcal{S}^1$ for securely signing a single n-bit message

→ PKSS (pub key signature scheme):

- $GEN(1^n, \text{random bits}) = pub, pri$
- $SIGN_{pri}(M) = \sigma \in \{0,1\}^n$
- $VER_{pub}(\sigma, M) \in \{0,1\}$



→ need to hash to shrink

- Want stronger hash (who doesn't) than previous family of privately collision resistant hashing functions

→ without k, cannot find things that hash to same thing

Def: A hash function is public collision resistant if given k cannot find 2 things hashing to same.

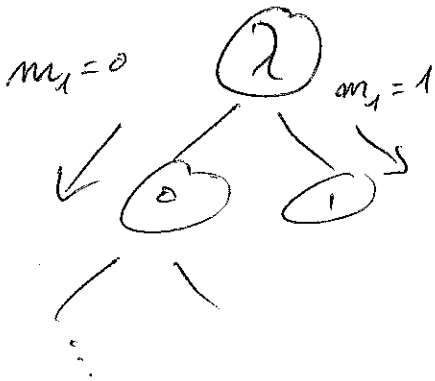
→ To sign arbitrary n-bit messages; $\mathcal{S}^2$:

- private key will contain a key k for PRF g F.

→ For each $\alpha \in \{0,1\}^{\leq n+1}$, define $pub_\alpha, pri_\alpha = GEN(1^n, F_k(\alpha))$

- private key of $\mathcal{S}^2$ also contains pri_λ

- public key contains key k' for a publicly collision resistant hash family
- public key will also contain pub_λ



→ signature of message $m = m_1 m_2 \dots m_n$ will be $\sigma_0 \sigma_1 \sigma_2 \dots \sigma_{n+1}$ where:

- $\sigma_0 = SIGN'_{PRI_\lambda}(H_{k'}(pub_0 \circ pub_1))$
- $\sigma_1 = SIGN''_{PRI_{m_1}}(H_{k'}(pub_{m_1,0} \circ pub_{m_1,1}))$

signature will also contain:
$pub_0, pub_1, pub_{m_1,0}, pub_{m_1,1}$ —

→ To sign arbitrary-length m in Message, Σ^3:

- public key: k'' for a pub. colli. resis hash f°
  - → public key for Σ^2
- private key: → private key for Σ^2

but this might collide
↓
use public colli. resis-
→ signature of $m \in \{0,1\}^*$

Charlie's previous student's theorem

Cannot get pub coll res hash family from 1-way functions! (need # theory complexity) —

Def weakly pub col res hash family:

- Adv knows n,
- chooses string $m_1 \in \{0,1\}^*$
- sees hash key $k \in \{0,1\}^{l(n)}$, chooses $m_2 \in \{0,1\}^*$ such that $m_1 \neq m_2$ and wants $H_k(m_1) = H_k(m_2)$

Thm 0: ∃ 1-way f ⟺ ∃ hash fan of this type

Thm 1: Using such a family, we can fix up previous 2 constructions

NIST presents SHA3: $\{0,1\}^* \to \{0,1\}^n$ with no key
(secure hashing algorithm)
and claims it's "collision resistant"

→ signing trees are beautiful, but I don't see them in practice ---

Charlie: this is not a mathematical assertion, they say you as a human being cannot find collision

→ In a PKI (pub key infrastruct), k is obtained through a "key agreement (or exchange protocol)", such a protocol we see some form of pub key crypto —

→ example of pub key encrypt sys: (Diffie-Hellman):
- Let p, an "appropriate" prime, such that p has n bits
- Let g, a generator of $\mathbb{Z}_p^*$
- GEN(random) = random $x \in \{0,1 \dots p-2\}$
- public key =
- to encrypt m:
  - $y \xleftarrow{\$} \{0,1 \dots p-2\}$
  - compute $g^{xy} \bmod p = (g^x \bmod p)^y \bmod p$
  - send $m \cdot (g^{xy}) \bmod p$ and $g^y \bmod p$
- to decrypt (knowing x)
  - given g^y: compute $g^{xy} \bmod p = (g^y \bmod p)^x \bmod p$ and then compute m —

