Classical Task: A sender (Alice) wants to transmit a message to a receiver with a potential eavesdropper (Eve) being going to it.

Alice & Bob could be different entities at diff locations or same entity at diff times (file encryption)

Follow the methodology:

1. Form a realistic model of scenario. (adjust as needed)
2. Precisely define the desired functionality & security property
3. Construct and adjust a potential system, and prove that it satisfies the properties in 2.

1st attempt @ model:

- Sender is represented by algorithm Enc(.)
  takes a plain text (message) $m \in M$ message space
  and outputs a ciphertext $c \in C$ ciphertext space

- Receiver is represented by algorithm Dec(.)
  takes $c$ and outputs $m$

Correctness: $\text{Dec}(\text{Enc}(m)) = m \text{ } \forall m \in M$

- Eavesdropper is represented by algorithm E(.)
  takes $c$ and outputs a bit $b = \{0 \text{ she doesn't care } 1 \text{ she cares what the message says}

At minimum, Eve should not always be able to output the true message $m$ (in reasonable time)

Claim: we can't achieve this within our model

Pf: Let $E = \text{Dec} \neq$

SOLUTION: Change the model! Somehow distinguish Eve from Bob $E$ from Dec)
IDEA #1: Make Dec secret. (TERrible IDEA: security by security)
Kerckhoffs principal: the system should remain secure even if
the attacker/public knows all its algorithms

IDEA #2: provide Dec with a secret input (called a key)
generated ahead of time using randomness.

2nd attempt @ model: Symmetric key model
- Sender is represented by algorithm $Enc(\cdot, \cdot)$
  - takes $k$ (key) & $m$ (message) and outputs $c$
- Receiver is represented by algorithm $Dec(\cdot, \cdot)$
  - takes $k, c$ and outputs $m$
- Key generator $Gen$ outputs some $k \in K$

- Eve gets $c$ but not $k$

Correctness: $\forall m \in M, \forall k \in K,$
$$Dec_k(Enc_k(m)) = m$$

What should "secure" mean? Maybe Eve should not be able to...
- always recover/output message $m$
- always output the real key
- always tell if $m$ is a love letter or hate mail.
- always tell if $m$ is gossip about Eve

Claude Shannon: seeing the ciphertext should give Eve no
extra info about $m$ beyond what she knew before.

Def: a system $(Gen, Enc, Dec)$ is Shannon-secret if
$$\forall \text{ distribution } D \text{ on } M, \text{ and any fixed } \bar{m} \in M, \forall \varepsilon \in \mathbb{C}$$
$$\Pr[ m = \bar{m} | Enc_k(m) = \bar{c}] = \Pr[ m - \bar{m}]$$
$$k \leftarrow Gen \quad \text{a posteriori estimate} \quad \text{a priori estimate}$$
Today: Shannon/perfect secrecy

One-time pad & limitations

**Readings:** see 2-2.3 of K-L.

\[
LHS = \frac{\Pr_{m,k}[m = \bar{m} \land \text{Enc}_k(m) = \bar{c}]}{\Pr[	ext{Enc}_k(m) = \bar{c}]}
\]

\[
= \frac{\Pr[	ext{Enc}_k(\bar{m}) = \bar{c} \land m = \bar{m}]}{\Pr[	ext{Enc}_k(m) = \bar{c}]}
\]

\[
= \frac{\Pr[	ext{Enc}_k(\bar{m}) = \bar{c}]}{\Pr[	ext{Enc}_k(m) = \bar{c}]}
\]

\[
\text{Shannon Secrecy } \iff \Pr_k[\text{Enc}_k(\bar{m}) = \bar{c}] = \Pr_{m,k}[\text{Enc}_k(m) = \bar{c}]
\]

for all \( \bar{m} \in \text{supp}(D) \)

Exam condition (\#) \( \Pr_k[\text{Enc}_k(\bar{m}) = \bar{c}] = \Pr_{m,k}[\text{Enc}_k(m) = \bar{c}] \)

for \( \bar{m} \in \text{supp}(D) \)

so \( \Pr_k[\text{Enc}_k(\bar{m}) = \bar{c}] \) is the same for all \( \bar{m} \in \text{supp}(D) \)

This implies **Perfect Secrecy**: \( \Pr_k[\text{Enc}_k(m_0) = \bar{c}] = \Pr_k[\text{Enc}_k(m) = \bar{c}] \)

\forall m_0, m, \in M

No matter what message sending, the ciphertext has the same distribution.

**Conversely**, **Perfect Secrecy** \( \Rightarrow \) **Shannon Secrecy**

\[
\Pr_k[\text{Enc}_k(m) = \bar{c}] = \sum_{m', k} \Pr_k[\text{Enc}_k(m') = \bar{c}] \cdot \Pr_k[m = m']
\]

\[
= \Pr_k[\text{Enc}_k(\bar{m}) = \bar{c}] \cdot \sum_{m' \in M} \Pr_k[m = m']
\]

\[
= \Pr_k[\text{Enc}_k(\bar{m}) = \bar{c}] \ \Rightarrow (\#)
\]

**Thm**: **Shannon Secrecy** \( \iff \) **Condition (\#)** \( \iff \) **Perfect Secrecy**
Can we achieve Shannon/Perfect secrecy? **YES!**

Miller in 1882 defined, and Vernam (1917) patented: One-time pad

Construction: \( M = C = K = \{0,1\}^l \) \( l \)-bit binary strings for some desired \( l \geq 1 \)

\[ \text{Gen}(-): \text{Choose } k \in \{0,1\}^l \text{ uniformly at random} \]

\[ \text{Enc}_k(m): \text{Output } c = m \oplus k \text{ bitwise XOR} \]

\[ \text{e.g. } m = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ k = 0 & 0 & 1 & 1 & 0 \\ c = 1 & 0 & 1 & 0 & 1 \end{bmatrix} \]

\[ \text{Dec}_k(m): \text{Output } c \oplus k \]

\[ \text{e.g. } k = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \end{bmatrix} \]

\[ c = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \end{bmatrix} \]

\[ m = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 \end{bmatrix} \]

**Correctness:** \( \forall k,m \; \text{Dec}_k(\text{Enc}_k(m)) = (m \oplus k) \oplus k = m \)

**Thm:** The OTP is Shannon/Perfect secret.

**Proof:** Let's show (\( \dagger \)): WTS \( \forall E \in \mathcal{E} \)

\[ \Pr \left[ \text{Enc}_k(\overline{m}) = \overline{c} \right] \text{ is the same for all } \overline{m} \in \mathcal{M} \]

Fix any \( \overline{c} \in \{0,1\}^l \), \( \overline{m} \in \mathcal{M} = \{0,1\}^l \)

\[ \Pr \left[ \text{Enc}_k(\overline{m}) = \overline{c} \right] = \Pr_{k \leftarrow \text{Gen}} \left[ \overline{m} \oplus k = \overline{c} \right] \]

\[ = \Pr_{k \leftarrow \text{Fixed}} \left[ k = \overline{m} \oplus \overline{c} \right] = \frac{1}{2^l} \]

This is the same for all \( \overline{m} \) & \( \overline{c} \)

**Cor:** For any \( \overline{m} \), the ciphertext \( \overline{c} = \text{Enc}_k(\overline{m}) \) is uniformly random over \( \overline{c} \in \{0,1\}^l \).
BIG caveats about OTP

1. If Alice encrypts two messages w/ same key $k$, NO SECURITY AT ALL
   
   $c_1 = m_1 \oplus k$  \quad $\Rightarrow$  \quad $c_1 \oplus c_2 = m_1 \oplus m_2$
   
   Natural language makes it easy to tease apart $m_1$ & $m_2$.
   This (mis)use falls outside the model that Shannon contemplates
   because there are 2 messages.

2. The key $k$ must be uniformly random, & as long as
   the message.
   Q: Could there be another scheme that is Shannon secret but
   with shorter keys?

THM: Any Shannon / Perfect secret system has a key space
   $|K| \geq |M|$

Pf: Proceed by contradiction. Suppose $|K| < |M|$

   Let $c$ be some arbitrary ciphertext that can arise as $Enc_k(m)$
   i.e. $Pr[Enc_k(m) = c] > 0$

   Next step, we identify $m$ s.t. $Pr[Enc_k(m) = c] = 0$

   Define a set $D = \{Dec_k(c) : \forall k \in K\} \subseteq M$

   Observation: $|D| \leq |K| < |M|$
   so there's some message $m \in M - D$
   
   Then $Pr[Enc_k(m) = c] = 0$ for sure

01/16/2019

Read: Sec 3.3.1 & 3.3.1

THM (Shannon): If $|K| < |M|$, the cryptosystem is not Shannon secrecy.

Recall from last time
Notice: if we see C, then we know the real message \( \in \mathcal{D} \). We can list \( \mathcal{D} \) by enumerating \( \mathcal{K} \). This tells us what the messages are possible & rules out all others.

Issue: if \( \mathcal{K} \) is humongous, then it's not feasible to list it in real!

Computational approach to security:

1. We only care about "feasible" attacks.
2. We need to allow a "tiny" chance of attacks success. (e.g. attacker could randomly guess key.)

Template Claim: "No feasible attackers can get more than a tiny advantage in attacking system \( \mathcal{X} \)."

Principal: define feasible very liberally

- \( 2^{10^0} \) operations (\( \approx 1 \) trillion) feasible?
  - Yes, minutes on a PC

- \( 2^{10^1} \) operations (\( \approx 1 \) quadrillion) feasible?
  - Yes, minutes on super computers

- \( 2^{10^2} \) operations feasible?
  - Yes, 1-2 years on super computers

- \( 2^{10^3} \) ops
  - Maybe, 100-200 years on super computers

- \( 2^{10^8} \) ops
  - trillion years on a super after 200 years of Moore's law.
roughly \(2^{37}\) ops/sec since the big bang.

\[2^{286} > \# \text{particles in solar system}\]

How to define tiny chance?

\[-2^{-60} \ll \text{prob that both Alice & Bob get hit by lightning in a year}\]

Concrete security: picking actual #s

but quite cumbersome to track.

