ENG 5383
Applied Cryptography

Sherman Chow
Chinese University of Hong Kong
Fall 2020
Lecture 2: Encryption
“Basic” Requirement of IT-Security

- Alice sends a message to Bob, Eve wants to eavesdrop
- Bob must have a secret from Eve (otherwise Eve = Bob)
- Alice and Bob must share a “secret” not known to Eve
  - Consider Alice used “information” X other than m to create c
  - Eve has unbounded power, she can try all possible X’s
  - Yet ciphertext is uniquely decipherable, so Bob must know part of X
- Key generation algorithm \( G(1^\lambda) \rightarrow k \)
  - Security parameter \( \lambda \) is for determining the security level
- \( E_k (m) \rightarrow c, D_k (c) \rightarrow m \) // Symmetric-key encryption
(Shannon) Entropy

- $H(x)$ --- A measure of the amount of information (about $x$) that is missing before reception

- Consider a fair coin, $\Pr(x_0) = \Pr(x_1) = \frac{1}{2}$
- $-\lg(1/2) = -(\lg 1 - \lg 2) = -(0 - 1) = 1$
  - Here, $\lg$ means $\log_2$ (so $\lg 2 = 1$), not following "ISO notation"
- So, we use 1 bit to denote this event (of coin tossing)
  - with probability $\frac{1}{2}$ we gain this 1 bit of info is 0 (or 1)

- Looking ahead: for a set $Y$ of size $2^c$ (without any "compression")
- each element in $Y$ can be represented by $\lg(2^c) = c$-bit string
“Insecurity” of Substitution Ciphers

- Let's use the definition of entropy to evaluate English characters
- True prob. distribution of alphabets in English is “unknown”
- Character entropy: 4.219 according to character occurrence frequency (when all characters are independent)
- Yet, English has common bi-/tri-gram → 2.77
  - E.g., qu is a character-level bigram
- Shannon's estimation: 0.6 - 1.3 bits per character

- Frequency analysis
  - E.g., www.cryptool-online.org/?Itemid=117
- Check Chapter 1 of Stinson's book
  - www.math.uwaterloo.ca/~dstinson/books.html
XOR, One-Time Pad, and IT-Security

- XOR: For \( b_1 \oplus b_2 \) where \( b_1 \) is a bit, see the table below.
- For bit-string operation \( S_1 \oplus S_2 \), just \( \oplus \) in a bit-wise manner.

- OTP: \( G(1^\lambda) \) outputs a random \( \lambda \)-bit string as \( k \).
- \( E_k (m) \rightarrow c = m \oplus k; \)
  - \( m \) is \( \lambda \)-bit long.
- \( D_k (c) \rightarrow m = c \oplus k \)

<table>
<thead>
<tr>
<th>XOR</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- Def. of Information-Theoretic Security: \( H(m) = H(m \mid c) \)
  - R.H.S.: conditional entropy of the plaintext given the ciphertext.
  - i.e., \( \Pr(M = m \mid C = c) = \Pr(M = m) \)
OTP is Secure

- \( \Pr(M = m \mid C = c) \)
- \( = \Pr(M = m \land C = c) / \Pr(C = c) \) [Bayes’ law]
- \( = \Pr(M = m) \Pr(C = c \mid M = m) / \Pr(C = c) \) [prob. def.]
- \( = \Pr(M = m) \Pr(C = c \mid M = m) / \sum_{m' \in M} \Pr(M = m') \Pr(C = c \mid M = m') \)
- \( = \Pr(M = m) \cdot (1/2^\lambda) / \sum_{m' \in M} \{ \Pr(M = m') \cdot (1/2^\lambda) \} \)
- \( = \Pr(M = m) / 1 \)

since \( \Pr(C = c \mid M = m') = 1/2^\lambda \) for all \( c \) and \( m' \) for OTP
This Lecture

- Security against *computationally-bounded adversary*
  - also known as (a.k.a.) *probabilistic polynomial-time adversary*

- “Symmetric-Key Primitives”
- and constructing Symmetric-Key Encryption from them

- “Public-Key Primitives”
- and their number-theoretic candidate constructions
- and constructing public-key encryption from them

- Modeling security of *Public-Key Encryption*
Probabilistic Polynomial Time (PPT) Algo

