ECS 127 – Cryptography – Winter 2019 – Phillip Rogaway

These are rough notes to keep me on track in lecture. They are written for me, not for students. Use at your own risk, knowing that the notes will not explain everything and may diverge greatly from what we actually do in class.

Lecture 1 - ECS 127 - Spring 2016 – 1/9/2019

Today: o Admin stuff:
o Introduction:
- Four "basic" problems of cryptography

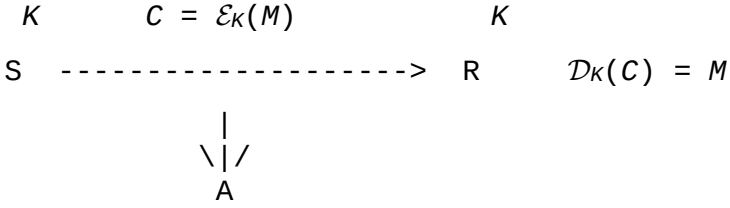
Admin stuff

- Course homepage: search "Rogaway"; follow the 127 link. **https://tinyurl.com/ecs127-spring19** class webpage, incl. course info sheet: go to my homepage.
- Modest over-enrollment. It will clear
- Homework – work with at most one partner
- Academic misconduct warning: an "F" grade for any academic misconduct.
- Needed background: mathematical maturity. Understanding ECS 20 or MATH 108 is the *minimum*.
- We will not follow any book. This is a problem for some students. Understanding what is presented in the classroom is key. You will be in trouble if you miss lectures.
- Videos: Do not rely on them.

Introduction

4 Basic problem of cryptography

Basic sample cryptographic problems: start with the "basic five" {sym, asym} x {priv, auth}. Explain! Eg:



Introduce basic vocabulary: **plaintext**, **ciphertext**, **key**, **signature**,

MAC, public key, symmetric, asymmetric, digital signature, encryption, decryption.

Discuss the **implicit assumption** in this picture (A doesn't control what's encrypted, encryption is deterministic).

Discuss that which is necessarily leaked in the picture.

Discuss alternative ways to create asymmetry (mostly suggested by students):

1. Multiple channels not all of which are tapped (secret-sharing approach)
2. Interaction (SKE-first approach)
3. Noisier channel for the adversary than the good player
4. Meet up in person

Lecture 2 - ECS 127 – Winter 2019 – 1/9/2019

Today: o Introduction, part 2: beyond the "basic four"
1 Secret Key Exchange (SKE) and Authenticated Key Exchange (AKE)
2 Secret-sharing
3 MPC/SFE. Average-salary special case. Physical soln, math. soln
4 FHE
5. Obfuscation

Review
1. Four "core" problems

Quiz questions:
1) What did we call the tool usually used for achieving **authenticity** in the **symmetric** setting?
2) Suppose we are secret-sharing a byte among 10 people. What's the smallest prime field we can use? (ans: Z_257)

Secret Sharing:
(Share, Recover). Message space \calM, say 0,…, m-1. Let S be secret we want to share. Share(S) → S_1, …, S_n such that any k of them let us recover the secret S; but k-1 or fewer of them tell you nothing. A (k,n) threshold scheme.
First show how to do 1-out-of-2 secret sharing.
Solution: choose k-1 random numbers in Z_p (p to be determined) a_1, …, a_k-1 and let f(x)=s + a_1 x + x_2 x^2 + … + a_{k-1} x^{k-1} mod p. Give party i the value f(i).
To recover: we have k points on a polynomial (j, f(j)).
Fundamental Theorem of Algebra: k points on a polynomial determine a unique polynomial of degree <= k-1. This is true in any **finite field**.
Finite field: a field has two operations, called *addition* and *multiplication*; it is an abelian group under the addition, with 0 as additive identity; the nonzero elements are an abelian group

under the multiplication, with 1 as multiplicative identity; the multiplication is distributive over the addition.

Two new problems.

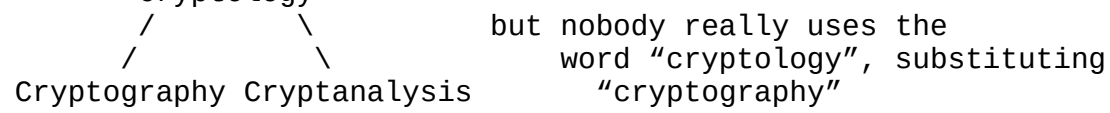
1. **Average salary** – Explain problem. Physical soln. Eg: pour 1ml H_20 for every $1000/yr.
earned. Assumes a prior bound – the size of the vessel into which water is poured.

give a communication-based solution. (Goldreich, Micali, Wigderson 1987)/ Use Group of integers mod N. Introduce: modular arithmetic: the ring $\mathbf{Z}_n$.

What is cryptography all about:
- Rivest: "communications in the presence of an adversary"

Basic character

- Tiny field, but very well known
- **Traditional taxonomy**



- **Alternative taxonomy**
  1. Academic folks
     Crypto community
     Academic users of cryptography
     (people who do security, privacy, …)
  2. Industry people
  3. Cypherpunks and hobbyists
  4. Spooks: military (NSA)

- **Yet another taxonomy**
  1. Practical protocols
     for conventional aims

for exotic aims
2. Impractical protocl for “exotic” aims

The character of cryptographic work

a. crypto-for-security -- industry side
b. crypto-for-privacy -- tiny, cypherpunks, PETS community
c. crypto-for-crypto -- crypto community
d. crypto-for-power -- intelligence agency.
Other end of crypto-for-privacy

Academic crypto: Small. Well recognized. Subtle. Mathematical. Often of limited utility.

Cryptographic activities:

Recognize - a new problem
Define (definitional enterprise) - making definitions
Construct - making abstract schemes
Prove
Standardize
Design SW/HW artifact
Implement
Cryptanalyze (break) a scheme
Cryptanalyze (break) a system
Contextualize - legally, politically,
Critique - not much of this goes on

- Academic crypto: tons of recognition:
10/64 Turing awards.

- Hard. Defeat universal quantifier. "Don't try this at home."

- Many people think cryptography = encryption. Seriously wrong.
Cryptography includes encryption, but is much broader.

- Many people think that cryptography = foundations of computer security. Also wrong.

- Strong military history. Some date cryptography back to Julius Caesar

- Subtle. Some books do not get across the subtlety of this subject)

- Kerckhoffs's principle:

A cryptosystem should remain secure even if everything about the system, except the key, is publicly known.

Negation: "Security through obscurity"

"Kerckhoffs said neither 'publish everything' nor 'keep everything secret'; rather, he said that the system should still be secure even if the enemy has a copy." (Steve Bellovin, 2009)

[Wikipedia] Original paper (**1883**) named six design principles:

1. The system must be practically, if not mathematically, indecipherable;
2. **It should not require secrecy, and it should not be a problem if it falls into enemy hands;** (The only part that people remember)
3. It must be possible to communicate and remember the key without using written notes, and correspondents must be able to change or modify it at will;
4. It must be applicable to telegraph communications;
5. It must be portable, and should not require several persons to handle or operate;
6. Lastly, given the circumstances in which it is to be used, the system must be easy to use and should not be stressful to use or require its users to know and comply with a long list of rules.

"Kerckhoffs' principle applies beyond codes and ciphers to security systems in general: every secret creates a potential failure point. Secrecy, in other words, is a prime cause of brittleness—and therefore something likely to make a system prone to catastrophic collapse. Conversely, openness provides ductility." B. Schneier, 2002

Philosophical/theoretical problem with Kerckhoffs's principle: its formal meaninglessness. One can always regard the key as specifying the algorithm, in the spirit of a universal TM.

Lecture 3 - ECS 127 - Spring 2016 - 4/1/2016

Today:

- **Finite fields, less rushed**
- **Secure Multiparty Computation (MPC)(SMC)**
- **Probability review**

Finite fields (less rushed).

Background on finite fields: First review finite fields: a set ***F*** with operations $+, \cdot$ satisfying the usual properties: ***F*** under addition is a group (the identity denotes 0); F\{0} under multiplication is a group

(its identity denoted 1); and multiplication distributes over addition. Finite fields: the underlying set is finite.

A well-known theorem of algebra says that there are finite fields of size p^α for any prime p and number $\alpha \geq 1$; that there are no other finite fields; and that each of these fields is unique, up to the naming of elements. So when I describe them, you know all the finite fields. The field with N elements is denoted $\boldsymbol{F}_N$ or GF(p^α). Eg, GF(2^{128}).

```
  0 1 2 3 4
0 0 0 0 0 0
1 0 1 2 3 4
2 0 2 4 1 3
3 0 3 1 4 2
4 0 4 3 2 1
```

```
f(x) = x^3 + 1   over GF(5)
f(0) = 1
f(1) = 2
f(2) = 4
f(3) = 3
f(4) = 0
```

Go back to the 1-of-2 secret sharing. Can use to transmit a bit. This is the **one-time pad**.

Generalize to transmitting a two-digit number secure.

Multiparty computation problem.
Special case of **computing the average**.
First, a physical-model solution

Review:
Last time: Dating Problem and the Avg Salary Problem.
Give new soln. for dating problem: “Alice and Bob agree:
if you want to go on a date, show up at such-and-such
coffee shop at such-and-such time.”

Substitution cipher:
Our first encryption scheme. Lousy one, but it is an encryption scheme.

Plaintext is regarded as a sequence of characters over some alphabet Σ: message $M \in \Sigma^*$.

[tutorial: briefly review **formal languages**: alphabet, strings, Σ^*]

For a substitution cipher, the encryption key K specifies a permutation $e(x)$: $\Sigma \to \Gamma$.
(one-to-one and onto function)
Let $\mathcal{K}$ be the set of all keys, ie, all permutation on Σ. How big is this set?

[[tutorial on **permutations**.
Recall definition of a permutation. Explain how to procedurally understand it. Use Σ={a,...,Z}
What is |*K*|? d! where d=|\calK|.
If we think of \calK as "ASCII characters": 128! \approx 10^215
If we think of \calK = {a,...,z} then 26! \approx 2^88
(how to approximate 128! on your calculator?
ln(n!) ≈ n ln(n) // Stirling's formula.
// proof: ln(n!) = Σ_{x=1}^n ln(x)
≈ Integral_1^n ln(x) dx
= x ln x - x \bigbar_1^n
= n ln n - n - 1
≈ n ln n – n
≈ n ln n

log(128!) = ln(128!)/ln(128) ≈ (128*ln(128)-128)/ln(10)
= 214
]]

Substitution cipher

Given permutation e: $\Sigma \to \Sigma$ (the key), extend e pointwise to define
$e(x_1 \ldots x_m) = e(x_1) \ldots e(x_m)$
Let f = inverse of e.

Example: Σ = 26 characters {a,..., z}
27 characters {a, ..., z, BLANK}
95 characters of printable ASCII
Note: when Σ gets really large, like $\{0,1\}^{128}$, the ideal substitution cipher becomes a useful tool we will study extensively – a *blockcipher*.

Cryptanalysis of substitution cipher
(ciphertext-only attack, assuming message space is English, say)

Given a ciphertext $C = e(M)$, want to find M. In fact, find e (or, equivalent, f).

Write $f(c_1 \ldots c_m) = f(c_1) \ldots f(c_m)$

Practical problem: given knowledge of the plaintext space, as specified by a **corpus**; and given a ciphertext that we suspect "looks like" the corpus. Want: the plaintext.

Could try to use **single-letter** frequencies, a dictionary, and backtracking. Here is a somewhat more sophisticated approach, using a **hidden Markov model** and a **random walk**.

Diaconis algorithm

Based on the corpus text, compute
$M[x,y]$ = the probability of the two-letter sequence xy

where to get these values? Just grab text that you think looks like the target text, eg, they are of the same language.)

Proposed decryption algorithm: see next lecture.

Lecture 4 - ECS 127 - Winter 2019 - 1/13/2019

Today: o Symmetric encryption
- OTP & notions of security

Review: o Groups, Fields. $\mathbf{Z}_N$, $\mathbf{F}_N$. Average salary protocol. Let **James** finish it.

n parties, 1..n. Each party holds a secret value $0 <= a^i < \max$

Round 1: Party i generates uniform values in $\mathbf{Z}_M$ $a^i_1,\ldots,a^i_n$ that sum to a^i. Send a^i_j to party j.

Round 2: Sum the n values received mod M. Announce it.

Finish: Each party sum the n sums just announced mod M. They output 1/3 of this value (in the reals).

Let M = n max.

Encryption scheme syntax

The following **subject to change** (when we allow probabilistic, stateful, or nonce-based schemes)

Encryption scheme $\Pi = (\mathcal{K},\mathcal{E},\mathcal{D})$

$\mathcal{K}$ prob alg that takes no input and outputs a string in {0,1}*.

$\mathcal{E}$: $\mathcal{K} \times \mathcal{M} \to \mathcal{C}$ alg that takes input $K \in \mathcal{K}$ and $M \in \mathcal{M}$ and returns $C \in \mathcal{C}$. May be deterministic, probabilistic, or stateful. Show what that does to signature. **Encryption schemes that "shut up" – output $\perp$ if their unhappy with the number or length of inputs.**

$\mathcal{D}$: $\mathcal{K} \times \mathcal{C} \to \mathcal{M} \cup \{\perp\}$ det alg that takes $K \in \mathcal{K}$ and $C \in \mathcal{M}$ and returns $M \in \mathcal{M}$ or $\perp$.

Usually assume that $\mathcal{M}, \mathcal{C} \subseteq \{0,1\}^*$

Assume reasonableness of $\mathcal{M}$**:**

$M \in \mathcal{M} \Rightarrow \{0,1\}^{|M|} \subseteq \mathcal{M}$

Develop correctness for one-time and then probabilistic encryption, leaving stateful as an exercise.

Correctness:

$(\forall K)(\forall m)(\forall M_1, \ldots, M_m \in \mathcal{M})$

for i from 1 to m do if $C_i \xleftarrow{} \mathcal{E}_K(M_i)$ and $C_i \neq \perp$ then $M_i = \mathcal{D}_K(C_i)$

Finished is final, and only depends on lengths:

Once $\mathcal{E}$ outputs $\perp$, its output stays $\perp$.

The length of an output depends only on the length of an input.

The reasonableness condition ensures that the **message space** $\mathcal{M}$ of Π can be unambiguously defined: the set of all M such that $\mathcal{E}_K(M) \neq \perp$.

Require: $\mathcal{M} \neq \emptyset$.

OTP(k):

$\mathcal{K}$: output a uniformly random string in $\{0,1\}^k$

$\mathcal{E}_K(M)$: return $M \oplus K[1..|M|]$

$\mathcal{D}_K(C)$: if $|C|>k$ then return $\perp$ else return $C \oplus K[1..|C|]$

Three measures of security:

==========

(1) Perfect privacy (Shannon)

$\forall$ M_0, M_1 where $|M_0|=|M_1|$

$\forall$ C

$\Pr[\mathcal{E}(K,M_0)=C] = \Pr[\mathcal{E}(K,M_1)=C]$

(2) Shannon privacy: For all distributions $\mathcal{M}$

where $P(M)>0$ and $P(M')>0$ implies $|M|=|M'|$

$\Pr[M=M_0 \mid \mathcal{E}_K(M)=C] = \Pr[M=M_0]$

(Review the def of conditional probability.)

(3) "Modern" privacy, or, better, **real-or-zero privacy**

$\Pr[A^{\mathcal{E}(K,\cdot)} \to 1] = \Pr[A^{\mathcal{E}(K,0^{|\cdot|})} \to 1]$

where A asks only a single query to its oracle **

Said differently:

$\mathrm{Adv}_{\Pi}(A) = \Pr[A^{\mathcal{E}(K,\cdot)} \to 1] - \Pr[A^{\mathcal{E}(K,0^{|\cdot|})} \to 1] = 0$

Lecture 5 - ECS 127 - Winter 2019 - 1/15/2019

Today: o Symmetric encryption, cont.
- To more notions of security

First review syntax, $\Pi = (\mathcal{K},\mathcal{E},\mathcal{D})$ where

$\mathcal{K}$ prob alg that takes no input and outputs a string in {0,1}*.

$\mathcal{E}$: $\mathcal{K} \times \mathcal{M} \to \mathcal{C}$ det alg that takes input $K \in \mathcal{K}$ and $M \in \mathcal{M}$ and returns $C \in \mathcal{C}$.

$\mathcal{D}$: $\mathcal{K} \times \mathcal{C} \to \mathcal{M} \cup \{\perp\}$ det alg that takes $K \in \mathcal{K}$ and $C \in \mathcal{C}$ and returns $M \in \mathcal{M}$ or $\perp$.

Usually assume (1) **string-valued** M and C; (2) the reasonableness property $\mathcal{M}$: if you can encrypt some string of a given length, you can encrypt all strings of that length; and (3)one more property I think I forgot to mention, that the length of a ciphertext depends only on the

length of the corresponding plaintext: |C| = ε (|M|), normally |C|=|M|+τ. Then we defined **perfect privacy**.

