Recap

- Random Oracle Model
  - Pros (Easier Proofs/More Efficient Protocols/Solid Evidence for Security in Practice)
  - Cons (Strong Assumption)
- Hashing Applications
- Block Ciphers, SPNs, Feistel Networks, DES
- Meet in the Middle, 3DES
- Building Stream Ciphers
  - Linear Feedback Shift Registers (+ Attacks)
  - RC4 (+ Attacks)
  - Trivium

DES Security

- Best Known attack is brute-force 2^{56}
  - Except under unrealistic conditions (e.g., 2^{43} known plaintexts)
- Brute force is not too difficult on modern hardware
- Attack can be accelerated further after precomputation
  - Output is a few terabytes
  - Subsequently keys are cracked in 2^{38} DES evaluations (minutes)
- Precomputation costs amortize over number of DES keys cracked
- Even in 1970 there were objections to the short key length for DES
- How could we increase key-length?

Double DES

- Let $F_k(x)$ denote the DES block cipher

- A new block cipher F' with a key $k = (k_1, k_2)$ of length 2n can be defined by

$$F'_k(x) = F_{k_2}\left(F_{k_1}(x)\right)$$

- Can you think of an attack better than brute-force?

Meet in the Middle Attack

$$F'_k(x) = F_{k_2}\left(F_{k_1}(x)\right)$$

Goal: try to find secret key k in time and space $O(n2^n)$ given known plaintext/ciphertext pair(s) (x, $c = F'_k(x)$).

- **Solution?**
  - **Key Observation**

$$F_{k_1}(x) = F_{k_2}^{-1}(\mathrm{c})$$

  - **Compute $F_K^{-1}(\mathrm{c})$ and $F_K(x)$ for each potential n-bit key K and store $\left(\boldsymbol{K}, F_K^{-1}(\mathrm{c})\right)$ and $\left(\boldsymbol{K}, F_K\ (\mathrm{x})\right)$**
  - **Sort each list of pairs (by $F_K^{-1}(\mathrm{c})$ or $F_K\ (\mathrm{x})$) to find K_1 and K_2.**

Triple DES Variant 1

- Let $F_k(x)$ denote the DES block cipher

- A new block cipher F' with a key $k = (k_1, k_2, k_3\ )$ of length 2n can be defined by

$$F'_k(x) = F_{k_3}\left(F_{k_2}^{-1}\left(F_{k_1}(x)\right)\right)$$

- Meet-in-the-Middle Attack Requires time $\Omega(2^{2n})$ and space $\Omega(2^{2n})$

Triple DES Variant 1

Allows backward compatibility with DES by setting $k_1=k_2=k_3$

- Let $F_k(x)$ denote the DES block cipher

- A new block cipher F' with a key $k = (k_1, k_2, k_3)$ of length 3n can be defined by

$$F'_k(x) = F_{k_3}\left(F_{k_2}^{-1}\left(F_{k_1}(x)\right)\right)$$

- Meet-in-the-Middle Attack Requires time $\Omega(2^{2n})$ and space $\Omega(2^{2n})$

Triple DES Variant 2

Just two keys!

- Let $F_k(x)$ denote the DES block cipher
- A new block cipher F' with a key $k = (k_1, k_2)$ of length 2n can be defined by

$$F'_k(x) = F_{k_1}\left(F_{k_2}^{-1}\left(F_{k_1}(x)\right)\right)$$

- Meet-in-the-Middle Attack still requires time $\Omega(2^{2n})$ and space $\Omega(2^{2n})$
- Key length is still just 112 bits (NIST recommends 128+ bits)

Triple DES Variant 1

$$F'_k(x) = F_{k_3}\left(F_{k_2}^{-1}\left(F_{k_1}(x)\right)\right)$$

- Standardized in 1999
- Still widely used, but it is relatively slow (three block cipher operations)
- Current gold standard: AES

Stream Cipher vs PRG

- PRG pseudorandom bits output all at once

- Stream Cipher
  - Pseudorandom bits can be output as a stream
  - RC4, RC5 (Ron's Code)

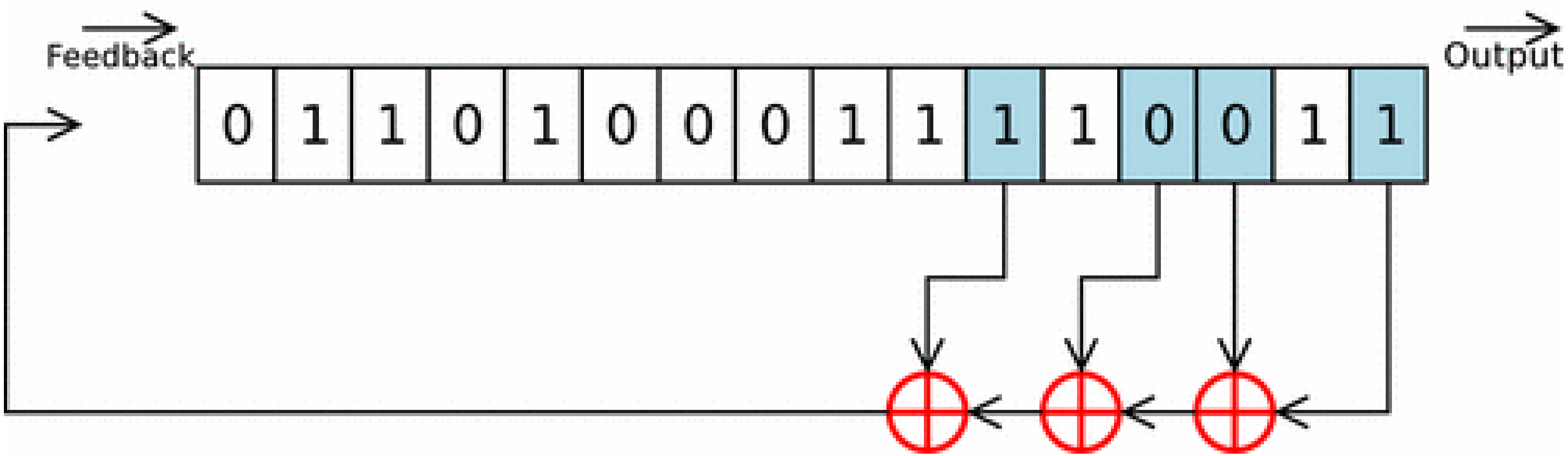
st_0 := Init(s)

For i=1 to ℓ:

    (y_i, st_i):=GetBits(st_{i-1})