Not field: multiplicative group mod p: (1,2 ... p-1)

How do we compute $a^b \bmod c$? for big a, b, c ($2^{1000\dots}$)

→ use repeated squares technique

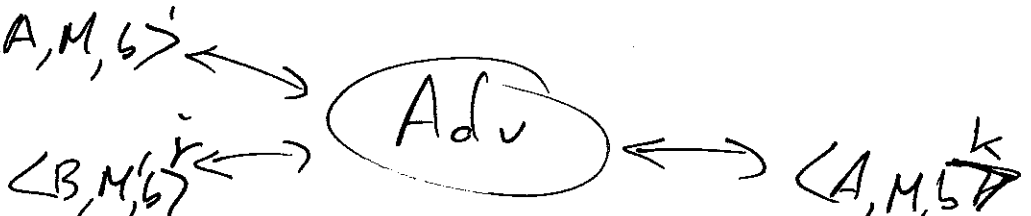
compute $\begin{cases} a^{2^0} \bmod c = u_0 \\ u_i = a^{2^i} \bmod c \end{cases}$

(missed lecture for the 3rd time) → see notes

Adv (w/ A, B in sec. session) can do (wrt. GEN for gen. long term keys):

- knows n & related params
- Choose n-bit name for a good guy & an n-bit name for a bad guy; all names must be unique
- For each good guy name A, GEN is used to choose pub_A & pri_A & Adv sees pub_A. For each bad guy name X, Adv chooses pub_X however he likes
- Adv creates $proc\langle A, M, b\rangle^i$, "$i^{th}$ proc at line 1^a from 1 to $\langle A, M, b\rangle^i$"

Chantie: use
- A, B, C... for good guys
- X, Y, Z for bad guys
- between for unknown



- When a proc terminates, it will return FAIL (which Adv sees) or output a session key (which Adv doesn't see)
- At exactly one point, Adv will "challenge" a process $\langle A, B, b\rangle$ (A & B both good guys) that has output a session key w/ prob $\frac{1}{2}$, he will be given the actual key, and $\frac{1}{2}$ a random key
- Adv can "open" (i.e. see) any non-challenge session key he wants to w/ exception: If he chatterses an $\langle A, B, b\rangle$ proc, he can't open any $\langle B, A, \bar{b}\rangle$ that oracle tells him outputs the same key
- Whenever 2 matching session keys, oracle tells Adv if they are the same
  → want to avoid Adv 'stealing'

Charlie: This def of Security requires a lot of thought. Too weak is the worst - think something is secure that isn't; too strong - cannot use what we would want to.

Adv tries to guess whether challenge key was correct or random

Charlie: so that's the definition of security.
- recall ECB → (CS) ← MAT
- Definitions can be wrong!
- "if things which you feel should be secure are not on the reverse"

{ it was definitely not the first I thought of

Charlie: no one reads the proofs in the notes. I realised that when I saw a bad typo in my notes that no one has spotted in the last 10 years.
Reminds me of the guy who wrote the book knowing no one is going to read it & offers $1000 if you spot the typo on page 450 or smthg.

① Long-term keys are only for sig. scheme
② Assume: given $g^x \bmod p$, $g^y \bmod p$, $x, y \in \{0..p-2\}$, $H(g^{xy} \bmod p)$ is indistinguishable from random
↳ This makes it such that even if Adv learns my LT key, s/he can't use it to read my old conv. history.