(1) Perfect privacy (Shannon)

$(\forall\ M_0,\ M_1$ where $|M_0|=|M_1|)$
$(\forall\ C)$
$\Pr[\mathcal{E}(K,M_0)=C] = \Pr[\mathcal{E}(K,M_1)=C]$

"Ciphertexts are attributable no more to one plaintext than to any other plaintext of the same length"

Scheme: OTP(k):

$\mathcal{K}$: outputs a uniformly random string in $\{0,1\}^k$

$\mathcal{E}_K(M)$: return $M \oplus K[1..|M|)$

$\mathcal{D}_K(C)$: return $C \oplus K[1..|C|]$ (if we take $\mathcal{C} = \{0,1\}^k$)

(2) Shannon privacy: ($\forall$ distributions $\mathcal{M}$

where $P(M)>0$ and $P(M')>0$ implies $|M|=|M'|$)

$(\forall M_0 \in \mathcal{M})\ (\forall C \in \mathcal{C}$ s.t. $\Pr_{M\,K}[\mathcal{E}_K(M)=C]>0\ )$

$$\Pr_{M\,K}[M=M_0 \mid \mathcal{E}_K(M)=C] = \Pr[M=M_0]$$

(Review the def of conditional probability.) I don't actually like this definition, because assuming a distribution on $\mathcal{M}$ seems off. We don't now anything about it. You could say we're not assuming a distribution on it; we quantify over all distributions. But that seems off, too. It might not be a distribution: I could encrypt a random string, and then encrypt the ciphertext. That's not any (fixed) distribution. And we know some distributions aren't even samplable: why are we attending to them!

The truth is that both of these definitions are lousy, even if they're kind of classical. They don't generalize well.

(3) IND privacy (Real-or-Enc0)

$$\Pr[A^{\mathcal{E}(K,\cdot)} \to 1] = \Pr[A^{\mathcal{E}(K,0^{\bullet})} \to 1]$$

where A asks only a single query to its oracle **

Said differently:

$\mathrm{Adv}_{\Pi}(A) = \Pr[A^{\mathcal{E}(K,\cdot)} \to 1] - \Pr[A^{\mathcal{E}(K,0^{|\cdot|})} \to 1] = 0$

==========

Slowly explain intuition behind each definition, and the necessity of the "technical" conditions added.

All of these equivalent (once we add in all the red stuff). Omit proof, but argue that the OTP(k) satisfies perfect privacy and real-or-zero privacy. In particular, explain why for OTP[k], all M and C.

$\Pr[\mathcal{E}_K(M)=C] = 2^{-|M|}$ if $|C| = |M|$

$= 0$ o.w.

From a modern point of view, definitions (1) and (2) are not satisfactory, and the OTP is not satisfactory, because the one-time restriction is severe and, also, one could reasonably expect an encryption scheme to achieve *more* than pure privacy.

Lecture 6 - ECS 127 - Winter 2019 - 1/17/2019

Today: o Multiquery IND security
o Vernam ciphers and PRGs
o A concrete solution: RC4

Q3. Define **Perfect Privacy** for a symmetric encryption scheme. I will start you out.

Sym enc scheme Π=(K,E,D) achieves **perfect privacy** if

(for all such that)

(for all)

Pr [ ] = Pr [ ]

Encryption scheme $\Pi = (\mathcal{K},\mathcal{E},\mathcal{D})$

Perfect privacy

$(\forall\ M_0, M_1$ where $|M_0|=|M_1|)$

$(\forall\ C)$

$\Pr[\mathcal{E}(K,M_0)=C] = \Pr[\mathcal{E}(K,M_1)=C]$

"Ciphertexts are attributable no more to one plaintext than to any other plaintext of the same length"

IND privacy (Real-or-Enc0)

$\mathrm{Adv}_{\Pi}(A) = \Pr[A^{\mathcal{E}(K,\cdot)} \to 1] - \Pr[A^{\mathcal{E}(K,0^{|\cdot|})} \to 1] = 0$

"An oracle that encrypts what you ask is **indistinguishable** from an oracle that encrypts garbage"

Proposition An encryption scheme $\Pi = (\mathcal{K},\mathcal{E},\mathcal{D})$ achieves perfect privacy iff it achieves (perfect) 1-query IND privacy.

The real power is IND cleanly generalizes to allowing **multiple queries** and allowing **nonzero advantage**. Suppose we first change the syntax in of an encryption scheme to allow for state and/or probabilism. Then we have our first "satisfactory" definition for what an encryption scheme should **do**. A **good** encryption scheme is one for which "reasonable" adversaries achieves "small" advantage.

First two problems. (1) Only good for one message – solution is to move to IND security. (2) Keys are necessarily long:

Proposition An enc scheme Π that achieves perfect privacy must have at least as many keys as possible plaintexts (whence $|K| \geq |M|$).

Proof. Assume here the $\mathcal{K}$ is finite (o.w. we are done). Fix a ciphertext C in the ciphertext space. Let $\mathrm{Possible}_C = \{M \in \mathcal{M}: \exists K \in \mathcal{K} \text{ s.t. } \mathcal{E}_K(M)=C\}$. Because $|\mathrm{Possible}_C| < |\mathcal{M}|$, there is an $M_0 \in \mathcal{M}$ for which $M_0 \notin \mathrm{Possible}_C$. Because $\mathrm{Possible}_C \neq \varnothing$ $\exists\ M_1 \in \mathrm{Possible}_C$. The $\Pr[\mathcal{E}_K(M_0)=C]=0$ and $\Pr[EK(M_1)=C]>0$ because is $\mathcal{K}$ finite.

OTP*(k):

$\mathcal{K}$: output a uniformly random string in $\{0,1\}^k$

$\mathcal{E}_K(M)$: static $s \leftarrow 0$

if $s + |M| > k$ then return $\perp$
$C \leftarrow M \oplus K[s+1..s+|M|]$
$C \leftarrow (C,s)$
$s \leftarrow s + |M|$
return C

$\mathcal{D}_K(C)$: parse C into (C, s)

if $|C|+s>k$ then return $\perp$
return $C \oplus K[s+1..|C|]$

Stateful: No longer matches the syntax of an encryption scheme as we defined it: have to modify $\mathcal{E}$. In this case, the syntax is updated so that $\mathcal{E}$ is regarded as stateful.

Def. A (classical) PRG is a function g: $\{0,1\}^n \rightarrow \{0,1\}^N$ where N > n or. An (arbitrary-stretch) PRG is a procedure g: $\mathcal{M} \rightarrow \{0,1\}^\infty$ that, on input of a key K, returns an "infinite" string g(K).

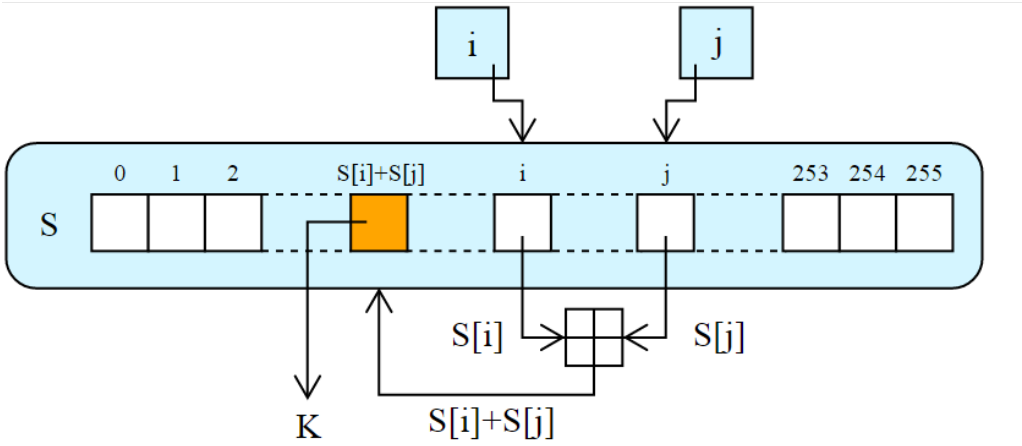
$\mathrm{Adv}^{\mathrm{prg}}(A)$

RC4: Rivest, 1987. Initially, proprietary to RSA Security. Attacks began in 2001, continue until just last year

Algorithm RC4(K) //PRG. K is a key of length 1 <= |K| <= 255

```
// Key scheduling.  K a byte string of 1..128 bytes (typically 5-16)
for i=0 to 255 do S[i] := i
j := 0
for i := 0 to 255    same as K||K||K||K …
    j := (j + S[i] + K[i mod |K|]) mod 256
    swap values of S[i] and S[j]

// Generate stream //
i := j := 0
do forever
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap values of S[i] and S[j]
    output S[(S[i] + S[j]) mod 256]
```

```
0 1 2 3 4 5 6 7
i
j
0 1 2 3 4 5 6 7
1 2 7 1 2 7 1 2
```



In use, we need something like the IV, and so, in applications like WEP, it follows the key. Gives rise to poor key agility.

Regardless not even close to being secure in the PRG sense (as above, but with no IV – only one output).

Lecture 7 - ECS 127 - Winter 2019 - 1/23/2019

Today: o Review of RC4; review of security notions: IND, PRG
o Playing with the definitions: two variants
o The idea of a reduction

IND: An encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$

$$\mathrm{Adv}_{\Pi}(A) = \Pr[A^{\mathcal{E}(K,\cdot)} \to 1] - \Pr[A^{\mathcal{E}(K,0^{\wedge}\cdot)} \to 1] = 0$$

"An oracle that encrypts what you ask is **indistinguishable** from an oracle that encrypts garbage"

A **PRG** is a fnct G: $\mathcal{K} \to \mathcal{L}$ where $\mathcal{K} = \{0,1\}^k$ and $\mathcal{L} = \{0,1\}^l$ for $l>k$ or $l=\infty$

$$\mathrm{Adv}_G(A) = \Pr[K \xleftarrow{R} \{0,1\}^k;\ Y \leftarrow G(K): A(Y) \Rightarrow 1] - \Pr[Y \xleftarrow{R} \mathcal{L}: A(Y) \Rightarrow 1]$$

Discuss what **good** and **bad** mean for IND or PRG security: bad schemes are those for which there exists a reasonable adversary that gains high advantage. Good schemes are those for which there does not exist a reasonable adversary that get high advantage. High advantage means "close pretty far from 0", like 0.5, 0.1, even 0.01. Small advantage means really close to 0, like 2^{-50}. Reasonable means that it doesn't consume *too* much data (eg, less than a few terabytes)and it doesn't take too much time (like, less than 2^{60} steps).

Variants: (1) Adding absolute values

$$\mathrm{Adv}^{1}_{G}(A) = \left| \Pr[K \xleftarrow{R} \{0,1\}^k;\ Y \leftarrow G(K): A(Y) \Rightarrow 1] - \Pr[Y \xleftarrow{R} \mathcal{L}: A(Y) \Rightarrow 1] \right|$$

The notion does change, but in a rather trivial way: in there exists an adversary that does well in one sense, there exists a comparably efficient adversary that does well in the other sense. It just negates what it is about to output.

(2) Guessing which world one is in:

$$\mathrm{Adv}^{2}_{G}(A) = 2\ \Pr[b \xleftarrow{R} \{0,1\};\ \text{if } b=1 \text{ then } K \xleftarrow{R} \{0,1\}^k;\ Y \leftarrow G(K) \text{ else } Y \xleftarrow{R} \mathcal{L}: A(Y) \Rightarrow b] - 1$$

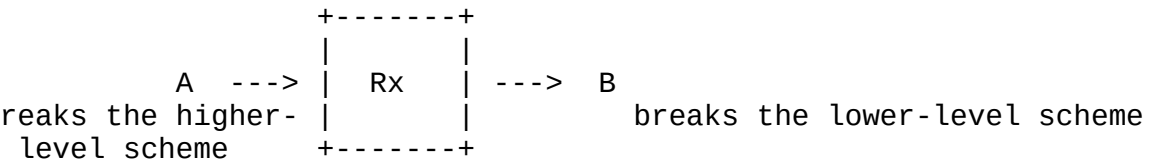
This one is exactly equivalent:

$$\begin{aligned}
2\ \Pr[A(Y) \Rightarrow b] - 1 &= 2\ (\Pr[A(Y) \Rightarrow 1 \mid b=1]\ \Pr[b=1] + \\
&\qquad \Pr[A(Y) \Rightarrow 1 \mid b=0]\ \Pr[b=0]) - 1 \\
&= \Pr[A(Y) \Rightarrow b \mid b=1] + \Pr[A(Y) \Rightarrow b \mid b=0] - 1 \\
&= \Pr[A(G(K)) \Rightarrow 1] + \Pr[A(\$) \Rightarrow 0] - 1 \\
&= \Pr[A(G(K)) \Rightarrow 1] + (1 - \Pr[A(\$) \Rightarrow 1]) - 1 \\
&= \Pr[A(G(K)) \Rightarrow 1] - \Pr[A(\$) \Rightarrow 1] \\
&= \mathrm{Adv}_{G}(A)
\end{aligned}$$

We showed had to make a (stateful) encryption scheme out of a PRG. It can encrypt up to l bits. The Vernam construction, Vernam[G]. Claim: Vernam[G] is IND-secure if the PRG we start from is PRG-secure.

G is a good PRG ⇒ Vernam[G] is a good enc scheme (in the IND sense)
G is a bad PRG ⇐ Vernam[G] is a bad enc scheme (in the IND sense)

∃ B breaks G in the PRG sense ⇐ ∃ A breaks Vernam[G] in the IND sense



Given A, how will we construct B:
B is given a long or infinite string Y
It runs A, which expects to have an oracle that encrypts strings
When A asks to encrypt a string X, B emulates what Vernam(G)

would do with *Y* being the output of the PRG.
When A halts, outputting a bit b, B outputs the exact same bit.

Claim: $Adv^{prg}_{G}(B) = Adv^{ind}_{Vernam(G)}(A)$

$$\begin{aligned} Adv^{prg}_{G}(B) &= Pr[B(G(K)) \Rightarrow 1] - Pr[B(\$) \Rightarrow 1] \\ &= Pr[A^{Enc} \Rightarrow 1] - Pr[A^{Enc(\$)} \Rightarrow 1] \\ &= Adv^{ind}_{Vernam(G)}(A) \end{aligned}$$

Lecture 8 - ECS 127 - Winter 2019 - 1/25/2019

Today: o Review of PRG, reduction idea
o Difficulties with RC4
o More constructions: chacha20, DES, AES
Announcements:
O Dog day: Fri, Feb 8

A **PRG** is a fnct $G: \mathcal{K} \to \mathcal{L}$ where $\mathcal{K} = \{0,1\}^k$ and $\mathcal{L} = \{0,1\}^l$ for $l>k$ or $l=\infty$

$$Adv_G(A) = Pr[K \xleftarrow{R} \{0,1\}^k;\ Y \leftarrow G(K): A(Y) \Rightarrow 1] - Pr[Y \xleftarrow{R} \mathcal{L}: A(Y) \Rightarrow 1]$$

Beautiful idea of **indistinguishability** as answer to what is random. Philosophical answer. Says nothing about a *particular* string being random. Only addresses a collection of strings being pseudorandom. There are other answers, particular **Kolmogorov complexity.**

Difficulties with RC4

Designed 1987 Ron Rivest. Proprietary. In BSAFE toolkit.
Reverse-engineered in 1994.
Used in SSL 1995, WEP in 1997, TLS 1999, WPA 2004
Bias seen back in 1995!
Devastating attack in 2001, as used in WEP, by Fluhrer, Mantin, Shamir
Attacks have continued: Kenny Patterson, 2015, I believe is the latest.

Was never the right signature!!
Highly inconvenient to use a PRG $G: \mathcal{K} \to \mathcal{L}$ for a Vernam cipher because lost messages will mean lots of work, can't "jump around

More convenient: PRF F: $\mathcal{K} \times \mathcal{N} \rightarrow \{0,1\}^{\infty}$

or PRF F: $\mathcal{K} \times \mathcal{N} \times \mathcal{N}' \rightarrow \{0,1\}^{l}$

Key, nonce, counter

ChaCha20: $\{0,1\}^{256} \times \{0,1\}^{128} \rightarrow \{0,1\}^{512}$
32-bit counter
96-bit nonce

Google has been moving to Chacha20, a cipher invented by Dan Bernstein (2008) derivative of an earlier suggestion of his, Salsa20. (This follows a tradition of using silly names for ciphers.)