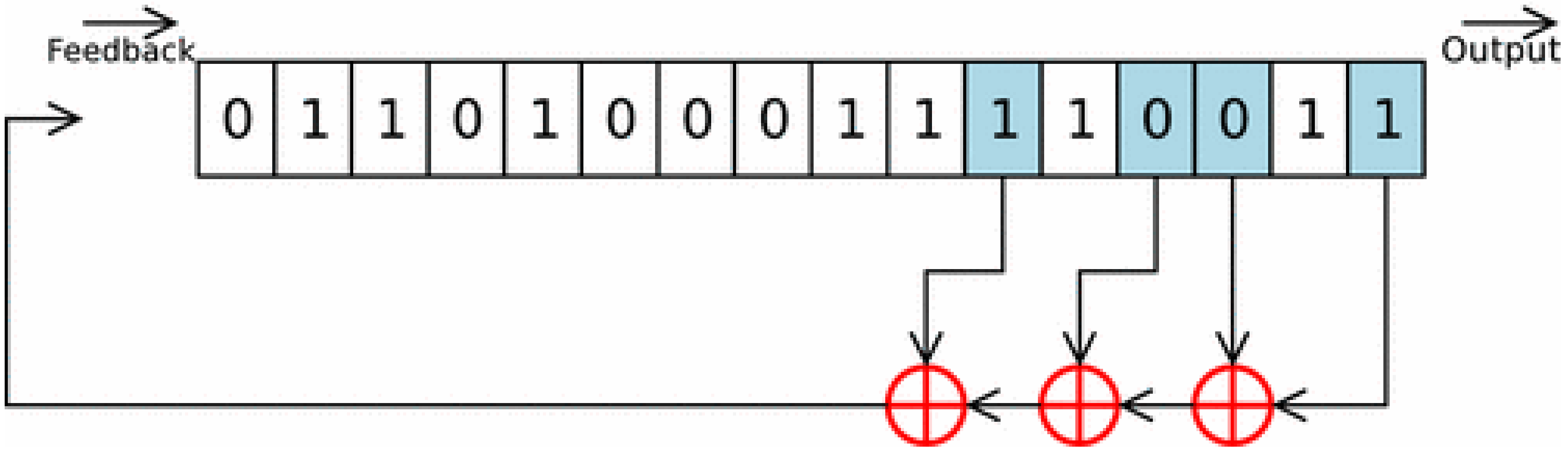
Output: $y_1,\ldots,y_\ell$

Linear Feedback Shift Register



Linear Feedback Shift Register

- State at time t: $s_{n-1}^t, \ldots, s_1^t, s_0^t$ (n registers)
- Feedback Coefficients: $\mathbf{S} \subseteq \{0, \ldots, n\}$

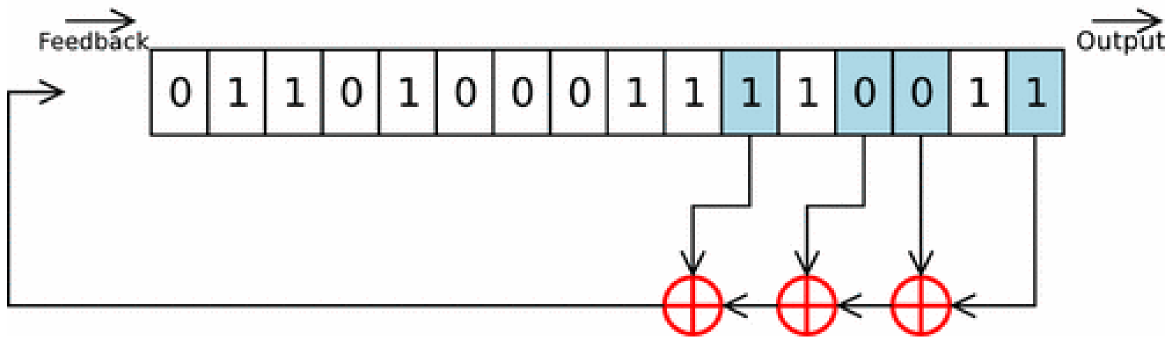


Linear Feedback Shift Register

- State at time t: $s_{n-1}^t, \ldots, s_1^t, s_0^t$ (n registers)
- Feedback Coefficients: $\mathbf{S} \subseteq \{0, \ldots, n-1\}$
- **State at time t+1:** $\bigoplus_{i \in S} s_i^t, s_{n-1}^t, \ldots, s_1^t,$

$$s_{n-1}^{t+1} = \bigoplus_{i \in S} s_i^t, \qquad \text{and} \qquad s_i^{t+1} = s_{i+1}^t \text{ for } i < n-1$$

Output at time t+1: $\boldsymbol{y_{t+1} = s_0^t}$



Linear Feedback Shift Register

- **Observation 1**: First n bits of output reveal initial state

$$y_1, \dots, y_n = s_0^0, s_1^0, \dots, s_{n-1}^0$$

- **Observation 2**: Next n bits allow us to solve for n unknowns

$$x_i = \begin{cases} 1 & \text{if } i \in S \\ 0 & otherwise \end{cases}$$

$$y_{n+1} = y_n x_{n-1} + \cdots + y_1 x_0$$

Linear Feedback Shift Register

- **Observation 1**: First n bits of output reveal initial state

$$y_1, \ldots, y_n = s_0^0, s_1^0, \ldots, s_{n-1}^0$$

- **Observation 2**: Next n bits allow us to solve for n unknowns

$$x_i = \begin{cases} 1 & \text{if } i \in S \\ 0 & otherwise \end{cases}$$

$$y_{n+1} = y_n x_{n-1} + \cdots + y_1 x_0 \text{ mod } 2$$

Linear Feedback Shift Register

- **Observation 2**: Next n bits allow us to solve for n unknowns

$$x_i = \begin{cases} 1 & \text{if } i \in S \\ 0 & otherwise \end{cases}$$

$$y_{n+1} = y_n x_{n-1} + \cdots + y_1 x_0 \text{ mod } 2$$

$$\vdots$$

$$y_{2n} = y_{2n-1} x_{n-1} + \cdots + y_n x_0 \text{ mod } 2$$

N unknowns &
N linear independent constraints

Removing Linearity

- Attacks exploited linear relationship between state and output bits

- **Nonlinear Feedback:**

$$\cancel{s_{n-1}^{t+1} = \bigoplus_{i \in S} s_i^t ,}$$

Non linear function

$$s_{n-1}^{t+1} = g(s_0^t, s_1^t, \ldots, s_{n-1}^t)$$

Removing Linearity

- Attacks exploited linear relationship between state and output bits

- **Nonlinear Combination:**

$$\cancel{\boldsymbol{y_{t+1} = s_0^t}}$$

Non linear function

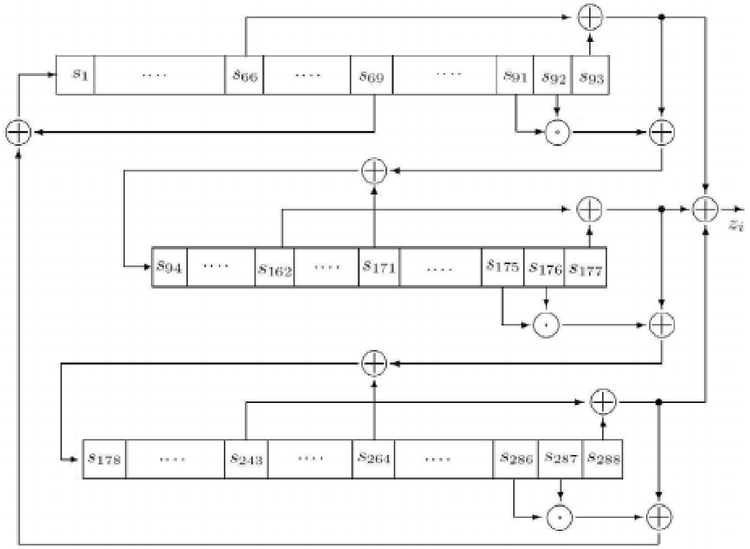
$$y_{t+1} = f(s_0^t, s_1^t, \dots, s_{n-1}^t)$$

- **Important**: f must be balanced!