Asymptotic security: instead of concrete #’s (as \(2^{128}\))

parametrize our scheme by security parameter \(n \ (or \ k, \lambda)\)

- When ”good” parties initialize system, they choose an actual value for \(n \) — Known to All.

- The ”good” algorithms (Enc, Dec) should be efficient in \(n\):
  
  (small) polynomial time

  \(i.e. \ 0(n^k), \ 0(n \log n)\)

- The ”attacker” should be ”feasible” (possibly large)
  
  polynomial time

  \(i.e. \ 0(n^{100}), \ldots\)

  but not exponential time

  not \(2^n\) nor \(2^{\sqrt{n}}\)

Write \(poly(n)\) to mean \(O(n^c)\) for some unspecified constant \(c\)

- The ”attackers” tiny advantage should be negligible:
  
  decays faster than any inverse polynomial.

\[\text{Def: a function } \varepsilon(n) \text{ is negligible, written } \varepsilon(n) = \text{negl}(n),\]

\[\text{if } \varepsilon(n) = o(n^{-c}) \text{ for all constants } c.\]
Equivalently, \( \lim_{n \to \infty} \varepsilon(n) \cdot n^c = 0 \) for a constant \( c \).

\[ \exists \varepsilon > 0 : \begin{align*}
\varepsilon(n) &= \frac{1}{n^5} & \text{not negligible} \\
\varepsilon(n) &= \frac{1}{2^n} & \text{negligible}
\end{align*} \]

**KEY IMPLICATION:** by choosing \( n \) "large enough", we can drive the \( \text{neg}(n) \) advantage to be \( \approx 0 \) holds for any attackers with some huge \( (\approx 2^{128}) \) runtime.

**TOPIC:** (Cryptographic) Pseudorandom Generators (PRG)

**Application:** let us circumvent "\( |K| = |M| \)" barrier but still keep computational security.

PRG lets us "expand" a small amount of true randomness into a large amount that is "as good as random" for any practical purpose.

**Def:** A deterministic function \( G \) (poly-time) is a PRG if following holds

1. (Expansion) \( |G(s)| = \ell(n) > n \) for all \( s \in \{0,1\}^n \)
   \( \ell(n) \) is the expansion of \( G \)
2. (Pseudorandomness) for uniformly random \( s \leftarrow \{0,1\}^n \),
   \( G(s) \) "looks like" a uniformly random \( \ell(n) \) bit string.

What should "looks like" mean?

**Turing: imitation game. (paper)**

\[
\begin{array}{c|c}
\text{Human or A.I.} & \text{Judge's advantage} = P_J \left[ J \leftrightarrow \text{A.I. says A.I.} \right] - P_J \left[ J \leftrightarrow \text{Human says A.I.} \right] \\
\hline
\text{Judge} & \\
\end{array}
\]
If \( \text{adv}(J) > 0 \), then \( J \) is distinguishing AI vs. human
Otherwise, we say AI fools the judge.

**Pseudo-randomness Game:**

\[
\begin{align*}
\text{"REAL" World:} & \quad s \leftarrow \{0,1\} \quad \text{OR} \quad \text{"IDEAL" World:} \quad y \leftarrow \{0,1\}^n \\
G(s) & \quad \downarrow \\
(\text{feasible) differentiator } D
\end{align*}
\]

**Def.** \( G \) is pseudorandom if for every (randomised) poly-time \( D \)
its advantage \( \text{Adv}_G(D) = \left| \Pr[D \left( 1^n, G(s) \right) = 1] - \Pr[D \left( 1^n, y \right) = 1] \right| 
\)

\[= \text{negl}(n) \]

Do PRGs exist? If \( P=NP \), then no

If \( P \neq NP \), then maybe

Existence of PRG \( \Rightarrow P \neq NP. \)

\(01/23/2018\)

**TODAY:** Stream Ciphers, Eavesdropper Security, Reductions

**READING:** 3.2, 3.2.2-3

**Tweak of PRGs:** stream Ciphers: can generate as many PR bits as desired

**Modeled as** \( \text{(Init, NextBit)} \)

\[
s \rightarrow \boxed{\text{Init}} \xrightarrow{\text{st}_0 \text{"state"}} \boxed{\text{NextBit}} \xrightarrow{\text{st}_1} \boxed{\text{NextBit}} \xrightarrow{\text{st}_2} \ldots \\
\downarrow y_1 \in \{0,1\} \quad \downarrow y_2 \in \{0,1\}
\]

\[
G_k(s) \equiv y_1 \ y_2 \ldots \ y_k
\]

**Requirement:** for any \( l(n) = \text{poly} (n) \), \( G_k \) should be a PRG.
**THM:** We can build a stream cipher from any PRG.

(Trivially the converse direction is trivial: building a PRG out of SC)

**Real World:** SC's are engineered.

i.e. RC4: not so pseudorandom

**Why not use real randomness everywhere?**

① True randomness may be "expensive" to get

② PRGs let us expand small shared key into a long pseudorandom string.

Can use this to circumvent "\(|K| > |M|\)" required by perfect secrecy.

First need to define security for encryption in computational setting.

Recall: For perfect secrecy: every \(c \in \text{Enc}_k(m)\) has the same distribution.

Relax: every \(c \in \text{Enc}_k(m)\) has an indistinguishable distribution.

Formulate this by a distinguishing game "EAV game"

between the attacker \(A\) and the system

\[ k \leftarrow \text{Gen}(1^n) \]

"World 0": \(m\) encrypted

"World 1": \(m\) encrypted

\[ 1^n \]

\[ m_0, m_1, \text{with the same length} \]

\[ c \leftarrow \text{Enc}_k(m) \]

\[ \text{make a guess} \]

\[ \text{Adv}^{EAV}(A) \leq |\Pr[A \text{ in } W^1 \text{ outputs } 1] - \Pr[A \text{ in } W^0 \text{ outputs } 1]| \]

Rmk: we require \(|m_0| = |m_1|\) because there's no way to encrypt a huge message as short as a small message.

We have no requirement to let Enc hide the length of the message.
Why does `A` pick `m0` & `m1`? This reflects the influence of `A` in the real world.

**DEF:** An encrypted system `TI = (Gen, Enc, Dec)` is (one-time) indistinguishable under eavesdropping attack (EAV-secure) if every PPT attacker `A` has negligible advantage in this EAV-game against `TI`.

Construct a system `TI = (Gen, Enc, Dec)` out of a PRG `G` with expansion factor `l(n) > n`:

- `Gen(1^n)`: choose uniformly random key `k ∈ \{0, 1\}^n`
- `Enc(m)` where `m ∈ \{0, 1\}^{l(n)}`: output `c ← m \oplus G(k)`
- `Dec_k(c)`: output `c \oplus G(k)`

**Exercise:** generalize to use stream cipher and allow any msg length.

**THM:** If `G` is a PRG, then `TI` described above is EAV-secure.

**PROOF:** by Reduction: we will show that any EAV attacker `A` against `TI` can be transformed into an attacker `D` against the PRG `G` and their advantages are closely related.

Let `A` be any PPT EAV-attacker vs. `TI`. We'll build `D` as follows to simulate the EAV-game to `A`.

\[
\text{Adv}(D) = \left| \Pr \left[ D(y) = 1 \right] - \Pr \left[ D(y) = 1 \right] \right|
\]
\[ \Pr [A \text{ in } W_1 \text{ outputs } 1] - \Pr [A \text{ in } \text{"random c world" outputs } 1] \]

^ because D is perfectly simulating World 1.

\[ \text{Adv} (A) = | \Pr [A \text{ in } W_1 \text{ outputs } 1] - \Pr [A \text{ in } W_0 \text{ outputs } 1] | \]

\[ \leq | \Pr [A \text{ in } W_1 \text{ outputs } 1] - \Pr [A \text{ in } \text{"random c world" outputs } 1] | \]

\[ + | \Pr [A \text{ in } \text{"random c world" outputs } 1] - \Pr [A \text{ in } W_0 \text{ outputs } 1] | \]

**01/28/2019**

TODAY: Chosen-Plaintext Attack (CPA) security, Pseudorandom functions (PRFs)

A stronger security notion: CPA game: parameters by \( b \in \{0, 1\} \)

\( (W_0 \text{ or } W_1 \text{ (left or right)}) \)

\[ \begin{array}{c}
R_{k,b} (\cdot, \cdot) \\
\downarrow 1^n \\
\vdots \\
c \leftarrow B_n(m) \\
\downarrow \text{decision } b' \\
\hline \\
A
\end{array} \]

1. \( A \) is given the security parameter \( 1^n \) and a key \( k \leftarrow \text{Gen}(1^n) \) is generated
2. \( A \) can make (adaptive) queries to an oracle \( LR_{k,b} (\cdot, \cdot) : LR_{k,b} (m_0, m_1) = \text{Enc}_k (m_b) \)
3. Eventually, \( A \) makes a decision \( b' \in \{0, 1\} \)

Important Features:
1. Encryption of many messages under same key \( k \)
2. \( A \) chooses exactly which msgs to encrypt.

Define \( A \)'s advantage:

\[ \text{Adv}_{\Pi}^{\text{cpa}} (A) = | \Pr [A^{LR_{k,1} (\cdot, \cdot)} \text{ accepts}] - \Pr [A^{LR_{k,0} (\cdot, \cdot)} \text{ accepts}] | \]

\( \& \text{ "right" world} \)

\( \& \text{ "left" world} \)
"fake" Theorem: No CPA-secure encryption system exists

Proof: Let T be any encryption system
we'll construct an PPT attacker A which runs the CPA-game
with advantage 1

A choose two arbitrary message m0, m1 with |m0| = |m1|
Query L, Rk,1(·, ·) with (m0, m0), receive ciphertext C0
Query L, Rk,1(·, ·) with (m0, m1), receive ciphertext C1
if C0 = C1, output W0, otherwise output W1

BUG in the proof: L, R encrypts m0 twice ⇔ C0 = C1
that's not true!

