

# Advanced Cryptography

Serge Vaudenay



<http://lasec.epfl.ch/>

LASEC

- 1 The Cryptographic Zoo
- 2 Cryptographic Security Models
- 3 Cryptanalysis (Public-Key)
- 4 The Power of Interaction
- 5 Cryptanalysis (Conventional)
- 6 Proving Security

# 1 The Cryptographic Zoo

2 Cryptographic Security Models

3 Cryptanalysis (Public-Key)

4 The Power of Interaction

5 Cryptanalysis (Conventional)

6 Proving Security

# 1 The Cryptographic Zoo

- The Menagery
- The Math Toolbox
- The Algorithmic Toolbox
- The Complexity Theory Toolbox

## 2 Cryptographic Security Models

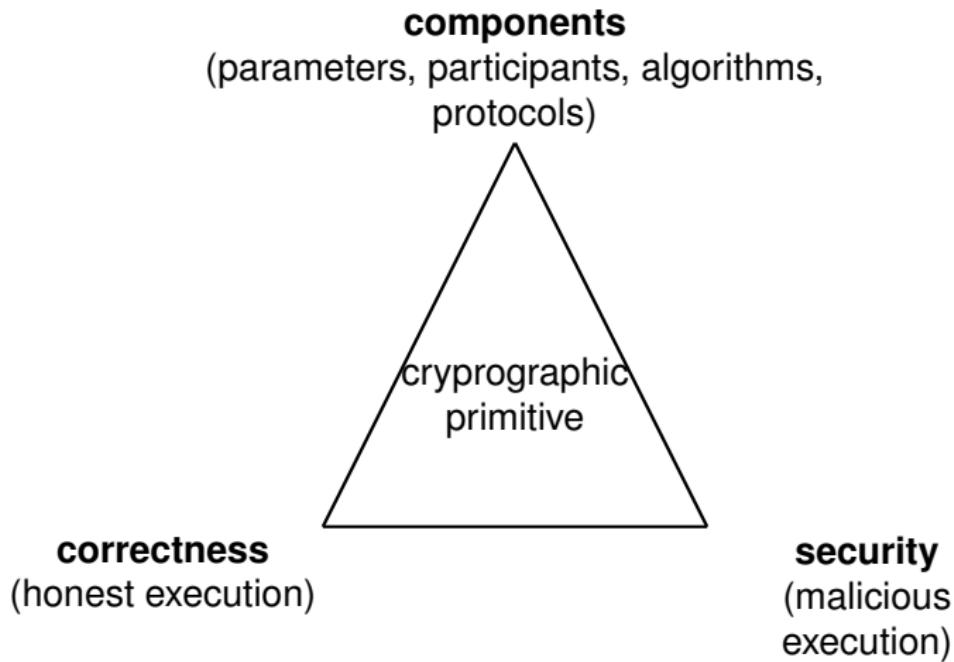
## 3 Cryptanalysis (Public-Key)

## 4 The Power of Interaction

## 5 Cryptanalysis (Conventional)

## 6 Proving Security

# Meta-Definition of a Cryptographic Primitive



# Correctness vs Security

- **correctness:**

what happens if everyone honestly follow the protocol

operations must be easy to perform      *easy = cost has low as possible*

example:  $\text{Dec}_K(\text{Enc}_K(X)) = X$

- **security:**

what should not happen even if someone is malicious  
attacks should be hard to perform

practically: the threat model defines a game in which  
malicious adversary following any polynomially bounded  
strategy gets a negligible advantage

example: key recovery under chosen plaintext attack is  
hard

## Easy vs Hard

There exist several notions of easiness/hardness. In this course we mostly follow the polynomial/non-polynomial complexity one:

- a computational problem with security parameter  $s$  is **easy** if it can be solved with a probabilistic algorithm of complexity  $s^{\mathcal{O}(1)}$  when  $s \rightarrow +\infty$ , i.e. probabilistic polynomial-time (PPT) algorithm  $\rightarrow$  easy=PPT (it is **hard** otherwise)
- example: a system is secure if it is hard to break

Sometimes, we may consider the **information theoretic approach** (where adversaries are not limited in terms of complexity).

# Negligible and Secure

- a function  $f$  is **negligible** if

$$\forall n \quad f(s) = \mathcal{O}(s^{-n}) \quad (s \rightarrow +\infty)$$

- example:  $f(s) = 2^{-s}$
- avoid writing *non-negligible* as it can be confusing  
(for some authors, non-negligible  $\neq \neg$  negligible...)
- a system is **secure** if every PPT adversary has a negligible advantage

$s$  is a security parameter

# Notation: Security Parameter

A cryptographic algorithm  $\mathcal{A}$  often depends on some **security parameter**  $s$  (e.g. a key length).

We denote  $1^s = \overbrace{11 \cdots 1}^{s \text{ times}}$ , the **unary** representation of  $s$ , and consider  $1^s$  as the first input of  $\mathcal{A}$ :

$\mathcal{A}(1^s, x; r)$     sequence of coin flip  
made during execution of  
the program

Caution:  $1^s$  is often implicit and omitted from notations for simplicity!

In practice, we consider **polynomially bounded** algorithms. Their complexity is bounded by a polynomial in terms of the length of the inputs, i.e.  $s + |x|$ . Since  $|x|$  is polynomially bounded in terms of  $s$ , the complexity of  $\mathcal{A}$  is  $s^{\mathcal{O}(1)}$ .

## Notation: Probabilistic Algorithms

A **probabilistic algorithm**  $\mathcal{A}(\text{input}; \text{coins})$  is an algorithm  $\mathcal{A}$  fed with input which sometimes needs to flip a coin to make a decision. For convenience, we prepare “coins”: an infinite sequence of coin flips.  $\mathcal{A}$  can read them in sequence whenever needed.

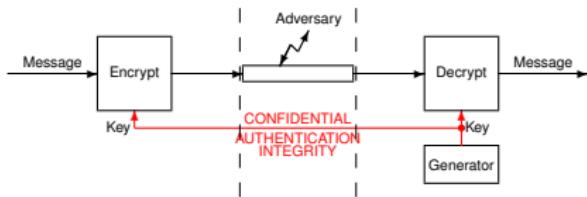
So,  $\mathcal{A}$  can also be seen as a **deterministic** function of both input and coins. By convenience, we separate input and coins by a semi-colon:  $\mathcal{A}(\text{input}; \text{coins})$

Not all coins are used: there is always a minimal prefix  $r$  of coins such that for all sequences  $s$ , we have  
 $\mathcal{A}(\text{input}; r\|s) = \mathcal{A}(\text{input}; \text{coins})$ . We denote this common value by  $\mathcal{A}(\text{input}; r)$ .

# Big Picture

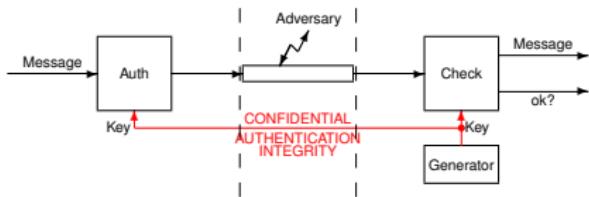
## confidential transmission

*symmetric encryption*



## authenticated transmission

*NAC*



*Asymmetric encryption*



*Digital signature*

# Symmetric Encryption

plaintext, ciphertext  $\in \mathcal{D} \subseteq \{0, 1\}^*$

encryption : key, nonce, plaintext  $\mapsto$  ciphertext

decryption : key, nonce, ciphertext  $\mapsto$  plaintext

Security goal: protect the confidentiality of plaintexts

- **block cipher:**  $\mathcal{D} = \{0, 1\}^{\text{blocklength}}$   
each key defines a permutation over  $\mathcal{D}$
- may use a **nonce** (must not be reused in encryption)
- **variable-length encryption:**  $|\text{plaintext}| = |\text{ciphertext}|$
- **authenticated encryption:** may use **associated data** as additional input (to authenticate)

encryption : key, nonce, ad, plaintext  $\mapsto$  ciphertext

decryption : key, nonce, ad, ciphertext  $\mapsto$  plaintext

# Using Encryption

- **usage:**
  - share a random key  $K$  between the sender and the receiver
  - to transmit a message  $pt$ , compute  $ct = \text{Enc}_K(pt)$  and send  $ct$  over the communication channel
  - to receive the message, get  $ct$  and compute  $pt = \text{Dec}_K(ct)$

- **correctness:**

$$\text{Dec}_K(\text{Enc}_K(pt)) = pt$$

- **security:** notion of confidentiality

MAC : key, message  $\mapsto$  tag

- (typical construction)
- tag of fixed length

Security goal: ensures that the message was authenticated by the key holder and not modified (authentication + integrity)

# Using MAC for Authentication

- **usage:**

$$\text{Auth}_K(X) = X \parallel \text{MAC}_K(X)$$

$$\text{Check}_K(X \parallel t) = (X, 1_{t=\text{MAC}_K(X)})$$

- **correctness:**

$$\text{Check}_K(\text{Auth}_K(X)) = (X, 1)$$

- **security:** notion of authentication and integrity

# Public-Key Cryptography

- **public-key cryptosystem:** encryption

Gen :  $\perp \xrightarrow{\$} pk, sk$

Enc :  $pk, pt \xrightarrow{\$} ct$

Dec :  $sk, ct \mapsto pt$

- **digital signature scheme:** authentication + integrity

Gen :  $\perp \xrightarrow{\$} pk, sk$

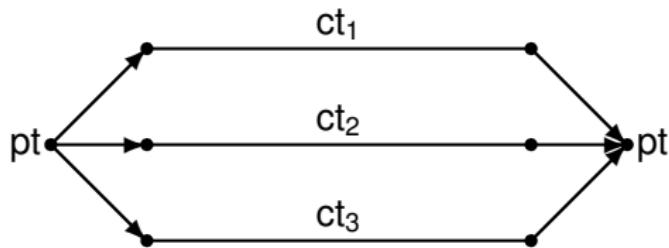
Sig :  $sk, X \xrightarrow{\$} \sigma$

Ver :  $pk, X, \sigma \mapsto 0/1$

# Using Public-Key Encryption

- **usage:**
  - generate a key pair  $(pk, sk)$  using  $\text{Gen}$
  - the receiver keeps  $sk$  and publicly reveals  $pk$
  - to transmit a message  $pt$ , compute  $ct = \text{Enc}(pk, pt)$  and send  $ct$  over the communication channel
  - to receive the message, get  $ct$  and compute  $pt' = \text{Dec}(sk, ct)$
- **correctness:**  $pt' = pt$
- **security:** notion of confidentiality

# Non-Deterministic Encryption



Encrypt

Decrypt

# Using Signature

- **usage:**
  - generate a key pair  $(pk, sk)$  using  $\text{Gen}$
  - the sender keeps  $sk$  and publicly reveals  $pk$
  - to sign a message  $X$ , compute  $\sigma = \text{Sig}(sk, X)$  and send  $X$  with  $\sigma$  over the communication channel
  - to validate the message, get  $(X, \sigma)$  and compute  $\text{Ver}(pk, X, \sigma)$
- **correctness:**  $\text{Ver}$  evaluates to true
- **security:** unforgeability

# Key Agreement Protocol

- **usage:**  $A$  and  $B$  run an interactive protocol which uses no common secret input and each produce a private output  $K_A$  and  $K_B$
- **correctness:**  
running  $A(1^s; r_a)$  and  $B(1^s; r_b)$  together leads to  $K_A = K_B$
- **security:** (under passive attacks)  
key recovery:  $\Pr[\mathcal{A}(\text{transcript}) = K : A \leftrightarrow B] = \text{negl}(s)$   
hard to compute  $K$  from protocol exchanges and public inputs  
(+ many other refinements)

# Commitment Scheme

(most common construction)

## Definition

A **commitment scheme** is a tuple  $(\mathcal{D}, \text{Commit})$  with a message domain  $\mathcal{D}$  and one PPT algorithm  $\text{Commit}$  implementing a function

$$\begin{aligned}\text{Commit} : \quad & \mathcal{D} \times \text{Random} & \longrightarrow & \quad \{0, 1\}^* \\ & (X, r) & \longmapsto & \quad \text{Commit}(X, r)\end{aligned}$$

# Using Commitment

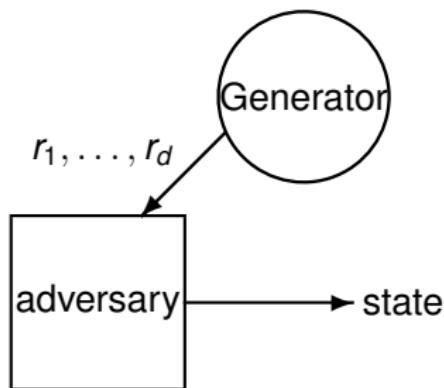
- **usage:**
  - commit to  $X$ : pick  $r$  and send  $c = \text{Commit}(X, r)$
  - open commitment: send  $X$  and  $r$ ; receiver verifies  
 $c = \text{Commit}(X, r)$
- **correctness:** verification succeeds
- **security:** binding and hiding

# Other Conventional Primitives

(informal)

- **pseudorandom number generator (PRNG)**  
typically:  $\text{PRNG}(\text{state}) \rightarrow (\text{new state}, \text{output } r)$
- **key derivation function (KDF)**  
maps some random seed with a bias (e.g. a group element coming from Diffie-Hellman key agreement) to a (set of) secret key(s) with no bias
- **hash function**  
maps arbitrary length input to fixed-length output

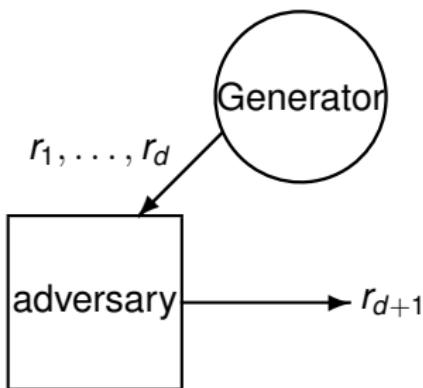
# 1st Adversarial Model for PRNG: State Recovery



- the goal of the adversary is to recover the internal state of the generator

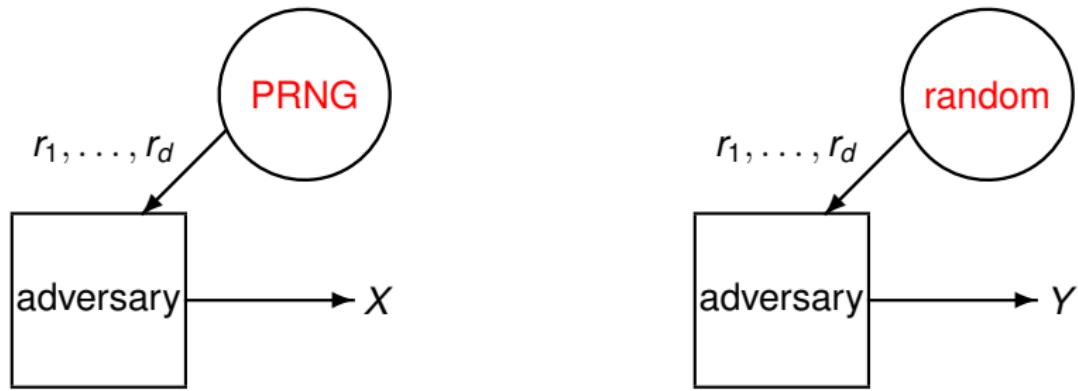
not enough: we could still predict the next  $r_i$

## 2nd Adversarial Model for PRNG: Unpredictability



- the goal of the adversary is to predict the next generation (possibly: by recovering the internal state)
- not enough: it does not say  $r_i$  looks like random

## 3rd Adversarial Model for PRNG: Indistinguishability



$$\text{Adv} = \Pr[X = 1] - \Pr[Y = 1]$$

- the goal of the adversary is to have  $|\text{Adv}|$  large  
(possibly: by predicting the next generation)

# Swiss Army Hash Function

- PRNG:

$$\text{PRNG}(\text{state}) = \text{trunc}_{\text{required length}}(H(\text{state}\|1)\|H(\text{state}\|2)\|\dots)$$

- KDF:

$$\text{KDF}(\text{seed}) = \text{trunc}_{\text{required length}}(H(\text{seed}\|1)\|H(\text{seed}\|2)\|\dots)$$

- Commitment:

$$\text{Commit}(X, r) = H(\langle X, r \rangle)$$

$\langle X, r \rangle$ : encoding of  $X$  and  $r$  with non-ambiguous decoding  
(example: concatenation, if  $X$  has a fixed length)

- Domain expander:

$$\text{primitive}_{\text{large domain}}(\text{input}) = \text{primitive}_{\text{limited domain}}(H(\text{input}))$$

# A Few Adversarial Models for Hash Functions

(informal)

- adversary objective: depends on the application
- **first preimage attack**: given  $y$ , find  $x$  such that  $H(x) = y$
- **second preimage attack**: given  $x$ , find  $x' \neq x$  such that  $H(x) = H(x')$
- **collision attack**: find  $x$  and  $x'$  such that  $x' \neq x$  and  $H(x) = H(x')$

# 1 The Cryptographic Zoo

- The Menagerie
- The Math Toolbox
- The Algorithmic Toolbox
- The Complexity Theory Toolbox

## 2 Cryptographic Security Models

## 3 Cryptanalysis (Public-Key)

## 4 The Power of Interaction

## 5 Cryptanalysis (Conventional)

## 6 Proving Security

# Finite Abelian Group

- **finite Abelian group:** set with an operation which satisfies **closure, associativity, existence of neutral element, universal invertibility, commutativity**, and finite **cardinality**  
examples:  $\mathbf{Z}_n$ ,  $\mathbf{Z}_p^*$ ,  $\overbrace{\text{GF}(q)}^{\text{all invertible elements to } q \text{ in }}{}^*$ , elliptic curve  $E_{a,b}(\mathbf{K})$
- **additive or multiplicative notation:**

|                 |         |   |       |         |       |
|-----------------|---------|---|-------|---------|-------|
| additive        | $a + b$ | 0 | $-a$  | $a - b$ | $n.a$ |
| multiplicative: | $ab$    | 1 | $1/a$ | $a/b$   | $a^n$ |

- **group constructors:** spanned subgroup, product group, power group, quotient by subgroup
- **group order:** cardinality of the group
- **element order:** order of the subgroup spanned by the element = smallest power  $n$  such that  $x^n = 1$
- **group exponent:** smallest power  $n$  such that  $x^n = 1$  for all  $x$
- **Lagrange property:** the element order divides the group exponent which divides the group order

# Commutative Ring

- **commutative ring:** set with two operations  $+$  and  $\times$  which satisfies
  - for  $+$ : **Abelian group**
  - for  $\times$ : **closure, associativity, existence of neutral element, commutativity**
  - **distributivity**
- examples:  $\mathbf{Z}$ ,  $\mathbf{Z}_n$ ,  $\mathbf{Z}[x]$ ,  $\mathbf{Z}_p[x]$
- **ring constructors:** spanned ideal, product ring, power ring, quotient by ideal
- **Euclidean ring:** ring with a Euclidean division (e.g.  $\mathbf{Z}$ ,  $K[x]$ )
- **principal ring:** ring in which all ideals have a generator  
ring is Euclidean  $\implies$  ring is principal
- **group of units:**  $R^*$  is the set of all invertible elements in  $R$   
it is a group for multiplication!

# Irreducibility and Primality in Rings

- in  $\mathbf{Z}$ :

$$p \text{ prime iff } p > 1 \text{ and} \\ \forall a, b \in \mathbf{Z}, \quad p = ab \implies |a| = 1 \text{ or } |b| = 1$$

- in  $K[x]$ :

$$P(x) \text{ irreducible iff } \forall A(x), B(x) \in K[x], \quad P(x) = \\ A(x)B(x) \implies \deg(A) = 0 \text{ or } \deg(B) = 0$$

(in a principal ring)

every ring element  $x$  can be written as  $x = p_1 \cdots p_m$  where each  $p_i$  is irreducible and the **factorization is unique** in the following sense: if  $x = q_1 \cdots q_n$  where each  $q_i$  is irreducible, then there exists a bijection  $f : \{1, \dots, m\} \rightarrow \{1, \dots, n\}$  such that for every  $i$  there exists a unit  $u_i$  such that  $q_{f(i)} = u_i p_i$  (so  $m = n$ )

# Finite Field

- **finite field:** ring with finite cardinality in which all nonzero elements are invertible
- **multiplicative group:**  $K^* = K - \{0\}$
- **Galois field:**  $\text{GF}(p^n) = \mathbf{Z}_p[x]/(P(x))$  where  $p$  is prime,  $P(x)$  is a monic (i.e., with leading coefficient 1) irreducible polynomial of degree  $n$  in  $\mathbf{Z}_p[x]$
- **Galois theorem:** the set of all finite field cardinalities is the set of all integers of form  $p^n$  for  $p$  prime, fields of same cardinality are isomorphic, and the above construction is possible for every prime power  
useful fields in crypto:  $\mathbf{Z}_p$ ,  $\text{GF}(2^n)$

# Facts About the $\mathbf{Z}_n$ Ring

Smallest power st any elements remain  
to this power is equal to 1.

- $x \in \mathbf{Z}_n$  is invertible iff  $\gcd(x, n) = 1$
- $\mathbf{Z}_n^*$  is of cardinality  $\varphi(n)$  and exponent  $\lambda(n)$
- if  $p_1, \dots, p_r$  are pairwise different prime numbers

$$\begin{aligned}\varphi(p_1^{\alpha_1} \cdots p_r^{\alpha_r}) &= (p_1 - 1)p_1^{\alpha_1 - 1} \cdots (p_r - 1)p_r^{\alpha_r - 1} \\ \lambda(p_1^{\alpha_1} \cdots p_r^{\alpha_r}) &= \text{lcm}(\lambda(p_1^{\alpha_1}), \dots, \lambda(p_r^{\alpha_r}))\end{aligned}$$

with  $\lambda(p^\alpha) = \varphi(p^\alpha)$  except for  $\lambda(2^\alpha)$  with  $\alpha \geq 3$   
(for which  $\lambda(2^\alpha) = \frac{1}{2}\varphi(2^\alpha)$ )

- Euler theorem: for  $x \in \mathbf{Z}_n^*$  we have  $x^{\varphi(n)} \bmod n = 1$
- for  $x \in \mathbf{Z}_n^*$  we have  $x^{\lambda(n)} \bmod n = 1$

# Facts About the $\mathbf{Z}_p$ Field

given  $p > 2$  prime

- $\mathbf{Z}_p^*$  has a generator
- little Fermat theorem: for  $x \in \mathbf{Z}_p^*$  we have  $x^{p-1} \bmod p = 1$
- $\text{QR}(p)$  is the group of **quadratic residues** modulo  $p$   
it is of order  $\frac{p-1}{2}$
- $x \in \text{QR}(p)$  iff  $x^{\frac{p-1}{2}} \bmod p = 1$

# Chinese Remainder Theorem

## Theorem (Chinese Remainder Theorem)

Let  $m$  and  $n$  be two integers such that  $\gcd(m, n) = 1$ . We have

- $f : \mathbf{Z}_{mn} \rightarrow \mathbf{Z}_m \times \mathbf{Z}_n$  is a ring isomorphism  
 $x \mapsto (x \bmod m, x \bmod n)$
- $f^{-1}(a, b) \equiv an(n^{-1} \bmod m) + bm(m^{-1} \bmod n) \pmod{mn}$

# Random Variables

**random variable:** process  $X$  taking random coins  $r$  as input and producing some values in a given set  $\mathcal{Z}$

**support:** set of possible outputs

**its probability distribution:** function  $P$  from a set including the support to  $\mathbf{R}$  mapping values  $x$  to their probabilities  $\Pr[X = x]$

**independent variables:** random variables  $X$  and  $Y$  s.t. for all  $x, y$

$$\Pr[X = x \text{ and } Y = y] = \Pr[X = x] \times \Pr[Y = y]$$

examples: random variables using two disjoint sets of random coins

# Expected Value and Variance

- **expected value:** given a random variable  $X$  with a range in a vector space over the real numbers,  $E(X)$  is a vector

$$E(X) = \sum_r \Pr[r]X(r) = \sum_{x \in \text{support}} x \Pr[X = x]$$

- **variance:** given a random variable  $X$  with a range in  $\mathbf{R}$ ,

$$V(X) = E((X - E(X))^2) = E(X^2) - (E(X))^2$$

- **Boolean random variable:** if  $\Pr[X \in \{0, 1\}] = 1$

$$E(X) = p \quad \text{and} \quad V(X) = p(1 - p) \quad \text{where} \quad p = \Pr[X = 1]$$

- **linearity:** given random variables  $X, Y$  and scalars  $\lambda, \mu$ :

$$E(\lambda X + \mu Y) = \lambda E(X) + \mu E(Y) \quad \text{and} \quad V(\lambda X) = \lambda^2 V(X)$$

- **case of independent variables:** given **independent**  $X, Y$ :

$$E(XY) = E(X)E(Y) \quad \text{and} \quad V(X + Y) = V(X) + V(Y)$$

# Other Properties

$$\begin{aligned} E(f(X)) &= \sum_r f(X(r)) \Pr[r] \\ &= \sum_{x \in \text{support}} f(x) \Pr[X = x] \\ &= \sum_y y \Pr[f(X) = y] \end{aligned}$$

# 1 The Cryptographic Zoo

- The Menagerie
- The Math Toolbox
- **The Algorithmic Toolbox**
- The Complexity Theory Toolbox