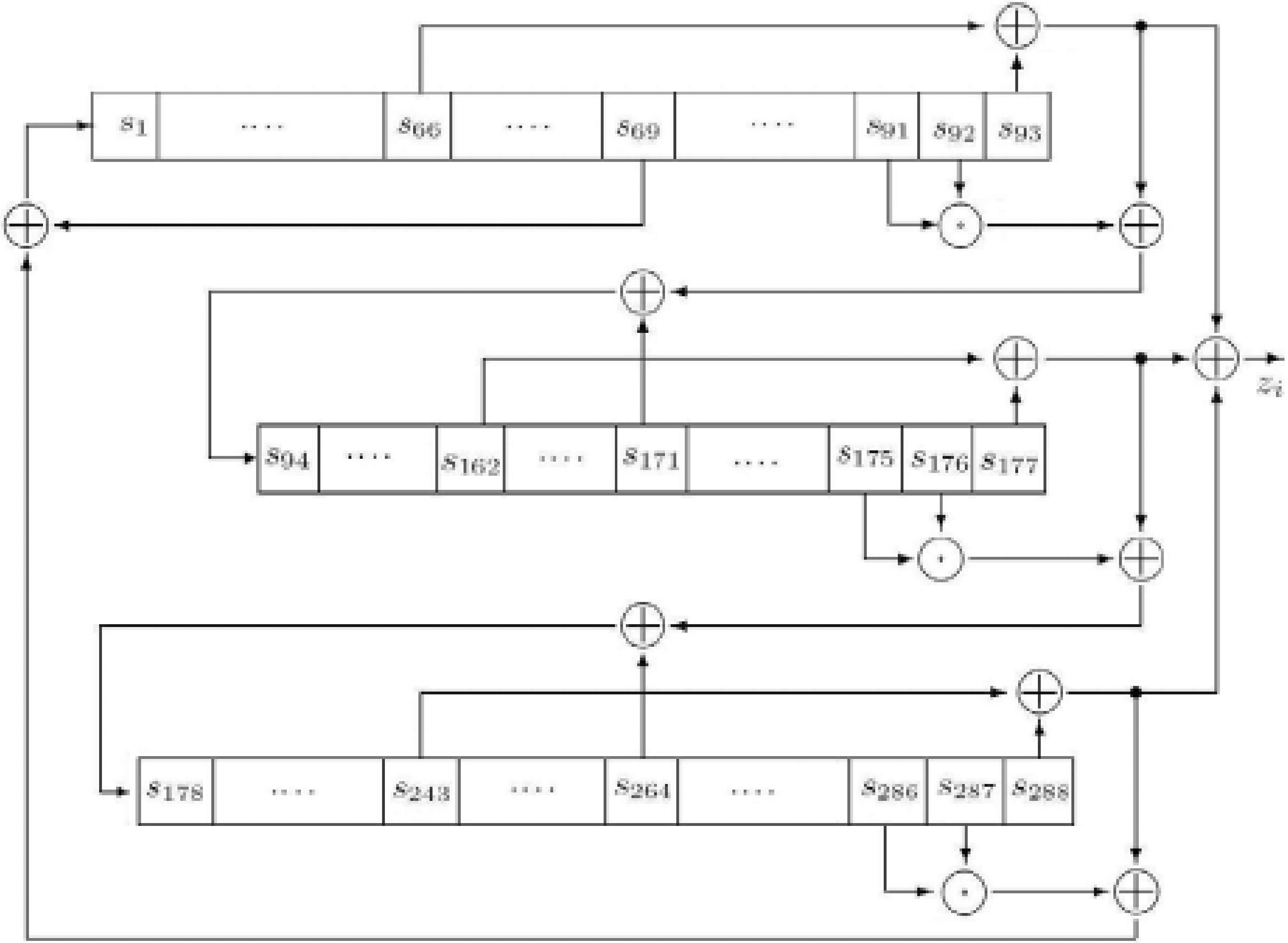
$$\Pr[f(x) = 1] \approx \frac{1}{2}$$

Trivium (2008)

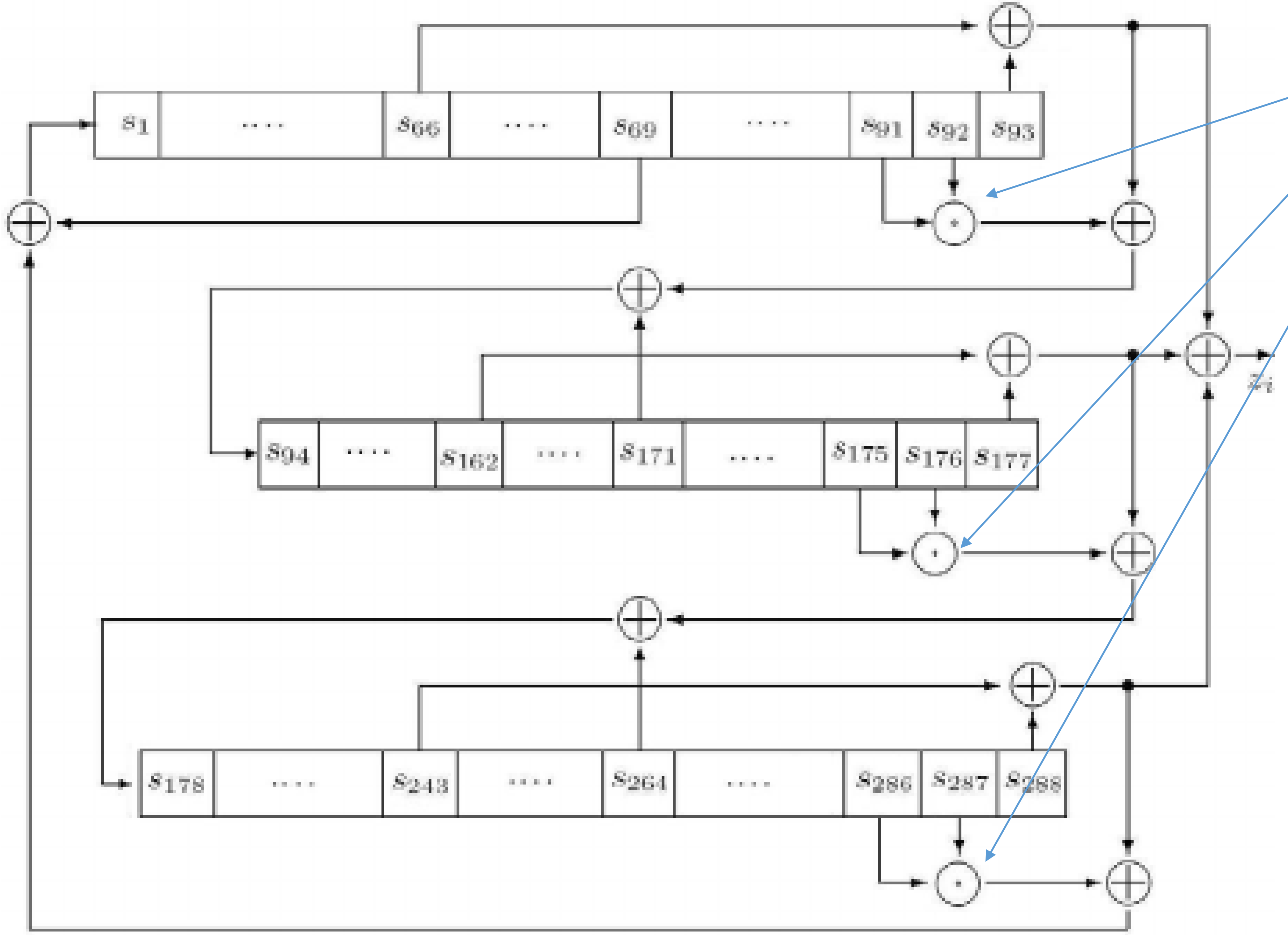
- Won the eSTREAM competition
- Currently, no known attacks are better than brute force
- Couples Output from three nonlinear Feedback Shift Registers
- First 4*288 “output bits” are discared