1. Stream Cipher: L, R will keep the state St until next call
   and update the state to St+1

2. Enc could be a randomized algorithm

THM: No CPA-secure encryption with a stateless deterministic Enc(·) exists

Possibility:

Is CPA too strong?
No, because the same ciphertext could leak some info.
Main tool for achieving CPA-secure encryption: PRF (or block cipher)

Motivation: \( \text{Enc}_k(m) = m \oplus \text{G}(k) \)

PRF will give us a "huge" # of random looking bits
also random access to them

PRF looks like a random function.

A random function (or randomly generated function)
\( U: \{0,1\}^n \rightarrow \{0,1\} \) has every output
uniquely random independent of others.

<table>
<thead>
<tr>
<th>Small example:</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>0 1 0</td>
<td></td>
</tr>
<tr>
<td>0 0 1</td>
<td>1 0 0</td>
<td></td>
</tr>
<tr>
<td>0 1 0</td>
<td>1 0 1</td>
<td></td>
</tr>
<tr>
<td>0 1 1</td>
<td>1 1 1</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>1 1 1</td>
<td>1 0 1</td>
<td></td>
</tr>
</tbody>
</table>

02/04/2019

TODAY: PRFs & CPA-secure cryptography

Readings: Section 3.5, 3.6.1

Key idea: a small secret key will determine a huge (exponential)
pool of "random-looking" bits

Recall: A truly random function \( U: X \rightarrow Y \)
has each \( U(x) \in Y \) uniform & independent for each \( x \in X \)

Think \( X = Y = \{0,1\}^n \)

Can think of \( U \) by its "lookup table" of \( n \cdot 2^n \) (random) bits
As \( n \) grows, very likely to have repeated outputs
(Defer: can you find some efficiently)
Model for a PRF is a key function

\[ F: K \times X \rightarrow Y \]

the key \( k \in K \) determines a function \( X \rightarrow Y \) entirely

(Think \( K = X = Y = \{0,1\}^n \))

WANT: if we pick a secret key \( k \) random
then \( F_k(\cdot) \) should look like a random function.

What should "look like" a random function \( U \) mean?

IDEA #1: the lookup table of \( F_k(\cdot) \) should be indistinguishable from
the lookup table of a truly random function.

Downside: the lookup table is exponential while the attacker
is poly-time. No one can look through the whole table

IDEA #2: Allow unlimited queries to an oracle \((F_k \text{ or } U)\)

PRF game of an efficient (ppt) distinguishes \( D \) against keyed function \( F \)

either \( O = F_k(\cdot) \) for random key \( k \) ("Real")
or \( O = U \) for truly random function ("Ideal")

\( D \) gets the security parameter \( 1^n \)

\( D \) can make many queries \( xi \), receive \( O(xi) \)

\( D \) makes a decision

DEF: A keyed function \( F \) is a PRF if every ppt distinguisher \( D \)
has negligible advantage in the PRF-game.

\[
Adv^\text{PRF}_F(D) \triangleq \left| \Pr_{k \in \{0,1\}^n}[D \text{ accepts } F_k(\cdot)] - \Pr_{U \in \{0,1\}^n}[D \text{ accepts } U]\right|
\]
Candidate PRF: \( F_k(x) = x \oplus k \) is not a PRF

\[ D : \text{query } y_0 = \mathcal{O}(0 \cdots 0) \]
\[ y_1 = \mathcal{O}(1 \cdots 1) \]

test if \( y_0 \oplus y_1 = 1 \cdots 1 \)

\[ \text{then } \text{Adv}(D) = \left| 1 - \frac{2^n}{2^n \times 2^n} \right| = 1 - \frac{1}{2^n} \text{ nonnegligible} \]

Do PRF exists?

**Theorem:** PRF exists \(\iff\) PRG exists

PRF Candidates that experts design and try to break

E.g. DES (1976-7), \( K=\{0,1\}^{56} \), \( X=\{0,1\}^{64} \)

- can brute-force all keys

AES (2001-2), \( K=\{0,1\}^{\text{or } 128} \)

appears to be PRF because of many attempts to break

Now: use PRF \( F \) for CPA-secure encryption (stateless)

Key idea: each value \( F_k(x) \) (for distinct \( x \))

"looks like" a fresh random string to attacker

but \( A \& B \) can compute it

For each message to encrypt, choose a random \( x \)
and use \( F_k(x) \) as pad to conceal message

because each \( x \) is random, no need to remember
what we used before
Assume we have a PRF $F: K \times \{0,1\}^n \rightarrow \{0,1\}^n$.

**Gen** ($1^n$): choose $k \leftarrow K$ at random
**Enc** ($m \in \{0,1\}^n$): choose random $X \leftarrow \{0,1\}^n$, output ciphertext $c = (x, c' = m \oplus F_k(x))$

**Dec** ($c = (x, c')$): output $c' \oplus F_k(x)$ (deterministic)

**Correctness:** Easy.

**CPA-game:**

```
\[ k \leftarrow \text{Gen}(1^n) \quad 1^n \rightarrow \]

\[ \text{LR}_{k, b} \leftarrow \begin{cases} \text{Gen}(1^n) & \text{"left" or "right" oracle} \end{cases} \]

\[ c \leftarrow \text{Enc}(m) \]

\[ \rightarrow A \quad (\text{CPA}) \]

\[ \rightarrow \text{decision } \epsilon \{0,1\} \]

In this case, $c = (x, F_k(x) \oplus m)$

or $c = (x, U(x) \oplus m)$

02/06/2019

"Birthday Paradox": general formula

$N$ possible birthdays, $q$ "people"

\[ \Pr[\text{two "people" equal}] = \prod_{i=0}^{q-1} \frac{\frac{N-i}{N}}{\frac{N-i}{N}} \approx \frac{\frac{q}{2}}{N} \approx \frac{q^2}{2N} \]

Last time: Ti = (Gen, Enc, Dec) for a PRF $F: K \times \{0,1\}^n \rightarrow \{0,1\}^n$

**Gen** ($1^n$): choose $k \leftarrow K$

**Enc** ($m \in \{0,1\}^n$): choose random $r \leftarrow \{0,1\}^n$, output

$c = (r, c' = m \oplus F_k(r))$

**Dec** ($r, c'$): output $c' \oplus F_k(r)$

**THM:** If $F$ is a PRF, this Ti is CPA-secure.

**PROOF:** IDEA: $F_k(\cdot)$ "looks like" random $U(\cdot)$

so every $F_k(r)$ for distinct $r$ "looks like" a truly random independent pad.
Formally, go by reduction.

Let $A$ be any PPT attacker vs. $T_1$ in a CPA-game.

**GOAL:** $\text{Adv}_{\text{CPA}}^\text{CPA} (A) = \text{negl}(n)$

Construct $D_0$ around $A$

(oracle)

either $\mathcal{O} = F_k$

or $\mathcal{O} = \mathbb{U}$

$\xrightarrow{1^n} r$

$\xleftarrow{\text{randomly choose } r}$

$P = \mathcal{O}(r)$

$c = (r, m_0, P)$

$D_0 (\text{vs. PRF})$

repeat poly time

$A (\text{CPA})$

$\xrightarrow{1^n}$

Analyze $D_0$'s advantage

"REAL" to $D_0 \Rightarrow "LEFT"$ to $A$

"IDEAL" to $D_0 \Rightarrow "HYBRID"$ to $A$

problem: $(r, m_0 \oplus U(k))$ is not complete random

i.e. what if the same $r$ occurs twice?

we have to assume that $D_0$ is not

choosing the same $r$ twice.

because the chances that this happen is small.

What is $\Pr [D_0 \text{ choose some } r \text{ values more than once?}] \approx \frac{\text{poly}(n)^2}{2 \cdot 2^n} = \text{negl}(n)$

$\Rightarrow \Pr [D_0^{U(1)} \text{ accepts}] = \Pr [A \text{ in HW accepts}] \pm \text{negl}(n)$

$\text{Adv}_{\text{PRF}}^\text{PRF} (\mathcal{O}) = | \Pr [A \text{ in LW accepts}] - \Pr [A \text{ in HW accepts}] | \pm \text{negl}(n)$

Now design $D_1$ with $m_0$ replaced $m_0$.

$\Rightarrow \text{Adv}_{\text{PRF}}^\text{PRF} (D_1) = | \Pr [A \text{ in RW accepts}] - \Pr [A \text{ in HW accepts}] | \pm \text{negl}(n)$

$\Rightarrow \text{Adv}_{\text{CPA}}^\text{CPA} (A) = | \Pr [A \text{ in LW accepts}] - \Pr [A \text{ in RW accepts}] |$

$\leq \text{Adv}_{\text{PRF}}^\text{PRF} (D_0) + \text{Adv}_{\text{PRF}}^\text{PRF} + \text{negl}(n) = \text{negl}(n)$
What about arbitrary lengths?
IDEA: invoke $F_k(\cdot)$ on many different inputs to get a longer pad.

$Enc_k (m \in \{0,1\}^*)$ : choose random $r \in \{0,1\}^n$

- break up $m = m_0 \parallel m_1 \parallel m_2 \parallel \cdots \parallel m_t$
  - where each $|m_i| = n$
  - (except maybe $|m_t| \leq n$)

- output $C = r \parallel m_0 \oplus F_k(r) \parallel m_1 \oplus F_k(r+1) \parallel \cdots \parallel m_t \oplus F_k(r+t)$$

$Dec_k (C = r \parallel m_0 \parallel m_1 \parallel \cdots \parallel m_t)$: compute $P = F_k(r) \parallel F_k(r+1) \parallel \cdots \parallel F_k(r+t)$

  XOR $C' = C_0 \parallel C_1 \parallel \cdots \parallel C_t$ with $P$

**THM:** this "counter mode" (CTR) is CPA-secure if $F$ is a PRF.

**PROOF:** idea: we never (or negl(n) rarely) invoke $F_k$ on the same input twice.

02/11/2019

(Continued) 1) In a single call to $Enc_k$, there cannot be repetition because the message is only poly(n) blocks

2) across different calls to $Enc_k$, because IV is random for each call, we have a $\frac{poly(n)}{2^n} \approx negl(n)$ prob of a repeated $F_k$ input (by birthday paradox)

⇒ because $F$ is PRF, all outputs of $F_k$ we use look like a random, independent pad

This is efficiently a fresh OTP for each encryption msg #
Thesis today: For CPA security, CTR is all you need. It’s:

1. simple: just generated ‘stream’ ("offline") \( F_k(N+1), \ldots \)
and XOR with messages

2. Parallel: we can Enc & Dec any chunk of m/c in parallel
with others

can also decrypt any chunk of c without doing the rest

3. "Padding free": no need to lengthen ("pad out") m to be multiple
of block length

DEF: A length-preserving key function \( F: K \times \{0,1\}^n \rightarrow \{0,1\}^n \)
is called a block cipher if each \( F_k \) is a bijection and
each \( F_k \) could be efficiently computed if \( k \) is known

DEF: \( F \) is a pseudorandom permutation (PRP) if \( F_k \)
"looks like a random permutation", i.e. for each efficient
PPT attacker \( A \)

\[
\text{Adv}_{F}^{\text{PRP}}(A) \leq \left| \Pr(A^{F_k}(c) \text{ accepts}) - \Pr(A^{P} \text{ accepts}) \right|
\]
is negligible

THM: Any PRP is a PRF, but not the opposite way

"PROOF": Idea: A random permutation is almost a random function
except for negligible chances that it has two equal outputs on
different inputs.

Candidate PRPs: -DES not PRP @ small keyspace, @ small input size

- AES 128, 192, 256 \( \checkmark \) (good so far)

Notes of operation can use PRPs for decryption, but treat as PRFs
for security

Ex 4) Electronic Codebook (ECB) mode:

\( \text{Enc} (\cdot) \) pad m to be an exact multiple of block length n
write $m = m_1 \| m_2 \| \ldots \| m_t$ where $|m_i| = n$

$$
\begin{align*}
& \quad c = c_1 \quad c_2 \quad \ldots \quad c_t \\
& \quad \text{Dec}_k (c) = m_1 \quad m_2 \quad \ldots \quad m_t
\end{align*}
$$

**THM**: ECB is not CPA-secure

This is totally insecure

**PROOF**: $\text{Enc}_k$ is deterministic & stateless.

**EX**: Cipher Block Chaining (CBC) mode

$\text{Enc}_k (m \in \{0,1\}^n)$: padding $m$ to $m = m_1 \| m_2 \| \ldots \| m_t$ where $|m_i| = n$

Choose random $IV \leftarrow \{0,1\}^n$, then

$$
\begin{align*}
& \quad IV \rightarrow m_1 \\
& \quad \downarrow \quad \downarrow \\
& \quad F_k \quad F_k \\
& \quad \downarrow \quad \downarrow \\
& \quad c_1 \quad c_2 \quad \ldots \quad c_t
\end{align*}
$$

$C_0 = IV$

$C_i = F_k (C_{i-1} \oplus m_i)$

**THM**: If $F$ is a PRP, then CBC is CPA-secure.

**PROOF idea**: Across the CPA-game, all inputs are different to $F_k$

Thus, all outputs look like random independent strings.

**Caveats**:
1. Requiring padding message
2. $\text{Enc}$ is sequential
Brittle attacker slightly outside CPA-attack mode if attacker can choose later mi block based on early Ci blocks, secrecy can be lost.

