Cryptography

Lecture 05
Where do we stand?

- We defined the notion of perfect secrecy
- We proved that the one-time pad achieves it!
- We proved that the one-time pad is optimal!
  - i.e. we cannot improve the key length
- We saw other drawbacks of perfect secrecy
- If we want to do better we need to relax the definition
  - But in a meaningful way...
Perfect secrecy

- Requires that *absolutely no information* about the plaintext is leaked, even to eavesdroppers *with unlimited computational power*
  - Seems unnecessarily strong
Computational secrecy

- Would be ok if a scheme leaked information *with tiny probability* to eavesdroppers *with bounded computational resources*

- i.e. we can relax perfect secrecy by
  - Allowing security to “fail” with tiny probability
  - Restricting attention to “efficient” attackers
Tiny probability of failure

- Say security fails with probability $2^{-60}$
  - Should we be concerned about this
  - With probability $> 2^{-60}$, the sender and receiver will both be struck by lightning in the next year...
  - Something that occurs with probability $2^{-60}$/sec is expected to occur once every 100 billion years
Bounded attackers?

- Consider brute-force search of key space; assume one key can be tested per clock cycle

- Desktop computer: \( \approx 2^{57} \) keys/year

- Supercomputer: \( \approx 2^{80} \) keys/year

- Supercomputer since Big Bang: \( \approx 2^{112} \) keys
  - Restricting attention to attackers who can try \( 2^{112} \) keys is fine!

- Modern key space: \( 2^{128} \) keys or more . . .
Roadmap

- We will give an alternate (but equivalent) definition of perfect secrecy
  - Using a randomized experiment
- That definition has a natural relaxation
- **Warning:** the material gets much more difficult now
Perfect indistinguishability

- π = (Gen, Enc, Dec), message space \( M \)

- Informally:
  - Two messages \( m_0, m_1 \): one is chosen and encrypted (using unknown \( k \)) to give \( c \leftarrow Enc_k(m_b) \)
  - Adversary \( A \) is given \( c \) and tries to determine which message was encrypted
  - \( π \) is perfectly indistinguishable if no \( A \) can guess correctly with probability any better than \( \frac{1}{2} \)
Perfect indistinguishability

- Let $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ be an encryption scheme with message space $\mathcal{M}$, and $A$ an adversary.

- Define a randomized experiment $\text{PrivK}_{A,\Pi}$:
  1. $A$ outputs $m_0, m_1 \in \mathcal{M}$
  2. $k \leftarrow \text{Gen}$, $b \leftarrow \{0, 1\}$, $c \leftarrow \text{Enc}_k(m_b)$
  3. $b' \leftarrow A(c)$

- Adversary $A$ succeeds if $b = b'$, and we say the experiment evaluates to 1 in this case.
Perfect indistinguishability

- Easy to succeed with probability $\frac{1}{2}$

- $\Pi$ is *perfectly indistinguishable* if for all attackers (algorithms) $A$, it holds that

  $$Pr[PrivK_{A,\Pi} = 1] = \frac{1}{2}$$
Perfect indistinguishability

- Claim: $\Pi$ is perfectly indistinguishable $\iff$ $\Pi$ is perfectly secret
- i.e. perfect indistinguishability is just an alternate definition of perfect secrecy
Computational secrecy?

- Idea: relax perfect indistinguishability

- Two approaches
  - Concrete security
  - Asymptotic security
Computational indistinguishability (concrete)

- $(t, \epsilon)$-indistinguishability
  - Security may fail with probability $\leq \epsilon$
  - Restrict attention to attackers running in time $\leq t$
Computational indistinguishability (concrete version)

- $\Pi$ is $(t, \epsilon)$-indistinguishable if for all attackers $A$ running in time at most $t$, it holds that

$$Pr[\text{PrivK}_{A, \Pi} = 1] \leq \frac{1}{2} + \epsilon$$
Concrete security

- Parameters $t, \epsilon$ are what we ultimately care about in the real world

- Does not lead to a clean theory
  - Sensitive to exact computational model
  - $\Pi$ can be $(t, \epsilon)$-secure for many choices of $t, \epsilon$

- Would like to have schemes where users can adjust the achieved security as desired
Asymptotic security