Trivium (2008)

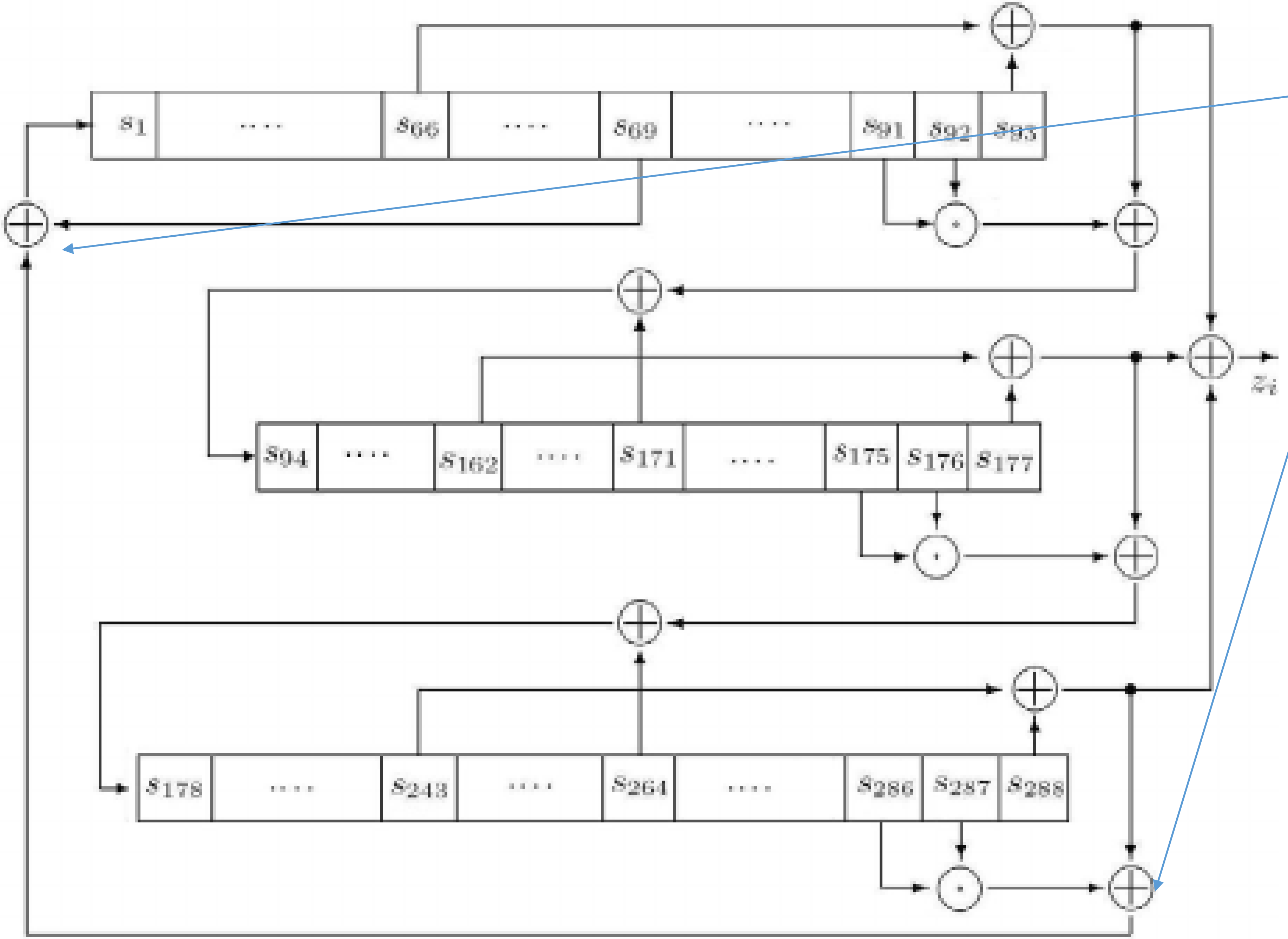


Trivium (2008)



AND (Non-linear)

Trivium (2008)



(Non-linear) Feedback

Combination Generator

- Attacks exploited linear relationship between state and output bits

- **Nonlinear Combination:**

Non linear function

$$\cancel{\boldsymbol{y_{t+1} = s_0^t}}$$

$$y_{t+1} = f(s_0^t, s_1^t, \dots, s_{n-1}^t)$$

- **Important**: f must be balanced!

$$\Pr[f(x) = 1] \approx \frac{1}{2}$$

Feedback Shift Registers

- Good performance in hardware
- Performance is less ideal for software

Cryptography
CS 555

Week 7:

- Hash Functions from Block Ciphers
- Block Ciphers, AES
- Stream Ciphers
- One Way Functions
- **Readings:** Katz and Lindell Chapter 6.2.5, 6.3, 7.1-7.4

CS 555: Week 7: Topic 1
Block Ciphers (Continued)

Hash Functions from Block Ciphers

- Davies-Meyer Construction from block cipher F_K

$$H(K, x) = F_K(x)$$

Theorem: If $F: \{0,1\}^\lambda \times \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$ is modeled as an ideal block cipher then Davies-Meyer construction is a collision-resistant hash function

(**Concrete:** Need roughly $\mathrm{q} \approx 2^{\lambda/2}$ queries to find collision)

- **Ideal Cipher Model:** For each key K model F_K as a truly random permutation which may only be accessed in black box manner.
  - (Equivalent to Random Oracle Model)

Advanced Encryption Standard (AES)

- (1997) US National Institute of Standards and Technology (NIST) announces competition for new block cipher to replace DES
- Fifteen algorithms were submitted from all over the world
  - Analyzed by NIST
- Contestants given a chance to break competitors schemes
- October, 2000 NIST announces a winner Rijndael
  - Vincent Rijmen and Joan Daemen
  - No serious vulnerabilities found in four other finalists
  - Rijndael was selected for efficiency, hardware performance, flexibility etc…

Advanced Encryption Standard

- **Block Size:** 128 bits (viewed as 4x4 byte array)
- **Key Size:** 128, 192 or 256

- Essentially a Substitution Permutation Network
  - **AddRoundKey:** Generate 128-bit sub-key from master key XOR with current state
  - **SubBytes:** Each byte of state array (16 bytes) is replaced by another byte according a a single S-box (lookup table)
  - **ShiftRows –** shift ith row by i bytes
  - **MixColumns –** permute the bits in each column

Substitution Permutation Networks

- S-box a public "substitution function" (e.g. $S \in \mathbf{Perm_8}$).