```
EX 3: Output Feedback Mode (OFB)
```

\[ \text{Enc}_k(m \in \{0,1\}^n) \quad m = m_1 \parallel m_2 \parallel \cdots \parallel m_t \quad \text{where } |m_i| = n \quad \text{except } |m_t| < n \]

\[ \text{choose } IV \in \{0,1\}^n \]

\[
\begin{array}{c}
\text{IV} \\
\downarrow \quad \downarrow \quad \downarrow \\
F_k \\
\downarrow \quad \downarrow \\
P_2 \\
\downarrow \\
P_3 \\
\text{...}
\end{array}
\]

\[
\begin{array}{c}
\text{IV} \\
\downarrow \\
\text{XOR} \\
\downarrow \\
0 \\
\downarrow \\
m_1 \\
\downarrow \\
m_2 \\
\downarrow \\
\cdots \\
\downarrow \\
m_t \\
\downarrow \\
\text{IV} \\
\downarrow \\
C_1 \\
\downarrow \\
C_2 \\
\downarrow \\
\cdots \\
\downarrow \\
C_t
\end{array}
\]

\[
C_i = m_i \oplus F_k^{(i)}(IV) \quad \text{where } F_k^{(i)} = \underbrace{F_k \circ F_k \circ F_k \circ \cdots \circ F_k}_i \text{ times}
\]

Pro: 
- Fast-free
- Precompute "offline"

Cons: 
- Enc/Dec is sequential non parallelizable
- THM: it's CPA-securce

02/13/2019

TODAY: Message Authentication Codes (authenticity)

Reading: sections 4.1, 4.2, 4.3.1

A message authentication code (MAC) is:

1. \text{Gen}(1^n): outputs a key \( k \gets K \)

2. \text{Tag}_k(m): outputs the same tag \( t \in T \) ("tag space")

3. \text{Ver}_k(m', t'): either accepts (legit!) or reject (fishy!)

Correctness: \( \text{Ver}_k(m, \text{Tag}_k(m)) \) accepts always \( \forall k \in K, m \in M \)
What should be considered by security?

At least: Eve cannot produce a good $t$ for a new $m'$ of her choice after seeing some good $(m,t)$ produced by Alice.

We don't hope to rule out "replay attack" by MAC.

**DEF**: the chosen message attack (CMA) game of $F$ against a MAC works as follows: $\mathcal{O}$ $1^n \rightarrow t$

1. A key $k \leftarrow \text{Gen}(1^n)$ is queried (kept secret from forger).
2. $F$ can get tags for messages of its choice by querying an oracle that implements $\text{Tag}_k(\cdot)$
3. $F$ outputs an (attempted) forgery $(m^*, t^*)$

$F$ "wins" if

- $\text{Ver}_k(m^*, t^*)$ accepts
- $m^*$ is not queried to $\text{Tag}_k(\cdot)$ oracle - it "fresh"

**DEF**: A MAC is unforgeable under CMA (UF-CMA) if for PPT forger $F$, $\text{Adv}_{\text{MAC}}(F) = \Pr_{k \leftarrow \mathcal{K}} [\text{Ver}_k(m^*, t^*) \text{ wins } | k \leftarrow \text{Gen}(1^n)]$ is negligible in $n$.

Simple construction of a MAC for fixed length messages $\in \{0,1\}^k$ using a PRF $F : K \times \{0,1\}^k \rightarrow \{0,1\}^n$

$\text{Gen}(1^n)$: choose $k \leftarrow K$ at random

$\text{Tag}(m \in \{0,1\}^k)$: output $t = F_k(m)$

$\text{Ver}(m' \in \{0,1\}^k, t' \in \{0,1\}^n)$: accept if $t' = F_k(m')$

$\text{reject}$ otherwise

**THM**: The MAC constructed above is UF-CMA if $F$ is a PRF.

**PROOF**: Let $F$ be any forger, we need to show $\text{Adv}(F)$ is negligible.

We build a $D$ that attacks $F$
02/18/2019  

Readings: 4.3.1, 4.3.2, 4.4


Replay attack works: msg itself should have some
“freshness” mechanism
  e.g., date & time as part of msg itself
  e.g. remembers the msg sent/received

2. Bob keeps a list of (m, t) pairs. Every time he receives
  a msg pair, he checks if that is in his list or not.

CMA doesn’t rule out coming up with a new tag for an
  old message

Home Chris could reply the old message with new tags

We call this a “weak forgery”

A defn rules out “weak forgery” is called “strong unforgeability”
Redefine “forges” to mean 1. Verk(m*, t*) accepts
2. (m*, t*) is not part of a queried pair
THM: If $Tag_i(\cdot)$ is deterministic & $Ver_x$ is canonical then strong unforgeable is equivalent to (ordinary) unforgeable

3. Mallory tries to fool Bob with a lot of msg $(m_i, t_i)$ observes whether Bob accepts/rejects

If the tag is unique, then Mallory already forges a tag hence he doesn't need a response from Bob

If NOT, there's pathological MACs that help Mallory

Goal: work for arbitrary length messages

Candidate ①: "tag" each block independently

$$m = m_1 \| m_2 \| \ldots \| m_s \quad \text{where} \quad |m_i| = l \quad \text{(after padding)}$$

$Ver$ is canonical

Insecure! Reorder attack!

②: "tag" with order attached

$$m = m_1 \| \ldots \| m_s \quad t_i = F_k (m_i \| <i>)$$

where $|m_i| = \frac{l}{2}, \quad <i> \text{ is } \frac{l}{2} \text{-bit encoding of } i$

Insecure! Truncation attack! query on $m_1 \| m_2 \Rightarrow t_1 \| t_2$

③: "tag" with msg length & index of each block

$$m = m_1 \| \ldots \| m_s \quad \text{where} \quad |m_i| = \frac{l}{3}$$

let $t_i = F_k (m_i \| <i> \| \langle L \rangle)$ where $L = |m|$ is total block $\langle L \rangle, \langle i \rangle$ is $\frac{l}{3}$-bit encoding.

Insecure! query $m = m_1 \| m_s \Rightarrow t_1 \| t_2$

& $m' = m_i \| m_\ell \Rightarrow t_1 \| t_2$ where $m_i \neq m_\ell$

outputs

$m^* = m_1 \| m_2$

$t^* = t_1 \| t_2$

Mixed & Match Attack or alternate attack
Including a random message ID in each block

\[
\text{Tag}_k(m) := m_1 || m_2 || \cdots || m_t \quad \text{where } |m_j| = \frac{1}{l}
\]

Choose random \( r \in \{0,1\}^{\frac{1}{l}} \)

let \( t_i = F_k(m_i || r || i || <\gamma> \) \where \( \gamma = |m_j| \)

-output \( (t=t_1 || t_2 || \cdots || t_t , r) \) as a tag.

\textbf{THM:} This is CMA-secure assuming \( F \) is PRF \& \( l \) is large enough to avoid a birthday attack.

\textbf{PROOF:} by exhaustion: any possible forgery must contains a block \( F_k(m^* || r^* || <\gamma> \) \) that was not previous queried which is impossible.

In real life, we (sometime) use CBC-MAC based on a length-preserving PRF:

\[
\text{Tag}_k(m \in \{0,1\}^{l\cdot n}) := t_0 = 0^n, \quad t_i = F_k(t_{i-1} \oplus m_i) \quad \text{for } i=1,2,\ldots,l
\]

\[
IV=(t_0=0^n) \rightarrow \oplus \rightarrow t_1 \rightarrow t_2 \rightarrow \cdots \rightarrow t_l \rightarrow F_k(t_l) \rightarrow t_l \text{ as a tag}
\]

1. \( IV = 0^l \) not random
   Can't let IV vary!

2. The tag is just \( t_1 \), no more!

02/20/2019

\textbf{TODAY:} CBC-MAC wrap up, Authenticated Encryption

\textbf{Readings:} 4.5 - 4.5.3

\textbf{THM:} if \( F \) is a PRF, then CBC-MAC is UF-CMA secure for messages of fixed length \( s.n: \) (for desired \( s \geq 1 \), fixed in advance)

all tags & verified messages must be this length.
For variable message length:

1. Don't use CBC-MAC
2. If you must, use a prefix-free encoding to allow variable-length messages

Encoding \( m \rightarrow \hat{m} \) and allows recovery of \( m \) from \( \hat{m} \)

prefix-free: for any distinct \( m, m' \in \{0,1\}^* \), \( \hat{m} \) is not a prefix of \( \hat{m}' \)

Ex1: ordinary string is not prefix-free

Ex2: prepend a block representing the msg length, and fill zeros to last block:

\[
\begin{align*}
\text{e.g. } \quad m = 001 & \quad \rightarrow \quad \hat{m} = 011 0010 \\
\text{n = 4} & \quad \begin{array}{c}
\text{3} \\
\text{m + 0}
\end{array}
\end{align*}
\]

Claim: \( \hat{m} \) is prefix-free.

Often, the sender & receiver want both confidentiality & authenticity of messages together: Encryption gives conf, but not auth
MAC gives auth, but not conf

Idea: combine/compose encryption with MAC to get both or neither!

This time Dec either decrypts correctly, or outputs \( \bot \) (error)

Correctness: \( \text{Dec} (\text{Enc}(m)) = m \) (in particular, not \( \bot \)) \( \forall k, m \)

What security properties do we want?

1. Ciphertext should reveal nothing about \( m \) (apart from length)
   (CPA-security \( \checkmark \))

2. Attacker shouldn't be able to provide valid ciphertexts on its own.

Def: An encryption scheme \( T_1 = (\text{Gen}, \text{Enc}, \text{Dec}) \) is unforgeable if \( \forall \text{ ppt } A \)

\[
\text{Adv}^\text{unf}_{T_1} (A) \triangleq \text{Pr}[A_{\text{Enc}(1^n)} \text{ forges}] = \text{negl}(n)
\]