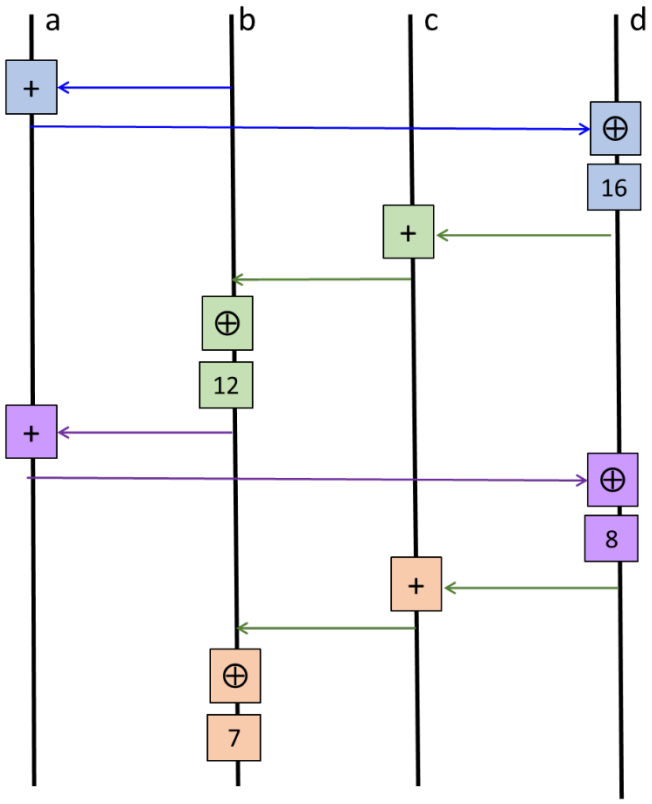
Chacha20 has 256 bit keys, a 96 bit IV, and a 32-bit block index (to indicate a particular block of output).

```
 0   1   2   3
 4   5   6   7
 8   9  10  11
12  13  14  15
```

Sixteen 32-byte words

One quarter round:

```
Algorithm QR(a,b,c,d)
a += b; d ^= a; d <<<= 16;
c += d; b ^= c; b <<<= 12;
a += b; d ^= a; d <<<= 8;
c += d; b ^= c; b <<<= 7;
```



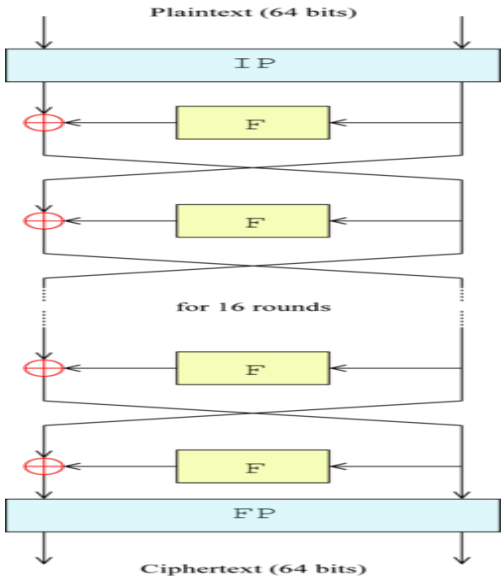

```
Algorithm QR10(s)
   QR(s, 0, 4, 8,12)  // col 1
   QR(s, 1, 5, 9,13)  // col 2
   QR(s, 2, 6,10,14)  // col 3
   QR(s, 3, 7,11,15)  // col 4
   QR(s, 0, 5,10,15)  // diag 1
   QR(s, 1, 6,11,12)  // diag 2
   QR(s, 2, 7, 8,13)  // diag 3
   QR(s, 3, 4, 9,14)  // diag 4

Algorithm chacha20(key, counter, nonce): //8, 1, 3 words
   state = constant | key | counter | nonce // 4, 8, 1, 3 words
   s = state
   for i=1 to 10 do QR10(s)
   state += s
   return serialize(state)  // bytes of each word in order 4321 8765 …
```

Lecture 9 - ECS 127 - Winter 2019 - 1/28/2019

Today: o PRFs and PRPs; DES and AES
Announcements:
o Dog day! Fri, Feb 8

DES

Developed in the 1970's by IBM+NSA. Derived from Lucifer. 10 people. HW centric. Standardized by NBS (FIPS 46) (1977). 64-bit block, 56-bit key, Feistel network. Horst Feistel, Walter Tuchman, Don Coppersmith, Alan Konheim, Carl Meyer, Mike Matyas, Roy Adler, Edna Grossman, Bill Notz, Lynn Smith, and Bryant Tuckerman.



Explain how Feistel networks always induce a permutation. Can be thought of as a technique to turn a PRF into a PRP.

Exhuastive key search. In 1977, Diffie and Hellman proposed a machine costing an estimated $20 million which could find a DES key in a single day. 1998: EFF machine for $250K in abou 3.5 hrs, 1856 custom FPGSas. Now: FPGA-based machine for about 10K, few days. Fixing this problem: **Triple DES**. Show.

Differential cryptanlaysis: Biham-Shamir (1992). 2^{49} chosen plaintext/ciphertext paris. From Don Coppersmith's paper "The Data Encryption Standard (DES) and its strength against attacks" (1994):

As one can imagine, if he [the cryptanlalyst] starts with a known plaintext *m* and unknown key k and tries to trace the encipherment through 16 rounds of DES encryption, he soon becomes hopelessly entangled, because bits of the unknown key *k* are XORed with the message at the input of every S-box. In differential cryptanalysis, however, he starts with two messages, *m* and *m*', differing by a known difference Δ_m.

He considers the difference between the intermediate message halves: Δ_m, = $m_i \oplus m'_i$. The input to S-box S_1, for example, at round *i* of the encipherment of message *m* is m_i[32,1,2,3,4,5] $\oplus$ k_i [1,2,3,4,5,6], and the input to S_1 at round *i* of the encipherment of message *m* ' is m_i' [32,1,2,3,4,5] $\oplus$ k_i' [1,2,3,4,5,6]. [So the xor of the inputs to the S-box is] m_i[32,1,2,3,4,5] $\oplus$ m_i' [32,1,2,3,4,5]. The dependence on *k* has disappeared.

Linear cryptanalysis: Matsui (1993): 2^{43} known plaintext/ciphertext pairs.

Lecture 10 - ECS 127 - Winter 2019 - 1/30/2019

Today: o AES
o PRPs vs. PRFs
o Use of PRFs
Announcements:
o Dog day! Fri, Feb 8

Review definitions and construction
PRG, PRF, PRP; RC4, ChaCha20, DES

"Politics through mathematics" – DES. Contrast with bridges
Langdon Winner (1980) – 3558 citations. Robert Moses.
Bernward Joerges challenged narrative (1999)

AES

History. FIPS 197. 2002, Joan Daemen and Vincent Rijmen (Belgium)
-128/192/256, 128, SW/HW, 10-rounds, roughly 20 cpb
-Rijndael: other finalists: Mars, RC6, Serpent, Twofish

Background on finite fields: First review finite fields: a set ***F*** with operations $+, \cdot$ satisfying the usual properties: ***F*** under addition is a group (the identity denotes 0); F\{0} under multiplication is a group

(its identity denoted 1); and multiplication distributes over addition. Finite fields: the underlying set is finite.

A well-known theorem of algebra says that there are finite fields of size p^α for any prime p and number $\alpha \geq 1$; that there are no other finite fields; and that each of these fields is unique, up to the naming of elements. So when I describe them, you know all the finite fields. The field with N elements is denoted $\boldsymbol{F}_N$ or GF(p^α). Eg, GF(2^{128}).

Construction of the finite field GF(2^8). Addition = xor. Multiplication: Fix an irreducible polynomial m of degree 8. Regard bytes as specifying polynomials of degree at most 7. To

$m(x) = x^8 + x^4 + x^3 + x + 1$

exercise: a4 * 00 = **00**
a4 * 01 = **a4**
a4 * 02 = **53**

1010 0100 = $x^7 + x^5 + x^2$
0000 0010 = x

product: $x^8 + x^6 + x^3 = x^4 + x^3 + x + 1 + x^6 + x^3$
$= x^6 + x^4 + x + 1$
= 0101 0011
= **53**

Description of AES.

We will now manipulate this state repeatedly. It's final value be the ciphertext Y = AESK(X).

1. **Key expansion**
   Stretch the 128-bit key K into 11 strings K_0 ... K_{10} of 128 bits each. Details omitted here.

2. Initialize State ← X, the 128-bit word we wish to encipher.

   Sometimes we think of State as a 4 x 4 array of bytes:

   State = State[0] State[1] ... State[15] are arranged as

	col 0	col 1	col 2	col 3
row 0 ->	State[0]	State[4]	State[8]	State[12]

```
row 1 ->   State[1]  State[5]  State[9]  State[13]
row 2 ->   State[2]  State[6]  State[10] State[14]
row 3 ->   State[3]  State[7]  State[11] State[15]
```

3. State ← State ⊕ K_0

4. for i ← 1 to 10 do

4.1 **SubBytes**: Replace each byte State(i) by S[State(i)] where S: $\{0,1\}^8 \rightarrow \{0,1\}^8$ is a particular (fixed) permutation. (The permutation selected happens to be the affine translate of the inverse of the point, treating the point as an element of GF(2^8).)

affine translate:

```
1 0 0 0 1 1 1 1
1 1 0 0 0 1 1 1
1 1 1 0 0 0 1 1
1 1 1 1 0 0 0 1
1 1 1 1 1 0 0 0
0 1 1 1 1 1 0 0
0 0 1 1 1 1 1 0
0 0 0 1 1 1 1 1
```

4.2 **ShiftRows**: Left circularly shift row 0 by 0; row 1 by 1; row 2 by 2; row 3 by 3.

4.3 **MixColumns**: if i≠10 then treat EACH column a_0 a_1 a_2 a_3 replace this by the column obtained by the GF(2^8) multiplication by the matrix:

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

This matrix is invertible (its determinant in the finite field is nonzero); it's inverse is

```
0E 0B 0D 09
09 0E 0B 0D
```

0D 09 0E 0B
0B 0D 09 0E

4.4 **AddRoundKeys**: State = State $\oplus K_i$

5. return State

What does it DO?? Ie., what security notion do we aim for?

Lecture 11 - ECS 127 - Winter 2019 - 1/30/2019

Today: o PRPs vs. PRFs
o Use of PRFs

Lets start off with the classical **birthday bound**.
Let C(q,N) = Probability that two balls land in the same bin in the experiment of throwing q balls, uniformly and independently, into N bins.

Prop. $C(q,N) \le q(q\text{-}1)/2N \le q^2/N$

Let C_i be the event that the i^{th} ball collides with one of the previous ones. The $\Pr[C_i] \le (i\text{-}1)/N$. So

$C(q,N) = \Pr[C_1 \text{ or } C_2 \text{ or } \dots \text{ or } C_q]$
$\le \Pr[C_1] + \Pr[C_2] + \dots + \Pr]C_q]$
$= 0/N + 1/N + \dots + (q\text{-}1)/N$
$= (1 + 2 \dots + q\text{-}1)/N$
$= q(q\text{-}1)/2N$

Paying attention to the constant,
$C(q,N) \approx 0.5$ when $q = \sqrt{2\ln(2)}\sqrt{N} \approx 1.1774\sqrt{N}$

Eg: N = 365 this formula gives 22.5, say 23, which is the right answer.

Now develop
PRP/PRF switching lemma…

$\mathrm{Adv}^{\mathrm{prp}}_E(A) = \Pr[K \xleftarrow{\$} \mathcal{K}: A^{E(K,\cdot)} \to 1] - \Pr[\pi \xleftarrow{\$} \mathrm{Perm}(n): A^{\pi(\cdot)} \to 1]$

$\mathrm{Adv}^{\mathrm{prf}}_E(A) = \Pr[K \xleftarrow{\$} \mathcal{K}: A^{E(K,\cdot)} \to 1] - \Pr[\rho \xleftarrow{\$} \mathrm{Func}(n): A^{\rho(\cdot)} \to 1]$

Explain initiation.

Prop: For any adversary A that makes at most q queries to a PRP $E: \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$, we have that $|\mathrm{Adv}^{\mathrm{prp}}_E(A) - \mathrm{Adv}^{\mathrm{prf}}_E(A)| \le q^2/2^{n+1}$

$|\Pr[A^{E(K,\cdot)} \to 1] - \Pr[A^{\pi(\cdot)} \to 1] - (\Pr[A^{E(K,\cdot)} \to 1] - \Pr[A^{\pi(\cdot)}]) \to 1]|$
$= |\Pr[A^{\pi(\cdot)} \to 1] - \Pr[A^{\rho(\cdot)}]) \to 1]|$

Game-playing argument

Game PERM or GAME RAND
Initialize f as the partial function from $\{0,1\}^n$ to $\{0,1\}^n$ that is everywhere undefined.

Oracle $E(X)$
  if $X \in \mathrm{Dom}(f)$ then return $f(X)$
  $Y \xleftarrow{\$} \{0,1\}^n$
  if $Y \in \mathrm{Ran}(f)$ then bad $\leftarrow$ true, $Y \xleftarrow{\$} \{0,1\}^n \setminus \mathrm{Ran}(f)$
  return Y

Fundamental lemma of game playing (Bellare-Rogaway): If games G1 and G0 are identical-until-*bad*, then $\Pr[A^{G1} \to 1] - \Pr[A^{G0} \to 1] = \Pr[\text{G0 sets } bad]$.

Lecture 12 - ECS 127 - Winter 2019 – 2/04/2019

Today: o PRPs vs. PRFs
       o Use of PRFs
       o WHY the PRP/PRF notions?

Review:
Not the only way to define OTP, and not the only way to define an encryption scheme, either. One approach is to be stateful. Here is a stateful version of our OTP, OTP*[k]:

CTR[*E*] where $E:\{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$

$\mathcal{K}$: output a uniformly random string in $\{0,1\}^k$

$\mathcal{E}_K(M)$: static $S \leftarrow 0$

$C \leftarrow M \oplus F_K(S)\ F_K(S+1)\ F_K(S+2)\ \ldots$
$\boldsymbol{C} \leftarrow S \,||\, C$
$S \leftarrow S + \lceil |M|/n \rceil$
return C

$\mathcal{D}_K(C)$: $S \,||\, C \leftarrow \boldsymbol{C}$

return $C \oplus F_K(S)\ F_K(S+1)\ F_K(S+2)\ \ldots$

Stateful scheme. Achieves perfect privacy and 0-advantage IND as long as total number of block queried is less than 2^n.

Informal Theorem. If E is secure as a PRP than CTR[E] is IND-secure.

Draw a picture. Replace E_K by pi. Then replace pi by rho. Then explain that there is zero advantage in this setting. (Stateful Scheme without wraparound)

Theorem. (say it after you have proven it)
Let A be an adversary attacking CTR[*E*] and achieving ind-advantage ϵ. Suppose A asks a total of σ blocks worth of plaintext Then there exists an adversary B, easily constructed from A, for distinguishing E from a random permutation. It achieves advantage $\epsilon/2 - \sigma^2/2^{n+1}$ and asks σ queries and has running time about that of A.

Proof. Think of the scheme as given by a random function instead of being given by a blockcipher.

CTR[E].Enc(.)

CTR[P].Enc(.)

A CTR[R].Enc(.)

CTR[P].Enc($0^{|\cdot|}$)

CTR[E].Enc($0^{|\cdot|}$)

One of these differences is large.

Randomized scheme

Lecture 13 - ECS 127 - Winter 2019 – 2/06/2019

Today: o Finishing CTR mode
o More encryption modes: CTR$, ECB, CBC and their security
o Why PRPs?

Reminder:
- Dog Day on Friday!
- Questions on how much people understand. LOTS of ½ or less. Lots of requests for more examples; some for easier HWs; write bigger. Want a book (that ship has sailed).

Go over slides.

Adv^ind(A) = (a-b) + (b-c) + (c-d) + (d-e) + (e-f)
<= Advprp(B) + s^2/2^{n+1} + 0 + s^2/2^n+1 + Advprp(C)
Advprp(B) + Advprp(C) >= Adv^ind(A) + s^2/2^n
Adv^(B) >= ½ Adv^ind(A) + s^2/2^{n+1}

More schemes.

CTR$[*E*] where $E:\{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$

$\mathcal{K}$: output a uniformly random string in $\{0,1\}^k$

$\mathcal{E}_K(M)$: $S \leftarrow \{0,1\}^n$

$C \leftarrow M \oplus F_K(S)\ F_K(S+1)\ F_K(S+2)\ \ldots$
$\boldsymbol{C} \leftarrow S \,||\, C$
return C

$\mathcal{D}_K(C)$: $S \,||\, C \leftarrow \boldsymbol{C}$

return $C \oplus F_K(S)\ F_K(S+1)\ F_K(S+2)\ \ldots$

Probabilistic scheme: No longer matches the syntax of an encryption scheme as we defined it: have to modify $\mathcal{E}$. In this case, the syntax is updated so that $\mathcal{E}$ is regarded as stateful.

Lets switch to IND$ first ..

Theorem. Let A be an adversary attacking CTR[E] and achieving ind-advantage ϵ and asking a total of $\sigma = [\ |M_i|/n\ ]$. Then there exists an adversary B for distinguishing E from a random function that achieves advantage $\epsilon - \sigma^2/2^n$. Adversary B is about as efficient as A.

Lecture 14 - ECS 127 - Winter 2019 – 2/08/2019

Today: Dog Day! Gabriella : Tsuki

- o Comparing security notions
  (kr-security of a blockcipher, ind$ security of an encryption scheme, maybe CCA security and nonmalleability)

Announcements:

- o Midterm one week from today

Security notions so far:

- prg -dvantage of a PRG G
- prf-advantage of a PRF or blockcipher E
- prp-advantage of a blockcipher E
- ind-advantage of a prob or stateful enc scheme P=(K,E,D)

Why **these** notions?
Lots of alternatives.
Won't let you prove something like CTR's security.

Key recovery:

$\mathrm{Adv}_E^{\mathrm{kr}}(A) = \Pr[K \xleftarrow{\$} \mathcal{K}: A^{E(K,\cdot)} \rightarrow K\ ]$

The inadequacy of KR-security

1) Not strong enough: E_K(X)=X example. Key cant recover but clearly bad.
2) Not useful. How are you going to create something like an IND-secure encryption scheme from this property? (In some abstract sense it *is* possible, but it certainly will not be practical.)