~~proof (given an Adv that breaks~~

Process alg

<A, B, L>

$x \leftarrow \{0..p-2\}$

$\alpha \leftarrow g^x \bmod p$

$\alpha, SIGN_A(\alpha \| B)$ → <A, B, 0>

← γ, δ <A, C, 0>

check $\gamma \in \{0..p-1\}$

$VER_B(\gamma \| A, \delta) = 1$

$H(\gamma^x \bmod p)$

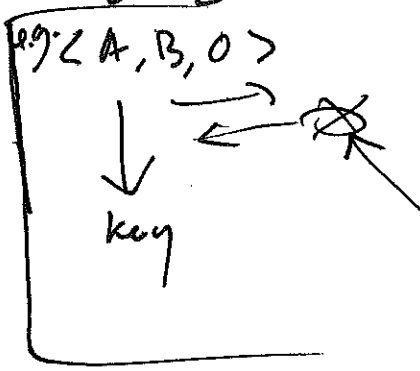
. proof that ~~this~~ protocol respects security

→ Say Adv breaks this protocol. Fix n, say prob success is $\frac{1}{2}+\varepsilon$

case 1: with prob $\geq \frac{\varepsilon}{2}$, Adv "forges" some signature
⇒ breaks sig. scheme (①)

case 2: o/w.

. Modify Adv so that it never forges anything, it succeeds with prob $\frac{1}{2}+\frac{\varepsilon}{2}$



imagine this checks out → then by the protocol ($VER_B(\gamma, A, \delta) = 1$) this has to be the signer B w/: $\langle B, A, 1\rangle$

. Say good guy names are A_1, A_2 ~ Adv' will guess i, j, b, R, S and hopes that Adv challenges $\langle A_i, B, b\rangle^R$ proc and gets its message coming from $\langle A_j, A_i, b\rangle^S$ (let $A = A_i$ & $C = A_j$) –

. Adv' has (p, g) as input, $g^x \bmod p$, $g^y \bmod p$ –
Adv' simulates Adv & chooses sig. keys correctly for all good guys. Adv' simulates all procs other than $\langle A, C, b\rangle^R$ & $\langle C, A, b\rangle^S$ correctly.

. But for $\langle A, C, b\rangle^R$, simulate it sending $g^x \bmod p$, $SIGN_A(m)$ & for $\langle C, A, b\rangle^S$, simulate sending $g^y \bmod p$, $SIGN_C(m)$
Give CH as challenge to Adv –

$\langle A, C, b\rangle^R$ → $g^x \bmod p$; $\langle C, A, b\rangle^S$ ← $g^y \bmod p$

$H(g^{xy} \bmod p)$ ↪ CH

. Now we have to give answers from oracle & from opening keys.

④ Answer oracle (how to)
When do $\langle B, D, m\rangle$ & $\langle D, B, \bar{m}\rangle$ output same key?

Claim: (almost) exactly when output of one is input of another (or we break assumption)

(2) Open keys (how to)

(i) for procs $\langle B, M, \sim\rangle$ that is not $\langle A, C, b\rangle^R$ or $\langle C, A, b\rangle^S$, since Adv' is simulating B, Adv' knows whatever key B will output.

(ii) proc $\langle C, A, b\rangle^S$:

(a) $\langle C, A, b\rangle^S$ gets input from $\langle A, C, b\rangle^R$, then Adv will not have to open this key —

(b) $\langle A, b\rangle^S$ gets input from some other proc that begins by choosing z & sending $g^z \bmod p$. Adv computes key of $\langle C, A, b\rangle^S$ as $H(g^y \bmod p)^z \bmod p$.