where "forges" means output $c^* \neq t$. 1. $\text{Dec}_k(c^*) \neq t$
2. $c^*$ was not an answer to any $\text{Enc}_k(\cdot)$ query.

**Def.** $T_{I} = (\text{Gen}, \text{Enc}, \text{Dec})$ is an authenticated encryption (AE) scheme if it satisfies
1. CPA security
2. Unforgeability

Seek a generic combination of an CPA-secure encryption $T_{I}$ and a UF-CPA MAC

Why generic: 1. If one of the system happen to be insecure then its easy to replace that part
2. Better understanding

Choose $k_{E} \leftarrow \text{Gen}(1^{q})$, $k_{m} \leftarrow \text{Gen}(1^{q})$ independently randomly

Combination #1: "encryption and tag"

$\text{Enc}_{k_{E}, k_{m}}(m)$: $c = \text{Enc}_{k_{E}}(m)$, $t = \text{Tag}_{k_{m}}(m)$, output $(c, t)$

$\text{Dec}_{k_{E}, k_{m}}(c, t’)$: compute $m’ = \text{Dec}_{k_{E}}(c)$, check if $t’ \overset{?}{=} \text{Tag}_{k_{m}}(m’)$

This is (almost) never secure: because $\text{Tag}$ is insecure and it will reveal some info. of $m$ so it destroys CPA-security.

This is also forgeable under some circumstances

Combination #2: "tag then authenticate"

$\text{Enc}_{k_{E}, k_{m}}(m)$: $t = \text{Tag}_{k_{m}}(m)$, output $c = \text{Enc}_{k_{E}}(m | t)$

$\text{Dec}_{k_{E}, k_{m}}(c’)$: compute $m’ | t’ = \text{Dec}_{k_{E}}(c’)$ and verifies $\text{Ver}(m’, t’)$

This is not CPA-secure as the length of $t$ is not fixed so $c$ reveals some info about length of $t$ hence some info about $m$

i.e. say $|\text{Tag}_{k_{m}}(m)| > |\text{Tag}_{k_{m}}(m_i)|$

then $(m, m_i)$ is distinguishable as $|c_i| > |c_j|$

This is not unforgeable (Homework)
Combination #3: "Encrypt then tag"

\[ Enc_{\text{km},k_E}(m) : c \leftarrow Enc_{k_E}(m) \text{, then } t \leftarrow Tag_{\text{km}}(c) \text{, output } c\texttt{lit} \]

\[ Dec_{k_E,k_m}(c,t') : \text{ if } Ver_{k_m}(c,t') \text{ accepts, then output } Dec_{k_E}(c) \]

02/25/2018

TODAY: Wrap up AE, Cryptographic Hash functions

Readings: 5-5.2, 5.4.1

THM: "encrypt-then-tag" is a secure AE scheme if both schemes are secure.

Criticism: 
1. extra \(2^{256}\) bits from msg to ciphertext
2. \(\text{msg} \) generates two keys \(k_E, k_m\) need to be independent
3. two passes over the data: one to encrypt, one to tag

Twice the work, could be bad if msg is big (e.g. DVD)

In practice: we prefer special AE mode operation that make

pass with one key

E.g. 0: Galois Counter Mode (GCM) [supplanted IAPM]

0: Offset Codebook Mode (OCB)

warning: OCB v2 not CPA nor authentic!

3. etc.

Cryptographic Hash Functions

Idea: want some (deterministic) \(H: \text{UNIVERSE} \rightarrow \text{HASH-VALUES}\)

that "spread" out data values roughly evenly over the

hash values.

we really want few collisions: two data values \(x \neq x'\)

where \(H(x) = H(x')\)
In crypto, we ask for stronger: **there should be no collisions among our data values even if a malicious attacker chooses our data based on what the hash value is.**

Model: a hash function is a deterministic function
\[ H: K \times X \rightarrow \gamma \quad \text{where} \quad |X| > |Y| \]

- hash key \( K \)
- input \( X \)
- output \( Y \)

i.e. the input is “larger” than the output

(Formerly, \( K, X, Y \) can depend on the security parameter)

Also comes with a key gen alg \( \text{Gen}(1^n) \), chooses a key \( k \in K \)

Typically: \( Y = \{0, 1\}^n \quad |Y| = 2^n \)

\( X = \{0, 1\}^\infty \quad |X| = \infty \)

When \( X \) is of fixed-length strings, then \( H \) is a compressing family.

by pigeonhole principal, there must be collisions

More over, roughly approximate \( \frac{|X|}{|Y|} \) elements map to the same image.

(i.e. \( \frac{|X|}{|Y|} = 2^n \) if \( |Y| = 2^n, |X| = 2^{2n} \))

Def: A hash family \((\text{Gen, } H)\) is collision resistant, if any PPT \( A \), given the key of the hash function,

\[ \text{Adv}^C_H(A) = \Pr \left[ A(k) \text{ outputs collisions} \right] = \text{negl}(n) \]

**Critical:** the attacker \( A \) gets the key as explicit input!

In typical scenarios, all parties know the hash key it’s chosen by some “trusted” party & used by the world. (SHA-1, 2, 3 all have this in)

we have to allow this party not to embed “backdoor” that would
allow generating collisions.

We often say \( H_k \) itself is CR even if the key \( k \) is fixed.

A weaker notion:

**Def:** A hash function \((Gen, H)\) is 2nd preimage resistant (2PR)

or twist collision resistant (TCR)

if \( \forall \text{ PPT } A, \text{ Adv}^{\text{2PR}}_H(A) = \left[ \Pr_{k \leftarrow \text{Unif}([n]), x \leftarrow X} \left( A(k, x) \text{ outputs } x' \neq x \text{ s.t. } \forall_i H(x_i) = H(x'_i) \right) \right] = \text{negl}(n) \)

**THM:** C.R. \( \Rightarrow \) 2PR \( \Leftarrow \) Proof left as an exercise.

Generic attacks on C.R / 2.P.R. Say \( H: K \times X \rightarrow Y = \{0, 1\}^d \)

(0) Pigeonhole attack on C.R.

Given a hash key \( k \), choose \( 2^d + 1 \) different inputs \( x_i \in X \)

**WORK** \( \approx 2^d \) hash evaluations

(1) Birthday attacks on C.R.

Given hash key \( k \), choose \( q \approx 2^{\frac{d}{2}} \) random different inputs \( x_i \in X \)

Look for two that collide under \( H_k \)

\[ \Pr[\text{ collision happens } ] \approx \frac{q^2}{2^{2d}} \]

Suppose \( l = 128 \) (MD5) \( \Rightarrow \) \( \approx 2^{64} \) work feasible!

\( l = 160 \) (SHA-1) \( \Rightarrow \) \( \approx 2^{80} \) work edge of feasible!

Take \( l \geq 256 \) to resist birthday attack, i.e. SHA-2, -3

(2) Attack on 2.P.R.

given hash key \( k \), and a target \( x \).

Choose many random \( x_i \in X \)

\[ \Pr[ H_k(x_i) = H_k(x)] = \frac{1}{|Y|} = 2^{-l} \]

Need to try roughly \( 2^l \) \( x_i \)'s to get a collision.

So for 2.P.R., \( l = 128 \) is fine.
These are generic; work against any hash functions.

Specific hash functions may admit better attacks.

Candidate 1: \( H: \{0,1\}^* \times \{0,1\}^n \to \{0,1\}^n \)

\[ H_k (x = x_1 \parallel x_2) = k \oplus x_1 \oplus x_2 \]

Then \( H_k (x_1 \parallel x_3) = H_k (x_1 \parallel x_1) \) (Neither C.R. nor 2.P.R.

(2) say \( F: \{0,1\}^* \times \{0,1\}^n \to \{0,1\}^n \) is a PRF

let \( H = F \)

\( A(k): \) for particular \( F, \) it could be that \( F_k(k) = F_k(k+1) \)

this doesn't conflict with PRness.

A PRF is NOT necessarily a hash func.

02/27/2019

TODAY: Application of Hash Functions

Readings: See 5.2, 5.3

Recall: Def - (Gen, H) when \( H : K \times X \to Y \) is a hash family

if \( |X| > |Y| \) ("compression")

and is collision-resistant if \( \forall \) PPTA

\[ \text{Adv}_{H}^{\text{c.r.}} (A) = \Pr \left[ A(k) \text{ accepts distinct } x \neq x' \mid \begin{array}{l}
\text{Gen} \{1^n\} \\
\text{s.t. } H_k(x) = H_k(x')
\end{array} \right] \]

= \text{negl}(n)

"Length Extension of Hash Functions"

Suppose have compression functions \( h: K \times \{0,1\}^n \to \{0,1\}^n \)

\[ \text{fixed length input} \]

Convert this into a full hash function \( H: K \times \{0,1\}^\ast \to \{0,1\}^n \)

Use the Merkle-Damgard transform

Gen: stays the same
\( H_k(x) \in \{0,1\} < 2^n \) : let \( B = \left\lceil \frac{\text{length}}{n} \right\rceil \) where \( L = |x| < 2^n \)

pad \( x \) with 0s to make its length a multiple of \( n \)

write \( x = x_1 \parallel x_2 \parallel \ldots \parallel x_n \) where \( |x_i| = n \)

\[
\begin{align*}
Z_0 &= 0^n \\
Z_i &= h_k(x_{i-1} \parallel x_i) \quad 1 \leq i \leq B+1
\end{align*}
\]

output \( Z_{B+1} \)

**THM**: If \((\text{Gen}, h)\) is C-R, then \((\text{Gen}, H)\) is C-R.

**Proof**: by reduction: let \( A \) be any PPT attacker against C-R of \((\text{Gen}, H)\), we build \( A' \) against C-R of \((\text{Gen}, h)\)

let \( k \overset{\text{Gen}}{\rightarrow} k \)

\[
\begin{array}{c|c}
\text{let} & A(k) \text{ to get } x \neq x' \\
\hline
x \neq x' & A \text{ (us. } H) \\