IND-secure → KR secure
IND-insecure ← KR insecure
Exists adv B breaking IND-seucrity ← Exists adv A breaking KR security

Def of B^f:
   Run A, answering oracle queries with B's oracle f.
   When A finishes, outputting a key K,
      Choose a not-yet-queries point X and ask f(x).
      If the answer Y is E_K(X), return 1, else return 0

Suppose A asks q queries. Algorithm assumes q < 2^n.

$$\begin{aligned} \mathrm{Adv}^{\mathrm{prp}}(B) &= \Pr[B^{E_K} \to 1] - \Pr[B^{\pi} \to 1] \\ &\le 1 - (1 - 1/(2^n - q)) \\ &\le 1/(2^n - q) \\ &\le 2/2^n \quad \text{if } q < 2^{n-1} \end{aligned}$$

More notions for blockcipher-security

Unpredictability:

$\mathrm{Adv}^{\mathrm{unp}}_E(A) = \Pr[K \xleftarrow{R} \mathcal{K};\ (X,Y) \leftarrow A^{E(K,\cdot)}: E(K,X)=Y \text{ and } A \text{ made no query } X]$

PRP security:

$\mathrm{Adv}^{\mathrm{prp}}_E(A) = \Pr[K \xleftarrow{R} \mathcal{K}: A^{E(K,\cdot)} \to 1] - \Pr[\pi \xleftarrow{R} \mathrm{Perm}(n): A^{\pi(\cdot)} \to 1]$

Strong PRP security:

$\mathrm{Adv}^{\pm\mathrm{prp}}_E(A) = \Pr[K \xleftarrow{R} \mathcal{K}: A^{E(K,\cdot),\, E\mathrm{inv}(K,\cdot)} \to 1]$

$\quad - \Pr[\pi \xleftarrow{R} \mathrm{Perm}(n): A^{\pi(\cdot)\pi\mathrm{inv}(\cdot)} \to 1]$

IND$ notion of an encryption scheme.
How does IT compare?

1) If we keep our current syntax. Ind$-security implies ind-security, but not the reverse. Prove.

First party. **IND$-secure → IND-secure**

IND$-security means that an
$\mathcal{E}_K(.)$ oracle can't be distinguished from a $ oracle.

Which means that an $\mathcal{E}_K(.)$ restricted to 0* queries cant be distinguished from a $ oracle.

Indistinguishability is transitive (with a reduction in bound for the number of steps), so what we're saying is that an $\mathcal{E}_K(.)$oracle can't be distinguished from a $\mathcal{E}_K(0^{|.|})$ oracle.

Other direction. **IND-security does not imply IND$ security.**
(1) CTR[E] is IND-secure (for E a PRP) but not IND$-secure
(2) General approach, minimal assumptions: Want to say here exists a scheme Pi that is IND-secure but not IND$ secure.But cant exactly say that, because we don't know that there exists a scheme Pi that is IND-secure. Next best thing: what we show is that if we start with a scheme Pi that is IND-secure than we can modify it to a scheme Pi' that remains IND-secure but is now guaranteed to NOT be IND$-secure.

Pi=(K,E,D).
Pi=(K,E',D'): E'(K,M): C \getsr E_K(M); return 0^100 C
D'(K,C): C \gets M[101..]; return D_K(C)

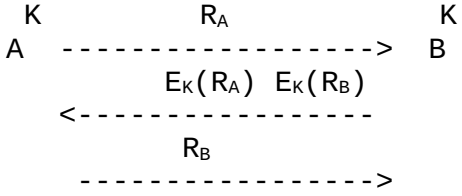
Lecture 15 - ECS 127 - Winter 2019 – 2/11/2019

Today: o Beyond IND$ security
- Stronger aims: nonmalleability, CCA-security, AE
- Nonce-based encryption and associated data
o Back to message authentication

Announcements:
- Colored chalk!
- MT on Friday.
- Offered to distribute notes, but nobody sent me any
- Added Monday Office Hours, 1-2.This week only: Wed Office hours, 1-2.

Two big problems with IND/IND$-encryption
1) Usability problem of using probabilistic encryption. Traditional schemes IV-based
2) The notions of security aren't actually that strong. It is misleading that the OTP is called "perfect" and that IND/IND$ is presented at something strong. No CCA security, no nonmalleability, no authenticity:

CCA security: An encryption scheme should remain secure even if an adversary gets access to a **decryption** capability. CCA stands for *chosen-ciphertext attack*. What does this really model? Traditional answer: lunchtime attack. Better answer: the ability in protocols built from encryption schemes to influence ciphertexts and get back resonses from them. Example



Def: $\mathbf{Adv}^{\mathrm{cca}}_{\Pi}(A) = \Pr[A^{\mathcal{E}(K,.),\ \mathcal{D}(K,.)} \rightarrow 1] - \Pr[A^{\$,\ \mathcal{D}(K,.)} \rightarrow 1]$

Nonmalleability: An encryption scheme should not allow an adversary to modify ("maul") a ciphertext into another ciphertext whose underlying plaintext is related to the plaintext of the original ciphertext. We briefly discussed this in the context of a OTP, which is highly malleable. Ex based on ASCII(0)=30, ASCII(1)=0x31, ASCII(9)=39

Authenticity: A symmetric encryption scheme should guarantee to the recipient of a message that the message he is recovering was actually sent by the party with whom he shares his key. (The only parties that can make a valid ciphertext are the parties that have the underlying key.)

Def: $\mathbf{Adv}^{\mathrm{ae}}_{\Pi}(A) = \Pr[A^{\mathcal{E}(K,.),\ \mathcal{D}(K,.)} \rightarrow 1] - \Pr[A^{\$,\perp} \rightarrow 1]$

Let's focus on the last one, for adding it, as an adjunct to ind-cpa security, is going to be enough to guarantee the other two properties.

Lets try to see something that doesn't work: adding redundancy to a standard mode like CTR or CBC.

Draw pictures – give attacks – on CTR/CBC with redundancy at end.

Another direction: **Nonce-based** instead of probabilism or state

1) More in line with IVs, traditional practice.
2) Less likely to be misused – effectively lowers requirement on randomness
3) Software doesn't have to reach into some library we don't control to get randomness.
4) Easier testing

Also: **associated data**. Give example of utility from networking context.

Experience indicates that people don't get the IV right: one needs more than their being nonce, and random bits are sometimes not even available. Important, for practice, for schemes to work with *just* a nonce.

Syntax: A **nonce-based** symmetric encryption scheme would still a three-tuple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, but now the encryption map $\mathcal{E}$ now takes in K, N, M

and, similarly $\mathcal{D}$ takes in K, N, C. Adjust other aspects in the natural way, eg, the correctness condition now applying to all N.

Security notion: ind$ would be adapted in the natural way. The adversary is given either a **real** encryption oracle or a **random-bits** oracle. The former, on input (N,M), returns $\mathcal{E}(K,N,M)$ for a K chosen at the beginning of the game. The random-bits oracle, on input N,M, returns the appropriate number of random bits. (In most cases, "appropriate" is no $|M|$.)

CBC with a nonce-IV: doesn't work, break it, adding column above.
CTR with nonce IV. Describe it, encrypting N to make the IV.

Lecture 16 - ECS 127 - Winter 2019 – 2/13/2019

Today: o message authentication
Announcements:
 -MT on Friday
 - "The Moral Character of Cryptographic Work" 3:10 pm, 1131 Kemper

Quiz

Message authentication codes (MACs)

Go back to 2x2 grid.

Review trust model and informal goal, then syntax, treating **deterministic, stateless MACs.**

Usual way to authenticate a message: accompany it with a short tag, called the *MAC*. Traditionally, just 32 bits; nowadays, usually 96-128 bits. Syntax:

MAC: $\mathcal{K} \times \mathcal{M} \to \{0,1\}^t$

for a deterministic MAC. (Stateful or probabilistic MACs are possible too, and sometimes used, e.g., for reasons of improving the security bound.)

Explain usage: sender accompanies a message M with a tag T = MAC(K,M). Receiver gets a pair (M,T) and compute if T == MAC(K,M), rejecting M otherwise.

Security definition

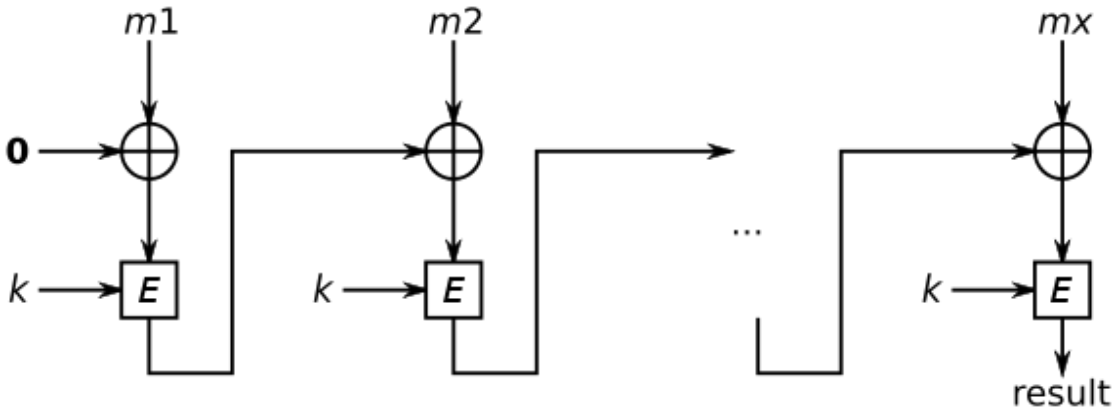
Unforgeability under an ACMA. Let F: $\mathcal{K} \times \mathcal{M} \to \{0,1\}^t$ be a MAC and A an adversary. Define

$$\mathbf{Adv}_{\Pi}^{\text{mac}}(A) = \Pr[A^{F(K,\cdot)} \text{ forges}]$$

where A forges if it outputs an (M,T) such that $T = F(K,M)$ even though A never asked a query M returning T. [last two words can be omitted for deterministic MACs).

Constructing MACs

Traditional approach: the CBC MAC:



Algorithm CBC-MAC(K,M) //raw CBC MAC: no padding, $|M|$ a multiple of n
if $|M|$ is not a positive multiple of n then return ERROR
$M_1 \ldots M_m \leftarrow M$ where $|M_i|=n$
$Y \leftarrow 0^n$
for $i \leftarrow 1$ **to** m **do** $Y \leftarrow Y \oplus E(K,M_i)$
return Y

Show that the (raw) CBC-MAC is **insecure** across messages of varying lengths: from the MAC of a one-block string you can forge the MAC of a two-block string. Indeed you can forge infinitely many MACs. And it's pretty bad: given the MACs for a few modest-length messages, you can forge a very rich set of messages.

Carter-Wegman MACs. Review definition of MAC security (unforgeability under an ACMA), the CBC MAC, and why the CBC MAC is not secure over variable-length messages. But it **is** secure (in the unforgeability sense) on fixed length messages; it is even a **PRF**. (Emphasize that the PRF notion doesn't apply to blockciphers; and, also, that a good

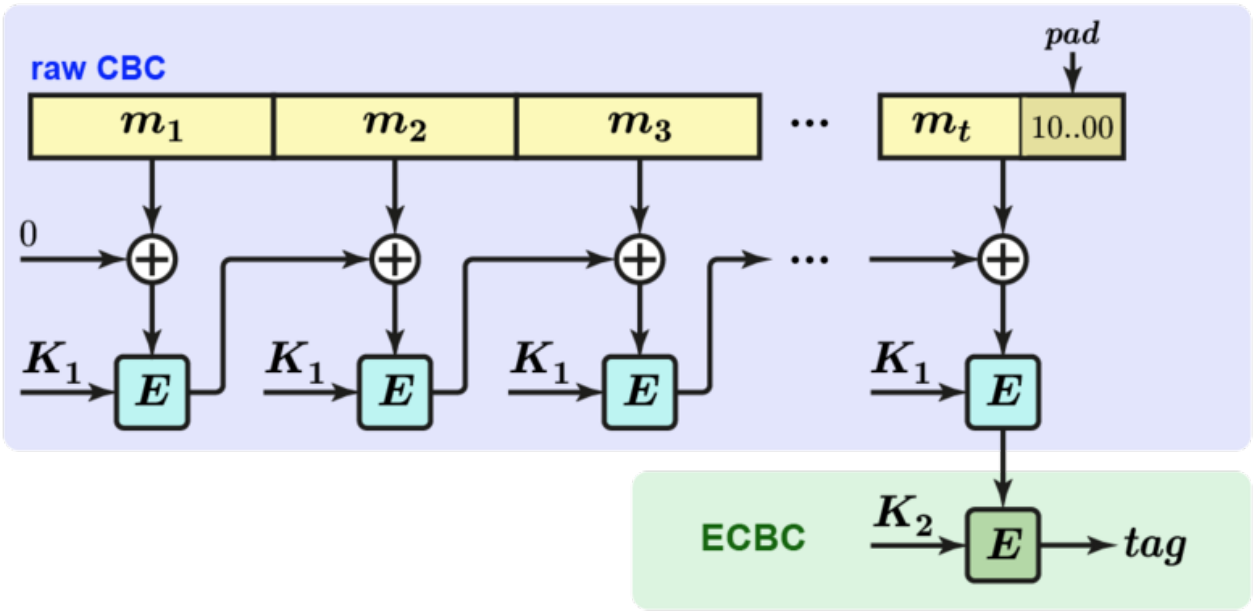
PRF is always a good MAC. Perhaps make a **HW problem** out of the last claim.) Also, on variable length messages, the CBC MAC is ε-**almost universal** (ε-AU) for small ε. Define this concept:

Definition: a function H: $\mathcal{K} \times \mathcal{M} \to \{0,1\}^\tau$ is **ε-AU** if for all distinct $M, M' \in \mathcal{M}$, $\Pr[K \xleftarrow{} \mathcal{K}: H_K(M) = H_K(M')] \leq \varepsilon$.

Explain the viewpoint of H comprising a *family* of hash functions, each one named by a key K. We are asking that, for all distinct messages in the message space, the probability that they collide under a randomly chosen hash function the family is small.

Claim [Black, Rogaway 2000]: The CBC MAC is ε-AU for a small ε. Specifically, if M and M' are distinct messages of at most m blocks (each block having n bits), then $\Pr[\text{CBCMAC}_\pi[M] = \text{CBCMAC}_\pi[M']] \leq m^2/2^n$ (where π is randomly chosen from Perm(n)).

Now consider the following construction:



We can think of the Raw CBCMAC on top as a hash function, the result being enciphered (with an independent key) to produce the tag (which can, if desired, be truncated to give a shorter tag).

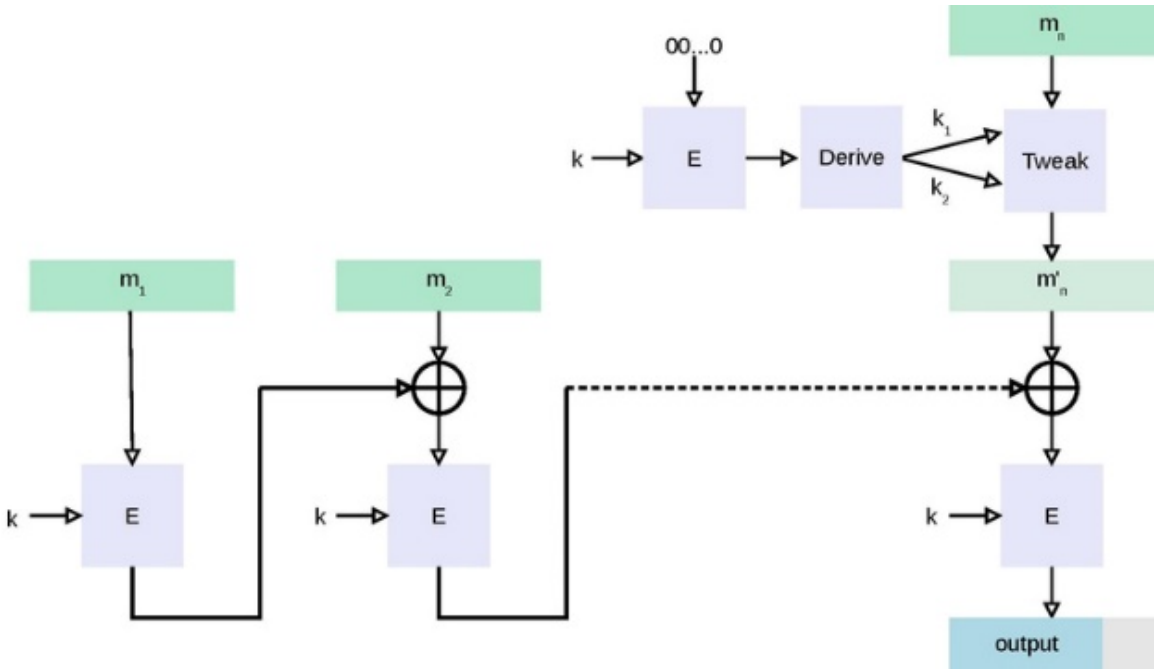
The above is an instance of the **Carter-Wegman paradigm**. That approach is to make a MAC –or, in fact a PRF– by combining an e-AU hash function and (say) a PRP, as by:

$F_{K\,K'}(M) = E_{K'}(H_K(M))$

I claim that this always works:

Informal proposition (rooted in Carter-Wegman … long history after that): If H is ε-AU and E is a secure PRP, then $F_{K\,K'}(M) = E_{K'}(H_K(M))$ is a secure PRF. (How good it is depends on ε and the quality of the PRP, of course.)