- \( y = A(x) \)
- Input \( x \) is of size/length \( n \)
  - We write \( |x| = n \)
- Run-time is \( O(n^c) \): \( c \) being a constant
  - We say a PPT algorithm is an “efficient” algorithm
- Probabilistic: allows to “flip a coin” to make it randomized
- \( y \leftarrow A(x) \)
- \( y \) denotes the random variable corresponds to \( A \)’s output
- Or \( y = A(x; r) \), where \( r \) denotes \( A \)’s *internal coin tossing*
  - \( r \)’s length is also polynomial in \( n \)
A function $v(n)$ is called negligible, denoted $\text{negl}(n)$, if:

$(\forall c > 0) \ (\exists n') \ (\forall n \geq n') \ [v(n) \leq 1/n^c]$

Less than the inverse of any polynomial for large enough $n$

Prob. of breaking a secure system should be negligible in $n$

Let $\text{poly}(n)$ denote some polynomial function in $n$

We have $\text{poly}(n) \cdot \text{negl}(n) = \text{negl}(n)$
Security Parameter (& some notations)

- We want a “set” of cryptosystems parameterized by \( n \)
- Algo.’s run by all parties take commonly agreed input \( n \)
- They run in time polynomial in their input length \( n \)
- Notations:
  - \( \text{poly}(n) \): runtime of all parties are sufficiently fast, e.g., \( n^3 \)
  - \( \text{negl}(n) \): e.g., \( 1/2^n \) is in \( \text{negl}(n) \)
  - \( 1^n \) denotes \( n \) “copies” of \( 1 \)'s, i.e., \( 1^n \) is in \( \{0, 1\}^n \)
- Security parameter of the system is \( 1^n \) (with length \( n \) bits)
  - but not \( n \) (length of \( n \) is \( \log(n) \) bits)
Symmetric-Key Primitives

- Pseudorandom Generator (PRG)
- Pseudorandom Function (PRF)
- Pseudorandom Permutation (PRP)

- Pseudorandomness: its output is “computationally indistinguishable” from true randomness
- “Computational” indistinguishability
  - only holds against a computationally bounded adversary
  - distinguish: the difference in the probabilities of concluding it is a truly random function / a randomly chosen function is negligible
Pseudorandom Generator (PRG)

- **Definition:**
  - Inputs a short, truly random “seed” (or calling it a secret key)
  - Outputs a long string of “pseudorandom” bits

- **Construction:**
  - E.g., Blockcipher (e.g., AES) run in “counter mode”
    - Mode of operation: to use a fixed input-output block-cipher E() to encrypt an arbitrarily long message (with many message blocks)

- **Application:** can be used to build “Stream cipher”
  - Stream cipher”: Encrypt 1 bit at a time
  - It can be treated as a “Stateful symmetric-key encryption”
Counter Mode and PRG

- Counter mode for encrypting 4 blocks of message:
  - Here $i$, $i+1$, ... are the input of $E(k, \cdot)$

- Counter-mode construction of PRG from blockcipher $E()$: no exclusive OR and just treat $c[0]$ ... $c[n]$ as the output
  - Recall that PRG only has the key $k$ as input and there is no $m$
PRG $\Rightarrow$ Stateful SKE

- $(c, s') \leftarrow E(m; s), (m, s') \leftarrow D(c; s)$
- where $s$ is the old (secret) state and $s'$ is the new state
- Ingredient: A (non-trivial PRG) $G: \{0, 1\}^\lambda \rightarrow \{0, 1\}^{n+\lambda}$
  - from $\lambda$-bit string to $(n+\lambda)$-bit string (stretching)
  - a trivial PRG: if $\{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda$, just directly output the input
  - denote it as $G() = G_1() \circ G_2()$ where $|G_1()| = n$, and $|G_2()| = \lambda$

- To encrypt $m$ with state $s$, output $c = m \oplus G_1(s), s' = G_2(s)$
- To decrypt $c$ with state $s$, output $m = c \oplus G_1(s), s' = G_2(s)$
- Problem: encryptor and decryptor need to synchronize
Pseudorandom Func. & Permutation

- **Pseudorandom Function (PRF)** defined over \((K, X, Y)\): \(F: K \times X \rightarrow Y\)
  - \(K\) is the key space, usually just a set of bit-strings
  - “efficient” (a.k.a. PT) algorithm to evaluate \(F(k, x)\) exists