- S is not part of a secret key, but can be used with one

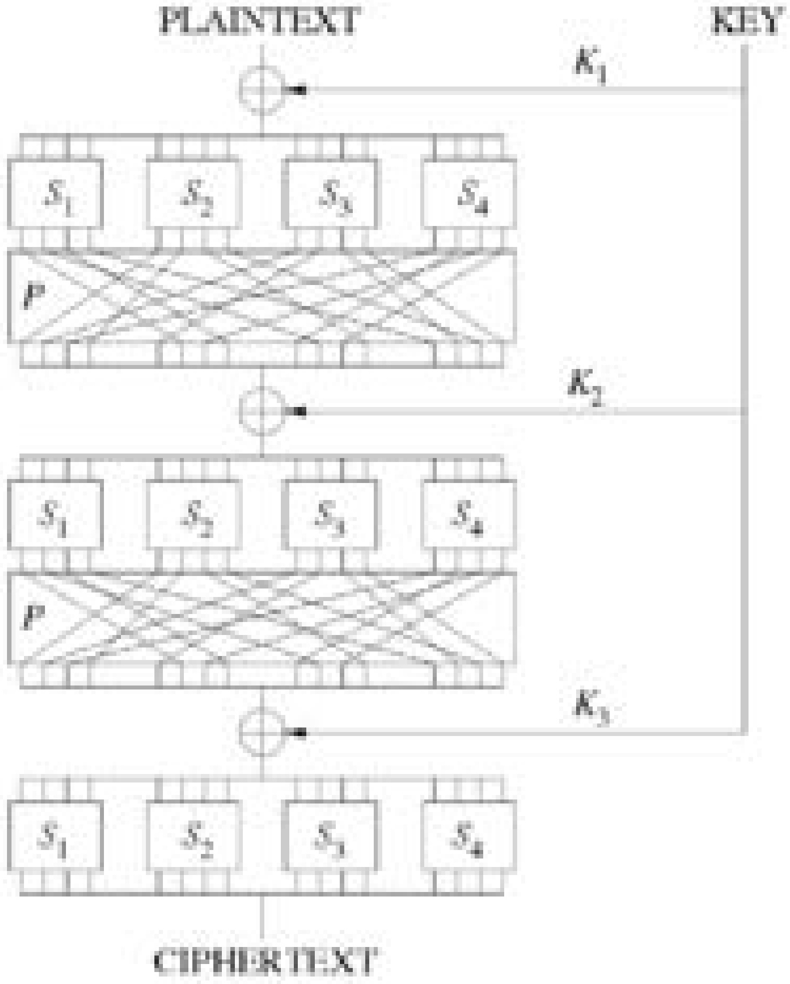
$$f(x) = S(x \oplus k)$$

Note: there are only n! possible bit mixing permutations of [n] as opposed to 2^n! Permutations of $\{0,1\}^n$

Input to round: x, k (k is subkey for current round)

1. **Key Mixing**: Set $x := x \oplus k$
2. **Substitution**: $x := S_1(x_1) \parallel S_2(x_2) \parallel \cdots \parallel S_8(x_8)$
3. **Bit Mixing Permutation**: permute the bits of x to obtain the round output

Substitution Permutation Networks



- **Proposition 6.3:** Let F be a keyed function defined by a Substitution Permutation Network. Then for any keys/number of rounds F_k is a permutation.

- Why? Composing permutations f,g results in another permutation h(x)=g(f(x)).

Advanced Encryption Standard

- Block Size: 128 bits
- Key Size: 128, 192 or 256

Key Mixing

- Essentially a Substitution Permutation Network
  - **AddRoundKey:** Generate 128-bit sub-key from master key, XOR with current state array
  - **SubBytes:** Each byte of state array (16 bytes) is replaced by another byte according a single S-box (lookup table)
  - **ShiftRows**
  - **MixColumns**

Permutation

Substitution

AddRoundKey:

Round Key (16 Bytes)

00001111			
10100011	…		
11001100		…	
01111111			…

$\oplus$

State

11110000			
01100010	…		
00110000		…	
11111111			…

=

11111111			
11000001	…		
11111100		…	
10000000			…

AddRoundKey:

Round Key (16 Bytes)

10100011	…		
		…	
			…

State

11111111			
11000001	…		
11111100		…	
10000000			…

SubBytes (Apply S-box)

S(11111111)			
S(11000001)	S(…)		
S(11111100)		S(…)	
S(10000000)			S(…)

AddRoundKey:

Round Key (16 Bytes)

10100011	…		
		…	
			…

State

S(11111111)			
S(11000001)	S(…)		
S(11111100)		S(…)	
S(10000000)			S(…)

Shift Rows

S(11111111)			
	S(11000001)	S(…)	
S(…)		**S(11111100)**	
		S(…)	**S(10000000)**

AddRoundKey:

Round Key (16 Bytes)

10100011	…		
		…	
			…

State

S(11111111)			
	S(11000001)	S(…)	
S(…)		**S(11111100)**	
		S(…)	**S(10000000)**

Mix Columns

Invertible (linear) transformation.

Key property: if inputs differ in b>0 bytes then output differs in 5-b bytes (minimum)

AES

- We just described one round of the SPN

- AES uses
  - 10 rounds (with 128 bit key)
  - 12 rounds (with 192 bit key)
  - 14 rounds (with 256 bit key)

AES Attacks?

- Side channel attacks affect a few specific implementations
  - But, this is not a weakness of AES itself
  - Timing attack on OpenSSL's implementation AES encryption (2005, Bernstein)

- (2009) Related-Key Attack on 11 round version of AES
  - Related Key Attack: Attacker convinces Alice to use two related (but unknown) keys
  - recovers 256-bit key in time 2^{70}
  - But AES is 14 round (with 256 bit key) so the attack doesn't apply in practice
- (2009) Related Key Attack on 192-bit and 256 bit version of AES
  - recovers 256-bit key in time $2^{99.5}$.
- (2011) Key Recovery attack on AES-128 in time $2^{126.2}$.
  - Improved to $2^{126.0}$ for AES-128, $2^{189.9}$ for AES-192 and $2^{254.3}$ for AES-256
- First public cipher approved by NSA for Top Secret information
  - SECRET level (AES-128,AES-192 & AES-256), TOP SECRET level (~~AES-128~~,AES-192 & AES-256)

NIST Recommendations

Ok, as CRHF and in Digital Signatures

Ok, to use for HMAC, Key Derivation and as PRG

80 bits-security is no longer acceptable

Date	Minimum of Strength	Symmetric Algorithms	Factoring Modulus	Discrete Logarithm Key	Discrete Logarithm Group	Elliptic Curve	Hash (A)	Hash (B)
(Legacy)	80	2TDEA*	1024	160	1024	160	SHA-1**	
2016 - 2030	112	3TDEA	2048	224	2048	224	SHA-224 SHA-512/224 SHA3-224	
2016 - 2030 & beyond	128	AES-128	3072	256	3072	256	SHA-256 SHA-512/256 SHA3-256	SHA-1
2016 - 2030 & beyond	192	AES-192	7680	384	7680	384	SHA-384 SHA3-384	SHA-224 SHA-512/224
2016 - 2030 & beyond	256	AES-256	15360	512	15360	512	SHA-512 SHA3-512	SHA-256 SHA-512/256 SHA-384 SHA-512 SHA3-512

Recommendations from Other Groups (Including NIST): www.keylength.com

Linear Cryptanalysis

$$y = F_K(x)$$

Definition: Fixed set of input bits $i_1, \ldots, i_{in}$ and output bits $i_1', \ldots, i_{out}'$ are said to have ε-linear bias if the following holds

$$\left| Pr\left[x_{i_1} \oplus x_{i_2} \ldots \oplus x_{i_{in}} \oplus y_{i_1'} \oplus y_{i_2'} \ldots \oplus y_{i_{out}'} \right] \right| = \varepsilon$$

(randomness taken over the selection of input x and secret key K)

Linear Cryptanalysis

Definition: Fixed set of input bits $i_1, \dots, i_{in}$ and output bits $i_1', \dots, i_{out}'$ are said to have ε-linear bias if the following holds

$$\left|Pr\left[x_{i_1} \oplus x_{i_2} \dots \oplus x_{i_{in}} \oplus y_{i_1'} \oplus y_{i_2'} \dots \oplus y_{i_{out}'}\right]\right| = \varepsilon$$

(randomness taken over the selection of input x and secret key K, $y = F_K(x)$)

Matsui: DES can be broken with just 2^{43} *known* plaintext/ciphertext pairs.