Describe progression to CMAC: 1) First, combine the two blockcipher invocations at the end into a single blockcipher call, since the composition of random permutation is a random permutation. 2) Now switch from keying with a separate key to pre-whitening with an n-bit key. 3) Extend to deal with arbitrary-length input by 10* padding if needed, and using two different keys for pre-whitening. 4) Finally, save on underlying key material by computing pre-whitening keys as in $2E(K,0)$ and $4E(K,0)$. The result is **CMAC**, a NIST standard that one might think of as the "modern" reinvention of the CBC MAC, now that we have a rigorous definition in cryptography. Here's a picture



Lecture 17 - ECS 127 - Winter 2019 – 2/15/2019

Midterm

Lecture 18 - ECS 127 - Winter 2019 – 2/20/2019

Today: o Message authentication via AU-hash functions
O Back to AE

Announcement: Midterms graded, up online

Now there are more satisfying ways to make an ε-AU hash function. The most widely used is polynomial arithmetic over a finite field, say GF(2^{128}). This is the approach used in **GMAC** (part of the **GCM** authenticated encryption scheme). Let's describe a simplified and cleaned up version of GMAC. Assume we want to hash a message $M = M_1 \ldots M_m$, each block an n-bit key, with, say, n=128. Regard M as specifying a polynomial over GF(2^{128}):

$$M(x) = x^m + M_1 x^{m-1} + \ldots + M_{m-1}x + M_m$$

The family of hash function is keyed by 128-bit strings. Each key K is used so as to hash a message M to the point $M(K)$. Weird? We are using the message to name the coefficients of a polynomial, and then evaluating that polynomial at K.

I claim this approach gives an ε-AU hash family with good ε. Suppose that M and M' are distinct messages, each having at most m n-bit blocks. We need to upper bound

$$\Pr[H_K(M) = H_K(M)] = \Pr[M(K)=M'(K)] = \Pr[g(K)=0]$$

where g is a nonzero polynomial of degree at most m. The **fundamental theorem of algebra** (at least one form of it!) says that, over any finite field **F**, a nonzero polynomial of degree m has at most m zeros (in the field). Thus

$$\Pr[g(K)=0] \le m/|\mathbf{F}| = m/2^n$$

and we are done. Next time I'll spell out a bit more what sort of hash function this is, and then we'll use all this to build reasonably nice **authenticated encryption schemes**.

Authenticated encryption

Informally, an authenticated-encryption (AE) scheme is a symmetric encryption scheme $\Pi = (\mathcal{K},\mathcal{E},\mathcal{D})$ that achieves both privacy and authenticity, e.g., for the nonce-based setting,

$$\mathbf{Adv}_{\Pi}^{\text{priv}}(A) = \Pr[K \xleftarrow{}\mathcal{K}: A^{\mathcal{E}(K,\cdot,\cdot)} \to 1] - \Pr[K \xleftarrow{}\mathcal{K}: A^{\$^{\wedge}|\cdot 2|)} \to 1]$$

$$\mathbf{Adv}_{\Pi}^{\text{auth}}(A) = \Pr[A^{\mathcal{E}(K,\cdot,\cdot)} \text{ forges}]$$

are both "good". Alternative, we can give an all-in-one characterization

$$\mathbf{Adv}_{\Pi}^{\text{ae}}(A) = \Pr[A^{\mathcal{E}(K,\cdot,\cdot),\mathcal{D}(K,\cdot,\cdot)} \to 1] - \Pr[A^{\$(K,\cdot,\cdot),\perp(\cdot,\cdot)} \to 1]$$

that is equivalent. Don't formally show this, but provide the intuition as to each implication.
Claim: AE-security implies ind$-cca security and Nonmalleability-cca.
But don't formalize all these notions. Give intuition instead.

Review

- Definition of AE (two forms: privacy + authenticity, or all-in-one notion)

Definition of a MAC for F: K □ M □{0,1}t

- Ways to achieve a MAC: raw CBC MAC (if all messages of the same length), CMAC, WC paradigm (of which CMAC is an instance), and GMAC (another instance)
- These approaches actually achieve more than a MAC: they achieve a PRF F: $\mathcal{K} \times \mathcal{M} \to \{0,1\}^{\tau}$. Remind definition of a good PRF and a good MAC on this domain

A good PRF is a good MAC:

F is PRF-secure → F is MAC-secure
F is PRF-insecure → F is MAC-insecure
∃ good B breaking PRF security ← ∃ good A breaking MAC security

Definition of B^g:

Run A^f
When A asks its oracle f a query x, answer $g(x)$
When A outputs a forgery attempt (M,T),

return ($g(M)=T$)

$\mathbf{Adv}^{\mathrm{prf}}_{F}(B) = \Pr[B^{F} \to 1] - \Pr[B^{\pi} \to 1]$

$\geq \mathbf{Adv}^{\mathrm{prf}}_{F}(B) - 1/(2^{\tau} - q)$

where q is the number of queries asked by B.

Lecture 19 - ECS 127 - Winter 2019 – 2/22/2019

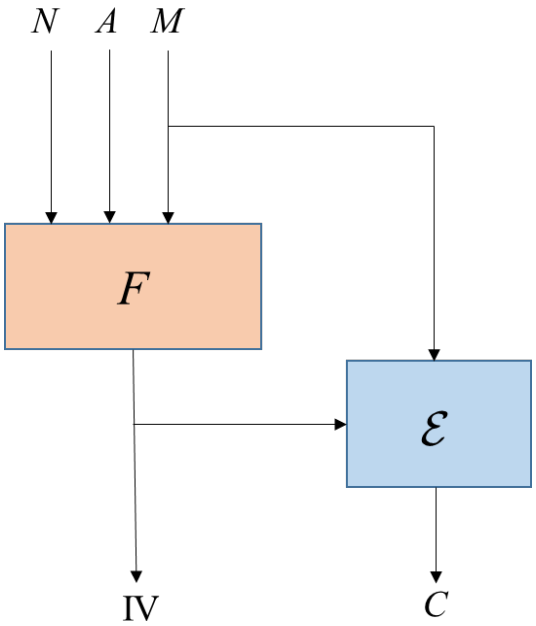
Today: o Survey of some AE schemes
O Cryptographic hash functions

Use slides today

Creating an AE scheme:

First: Encrypt-with-authenticity doesn't work: describe an attack on CBC-encryption with arbitrary unkeyed redundancy $R(M)$ as the last block of message.

Describe **SIV**. (Maybe give a history of the generic composition discourse.) At first, omit the associated data A. Then explain its role.



AEAD

Change the syntax one last time to include **associated data**: a string that should be authenticated but not encrypted. Now encryption and decryption will have signature

$$\mathcal{E}\colon \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M} \to \mathcal{C} \cup \{\perp\}$$

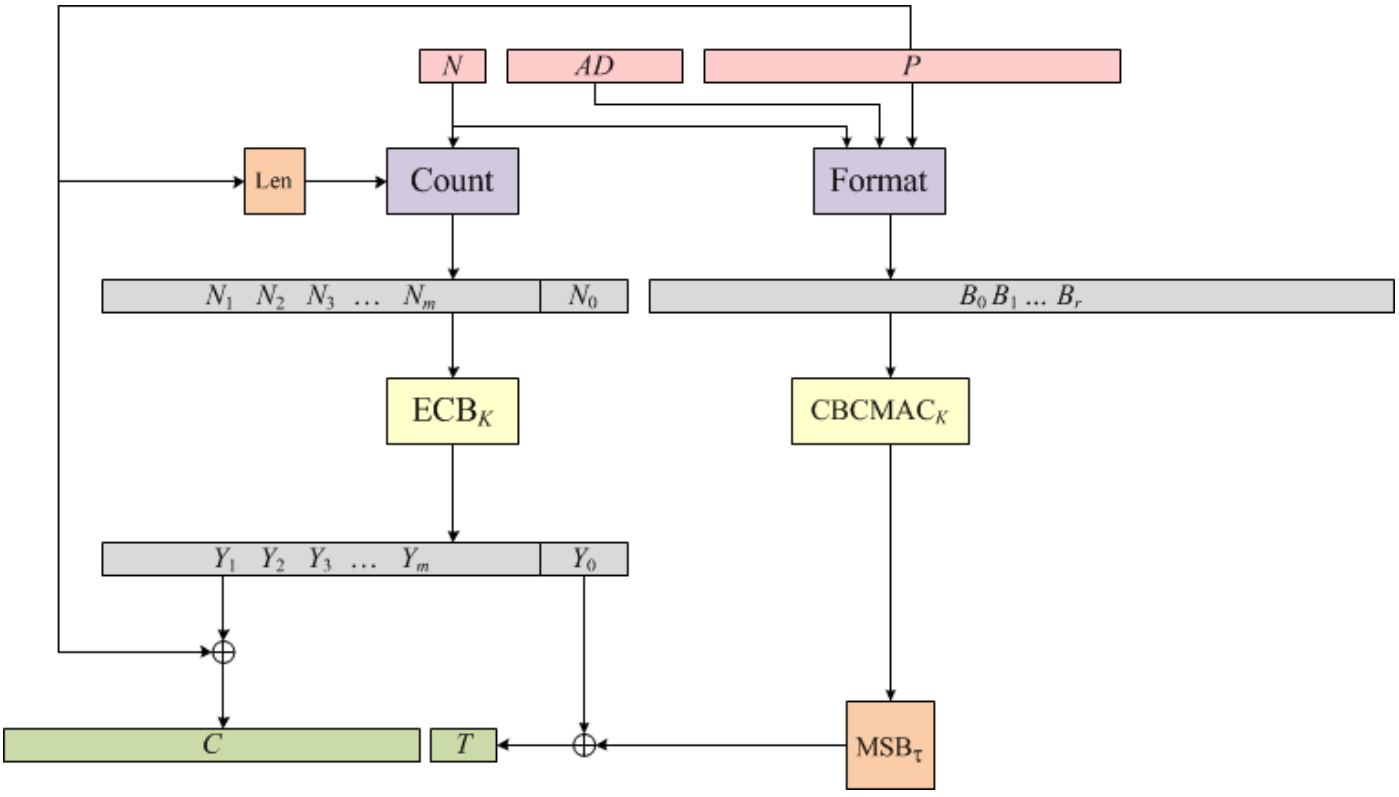
$$\mathcal{D}\colon \mathcal{K} \times \mathcal{N} \times \mathcal{C} \to \mathcal{M} \cup \{\perp\}$$

where we usually want $\mathcal{M} = \mathcal{C} = \mathcal{A} = \{0,1\}^*$ (or BYTE^*)

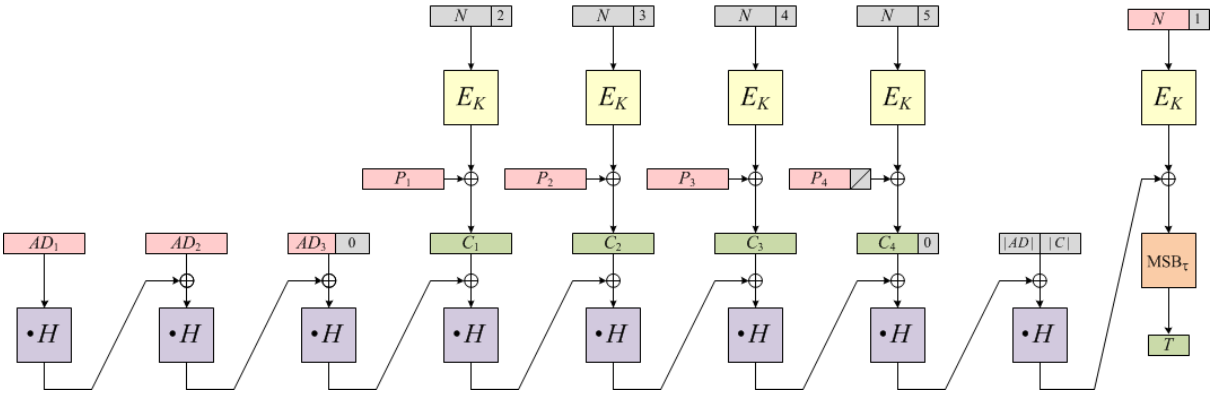
$$\mathbf{Adv}_{\Pi}^{\text{aead}}(A) = \Pr[A^{\mathcal{E}(K,\cdot,\cdot,\cdot),\mathcal{D}(K,\cdot,\cdot,\cdot)} \to 1] - \Pr[A^{\$(K,\cdot,\cdot,\cdot),\perp(\cdot,\cdot,\cdot)} \to 1].$$

Standardized schemes

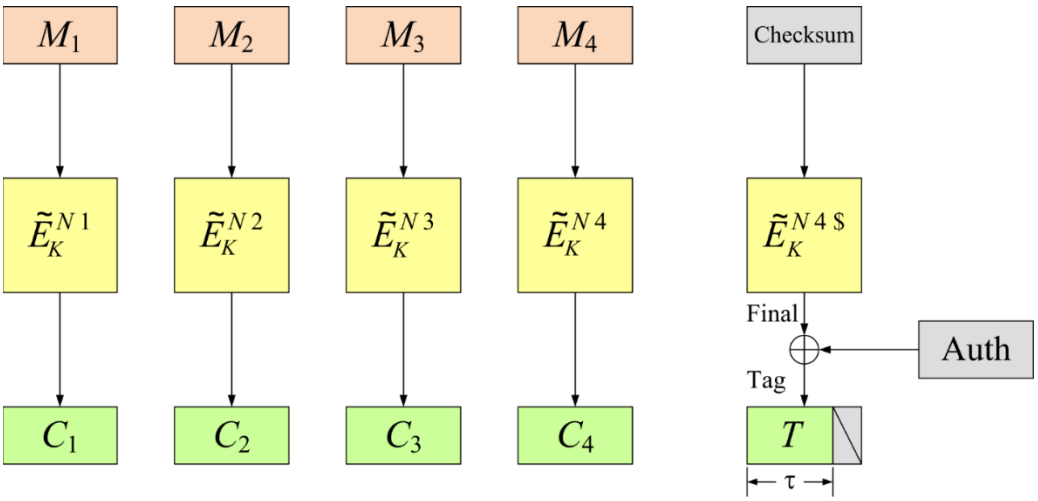
CCM:

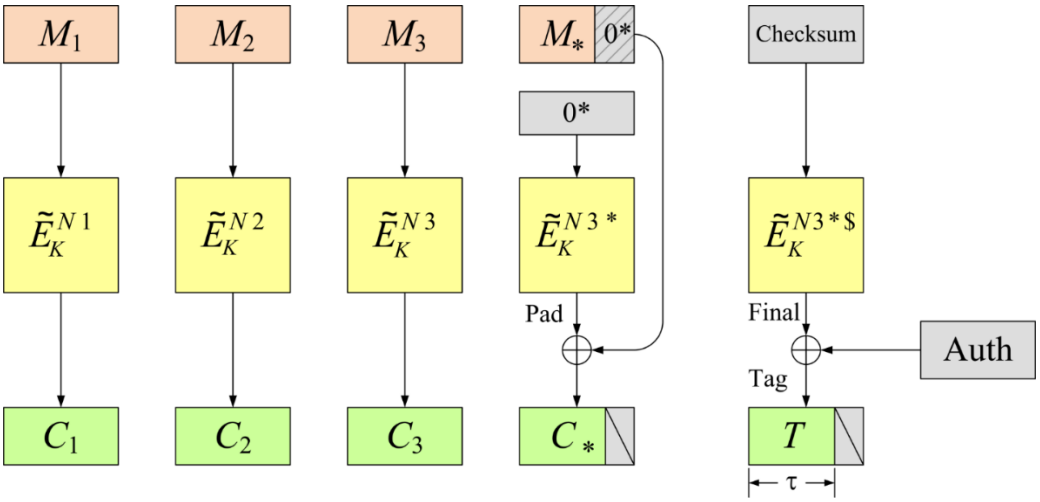


GCM


N 2
N 3
N 4
N 5
N 1
E_K
P_1
P_2
P_3
P_4
AD_1
AD_2
AD_3 0
C_1
C_2
C_3
C_4 0
|AD| |C|
•H
MSB_τ
T

OCB


M_1
M_2
M_3
M_4
Checksum
Ẽ_K^N1
Ẽ_K^N2
Ẽ_K^N3
Ẽ_K^N4
Ẽ_K^N4$
Final
Tag
Auth
C_1
C_2
C_3
C_4
T
τ



Lecture 20 - ECS 127 - Winter 2019 – 2/25/2019

Today: O Cryptographic hash functions
O Formalizing human ignorance

Cryptographic hash function: H: $\mathcal{M} \rightarrow \{0,1\}^n$

They have **no key** (although, as we will see, there is some debate about this choice). A variety of security properties discussed for these objects, the most basic is

Collision intractability (also called **collision resistance**): It is computationally infeasible to find distinct values M and M' such that $H(M) = H(M')$.

The frustration: **lots** of collisions exist. Indeed lots of collisions exist, by the pigeonhole principle, even if you restrict to inputs of length 2n, say. But the adversary, poor thing, can't find even one.

Other commonly discussed properties:

preimage resistance (or one-way-ness)
Given a hash output Z it must be computationally infeasible to find an input x such that $Z = H(X)$ (MOV, p. 297)

second preimage resistance

Given X_1 it is computationally infeasible to find $X_2 \neq X_1$ such that $H(X_2)=H(X_1)$

The "definitions" above aren't yet formal. How would one make them formal? Beware: this is a domain where informality ruled for many years. [attempts to clean up this area: *Cryptographic Hash Function Basics* [Rogaway, Shrimpton 2004]; Formalizing Human Ignorance [Rogaway, 2006])

We will focus on collision intractability.