A' \text{ (us. } h) & \end{array}
\]

if \( H_k(x) = H_k(x') \), \( A' \) use method below to find \( w \neq w' \) s.t. \( h_k(w) = h_k(w') \). And \( A' \) succeeds iff \( A \) succeeds, so they have the same advantage.

\( A' \) looks at the hash computation of \( H_k(x) \) and \( H_k(x') \)

say \( B = \left\lceil \frac{L}{n} \right\rceil \), \( B' = \left\lceil \frac{L'}{n} \right\rceil \) where \( L = |x|, \ L' = |x'| \)

\[
\begin{align*}
Z_0 &= 0^n \\
X_1 &\rightarrow h_k \\
X_2 &\rightarrow h_k \\
&\ldots \\
X_B &\rightarrow h_k <L> \rightarrow h_k \rightarrow Z_{B+1}
\end{align*}
\]
Application of hash functions: a lot... focus on integrity & authenticity applications

1. Alice sends a big file x via an unreliable channel.
2. Bob checks that if $H_k(x) = h$ and accepts if so.

Note: No secret keys involved.

3. arbitrary message/input length for MAC/PRF

Suppose we have PRF $F : K \times \{0,1\}^n \rightarrow \{0,1\}^d$.

**c-r hash** $H : X \rightarrow \{0,1\}^n$

Define: $F' : K \times X \rightarrow \{0,1\}^d$

$F'_k(x) = F_k(H(x))$

**THM**: If $F$ is a PRF & $H$ is a c-r hash, then $F'$ is a PRF.

(also works for MACs: hash msg before tagging preserves unforgeability of the MAC)

$$\begin{align*}
&x_1' \\
&z'_0 = 0^n \\
&\xrightarrow{h_k} \\
&\xrightarrow{h_k} \\
&\ldots \\
&\xrightarrow{h_k} \\
&\xrightarrow{h_k} \\
&z_{n+1}'
\end{align*}$$

If $|x| \neq |x'|$, then $h_k(z_B \| \langle \angle \rangle) = h_k(z_B' \| \langle \angle \rangle)$

But $z_B \| \langle \angle \rangle \neq z_B' \| \langle \angle \rangle$

Otherwise, look at $z_B \neq z_B'$: """$ightarrow$ done

look at $z_B \| z_B' 
eq z_B \| z_B' : """$ightarrow$ done

look at $z_{n+1} \| z_{n+1} \neq z_{z_{n+1}} \| z_{z_{n+1}} : \ldots$ keep going

If all """$ightarrow$"" happens, then $x = x'$, which is not possible.

Then there must be a collision of $h_k$. ✗
Proof: by reduction, say $A'$ is a PPT attacker against $F'$ in a PRF game build $A$ to attack $F$ in a PRF game

1. Real: $O = F_k$, then $A$ simulates the real world to $A'$
2. Ideal: $O = U$, then $A$ simulates the function $U(H(x))$ is as good as a random function because except the negligible probability that $H(x_i) = H(x_j)$ for $x_i \neq x_j$, we can always assume $H(x_i) \neq H(x_j) \Rightarrow$ the output is uniform and dependent $\Rightarrow$ uniform random

The rest is formal

3. Use a hash function alone as a MAC

   "Freshman Construction": $Tag_k(m) = H(k \| m)$

   IDEA: the presence of the key $k$ in input makes output unpredictable
   This is completely forgeable if $H$ follows Merkle-Damgård construction.

   Use a length extension attack: query on $x_1$, get $t = H(k \| x_1)$
   which is $h(<1> \| h(x_1 \| k \| 0)) : = t$
   Now output $m = x_1 \| x_2$ with tag $h(<2> \| h(t \| x_2))$

   Good way to get MAC from hash function

   $HMAC$: $Tag_k(m) = H((k \oplus \text{pad}) \| H((k \oplus \text{pad}) \| m))$

   Inf. THM: If $H$ has suitable PRness & the freshman construction is a secure MAC for fixed length msgs, then HMAC is unforgeable for arbitrary length msgs.

03/11/2019

TODAY: Exam Review

Many Categories:
Modeling encryption & Shannon secrecy
2 Pseudorandomness (PRGs, PRFs, PRPs)
3 Compl'y secure encryption (ZAUS, CPA-security)
4 Message Authentication (MACs), authenticated encryption (AE)
5 Hash functions, their properties & application

In each category:
6 Definition of security properties, usually in terms of "attack game"
7 Constructions of various objects (using underlying components)
8 (Often) reduction
9 (Other times) direct attack.

03/18/2019

TODAY Public-key Crypto & Number Theory

Readings: See 10.1-10.3, 8.1-8.1.3

So far we've assumed A & B have a secret random key (Ev doesn't)

REALISTIC? Maybe: if they can meet in private, sure!

NO: If they've never met, like for agents
... Google ... Apple ... Amazon...

NO #2: a separate secret key for every pair of communicating entities is unfeasible.

"Key Distrib Center" can mitigate "NO #2" for closed systems, where
we can trust the KDC completely
Central point of failure, knows all keys.

Early-mid 1970's: do sender & receiver really need to have a
shared secret key?

Basic model for asymmetric, or public key crypto:
related but inequivalent

When e.g. Bob wants to send an encrypted message to Alice:

he uses $p_{ka}$ (public) to encrypt, yielding message $c$

Alice uses secret key $sk_{a}$ to decrypt $c$

(Others, lacking $sk_{a}$, can't learn about message)

1976: Diffe and Hellman posed this goal and achieved part of it

1977: Rivest, Shamir, Adleman filled in a major other piece

Many other public-key systems

All successful ones have been built on Number Theory

Topics: Modular Arithmetic, Groups, efficient algorithms, Chinese Remainder Theorem, ...

TODAY: pub-key encryption

Model: of asymmetric (pub key) encryption \( T = (Gen, Enc, Dec) \)

- \( Gen(1^n) \): outputs a public/secret key pair \((pk, sk)\)
- \( Enc(pk, m \in M) \): given a public key (encryption) key \( pk \) & msg \( m \in M \), outputs a ciphertext \( c \)
- \( Dec(sk, c \in C) \): given a secret (decryption) key \( sk \) & ciphertext \( c \), outputs a message \( m \in M \)

Correctness: for all \((pk, sk) \leftarrow Gen(1^n)\) and any \( c \leftarrow Enc(pk(m))\)

\[
\forall m, \quad Dec(sk(c)) = m
\]

Public: \( pk \)

Alice encrypts

\( c \leftarrow Enc(pk("Hi Bob")) \)

Bob decrypts

\( m \leftarrow Dec(pk(c)) \)

"Hi Bob"

Public key is intended to be public.

WHAT’S IMPORTANT is the sender knows the actual key of the intended receiver.

Public keys must be **authentic**, but not **secret**

- a **passive** attacker can’t prevent sender from getting \( pk \)
- an **active** attacker could replace legit \( pk \) with its own \( pk \)
For now, assume sender possesses the legit pk.

Recall: In symm settings, we had EAV & CPA security.

Define a left-or-right game/ oracle for pub-key

The attacker is given the public key.

**CPA-Attack game of A against Tl**

$$(pk, sk) \leftarrow \text{Gen}(1^n)$$

$$(1^n, pk) \rightarrow$$

$$\begin{array}{c}
\text{LR} \\
\leftarrow \text{Enc:pk}(m) \\
\rightarrow c \leftarrow \text{Enc:pk}(m) \\
\rightarrow \text{decision} \\
\end{array}$$

$$A$$

**DEF:** a public key encryption $Tl = (\text{Gen}, \text{Enc}, \text{Dec})$ is CPA-secure if a PPT $A$

$$\text{Adv}^c_{\text{com}}(A) \leq | \Pr[A(pk) \text{ in left world accepts}] - \Pr[\text{...right...}] |$$

is negligible.

A generic attack: $A(pk)$ queries on $(m_0, m_1)$ $m_0 \neq m_1$ gets back $c$

Do $c_0 \leftarrow \text{Enc:pk}(m_0)$, check $c_0 \neq c$ if yes, accepts

no, rejects!

This attack has perfect advantage IF $\text{Enc:pk}$ is deterministic.

**THM:** no scheme with deterministic $\text{Enc:pk}(\cdot)$ can be CPA-secure.

**THM:** CPA security is an equivalent notion, whether we allow

a single LR query or

any poly(n) number (determined by $A$)

**PF idea:** poly(n) $\Rightarrow$ single is trivial.

Now one query $\Rightarrow$ many queries:

Consider a PPT adversary $A$ who is allowed to make $q(n) = \text{poly}(n)$
ATTACKER can create encryption of its own using $pk$

LEFT: when $A$ queries $(m_0, m_1)$, it gets back $c \leftarrow \text{Enc}_{pk}(m_0)$ always

HYBO

HYB1: first query is answered by $c \leftarrow \text{Enc}_{pk}(m_0)$;
    subsequent queries $c \leftarrow \text{Enc}_{pk}(m_0)$

HYB2: first & second query is answered by $c \leftarrow \text{Enc}_{pk}(m_1)$;
    subsequent queries $c \leftarrow \text{Enc}_{pk}(m_0)$

HYB $\frac{\%}{(i)}$

RIGHT: when $A$ queries $(m_0, m_1)$, it gets back $c \leftarrow \text{Enc}_{pk}(m_1)$ always

Adjacent hybrids HYB(i-1) and HYB(i) differ only in the $i$th query.

So we can design a reduction $Si(pk)$ built on one-query CPA