- Lots of examples needed!
- But the examples do not need to be chosen plaintext/ciphertext pairs…
- One encrypted file can provide a large amounts of known plaintext

Differential Cryptanalysis

Definition: We say that the differential $(\triangle_x, \triangle_y)$ occurs with probability p in the keyed block cipher F if

$$\left|Pr\left[F_K(x_1) \oplus F_K(x_1 \oplus \triangle_x) = \triangle_y\right]\right| \geq p$$

Can Lead to Efficient (Round) Key Recovery Attacks

Exploiting Weakness Requires: well over $\frac{1}{p}$ chosen plaintext-ciphertext pairs

Differentials in S-box can lead to (weaker) differentials in SPN.

CS 555: Week 8: Topic 1: One Way Functions

What are the minimal assumptions necessary for symmetric key-cryptography?

One-Way Functions (OWFs)

$$\mathrm{f}(x) = y$$

Definition: A function $\mathrm{f}: \{0,1\}^* \to \{0,1\}^*$ is one way if it is

1. **(Easy to compute)** There is a polynomial time algorithm (in |x|) for computing f(x).
2. **(Hard to Invert)** Select $\mathrm{x} \leftarrow \{0,1\}^n$ uniformly at random and give the attacker input 1^n, f(x). The probability that a PPT attacker outputs x' such that $\mathrm{f}(x') = f(x)$ is negligible.

One-Way Functions (OWFs)

$$\mathrm{f}(x) = y$$

Key Takeaway: One-Way Functions is a necessary and sufficient assumption for most of symmetric key cryptography.

- From OWFs we can construct PRGs, PRFs, Authenticated Encryption
- From eavesdropping secure encryption (weakest) notion we can construct OWFs

One-Way Functions (OWFs)

$$f(x) = y$$

Remarks:

- **A function that is not one-way is not necessarily always easy to invert (even often)**
- **Any such function can be inverted in time 2^n (brute force)**
- **Length-preserving OWF: $|f(x)| = |x|$**
- **One way permutation: Length-preserving + one-to-one**

One-Way Functions (OWFs)

$$\mathrm{f}(x) = y$$

Remarks:

1. **f(x) does not necessarily hide all information about x.**
2. **If f(x) is one way then so is $\mathbf{f}'(\mathbf{x}) = \mathbf{f}(\mathbf{x}) \parallel \boldsymbol{LSB}(\boldsymbol{x})$.**

One-Way Functions (OWFs)

$$\mathrm{f}(x) = y$$

Remarks:

1. **Actually we usually consider a family of one-way functions**

$$\boldsymbol{f_I}\colon \{\mathbf{0},\mathbf{1}\}^{\boldsymbol{I}} \rightarrow \{\mathbf{0},\mathbf{1}\}^{\boldsymbol{I}}$$

Candidate One-Way Functions (OWFs)

$$f_{p,g}(x) = [g^x \mod p]$$

(Discrete Logarithm Problem)

Note: The existence of OWFs implies $\mathrm{P} \neq NP$ so we cannot be *absolutely certain* that they do exist.

Hard Core Predicates

- Recall that a one-way function f may potentially reveal lots of information about input

- **Example**: $f(x_1,x_2)=(x_1,g(x_2))$, where g is a one-way function.
- **Claim**: f is one-way (even if $f(x_1,x_2)$ reveals half of the input bits!)

Hard Core Predicates

Definition: A predicate $\mathrm{hc}: \{0,1\}^* \rightarrow \{0,1\}$ is called a hard-core predicate of a function f if

1. (Easy to Compute) hc can be computed in polynomial time
2. (Hard to Guess) For all PPT attacker A there is a negligible function negl such that we have

$$\mathbf{Pr}_{x\leftarrow\{0,1\}^n}[A(1^n, f(x)) = \mathrm{hc}(x)] \leq \frac{1}{2} + negl(n)$$

Attempt 1: Hard-Core Predicate

Consider the predicate

$$\mathrm{hc(x)} = \bigoplus_{i=1}^{n} x_i$$

Hope: hc is hard core predicate for any OWF.

Counter-example:

$$\mathrm{f(x) = (g(x),}\ \bigoplus_{i=1}^{n} x_i)$$

Trivial Hard-Core Predicate

Consider the function

$$f(x_1,\ldots,x_n) = x_1,\ldots,x_{n-1}$$

f has a trivial hard core predicate

$$\mathrm{hc(x)} = x_n$$

Not useful for crypto applications (e.g., f is not a OWF)

Attempt 3: Hard-Core Predicate

Consider the predicate

$$\mathrm{hc}(\mathrm{x}, \mathrm{r}) = \oplus_{i=1}^{n} x_i r_i$$

(the bits r_1,…, r_n will be selected uniformly at random)

Goldreich-Levin Theorem: (Assume OWFs exist) For any OWF f, hc is a hard-core predicate of g(x,r)=(f(x),r).

Using Hard-Core Predicates

Theorem: Given a one-way-permutation f and a hard-core predicate hc we can construct a PRG G with expansion factor $\ell(n) = n + 1$.

Construction:

$$G(s) = f(s) \parallel \mathrm{hc}(s)$$

Intuition: f(s) is actually uniformly distributed

- s is random
- f(s) is a permutation
- Last bit is hard to predict given f(s) (since hc is hard-core for f)

Arbitrary Expansion

Theorem: Suppose that there is a PRG G with expansion factor $\ell(n) = n + 1$. Then for any polynomial p(.) there is a PRG with expansion factor p(n).

Construction:

- $G(x) = y||b$. (n+1 bits)
- $G^1(x) = G(y)||b$ (n+2 bits)
- $G^{i+1}(x) = G(y)||b$ where $G^i(x) = y||b$ (n+2 bits)

Any Beyond

Theorem: Suppose that there is a PRG G with expansion factor $\ell(n) = n + 1$. Then for any polynomial p(.) there is a PRG with expansion factor p(n).

Theorem: Suppose that there is a PRG G with expansion factor $\ell(n) = 2n$. Then there is a secure PRF.

Theorem: Suppose that there is a secure PRF then there is a strong pseudorandom permutation.

Any Beyond

Corollary: If one-way functions exist then PRGs, PRFs and strong PRPs all exist.

Corollary: If one-way functions exist then there exist CCA-secure encryption schemes and secure MACs.