- Introduce *security parameter* $n$
  - For now, can view as the key length
  - Fixed by honest parties at initialization
    - Allows users to tailor the security level
  - Known by adversary

- Measure running times of all parties, and the success probability of the adversary, as functions of $n$
Computational indistinguishability (asymptotic)

- Computational indistinguishability
  - Security may fail with probability \textit{negligible in } n
  - Restrict attention to attackers running in time (at most) \textit{polynomial in } n
Definitions

- A function $f : \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$ is (at most) polynomial if there exists $c$ such that $f(n) < n^c$ for large enough $n$

- A function $f : \mathbb{Z}^+ \rightarrow [0, 1]$ is negligible if for every polynomial $p$ is holds that $f(n) < \frac{1}{p(n)}$ for large enough $n$
  - Typical example: $f(n) = poly(n) \cdot 2^{-cn}$
Why these choices?

- Somewhat arbitrary

- “Efficient” = “(probabilistic) polynomial-time (PPT)”
  borrowed from complexity theory

- Convenient closure properties
  - poly * poly = poly
    - Poly-many calls to PPT subroutine (with poly-size inputs) is PPT
  - poly * negligible = negligible
    - Poly-many calls to subroutine that fails with negligible probability fails with negligible probability overall
(Re)defining encryption

- A *private-key encryption scheme* is defined by three PPT algorithms (Gen, Enc, Dec):
  - Gen: takes as input $1^n$; outputs $k$. (assumed $|k| \geq n$
  - Enc: takes as input a key $k$ and a message $m \in \{0, 1\}^x$; outputs ciphertext $c$
    
    $$c \leftarrow Enc_k(m)$$

  - Dec: takes key $k$ and ciphertext $c$ as input; outputs a message $m$ or “error”
Computational indistinguishability (asymptotic version)

- Fix $\Pi$, $A$

- Define a randomized exp't $PrivK_{A,\Pi}(n)$:
  1. $A(1^n)$ outputs $m_0, m_1 \in \{0, 1\}^x$ of equal length
  2. $k \leftarrow \text{Gen}(1^n)$, $b \leftarrow \{0, 1\}$, $c \leftarrow \text{Enc}_k(m_b)$
  3. $b' \leftarrow A(c)$

- Adversary $A$ succeeds if $b = b'$, and we say the experiment evaluates to 1 in this case
Computational indistinguishability (asymptotic version)

- $\Pi$ is *computationally indistinguishable* (aka *EAV-secure*) if for all PPT attackers $A$, there is a negligible function $\epsilon$ such that

$$
\Pr[\text{PrivK}_{A,\Pi}(n) = 1] \leq \frac{1}{2} + \epsilon(n)
$$
Consider a scheme where the best attack is brute-force search over the key space, and \( Gen(1^n) \) generates a uniform \( n \)-bit key.

So if \( A \) runs in time \( t(n) \), then

\[
Pr[\text{Priv}_{K_A,\Pi}(n) = 1] < \frac{1}{2} + O\left(\frac{t(n)}{2^n}\right)
\]

This scheme is EAV-secure: for any polynomial \( t \), the function \( \frac{t(n)}{2^n} \) is negligible.
Example

Consider a scheme and a particular attacker $A$ that runs for $n^3$ minutes and breaks the scheme with probability $2^{40}2^{-n}$

- This does not contradict asymptotic security

- What about real-world security (against this attacker)?
  - $n=40$: A breaks scheme with prob. 1 in 6 weeks
  - $n = 50$: A breaks scheme with prob. $\frac{1}{1000}$ in 3 months
  - $n = 500$: A breaks scheme with prob. $2^{-500}$ in 200 years
Example

- What happens when computers get faster?

- e.g. consider a scheme that takes time $n^2$ to run but time $2^n$ to break with prob 1

- What if computers get 4x faster?
  - Honest users double $n$; maintain same running time
  - Attacker’s work is (roughly) squared!
Encryption and plaintext length

- In practice, we want encryption schemes that can encrypt arbitrary-length messages
- In general, encryption does not hide the plaintext length
  - The definition takes this into account by requiring $m_0, m_1$ to have the same length
- But beware that leaking plaintext length can often lead to problems in the real world!
  - Obvious examples...
  - Database searches
  - Encrypting compressed data
From now on, we will assume the computational setting by default.