## 2 Cryptographic Security Models

## 3 Cryptanalysis (Public-Key)

## 4 The Power of Interaction

## 5 Cryptanalysis (Conventional)

## 6 Proving Security

# Arithmetics with Big Numbers

$\ell$ : size of the input

- addition ( $\mathcal{O}(\ell)$ ):  $x, y \mapsto x + y$
- multiplication ( $\mathcal{O}(\ell^2)$ ):  $x, y \mapsto x \times y$
- Euclidean division ( $\mathcal{O}(\ell^2)$ ):  $x, n \mapsto x \bmod n$
- **extended Euclid Algorithm** ( $\mathcal{O}(\ell^2)$ ):  $x, y \mapsto a, b$  s.t.  
 $ax + by = \gcd(x, y)$

# Modular Arithmetic

$\ell$ : size of  $n$

- addition ( $\mathcal{O}(\ell)$ ):  $x, y, n \mapsto (x + y) \bmod n$
- multiplication ( $\mathcal{O}(\ell^2)$ ):  $x, y, n \mapsto (x \times y) \bmod n$   
(Note: asymptotically better algorithm based on FFT, but not better in practice)
- Euclidean division ( $\mathcal{O}(\ell^2)$ ):  $x, n \mapsto x \bmod n$
- fast exponential ( $\mathcal{O}(\ell^2 \log e)$ ):  $x, e, n \mapsto x^e \bmod n$   
using the **square and multiply algorithm**
- inversion in  $\mathbf{Z}_n$  ( $\mathcal{O}(\ell^2)$ ):  $x, n \mapsto y$  s.t.  $xy \bmod n = 1$  (when feasible)  
using the **extended Euclid algorithm**

# Other Algorithms

$\ell$ : size of  $n$

- square root with factorization of  $n = p_1^{\alpha_1} \cdots p_r^{\alpha_r}$  ( $\mathcal{O}(\ell^3)$ ):  
 $x, p_1^{\alpha_1}, \dots, p_r^{\alpha_r} \mapsto y$  s.t.  $y^2 \equiv x \pmod{n}$
- primality test ( $\mathcal{O}(k\ell^3)$ ):  $n \mapsto 0$  or  $1$   
( $\mathcal{O}(\ell^3)$  if  $n$  is composite, with  $\Pr[\text{wrong answer}] \leq e^{-\Omega(k)}$ )
- prime number generation ( $\mathcal{O}(\ell^4)$ ):  $\emptyset \mapsto p$  ( $\ell$  is the size of  $p$ )

# Birthday Effect

Given a **random** hash function with output domain of size  $N$  we can find a collision with complexity  $\mathcal{O}(\sqrt{N})$

# Generic Attacks

| primitive     | attack          | complexity                | parameter $n$ |
|---------------|-----------------|---------------------------|---------------|
| encryption    | key recovery    | $\Theta(2^n)$             | key length    |
| hash function | preimage attack | $\Theta(2^n)$             | hash length   |
|               | collision       | $\Theta(2^{\frac{n}{2}})$ | hash length   |
| MAC           | key recovery    | $\Theta(2^n)$             | key length    |

## 1 The Cryptographic Zoo

- The Menagerie
- The Math Toolbox
- The Algorithmic Toolbox
- The Complexity Theory Toolbox

## 2 Cryptographic Security Models

## 3 Cryptanalysis (Public-Key)

## 4 The Power of Interaction

## 5 Cryptanalysis (Conventional)

## 6 Proving Security

# Membership Problem

- problem defined by a **language**  $L$  (set of words)
- instance specified by a **word**  $x$
- the membership problem is to decide whether  $x \in L$  or not
- languages in  $\mathcal{NP}$  are of form

$$L = \{x; \exists w \quad R(x, w)\}$$

- for some **predicate**  $R$  which can be computed in polynomial time
- proof of membership specified by a (polynomially-bounded-sized) **witness**  $w$

# Easy Problems

$\mathcal{P}$  problems:

class of problems which can be solved by a polynomially bounded algorithm

- primality:  
given an integer, decide if it is prime

# Hard Problems

- **$\mathcal{NP}$ -hard problems:**  
class of problems such that if we can solve them in polynomial time then we can solve all  $\mathcal{NP}$  ones
- **factoring problem:**  
given an integer generation algorithm, the problem is to find a non-trivial factor of a number generated by this algorithm  
(not known  $\mathcal{NP}$ -hard but still hard)
- **discrete logarithm problem:**  
given a group generated by some  $g$  in which operations are computable in polynomial time, given  $y$ , find  $x$  such that  $y = g^x$   
(not known  $\mathcal{NP}$ -hard but still hard)

# Non-Polynomial Algorithms

best algorithms so far:

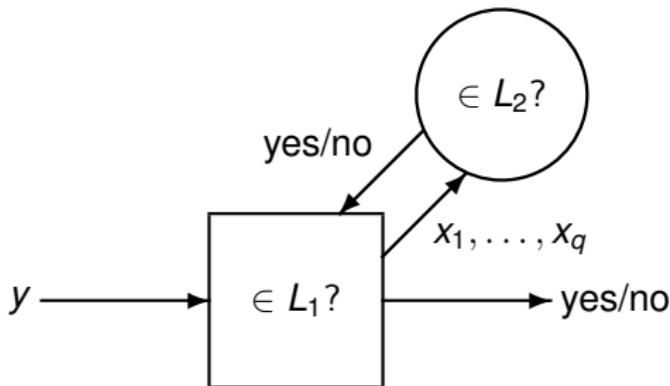
- NFS factoring (factoring  $n$ )  $e^{\mathcal{O}\left((\ln n)^{\frac{1}{3}}(\ln \ln n)^{\frac{2}{3}}\right)}$
- ECM factoring (find a small factor  $p$  of  $n$ )  $e^{\mathcal{O}(\sqrt{\ln p \ln \ln p})}$
- GNFS (discrete logarithm in  $\mathbf{Z}_p^*$ ): same as NFS
- Index calculus (discrete logarithm in  $\mathbf{Z}_p^*$ )  $e^{\mathcal{O}(\sqrt{\ln p \ln \ln p})}$

# Turing Reduction

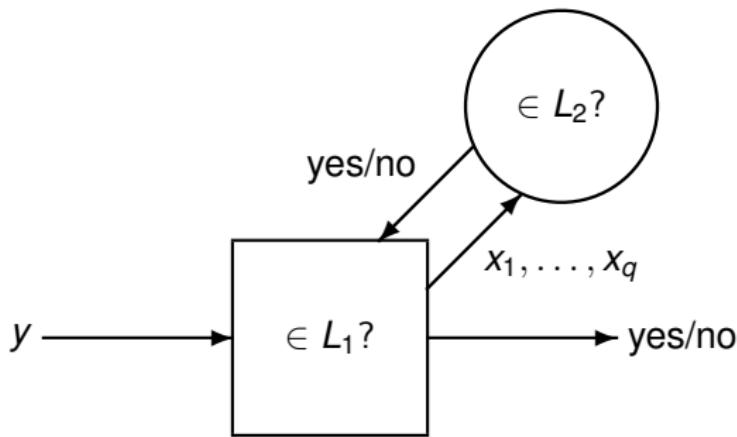
**Oracle:** Boolean function which says if a word is in a given language. The oracle is connected to a query tape which includes a finite number of non-blank cells.

**Oracle Turing machine:** Turing machine with distinguished query state and query tape

**Turing reduction:** a language  $L_1$  reduces to a language  $L_2$  if there exists a polynomial deterministic oracle Turing machine which recognizes  $L_1$  when plugged on an oracle  $L_2$ .



# Turing Reduction



- solving  $L_2$  implies solving  $L_1$
- $L_1$  is hard  $\implies L_2$  is hard

# Conclusion

- **a menagerie of cryptographic primitives:**  
encryption, MAC, commitment, key agreement, signature
- **a math toolbox:**  
number theory, probability theory
- **an algorithmic toolbox:**  
big number calculation, generic algorithms
- **a complexity theory toolbox:**  
hard problem, reduction

## References

- **Vaudenay.**  
A Classical Introduction to Cryptography — Applications for Communications Security.  
Springer. 2005.
- **Stinson.**  
Cryptography, Theory and Practice (3rd Edition).  
CRC. 2005.
- **Menezes-van Oorschot-Vanstone.**  
Handbook of Applied Cryptography.  
CRC. 1997. <http://www.cacr.math.uwaterloo.ca/hac/>
- **Shoup.**  
A Computational Introduction to Number Theory and Algebra.  
Cambridge University Press. 2005.  
<http://shoup.net/ntb>

# Train Yourself

- variant of collision search: midterm exam 2015–16 ex2



1 The Cryptographic Zoo

## 2 Cryptographic Security Models

3 Cryptanalysis (Public-Key)

4 The Power of Interaction

5 Cryptanalysis (Conventional)

6 Proving Security

## 1 The Cryptographic Zoo

## 2 Cryptographic Security Models

- Security Definitions
- The Game Proof Methodology
- RSA Security
- Rabin Cryptosystem
- Diffie-Hellman Security
- ElGamal Security

## 3 Cryptanalysis (Public-Key)

## 4 The Power of Interaction

## 5 Cryptanalysis (Conventional)

## 6 Proving Security

# Block Cipher

## Definition

A **block cipher** is a tuple  $(\{0, 1\}^{k(s)}, \mathcal{D}_s, \text{Enc}, \text{Dec})$  with a key domain  $\{0, 1\}^{k(s)}$ , a plaintext domain  $\mathcal{D}_s = \{0, 1\}^{n(s)}$ , and two “efficient” deterministic algorithms Enc and Dec.

It is such that

easy in sense of polynomial time in known of  $s$

$$\forall s \quad \forall K \in \{0, 1\}^{k(s)} \quad \forall X \in \mathcal{D}_s \quad \left\{ \begin{array}{l} \text{Dec}_s(K, \text{Enc}_s(K, X)) = X \\ \text{Enc}_s(K, X) \in \mathcal{D}_s \end{array} \right.$$

efficient = polynomially bounded in terms of  $s$   
( $s$  is often implicit in notations)

# Symmetric Encryption

## Definition

A (**nonce-based**, **variable-length**, **length-preserving**) **symmetric encryption scheme** is a tuple  $(\{0, 1\}^k, \mathcal{D}, \mathcal{N}, \text{Enc}, \text{Dec})$  with a key domain  $\{0, 1\}^k$ , a plaintext domain  $\mathcal{D} \subseteq \{0, 1\}^*$ , a nonce domain  $\mathcal{N}$ , and two polynomially bounded deterministic algorithms Enc and Dec. It is such that

$$\forall K \in \{0, 1\}^k \quad \forall X \in \mathcal{D} \quad \forall N \in \mathcal{N} \quad \begin{cases} \text{Dec}(K, N, \text{Enc}(K, N, X)) = X \\ |\text{Enc}(K, N, X)| = |X| \end{cases}$$

$N$  is supposed to be used only once for encryption random nonce (beware of random repetitions), counter, sent in clear or synchronized

# Security against Key Recovery

## Definition

A symmetric encryption scheme  $(\{0, 1\}^k, \mathcal{D}, \mathcal{N}, \text{Enc}, \text{Dec})$  is **secure against key recovery under chosen plaintext attacks (CPA)** if for any PPT algorithm  $\mathcal{A}$ , the advantage  $\text{Adv}$  is negligible.

$$\text{Adv} = \Pr[\text{game returns 1}]$$

### Game

- 1:  $K \xleftarrow{\$} \{0, 1\}^k$
- 2:  $\text{Used} \leftarrow \emptyset$
- 3:  $\mathcal{A}^{\text{OEnc}} \xrightarrow{\$} K'$  adversary allowed to query an oracle which knows the true key CPA part of the game
- 4: **return**  $1_{K=K'}$

### Oracle OEnc( $\mathcal{N}, X$ ):

- 5: **if**  $N \in \text{Used}$  **then return**  $\perp$  ▷ nonce-respecting: cannot reuse  $N$
- 6:  $\text{Used} \leftarrow \text{Used} \cup \{N\}$
- 7: **return**  $\text{Enc}(K, N, X)$

# Adaptive Security against Key Recovery

## Definition

A symmetric encryption scheme  $(\{0, 1\}^k, \mathcal{D}, \mathcal{N}, \text{Enc}, \text{Dec})$  is **secure against key recovery under chosen plaintext/ciphertext attacks (CPCA)** if for any PPT algorithm  $\mathcal{A}$ , the advantage  $\text{Adv}$  is negligible.

$$\text{Adv} = \Pr[\text{game returns 1}]$$

### Game

- 1:  $K \xleftarrow{\$} \{0, 1\}^k$
- 2: **Used**  $\leftarrow \emptyset$
- 3:  $\mathcal{A}^{\text{OEnc}, \text{ODec}} \rightarrow K'$
- 4: **return**  $1_{K=K'}$

### Oracle OEnc( $\mathcal{N}, X$ ):

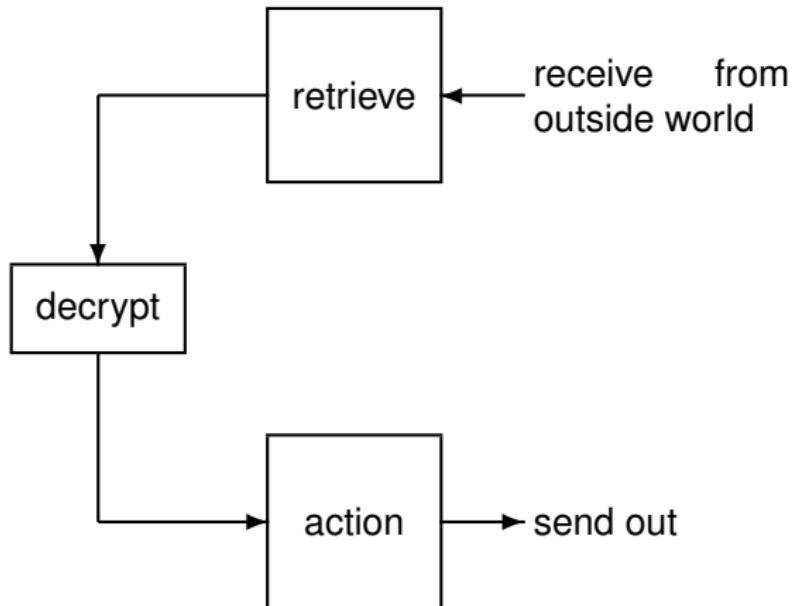
- 5: **if**  $N \in \text{Used}$  **then return**  $\perp$
- 6: **Used**  $\leftarrow \text{Used} \cup \{N\}$
- 7: **return**  $\text{Enc}(K, N, X)$

### Oracle ODec( $\mathcal{N}, Y$ ):

- 1: **return**  $\text{Dec}(K, N, Y)$

# Chosen Ciphertext Security: Motivation

- decryption device can be a freely available black box
- decryption device takes action after receiving external info



# CPCA Security is Stronger than CPA Security

- assume we have CPCPA security
- to prove CPA security, consider a CPA adversary  $\mathcal{A}$
- we define a CPCPA adversary  $\mathcal{B} = \mathcal{A}$   
(same adversary who just never use decryption queries)
- $\mathcal{B}$  and  $\mathcal{A}$  have the same advantage
- since the one of  $\mathcal{B}$  is negligible, the one of  $\mathcal{A}$  as well

□

CPA-breaking  $\implies$  CPCPA-breaking

CPCA-secure  $\implies$  CPA-secure

# Not Good Enough Security

Not good enough security: what follows is correct and secure!

- $\text{Enc}(K, N, X) = X$
- $\text{Dec}(K, N, Y) = Y$
- take  $\mathcal{A}$  playing the CPCA game
- reduce to  $\mathcal{B}$  not using the oracle with same advantage
  - simulate  $\mathcal{A}$
  - simulate  $\text{ODec}(N, Y) = Y$
  - simulate  $\text{OEnc}$  by simulating  $\text{Enc}(K, N, X) = X$
- the advantage of  $\mathcal{B}$  must be  $2^{-k}$

# Not Good Enough Security — Proof

$\Gamma_1$ :

- 1:  $K \xleftarrow{\$} \{0, 1\}^k$
- 2:  $\text{Used} \leftarrow \emptyset$
- 3:  $\mathcal{A}^{\text{OEnc}, \text{ODec}} \rightarrow K'$
- 4: **return**  $1_{K=K'}$

Oracle OEnc( $N, X$ ):

- 5: **if**  $N \in \text{Used}$  **then return**  $\perp$
- 6:  $\text{Used} \leftarrow \text{Used} \cup \{N\}$
- 7: **return** Enc( $K, N, X$ )

Oracle ODec( $N, Y$ ):

- 8: **return** Dec( $K, N, Y$ )

→

$\Gamma_2$ :

- 1:  $K \xleftarrow{\$} \{0, 1\}^k$
- 2:  $\text{Used} \leftarrow \emptyset$
- 3:  $\mathcal{A}^{\text{OEnc}, \text{ODec}} \rightarrow K'$
- 4: **return**  $1_{K=K'}$

Oracle OEnc( $N, X$ ):

- 5: **if**  $N \in \text{Used}$  **then return**  $\perp$
- 6:  $\text{Used} \leftarrow \text{Used} \cup \{N\}$
- 7: **return**  $X$

Oracle ODec( $N, Y$ ):

- 8: **return**  $Y$

simulate what's in red in  $\Gamma_2$

←

$\Gamma_3$ :

- 1:  $K \xleftarrow{\$} \{0, 1\}^k$
- 2:  $\mathcal{B} \rightarrow K'$
- 3: **return**  $1_{K=K'}$

$$\Pr[\Gamma_1 \rightarrow 1] = \Pr[\Gamma_2 \rightarrow 1] = \Pr[\Gamma_3 \rightarrow 1] = 2^{-k}$$

# Security against Decryption

## Definition

A symmetric encryption scheme  $(\{0, 1\}^k, \mathcal{D}, \mathcal{N}, \text{Enc}, \text{Dec})$  is **secure against decryption** under CPA resp. CPCA if for any PPT algorithm  $\mathcal{A}$ , the advantage  $\text{Adv}$  is negligible.

$$\text{Adv} = \Pr[\text{game returns } 1]$$

Game

- 1:  $K \xleftarrow{\$} \{0, 1\}^k$
- 2:  $X_0 \xleftarrow{\$} \mathcal{D}, N_0 \xleftarrow{\$} \mathcal{N}$
- 3:  $Y_0 \leftarrow \text{Enc}(K, N_0, X_0)$
- 4:  $\text{Used} \leftarrow \{N_0\}$
- 5:  $\mathcal{A}^{\text{OEnc}, \text{ODec}}(N_0, Y_0) \rightarrow X$
- 6: **return**  $1_{X=X_0}$

Oracle OEnc( $N, X$ ):

- 1: **if**  $N \in \text{Used}$  **then return**  $\perp$
- 2:  $\text{Used} \leftarrow \text{Used} \cup \{N\}$
- 3: **return**  $\text{Enc}(K, N, X)$

Oracle ODec( $N, Y$ ):

- 4: **if**  $(N, Y) = (N_0, Y_0)$  **then return**  $\perp$
- 5: **return**  $\text{Dec}(K, N, Y)$

# Decryption Security is Stronger than Key Recovery Security

- we take a scheme with(out) nonce
- assume we have decryption security and  $(\#\mathcal{N})^{-1} = \text{negl}(s)$
- to prove key-recovery security, consider a key recovery adversary  $\mathcal{A}$
- we define a decryption adversary  $\mathcal{B}$  as follows

$$\mathcal{B}(N_0, Y_0) : P_n[N_0 = N_1 \vee \dots \vee N_q = N_1] \leq P_n[N_0 = N_1] + \dots + P_n[N_q = N_1]$$

1: run  $\mathcal{A} \rightarrow \kappa$

$N_1, \dots, N_q$

(if  $\mathcal{A}$  queries  $N_0$  then abort)  $\triangleright$  happens with  $\Pr \leq \frac{q}{\#\mathcal{N}}$



2: compute  $X = \text{Dec}(\kappa, N_0, Y_0)$

3: return  $X$

- $\Pr[\mathcal{B} \text{ wins}] \geq \Pr[\mathcal{A} \text{ wins}] - \frac{q}{\#\mathcal{N}}$
- we know that  $\Pr[\mathcal{B} \text{ wins}] = \text{negl}(s)$  (due to assumption)
- hence,  $\Pr[\mathcal{A} \text{ wins}] = \text{negl}(s)$



## Note: No Decryption Security over Small Domains

- Consider

$\mathcal{A}(N_0, Y_0)$  :

1: pick  $X \xleftarrow{\$} \mathcal{D}$

2: **return**  $X$

- The advantage is  $\frac{1}{\#\mathcal{D}}$

- Therefore, if we have decryption security, then  $\frac{1}{\#\mathcal{D}} = \text{negl}$

# Not Good Enough Security

- some parts of the plaintext may be more private than others  
how about a cipher letting half of the plaintext in clear and strongly encrypting the other half?  
it would be secure against decryption

$$\begin{aligned}\text{Enc}(K, N, X) &= \text{Enc}_0(K, N, \text{lefthalf}(X)) \parallel \text{righthalf}(X) \\ \text{Dec}(K, N, Y) &= \text{Dec}_0(K, N, \text{lefthalf}(Y)) \parallel \text{righthalf}(Y)\end{aligned}$$

$$\begin{array}{c} (\{0, 1\}^k, \mathcal{D}, \mathcal{N}, \text{Enc}_0, \text{Dec}_0) \text{ Dec-secure} \\ \Downarrow \\ (\{0, 1\}^k, \mathcal{D}^2, \mathcal{N}, \text{Enc}, \text{Dec}) \text{ Dec-secure} \end{array}$$

# The Ideal Cipher

- the “ideal cipher”: taking  $K$  random is equivalent to picking a random length-preserving permutation  $\Pi_N$  for every  $N$

$$\begin{aligned}\text{Enc}(K, N, X) &= \Pi_N(X) \\ \text{Dec}(K, N, Y) &= \Pi_N^{-1}(Y)\end{aligned}$$

- security would mean that we cannot tell the real cipher and the ideal one apart from a black-box usage

# Security against Distinguisher (Real or Ideal)

## Definition

A symmetric encryption scheme  $(\{0,1\}^k, \mathcal{D}, \mathcal{N}, \text{Enc}, \text{Dec})$  is **secure against distinguishers** under CPA resp. CPCA if for any PPT algorithm  $\mathcal{A}$ , the advantage  $\text{Adv}$  is negligible.