Uses

1. **Did I get the right file?** Did I download the intended file?
2. **Is the cloud keeping what I told it to?**
3. **Commitment** – I have a proof of some theorem now, I want to reveal it later. With x the proof, would I send H(x)? No, $H(R,x)$. Does this really stem from collision intractability? (No)
4. **Bit-coin mining. Given X, find Y s.t. H(X||Y) ends in 40 zero bits, for example.**

5. Digital signatures
6. Challenge-response protocols
7. MACs
8. Password hashing
9. **Intentionally slow password hashing and, now, memory-hard, intentionally slow password hashing (eg, scrypt, Argon2)**
10. Protocol design in the ROM.
11. And lots more

How long should a hash function be so that it to begin to be feasible for it to be collision resistance? Conventional answer: 128 bits is probably not enough; 160 bits is probably ok; 256 bits is good. Because of **birthday bound**. Simple attack:

```
for i from 1 do
    compute and store H(i) until you find a collision
```

Expect to take about $2^{n/2}$ time and $n\ 2^{n/2}$ space.

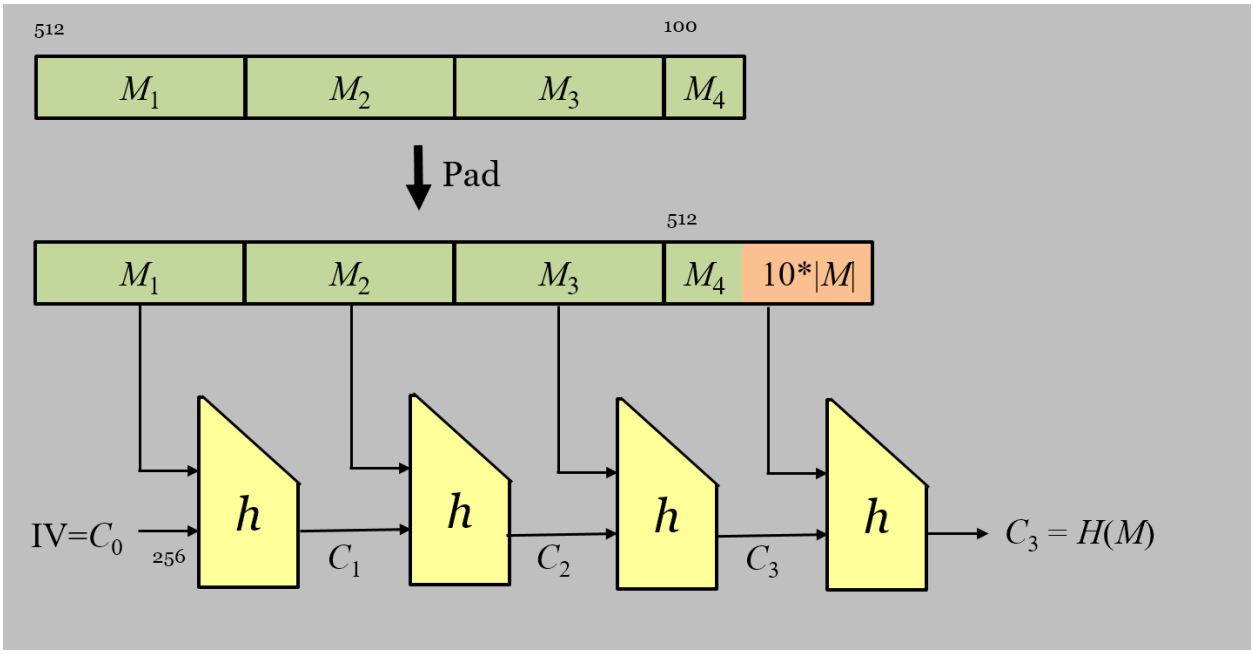
How to achieve CR-hash function:

Earliest construction: **MDC-2** – make CR hash from DES. 1987

Lecture 21 - ECS 127 - Winter 2019 – 2/27/2019

Today: O Cryptographic hash functions, cont
O Public-key encryption

Earliest popular construction: **MD4**, by Ron Rivest, 1990. Beautiful construction, the source of SHA1.
SHA1: NSA, 1993/1995



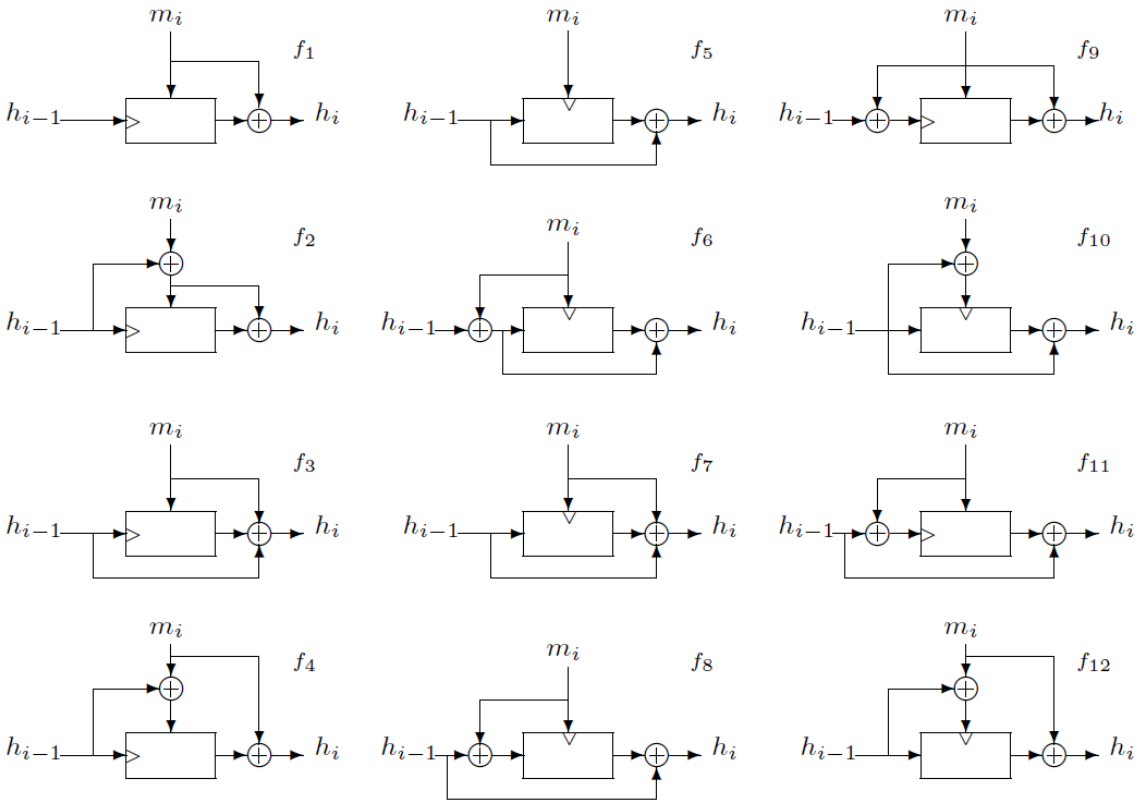
We assume that the padding entails length annotation, so that messages of different lengths have different final blocks. Example: 10* pad so that you are 64 bits less than a multiple of 512 bits; and then encode the bit length of the input as the last 64 bits.

Theorem [Merkle–Damgård]
If f is a CR compression function then MD[f] is CR ⇔
∃ a related B that finds collision in f ⇐ ∃ A finds collision in MD[f]

Proof. Run A. It outputs a colliding M, M'. From this pair, we can recover a colliding pair of inputs to f. If M and M' have different lengths, then the last input to f produced by processing M and M' collide. If they have the same length and $M_m \neq M_m'$, then we are done: compute the inputs to the final compression function, and you have your collision in f. If $M_m = M_m'$ but $H_m \neq H_m'$ (this the chaining value

at the bottom: IV = H_0, H_1, …, H_m), then again we are done: compute the inputs and you have your collision. Otherwise, $M_m = M_m'$ and $H_m = H_m'$, so just back up to the prior block and repeat. This must terminate because the messages are distinct. Said differently: we are just searching for the first point where we have differing inputs into the compression function f, and this must exist because the messages differ but end up hashing to the same value.

Now: how to make the compression function? One approach: use a **blockcipher** and, say, the **Davies-Meyer construction** (construction number f_5 below – but all of the following are correct, according to the work of Rogaway-Shrimpton).



Method 5 is Davies-Meyer. But all of these methods work. Analysis by Rogaway and Shrimpton, 2004.

Eg: SHA-1 blockcipher with 512-bit key, 160-bit input → 160-bit input

All variables are unsigned 32 bits and wrap modulo 2^{32} when adding

Algorithm SHA1(M)

h0←0x67452301; h1←0xEFCDAB89; h2←0x98BADCFE; h3←0x10325476; h4←0xC3D2E1F0

Append to *M* a 1 then the min # k < 512 of 0's s.t. |M|= -64 (mod 512))
Then, for each 512 bit chunk W = w[0] w[1] … w[15] of M //|w[i]|=32

Underlying blockcipher: maps a 160-bit "plaintext" abcde and a 512 bit "key" W=w[0]… w[15] (the message) to a new 160-bit "ciphertext" abcde.

for i←16 **to** 79 **do** w[i] ← (w[i-3]⊕ w[i-8]⊕ w[i-14]⊕ w[i-16])<<< 1

a ← h0; b ← h1; c ← h2; d ← h3; e ← h4;
for i **from** 0 **to** 79

if 0 ≤i≤19 **then** f ← (b and c) or ((not b) and d), k ← 0x5A827999

if 20≤i≤39 **then** f ← b ⊕ c ⊕ d, k ← 0x6ED9EBA1

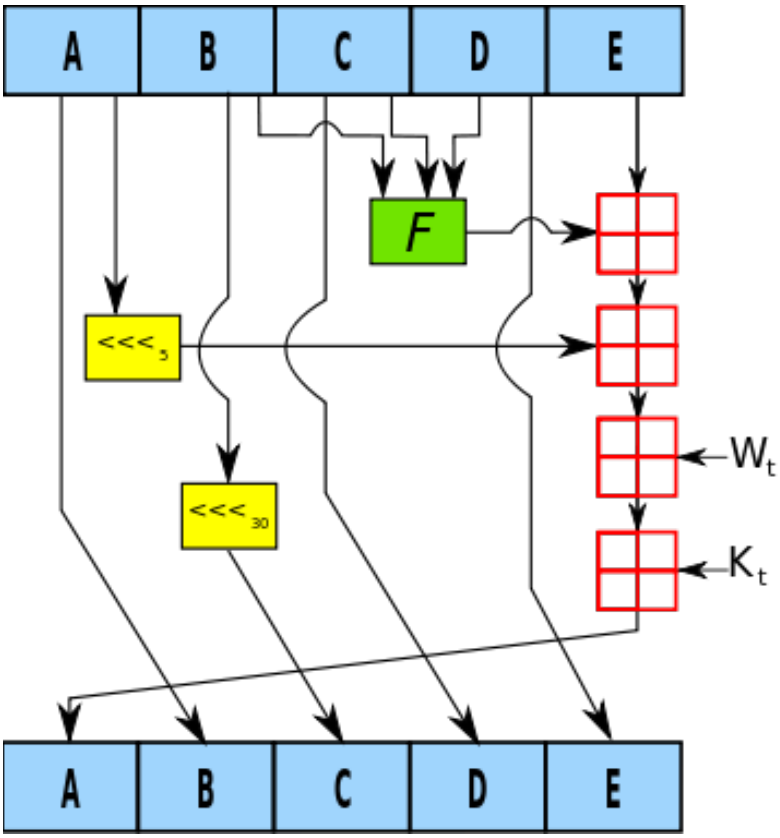
if 40≤i≤59 **then** f ← (b and c) or (b and d) or (c and d), k ← 0x8F1BBCDC

if 60≤i≤79 **then** f ← b ⊕ c ⊕ d, k ← 0xCA62C1D6

temp ← (a <<< 5) + f + e + k + w[i]

e ← d; d ← c; c ← b <<< 30; b ← a; a ← temp

h0 ← h0 + a; h1 ← h1 + b; h2 ← h2 + c; h3 ← h3 + d; h4 ← h4 + e
return h0 || h1 || h2 || h3 || h4

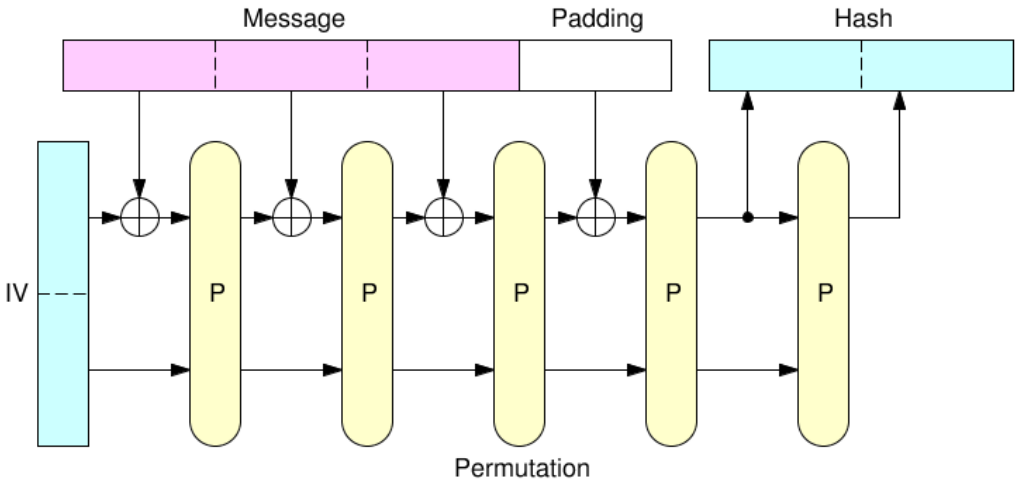


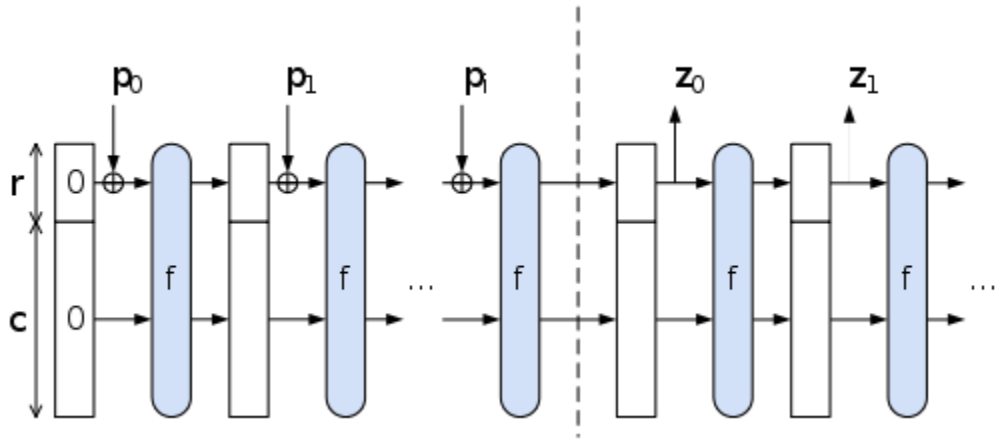
Attacks: "In February 2005, an attack by Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu was announced. The attacks can find collisions in the full version of SHA-1, requiring fewer than 2^{69} operations." Subsequently lowered to 2^{69} operations. Estimated cost of finding a collision about $3 million??

SHA-3 competition. Formally announced 2015.

Below is the **sponge construction** from Bertoni, Daemen, Peeters, & van Assche. It is based on a **cryptographic permutation**. SHA-3 = Keccak is based on this.

Permutation P below is quite wide – 1600 bits – with





SHA-3 competition. Formally SHA-3 competition. Formally

Lecture 22 - ECS 127 - Winter 2019 – 3/01/2019

Today: O Public-key encryption: notion and a first construction
O Trapdoor permutations

Syntax

pk *pk, sk*
A ---------------> B

Syntax: a **public-key encryption scheme** is a three-tuple of algorithms $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ where

- $\mathcal{K}$ is a prob. algorithm that, on input $k \in \mathbf{N}$ (or 1^k if you wish to emphasize what polynomial-time" is measured in terms of k), outputs a pair of strings (pk, sk).

- $\mathcal{E}$ is a prob. algorithm that, on input $pk \in \{0,1\}^*$ and $M \in \{0,1\}^*$, outputs a value $C \twoheadleftarrow \mathcal{E}(pk,M)$ that is either a binary string or the symbol $\perp$.
- $\mathcal{D}$ is a deterministic algorithm that, on input of $sk \in \{0,1\}^*$ and $C \in \{0,1\}^*$, outputs a value string $C \leftarrow \mathcal{D}(Sk,S)$ that is either a binary string or the distinguished value $\perp$.

We assume that there's a well defined **message space** for each public key: if *pk* is output with nonzero probability on input *k*, then whether or not encrypting *M* with *pk* gives $\perp$ is independent of the coins used. And we assume **correctness**: if (*pk*, *sk*) is produced by running K on *k*, and if $C \twoheadleftarrow \mathcal{E}(pk,M)$is a string, then $\mathcal{D}(sk,C)=M$.