- **Pseudorandom Permutation (PRP)** defined over \((K, X)\): \(E: K \times X \rightarrow X\)
  - “efficient” deterministic algorithm to evaluate \(E(k, x)\) exist
  - \(E(k, \cdot )\) is one-to-one
  - “efficient” inversion algorithm \(D(k, y)\) exists
  - Note: it is an abstraction of “stateless” “block-cipher”
Security of PRF

- For a random key $k$
  - Note: $|k|$ bit of randomness
  - $F(k, \cdot)$ is “indistinguishable” from
  - a random function in $\text{Funs}[X, Y]$
    - the set of all functions from $X$ to $Y$
    - $|X| \cdot \log(|Y|)$ bit of randomness
    - Notation: the size of set $X$ (the input space here) is $|X|$
  - When $|X| \cdot \log(|Y|) > |k|$, how can it be possible?
  - Intuition: an adversary can only make polynomially-many queries
    - and thus it does not have enough time to infer which “world” it is in

- Secure PRF $\Rightarrow$ Secure (non-stateful) SKE
  - But we did not study the security definition of the latter yet (stay tuned)
PRG vs. PRF

- **PRG**: a single output appears random if the input was chosen at random.
- **PRF**: all its outputs appear random, regardless of how the corresponding inputs were chosen, as long as the key is randomly drawn.

Can we use PRG to build PRF?
- Suppose PRG $G()$ outputs a bit string of $n^c$ long.
- We build a PRF with key being the seed $k$ of PRG.
- and use a $c \cdot \lg(n)$ input to specify the 1 bit in $G(k)$’s output.
- i.e., Output $G(k)[x]$, where $S[i]$ denote the $i$-th bit of string $S$.
- So for a non-trivial PRF, we have $|X| = \omega(\lg(n))$.

There exists a way to use PRG to build (non-trivial) PRF [details omitted].
Key Management for Symmetric-Keys

- How many keys for the whole network if each of them establish a confidential channel for each other?

- Recall: Alice and Bob must share a secret not known to Eve because Eve is unbounded
  - What if the eavesdropper we are worrying of is bounded?
Public-Key Encryption

- KeyGen($1^\lambda$) $\rightarrow$ (sk, pk)
- $E_{pk}(m) \rightarrow c$, $D_{sk}(c) \rightarrow m$
- How many key(pair)s are needed for $n$ nodes?

- Computable: Anyone can efficiently execute $E()$
- Invertible: Alice can “invert” the function $E()$
- Easily invertible given auxiliary information sk
- Difficult to invert without sk
Public-Key Primitives: OWF, OWP, TDP

- **One-Way Functions (OWF)**
  - Easy to compute
    - Computation: Given $x$, derive $f(x)$ can be done in polynomial time
  - Difficult to invert
    - Inversion: Given $f(x)$ (and the description of $f()$ itself), deriving $x$ will take an exponential time

- **One-Way Permutations (OWP)**
  - + Must be possible to invert
    - cf. Can’t invert $f$ at -1 for $f(x) := x^2$: $\mathbb{R} \to \mathbb{R}_{\geq 0}$
    - where $\mathbb{R}$ = set of real numbers and $\mathbb{R}_{\geq 0}$ = set of non negative real numbers

- **Trap-door Permutations (TDP)**
  - + Easy to invert when given some “auxiliary info” is given
Formal Definition of OWF/P

- \( \Pr(f(z) = f(x) \mid x \in \{0, 1\}^\lambda, y \leftarrow f(x), z \leftarrow A(y, 1^\lambda)) = \text{negl}(\lambda) \)

- For a random \( x \)
- Give \( y = f(x) \) to the adversary \( A() \)
- \( A \) is supposed to return \( z \)
- Such that \( y = f(z) \)
- Question: why not just require \( x = z \)?

- If \( f \) is also one-to-one, \( f \) is called a one-way permutation
For every $\lambda$
- There exists a string $T$ (the “trapdoor”)
  - of polynomial length, $|T| \leq \text{poly}(\lambda)$
  - and a polynomial-time “inversion” algorithm $\text{Inv}()$ s.t. $\text{Inv}(f(x), T) = x$, for any $x \in \{0, 1\}^\lambda$

Wait, you told me that it is one-way, but now you told me that there is a trapdoor, so is it one-way or not?