$b = 0 \Rightarrow \text{ideal cyph}$

$$\text{Adv} = \Pr[\Gamma_1 \text{ returns } 1] - \Pr[\Gamma_0 \text{ returns } 1]$$

Game  $\Gamma_b$

- 1:  $K \xleftarrow{\$} \{0,1\}^k$
- 2: for every  $N$ , pick a length-preserving permutation  $\Pi_N$  over  $\mathcal{D}$
- 3:  $\text{Used} \leftarrow \emptyset$
- 4:  $\mathcal{A}^{\text{OEnc}, \text{ODec}} \rightarrow z$
- 5: **return**  $z$

Oracle  $\text{OEnc}(N, X)$ :

- 1: **if**  $N \in \text{Used}$  **then return**  $\perp$
- 2:  $\text{Used} \leftarrow \text{Used} \cup \{N\}$
- 3: **if**  $b = 0$  **then return**  $\Pi_N(X)$
- 4: **return**  $\text{Enc}(K, N, X)$

Oracle  $\text{ODec}(N, Y)$ :

- 5: **if**  $b = 0$  **then return**  $\Pi_N^{-1}(Y)$
- 6: **return**  $\text{Dec}(K, N, Y)$

# Distinguisher Security is Stronger than Decryption Security

- assume we have distinguisher security and  $\frac{1}{\#\mathcal{D}} = \text{negl}$
- to prove decryption security, consider a decryption adversary  $\mathcal{A}$
- we define a distinguisher  $\mathcal{B}$  as follows

$\mathcal{B} :$

- $X_0 \xleftarrow{\$} \mathcal{D}, N_0 \xleftarrow{\$} \mathcal{N}$
- $Y_0 \leftarrow \text{OEnc}(N_0, X_0)$
- run  
 $\mathcal{A}^{\text{OEnc}, \text{ODec}'}(N_0, Y_0) \rightarrow X$

4: **return**  $1_{X_0=X}$

$\text{ODec}'(N, Y) :$

- if**  $(N, Y) = (N_0, Y_0)$  **then return**  $\perp$
- return**  $\text{ODec}(N, Y)$

- $\Pr[\Gamma_1^{\mathcal{B}} \rightarrow 1] = \Pr[\Gamma^{\text{Dec}}(\mathcal{A}, \text{real cipher}) \rightarrow 1] = \text{Adv}_{\mathcal{A}}$
- $\Pr[\Gamma_0^{\mathcal{B}} \rightarrow 1] = \Pr[\Gamma^{\text{Dec}}(\mathcal{A}, \text{ideal cipher}) \rightarrow 1]$
- $\Pr[\Gamma^{\text{Dec}}(\mathcal{A}, \text{ideal cipher}) \rightarrow 1] = \text{negl}$  (see next slide)
- $\text{Adv}_{\mathcal{B}} = \text{negl}$
- hence,  $\text{Adv}_{\mathcal{A}} = \text{negl}$



# Detail in the Ideal Cipher Case

(no nonce for simplicity, CPA for simplicity)

$$\begin{aligned} & \Pr[\mathcal{A}^{\Pi(\cdot)}(\Pi(X)) = X] \\ = & \Pr[\mathcal{A}^{\Pi(\cdot)}(Y) = \Pi^{-1}(Y)] \quad (\text{where } Y \text{ is random}) \\ & (\text{wonder if } Y \text{ is answered to any query by the oracle or not}) \\ = & \Pr[\mathcal{A}^{\Pi(\cdot)}(Y) = \Pi^{-1}(Y), Y \text{ not answered}] + \Pr[\mathcal{A}^{\Pi(\cdot)}(Y) = \Pi^{-1}(Y), Y \text{ answered}] \\ \leq & \Pr[\mathcal{A}^{\Pi(\cdot)}(Y) = \Pi^{-1}(Y), Y \text{ not answered}] + \Pr[Y \text{ answered}] \\ = & \Pr[\mathcal{A}^{\Pi(\cdot)}(Y) = \Pi^{-1}(Y)|Y \text{ not answered}] \Pr[Y \text{ not answered}] + \Pr[Y \text{ answered}] \\ \leq & \Pr[\mathcal{A}^{\Pi(\cdot)}(Y) = \Pi^{-1}(Y)|Y \text{ not answered}] + \Pr[Y \text{ answered}] \\ \leq & \frac{1}{\#\mathcal{D} - q} + \Pr[Y \text{ answered}] \\ & \quad \underbrace{\phantom{\Pr[Y \text{ answered}]}}_{\text{number of queries}} \\ = & \frac{1}{\#\mathcal{D} - q} + \Pr\left[\bigvee_{i=1}^q Y \text{ answered to } i\text{th fresh query}\right] \\ \leq & \frac{1}{\#\mathcal{D} - q} + \sum_{i=1}^q \Pr[Y \text{ answered to } i\text{th fresh query}] \\ = & \frac{1}{\#\mathcal{D} - q} + \sum_{i=0}^{q-1} \frac{1}{\#\mathcal{D} - i} \leq \frac{q+1}{\#\mathcal{D} - q} \leq \text{negl}(s) \end{aligned}$$

# Security Notions

|             | <b>key recovery</b> | <b>decryption</b> | <b>distinguisher</b> |
|-------------|---------------------|-------------------|----------------------|
| <b>CPA</b>  | weakest security    |                   |                      |
| <b>CPCA</b> |                     |                   | strongest security   |

- if we can recover the key, we can decrypt
- if we can decrypt, we can recognize from the ideal cipher
- if we can break without chosen ciphertext, we can also break with

## Note: Another Distinguisher Style

Game  $\Gamma'$ :

$$\forall \mathcal{A} \quad \text{Adv}_{\mathcal{A}}^{\Gamma'} = \frac{1}{2} \text{Adv}_{\mathcal{A}}^{\Gamma}$$

- 1: pick  $b \in \{0, 1\}$  Secure notion 1:  $\forall \mathcal{A} \quad \text{Adv}_{\mathcal{A}}^{\Gamma'} = \text{negl}$
- 2: run  $\Gamma_b \rightarrow z$  Secure notion 2:  $\forall \mathcal{A} \quad \text{Adv}_{\mathcal{A}}^{\Gamma'} = \text{negl}$
- 3: **return**  $1_{b=z}$

$$\begin{aligned}\text{Adv}(\Gamma') &= \Pr[\Gamma' \rightarrow 1] - \frac{1}{2} \\ &= \Pr[\Gamma' \rightarrow 1 \wedge b = 1] + \Pr[\Gamma' \rightarrow 1 \wedge b = 0] - \frac{1}{2} \\ &= \Pr[b = 1] \Pr[\Gamma_1 \rightarrow 1] + \Pr[b = 0] \Pr[\Gamma_0 \rightarrow 0] - \frac{1}{2} \\ &= \frac{1}{2} \Pr[\Gamma_1 \rightarrow 1] + \frac{1}{2}(1 - \Pr[\Gamma_0 \rightarrow 1]) - \frac{1}{2} \\ &= \frac{1}{2} \text{Adv}(\Gamma)\end{aligned}$$

(most common construction)

## Definition

A **message authentication code** is a tuple  $(\{0, 1\}^k, \mathcal{D}, \{0, 1\}^\tau, \text{MAC})$  with a key domain  $\{0, 1\}^k$ , a message domain  $\mathcal{D} \subseteq \{0, 1\}^*$ , an output domain  $\{0, 1\}^\tau$ , and one polynomially bounded deterministic algorithm MAC implementing a function

$$\begin{aligned}\text{MAC} : \quad & \{0, 1\}^k \times \mathcal{D} \quad \longrightarrow \quad \{0, 1\}^\tau \\ & (K, X) \quad \longmapsto \quad \text{MAC}_K(X)\end{aligned}$$

# Using MAC for Authentication

- **usage:**

$$\text{Auth}_K(X) = X \parallel \text{MAC}_K(X)$$

$$\text{Check}_K(X \parallel t) = (X, 1_{t=\text{MAC}_K(X)})$$

- **correctness:**

$$\text{Check}_K(\text{Auth}_K(X)) = (X, 1)$$

- **security:** notion of authentication and integrity

# Security against Key Recovery

## Definition

A message authentication code  $(\{0, 1\}^k, \mathcal{D}, \{0, 1\}^\tau, \text{MAC})$  is **secure against key recovery under chosen message attacks** if for any PPT algorithm  $\mathcal{A}$ , the advantage  $\text{Adv}$  is negligible.

$$\text{Adv} = \Pr[\text{game returns 1}]$$

### Game

- 1:  $K \xleftarrow{\$} \{0, 1\}^k$
- 2:  $\mathcal{A}^{\text{OMac}} \rightarrow K'$
- 3: **return**  $1_{K=K'}$

### Oracle OMac( $X$ ):

- 4: **return**  $\text{Mac}(K, X)$

(+ similar notion with **known message attacks**)

# Security against Forgery

## Definition

A message authentication code  $(\{0, 1\}^k, \mathcal{D}, \{0, 1\}^\tau, \text{MAC})$  is **secure against forgery under chosen message attacks** if for any PPT algorithm  $\mathcal{A}$ , the advantage  $\text{Adv}$  is negligible.

$$\text{Adv} = \Pr[\text{game returns 1}]$$

Game

- 1:  $K \xleftarrow{\$} \{0, 1\}^k$
- 2:  $\text{Queried} \leftarrow \emptyset$
- 3:  $\mathcal{A}^{\text{OMac}} \rightarrow (X, t)$
- 4: **if**  $X \in \text{Queried}$  **then**  
**return** 0
- 5: **return** 1 <sub>$\text{Mac}(K, X) = t$</sub>

Oracle OMac( $X$ ):

- 6:  $\text{Queried} \leftarrow \text{Queried} \cup \{X\}$
- 7: **return**  $\text{Mac}(K, X)$

# Existential vs Universal

- **universal forgery:**

adversary is

able to forge a valid MAC/signature for *an arbitrary message*

Game

$$1: K \xleftarrow{\$} \{0, 1\}^k$$

$$2: X_0 \xleftarrow{\$} \mathcal{D}$$

$$3: \mathcal{A}^{\text{OMac}}(X_0) \rightarrow t$$

$$4: \text{return } \mathbf{1}_{\text{Mac}(K, X_0)=t}$$

Oracle  $\text{OMac}(X)$ :

1: **if**  $X = X_0$  **then return**  $\perp$

2: **return**  $\text{Mac}(K, X)$

- **existential forgery:**

adversary is able to forge a valid MAC/signature for *a new message* of his choice (previous slide)

# Security against Distinguisher (PRF)

## Definition

A message authentication code  $(\{0, 1\}^k, \mathcal{D}, \{0, 1\}^\tau, F)$  is a **pseudorandom function (PRF)** if for any PPT algorithm  $\mathcal{A}$ , the advantage  $\text{Adv}$  is negligible.

$$\text{Adv} = \Pr[\Gamma_1 \text{ returns } 1] - \Pr[\Gamma_0 \text{ returns } 1]$$

Game  $\Gamma_b$

- 1:  $K \xleftarrow{\$} \{0, 1\}^k$
- 2: pick  $F^* : \mathcal{D} \rightarrow \{0, 1\}^\tau$
- 3:  $\mathcal{A}^O \rightarrow z$
- 4: **return**  $z$

Oracle  $O(X)$ :

- 1: **if**  $b = 0$  **then return**  $F^*(X)$
- 2: **return**  $F(K, X)$

Note: when talking about MAC, security usually refers to unforgeability. Otherwise, we talk about PRF.

(for comparison of these notions: see midterm exam 2016–17 ex2)

# Security Relations

(Exercise: prove it like what was done for encryption!)

- EF-secure  $\implies$  UF-secure  $\implies$  key recovery-secure
- $(\text{PRF} \wedge 2^{-\tau} = \text{negl}) \implies \text{EF-secure}$

# Key Agreement Protocol

- **components:**
  - domain parameters: security parameter  $s$ , domain for  $K$
  - two characters: Alice and Bob
  - one protocol (two probabilistic algorithms):  $A$  and  $B$  with no common secret input;  $A \rightarrow K_A$ ,  $B \rightarrow K_B$
- **functionality:**

running  $A(1^s; r_a)$  and  $B(1^s; r_b)$  together lead to the same output  $K_A = K_B = K$  on both sides
- **security:** (against passive attacks)  
→ next slide

# Security of Key Agreement Protocol

$$A(1^s; r_a) \longleftrightarrow B(1^s; r_b)$$

should protect against **passive** attacks:

- **key recovery**: (next slide)
- **key distinguisher**: (next slide)

**active** attacks:

- **man-in-the-middle**:  $A \longleftrightarrow E$  gives  $K_A$  and  $E \longleftrightarrow B$  gives  $K_B$   
(always possible, unavoidable)
- more devastating:  $A \longleftrightarrow E \longleftrightarrow B$  making  $K_A = K_B$  known by  $E$   
(should be avoided)

# Security against Passive Attacks

transcript = exchanges in  $A(1^s; r_a) \leftrightarrow B(1^s; r_b)$

- **key recovery:**  $\text{Adv} = \Pr[\text{Game returns 1}]$

Game

- 1: pick  $r_a, r_b$
- 2: execute  $A(1^s; r_a) \leftrightarrow B(1^s; r_b)$
- 3: get transcript and  $K$
- 4: run  $\mathcal{A}(1^s, \text{transcript}) \xrightarrow{\$} K'$
- 5: **return**  $1_{K=K'}$

- **key distinguisher:**

$\text{Adv} = \Pr[\Gamma_1 \text{ returns 1}] - \Pr[\Gamma_0 \text{ returns 1}]$

Game  $\Gamma_b$

- 1: pick  $r_a, r_b$
- 2: execute  $A(1^s; r_a) \leftrightarrow B(1^s; r_b)$
- 3: get transcript and  $K_1$
- 4: pick  $K_0$  of same length as  $K_1$  at random
- 5: run  $\mathcal{A}(1^s, \text{transcript}, K_b) \xrightarrow{\$} z$
- 6: **return**  $z$

## Definition

A **public-key cryptosystem** is a tuple  $(\text{Gen}, \mathcal{M}, \text{Enc}, \text{Dec})$  with a plaintext domain  $\mathcal{M}$  and three polynomially bounded algorithms  $\text{Gen}$ ,  $\text{Enc}$ , and  $\text{Dec}$ . The algorithm  $\text{Dec}$  is deterministic and outputs either something in  $\mathcal{M}$  or an error  $\perp$ . It is such that

$$\forall \text{pt} \in \mathcal{M} \quad \Pr_{r_g, r_e} [\text{Dec}(\text{sk}, \text{Enc}(\text{pk}, \text{pt}; r_e)) = \text{pt}] = 1$$

where  $(\text{pk}, \text{sk}) = \text{Gen}(1^s; r_g)$ .

# Threat Models

Adversary model: can always encrypt chosen plaintexts (enc key is public), may access to a decryption oracle, ...

- **Key recovery:** an adversary can recover the secret key
- **Decryption:** an adversary can decrypt a *random* ciphertext
- **Information recovery:** an adversary can infer a given bit of information on the plaintext given a random ciphertext
- **Distinguishability:** an adversary can recognize whether a ciphertext encrypts  $pt_0$  or  $pt_1$  for some  $pt_0$  and  $pt_1$  of her choice

# Hard Core Bit

## Definition

Let  $R$  be a predicate over a domain  $\mathcal{M}$ . We say that  $R$  is a **hard-core bit** for a PKC  $(\text{Gen}, \mathcal{M}, \text{Enc}, \text{Dec})$  if for any PPT adversary  $\mathcal{A}$ , the advantage  $\text{Adv}$  is negligible.

*only for bits  
that are balanced.*       $\text{Adv} = 2 \Pr[\text{game returns } 1] - 1$

## Game

- 1:  $\text{Gen} \xrightarrow{\$} (\text{pk}, \text{sk})$
- 2:  $\text{pt} \xleftarrow{\$} \mathcal{M}$
- 3:  $\text{ct} \xleftarrow{\$} \text{Enc}(\text{pk}, \text{pt})$
- 4:  $\mathcal{A}(\text{pk}, \text{ct}) \xrightarrow{\$} z$
- 5: **return**  $1_{z=R(\text{pt})}$

# Security against Distinguisher (Left or Right)

## Definition

A PKC  $(\text{Gen}, \mathcal{M}, \text{Enc}, \text{Dec})$  is **secure under chosen plaintext attacks** (IND-CPA-secure) if for any interactive PPT process  $(\mathcal{A}_1, \mathcal{A}_2)$ , the advantage  $\text{Adv}$  is negligible.

$$\text{Adv} = \Pr[\Gamma_1 \text{ returns } 1] - \Pr[\Gamma_0 \text{ returns } 1]$$

Game  $\Gamma_b$

- 1:  $\text{Gen} \xrightarrow{\$} (\text{pk}, \text{sk})$  with the knowledge of  $\text{pk}$
- 2:  $\mathcal{A}_1(\text{pk}) \xrightarrow{\$} (\text{pt}_0, \text{pt}_1, \text{st})$
- 3: **if**  $|\text{pt}_0| \neq |\text{pt}_1|$  **then return**  
    0
- 4:  $\text{ct} \xleftarrow{\$} \text{Enc}(\text{pk}, \text{pt}_b)$
- 5:  $\mathcal{A}_2(\text{st}, \text{ct}) \xrightarrow{\$} z$
- 6: **return**  $z$

# Remark: Two Styles of Interactive Adversaries

Two algorithms with state variable

Game  $\Gamma_b$

- 1:  $\text{Gen} \xrightarrow{\$} (\text{pk}, \text{sk})$
- 2:  $\mathcal{A}_1(\text{pk}) \xrightarrow{\$} (\text{pt}_0, \text{pt}_1, \text{st})$
- 3: **if**  $|\text{pt}_0| \neq |\text{pt}_1|$  **then return**  
    0
- 4:  $\text{ct} \xleftarrow{\$} \text{Enc}(\text{pk}, \text{pt}_b)$
- 5:  $\mathcal{A}_2(\text{st}, \text{ct}) \xrightarrow{\$} z$
- 6: **return**  $z$

Deterministic algorithm with variable number of inputs

Game  $\Gamma_b$

- 1: **pick**  $\rho$  at random
- 2:  $\text{Gen} \xrightarrow{\$} (\text{pk}, \text{sk})$
- 3:  $\mathcal{A}(\text{pk}; \rho) \xrightarrow{\$} (\text{pt}_0, \text{pt}_1)$
- 4: **if**  $|\text{pt}_0| \neq |\text{pt}_1|$  **then return**  
    0
- 5:  $\text{ct} \xleftarrow{\$} \text{Enc}(\text{pk}, \text{pt}_b)$
- 6:  $\mathcal{A}(\text{pk}, \text{ct}; \rho) \xrightarrow{\$} z$
- 7: **return**  $z$

# Problem with Deterministic Cryptosystems

- IND-CPA is a modern notion of security
- problem: if  $\text{Enc}$  is deterministic, then PKC is insecure!
- example: plain RSA not IND-CPA secure (since deterministic)
- modern PKC are probabilistic
- example: ElGamal cryptosystem (and variants) is IND-CPA secure

# Comments on Semantic Security

- Semantic security  $\approx$  all bits are hard core bits
- Semantic security  $\Leftrightarrow$  IND-CPA security

# Chosen Ciphertext Security

- Adversary in “lunch time” attack: CCA1  
Adversary can submit chosen ciphertexts before choosing  $pt_0$  and  $pt_1$ .  
→ *IND-CCA1*
- CCA (aka CCA2) Adversary: he can submit chosen ciphertexts even after (except ct)  
→ *IND-CCA*

# Adaptive Security against Distinguisher

## Definition

A PKC  $(\text{Gen}, \mathcal{M}, \text{Enc}, \text{Dec})$  is **secure under chosen ciphertext attacks** (IND-CCA-secure) if for any interactive PPT process  $(\mathcal{A}_1, \mathcal{A}_2)$ , the advantage  $\text{Adv}$  is negligible.

$$\text{Adv} = \Pr[\Gamma_1 \text{ returns } 1] - \Pr[\Gamma_0 \text{ returns } 1]$$

Game  $\Gamma_b$

- 1:  $\text{Gen} \xrightarrow{\$} (\text{pk}, \text{sk})$
- 2:  $\mathcal{A}_1 \xrightarrow{\text{ODec}_1} (\text{pk}) \xrightarrow{\$} (\text{pt}_0, \text{pt}_1, \text{st})$
- 3: **if**  $|\text{pt}_0| \neq |\text{pt}_1|$  **then return**  
    0
- 4:  $\text{ct}^* \xleftarrow{\$} \text{Enc}(\text{pk}, \text{pt}_b)$
- 5:  $\mathcal{A}_2 \xrightarrow{\text{ODec}_2} (\text{st}, \text{ct}^*) \xrightarrow{\$} z$
- 6: **return**  $z$

Oracle  $\text{ODec}_1(\text{ct})$ :

- 7: **return**  $\text{Dec}(\text{sk}, \text{ct})$

Oracle  $\text{ODec}_2(\text{ct})$ :

- 8: **if**  $\text{ct} = \text{ct}^*$  **then return**  $\perp$
- 9: **return**  $\text{Dec}(\text{sk}, \text{ct})$

# Non-Malleability

## Definition (Malleability (intuitively))

There is an  $R$ , a samplable distribution  $D$ , and an adversary who given  $\text{ct} = \text{Enc}(\text{pt})$  for some unknown random  $\text{pt}$  (sampled following  $D$ ) can forge  $\text{ct}'$  such that  $R(\text{pt}, \text{Dec}(\text{ct}'))$  “in a nontrivial way”.

Example: in a regular stream cipher, if  $R(a, b) = 1_{a \oplus b = \delta}$  and  $D$  is any distribution, an adversary can compute  $\text{Dec}(\text{ct}') = \text{Dec}(\text{ct}) \oplus \delta$  so it is malleable

## Theorem

*IND-CCA security and non-malleability under chosen ciphertext attacks are equivalent.*

# Plaintext Awareness

“For every  $\mathcal{A}$ , there exists  $\mathcal{E}$  such that if  $\mathcal{A}$  can forge a valid ciphertext  $ct$ , then  $\mathcal{E}$  with the same view gives its decryption.”

There exist several flavors of PA-security

# Security Notions

|            | <b>key recovery</b> | <b>decryption</b> | <b>distinguisher</b> |
|------------|---------------------|-------------------|----------------------|
| <b>CPA</b> | weakest security    |                   |                      |
| <b>CCA</b> |                     |                   | strongest security   |

- if we can recover the key, we can decrypt
- if we can decrypt, we can recognize from the ideal cipher
- if we can break without chosen ciphertext, we can also break with

# Signature Scheme

## Definition

A **digital signature scheme** is a tuple  $(\text{Gen}, \mathcal{D}, \text{Sig}, \text{Ver})$  with a message domain  $\mathcal{D} \subseteq \{0, 1\}^*$  and three PPT algorithms Gen, Sig, and Ver. The algorithm Ver is deterministic and outputs 0 (reject) or 1 (accept). It is such that

$$\forall X \in \mathcal{D} \quad \Pr_{r_g, r_s} [\text{Ver}(\text{pk}, X, \text{Sig}(\text{sk}, X; r_s)) = 1] = 1$$

where  $(\text{pk}, \text{sk}) = \text{Gen}(1^s; r_g)$ .

(could also define signature schemes with message recovery)

# Threat Models

- **Total break:** an adversary can recover the secret key
- **Universal forgery:** an adversary can forge the signature of *any* or a *random* message
- **Existential forgery:** an adversary can forge a valid message-signature pair

(same as for MAC)

Adversary model: may intercept signatures (known message attack), may access to a signing oracle (chosen message attack), ...

# EF-CMA Security

## Definition

A digital signature scheme  $(\text{Gen}, \mathcal{D}, \text{Sig}, \text{Ver})$  is **secure against existential forgery under chosen message attacks** (EF-CMA) if for any PPT  $\mathcal{A}$ , the advantage  $\text{Adv}$  is negligible.

$$\text{Adv} = \Pr[\text{game returns 1}]$$

### Game

- 1:  $\text{Gen} \xrightarrow{\$} (\text{pk}, \text{sk})$
- 2:  $\text{Queries} \leftarrow \emptyset$
- 3:  $\mathcal{A}^{\text{OSig}}(\text{pk}) \rightarrow (X, \sigma)$
- 4: **if**  $X \in \text{Queries}$  **then**  
**return** 0
- 5: **return**  $1_{\text{Ver}(\text{pk}, X, \sigma)}$

### Oracle $\text{OSig}(X)$ :

- 6:  $\sigma \leftarrow \text{Sig}(\text{sk}, X)$
- 7:  $\text{Queries} \leftarrow \text{Queries} \cup \{X\}$
- 8: **return**  $\sigma$

# (Strong) EF-CMA Security

## Definition

A digital signature scheme  $(\text{Gen}, \mathcal{D}, \text{Sig}, \text{Ver})$  is **strongly secure against existential forgery under chosen message attacks** (strong EF-CMA) if for any PPT  $\mathcal{A}$ , the advantage  $\text{Adv}$  is negligible.

$$\text{Adv} = \Pr[\text{game returns } 1]$$

### Game

- 1:  $\text{Gen} \xrightarrow{\$} (\text{pk}, \text{sk})$
- 2:  $\text{Queries} \leftarrow \emptyset$
- 3:  $\mathcal{A}^{\text{OSig}}(\text{pk}) \rightarrow (X, \sigma)$
- 4: **if**  $(X, \sigma) \in \text{Queries}$  **then**  
**return** 0
- 5: **return**  $1_{\text{Ver}(\text{pk}, X, \sigma)}$

### Oracle $\text{OSig}(X)$ :

- 6:  $\sigma \leftarrow \text{Sig}(\text{sk}, X)$
- 7:  $\text{Queries} \leftarrow \text{Queries} \cup \{(X, \sigma)\}$
- 8: **return**  $\sigma$

# Security Models

Universal Forgery (UF):  $X$  is selected at random by the game

No Message Attack (0MA): no OSig available

Known Message Attack (KMA): input to OSig is random

|             |               | adversary power |     |                 |
|-------------|---------------|-----------------|-----|-----------------|
|             |               | 0MA             | KMA | CMA             |
| total break | weakest model |                 |     |                 |
|             | UF            |                 |     |                 |
| EF          |               |                 |     |                 |
| strong EF   |               |                 |     | strongest model |

Note: stronger security means security against weaker attacks...