Merkle Puzzle: *(Ralph Merkle, submitted 1974, as a grad student)*
Choose 10^7 random 128-bit numbers $R_1, \ldots, R_{10^\wedge 7}, K_1, \ldots, K_{10^\wedge 7}$ and have Alice send to Bob, in alphabetical or random order,

$\text{Puzzle}_1 = \text{Allbut}_{40}(K_1) \,||\, H(K_1) \oplus ([1]_{128} \,||\, R_1)$
$\text{Puzzle}_2 = \text{Allbut}_{40}(K_2) \,||\, H(K_2) \oplus ([2]_{128} \,||\, R_2)$
…
$\text{Puzzle}_{10^\wedge 7} = \text{Allbut}_{40}(K_{10^\wedge 7}) \,||\, H(K_{10^\wedge 7}) \oplus ([10^7]_{128} \,||\, R_{10^\wedge 7})$

Allbut(K,t) = K[1..|K|-t]

--->
430 MBytes

Bob choose a random one of these puzzles, solves it (in 2^40) time, and sends to Alice the number j of the puzzle she solved, a number between 1 and 10^7. They now share j.

If it take Bob **10 minute** to do his work. If an attacker needs the same amount of time, it will take the attacker 2^{20} times more time (worst case), which is about **200 years** (worst case; **100 years** expected).

Diffie-Hellman key exchange
First a bit of number theory background: For *p* prime, $\mathbf{Z}_p$ is a field, and $\mathbf{Z}_p^*$ is a group, the **multiplicative subgroup** of $\mathbf{Z}_p$. It is a **cyclic** group, generated by a single element = $\langle g \rangle = \mathbf{Z}_p^*$

Eg: $\mathbf{Z}_5 = \{0,1,2,3,4\}$

$\mathbf{Z}_5^*$ = {1,2,3,4}. | $\mathbf{Z}_p^*$| = p - 1.

	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	1	2	4	2
4	4	3	2	1

Generated by 2: $2^0 = 1$, $2^1 = 2$, $2^2 = 4$, $2^3 = 3$ $2^4 = 1$

Lagrange's theorem: if G is a finite group with m elements and $a \in G$, then $a^m = 1$.

Now for the mechanism:

$a \twoheadleftarrow [0..p-2]$

```
            A=g^a
Alice ------------> Bob
                    b ↞[0..p-2]
            B=g^b
       <------------
```

Compute
$DHK = B^a = g^{ab}$
$K = H(Secret)$

Compute
$DHK = A^b = g^{ab}$
$K = H(Secret)$

← Version 1: output DHK
← Version 2: output $H(DHK)$ for some cryptographic hash function H

Lecture 23 - ECS 127 - Winter 2019 – 3/04/2019

Today: O Public-key encryption

$a \twoheadleftarrow [0..p-2]$

```
            A=g^a
Alice ------------> Bob
                    b ↞[0..p-2]
            B=g^b
       <------------
```

Decisional Diffie-Hellman Assumption

Fix a finite cyclic group $G = \langle g \rangle$ of size m.
Let $\mathbf{Adv}^{\text{ddh}}_G(A) = \Pr[a,b \xleftarrow{}[0..m\text{-}2]: A(g^a, g^b, g^{ab}) \to 1] - \Pr[a,b,c \xleftarrow{}[0..m\text{-}2]: A(g^a, g^b, g^c) \to 1]$
is "small" for any "reasonable" adversary A.

For an **asymptotic definition**, let the security parameter k determine the length of a randomly selected prime p, let g be the smallest generator for this group (alternatively, reject any prime p for which 2 is not a generator), and give the adversary g and p as input.)

Problem: DDH isn't true for $\mathbf{Z}_p^*$ – and it's a very strong assumption for any group. DDH = "Day-dreamer's hypothesis"?

Computational Diffie-Hellman Assumption
Fix a finite cyclic group $G = \langle g \rangle$ of size m.
Let $\mathbf{Adv}^{\text{cdh}}_G(A) = \Pr[a,b \xleftarrow{}[0..m\text{-}2]; y \xleftarrow{} A(g^a,g^b): y = g^{ab}]$
is "small" for any "reasonable" adversary A.

Let $G=\langle g \rangle$ be a cyclic group of order m and let $H: G \to \{0,1\}^n$ be a cryptographic hash function. Define

$$\mathbf{Adv}^{\text{hdh}}_{G,H}(A) = \Pr[a,b \xleftarrow{}[0..m\text{-}2]: A(g^a, g^b, H(g^{ab})) \to 1] - \Pr[a,b \xleftarrow{}[0..m\text{-}2]; C \xleftarrow{}\{0,1\}^n: A(g^a, g^b, C) \to 1]$$

In the **random-oracle model** (ROM), CDH is enough for Diffie-Hellman based encryption (with hashing) to work.

Standard-model definition for an IND-CPA secure encryption scheme:

$$\mathbf{Adv}^{\text{priv}}_{\Pi}(A,k) = \Pr[(pk,sk) \xleftarrow{} \mathcal{K}(k): A^{\mathcal{E}(pk,\cdot)}(pk) \to 1] - \Pr[(pk,sk) \xleftarrow{} \mathcal{K}(k): A^{\mathcal{E}(pk,0|\cdot|)}(pk) \to 1]$$

Asymptotic definition: for any PPT A, $\mathbf{Adv}^{\text{priv}}_{\Pi}(A,k)$ is negligible.

In the **ROM**:

$$\mathbf{Adv}^{\text{priv}}_{\Pi}(A,k) = \Pr[H \xleftarrow{} \Omega; (pk,sk) \xleftarrow{} \mathcal{K}^H(k): A^{H,\ \mathcal{E}^H(pk,\cdot)}(pk) \to 1] - \Pr[H \xleftarrow{} \Omega; (pk,sk) \xleftarrow{} \mathcal{K}^H(k): A^{H,\ \mathcal{E}^H(pk,0|\cdot|)}(pk) \to 1]$$

From key exchange to public-key encryption

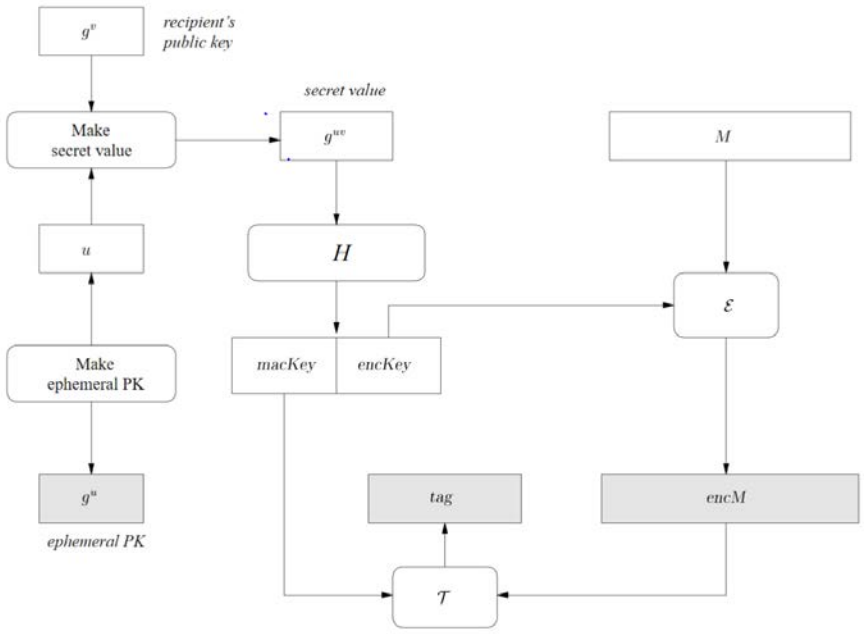
How to go from a key-exchange scheme to an encryption scheme? Illustrate with DH (whence the key-exchange is called ElGamal encryption) (more or less).

Bob's public key is g^b. Alice encrypts M by choosing an **ephemeral public key** g^a and sending $g^a \,||\, C$ *where* $C \leftarrow \mathcal{E}(K,M)$ for $\mathcal{E}$ the encryption algorithm of a symmetric encryption scheme and $K = H(g^{ab})$ the shared key from the key exchange. Works to transform any key exchange plus symmetric encryption scheme into a public-key encryption scheme.

Since we're using the cryptographic hash function in the desired form of DH key exchange, we can just use it exclusively, as though with a OTP/Vernam cipher built from H. This would make the ciphertext for plaintext M and public key $B = g^b$:

$$g^a \,||\, H(g^{ab}) \oplus M \qquad \textit{Hashed Diffie-Hellman encryption}$$

A problem: highly malleable – only achieves IND-CPA security. There are natural approaches to do better.



-
-
- Review **hybrid DH encryption**, no hash

Bob's public key is g^b. Alice encrypts M by choosing an **ephemeral public key** g^a and sending $g^a \,||\, C$ *where* $C \leftarrow \mathcal{E}(K,M)$ for $\mathcal{E}$ the encryption algorithm of a symmetric encryption scheme and $K = g^{ab}$ the shared key from the key exchange. Works to transform any key exchange plus symmetric encryption scheme into a public-key encryption assuming the DDH. But serious problems: 1) weird, non-uniform key for symmetric encryption scheme; 2) that key is actually not indistinguishable from random bits (even in $\mathbf{Z}_p$): unpredictable, not pseudorandom. DDH vs CDH. Better: $K = H(g^{ab})$. What assumption is needed for H? It is not collision resistance. Explain the **random-oracle model** (ROM) and its genesis.

Diffie and Hellman didn't actually seem to be interested in encrypting with the method described above, and it's not clear they understood it. What the did put forward was the idea of a trapdoor permutation. That's what the imagined using to encrypt:

```
      easy
X ------------> f(X)

      hard
  <-------------
```

Lecture 24 - ECS 127 - Winter 2019 – 3/06/2019

Today: O Public-key encryption with RSA and the ROM
O Digital signatures

Def: A **trapdoor permutation generator** is a probabilistic algorithm $\mathcal{G}$ that, on input k, outputs a pair of strings (each encoding an algorithm) $(\_f, \_g)$.

$\_f$ describes a permutation f on some set Dom(f)
$\_g$ describes its inverse: $\_g \circ \_f = \mathbf{id}$ on Dom(f)

We intend that it is **easy** to compute $\_f$ given its description, but that it's **hard** to compute its inverse in the absence of the string g.

Formally,

$$\mathbf{Adv}^{\mathrm{tdp}}_{\mathcal{G}}(A,k) = \Pr[(\_f,\_g) \leftarrow \mathcal{G}(k);\ x \leftarrow \mathrm{Dom}(f);\ x' \leftarrow A(f,f(x)):\ x = x']$$

We want this to be "small" for any "reasonable" adversary A. For an asymptotic definition, $\mathcal{G}$ must run in PT in we insist that $\mathbf{Adv}^{\mathrm{tdp}}_{\mathcal{G}}(A,k)$ be negligible for any PPT A.

Now Diffie and Hellman wanted to base encryption on a trapdoor permutation because they were interested in using the same tool for encryption and for digital signatures, and idea they also invented:

> **encryption**: public key = _f, secret key = _g, encrypt by applying f to the plaintext, decrypt by applying g to the ciphertext.
>
> **signatures**: signature key = g, verification key = f, sign by applying g to the message, verify a signature by apply f to the signature and seeing if you get back the message.

The irony is that this is
(1) **not** a 'correct' way to do encryption
(2) **not** a 'correct' way to do signatures
(3) We **don't need** trapdoor permutations to do signatures. Indeed we can do them from something the ElGamal encryption method, or even from a hash function like SHA1 or DES (as Ralph Merkle first showed).

Why is a trapdoor permutation a wrong way to do encryption?

First, it's **deterministic**: encrypt the same message twice, you get the same thing. So it can't meet the security definitions we've described.

In fact, we do **not** use trapdoor permutations (eg, school book RSA) to encrypt.

Nor, for that matter, do we use raw trapdoor permutation to sign.

RSA Trapdoor permutation

Generator $\mathcal{G}$: Given k,

1. Compute two random k-bit primes p and q. Let $N=pq$. Let $\phi(N)=(p-1)(q-1)= |\mathbf{Z}_N^*|$. Let e=3. If e is not relatively prime to $\phi(N)$then go to 1. Let $d = 1/e \bmod \phi(N)$ – that is, choose d such that $de = 1 \bmod \phi(N)$.
2. The forward function f, described by (N,e), is the function $f(x) = x^e \bmod N$. It's domain is $\mathrm{Dom}(f) = \mathbf{Z}_N^*$.
3. The backward direction g, described by (N,d), is the function $g(y) = y^d \bmod N$.

Check that the functions are actually inverses of one another: for x in $\mathbf{Z}_N^*$,

$$\begin{aligned} g(f(x)) &= (x^e \bmod N)^d \bmod N \\ &= x^{ed} \bmod N \\ &= x^{1+k\phi(N)} \bmod N \\ &= x\, x^{k\phi(N)} \bmod N \end{aligned}$$

$= x \bmod N$
$= x$

Implicitly used mathematical facts:

1. Lagrange's theorem, $a^{|G|} = 1$ (in any finite group G)
2. you can test primality
3. there are plenty of prime numbers (the prime-number density theorem:

$$\pi(x) \sim \frac{x}{\log x}.$$

Meaning

$$\lim_{x\to\infty} \frac{\pi(x)}{\left[\frac{x}{\log(x)}\right]} = 1,$$

4. the set $\mathbf{Z}_N^*$ forms a group; its cardinality is $\phi(n) = |\mathbf{Z}_N^*| = (p-1)(q-1)$ when $N=pq$ *is the product of distinct primes*
5. finding inverses mod m is easy, and things relatively p*rime to m have inverses [use Eucl*id's algorithm:
   for all a,b exists x,y s.t. ax + by = gcd(*a*,*b*)
   So when (*a*,*b*)=1 a*x*+by =gcd(a,b)
   so when (*a*,*m*)=1 $ax+my = 1 \pmod m$.
   So $ax = 1 \pmod n$ *so a(mod m) is the inverse of x*)

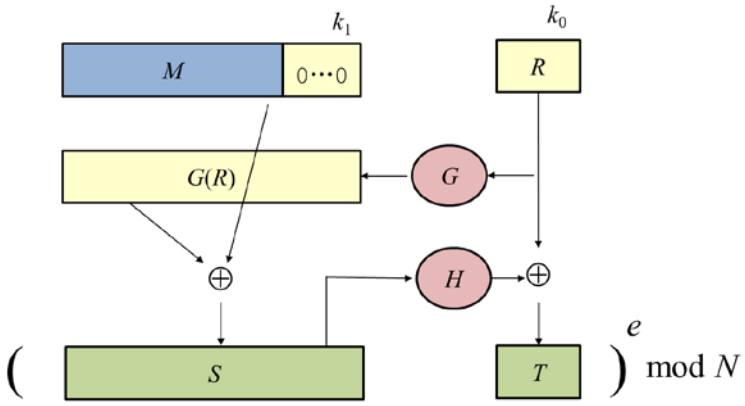
Raw RSA is not a good encryption scheme: making an encryption scheme and a signature scheme out of it.

Simple **hybrid encryption** with a trapdoor permutation:

Choose a random *R* in Dom(f), then the ciphertext for *M* will be

$$\mathcal{E}_f(M): \quad R \xleftarrow{} \mathrm{Dom}(f);\ \text{return}\ \ \langle f(R),\ \mathcal{E}(H(R),M)\rangle$$

A method with stronger guarantees – CCA2 for the RSA trapdoor permutation: RSA-OAEP (Bellare-Rogaway, 1994)



The RSA Assumption

"RSA is one-way" or "RSA is trapdoor"

$\mathbf{Adv}^{\mathrm{rsa}}_{\mathcal{G}}(A,k) = \Pr[((N,e),(N,d)) \leftarrow \mathcal{G}(k);\ x \leftarrow \mathbf{Z}_N^*;$

$y \leftarrow x^e \pmod N : A(N,e,y) \rightarrow x]$

In other words, you can't get all of x from x^e. But it doesn't mean you can't get *some* of x. What can't you get?

a) Generically, for **any** one-way function, you can't get the inner-product bit: that is, $\langle x,r\rangle$ for a random bit string r. This is a famous result of Goldreich and Levin. If you could compute the inner product with random values r, you could, albeit less efficiently, invert the underlying one-way function.

   The inner product bit provides a way to encrypt **a bit** b: Choose $x \leftarrow \mathbf{Z}_N^*$; choose $r \leftarrow \{0,1\}^k$; let the ciphertext be $C = (x^e \bmod N,\ r,\ \langle x,r\rangle)$. The method works for any trapdoor permutation: $C = (f(x),\ r,\ \langle x,r\rangle \oplus b)$ *when the trapdoor permutation generator produced* $(f,finv) \leftarrow \mathcal{G}(k)$.

b) For the RSA function, you can't get the **lsb**. If you could compute the lsb, you could (albeit less efficiently) invert RSA. Whence the ciphertext can be $C = (x^e \bmod N,\ \mathrm{lsb}(x) \oplus b)$ for $x \leftarrow \mathbf{Z}_N^*$.

c) Go to the ROM

But not efficient. Each bit of plaintext encrypts to 1024 bits of ciphertext, say. What's done in practice.

Lecture 25 - ECS 127 - Winter 2019 – 3/08/2019

Today: O A bit of review
O Digital signatures

Quiz: trapdoor permutation;

PKCS #1 encryption

Original approach, early 1990s.