```
\text{Si aims to simulate } HYB(i-1) \text{ or } HYB(i) \text{ based on its LEFT/RIGHT world.}

\text{when } A \text{ makes its } j \text{th query}

\begin{align*}
\text{return } \text{Enc}_{pk}(m_j) \text{ if } j < i \\
\text{return } \text{Enc}_{pk}(m_j) \text{ if } j = i \\
\text{return } \text{Enc}_{pk}(m_j) \text{ if } j > i
\end{align*}
```

Finally, \[ \text{Adv}_{\Pi}^{\text{CPA}}(A) \leq \sum_{i=1}^{2} \text{Adv}_{\Pi}^{\text{CPA}}(S_i) = \delta(n). \text{negl}(n) = \text{negl}(n). \]

Col: WLOG, we can encrypt "long" messages $m \in \{0, 1\}^*$

bit-by-bit under a bit encryption scheme (where $M = \{0, 1\}$)

If bit encryption is CPA-secure, then so is this bit-by-bit encryption.

03/27/2019

TODAY: El Gamal & RSA (Encryption)

Readings: 11.4.1, 8.2.1, 8.2.3-4, 11.5.1
The Diffie-Hellman protocol has all we need.

Six years later, El Gamal sees how to rearrange DH to get PKE.

**DH protocol:** have cyclic group $G$, known order $q$, generator $g$

$\begin{align*}
\text{Alice} & \quad \text{choose random} \quad a \in Z_q \\
A & \quad \text{choose random} \quad A = g^a \\
\text{Bob} & \quad \text{choose random} \quad b \in Z_q \\
B & \quad B = g^b
\end{align*}$

WANT: the key $K$ should be "random looking" to an attacker who sees $g, A, B$.

(Re)Arrange DH protocol to get El Gamal's PKE:

**Gen:** choose random $a \in Z_q$; output $(pk= A = g^a, sk = a)$ (Alice)

**Enc:** $(pk = A, M \in G)$ choose random $b \in Z_q$; output ciphertext

$\quad (B = g^b \in G, M \cdot A^b)$

**Dec:** $(sk = a, (B, C))$; output $(B^a)^{-1}C$

**Correctness:**

$(B^a)^{-1} \cdot A^b \cdot M = (g^a)^{-1} \cdot (g^b) \cdot M = M$

Can we compute $K^{-1}$? Compute $K^{g^{-1}} = K'$

*Is El Gamal secure? Intuitively, if $K$ is random, then $C = K \cdot M$ is independent of $M$.*

**DDH (Decisional Diffie-Hellman) assumption on $(G, g, g')$**

The tuple $(g, A = g^a, B = g^b, K = g^{ab})$ is indistinguishable from $(g, g^c, g^b, g^c)$ for some random $c \in Z_q$

$\ast : \text{DH tuple} \quad \ast \ast : \text{random tuple}$

**THM:** If DDH holds for group $G = \langle g, g' \rangle$, then El Gamal on $G$ is CPA-secure.

**PROOF:** by reduction, let $A$ be any feasible CPA attacker.
v.s. \( El \) Gamal.

We'll build a DDH distinguisher around \( A \).

\[
\begin{array}{c}
(g, A, B, C) \\
A = g^a \\
B = g^b \\
C = g^c \text{ or } g^{ab}
\end{array}
\]

\begin{itemize}
  \item \( D \) vs DDH.
  \item If \( D \)'s input was a DH tuple (i.e. \( K = g^{ab} = C \))
    then \( D \) perfectly sim's LEF's CPA world to \( A \).
  \item If \( D \)'s input was a random tuple
    \( D \) sim's HYB world where c'text is uniform random
    Make \( D \) use \( M \) to connect HYB to \( \text{RIGHT} \)
    \[
    \text{Adv}_{\text{CPA}}^D (A) \leq \text{Adv}_{\text{DDH}}^D (D) + \text{Adv}_{\text{DDH}}^D (D')
    \approx \text{negl.}
    \]
\end{itemize}

Another: Rivest - Shamir - Adelman '77 uses very different method to get PKE

\& pub-key authentication (now!)

Math backgroungs: let \( N = p \cdot q \)

Then \( \mathbb{Z}_N^* = \{ a \in \mathbb{Z}_n \mid \gcd(a, N) = 1 \} \)

\textbf{2019/04/01}

Claim: A lot of PRGs are insecure

\( G : \{0, 1\}^n \rightarrow \{0, 1\}^{2^n} \) a PRG is never

\( G(s) = G_0(s) \cdot G_1(s) \)

Fact: at least one of \( G_0(s) \), \( G_1(s) \) is far from uniform

\# possible \( G_0(s) \) is \( < 2^n \)

\# \( G_1(G_0(G_1(s))) \) becomes really small: \( \text{poly}(n) \)

\( \text{"fail loops" } \Rightarrow \text{ April fool} \)
We suppose that \( \exists e \in \mathbb{Z}^+ \) s.t. \( \gcd(e, \phi(N)) = 1 \).

Extended Euclid gives us \( A \cdot e + B \cdot \phi(N) = 1 \) \( \Rightarrow A \cdot e = 1 \pmod{\phi(N)} \).

Such a choice of \( N, e, d \) gives us RSA function and its inverse DEF.LEM for \( N = P^g \) (distinct primes \( P \& g \)) and \( e \in \mathbb{Z}^*_N \) with \( d = e^{-1} \pmod{\phi(N)} \).

The RSA function : \( \text{RSA}_{N,e} : \mathbb{Z}^*_N \rightarrow \mathbb{Z}^*_N \) is a bijection \( x \mapsto x^e \).

RSA is a "trapdoor" function : given \( N, e \) it's easy to calculate \( x^e \) given trapdoor \( d \) : it's easy to calculate its inverse.

Without trapdoor, hope that it's hard to calculate its inverse.

The RSA key-generation procedure

Gen RSA \( (1^n) \) : ① Choose random int primes \( P \& g \) length depending on \( n \). [It's possible]

② Define \( N = P^g \). Also compute \( \phi(N) = (P-1)(g-1) \).

③ Choose \( e > 1 \) s.t. \( \gcd(e, \phi(N)) = 1 \). Gets \( d = e^{-1} \) for free.

④ Output \( pk = (N, e) \) \( sk = (N, d) \).

How to choose \( e \)?

① \( e = 3 \) requires \( 3 \nmid P-1 \) \& \( 3 \nmid g-1 \).

② \( e = 2^k+1 \) it's prime, and efficient for repeated squaring.

③ Choose randomly from \( \mathbb{Z}^*_N \).

DEF: the RSA (hardness) assumption is : \( \forall \text{PPT } A \)

\[
\text{Adv}_{\text{RSA}}(A) \leq \Pr_{(pk, y) \leftarrow \text{Gen}(1^n)}[A(pk, y) \text{ outputs } x = \text{RSA}_{pk}(y)] = \text{negl}(n)
\]
In words, given a RSA pub key \((N,e)\) & a uniq random \(y \in \mathbb{Z}_N^*\)

it is infeasible to compute the (unique) preimage \(x = RSA_{m,e}^{-1}(y)\).

How plausible is this RSA assumption?

1. RSA \(\leq\) Factoring big \(N\)

2. Finding \(\varphi(N)\) given \(N \equiv\) Factoring \(N\)

   because knowing \((p-1)(q-1) = N - p - q + 1\)

   \(\Rightarrow\) knowing \(p + q\), we also know \(pq\) done!

3. Finding \(d\) given \((N,e) \equiv\) Factoring \(N\)

   Know \(d \cdot e \equiv 1 \mod \varphi(N) \Rightarrow d \cdot e - 1 = k \cdot \varphi(N)\)

   It does not work, can find \(p, q\) from this

4. Computing \(y^d = y^{e^{-1}}\) given \((N,e, y) \equiv\) Factoring \(N\)

   \(\not\equiv\) Factoring \(N\)

The best known attack on RSA is to factor \(N\).

Textbook RSA” PKE

\(\text{Gen}(1^*):\) generate \(pk = (N, e), sk = (N, d)\)

\(\text{Enc}(pk = (N, e), m \in \mathbb{Z}_N^*)\) output \(c = RSA_{N,e}(m)\)

\(\text{Dec}(sk = (N, d), c \in \mathbb{Z}_N^*)\) output \(RSA_{N,d}(c)\)

Intuition: security: by RSA assumption, hard to find \(m\) given \((pk, c)\)

Conclusion: Textbook RSA is not CPA-secure.

Other RSA encryption:

\(\text{Gen}(1^*):\) as before

\(\text{Enc}(pk = (N, e), m \in \{0,1\}^*):\) choose random \(r \in \mathbb{Z}_N^*\)