# Exercise

write formal definitions for the security notions of other primitives and prove implications between them

## 1 The Cryptographic Zoo

## 2 Cryptographic Security Models

- Security Definitions
- The Game Proof Methodology
- RSA Security
- Rabin Cryptosystem
- Diffie-Hellman Security
- ElGamal Security

## 3 Cryptanalysis (Public-Key)

## 4 The Power of Interaction

## 5 Cryptanalysis (Conventional)

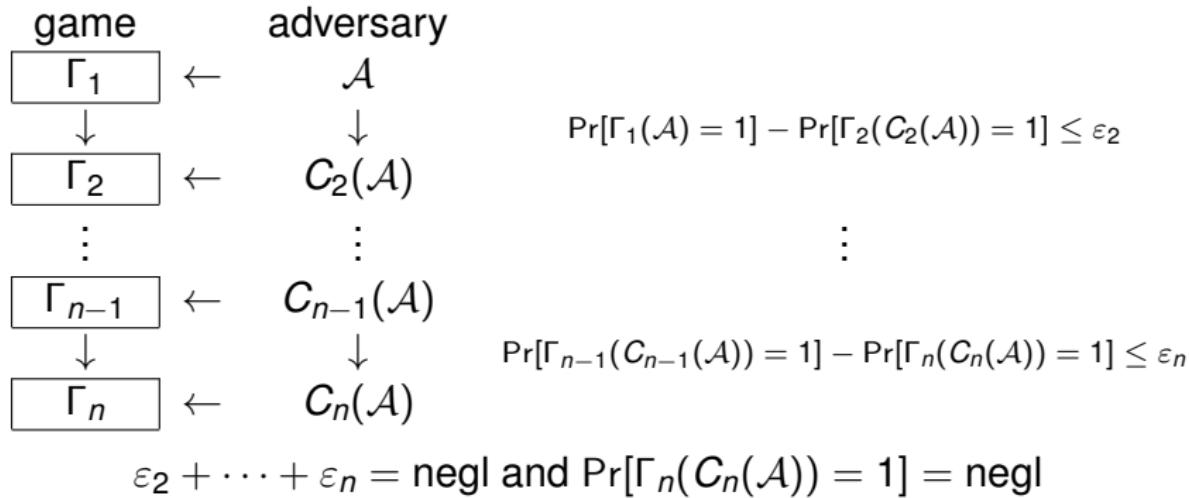
## 6 Proving Security

# Methodology

- security properties are defined by a **game**  
security parameter, different steps, rules, outcome
- assumptions: game runs in polynomial time
- **adversary**: strategy for the player of the game
- **advantage**: difference of the outcome of the adversary  
and of a trivial one
- security means that every adversary has negligible  
advantage

# Proof Methodology

build a sequence of games and adversary compilers so that the first game is the one from the security definition and the last one is a trivial one and such that the difference between the advantages are negligible



## Transition Tool 1: Indistinguishability

- consider a game  $\Gamma(X, \mathcal{A})$  in which we use a r.v.  $X$
- consider  $X$  and  $Y$  with indistinguishable distributions
- since  $X$  and  $Y$  are indistinguishable and  $\Gamma(\cdot, \mathcal{A})$  is computable in polynomial time,

$$\Pr[\Gamma(X, \mathcal{A}) = 1] - \Pr[\Gamma(Y, \mathcal{A}) = 1] = \text{negl}$$

## Transition Tool 2: Difference Lemma

- consider a game  $\Gamma$                        $\Gamma'(\mathcal{A})$ :
- $F$ : a (“failure”) event in  $\Gamma$         1:  $\Gamma(\mathcal{A}) \rightarrow z$  (check if  $F$  holds)
- ( $\Gamma$  becomes “simpler”  
when  $\neg F$  holds)                                  2: **return**  $1_{z=1 \wedge \neg F}$
- define  $\Gamma'$  like  $\Gamma$  with  $\neg F$  as  
an extra winning condition

### Lemma

$$|\Pr[\Gamma(\mathcal{A}) \rightarrow 1] - \Pr[\Gamma'(\mathcal{A}) \rightarrow 1]| \leq \Pr[F]$$

### Proof.

$$\begin{aligned} & \Pr[\Gamma(\mathcal{A}) \rightarrow 1] - \Pr[\Gamma'(\mathcal{A}) \rightarrow 1] \\ = & \Pr[\Gamma(\mathcal{A}) \rightarrow 1, F] + \Pr[\Gamma(\mathcal{A}) \rightarrow 1, \neg F] - \Pr[\Gamma'(\mathcal{A}) \rightarrow 1, \neg F] \\ = & \Pr[\Gamma(\mathcal{A}) \rightarrow 1, F] \\ \leq & \Pr[F] \end{aligned}$$

## Transition Tool 3: Bridging Step

- consider a game  $\Gamma$
- consider a game variant  $\Gamma'$  and a compiler  $C$  such that  $\Gamma(\mathcal{A})$  and  $\Gamma'(C(\mathcal{A}))$  produce the same distribution

$$\Pr[\Gamma(\mathcal{A}) \rightarrow 1] = \Pr[\Gamma'(C(\mathcal{A})) \rightarrow 1]$$

$\Gamma'$  is a kind of rewriting of  $\Gamma$

## Transition Tool 3: Bridging Step Examples

- case 0: permute two operations which are independent
- case 1: move operations between  $\Gamma$  and  $\mathcal{A}$  without affecting  $\Gamma(\mathcal{A})$
- case 2: replace a random oracle by a “gnome” performing the lazy sampling technique:

$\mathcal{O}(x)$ : ▷ uses an associative array  $T$

- 1: **if**  $T(x)$  is defined **then**
- 2:      $y \leftarrow T(x)$
- 3: **else**
- 4:     pick  $y$  at random
- 5:      $T(x) \leftarrow y$
- 6: **end if**
- 7: **return**  $y$

# Double Bridge

bridge  $\Gamma_\beta(\mathcal{A})$  to  $\Gamma'_\beta(C(\mathcal{A}))$

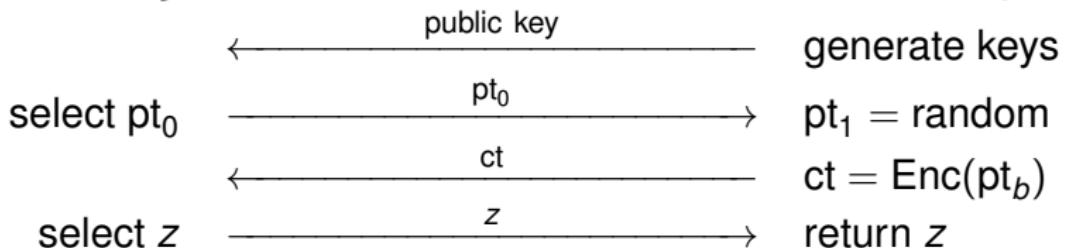
$$\begin{array}{lllll} \Gamma_0(\mathcal{A}): & \Gamma'_0(C(\mathcal{A})): & \Gamma'_1(C(\mathcal{A})): & \Gamma_1(\mathcal{A}): \\ \text{1: } \dots & \xrightarrow{\text{bridge}} \text{1: } \gamma_0 \rightarrow z & \approx \text{1: } \gamma_1 \rightarrow z & \xrightarrow{\text{bridge}} \text{1: } \dots \\ \text{2: return } b & \text{2: } C(\mathcal{A})(z) \rightarrow b' & \text{2: } C(\mathcal{A})(z) \rightarrow b' & \text{2: return } b' \\ & \text{3: return } b' & & \end{array}$$

$$\begin{array}{ccc} \Gamma_0 & \xrightarrow{\text{bridge}} & \Gamma'_0 \\ & \downarrow & \\ \Gamma_1 & \xrightarrow{\text{bridge}} & \Gamma'_1 \end{array}$$

↑ distinguisher between  $\gamma_0$  and  $\gamma_1$  (tool 1)

# IND\$-CPA: Real-or-Random IND-CPA Game

## Adversary



## Game $\Gamma_b$

- 1:  $\text{Gen} \xrightarrow{\$} (\text{pk}, \text{sk})$
- 2:  $\mathcal{A}_1(\text{pk}) \xrightarrow{\$} (\text{pt}_0, \text{st})$
- 3: pick  $\text{pt}_1$  s.t.  $|\text{pt}_0| = |\text{pt}_1|$
- 4:  $\text{ct} \xleftarrow{\$} \text{Enc}(\text{pk}, \text{pt}_b)$
- 5:  $\mathcal{A}_2(\text{st}, \text{ct}) \xrightarrow{\$} z$
- 6: **return**  $z$

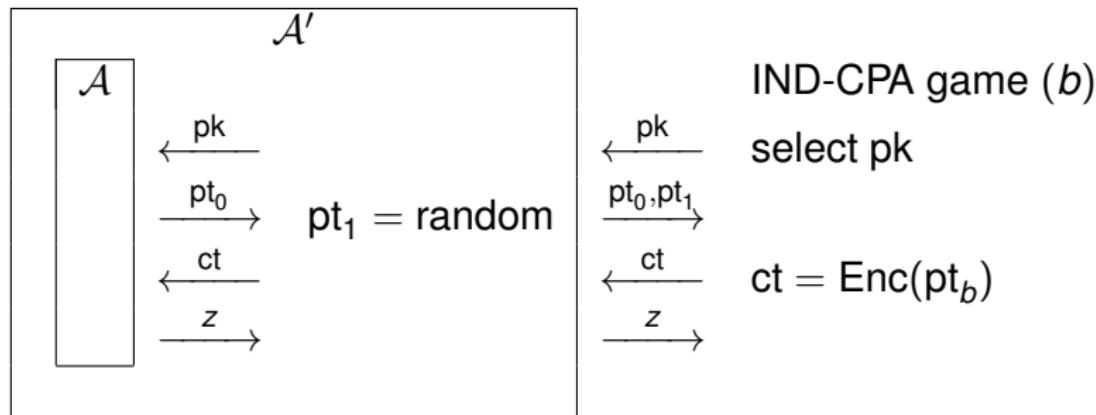
$\mathcal{A}$  works in two steps:  $\mathcal{A}_1, \mathcal{A}_2$   
a state variable  $\text{st}$  is kept

$$\text{Adv} = \Pr[\Gamma_1 \rightarrow 1] - \Pr[\Gamma_0 \rightarrow 1]$$

# Equivalence with IND-CPA Game — i

IND-CPA secure  $\Rightarrow$  IND\$-CPA secure:

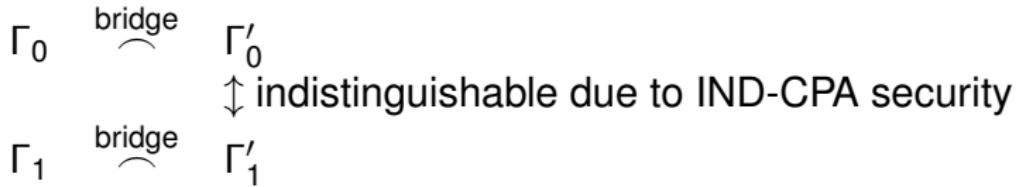
an adversary  $\mathcal{A}$  playing the IND\$-CPA game with advantage  $\varepsilon$   
can be transformed into an adversary  $\mathcal{A}'$  playing the IND-CPA  
game with advantage  $\varepsilon$



$$\text{Adv}_{\mathcal{A}'} = \Pr_{b=1} [z = 1] - \Pr_{b=0} [z = 1] = \text{Adv}_{\mathcal{A}} = \varepsilon$$

## Equivalence with IND-CPA Game — i (bis)

- $\Gamma_b$ :  $\mathcal{A}$  playing IND\$-CPA
- $\Gamma'_b$ :  $\mathcal{A}'$  playing IND-CPA
- $\Gamma_b(\mathcal{A})$  and  $\Gamma'_b(\mathcal{A}')$  are identical (bridge)
- actually, there is a double-bridge and IND-CPA security



# Equivalence with IND-CPA Game — i (ter)

$$\Pr \left[ \begin{bmatrix} \text{Gen} \rightarrow (\text{pk}, \text{sk}) \\ \mathcal{A}_1(\text{pk}) \rightarrow (\text{pt}_0, \text{st}) \\ \text{pick pt}_1 \\ \text{Enc}(\text{pk}, \text{pt}_0) \rightarrow \text{ct} \\ \mathcal{A}_2(\text{st}, \text{ct}) \rightarrow z \\ \text{return } z \end{bmatrix} \rightarrow 1 \right] \stackrel{\text{bridge}}{=} \Pr \left[ \begin{bmatrix} \text{Gen} \rightarrow (\text{pk}, \text{sk}) \\ \mathcal{A}_1(\text{pk}) \rightarrow (\text{pt}_0, \text{st}) \\ \text{pick pt}_1 \\ \text{Enc}(\text{pk}, \text{pt}_0) \rightarrow \text{ct} \\ \mathcal{A}_2(\text{st}, \text{ct}) \rightarrow z \\ \text{return } z \end{bmatrix} \rightarrow 1 \right]$$

IND-CPA<sub>b</sub>( $\mathcal{A}'$ )

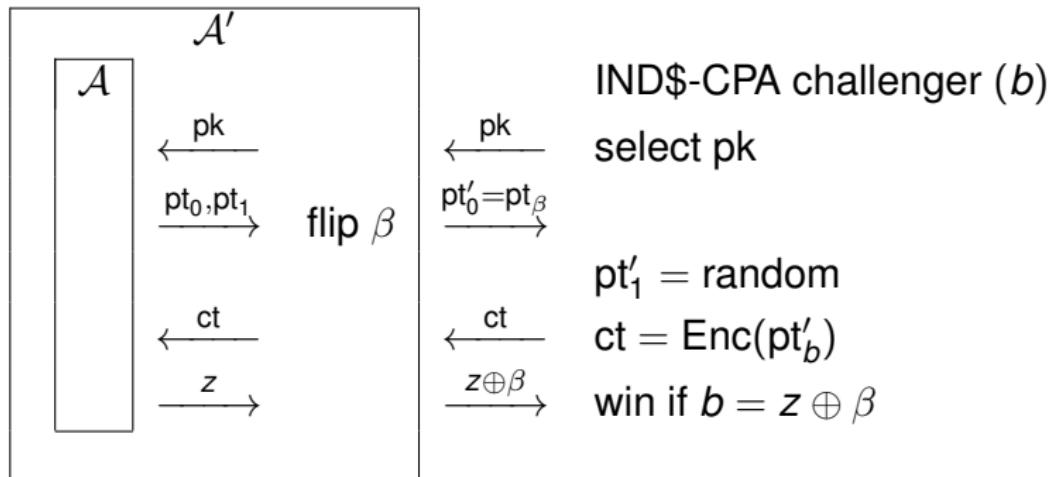
$$\Pr \left[ \begin{bmatrix} \text{Gen} \rightarrow (\text{pk}, \text{sk}) \\ \mathcal{A}_1(\text{pk}) \rightarrow (\text{pt}_0, \text{st}) \\ \text{pick pt}_1 \\ \text{Enc}(\text{pk}, \text{pt}_1) \rightarrow \text{ct} \\ \mathcal{A}_2(\text{st}, \text{ct}) \rightarrow z \\ \text{return } z \end{bmatrix} \rightarrow 1 \right] \stackrel{\text{bridge}}{=} \Pr \left[ \begin{bmatrix} \text{Gen} \rightarrow (\text{pk}, \text{sk}) \\ \mathcal{A}_1(\text{pk}) \rightarrow (\text{pt}_0, \text{st}) \\ \text{pick pt}_1 \\ \text{Enc}(\text{pk}, \text{pt}_1) \rightarrow \text{ct} \\ \mathcal{A}_2(\text{st}, \text{ct}) \rightarrow z \\ \text{return } z \end{bmatrix} \rightarrow 1 \right]$$

$\approx$  (IND-CPA)

## Equivalence with IND-CPA Game — ii

IND\$-CPA secure  $\implies$  IND-CPA secure:

an adversary  $\mathcal{A}$  playing the IND-CPA game with advantage  $\varepsilon$   
can be transformed into an adversary  $\mathcal{A}'$  playing the  
real-or-random game with advantage  $\frac{\varepsilon}{2}$



$$\text{Adv}_{\mathcal{A}'} = \dots = \frac{\varepsilon}{2}$$

# Equivalence with IND-CPA Game — ii (bis)

IND\$-CPA<sub>b=1</sub>( $\mathcal{A}'$ )

$$\Pr \left[ \begin{array}{l} \text{Gen} \rightarrow (\text{pk}, \text{sk}) \\ \mathcal{A}_1(\text{pk}) \rightarrow (\text{pt}_0, \text{pt}_1, \text{st}) \\ \text{pick } \beta \\ \text{pt}'_0 \leftarrow \text{pt}_\beta \\ \text{pick pt}'_1 \\ \text{Enc}(\text{pk}, \text{pt}'_1) \rightarrow \text{ct} \\ \mathcal{A}_2(\text{st}, \text{ct}) \rightarrow z \\ z' \leftarrow z \oplus \beta \\ \text{return } z' \end{array} \right] \rightarrow 1 \stackrel{\text{bridge}}{=} \Pr \left[ \begin{array}{l} \text{Gen} \rightarrow (\text{pk}, \text{sk}) \\ \mathcal{A}_1(\text{pk}) \rightarrow (\text{pt}_0, \text{pt}_1, \text{st}) \\ \text{pick pt}'_1 \\ \text{Enc}(\text{pk}, \text{pt}'_1) \rightarrow \text{ct} \\ \mathcal{A}_2(\text{st}, \text{ct}) \rightarrow z \\ \text{pick } \beta \\ z' \leftarrow z \oplus \beta \\ \text{return } z' \end{array} \right] \rightarrow 1 = \frac{1}{2}$$

## Equivalence with IND-CPA Game — ii (ter)

$\text{IND\$-CPA}_{b=0}(\mathcal{A}')$

$$\Pr \left[ \begin{bmatrix} \text{Gen} \rightarrow (\text{pk}, \text{sk}) \\ \mathcal{A}_1(\text{pk}) \rightarrow (\text{pt}_0, \text{pt}_1, \text{st}) \\ \text{pick } \beta \\ \text{pt}'_0 \leftarrow \text{pt}_\beta \\ \text{pick pt}'_1 \\ \text{Enc}(\text{pk}, \text{pt}'_0) \rightarrow \text{ct} \\ \mathcal{A}_2(\text{st}, \text{ct}) \rightarrow z \\ z' \leftarrow z \oplus \beta \\ \text{return } z' \end{bmatrix} \rightarrow 1 \right] = \begin{cases} \frac{1}{2} \Pr \left[ \begin{bmatrix} \text{Gen} \rightarrow (\text{pk}, \text{sk}) \\ \mathcal{A}_1(\text{pk}) \rightarrow (\text{pt}_0, \text{pt}_1, \text{st}) \\ (\beta = 0) \\ \text{Enc}(\text{pk}, \text{pt}_0) \rightarrow \text{ct} \\ \mathcal{A}_2(\text{st}, \text{ct}) \rightarrow z \\ \text{return } z \end{bmatrix} \rightarrow 1 \right] \\ + \\ \frac{1}{2} \Pr \left[ \begin{bmatrix} \text{Gen} \rightarrow (\text{pk}, \text{sk}) \\ \mathcal{A}_1(\text{pk}) \rightarrow (\text{pt}_0, \text{pt}_1, \text{st}) \\ (\beta = 1) \\ \text{Enc}(\text{pk}, \text{pt}_1) \rightarrow \text{ct} \\ \mathcal{A}_2(\text{st}, \text{ct}) \rightarrow z \\ \text{return } z \oplus 1 \end{bmatrix} \rightarrow 1 \right] \end{cases}$$

$$= \frac{1}{2} \Pr[\text{IND-CPA}_0(\mathcal{A}) \rightarrow 1] + \frac{1}{2} \Pr[\text{IND-CPA}_1(\mathcal{A}) \rightarrow 0]$$

$$= \frac{1}{2} \Pr[\text{IND-CPA}_0(\mathcal{A}) \rightarrow 1] + \frac{1}{2} - \frac{1}{2} \Pr[\text{IND-CPA}_1(\mathcal{A}) \rightarrow 1]$$

$$= \frac{1}{2} - \frac{1}{2} \text{Adv}_{\mathcal{A}}(\text{IND-CPA})$$

## 1 The Cryptographic Zoo

## 2 Cryptographic Security Models

- Security Definitions
- The Game Proof Methodology
- RSA Security
- Rabin Cryptosystem
- Diffie-Hellman Security
- ElGamal Security

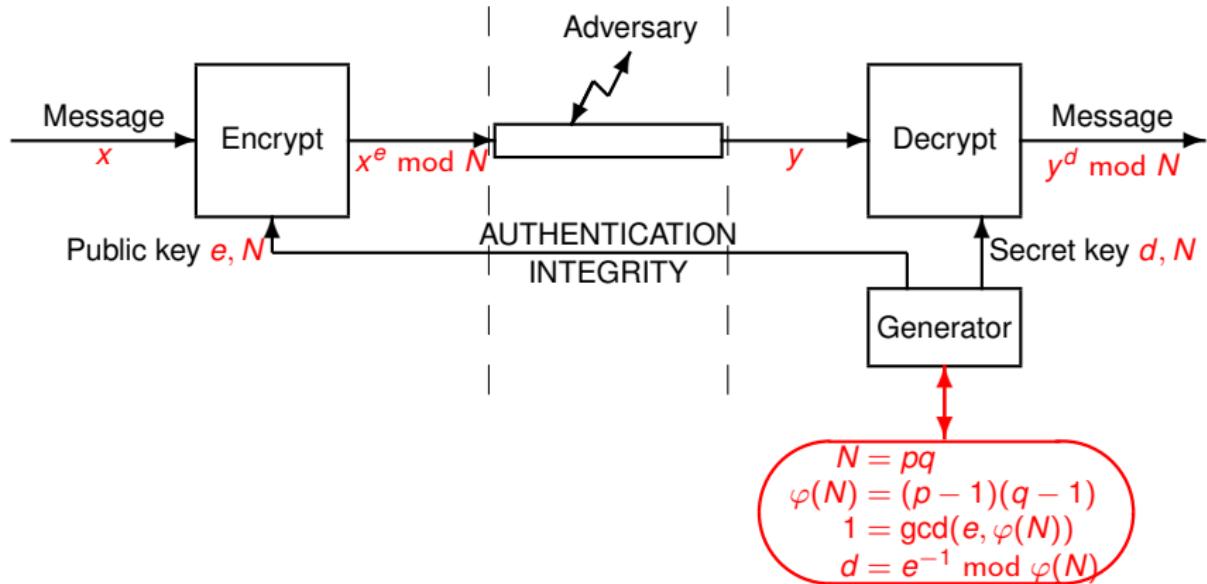
## 3 Cryptanalysis (Public-Key)

## 4 The Power of Interaction

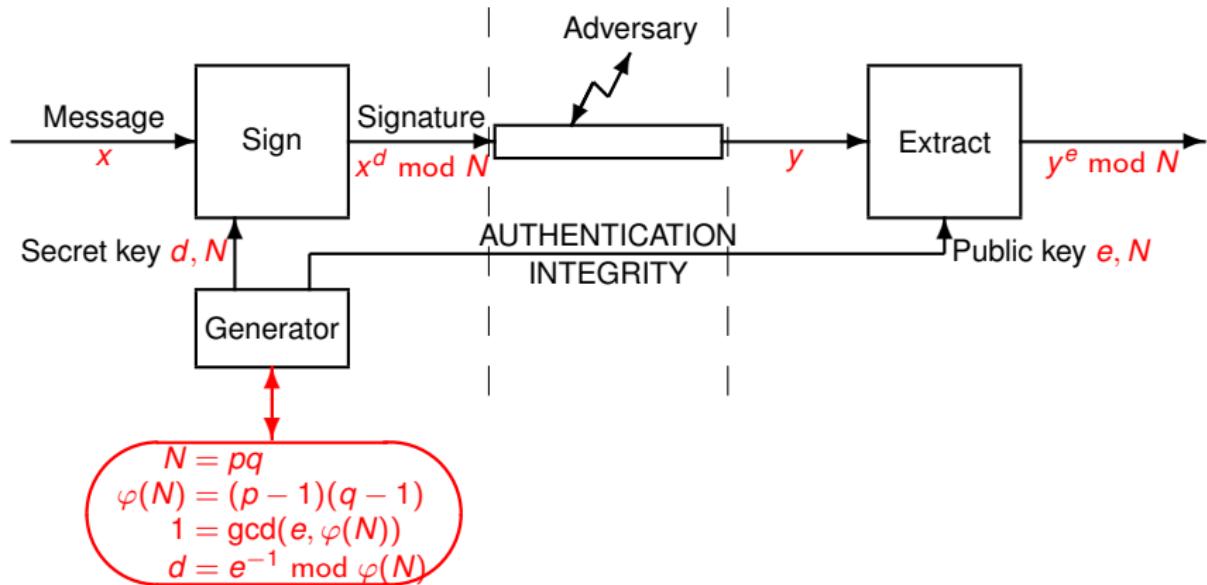
## 5 Cryptanalysis (Conventional)

## 6 Proving Security

# Plain RSA Encryption



# Plain RSA Signature



# RSA Problems

(implicit: Gen is the RSA key generation algorithm)

## RSA Problem

Given  $(N, e)$  generated by Gen and a random residue  $y$ , compute  $x$  such that

$$y = x^e \bmod N:$$

Game

$$1: \text{Gen}(1^s) \xrightarrow{\$} (N, e)$$

$$2: \text{pick } x \in \mathbf{Z}_N$$

$$3: y = x^e \bmod N$$

$$4: \mathcal{A}(N, y) \xrightarrow{\$} z$$

$$5: \mathbf{return} \ 1_{x=z}$$

RSA assumption: the RSA problem is hard

## RSA Factoring

Factor  $N$  generated by Gen:

Game

$$1: \text{Gen}(1^s) \xrightarrow{\$} N$$

$$2: \mathcal{A}(N) \xrightarrow{\$} (p, q)$$

$$3: \mathbf{return} \ 1_{1 < p, q < N, N = pq}$$

# Bit Security of Plain RSA

$\text{lsb}(x)$ : least significant bit of  $x$  ( $1$  iff  $x \bmod N$  is odd)

$\text{lsbdec}(y)$ : least significant bit of the decryption of  $y$

## Theorem

*If the RSA decryption problem is hard,  $\text{lsb}(y)$  is a **hard core bit**: recovering it is as hard as decrypting  $y$ .*

(even approximations...)