(00 02 $$ $$ $$ ... $$ $$ 00 *M*)^{e} mod *N*

where each $$ is a random nonzero byte and there are at least eight of these. Discuss problems. Problem: no provable-security guarantee. CCA security? Alternative, adopted as PKCS #1.5 and subsequent: OAEP (Bellare-Rogaway, 1996)

Digital Signatures

Review trust model, goal. Then give a proper definition:

Syntax: A digital signature scheme is a tuple $\Pi = (\mathcal{K}, \text{Sign}, \text{Vf})$ where

- $\mathcal{K}$ is a probabilistic algorithm that, on input of a number k, produces a pair $(\text{pk}, \text{sk}) \twoheadleftarrow \mathcal{K}(k)$.
- Sign is a probabilistic or deterministic algorithm that, on input of sk and $M \in \{0,1\}^*$, produces a signatures $\sigma \leftarrow \text{Sign}(\text{sk}, M)$.
- Vf is a deterministic algorithm that, on input pk,M,σ, produces a bit $v \leftarrow \text{Vf}(\text{pk}, M, \sigma)$.

Correctness: If $(\text{pk}, \text{sk}) \twoheadleftarrow \mathcal{K}(k)$ and $\sigma \leftarrow \text{Sign}(\text{sk}, M)$ then $\text{Vf}(\text{pk}, M, \sigma)=1$.

Diffie and Hellman's original idea: run a trapdoor permutation generator $\mathcal{G}$ to produce (f,g), which are the (pk, sk). Then

$\text{Sign}(\_g, M) = g(M)$. $\text{Vf}(f, M, \sigma) = (f(\sigma)=M)$.

Thus, for RSA, $\text{Sign}((N,d),M) = M^d \bmod N$.

But: even if you believe the RSA assumption, this completely doesn't work to give an unforgeable signature scheme:

- Forge $M = 1$ as $\sigma=1$

- Or given signatures $\sigma_1 = M_1^d \bmod N$ and $\sigma_2 = M_2^d \bmod N$, *you can forge* $M_1 M_2 \bmod N$ using a signature of $\sigma_1 \sigma_2 \bmod N$ because $(M_1 M_2)^d \bmod N = (M_1^d \bmod N)(M_2^d \bmod N) \bmod N$.

Lecture 26 - ECS 127 - Winter 2019 – 3/22/2019

Today: O Digital signatures
o AKE

Announcements:

o For **Wednesday**, please read before class the essay "The Moral Character of Cryptographic Work".

O For Friday: Evals in class. Bring your laptop!

o Final March 22: Review session: Next Tuesday or Wednesday early evening? 4-6 pm

Last time we defined signatures (review that defn), described trying to sign with a trapdoor permutation in general and RSA in particular (neither worked)and ended with this slide still up on the board showing RSA PKCS #1 v.1 signatures.

Review ROM, FDH.

- **PKCS #1 signature encoding**

(00 01 ff ff ff ... ff ff 00 $H(M)$)^d mod N

Discussion issues. Describe **FDH** (full-domain hash)

$$(H(M))^d \bmod N$$

o Digital signatures
- PKCS #1, v1
- FDH
- Signing from a hash function: Lamport signatures
- An important use of digital signatures: AKE by way of signing a DH key exchange: transition to KD/AKE

First finish PKCS #1 signatures and FDH

Lamport (one-time) signatures

Secure signatures exist if OWFs do: [Naor, Yung 1989] (UOWHF->signatures), and (OWF -> UOWHF) (Rompel 1990) (construction of UOWHFs from one-way functions). But for Lamport signatures we start with a cryptographic hash function, either thought of in the ROM or just collision-resistant.

We have a message $M = m_1\ m_2\ \ldots\ m_t$ we would like to sign.
Just one message to be signed – **one-time signature**
Public key
pk = $\boldsymbol{H(K_1^0)}$ $H(K_2^0)$ $H(K_3^0)$ $H(K_4^0)$ $H(K_5^0)$ $H(K_6^0)$
$H(K_1^1)$ $\boldsymbol{H(K_2^1)}$ $H(K_3^1)$ $H(K_4^1)$ $H(K_5^1)$ $\boldsymbol{H(K_6^1)}$

To sign X = 011001, release $K_1^0\ K_2^1\ K_3^1\ K_4^0\ K_5^0\ K_6^1$

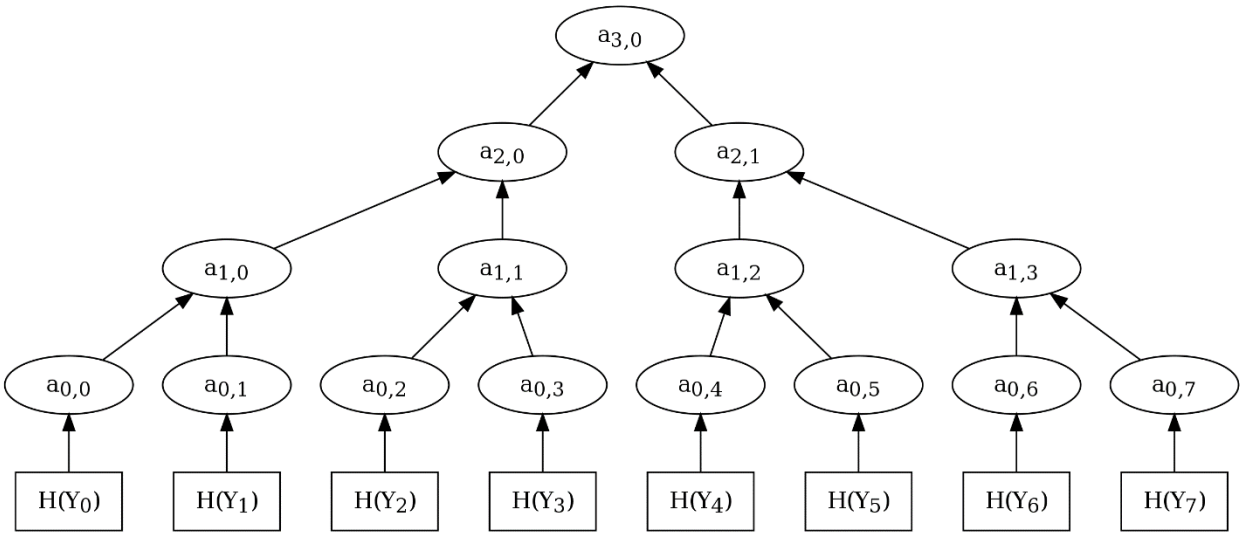
Why is it enough so sign 128 bit strings, for example, to allow you to sign any string? Answer: just hash first.

Can you think of a way to **shorten the public key** at the expense of the signature? Answer:

PK = H(pk) // outputs as many bits as you need
SIGN_SK(M) = (pk, Sign_sk(M))

C to

Merkle signatures
Above just takes care of one signature. How to sign a sequence of messages $M_1, \ldots, M_q$?

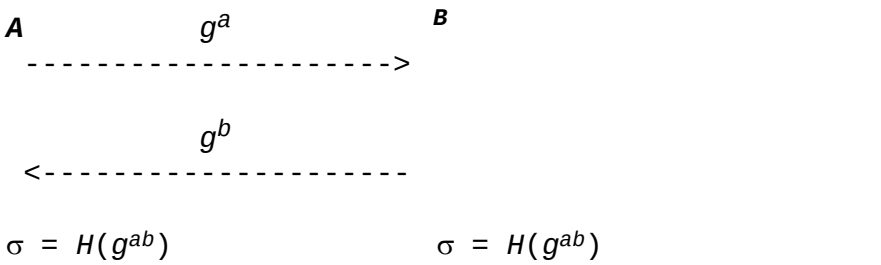


- Key distribution / AKE
  - 2-party, shared key (smart-card to terminal)
    2-party, shared PW
  - 2-party, pw and f(pw)

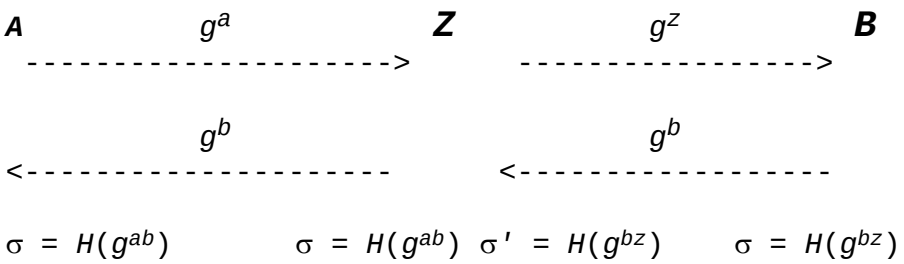
- 3-party model
- 2-party public-key setting, bilateral authentication
- 2-party public-key setting, unilateral authentication:
- TLS 1.3. https://www.youtube.com/watch?v=grRi-aFrbSE
  Basic structure: **record-layer protocol** followed by **handshake**

An important use of digital signatures: **AKE by way of signing a DH key exchange**. Transition to KD/AKE.

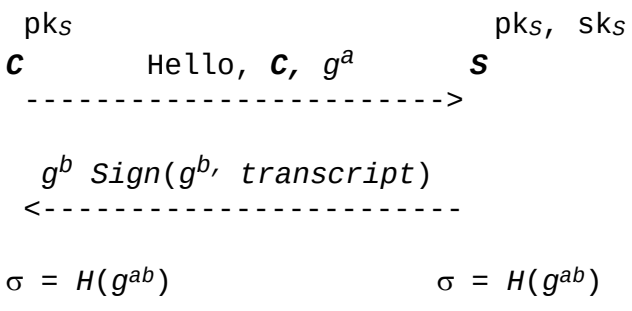
Recall



Does nothing to ensure that *A* and *B* are any.
Relatedly, susceptible to man-in-the-middle (MiM)attack ("grandmaster chess problem"):



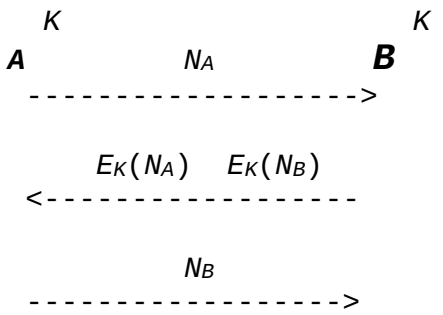
Authenticating a party and sharing a session key using public-key cryptography:



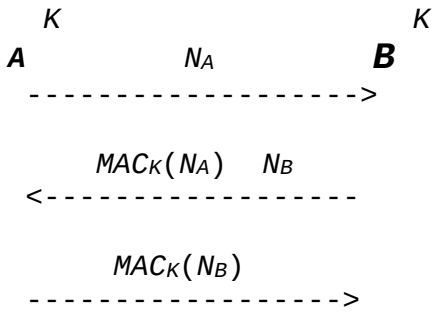
One-way authentication (server to client)
Perfect forward secrecy

Have to be extremely careful in the design of this sort of protocol. Describe alternative trust models

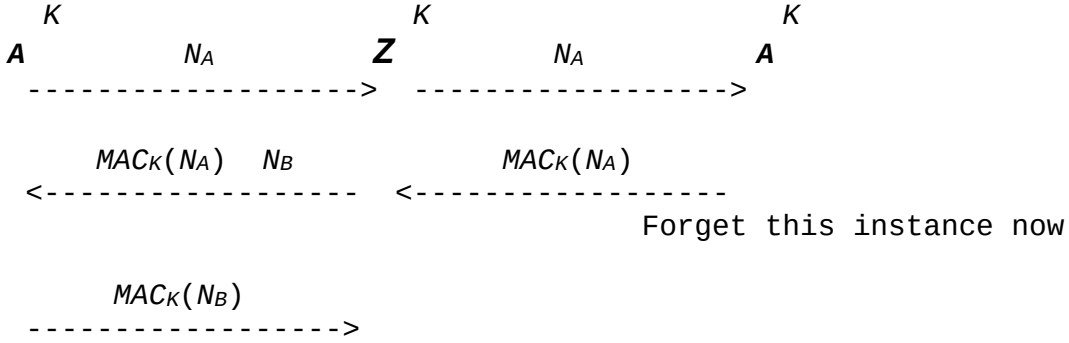
and attacks, in particular, ones staring up new **sessions**:



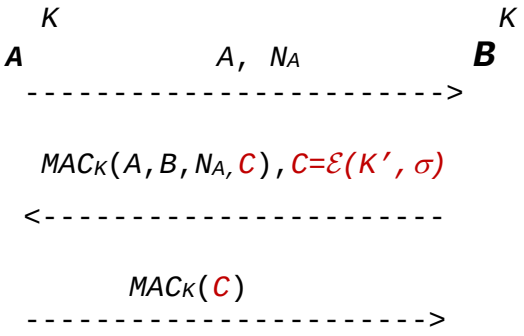
Conceptually wrong – what does privacy have to do with it?



Still all wrong: starting up **instances:**



What's the point of this, anyway, in the absence of a key being distributed?



Give some history of

SSL/TLS
Origins: SSL, from Netscape, circa 1995
SSL v2, v3, TLS 1.0, 1.1, 1.2, *1.3*

Basic structure:
Handshake protocol – negotiate parameters and keys, one-sided authentication. Main part of the complexity. Uses PK cryptography.
Record protocol – to actually carry the data. Uses symmetric cryptography. AES, ChaCha.

Lecture 27 - ECS 127 - Winter 2019 – 3/13/2019

Today: O Moral Character paper

Announcements:
- Review Session 3:30 – 5:30 1003 Giedt next Wed
- Bring laptop/phone on Friday. There will be a quiz on Fri.

Lecture 28 - ECS 127 - Spring 2016 – 6/01/2016

Today:
- **Garbled Circuits**
- **Contest winners:** to be announced.
- **Concluding remarks.**
- **Evals**

Announcements:

- OH **Thurs** 1-3 will be my last.
- Review Session 3:30 – 5:30 1003 Giedt next Wed
- Final **Mon** 3:30-5:30 here

Describe how garbled circuits work, using the two-key tweakable blockcipher abstraction. Show how to combine this with OT (oblivious transfer) to solve 2-party MPC.

Final comments: Today is a "youth strike" at the schools. Why are you all here? Do you even know that you're supposed to be out on strike, because of inaction on climate change? Why is a 16-year old depressive the leading voice on climate inaction? Maybe you should stop worrying about your final exam sand worry more about our fucked-up planet? Our world is in crisis, but, operationally, it is as though almost nobody cares. Please find the humanity within you to care.

Bye!!

Discussion section topic: the asymptotic approach
[illustrate with a PRG]

Anonymous post on piazza: what does it really mean to say that notion *X* implies notion *Y*? (Or, for that matter, what does it mean to say that if some object satisfies notion *X* then some object built from it satisfies notion *Y*? The viewpoint I have tried to give: we don't define such things, which are just shorthand for asserting the existence of reductions. But there is another view, that where we *do* define these things.

Fixed PRG: A map $G\colon \{0,1\}^n \to \{0,1\}^N$ for constants $N > n$.
$\mathrm{Adv}_G(A)$ = a real number.
A **secure PRG:** not formally defined.

Asymptotic PRG: A map G: $\{0,1\}^* \to \{0,1\}^*$ such that $|G(x)| = 2|x|$, say. (This is a PRG with *stretch* $s(n)=2n$. We could use other functions instead of doubling, as long as they are length-increasing.)

$\mathrm{Adv}_G^{\mathrm{prg}}(A)$ = a real-valued function of k:
$\mathbf{Adv}_G^{\mathrm{prg}}(A,k) = \Pr[A^G(1^k) \to 1] - \Pr[A^{\$}(1^k) \to 1]$

A function ε: $\mathbf{N} \to \mathbf{R}^+$ is **negligible** if for all polynomial p there exists a number N_0 such that for $k \geq N$, $\varepsilon(k) \leq 1/p(k)$ for all $k \geq N$.

An (asymptotic) PRG G is **secure** if
for any PPT adversary A, $\mathrm{Adv}_G^{\mathrm{prg}}(A,k)$is negligible.

PRP: A map E: $\mathcal{K}$ x $\{0,1\}^n \to \{0,1\}^n$ where $\mathcal{K}$ is a finite set and n is a constant and each E(K,.) is permutation.

Asymptotic PRP: A map E: $\{0,1\}^*$ x $\{0,1\}^* \to \{0,1\}^*$ where $E(K,.)$ is permutation on $|K|$-bit strings.
$\mathbf{Adv}_G^{\mathrm{prg}}(A,k) = \Pr[A^E(1^k) \to 1] - \Pr[A^{\pi}(1^k) \to 1]$

An (asymptotic) PRP E is **secure** if
for any PPT adversary A, $\mathrm{Adv}_G^{\mathrm{prp}}(A,k)$is negligible.

Now we can make statements like:
If there exists a secure PRG then there exists a secure PRP
or (after defining KR-security in the asymptotic sense)
If a blockcipher is PRP-secure then it is KR-secure.
(Both of these statements are true)

There is a **benefit** to going this route:
- We get to define notions like “a secure blockcipher”.
- We get simple but rigorous language for describing what implies what.
- We don’t have to concern ourselves about just how good reductions are: polynomiality and negligibility help us see beyond these things.

There is a **cost** to going this route:
- The language is harder to fully understand, with all those alternating quantifiers
- We don’t directly get to see how strong our reductions are
- We have to kind of **lie** about the nature of these objects.

The last one is killer for me: it is simply not true that objects like blockcipher are defined for every integer-valued security parameter. It is simply not what the objects provided by cryptographic practice **are**.