Answer: $A()$ did not see the trapdoor $T$. 
More Details in Formalisms

- How the trapdoor is born?
- Recall that we want to study (T)OWF/P for building PKE
- We want to generate a pair of public-private key together
- Everyone use the same function?
- We need a definition for a family of OWFs
  - Analogy: PRFs is also a family, each function in the PRF family is indexed by a key, so picking a key \( \Rightarrow \) choosing a function
  - So the definition of OWF/P should be fixed to also include the random choice in picking \( f() \) from the function family

- P.S. existence of PRG \( \Leftrightarrow \) existence of OWF [details omitted]
Some Complexity Theory

- **P = NP** means
  - every problem whose solution can be quickly verified (**NP**)
  - can also be quickly solved (**P**)
- The existence of OWFs trivially implies that **P** ≠ **NP**
- (The existence of PRG also implies **P** ≠ **NP**)
- (PRG exists if and only if OWF exists)
- Yet, even if we assume **P** ≠ **NP**, we do not know how to prove that OWFs exist
- We can only say we have some “candidate” for OWF
OWF Candidate (+ Provable Security)

- Integer Multiplication vs. Factoring
- Try factoring 2021
- \( f(p, q) = p \cdot q \)
  - where \( p \) and \( q \) are \( \lambda \)-bit primes
  - In other words, domain of \( f \) is a pair of two \( \lambda \)-bit primes

- If factoring is hard, then \( f \) is provably OWF
- Under assumption \( Y \), then \( X \) is provably secure.
- The only way to “break” system \( X \) is to solve problem \( Y \)
OWP (via Modular Exponentiation)

- $f(x, p, g) = (g^x \mod p, p, g)$
- OR $f_{p, g}(x) = g^x \mod p$
- To invert, it is to solve Discrete Logarithm Problem (DLP)
- Now, some Number Theory (or Abstract Algebra)!

- [http://cacr.uwaterloo.ca/hac](http://cacr.uwaterloo.ca/hac)
- [http://www.shoup.net/ntb](http://www.shoup.net/ntb)
Groups

- A (finite) set $G$ and a “group operation” (say $*$) that is
  - closed ($\forall a, b \in G$, $a * b$ is also in $G$)
  - associative ($\forall a, b, c \in G$, $(a * b) * c = a * (b * c)$ holds)
  - equipped with identity $e$, ($\forall a \in G$, $e * a = a * e = a$ holds)
  - invertible ($\forall a \in G$, $\exists b \in G$ s.t. $a * b = b * a = e$ holds)
  - commutative for Abelian group ($\forall a, b \in G$, $a * b = b * a$)

- Examples:
  - $\mathbb{Z}$ (integers, with addition operation; infinite group)
  - $G^n$ ($G$, a group; coordinate-wise operation)
Finite Cyclic Group

- **Order of a group**
  - denoted by \(|G|\)
  - number of elements in \(G\)

- **Cyclic group**
  - (in multiplicative notation)
  - there is one element \(g\) s.t.
  - \(G = \{g^0, g^1, g^2, ..., g^{|G|-1}\}\)

- E.g. \(\mathbb{Z}_N\)
  - consider \(g = 1\), or any \(g\) s.t. \(\gcd(g, N) = 1\)
Multiplicative Group of $\mathbb{Z}_N$

- $\mathbb{Z}_N^*$: set of elements from $\mathbb{Z}_N$ that are \textit{relatively prime} to $N$
  - (just notation “clash”, not to be confused with the operation $*$)
  - $a$ and $b$ are relatively prime if $\gcd(a, b) = 1$
- For any $N$, 0 is not in $\mathbb{Z}_N^*$
- For any prime $p$, $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\}$
Fermat’s Little Theorem

- Order of \(a\) (in \(\mathbb{Z}_p^*\)) is the smallest \(x\) s.t. \(a^x = 1 \mod p\)

- FLT: For any prime \(p\), and \(x\) in \(\mathbb{Z}_p^*\), \(x^{p-1} = 1 \mod p\)

- All elements’ orders are \((p - 1)\)?