## Reducing Decryption to lsbdec

Key trick:  $\text{lsb}(2^{i+1}x \bmod N) = i\text{th}$  bit of the binary expansion of  $x/N\dots$

- we write  $a = \frac{k}{2^i}N \leq x < \frac{k+1}{2^i}N = b$
- start with  $k = 0$  and  $i = 0$
- increment  $i$  and find the bit  $\beta$  s.t.  $k$  replaced by  $2k + \beta$
- end up with  $a \leq x < b$  with  $b - a \leq 1$

```
1:  $a \leftarrow 0, b \leftarrow N$ 
2: for  $i = 0$  to  $\lfloor \log_2 N \rfloor$  do
3:   if  $\text{lsbdec}(2^{(i+1)e}y \bmod N) = 1$  then
4:      $a \leftarrow (a + b)/2$ 
5:   else
6:      $b \leftarrow (a + b)/2$ 
7:   end if
8: end for
9: yield  $\lfloor a \rfloor$ 
```

# Not All Bits are Hard in Plain RSA

`jacdec(y)`: Jacobi symbol of the decryption of  $y$

`jac(x)`: Jacobi symbol  $(x/N)$

$$\text{jacdec}(y) = \left( \frac{y^d \bmod N}{N} \right) = \left( \frac{y^d}{N} \right) = \left[ \left( \frac{y}{N} \right) \right]^d = \left( \frac{y}{N} \right)$$

which is easy to compute even without the factorization of  $N$

# RSA Security

(seen in the previous lecture)

- key recovery is equivalent to **factoring**  $N$   
given  $N$  generated by Gen, factor  $N$
- the decryption problem is the **RSA problem**  
given  $(N, e)$  generated by Gen ( $e$  is coprime with  $\varphi(N)$ )  
and  $y \in \mathbf{Z}_N$  random, compute  $x$  such that  $x^e \bmod N = y$   
(not known to be equivalent to factoring)
- not IND-CPA secure (deterministic)
- no CCA security:  $\text{Dec}(y \cdot u^e \bmod N)/u = \text{Dec}(y)$
- **strong RSA problem:**  
given  $N$  generated by Gen and  $y \in \mathbf{Z}_N$  random, compute  $x$  and  $e > 1$  such that  $x^e \bmod N = y$

# Strong RSA Problem

(implicit: Gen is the RSA key generation algorithm)

## Strong RSA Problem

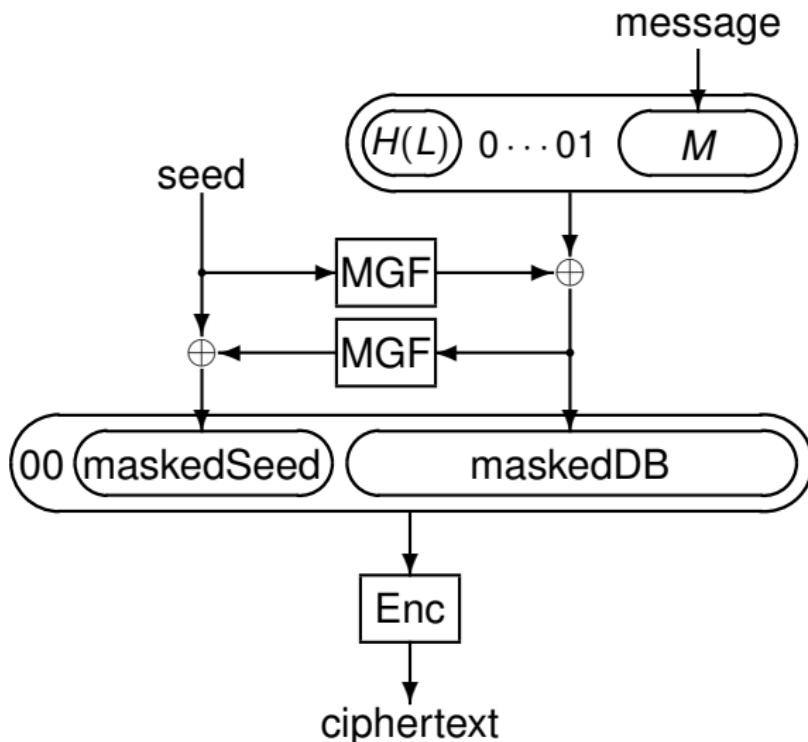
Given  $N$  generated by Gen and a random residue  $y$ , compute  $x$  and  $e > 1$  such that  $y = x^e \bmod N$ :

Game

- 1:  $\text{Gen}(1^s) \xrightarrow{\$} N$
- 2: pick  $y \in \mathbb{Z}_N$
- 3:  $\mathcal{A}(N, y) \xrightarrow{\$} (x, e)$
- 4: **return**  $1_{x^e \bmod N = y, e > 1}$

Strong RSA assumption: the strong RSA problem is hard

# RSA-OAEP



# Security of RSA-OAEP

- Security results are far beyond this lecture
- They exist for the strongest security notion IND-CCA (in the random oracle model, under the RSA assumption)
- Its significance was subject to a highly technical controversy  
(The original proof was wrong.)

# RSA Security: RSA Problems

(implicit:  $n$  is product of two different large primes and  $\gcd(e, \varphi(n)) = 1$ )

## RSADP (RSA Decryption Problem)

- 1:  $\text{RSA.Gen}(1^\lambda) \rightarrow (n, e, d)$
- 2: pick  $x \in \mathbf{Z}_n$
- 3:  $y \leftarrow x^e \bmod n$
- 4:  $\mathcal{A}(n, e, y) \rightarrow z$
- 5: **return**  $1_{x=z}$

## GOP (Group Order Problem)

- 1:  $\text{RSA.Gen}(1^\lambda) \rightarrow (n, e, d)$
- 2:  $\mathcal{A}(n) \rightarrow z$
- 3: **return**  $1_{\varphi(n)=z}$

## RSAKRP (RSA Key Recovery Problem)

- 1:  $\text{RSA.Gen}(1^\lambda) \rightarrow (n, e, d)$
- 2:  $\mathcal{A}(n, e) \rightarrow z$
- 3: **return**  $1_{z=d}$

## RSAFP (RSA Factorization Problem)

- 1:  $\text{RSA.Gen}(1^\lambda) \rightarrow (n, e, d)$
- 2:  $\mathcal{A}(n) \rightarrow (p, q)$
- 3: **return**  $1_{pq=n, 1 < p, q < n}$

## EMP (Exponent Multiple Problem)

- 1:  $\text{RSA.Gen}(1^\lambda) \rightarrow (n, e, d)$
- 2:  $\mathcal{A}(n) \rightarrow z$
- 3: **return**  $1_{\lambda(n) \text{ divides } z}$



## 1 The Cryptographic Zoo

## 2 Cryptographic Security Models

- Security Definitions
- The Game Proof Methodology
- RSA Security
- Rabin Cryptosystem
- Diffie-Hellman Security
- ElGamal Security

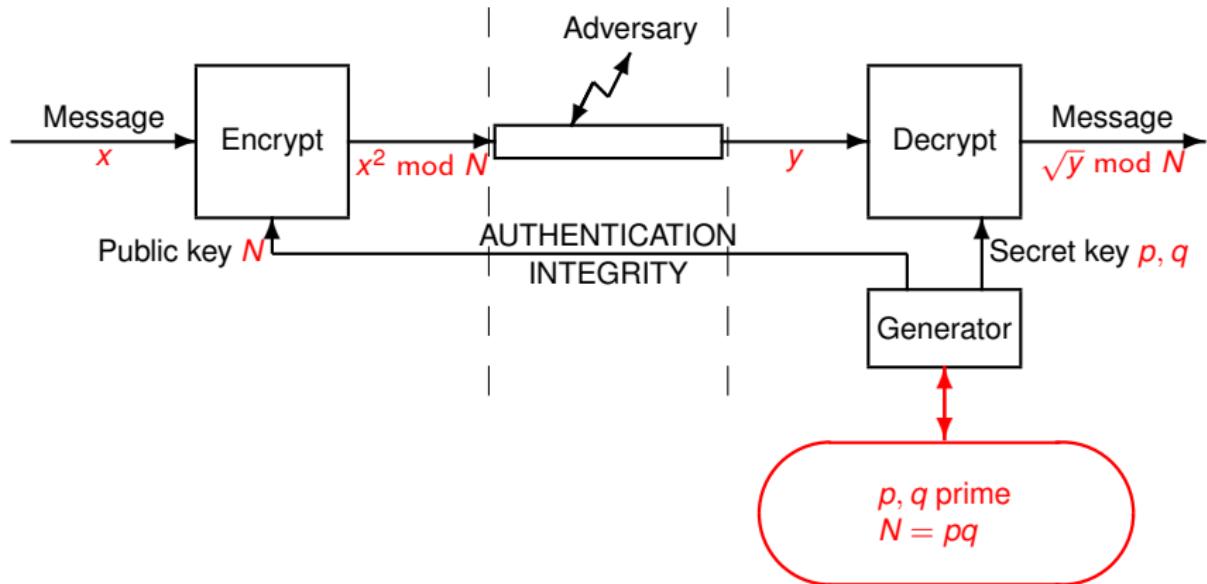
## 3 Cryptanalysis (Public-Key)

## 4 The Power of Interaction

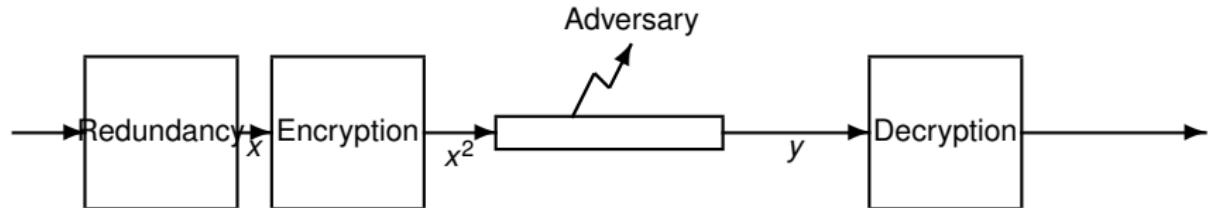
## 5 Cryptanalysis (Conventional)

## 6 Proving Security

# Plain Rabin Encryption



# Ensuring Non-Ambiguity in the Decryption



- we add redundancy in the plaintext so that valid plaintexts are sparse
- we make sure that no other square root has valid redundancy
- we take the only expected square root with valid redundancy
- we reject ciphertexts which fail to decrypt

# Rabin Security: Rabin Problems

(implicit:  $N$  is the product of two different large primes)

## RD (Rabin Decryption)

Given  $N$  generated by Gen and a random quadratic residue, compute a square root:

Game

- 1:  $\text{Gen}(1^s) \xrightarrow{\$} N$
- 2: pick  $x \in \mathbf{Z}_N$
- 3:  $y = x^2 \bmod N$
- 4:  $\mathcal{A}(N, y) \xrightarrow{\$} z$
- 5: **return**  $1_{z=x}$

## RKR (Rabin Key Recovery)

Factor  $N$  generated by Gen:  
Game

- 1:  $\text{Gen}(1^s) \xrightarrow{\$} N$
- 2:  $\mathcal{A}(N) \xrightarrow{\$} (p, q)$
- 3: **return**  $1_{1 < p < N, N = pq}$

$$\begin{array}{ccc} \text{decryption problem} & \iff & \text{factorization problem} \\ \text{key recovery problem} & \iff & \text{factorization problem} \end{array}$$

# Fact Hard $\implies$ RD Hard

Game RD

- 1:  $\text{Gen}(1^s) \xrightarrow{\$} N$
- 2: pick  $x \in \mathbf{Z}_N$
- 3:  $y \leftarrow x^2 \bmod N$
- 4:  $\mathcal{A}(N, y) \xrightarrow{\$} z$
- 5: **return**  $1_{z=x}$

$$\Pr[x \notin \mathbf{Z}_N^*] = \frac{p+q-1}{pq} = \text{negl}$$

for each  $y$ ,  $\Pr[\text{win}] = \frac{1}{2}$



Game Fact

- 1:  $\text{Gen}(1^s) \xrightarrow{\$} N$
- 2:  $\mathcal{B}(N) \xrightarrow{\$} (p, q)$
- 3: **return**  $1_{1 < p < N, N = pq}$

$\mathcal{B}(N)$ :

- 4: pick  $x \in \mathbf{Z}_N^*$
- 5:  $y \leftarrow x^2 \bmod N$
- 6:  $\mathcal{A}(N, y) \rightarrow z$
- 7:  $p \leftarrow \gcd(z - x, N)$
- 8: **if**  $p \in \{1, N\}$  **then** abort
- 9:  $q \leftarrow N/p$
- 10: **return**  $(p, q)$

$$\Pr[\text{Fact works}] = \frac{1}{2} \Pr[\text{RDP works}] - \text{negl}$$

# Paradoxical Security Result

- Rabin cryptosystem is PROVABLY as secure as the factorization is hard
- the security proof yields a CHOSEN CIPHERTEXT ATTACK
- the attack does not hold when adding the redundancy
- the security proof does no longer hold with the redundancy

## 1 The Cryptographic Zoo

## 2 Cryptographic Security Models

- Security Definitions
- The Game Proof Methodology
- RSA Security
- Rabin Cryptosystem
- Diffie-Hellman Security
- ElGamal Security

## 3 Cryptanalysis (Public-Key)

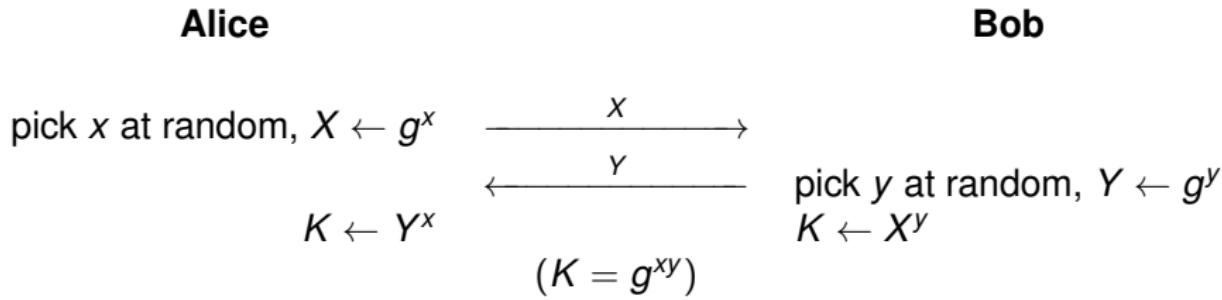
## 4 The Power of Interaction

## 5 Cryptanalysis (Conventional)

## 6 Proving Security

# The Diffie-Hellman Key Agreement Protocol

Assume a group ( $\mathbb{Z}_p^*$ , an elliptic curve, ...) generated by some  $g$



resists to passive adversaries

# CDH vs DL Problems

implicit:  $\text{Setup} \rightarrow (\text{group}, q, g)$

- group: parameters to be able to make group operations
- $q$ : integer (order of the group)
- $g$ : group element (generator)

## CDH (Computational Diffie-Hellman)

$$\text{Adv} = \Pr[\text{game returns 1}]$$

Game

- 1:  $\text{Setup}(1^s) \xrightarrow{\$} (\text{group}, q, g)$
- 2: pick  $x, y \in \mathbf{Z}_q$
- 3:  $X \leftarrow g^x, Y \leftarrow g^y$
- 4:  $\mathcal{A}(\text{group}, q, g, X, Y) \xrightarrow{\$} K$
- 5: **return**  $1_{K=g^{xy}}$

## DL (Discrete Logarithm)

$$\text{Adv} = \Pr[\text{game returns 1}]$$

Game

- 1:  $\text{Setup}(1^s) \xrightarrow{\$} (\text{group}, q, g)$
- 2: pick  $x \in \mathbf{Z}_q$
- 3:  $X \leftarrow g^x$
- 4:  $\mathcal{A}(\text{group}, q, g, X) \xrightarrow{\$} z$
- 5: **return**  $1_{X=g^z}$

CDH hard  $\implies$  DL hard

# CDH is the Key Recovery Problem in DH

Game CDH

- 1:  $\text{Setup}(1^s) \xrightarrow{\$} (\text{group}, q, g)$
- 2: pick  $x, y \in \mathbf{Z}_q$
- 3:  $X \leftarrow g^x, Y \leftarrow g^y$
- 4:  $\mathcal{A}(\text{group}, q, g, X, Y) \xrightarrow{\$} K$
- 5: **return**  $1_{K=g^{xy}}$

Game KR with DH

- 1:  $\text{Setup}(1^s) \xrightarrow{\$} (\text{pp})$
- 2: execute  $A(\text{pp}) \leftrightarrow B(\text{pp})$
- 3: get transcript and output  $K$
- 4: run  $\mathcal{A}(\text{pp}, \text{transcript}) \xrightarrow{\$} K'$
- 5: **return**  $1_{K=K'}$

# CDH Hard $\implies$ DL Hard

Game DL

- 1:  $\text{Setup}(1^s) \xrightarrow{\$} (\text{group}, q, g)$
- 2: pick  $x \in \mathbf{Z}_q$
- 3:  $X \leftarrow g^x$
- 4:  $\mathcal{A}(\text{group}, q, g, X) \xrightarrow{\$} z$
- 5: **return**  $1_{X=g^z}$

Game CDH

- 1:  $\text{Setup}(1^s) \xrightarrow{\$} (\text{group}, q, g)$
- 2: pick  $x, y \in \mathbf{Z}_q$
- 3:  $X \leftarrow g^x, Y \leftarrow g^y$
- 4:  $\mathcal{B}(\text{group}, q, g, X, Y) \xrightarrow{\$} K$
- 5: **return**  $1_{K=g^{xy}}$

$\mathcal{B}(\text{group}, q, g, X, Y)$ :

- 6:  $\mathcal{A}(\text{group}, q, g, X) \xrightarrow{\$} z$
- 7:  $K \leftarrow Y^z$
- 8: **return**  $K$

If  $X = g^z$  then  $K = g^{xy}$  so  $\Pr[\text{CDH}(\mathcal{B}) \rightarrow 1] \geq \Pr[\text{DL}(\mathcal{A}) \rightarrow 1]$

# Decisional DH Problem

## DDH (Decisional Diffie-Hellman)

$$\text{Adv} = \Pr[\Gamma_1 \rightarrow 1] - \Pr[\Gamma_0 \rightarrow 1]$$

Game  $\Gamma_b$

- 1:  $\text{Setup}(1^s) \xrightarrow{\$} (\text{group}, q, g)$
- 2: pick  $x, y \in \mathbb{Z}_q$
- 3: **if**  $b = 0$  **then**
- 4:     pick  $z \in \mathbb{Z}_q$
- 5: **else**
- 6:      $z \leftarrow xy \bmod q$
- 7: **end if**
- 8:  $X \leftarrow g^x, Y \leftarrow g^y, Z \leftarrow g^z$
- 9:  $\mathcal{A}(\text{group}, g, X, Y, Z) \xrightarrow{\$} c$
- 10: **return**  $c$

DDH is the key distinguisher problem with DH

# DDH is the Key Distinguisher Problem in DH

key distinguisher problem  
that can be apply to any key exchange  
protocol

## Game DDH<sub>b</sub>

- 1:  $\text{Setup}(1^s) \xrightarrow{\$} (\text{group}, q, g)$
- 2: pick  $x, y \in \mathbf{Z}_q$
- 3:  $z \leftarrow b = 0 ? \text{random} : xy$
- 4:  $X \leftarrow g^x, Y \leftarrow g^y, Z \leftarrow g^z$
- 5:  $\mathcal{A}(\text{group}, q, g, X, Y, Z) \xrightarrow{\$} c$
- 6: **return**  $c$

## Game KD with DH

- 1:  $\text{Setup}(1^s) \xrightarrow{\$} (\text{pp})$
- 2: execute  $A(\text{pp}) \leftrightarrow B(\text{pp})$
- 3: get transcript and  $K_1$
- 4: set  $K_0$  at random
- 5:  $\mathcal{A}(\text{pp}, \text{transcript}, K_b) \xrightarrow{\$} z$
- 6: **return**  $z$

# DDH Hard $\implies$ CDH Hard

Game CDH

- 1:  $\text{Setup}(1^s) \xrightarrow{\$} (\text{group}, q, g)$
- 2: pick  $x, y \in \mathbb{Z}_q$
- 3:  $X \leftarrow g^x, Y \leftarrow g^y$
- 4:  $\mathcal{A}(\text{group}, q, g, X, Y) \xrightarrow{\$} K$
- 5: **return**  $1_{K=g^{xy}}$

$$\begin{aligned}\text{Adv}(\mathcal{D}) &= \Pr[\text{DDH}_1 \rightarrow 1] - \Pr[\text{DDH}_0 \rightarrow 1] \\ &= \Pr[\text{CDH}(\mathcal{A}) \rightarrow 1] - \Pr[\text{DDH}_0 \rightarrow 1] \\ &= \text{Adv}(\mathcal{A}) - \frac{1}{q}\end{aligned}$$

Game DDH

- 1:  $\text{Setup}(1^s) \xrightarrow{\$} (\text{group}, q, g)$
- 2: pick  $x, y \in \mathbb{Z}_q$
- 3:  $z \leftarrow b = 0 ? \text{random} : xy$
- 4:  $X \leftarrow g^x, Y \leftarrow g^y, Z \leftarrow g^z$
- 5:  $\mathcal{B}(\text{group}, q, g, X, Y, Z) \xrightarrow{\$} c$
- 6: **return**  $c$

$\mathcal{B}(\text{group}, q, g, X, Y, Z)$ :

- 7:  $\mathcal{A}(\text{group}, q, g, X, Y) \xrightarrow{\$} K$
- 8: **return**  $1_{K=Z}$

$$\text{Adv}(\mathcal{A}) = \text{Adv}(\mathcal{B}) + \frac{1}{q}$$

next slide:  $\frac{1}{q} = \text{negl}$

# DDH Hard $\implies$ Large Group

```
 $\mathcal{B}'(\text{group}, g, X, Y, Z)$ :  
1: pick  $x$  at random  
2: if  $X = g^x$  then  
3:   if  $Z = Y^x$  then  
4:     return 1  
5:   else  
6:     return 0  
7: end if  
8: else  
9:   return 0  
10: end if
```

- $\Pr[\Gamma_1 \rightarrow 1] = \frac{1}{q}$
- $\Pr[\Gamma_0 \rightarrow 1] = \frac{1}{q^2}$
- $\text{Adv}(\mathcal{B}') = \frac{1}{q} \left(1 - \frac{1}{q}\right) \geq \frac{1}{2q}$
- if DDH is hard,  
 $\text{Adv}(\mathcal{B}') = \text{negl}$ , so  
 $\frac{1}{q} = \text{negl}$

## 1 The Cryptographic Zoo

## 2 Cryptographic Security Models

- Security Definitions
- The Game Proof Methodology
- RSA Security
- Rabin Cryptosystem
- Diffie-Hellman Security
- ElGamal Security

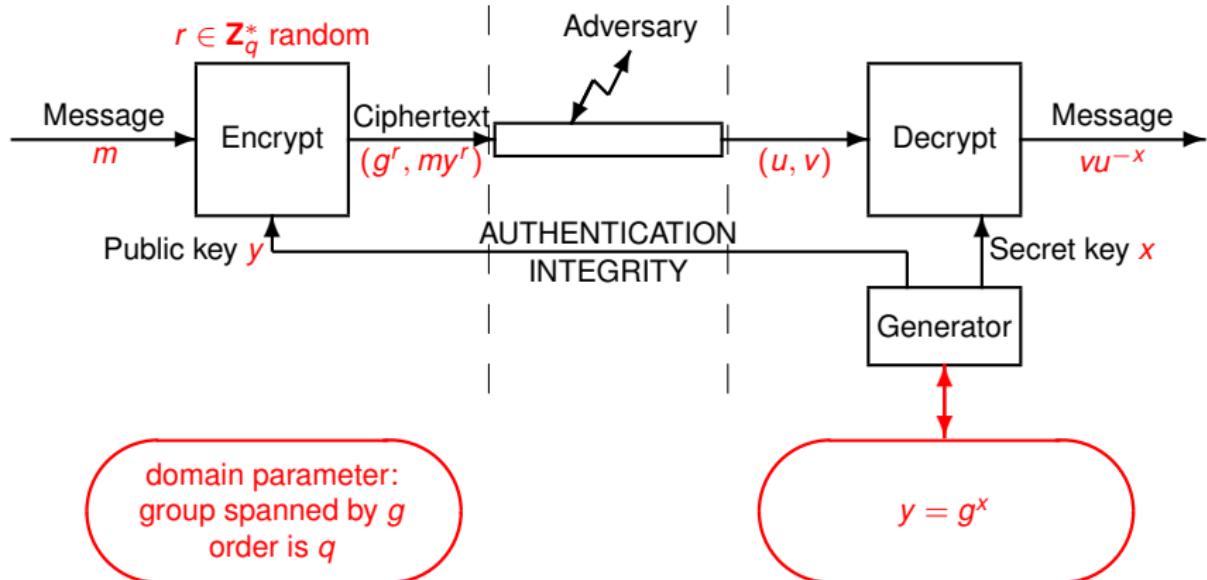
## 3 Cryptanalysis (Public-Key)

## 4 The Power of Interaction

## 5 Cryptanalysis (Conventional)

## 6 Proving Security

# Plain ElGamal Encryption



(assume  $m \in \langle g \rangle$ )

# EIGamal Security: EIGamal Problems

## EGD (EIGamal Decryption)

Given  $(g, y)$  generated by Gen and  $u, v \in \langle g \rangle$ , compute  $m$  such that there exists  $r$  such that  $u = g^r$  and  $v = my^r$ :

Game

- 1:  $\text{Gen}(1^s) \xrightarrow{\$} (\text{group}, g, y)$
- 2: pick  $m \in \langle g \rangle$
- 3:  $\text{Enc}(y, m) \xrightarrow{\$} (u, v)$
- 4:  $\mathcal{A}(\text{group}, g, y, u, v) \xrightarrow{\$} z$
- 5: **return**  $1_{z=m}$

## EGKR (EIGamal Key Recovery)

Given  $(g, y)$  generated by Gen, compute  $x$  such that  $y = g^x$ :

Game

- 1:  $\text{Gen}(1^s) \xrightarrow{\$} (\text{group}, g, y)$
- 2:  $\mathcal{A}(\text{group}, g, y) \xrightarrow{\$} z$
- 3: **return**  $1_{y=g^z}$

$$\begin{array}{ccc} \text{decryption problem} & \iff & \text{computational Diffie-Hellman problem} \\ \text{key recovery problem} & \iff & \text{discrete logarithm problem} \end{array}$$

## EGKR Hard $\implies$ DL Hard

Game DL

- 1:  $\text{Setup}(1^s) \xrightarrow{\$} (\text{group}, q, g)$
- 2: pick  $x \in \mathbf{Z}_q$
- 3:  $X \leftarrow g^x$
- 4:  $\mathcal{A}(\text{group}, q, g, X) \xrightarrow{\$} z$
- 5: **return**  $1_{X=g^z}$