PRFs from PRGs

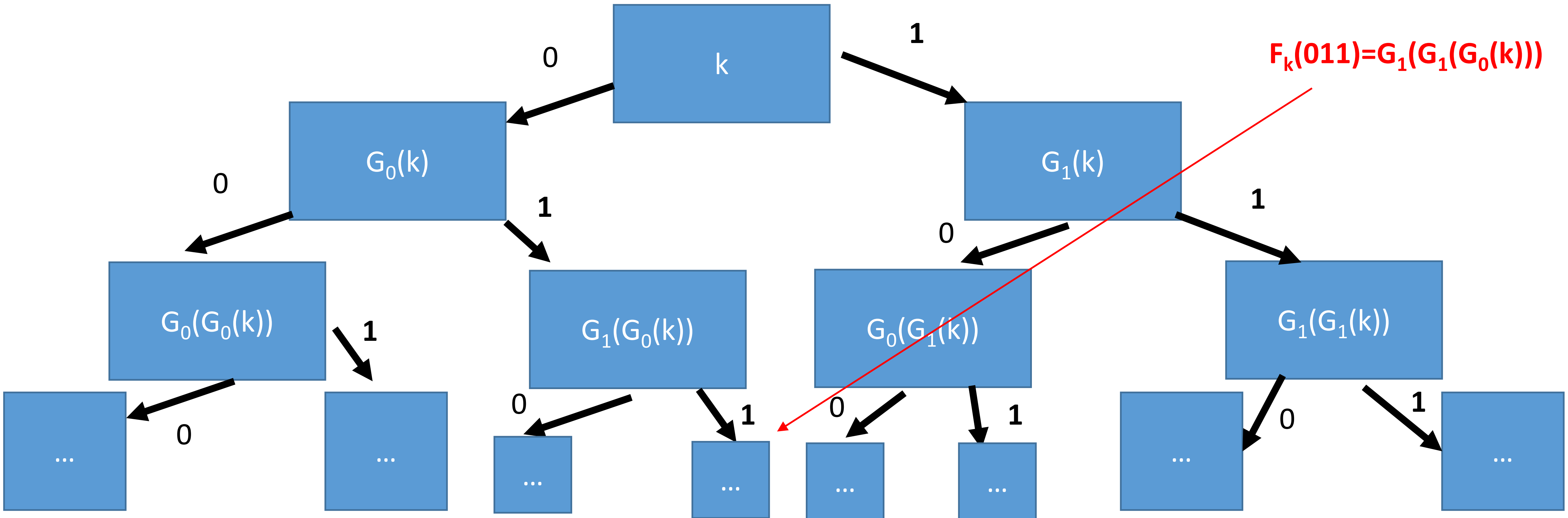
Theorem: Suppose that there is a PRG G with expansion factor $\ell(n) = 2n$. Then there is a secure PRF.

$\mathbf{Let\ G(x) = G_0(x)||G_1(x)}$ **(first/last n bits of output)**

$$\boldsymbol{F_K(x_1, \ldots, x_n) = G_{x_n}\left(\ldots\left(G_{x_2}\left(G_{x_1}(K)\right)\right)\ldots\right)}$$

PRFs from PRGs

Theorem: Suppose that there is a PRG G with expansion factor $\ell(n) = 2n$. Then there is a secure PRF.



PRFs from PRGs

Theorem: Suppose that there is a PRG G with expansion factor $\ell(n) = 2n$. Then there is a secure PRF.

Proof:

Claim 1: For any t(n) and any PPT attacker A we have

$$\left|\boldsymbol{Pr}\left[\boldsymbol{A}\left(\boldsymbol{r_1} \parallel \cdots \parallel \boldsymbol{r_{t(n)}}\right)\right] - \boldsymbol{Pr}\left[\boldsymbol{A}\left(\boldsymbol{G}(\boldsymbol{s_1}) \parallel \cdots \parallel \boldsymbol{G}\left(\boldsymbol{s_{t(n)}}\right)\right)\right]\right| < \boldsymbol{negl(n)}$$

PRFs from PRGs

Claim 1: For any t(n) and any PPT attacker A we have

$$\left|Pr\left[A\left(r_1 \parallel \cdots \parallel r_{t(n)}\right)\right] - Pr\left[A\left(G(s_1) \parallel \cdots \parallel G\left(s_{t(n)}\right)\right)\right]\right| < negl(n)$$

Proof by Hybrids: Fix j

$$Adv_j = \left|Pr\left[A\left(r_1 \parallel \cdots \parallel r_{j+1} \parallel G\left(s_{j+2}\right) \ldots \parallel G\left(s_{t(n)}\right)\right)\right]\right.$$

PRFs from PRGs

Claim 1: For any t(n) and any PPT attacker A we have

$$\left|\boldsymbol{Pr}\left[\boldsymbol{A}\left(\boldsymbol{r_1} \parallel \cdots \parallel \boldsymbol{r_{t(n)}}\right)\right] - \boldsymbol{Pr}\left[\boldsymbol{A}\left(\boldsymbol{G}(\boldsymbol{s_1}) \parallel \cdots \parallel \boldsymbol{G}\left(\boldsymbol{s_{t(n)}}\right)\right)\right]\right| < \boldsymbol{negl(n)}$$

Proof

$$\left|\boldsymbol{Pr}\left[\boldsymbol{A}\left(\boldsymbol{r_1} \parallel \cdots \parallel \boldsymbol{r_{t(n)}}\right)\right] - \boldsymbol{Pr}\left[\boldsymbol{A}\left(\boldsymbol{G}(\boldsymbol{s_1}) \parallel \cdots \parallel \boldsymbol{G}\left(\boldsymbol{s_{t(n)}}\right)\right)\right]\right|$$

$$\leq \sum_{\boldsymbol{j<t(n)}} \boldsymbol{Adv_j}$$

$$\leq \boldsymbol{t(n)} \times \boldsymbol{negl(n)} = \boldsymbol{negl(n)}$$

PRFs from PRGs

Claim 1: For any t(n) and any PPT attacker A we have

$$\left|Pr\left[A\left(r_1 \parallel \cdots \parallel r_{t(n)}\right)\right] - Pr\left[A\left(G(s_1) \parallel \cdots \parallel G\left(s_{t(n)}\right)\right)\right]\right| < negl(n)$$