- E.g.: Consider \(p > 3\) and \((p - 1)^2\)

- Exercise: show \(g^x \mod p\) is a OWP candidate
  - (Do not bother w/ one-wayness first, prove that it is a permutation)
RSA (Rivest-Shamir-Adleman 1978)

- Most widely accepted & implemented approach to PKE
- "Block cipher" where $0 \leq m, c \leq N - 1$ for some $N$
RSA (Some Mathematics Behind)

- $f_{e,N}(x) = x^e \mod N$, where $N = p \times q$, $x \in \mathbb{Z}_{N^*}$ and $e \in \mathbb{Z}_{\Phi(N)^*}$

- Euler phi-function: For any positive integer $m$, $\Phi(m)$ is the number of +ve int. $< m$ that are relatively prime to $m$.
  - Exercise 1: What is $|\mathbb{Z}_{m^*}|$?
  - Exercise 2: What is $\Phi(N)$ where $N = p \times q$, $p$ and $q$ are prime?

- Euler’s Theorem: For any int. $m$, & $x \in \mathbb{Z}_{m^*}$, $x^{\Phi(m)} \equiv 1 \mod m$

- So, what is the trapdoor and the corresponding Inv. algo.?
- Extended Euclid Algorithm for finding G.C.D. [details later]
Security Requirement of Encryption

- Without the “secret key”, the ciphertext is not “useful”
- What is the “secret key” is “useful” for?
  - Decryption
- Refined: Without the “secret key”, the ciphertext is not “useful” for decryption, i.e., “getting back” the plaintext
- Security requirement is related to an attack goal
  - getting back the plaintext
  - every part of it
  - i.e., attack against one-wayness
- Let’s study the security of PKE first
  - Why not SKE? Actually PKE security definition is “simpler”
One-Wayness for Public-Key Encryption

- **OWF:** \( \Pr(f(z) = f(x) \mid x \in \{0, 1\}^\lambda; y \leftarrow f(x); z \leftarrow A(y, 1^\lambda)) = \text{negl}(\lambda) \)

- The above def. is just 1 OWF
- It does not allow us to choose 1 OWF among a collection of OWFs

- More formally, a collection of OWFs is defined by:
  - \( \Pr(f_{pk}(z) = y \mid pk \leftarrow G(1^\lambda); x \leftarrow D(pk); y = f_{PK}(x); z \leftarrow A(y, pk, 1^\lambda)) \)
  - \( pk \) is the public system parameter, \( D(pk) \) is the domain determined by \( pk \)

- **Exercise:** Define a family of TDP

- **OW for PKE:** \( \Pr(m' = m \mid (pk, sk) \leftarrow G(1^\lambda); m \leftarrow M, c \leftarrow E_{pk}(m); m' \leftarrow A(c, pk, 1^\lambda)) = \text{negl}(\lambda) \)
One-Wayness, depicted

- A random plaintext is chosen by the challenger
- Challenge: a ciphertext encrypting it
- Adversary’s goal: recover the plaintext in full

Here is my public key, I would not tell you my private key
And I encrypt a random message, tell me what it is
Basic Principles in Cryptography

- What if we have a quantum computer which can factor number larger than 15 with accuracy higher than 50%?
- Use another candidate of TDP (to instantiate, say, PKE)
- Abstraction/Distillation of properties we want
  - “One-wayness” in TDP
- Is this kind of abstraction always the best?
- We may want more property, say, \( f(a) \ast f(b) = f(a + b) \)
- Well, we can make a “more specific generalization”
Is that OW-PKE good enough?

- Consider \( f'(x_1, x_2) = (f(x_1), x_2) \), is \( f' \) a TDP if \( f \) is?
  - Exercise: formally prove that \( f' \) is a TDP

- OW-PKE inherently leaks some information of \( m \).

- Have we proven anything if \( m \) is not random?
  - Is the message we want to protect always random?

- How about defining \( f(x) \) s.t. it hides all information of \( x \)?
  - Not really the right way... it becomes a chicken and egg situation
Two plaintexts are chosen by adversary
- Challenge: a ctxt. encrypting either 1 of them
- Adversary’s goal: outputs a single bit for distinguishing between 2 possibilities
- This game is also called **Chosen-Plaintext Attack (CPA)**
What is semantic security modeling?

- Two plaintexts are chosen by adversary
- The attacker knows and can even influence what are to be encrypted

- Adversary’s goal: outputs a single bit.
- i.e., to win this game (to break the security), the attacker only needs to learn 1 bit.
- Security means not even 1 bit of information is leaked!
Stay tuned

- Encryption
  - Key-Exchange
  - Decisional Diffie-Hellman Assumption
  - ElGamal Encryption

- Authentication
  - Hash Function
  - Message Authentication Code
  - Signatures