Game EGKR

- 1:  $\text{Setup}(1^s) \xrightarrow{\$} (\text{group}, q, g)$
- 2: pick  $x \in \mathbf{Z}_q$
- 3:  $y \leftarrow g^x$
- 4:  $\mathcal{B}(\text{group}, q, g, y) \xrightarrow{\$} z$
- 5: **return**  $1_{y=g^z}$

$\mathcal{B}(\text{group}, q, g, y)$ :

- 6:  $\mathcal{A}(\text{group}, q, g, y) \xrightarrow{\$} z$
- 7: **return**  $z$

$$\Pr[\text{EGKR}(\mathcal{B}) \rightarrow 1] = \Pr[\text{DL}(\mathcal{A}) \rightarrow 1]$$

## DL Hard $\implies$ EGKR Hard

Game EGKR

- 1:  $\text{Setup}(1^s) \xrightarrow{\$} (\text{group}, q, g)$
- 2: pick  $x \in \mathbf{Z}_q$
- 3:  $y \leftarrow g^x$
- 4:  $\mathcal{A}(\text{group}, q, g, y) \xrightarrow{\$} z$
- 5: **return**  $1_{y=g^z}$

Game DL

- 1:  $\text{Setup}(1^s) \xrightarrow{\$} (\text{group}, q, g)$
- 2: pick  $x \in \mathbf{Z}_q$
- 3:  $X \leftarrow g^x$
- 4:  $\mathcal{B}(\text{group}, q, g, X) \xrightarrow{\$} z$
- 5: **return**  $1_{X=g^z}$

$\mathcal{B}(\text{group}, q, g, X)$ :

- 6:  $\mathcal{A}(\text{group}, q, g, X) \xrightarrow{\$} z$
- 7: **return**  $z$

$$\Pr[\text{DL}(\mathcal{B}) \rightarrow 1] = \Pr[\text{EGKR}(\mathcal{A}) \rightarrow 1]$$

# EGD Hard $\implies$ CDH Hard

Game CDH

- 1:  $\text{Setup}(1^s) \xrightarrow{\$} (\text{group}, q, g)$
- 2: pick  $x, y \in \mathbf{Z}_q$
- 3:  $X \leftarrow g^x, Y \leftarrow g^y$
- 4:  $\mathcal{A}(\text{group}, q, g, X, Y) \xrightarrow{\$} K$
- 5: **return**  $1_{K=g^{xy}}$

Game EGD

- 1:  $\text{Setup}(1^s) \xrightarrow{\$} (\text{group}, q, g)$
- 2: pick  $x \in \mathbf{Z}_q, y \leftarrow g^x$
- 3: pick  $m \in \langle g \rangle$
- 4: pick  $r \in \mathbf{Z}_q, u \leftarrow g^r, v \leftarrow my^r$
- 5:  $\mathcal{B}(\text{group}, q, g, y, u, v) \xrightarrow{\$} z$
- 6: **return**  $1_{m=z}$

$\mathcal{B}(\text{group}, q, g, y, u, v)$ :

- 7:  $\mathcal{A}(\text{group}, q, g, y, u) \xrightarrow{\$} K$
- 8: **return**  $v/K$

$$\Pr[\text{EGKR}(\mathcal{B}) \rightarrow 1] = \Pr[\text{DL}(\mathcal{A}) \rightarrow 1]$$

# CDH Hard $\implies$ EGD Hard

## Game EGD

- 1:  $\text{Setup}(1^s) \xrightarrow{\$} (\text{group}, q, g)$
- 2: pick  $x \in \mathbf{Z}_q$ ,  $y \leftarrow g^x$
- 3: pick  $m \in \langle g \rangle$
- 4: pick  $r \in \mathbf{Z}_q$ ,  $u \leftarrow g^r$ ,  
 $v \leftarrow my^r$
- 5:  $\mathcal{A}(\text{group}, q, g, y, u, v) \xrightarrow{\$} z$
- 6: **return**  $1_{m=z}$

## Game CDH

- 1:  $\text{Setup}(1^s) \xrightarrow{\$} (\text{group}, q, g)$
  - 2: pick  $x, y \in \mathbf{Z}_q$
  - 3:  $X \leftarrow g^x$ ,  $Y \leftarrow g^y$
  - 4:  $\mathcal{B}(\text{group}, q, g, X, Y) \xrightarrow{\$} K$
  - 5: **return**  $1_{K=g^{xy}}$
- $\mathcal{B}(\text{group}, q, g, X, Y)$ :
- 6: pick  $v \in \langle g \rangle$
  - 7:  $\mathcal{A}(\text{group}, q, g, X, Y, v) \xrightarrow{\$} z$
  - 8: **return**  $v/z$

$$\Pr[\text{EGKR}(\mathcal{B}) \rightarrow 1] = \Pr[\text{DL}(\mathcal{A}) \rightarrow 1]$$

# EIGamal Encryption Security

- **key recovery** is equivalent to the discrete logarithm problem
- **decryption** is equivalent to the CDH problem
- **decryption** with parameter  $g$  and public key  $y$  is equivalent to the Diffie-Hellman problem with parameter  $g$  and static key  $y$
- **IND-CPA security** equivalent to the hardness of the DDH problem  
(to be seen next)

# Plain ElGamal Encryption is not OW-CCA Secure

- if  $(u, v)$  is the encryption of  $x$  then  $(u, vw)$  is the encryption of  $xw$
- OW-CCA attack:
  - 1 get  $y$
  - 2 get target  $(u, v)$
  - 3 pick  $w$  at random
  - 4 ask for decryption of  $(u, vw)$
  - 5 get  $x'$
  - 6 return  $x = x'/w$
  - 7  $x$  is the decryption of  $(u, v)$ !
- consequence: not IND-CCA either

# Semantic Security of ElGamal Encryption

## Theorem

*If the DDH problem is hard over the group generated by Gen, then the ElGamal cryptosystem is IND-CPA secure.*

# Semantic Security of ElGamal Encryption

(IND\$-CPA security)

- given  $g$ , get  $y$ , choose  $x_0$ , get  $(u, v)$ , decide if  $(u, v)$  was generated by

$$r \in_U \mathbf{Z}_q, u = g^r, v = x_0 y^r \quad \text{or} \quad r \in_U \mathbf{Z}_q, x_1 \in_U \langle g \rangle, u = g^r, v = x_1 y^r$$

- (set  $v' = v/x_0$ )

given  $g$ , get  $y$ , get  $(u, v')$ , decide if  $(u, v')$  was generated by

$$r \in_U \mathbf{Z}_q, u = g^r, v' = y^r \quad \text{or} \quad r \in_U \mathbf{Z}_q, x_1 \in_U \langle g \rangle, u = g^r, v' = \frac{x_1}{x_0} y^r$$

- given  $g$ , get  $y$ , get  $(u, v')$ , decide if  $(u, v')$  was generated by

$$r \in_U \mathbf{Z}_q, u = g^r, v' = y^r \quad \text{or} \quad r \in_U \mathbf{Z}_q, u = g^r, v' \in_U \langle g \rangle$$

this is equivalent to the decisional Diffie-Hellman problem in  $\langle g \rangle$

# IND-CPA Proof with Game Methodology

take a group with a generator  $g$  of order  $q$   
(to ease notations, we write “ $g$ ” instead of “group,  $g$ ”)

**key generation:** pick  $x \in_U \mathbf{Z}_q$ , set  $y = g^x$

**message space:**  $\text{pt} \in \langle g \rangle$

**encryption:**  $\text{Enc}_y(\text{pt}; r) = (g^r, \text{pty}^r)$

**decryption:**  $\text{Dec}_x(u, v) = vu^{-x}$

IND-CPA game  $\Gamma_0^b$ :

- 1: run key generation and get  $y$
- 2: pick random coins  $\rho$  and set  $\text{view} = (g, y; \rho)$
- 3: run  $\mathcal{A}(\text{view}) = (\text{pt}_0, \text{pt}_1)$
- 4: pick  $r \in_U \mathbf{Z}_q$ , and  $(u, v) = \text{Enc}_y(\text{pt}_b; r)$
- 5: set  $\text{view} = (g, y, u, v; \rho)$
- 6: run  $\mathcal{A}(\text{view}) = b'$
- 7: **return**  $b'$

# Transitions — i

game  $\Gamma_0^b$ :

- 1: run key generation and get  $y$
- 2: pick  $\rho$  and set view =  $(g, y; \rho)$
- 3: run  $\mathcal{A}(\text{view}) = (\text{pt}_0, \text{pt}_1)$
- 4:  $r \in_U \mathbb{Z}_q, (u, v) \leftarrow \text{Enc}_y(\text{pt}_b; r)$
- 5: set view =  $(g, y, u, v; \rho)$
- 6: run  $\mathcal{A}(\text{view}) = b'$
- 7: **return**  $b'$

↑ bridge

game  $\Gamma_1^b$ :

- 1: pick  $x, y \leftarrow g^x$
- 2: pick  $r, u \leftarrow g^r$
- 3:  $v_0 \leftarrow g^{xr}$  ▷ erase  $x, r$
- 4: pick  $\rho$  and set view =  $(g, y; \rho)$
- 5: run  $\mathcal{A}(\text{view}) = (\text{pt}_0, \text{pt}_1)$
- 6:  $v = \text{pt}_b v_0$
- 7: set view =  $(g, y, u, v; \rho)$
- 8: run  $\mathcal{A}(\text{view}) = b'$
- 9: **return**  $b'$

com  
be  
ser  
adversary  
▷

DDH assumption in the group

≠ two games are given  
the solution and the other a random number

game  $\Gamma_2^b$ :

- 1: pick  $x, y \leftarrow g^x$
- 2: pick  $r, u \leftarrow g^r$
- 3: pick  $v_0 \leftarrow g^{xr}$  ▷ erase  $x, r$
- 4: pick  $\rho$  and set view =  $(g, y; \rho)$
- 5: run  $\mathcal{A}(\text{view}) = (\text{pt}_0, \text{pt}_1)$
- 6:  $v = \text{pt}_b v_0$
- 7: set view =  $(g, y, u, v; \rho)$
- 8: run  $\mathcal{A}(\text{view}) = b'$
- 9: **return**  $b'$

selected at  
random

# Transitions — ii

**game**  $\Gamma_2^b$ :

- 1: pick  $x, y \leftarrow g^x$
- 2: pick  $r, u \leftarrow g^r$
- 3: pick  $v_0$
- 4: pick  $\rho$  and set view =  $(g, y; \rho)$
- 5: run  $\mathcal{A}(\text{view}) = (\text{pt}_0, \text{pt}_1)$
- 6:  $v = \text{pt}_b v_0$
- 7: set view =  $(g, y, u, v; \rho)$
- 8: run  $\mathcal{A}(\text{view}) = b'$
- 9: **return**  $b'$

$\uparrow$  bridge

**game**  $\Gamma_3^b$ :

- 1: pick  $x, y \leftarrow g^x$
- 2: pick  $r, u \leftarrow g^r$
- 3: pick  $\rho$  and set view =  $(g, y; \rho)$
- 4: run  $\mathcal{A}(\text{view}) = (\text{pt}_0, \text{pt}_1)$
- 5: **pick**  $v_0$
- 6:  $v = \text{pt}_b v_0$   $\triangleright$  erase  $v_0$
- 7: set view =  $(g, y, u, v; \rho)$
- 8: run  $\mathcal{A}(\text{view}) = b'$
- 9: **return**  $b'$

messages are in the group!



**game**  $\Gamma_4^b$ :

- 1: pick  $x, y \leftarrow g^x$
- 2: pick  $r, u \leftarrow g^r$
- 3: pick  $\rho$  and set view =  $(g, y; \rho)$
- 4: run  $\mathcal{A}(\text{view}) = (\text{pt}_0, \text{pt}_1)$
- 5: **pick**  $v$
- 6: set view =  $(g, y, u, v; \rho)$
- 7: run  $\mathcal{A}(\text{view}) = b'$
- 8: **return**  $b'$

## Transitions — iii

final step:  $\Gamma_4^0$  and  $\Gamma_4^1$  are identical!

$$\begin{array}{ccccccccc} \Gamma_0^0 & \xrightarrow{\text{bridge}} & \Gamma_1^0 & \xrightarrow{\text{DDH}} & \Gamma_2^0 & \xrightarrow{\text{bridge}} & \Gamma_3^0 & \xrightarrow{\text{domain}} & \Gamma_4^0 \\ & & & & & & & & || \\ \Gamma_0^1 & \xrightarrow{\text{bridge}} & \Gamma_1^1 & \xrightarrow{\text{DDH}} & \Gamma_2^1 & \xrightarrow{\text{bridge}} & \Gamma_3^1 & \xrightarrow{\text{domain}} & \Gamma_4^1 \end{array}$$

so,  $\Pr[\Gamma_0^0 = 0] - \Pr[\Gamma_0^1 = 0] \leq 2\text{Adv}_{\text{DDH}} = \text{negl}$

# Observation

- DDH must be hard for security
- messages must be group elements

# Practical Problems with ElGamal Encryption

- the DDH problem is not always hard  
example of bad groups:  $\langle g \rangle = \mathbf{Z}_n, \mathbf{Z}_p^*, \dots$  (see slide 197)
- we should take  $g$  of large **prime** order  $q$
- but then how to embed pt in  $\langle g \rangle$ ?
  - example: in the case of a subgroup of  $\mathbf{Z}_p^*$  with  $p \gg q$ , group elements are scarce → hard to embed pt in  $\langle g \rangle$
  - example: in the case of a subgroup of  $\mathbf{Z}_p^*$  with  $p = 2q + 1$ , one residue out of two is a group element
    - 1 is not a group element (since  $(-1)^q \neq 1$ )
    - we could take  $\text{map(pt)} = \pm \text{pt} \bmod p \in \langle g \rangle$  for  $1 \leq \text{pt} \leq q$
  - example: elliptic curves  
→ technical to embed pt in  $\langle g \rangle$

# Conclusion

- **key recovery**

RSA, Rabin: key recovery  $\iff$  factoring  $pq$

ElGamal: key recovery  $\iff$  discrete logarithm

- ciphertext **decryption**

RSA: decryption  $\iff$  RSA problem

Rabin: decryption  $\iff$  factoring  $pq$

ElGamal: decryption  $\iff$  CDH problem

- **hard core bit**

lsb in RSA  $\iff$  RSA problem

- **semantic security** (IND-CPA)

ElGamal IND-CPA  $\iff$  DDH problem

- **adaptive security** (IND-CCA)

none

# References

- **Shoup.**  
Sequences of Games: A Tool for Taming Complexity in  
Security Proofs  
Eprint 2004/332

# Train Yourself

- security of encryption: final exam 2011–12 ex3
- proofs:
  - final exam 2010–11 ex4 (security reduction)
  - final exam 2011–12 ex2 (MAC revisited)
  - final exam 2014–15 ex1 (security of Davies-Meyer)
- DDH:
  - final exam 2015–16 ex2 (fixed vs random  $g$  in DH problems)
  - midterm exam 2017–18 ex2 (gap DH problem)
  - midterm exam 2016–17 ex1 (solving DDH with small subgroups)
  - midterm exam 2014–15 ex1 (hard log and easy DH world)
  - final exam 2011–12 ex1 (easy DDH cases)
  - midterm exam 2010–11 ex2 (easy DDH from bilinear mappings)
- ElGamal:
  - final exam 2012–13 ex1
  - final exam 2009–10 ex2 (message mapping)
  - midterm exam 2016–17 ex3 (issue from weird distribution)
- BLS signature: final exam 2012–13 ex2
- PRF: final exam 2015–16 ex3 (equivalent PRF notions)
- PRP vs left-or-right security: final exam 2016–17 ex3
- IND-CPA implies collision-resistance: final exam 2017–18 ex1
- IND-CCA and NM-CCA are equivalent: final exam 2017–18 ex2
- difference between MAC and PRF: midterm exam 2016–17 ex2







## 1 The Cryptographic Zoo

## 2 Cryptographic Security Models

## 3 Cryptanalysis (Public-Key)

## 4 The Power of Interaction

## 5 Cryptanalysis (Conventional)

## 6 Proving Security

## 1 The Cryptographic Zoo

## 2 Cryptographic Security Models

## 3 Cryptanalysis (Public-Key)

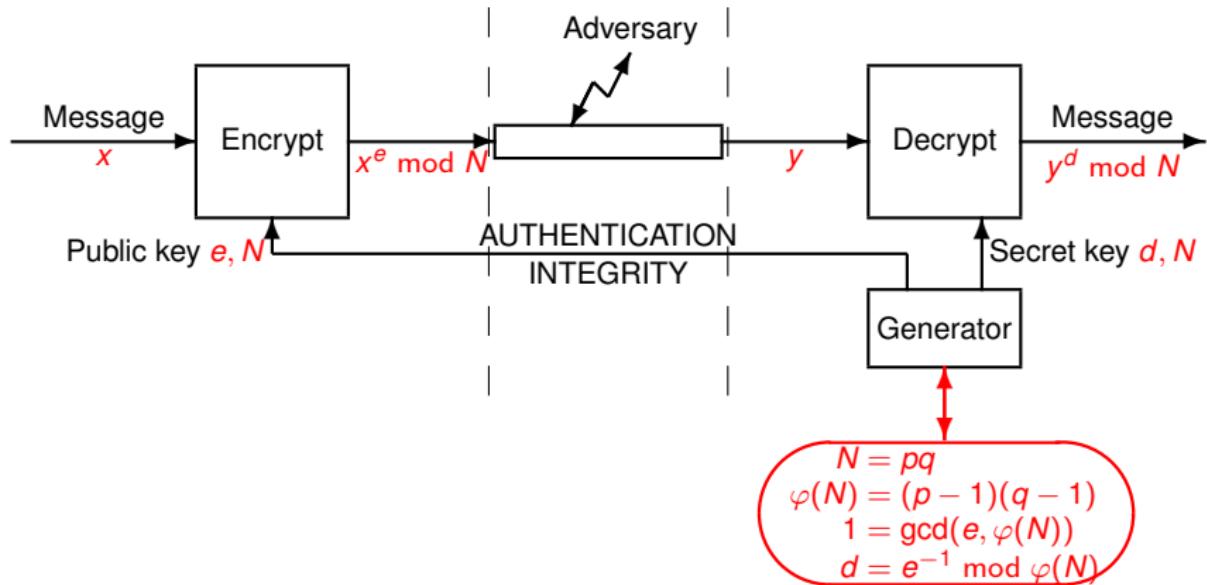
- RSA
- Diffie-Hellman
- ElGamal

## 4 The Power of Interaction

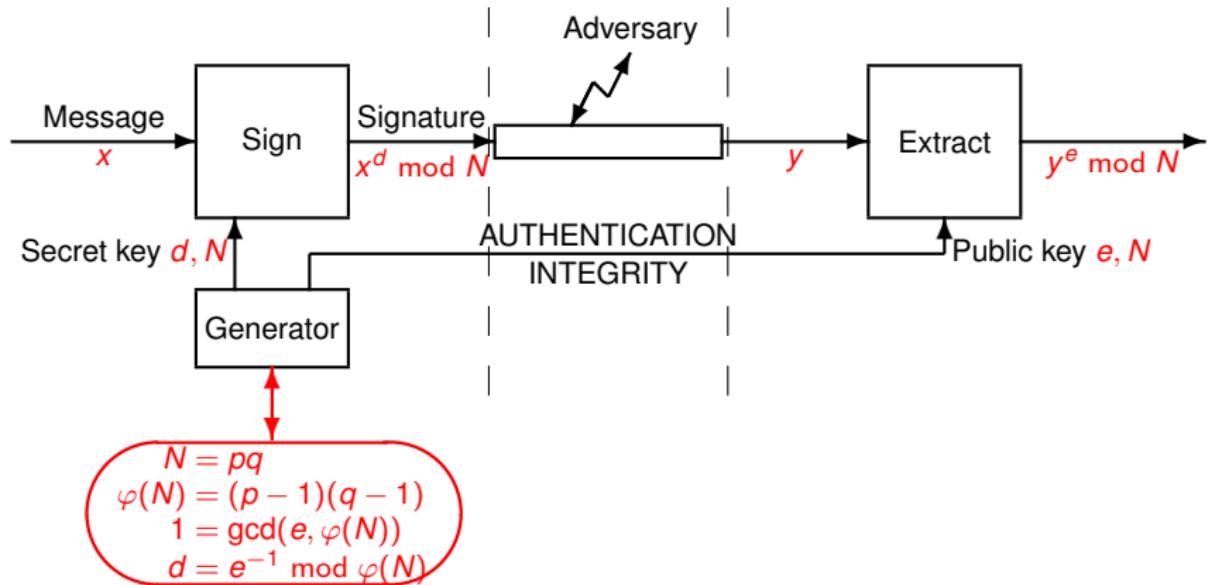
## 5 Cryptanalysis (Conventional)

## 6 Proving Security

# Plain RSA Encryption



# Plain RSA Signature



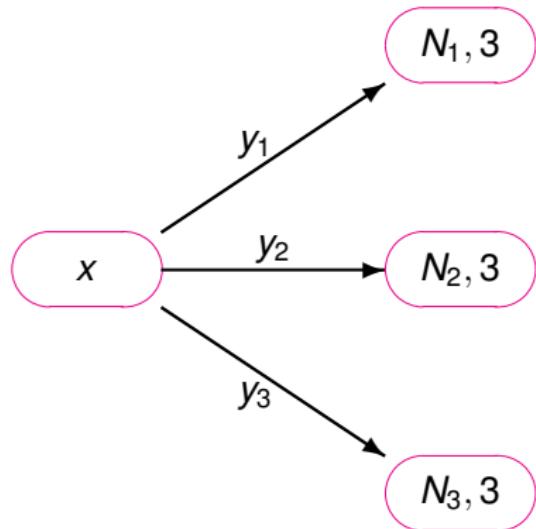
- Implementation issues (from plain RSA to real life standards)
  - Problems with broadcast encryption
  - Problems with low exponents
  - Problems with side channels
- Side channel attacks
  - Power analysis
  - Single/differential fault analysis
  - Timing attack, electromagnetic fields, sound
  - Side channel from protocols: formatting issues
- Relevance of the mathematical model

# Broadcast Encryption with Low Exponent

Sending the same message  $x$  to at least  $e$  participants with the same encryption exponent  $e$  and different moduli  $N_1, \dots, N_e$ .

- The  $i$ th participant receives  $y_i = x^e \bmod N_i$
- The attacker intercepts  $e$  values  $y_1, \dots, y_e$
- The attacker computes  $y = x^e \bmod N$  where  $N = N_1 \times \dots \times N_e$  by CRT (assume moduli are coprime)
- We have  $y = x^e$
- The attacker deduces  $x = \sqrt[e]{y}$

## Example with $e = 3$



Broadcast plaintext  $x$  to 3 receivers using  $e = 3$ :

Let  $y_i = x^3 \bmod N_i$   
We have  $\text{CRT}(y_1, y_2, y_3) = x^3 \bmod (N_1 N_2 N_3) = x^3$

So we can compute  $x^3$  then extract a cubic root and get  $x$

# Attack on Low Exponents

- Attack on low  $e$ : Coppersmith algorithm to find roots less than  $N^{\frac{1}{e}}$  of a polynomial of degree  $e$ .  
Example: decryption attack when  $e = 3$  and we know  $\frac{2}{3}$  of the plaintext bits (e.g. RSA.Enc(pattern|| $x$ ) with 1024-bit modulus when  $x$  is a 256-bit symmetric key and pattern is a constant pattern).
- Attack on low  $d$ : Wiener key recovery attack for  $d < \sqrt[4]{N}$  (e.g.  $N$  of 1024 bits and  $d$  of less than 256 bits).

# Simple Power Analysis (SPA)

Computing  $x = y^d \bmod N$  is performed by a device with external power supply by using the square-and-multiply algorithm.