- output \(c = (y = RSA_{N,e}(r) = H(r) \oplus m)\)
Dec ($sk = (N, d)$, $c \in \{0, 1\}^k$): compute $r = RSA_{N, d}(y)$ and output $H(r) \oplus c$

CPA security: If we model $H$ as a “random oracle” then this is CPA-secure under RSA assumption.

04/03/2019

**TODAY:** Digital Signature & Random Oracles

**Readings:** Section 12.1-12.4

Encryption provides confidentiality, not authenticity

**Ex:**
1. Info on ID is correct, as attested to by government
2. Someone’s last will
3. Financial contract agreed to by parties
4. Campus-wide email from president/dean.

None of these work with symmetric MAC - need asymmetry

Model for digital signature scheme

$T = (Gen, Sign, Ver)$

Gen($1^n$): outputs a verification key $vk$ (public)

a signing key $sk$ (private)

Sign($sk, m$): given sign key $sk$ & msg $m$, output “signature” $\sigma$

Ver($vk, m, \sigma$): given $vk$ & msg $m$, purported signature $\sigma$, accept/rej

This allows public to verify signatures, not just those who knew a secret

**Correctness:** $\forall (sk, vk) \in$ Gen($1^n$, $\forall m \in M$):

Ver($vk, m, Sign(sk, m)$) accepts always.

$\forall$ Ver can’t work like Sign because doesn’t know $sk$
Security: (for a MAC, wanted to be infeasible to forge a tag for any \( m \) that wasn’t already tagged – even after seen several tags for messages they choose)

**DEF:** A scheme \( T \) = (Gen, Sign, Ver) is UF-CMA if \( \forall \) ppt \( T \)

\[
Adv_{uf}^{uf}(T) \triangleq \Pr[ \text{Sign} \Rightarrow (vk) \text{ forges} ] = \operatorname{negl}(n)
\]

where “forges” means output \((m^*, o^*)\) s.t. \( \text{Ver}(pk, m^*, o^*) = 1 \)

\( \text{UF-CMA } \iff (m^*, o^*) \) is not queried & \( m^* \) is not queried before.

"Textbook" RSA signature:

\[
\text{Gen}(1^k) : (N, e, d) \leftarrow \text{GenRSA}(1^k)
\]

\[
\text{Sign}(sk = (N, d), m \in \mathbb{Z}_N^*) \ \text{output} \ o' = \text{RSA}_N,d (m) = m^d \mod N
\]

\[
\text{Ver}(pk = (N, e), m^*, o^*) \ \text{check} \ m^* \overset{?}{=} \text{RSA}_N,e (o^*) = o'^e \mod N
\]

This is no-message-attack forgeable:

output \((m^* = (o^*)^e, o^*)\) then it verifies

**Objection:** \( m^* \) is not a "meaningful" msg; \( T \) doesn’t control it.

**Answer 1:** who says what’s meaningful! The application

**2:** meaningful forges: query \( m \rightarrow o' \)

query \( m' = 2 \rightarrow o'' \)

output \((2 \cdot m, o', o'')\)

**IDEA:** remove \( T \)'s ability to control values going into \( \text{RSA}_N,e \)

**Assume:** "crazy" hash function \( H : \{0,1\}^* \rightarrow \mathbb{Z}_N \)

\( \text{msgs} \overset{\text{RSA domains}}{\rightarrow} \text{truly random function that's publicly computable} \)
"Hash-and-sign" RSA signatures:

\[ \text{Gen}(1^n) : (N, e, d) \leftarrow \text{GenRSA}(1^n) \]

\[ \text{Sign}(sk = (N, d), m \in \mathbb{Z}_N^*) \quad \text{output} \quad \sigma' = \text{RSA}_{N, d}(H(m)) \]

\[ \text{Ver}(pk = (N, e), m^*, \sigma^*) \quad \text{check} \quad H(m) = \text{RSA}_{N, e}^\ast(\sigma^*) = \sigma'^e \mod N \]

**THM:** If we model \( H \) as a random oracle & RSA assumption holds, then hash-and-sign RSA signs are SUF-CMA.

**Q:** How do you know \( pk \) belongs to Alice? Not from Eve?

**A1:** make Alice give it directly

**A2:** Certification Authorities (CAS) pub-key infrastructure

\[ \text{CA}(vk_c, sk_c) \]

\[ \text{Bob} \]

\[ \text{Alice} \]

Ver _pkA (“pkA \in Alice”, \( \sigma^A \))

Then Bob needs to know the authenticity of \( vk_c \)

But we reduced of checking many _pkA's_ to getting one valid \( vk_c \)

04/08/2019

**TODAY:** Identification Protocols & Discrete Log-Based Signatures

**Read:** 12.5

Recall: RSA "hash-and-sign" to sign \( m \), hash if \( \sigma = \text{RSA}_{N, e}^\ast(H(m)) \) using secret trapdoor \( d \)
If $H$ is a "random oracle", then this is UF if RSA hardness assumption holds.

Real World: no $H$! Instead, "mimic" $H$ with a real crypto hash function, then hope that it's secure!

Could we get signs from discrete log/DH style problem?
These don't (appear to) have trapdoors: no trapdoor info makes DLOG easy.

**TODAY:**
1. Construct ID schemes from DLOG.
2. Construct ID $\rightarrow$ signature scheme (using random oracle).

**Recall:** HUGE group $G$ of known prime order $q$ & with known generator $g$.

**1) Schnorr ID scheme:** A "prover" $P$ has a secret key $x \in \mathbb{Z}_q$ and a public key $y = g^x \in G$.

**Goal:** allow $P$ to convince a verifier that it knows the secret key $x = \log_y y$.

$$P^k(x) \quad \text{choose } k \in \mathbb{Z}_q$$
$$\quad \text{choose } c = g^k \in G$$
$$\quad \text{choose } r \in \mathbb{Z}_q$$
$$\quad \text{compute } s = k + rx \mod q$$
$$P^k(y) \quad \text{check } g^s \neq g^k \cdot (g^x)^r = c \cdot y^r$$

**Claim:** if $P^k$ answers more than one possible challenges, then $P^k$ knows $x$.

**Claim 2:** Schnorr protocol is "eavesdropper zero knowledge" an eavesdropper watching all the messages $P \leftrightarrow V$. 
learns nothing about \( x = \log_2(y) \), nor about anything it didn't know.

Two ways that Schnorr ID scheme differs from a signature scheme:
1. no message \( m \) involved
2. interactive b/w \( P \times V \) (Signature scheme has noninteractive sign alg)

Address both with a random oracle \( H : \{0,1\}^* \rightarrow \mathbb{Z}_q \)

**KEY IDEA:** “collapse” the \( V \)’s challenge by letting \( r = H(\text{first msg } c) \)

Now \( P(x) \) does on its own:
1. pick \( k \in \mathbb{Z}_q \), let \( c = g^k \in G \)
2. let challenge \( r = H(c) \)
3. output \( s = k + r \cdot x \mod q \)

“Collapsed” signature scheme:

\[
\text{Gen : choose } k = x \in \mathbb{Z}_q \text{, let } v_k = y = g^x \in G
\]

\[
\text{Sign (ck=x, m \in \{0,1\}^*) : choose } k \in \mathbb{Z}_q \text{, let } c = g^k \in G
\]

\[
\text{let } r = H(m, c)
\]

\[
\text{let } c = k + r \cdot x \mod q
\]

\[
\text{output } (r, s) \in \text{signature}
\]

\[
\text{Ver (vk=y, m \in \{0,1\}^*, o=((r, s)) : compute } c = g^k \cdot g^{r \cdot x} \in G
\]

\[
\text{check if } H(m, c) = r
\]

**THM** The Schnorr - Fiat - Shamir (Collapsed) is UF-CMA if DLOG is hard for \( G \) & \( H \) is a random oracle.

04/22/2019

**Potpourri / Topics / Future**

1. **Password Authenticated key exchange (PAKE)**
   Passwords used everywhere / everywhere