Charlie: "it's a very messy argument"

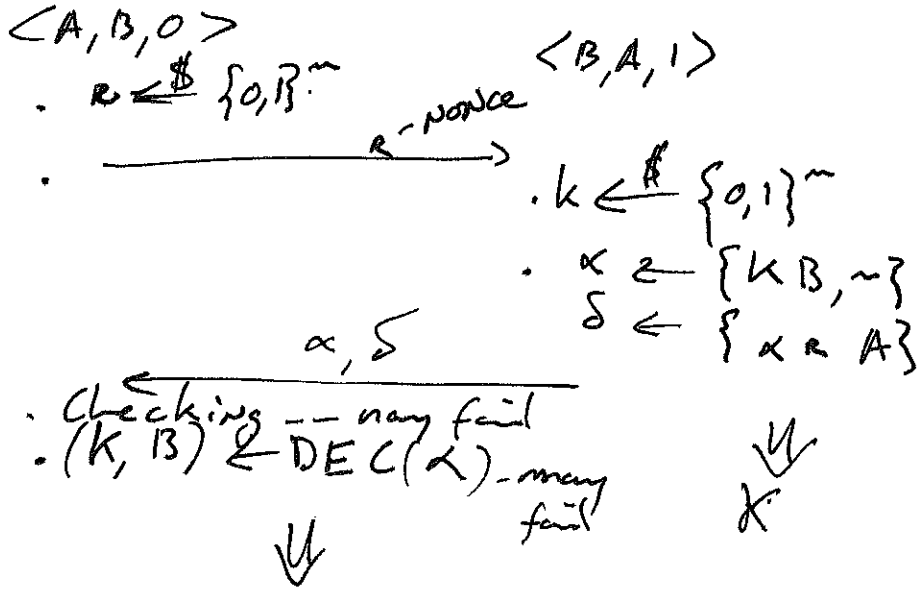
New PKI

GEN-E$(1^n, \sim)$ = (pub-e, pri-e) — randomness

GEN-S$(1^n, \sim)$ = (pub-s, pri-s)

pub = (pub-e, pub-s), pri = (pri-e, pri-s)

Alg (no forward-secrecy)



NONCE

~~a rand~~ / pseudorand word sent in sec. protocols usually to avoid replay attacks

CARBAMAZEPINE
LAMOTRIGINE
VALPROIC ACID

Security for public key encryption

① (weaker) Semantic Security:

- Adv sees pub, chooses m^0, m^1
- sees $ENC_{pub}(m^b, \text{randomness})$ — $b \leftarrow \{0,1\}$
- & tries to guess b.

[8 = Randomness]

Note: suffice to encrypt m one bit at a time — independently

② (stronger) CCA (Chosen Cipher-text Attack) security:

- Adv sees pub, makes queries DEC_{pri} (← decryption oracle)
- chooses m^0, m^1, sees $ENC_{pub}(m^b, 8) = c$ (for random b)
- queries DEC_{pri} (but not on c) —

Note: it's trivial to "maleate" encryptions that are maleable (e.g. a bit-by-bit enc.)

Charlie: "if you see CCA1, CCA2 in lit., this is CCA2. CCA1 is stupid"

Thm Say we have Adv breaking above protocol (success is $\frac{1}{2}+\varepsilon$)
Then either sig. scheme is insecure or pub key enc sch fails strong security

proof
- case 1: Adv forges sig w/ prob $\frac{\varepsilon}{2}$ ⇒ can break sig sch
- case 2: o/w: can transform Adv so that it never forges (success $\frac{1}{2}+\frac{\varepsilon}{2}$)
- subcase: Adv challenges a 0 process
  (...) don't take too much Diphenhydramine when using valerenic acid, melatonin, lamotrigine, cannabidiol, ethanol at night → antihistamine effect. Modafinil & caffeine don't help
- other subcase: Charlie: "I usually leave it as an assignment exercise"

prob(MDD | OCPD) > prob(GAD | OCPD)?

prob(DPD | SSRI)?

Charlie: "this is intrinsically boring, that's why we have crappy protocols")

Charlie: "the most secure thing you can do is write all your password on a text pad. In movies, bad guys will break into a house to steal that pad... but you are not (hesitation) important enough"

→ Charlie: "say the bad guy becomes sys admin for a day (then gets fired) and learns your password. Which primitive (learned in class) would you use to make it secure"