- The power usage tells what kind of operation is performed
- Cryptoprocessors have faster square than multiply algorithms
- The power usage tells when a square and a multiply is performed
- The attacker deduces  $d$

# Square-and-Multiply Algorithm (Left to Right)

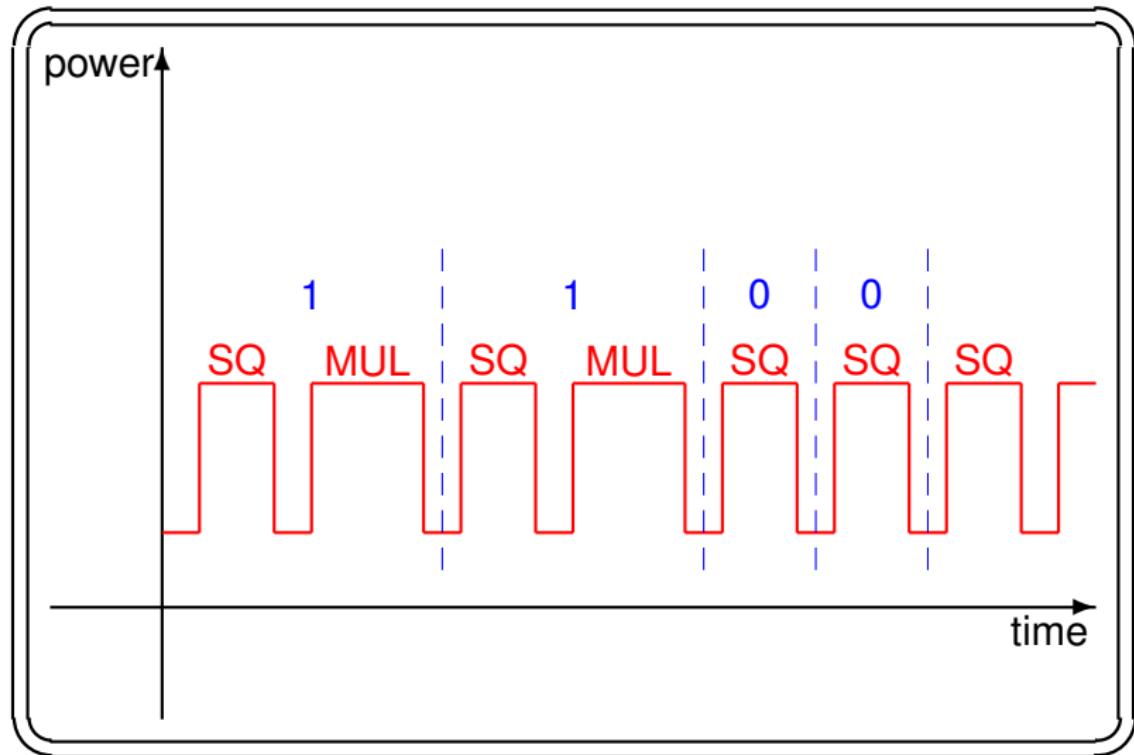
**Input:**  $y, d, N$ , three integers of at most  $\ell$  bits

**Output:**  $x = y^d \bmod N$

**Complexity:**  $\mathcal{O}(\ell^3)$

```
1:  $a \leftarrow 1$ 
2: for  $i = \ell - 1$  to 0 do
3:    $a \leftarrow a^2 \bmod N$ 
4:   if  $d_i = 1$  then
5:      $a \leftarrow a \times y \bmod N$ 
6:   end if
7: end for
8:  $x \leftarrow a$ 
```

# SPA



secret key is 1100... (from right to left or left to right)

# Countermeasures

- **hardware**  
flatten power usage
- **software**  
always do a multiplication

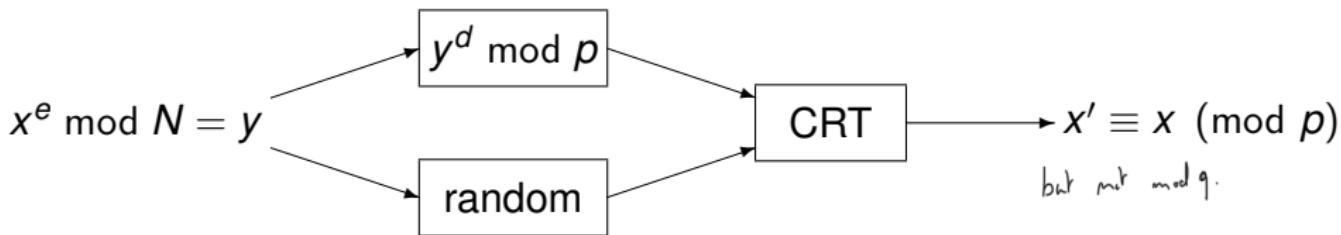
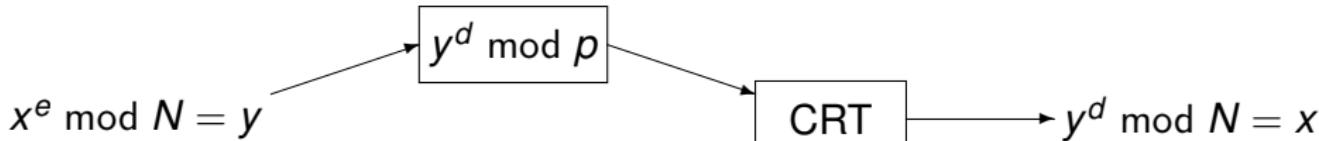
# Differential Fault Attack (DFA)

Computing  $x = y^d \bmod N$  is performed by a device using CRT acceleration.

- The attacker picks  $x$  and sends  $y = x^e \bmod N$  to the device
- The attacker aggressively (but mildly) stresses the device
- The device eventually makes computation errors
- Error may occur during the CRT acceleration
- The device computes  $x'$  and outputs it
- The attacker computes  $\gcd(x - x', N)$

# DFA

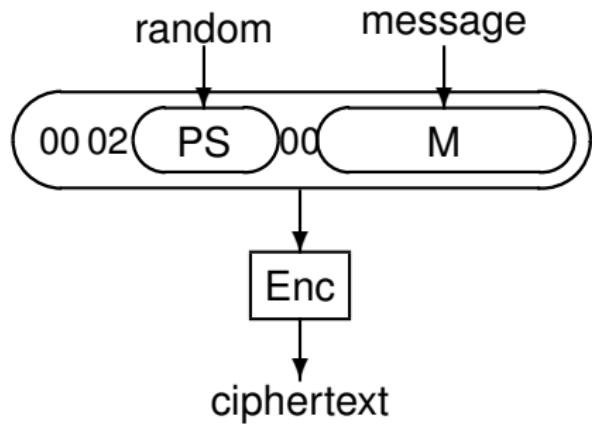
pick  $x$  and submit  $y = x^e \bmod N$  to normally obtain  $x$



# Countermeasures

- **hardware**  
sensors
- **software**  
verify the result

# PKCS#1v1.5 Encryption (Reminder) (OBSOLETE)



## Yet Another Side Channel Attack (PKCS#1v1.5)

Bleichenbacher's attack against PKCS#1v1.5 encryption:

- Attacker intercepts  $y = x^e \bmod N$  and aims at recovering  $x$
- Attacker plays with the server by sending fake ciphertexts  $y'$  of the form

$$y' = s^e y \bmod N$$

- Most of the time,  $y'$  does not decrypt well and the server issues an error message.
- If the server accepts, then  $(y')^d \bmod N$  starts with 00 02, hence

$$2 \times 256^{k-2} \leq sx \bmod N < 3 \times 256^{k-2}$$

- By using this oracle 14 500 times, Attacker can reconstruct  $x$  with probability 50% [Bardou et al., CRYPTO'12]

# Countermeasures

- hide (well) errors
- use the IND-CCA secure variant of RSA

# Other Side Channel Attacks

- Simple fault analysis
- Differential fault analysis
- Timing attack
- Electromagnetic fields
- Noisy machines
- Cache attacks
- Branch prediction algorithm
- ...

## 1 The Cryptographic Zoo

## 2 Cryptographic Security Models

## 3 Cryptanalysis (Public-Key)

- RSA
- Diffie-Hellman
- ElGamal

## 4 The Power of Interaction

## 5 Cryptanalysis (Conventional)

## 6 Proving Security

# CDH vs DL Problems (Reminder)

(implicit: relative to Setup)

## CDH (Computational Diffie-Hellman)

$$\text{Adv} = \Pr[\text{game returns 1}]$$

Game

- 1:  $\text{Setup}(1^s) \xrightarrow{\$} (\text{group}, q, g)$
- 2: pick  $x, y \in \mathbf{Z}_q$
- 3:  $X \leftarrow g^x, Y \leftarrow g^y$
- 4:  $\mathcal{A}(\text{group}, q, g, X, Y) \xrightarrow{\$} K$
- 5: **return**  $1_{K=g^{xy}}$

## DL (Discrete Logarithm)

$$\text{Adv} = \Pr[\text{game returns 1}]$$

Game

- 1:  $\text{Setup}(1^s) \xrightarrow{\$} (\text{group}, q, g)$
- 2: pick  $x \in \mathbf{Z}_q$
- 3:  $X \leftarrow g^x$
- 4:  $\mathcal{A}(\text{group}, q, g, X) \xrightarrow{\$} z$
- 5: **return**  $1_{X=g^z}$

# Easy Discrete Logarithm Cases

The DH and DL problems are relative to a group selection

- an **easy group** case:  $G = \mathbf{Z}_n$  for any  $n$   
If  $g$  generates  $G$  then the “exponential” of  $x$  is  
 $X = (xg) \bmod n$  so the “logarithm” of  $X$  is  $x = \frac{X}{g} \bmod n$
- **B-smooth order**:  $G$  of order  $n = p_1^{\alpha_1} \times \cdots \times p_r^{\alpha_r}$  with  $p_i$  prime and  $p_1 < \cdots < p_r < B$   
Use the Pohlig-Hellman algorithm

Hardness example: subgroup of  $\mathbf{Z}_p^*$  of large prime order  $q$

# Pohlig-Hellman Algorithm

- In a group of order  $n = p_1^{\alpha_1} \times \cdots \times p_r^{\alpha_r}$
- Pohlig-Hellman algorithm to compute discrete logarithm in  $\mathcal{O}((\alpha_1\sqrt{p_1} + \cdots + \alpha_r\sqrt{p_r})T)$  where  $T$  is the complexity of one group operation
- this is  $\mathcal{O}(\sqrt{B}T \log n)$  if  $n$  is  $B$ -smooth

# Pohlig-Hellman Algorithm

**Input:**  $g, X$  in a group  $G$  and the factorization  $\#G = n = p_1^{\alpha_1} \dots p_r^{\alpha_r}$   
such that  $p_i$  is prime,  $p_i \neq p_j$ , and  $\alpha_i > 0$ , for  $1 \leq i < j \leq r$

**Output:** the logarithm of  $X$  in base  $g$

**Complexity:**  $\mathcal{O}(\alpha_1 \sqrt{p_1} + \dots + \alpha_r \sqrt{p_r})$  group operations

- 1: **for**  $i = 1, \dots, r$  **do**
- 2:    $g' \leftarrow g^{n/p_i^{\alpha_i}}$
- 3:    $g'' \leftarrow g'^{p_i^{\alpha_i}-1}$
- 4:    $X' \leftarrow X^{n/p_i^{\alpha_i}}$
- 5:    $x_i \leftarrow 0$
- 6:   **for**  $j = 0$  to  $\alpha_i - 1$  **do**
- 7:      $X'' \leftarrow X'^{p_i^{\alpha_i-j-1}}$
- 8:     compute the discrete logarithm  $u$  of  $X''$  in the subgroup of  
order  $p_i$  which is spanned by  $g''$  (next slide)
- 9:      $X' \leftarrow X'/g'^{u \cdot p_i^j}$
- 10:     $x_i \leftarrow x_i + u \cdot p_i^j$
- 11:   **end for**
- 12: **end for**
- 13: reconstruct and yield  $x$  such that  $x \equiv x_i \pmod{p_i^{\alpha_i}}$
- $$\begin{array}{c} g'^{p_i^{\alpha_i}} = g^{\frac{n}{p_i^{\alpha_i}}} = 1 \\ g'^k = g^{\frac{n}{p_i^{\alpha_i}} \cdot k} \neq 1 \\ \hline \log_g X' = (\log_g X) \bmod p_i^{\alpha_i} \\ \log_g X' = x_0 + x_1 p_i + x_2 p_i^2 + \dots + x_{\alpha_i-1} p_i^{\alpha_i-1} \\ x_j \in \{0, \dots, p_i-1\} \end{array}$$

# Baby Step - Giant Step Algorithm

**Input:**  $g$  and  $X$  in a group  $G$ ,  $B$  an upper bound for  $\#G$

**Output:** the logarithm of  $X$  in base  $g$

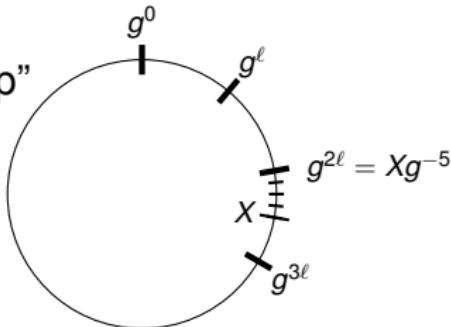
**Complexity:**  $\mathcal{O}(\sqrt{B})$  group operations

## Precomputation

- 1: let  $\ell = \lceil \sqrt{B} \rceil$  be the size of a “giant step”
- 2: **for**  $i = 0, \dots, \ell - 1$  **do**
- 3:     insert  $(g^{i\ell}, i)$  into a hash table
- 4: **end for**

## Algorithm

- 5: **for**  $j = 0, \dots, \ell - 1$  **do**
- 6:     compute  $z = Xg^{-j}$
- 7:     **if** we have a  $(z, i)$  in the hash table **then**
- 8:         yield  $x = i\ell + j$  and stop                       $\triangleright$  we get  $Xg^{-j} = g^{i\ell}$
- 9:     **end if**
- 10: **end for**



# Pollard Rho Discrete Logarithm Algorithm

$p$  prime size  $\sqrt{n}$  of field and circle

**Input:**  $g$  and  $y$  in a group  $G$  of order  $n$

**Output:** the logarithm of  $y$  in base  $g$

**Complexity:**  $\mathcal{O}(\sqrt{n})$  group operations

1: pick a random function

$$h : G \longrightarrow \{1, 2, 3\}$$

2:  $\vec{a}, \vec{b} \leftarrow (1, 0, 0) \in G \times \mathbf{Z}_n \times \mathbf{Z}_n$

3: **repeat**

4:      $\vec{a} \leftarrow f(\vec{a})$

5:      $\vec{b} \leftarrow f(f(\vec{b}))$

6: **until**  $a_1 = b_1$

7: **return**

$$(a_2 - b_2)/(a_3 - b_3) \bmod n \quad \triangleright \\ \text{fail if not possible}$$

- $G$  is multiplicatively denoted

- $f(x, \alpha, \beta)$  is defined to be

$$\begin{cases} (x \times g, \alpha + 1 \bmod n, \beta) & \text{if } h(x) = 1 \\ (x \times y, \alpha, \beta + 1 \bmod n) & \text{if } h(x) = 2 \\ (x^2, 2\alpha \bmod n, 2\beta \bmod n) & \text{if } h(x) = 3 \end{cases}$$

- vectors  $(x, \alpha, \beta)$  are all such that  $x = g^\alpha y^\beta$

- we could have taken another random function  $f$  with this property

# Decisional DH Problem (Reminder)

(implicit: relative to Gen) ➔ DDH

## Hardness of DDHP (Decisional Diffie-Hellman Problem)

$$\text{Adv} = \Pr[\Gamma_1 \rightarrow 1] - \Pr[\Gamma_0 \rightarrow 1]$$

Game  $\Gamma_b$

- 1:  $\text{Setup}(1^s) \xrightarrow{\$} (\text{group}, q, g)$
- 2: pick  $x, y \in \mathbb{Z}_q$
- 3: **if**  $b = 0$  **then**
- 4:     pick  $z \in \mathbb{Z}_q$
- 5: **else**
- 6:      $z \leftarrow xy \bmod q$
- 7: **end if**
- 8:  $X \leftarrow g^x, Y \leftarrow g^y, Z \leftarrow g^z$
- 9:  $\mathcal{A}(\text{group}, g, X, Y, Z) \xrightarrow{\$} c$
- 10: **return**  $c$

DDH is the key distinguisher problem with DH

## Easy Case: $\mathbb{Z}_p^*$ Safe for computational Diffie-Hellman but problem for DDH

The DDH problem is relative to some Setup may be easy

**$\mathbb{Z}_p^*$  with  $p$  prime:**  
Setup( $1^s$ )

- 1: pick a random prime  $p$  of size  $s$
- 2: pick a random generator  $g$  of  $\mathbb{Z}_p^*$
- 3: **return**  $(p, p - 1, g)$

then let  $\mathcal{A}(\text{pp}, X, Y, Z) = 1$  iff

$$\begin{array}{ccc} (\log_g Z \bmod 2) & = & (\log_g X \bmod 2) \times (\log_g Y \bmod 2) \\ \parallel & & \parallel \\ \log_{-1} \left( \frac{Z}{p} \right) & & \log_{-1} \left( \frac{X}{p} \right) \times \log_{-1} \left( \frac{Y}{p} \right) \end{array}$$

we have  $\left( \frac{\cdot}{p} \right) = -1$  iff the logarithm is odd, and

$\log Z = (\log X)(\log Y)$  in the DH case, so

$$\Pr_{b=0} [\mathcal{A}(\text{pp}, X, Y, Z) \rightarrow 1] = \frac{1}{2}, \quad \Pr_{b=1} [\mathcal{A}(\text{pp}, X, Y, Z) \rightarrow 1] = 1$$

thus  $\text{Adv}(\mathcal{A}) = \frac{1}{2}$

## Easy Case: Group of Order with Smooth Factor

The DDH problem is relative to some Setup may be easy

**$G$  of order  $n$  such that  $\frac{n}{w}$  is smooth:**

let  $\mathcal{A}(\text{pp}, X, Y, Z) = 1$  iff

$$\log_{g^w} Z^w = (\log_{g^w} X^w) \times (\log_{g^w} Y^w)$$

we have  $\text{Adv}(\mathcal{A}) = 1 - \frac{w}{n}$

Indeed,

$$\Pr_{b=0} [\mathcal{A}(\text{pp}, X, Y, Z) \rightarrow 1] = \frac{w}{n} \quad , \quad \Pr_{b=1} [\mathcal{A}(\text{pp}, X, Y, Z) \rightarrow 1] = 1$$

# Hard Cases

The DDH problem relative to some Setup is believed to be hard

- large subgroup of prime order of  $\mathbb{Z}_p^*$  ( $p$  prime)

**prime subgroup of  $\mathbb{Z}_p^*$  with  $p$  prime:**

Setup( $1^s$ )

- 1: pick a random prime  $q$  of size  $s$
- 2: pick a random  $p$  of size  $f(s)$  such that  $q|p-1$
- 3: start again until  $p$  is prime
- 4: pick a random  $g$  in  $\mathbb{Z}_p^*$  of order  $q$
- 5: **return**  $(p, q, g)$

- large subgroup of prime order of a “regular” elliptic curve

# Some Failure Cases

check the previous course

- attacks based on DL precomputation on a fixed group
- problems when not checking group membership
- problems with subgroups

## 1 The Cryptographic Zoo

## 2 Cryptographic Security Models

## 3 Cryptanalysis (Public-Key)

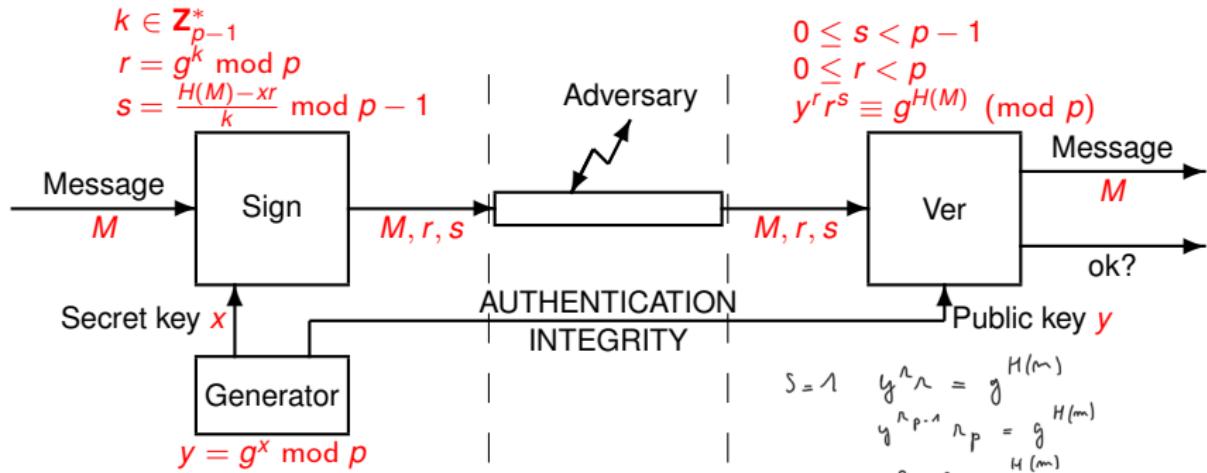
- RSA
- Diffie-Hellman
- ElGamal

## 4 The Power of Interaction

## 5 Cryptanalysis (Conventional)

## 6 Proving Security

# EIGamal Signature



$$e \in \mathbb{Z}_{p-1}^*$$

$$\downarrow \pmod p$$

$$g^{xn+s} \in \mathbb{Z}_p^* \quad g^{xn+hs} = g^{xn + h \left( \frac{H(m) - xn}{k} \right)} = g^{H(m)}$$

*p prime  
g generator of  $\mathbb{Z}_p^*$*

$$\begin{aligned} s &= 1 & y^{n_1} &= g^{H(m)} \\ y^{n_{p-1}} &= g^{H(m)} & n_p &= g^{H(m)} \\ n_{p-1} &= 0 & n_p &= g^{H(m)} \end{aligned}$$

To find such  $n$  we use the CRT  
such that  $\begin{cases} n \equiv 0 \pmod {p-1} \\ n \equiv g^{H(m)} \pmod p \end{cases}$   
so  $n \in \mathbb{Z}_{p(p-1)}$  because CRT  
 $n \in [0, p(p-1)]$

# EIGamal Signature Security

- key recovery is equivalent to the discrete logarithm problem
- results from EUROCRYPT 1994:
  - existential forgery is hard *on average* over the random choice of the public parameters in the *random oracle model* (covered in final chapter) provided that the discrete logarithm is hard
  - universal forgery attack on a specific parameter choice

## Security if we Miss the Second Inequality Check

(If we miss the first inequality check: no strong-forgery resistance...)

If we do not check that  $0 \leq r < p$ , we have a universal forgery attack:

- pick  $r_{p-1}, s \in \mathbf{Z}_{p-1}^*$  at random
- set  $r_p = g^{\frac{H(M)}{s}} y^{-\frac{r_{p-1}}{s}} \bmod p$
- pick  $r$  such that  $r \bmod p = r_p$  and  $r \bmod (p - 1) = r_{p-1}$  using the Chinese Remainder Theorem
- issue  $(r, s)$  as a signature for  $M$

## Bleichenbacher Attack (Setup Assumptions)

- Assume that  $p - 1 = bw$  with an integer  $b$  which is **smooth**. Example:  $b = 2$  which works for every odd prime  $p$ .
- Assume that we know some relation  $g^{1/t} \bmod p = cw$ . Example: if we have  $g = b$  (note that the complexity of the exponentiation is decreased if  $g$  is small) and  $p \equiv 1 \pmod{4}$ , the relation holds for  $t = \frac{p-3}{2}$  and  $c = 1$ :

$$(cw)^t \equiv \left(\frac{p-1}{g}\right)^{\frac{p-1}{2}-1} \equiv -g \frac{(-1)^{\frac{p-1}{2}}}{g^{\frac{p-1}{2}}} \equiv g \pmod{p}$$

since  $g^{\frac{p-1}{2}}$  is a square root of 1 which is not 1 (otherwise  $g$  would not be a generator) and  $(-1)^{\frac{p-1}{2}} = 1$  due to the assumption on  $p \bmod 4$ .

## Attack (Universal Forgery)

- We first take  $r = cw$ .
- We find  $z$  such that  $y^{cw} \equiv g^{cwz} \pmod{p}$ . This is nothing but the discrete logarithm of  $y^{wc} \pmod{p}$  in base  $g^{wc} \pmod{p}$  which spans a group of order factor of  $b$ . Thus the Pohlig-Hellman algorithm works, thanks to the assumption on  $b$ .
- We take  $s = t(H(M) - cwz) \pmod{p-1}$ .
- Yield the signature  $(r, s)$ .

We only have to prove that the signature is valid. First we check that  $0 \leq r < p$ . Next we have

$$y^r r^s \equiv y^{cw} (cw)^{t(H(M) - cwz)} \equiv y^{cw} g^{H(M) - cwz} \equiv g^{H(M)} \pmod{p}.$$

Therefore the signature is valid.  
→ use subgroups of prime order.

# Conclusion

- **insecurity cases**

- particular cases for parameters (short exponents...)
- groups with smooth order
- side channels

- **security analysis**

- key recovery problem
- decryption/forgery problem
- paradoxical cases when they are equivalent

## References — i

- **Håstad.**  
On Using RSA with Low Exponent in a Public Key Network.  
In *CRYPTO 1985*, LNCS 218.
- **Wiener.** Cryptanalysis of Short RSA Secret Exponents.  
In *EUROCRYPT 1989*, LNCS 434.
- In *EUROCRYPT 1996*, LNCS 1070:  
**Coppersmith.** Finding a Small Root of a Univariate Modular Equation.  
**Coppersmith.** Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known.  
**Coppersmith, Franklin, Patarin, Reiter.**  
Low-Exponent RSA with Related Messages.  
**Van Oorschot, Wiener.**  
On Diffie-Hellman Key Agreement with Short Exponents.  
**Bleichenbacher.** Generating ElGamal Signatures Without Knowing the Secret Key.

## References — ii

- **Kocher.**  
Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems.  
In *CRYPTO 1996*, LNCS 1109.
- **Boneh, DeMillo, Lipton.** On the Importance of Checking Cryptographic Protocols for Faults.  
In *EUROCRYPT 1997*, LNCS 1233.
- **Bleichenbacher.**  
Chosen Ciphertext Attacks against Protocols Based on RSA Encryption Standard PKCS #1.  
In *CRYPTO 1998*, LNCS 1462.
- **Kocher.** Differential Power Analysis.  
In *CRYPTO 1999*, LNCS 1666.
- **Bardou, Focardi, Kawamoto, Simionato, Steel, Tsay.**  
Efficient Padding Oracle Attacks on Cryptographic Hardware.  
In *CRYPTO 2012*, LNCS 7417.