- Usually, the *asymptotic* setting.
Pseudorandomness
Pseudorandomness

- Important building block for computationally secure encryption
- Important concept in cryptography
What does “random” mean?

▶ What does “uniform” mean?

▶ Which of the following is a uniform string?
  ▶ 0101010110010101
  ▶ 001011011100110
  ▶ 0000000000000000

▶ If we generate a uniform 16-bit string, each of the above occurs with probability $2^{-16}$
What does “uniform ” mean?

▶ “Uniformity” is not a property of a string, but a property of a distribution

▶ A distribution on $n$-bit strings is a function $D : \{0, 1\}^n \to [0, 1]$ such that $\sum_x D(x) = 1$
  
  ▶ The uniform distribution on $n$-bit strings, denoted $U_n$, assigns probability $2^{-n}$ to every $x \in \{0, 1\}^n$
What does “pseudorandom” mean?

- Informal: cannot be distinguished from uniform (i.e. random)

- Which of the following is pseudorandom?
  - 0101010101010101
  - 0010111011100110
  - 0000000000000000

- Pseudorandomness is a property of a *distribution*, not a *string*
Pseudorandomness (take 1)

- Fix some distribution $D$ on $n$-bit strings
  - $x \leftarrow D$ means “sample $x$ according to $D$”

- Historically, $D$ was considered pseudorandom if it “passed a bunch of statistical tests”
  - $\Pr_{x \leftarrow D}[\text{1st bit of } x \text{ is } 1] \approx \frac{1}{2}$
  - $\Pr_{x \leftarrow D}[\text{parity of } x \text{ is } 1] \approx \frac{1}{2}$
  - $\Pr_{x \leftarrow D}[A_i(x) = 1] \approx \Pr_{x \leftarrow U_n}[A_i(x) = 1]$ for $i = 1, \ldots, 20$
Pseudorandomness (take 2)

- This is not sufficient in an adversarial setting!
  - Who knows what statistical test an attacker will use?

- Cryptographic def’n of pseudorandomness:
  - $D$ is pseudorandom if it passes all *efficient* statistical tests
Let $D$ be a distribution on $p$-bit strings.

$D$ is $(t, \epsilon)$-pseudorandom if for all $A$ running in time at most $t$,

$$|Pr_{x \leftarrow D}[A(x) = 1] - Pr_{x \leftarrow U_p}[A(x) = 1]| \leq \epsilon$$
Pseudorandomness (asymptotic)

- Security parameter $n$, polynomial $p$
- Let $D_n$ be a distribution over $p(n)$-bit strings
- Pseudorandomness is a property of a sequence of distributions $\{D_n\} = \{D_1, D_2, \ldots\}$
Pseudorandomness (asymptotic)

- \( \{D_n\} \) is pseudorandom if for all probabilistic, polynomial-time distinguishers \( A \), there is a negligible function \( \epsilon \) such that

\[
|\Pr_{x \leftarrow D_n}[A(x) = 1] - \Pr_{x \leftarrow U_p(n)}[A(x) = 1]| \leq \epsilon(n)
\]
Pseudorandom generators (PRGs)

- A PRG is an efficient, deterministic algorithm that expands a short, uniform seed into a longer, pseudorandom output.
  - Useful whenever you have a “small” number of true random bits, and want lots of “random-looking” bits.
Let $G$ be a deterministic, poly-time algorithm that is expanding, i.e. $|G(x)| = p(|x|) > |x|$
PRGs

Let $G$ be a deterministic, poly-time algorithm that is expanding, i.e. $|G(x)| = p(|x|) > |x|$

$G$ defines a sequence of distributions!

- $D_n =$ the distribution on $p(n)$-bit strings defined by choosing $x \leftarrow U_n$ and outputting $G(x)$

$$Pr_{D_n}[y] = Pr_{U_n}[G(X) = y] = \sum_{x : G(x) = y} Pr_{U_n}[x]$$

$$= \sum_{x : G(x) = y} 2^{-n}$$

$$= \frac{|\{x : G(x) = y\}|}{2^n}$$

- Note that most $y$ occur with probability 0