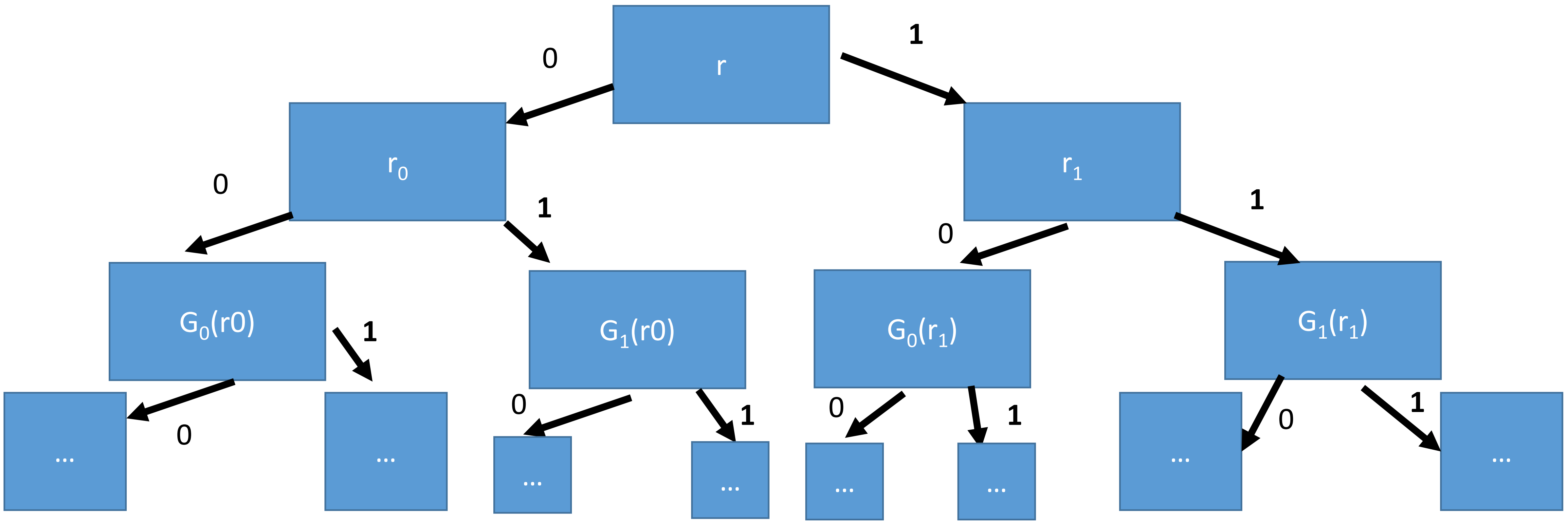
Proof

$$\left|Pr\left[A\left(r_1 \parallel \cdots \parallel r_{t(n)}\right)\right] - Pr\left[A\left(G(s_1) \parallel \cdots \parallel G\left(s_{t(n)}\right)\right)\right]\right|$$

$$\leq \sum_{j<t(n)} Adv_j$$

$$\leq t(n) \times negl(n) = negl(n)$$

Hybrid H_1



From OWFs (Recap)

Theorem: Suppose that there is a PRG G with expansion factor $\ell(n) = n + 1$. Then for any polynomial p(.) there is a PRG with expansion factor p(n).

Theorem: Suppose that there is a PRG G with expansion factor $\ell(n) = 2n$. Then there is a secure PRF.

Theorem: Suppose that there is a secure PRF then there is a strong pseudorandom permutation.

From OWFs (Recap)

Corollary: If one-way functions exist then PRGs, PRFs and strong PRPs all exist.

Corollary: If one-way functions exist then there exist CCA-secure encryption schemes and secure MACs.

Are OWFs Necessary for Private Key Crypto

- Previous results show that OWFs are <u>sufficient</u>.

- Can we build Private Key Crypto from weaker assumptions?

- **Short Answer:** No, OWFs are also <u>*necessary*</u> for most private-key crypto primitives

PRGs → OWFs

Proposition 7.28: If PRGs exist then so do OWFs.

Proof: Let G be a secure PRG with expansion factor $\ell(n) = 2n$.

Question: why can we assume that we have an PRG with expansion 2n?

Answer: Last class we showed that a PRG with expansion factor $\ell(n) = n + 1$. Implies the existence of a PRG with expansion p(n) for any polynomial.

PRGs → OWFs

Proposition 7.28: If PRGs exist then so do OWFs.

Proof: Let G be a secure PRG with expansion factor $\ell(n) = 2n$.

Claim: G is also a OWF!

(Easy to Compute?) ✓

(Hard to Invert?)

Intuition: If we can invert G(x) then we can distinguish G(x) from a random string.

PRGs → OWFs

Proposition 7.28: If PRGs exist then so do OWFs.

Proof: Let G be a secure PRG with expansion factor $\ell(n) = 2n$.

Claim 1: Any PPT A, given G(s), cannot find s except with negligible probability.

Reduction: Assume (for contradiction) that A can invert G(s) with non-negligible probability p(n).

Distinguisher D(y): Simulate A(y)

Output 1 if and only if A(y) outputs x s.t. G(x)=y.

PRGs → OWFs

Proposition 7.28: If PRGs exist then so do OWFs.

Proof: Let G be a secure PRG with expansion factor $\ell(n) = 2n$.

Claim 1: Any PPT A, given G(s), cannot find s except with negligible probability.

Intuition for Reduction: If we can find x s.t. G(x)=y then y is not random.

Fact: Select a random 2n bit string y. Then (whp) there does not exist x such that G(x)=y.

Why not?

PRGs → OWFs

Proposition 7.28: If PRGs exist then so do OWFs.

Proof: Let G be a secure PRG with expansion factor $\ell(n) = 2n$.

Claim 1: Any PPT A, given G(s), cannot find s except with negligible probability.

Intuition: If we can invert G(x) then we can distinguish G(x) from a random string.

Fact: Select a random 2n bit string y. Then (whp) there does not exist x such that G(x)=y.

- Why not? Simple counting argument, 2^{2n} possible y's and 2^n x's.
- Probability there exists such an x is at most 2^{-n} (for a random y)

What other assumptions imply OWFs?

- PRGs → OWFs
- (Easy Extension) PRFs → PRGs → OWFs

- Does secure crypto scheme imply OWFs?
  - CCA-secure? (Strongest)
  - CPA-Secure? (Weaker)
  - EAV-secure? (Weakest)
    - As long as the plaintext is longer than the secret key
  - Perfect Secrecy? X (Guarantee is information theoretic)

EAV-Secure Crypto → OWFs

Proposition 7.29: If there exists a EAV-secure private-key encryption scheme that encrypts messages twice as long as its key, then a one-way function exists.

Recap: EAV-secure.

- Attacker picks two plaintexts m_0,m_1 and is given c=$Enc_K(m_b)$ for random bit b.
- Attacker attempts to guess b.
- No ability to request additional encryptions (chosen-plaintext attacks)
- In fact, no ability to observe any additional encryptions

EAV-Secure Crypto → OWFs

Proposition 7.29: If there exists a EAV-secure private-key encryption scheme that encrypts messages twice as long as its key, then a one-way function exists.

Reduction: $\boldsymbol{f(m,k,r) = Enc_k(m;r) \| m}$.

Input: 4n bits

(For simplicity assume that $\mathbf{Enc}_k$ accepts n bits of randomness)

Claim: f is a OWF

EAV-Secure Crypto → OWFs

Proposition 7.29: If there exists a EAV-secure private-key encryption scheme that encrypts messages twice as long as its key, then a one-way function exists.

Reduction: $\boldsymbol{f(m, k, r) = Enc_k(m; r) \| m}$.

Claim: f is a OWF

Reduction: If attacker A can invert f, then attacker A' can break EAV-security as follows. Given c=$\mathrm{Enc}_k(m_b;r)$ run A(c$\|m_0$). If A outputs (m',k',r') such that $\mathrm{f(m', k', r') = c}\|m_0$ then output 0; otherwise 1;

MACs→ OWFs

In particular, given a MAC that satisfies MAC security (Definition 4.2) against an attacker who sees an arbitrary (polynomial) number of message/tag pairs.

Conclusions: OWFs are necessary and sufficient for all (non-trivial) private key cryptography.

→OWFs are a minimal assumption for private-key crypto.

Public Key Crypto/Hashing?

- OWFs are known to be necessary
- Not known (or believed) to be sufficient.

Computational Indistinguishability

- Consider two distributions X_ℓ and Y_ℓ (e.g., over strings of length ℓ).
- Let D be a distinguisher that attempts to guess whether a string s came from distribution X_ℓ or Y_ℓ.

The advantage of a distinguisher D is

$$Adv_{D,\ell} = \left| Pr_{s \leftarrow X_\ell}[D(s) = 1] - Pr_{s \leftarrow Y_\ell}[D(s) = 1] \right|$$

Definition: We say that an ensemble of distributions $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ are <u>computationally indistinguishable</u> if for all PPT distinguishers D, there is a negligible function negl(n), such that we have

$$Adv_{D,n} \leq negl(n)$$