# Train Yourself

- RSA:
  - midterm exam 2008–09 ex1 ex3
  - midterm exam 2009–10 ex2 (Wiener attack)
  - midterm exam 2010–11 ex1
  - midterm exam 2012–13 ex3 (broadcast encryption)
  - midterm exam 2015–16 ex1 (secret modulus recovery)
- threshold implementation: midterm exam 2017–18 ex1
- dedicated cryptanalysis: midterm exam 2013–14 ex1
- predicate encryption: midterm exam 2013–14 ex2















## 1 The Cryptographic Zoo

## 2 Cryptographic Security Models

## 3 Cryptanalysis (Public-Key)

## 4 The Power of Interaction

## 5 Cryptanalysis (Conventional)

## 6 Proving Security

## 1 The Cryptographic Zoo

## 2 Cryptographic Security Models

## 3 Cryptanalysis (Public-Key)

## 4 The Power of Interaction

- Interactive Proofs
- Zero-Knowledge
- Zero-Knowledge Construction from  $\Sigma$  Protocol
- Setup Models
- A Building Block for Making Cryptographic Primitives

## 5 Cryptanalysis (Conventional)

## 6 Proving Security

# Motivation

foundations: study the theory, models, and primitives for interactive proofs

this allows

- to make access control
- to make signature schemes
- to make encryption
- to give a methodology to analyze security
- ...

# Definitions

- $Z$  is an **alphabet** (a set of letters)
- $Z^*$  is the set of finite sequences of  $Z$  elements, including the empty sequence  $\perp$
- $L$  is a **language** (subset of  $Z^*$ )
- $x$  is a **word** (element of  $Z^*$ )
- $w$  is a **witness** (element of  $Z^*$ )
- $R(x, w)$  is a **relation** on  $x$  and  $w$  (either true or false)

$$L = \{x \in Z^* ; \exists w \in Z^* \quad R(x, w)\}$$

Problem: given  $x \in Z^*$ , prove that  $x \in L$

# Equivalent Form

- $Z$  is an **alphabet** (a set of letters)
- $Z^*$  is the set of finite sequences of  $Z$  elements, including the empty sequence  $\perp$
- $L$  is a **set of provable statements**
- $x$  is a **statement**
- $w$  is a **proof**
- $R(x, w)$  is a **proof checker predicate** on  $x$  and  $w$

$$L = \{x \in Z^*; \exists w \in Z^* \quad R(x, w)\}$$

Problem: given  $x \in Z^*$ , prove  $x$

non deterministic  
polynomial

## Definition ( $\mathcal{NP}$ Language)

A language  $L$  over the alphabet  $Z$  belongs to the class  $\mathcal{NP}$  if there exists a relation  $R$  and a polynomial  $P$  such that

- $R(x, w)$  can be evaluated in polynomial time (with respect to the length of  $x$  and  $w$ )
- $L = \{x \in Z^*; \exists w \in Z^* \quad R(x, w), |w| \leq P(|x|)\}$

= language of statements which can be proven by a proof with polynomial size

## Definition (co- $\mathcal{NP}$ Language)

A language  $L$  over the alphabet  $Z$  belongs to the class co- $\mathcal{NP}$  if  $Z^* - L \in \mathcal{NP}$ , i.e. there exist a relation  $R$  and a polynomial  $P$  such that

- $R(x, w)$  can be evaluated in polynomial time (with respect to the length of  $x$  and  $w$ )
- $L = \{x \in Z^*; \forall w \in Z^* \quad |w| \leq P(|x|) \implies \neg R(x, w)\}$

## Definition ( $\mathcal{P}$ Language)

A language  $L$  over the alphabet  $Z$  belongs to the class  $\mathcal{P}$  if there exists a polynomial time deterministic algorithm  $\mathcal{A}$  such that

$$L = \{x \in Z^*; \mathcal{A}(x) \rightarrow \text{accept}\}$$

= language of statements which are checked in polynomial time  
 $\text{NP} \iff$  membership can be decided in **non-deterministic** polynomial time

# $\mathcal{P}$ vs $\mathcal{NP}$ Problem

clearly:

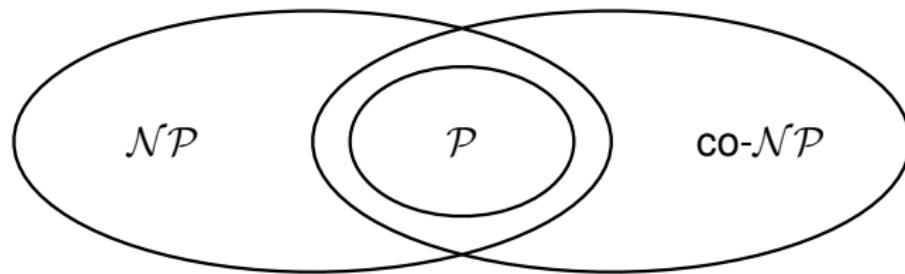
$$\mathcal{P} \subseteq \mathcal{NP}$$

**big** open question:

$$\mathcal{P} = \mathcal{NP} \quad \text{or} \quad \mathcal{P} \neq \mathcal{NP} \quad ?$$

# $\mathcal{NP}$ vs co- $\mathcal{NP}$ Problem

$\mathcal{NP} = \text{co-}\mathcal{NP}$  or  $\mathcal{NP} \neq \text{co-}\mathcal{NP}$  ?



# Karp-Reduction

## Definition (Karp Reduction)

Given two languages  $L_1$  and  $L_2$  over an alphabet  $Z$ , we say that  $L_1$  (**Karp-reduces**) to  $L_2$  if there exists a function  $f$  which can be computed by a polynomial-time algorithm such that

$$\forall x \in Z^* \quad x \in L_1 \iff f(x) \in L_2$$

The Karp reduction is stronger than the Turing reduction  
( $\approx$  Turing reduction restricted to a single oracle access)

# NP-Hardness

## Definition (NP-Hardness)

A language  $L$  over the alphabet  $Z$  is **NP-hard** if for all  $L' \in \mathcal{NP}$ ,  $L'$  (Karp-)reduces to  $L$ .

It is **NP-complete** if it further belongs to  $\mathcal{NP}$ .

# NP-Hardness

Example: SAT, the set of Boolean terms  $r$  such that there exists an assignment of all variables in  $r$  which satisfies the term.

## Theorem (Cook 1971)

SAT is NP-complete.

### Proof intuition.

- if  $L' \in \mathcal{NP}$ , there exists a relation  $R$  for  $L'$
- for each  $x$ , the size of the potential witness is bounded so we can write  $r(x) = R(x, \cdot)$  as a Boolean term holding on variables corresponding to  $w$   
(a Boolean term has AND, OR, NOT gates and variables)
- for each  $x$ ,  $w$  is a witness for  $x$  iff this it defines an assignment satisfying  $r(x)$
- for each  $x$ ,  $x \in L'$  iff  $r(x)$  is satisfiable
- the  $x \mapsto r(x)$  is polynomially bounded



# Consequence

$$\text{SAT} \in \mathcal{P} \iff \mathcal{P} = \mathcal{NP}$$

# Interactive Machines

“a Turing machine with an extra communication tape”

## Definition

We say that  $\mathcal{A}$  is an polynomial/deterministic/probabilistic **interactive machine** if it is a polynomial/deterministic/probabilistic algorithm mapping an input  $x$  and a list  $m_1, \dots, m_n$  of **incoming messages** to an **outgoing message**  $\mathcal{A}(x, m_1, \dots, m_n; r)$ .

- Messages can end by a special termination symbol. In this case we call it a terminal message.
- $\mathcal{A}$  shall be such that if  $m_n$  is a terminal message then  $\mathcal{A}(x, m_1, \dots, m_n; r)$  is a terminal message.
- $(x, m_1, \dots, m_n; r)$  is called a **partial view** of  $\mathcal{A}$ . It is a **final view** if the last message is a terminal one.

An interactive machine is an algorithm computing a next-message from a partial view.

# Interactive System

Given two interactive machines  $\mathcal{A}$  and  $\mathcal{B}$ , an input  $x$ , random tapes  $r_A$  and  $r_B$ , we call  $\mathcal{A}$  the **initiator** and define

- $\text{exp} = \left( \mathcal{A}(r_A) \xleftrightarrow{x} \mathcal{B}(r_B) \right)$  the experiment consisting in defining

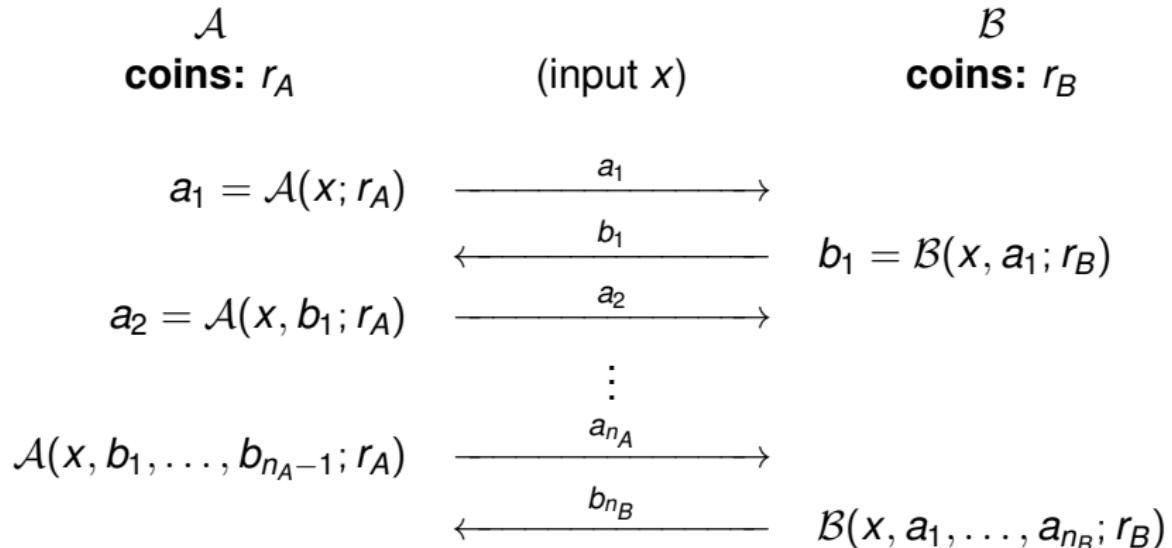
$$a_i = \mathcal{A}(x, b_1, \dots, b_{i-1}; r_A)$$

$$b_j = \mathcal{B}(x, a_1, \dots, a_j; r_B)$$

for  $i = 1, \dots, n_A$  such that  $a_{n_A}$  is the first final message  $a_i$  and  $j = 1, \dots, n_B$  such that  $b_{n_B}$  is the first final message  $b_j$

- $\text{Out}_{\mathcal{A}}(\text{exp}) = a_{n_A}$
- $\text{Out}_{\mathcal{B}}(\text{exp}) = b_{n_B}$
- $\text{View}_{\mathcal{A}}(\text{exp}) = (x, b_1, \dots, b_{n_A-1}; r_A)$
- $\text{View}_{\mathcal{B}}(\text{exp}) = (x, a_1, \dots, a_{n_B}; r_B)$

# Interactive System



$$\text{View}_{\mathcal{A}}(\text{exp}) = (x, b_1, \dots, b_{n_A-1}; r_A)$$

$$\text{View}_{\mathcal{B}}(\text{exp}) = (x, a_1, a_2, \dots, a_{n_B}; r_B)$$

# Example: Coin Flipping Game

challenge in the future



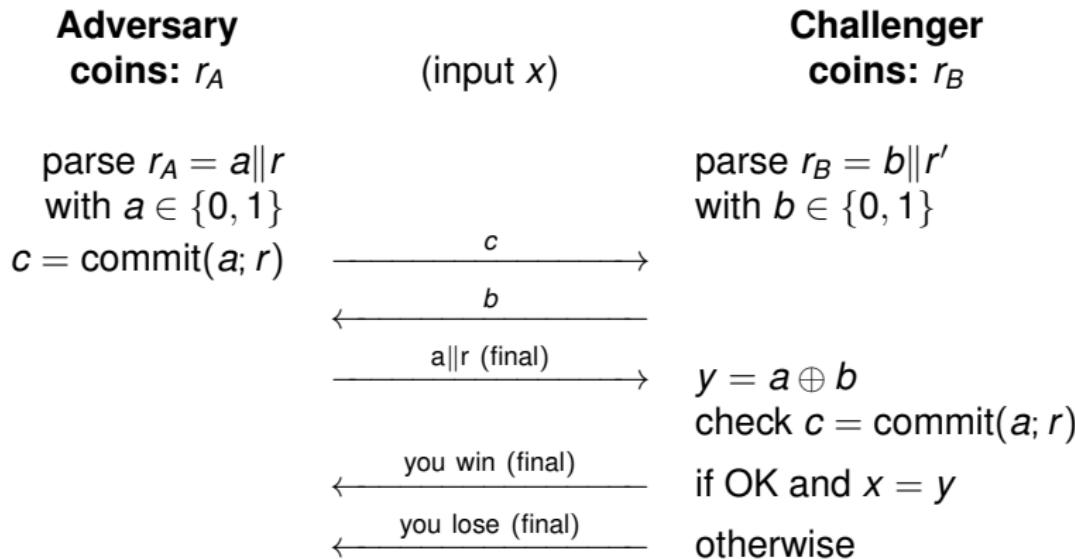
- **game:**

given a bit  $x$ ,  $\mathcal{A}$  and  $\mathcal{C}$  flip a coin together and  $\mathcal{A}$  wins if it is  $x$

- **coin flipping:**

- ➊  $\mathcal{A}$  commits to a random bit  $a$
- ➋  $\mathcal{C}$  flips a bit  $b$  and sends it
- ➌  $\mathcal{A}$  opens  $a$
- ➍ the final bit is  $a \oplus b$

# Example: Coin Flipping Game



$$\text{View}_C \left( \mathcal{A}(a\|r) \xleftrightarrow{x} \mathcal{C}(b\|r') \right) = (x, c, a\|r; b\|r')$$

$$\text{Out}_C \left( \mathcal{A}(a\|r) \xleftrightarrow{x} \mathcal{C}(b\|r') \right) = \begin{cases} \text{you win} & \text{if } x = a \oplus b \\ & \text{and } c = \text{commit}(a; r) \\ \text{you lose} & \text{otherwise} \end{cases}$$

# Interactive Proof

for a language  $L$

(prover, verifier)

## Definition

Given a language  $L$  over an alphabet  $Z$ , an **interactive proof system** for  $L$  is a pair  $(\mathcal{P}, \mathcal{V})$  of interactive machines such that there exist a polynomial  $P$ ,  $\beta$  such that  $0 \leq \beta < 1$  and

- termination: for any  $x$ , the total complexity of  $\mathcal{V}$  (until termination) in  $\mathcal{P} \leftrightarrow \mathcal{V}(r)$  is bounded by  $P(|x|)$
- **perfect completeness**: for any  $x \in L$  then

$$\Pr_{r_P, r_V} \left[ \text{Out}_{\mathcal{V}} \left( \mathcal{P}(r_P) \leftrightarrow \mathcal{V}(r_V) \right) = \text{accept} \right] = 1$$

- **$\beta$ -soundness**: for any  $x \notin L$  and any algorithm  $\mathcal{P}^*$  then

$$\Pr_{r_P, r_V} \left[ \text{Out}_{\mathcal{V}} \left( \mathcal{P}^*(r_P) \leftrightarrow \mathcal{V}(r_V) \right) = \text{accept} \right] \stackrel{\text{try to cheat}}{\leq} \beta$$

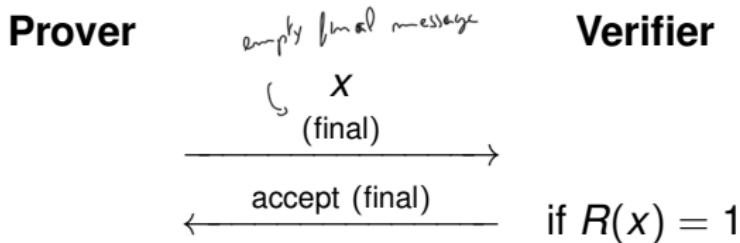
# Notes

- we assume **no bound** on the prover  $\mathcal{P}$  (powerful prover)  
w.l.o.g. it can be assumed deterministic
- the verifier  $\mathcal{V}$  is polynomially bounded
- variants consider imperfect completeness *(here consider perfect completeness)*
- $\beta$  is the maximal probability that  $\mathcal{V}$  can be fooled

# Example: Proof of some $\mathcal{P}$ Language

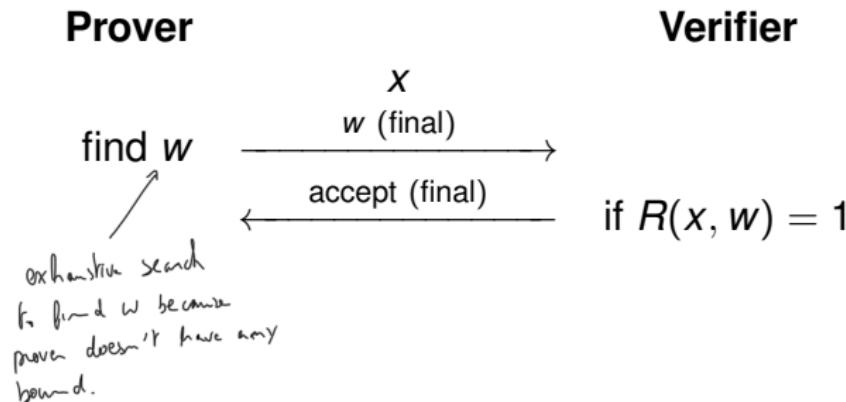
acceptance message  
↓

**language:** set of  $x$  such that  $R(x) = 1$  where  $R$  is a deterministic and polynomially bounded Turing machine



# Example: Proof of some $\mathcal{NP}$ Language

**language:** set of  $x$  such that there is some  $w$  such that  
 $R(x, w) = 1$  where  $R$  is a deterministic and  
polynomially bounded Turing machine

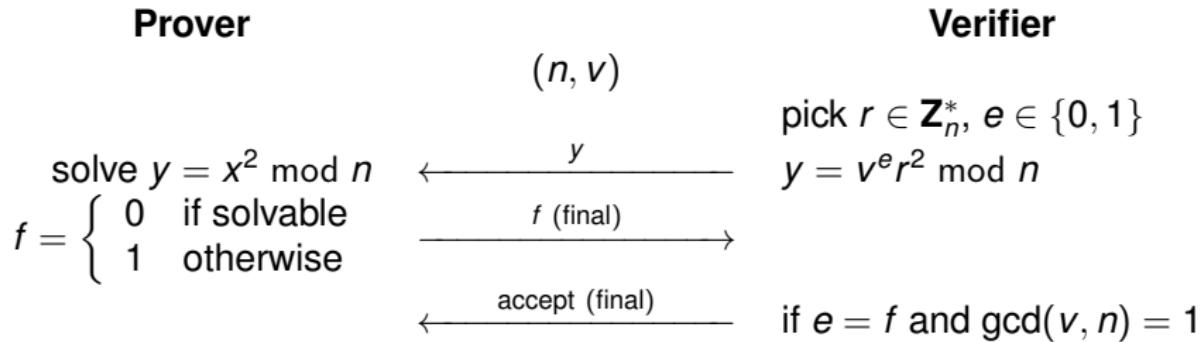


# Example: Goldwasser-Micali-Rackoff 1985

## A co-NP Case

reminder:  $\text{QR}(n)$  is the set of all squares modulo  $n$

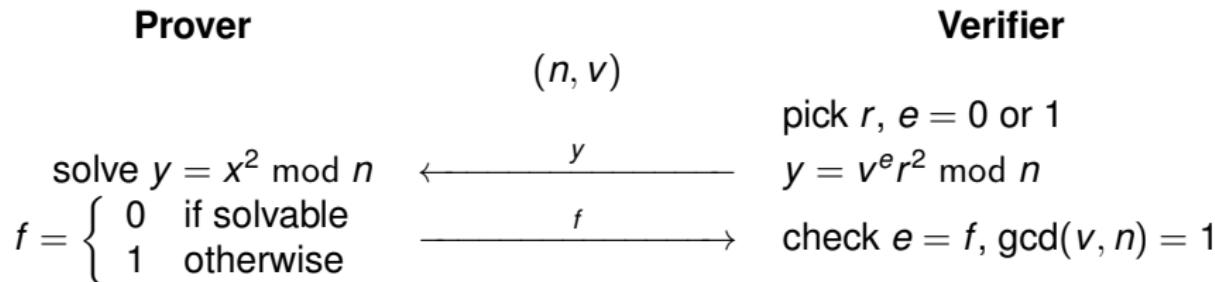
**language:** set of pairs  $(n, v)$  such that  $v \in \mathbf{Z}_n^*$  and  $v \notin \text{QR}(n)$



Termination: Verifier runs in polynomial time with respect to the size of  $(n, v)$

## Example: GMR85 Completeness

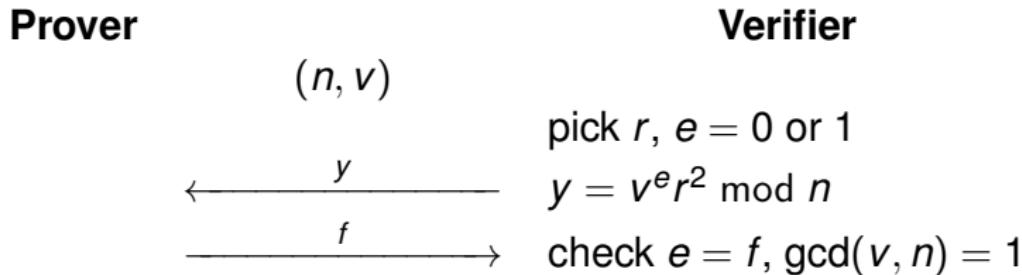
completeness:



- assume that  $\mathcal{P}$  and  $\mathcal{V}$  follow the protocol and  $(n, v) \in L$
- $v^e r^2 \equiv x^2$  is solvable iff  $e = 0$
- we always have  $e = f$ , so  $\mathcal{V}$  always accept

## Example: GMR85 Soundness

$\beta$ -soundness with  $\beta = \frac{1}{2}$ :



- assume that  $\mathcal{V}$  follows the protocol and  $(n, v) \notin L$
- we can write  $v = s^2 \bmod n$  then  $y = (s^e r)^2 \bmod n$
- if  $r \in_U \mathbf{Z}_n^*$  then  $s^e r \in_U \mathbf{Z}_n^*$  whatever  $e$
- $y$  and  $e$  are independent thus  $e$  and  $f$  are independent
- $\Pr[\text{accept}] = \frac{1}{2}$

# Sequential Composition

Given an interactive proof system  $(\mathcal{P}, \mathcal{V})$  for  $L$  which is complete and  $\beta$ -sound we define a new proof system  $(\mathcal{P}', \mathcal{V}')$  as follows:

- $\mathcal{P}'$  resp  $\mathcal{V}'$  simulates  $\mathcal{P}$  resp  $\mathcal{V}$  but have no terminal message until  $n$  iterations are made
- after an iteration completes, they restart the entire protocol with fresh random coins
- $\mathcal{V}'$  accepts all iterations accepted

the new interactive proof system is complete and  $\beta'$ -sound with

$$\beta' \stackrel{\text{next th}}{=} \beta^n$$

so  $\beta' \rightarrow 0$

Conclusion: by **sequential** composition we can tune  $\beta$  as close to 0 as we want

# Soundness Result for Sequential Composition

## Theorem

Consider an interactive proof system  $(\mathcal{P}, \mathcal{V})$  which is  $\beta$ -sound. Given  $n$ , we construct the interactive proof system  $(\mathcal{P}', \mathcal{V}')$  making  $n$  **sequential** iterations of  $(\mathcal{P}, \mathcal{V})$  then accepting if all iterations lead to an acceptance result (see previous slide). This new interactive proof system is  $\beta'$ -sound where

$$\beta' = \beta^n$$

▶ skip details

# Technical Lemma

## Lemma

Assume that no adversary can succeed one protocol session with probability higher than  $\beta$ .

Then, for each adversary  $\mathcal{A}$  repeating the protocol  $n$  times and any  $I \subseteq \{1, \dots, n\}$ , we have

$$\Pr \left[ \bigwedge_{i \in I} \mathcal{A} \text{ succeeds iteration } i \right] \leq \beta^{\#I}$$

Theorem follows with  $I = \{1, \dots, n\}$ .

## Proof of Lemma

- proceed by induction on  $i$  such that  $I \subseteq \{1, \dots, i\}$ :
  - trivial for  $i = 0$  ( $I$  empty)
  - assume this is true for  $i - 1$  and prove it for  $i$ :
    - let  $I \subseteq \{1, \dots, i\}$
    - if  $I \subseteq \{1, \dots, i - 1\}$ , the result is proven
    - otherwise, let  $I = I' \cup \{i\}$
    - consider an adversary  $\mathcal{B}$  who plays with  $\mathcal{A}$  and simulate  $i - 1$  honest verifiers at random
      - if  $\mathcal{A}$  passes each iteration  $i$  s.t.  $i \in I'$  then proceed
      - otherwise, reset  $\mathcal{A}$  and restart
    - play a challenge session with  $\mathcal{A}$  in the  $i$ th iteration
    - if  $\mathcal{B}$  halts, since he passes with probability at most  $\beta$ , we have

$$\Pr \left[ \mathcal{A} \text{ succeeds iteration } i \mid \bigwedge_{j \in I'} \mathcal{A} \text{ succeeds iteration } j \right] \leq \beta$$

- by induction, we prove (if  $\mathcal{B}$  does not halt as well) that

$$\Pr \left[ \bigwedge_{i \in I} \mathcal{A} \text{ succeeds iteration } i \right] \leq \beta^{\#I}$$

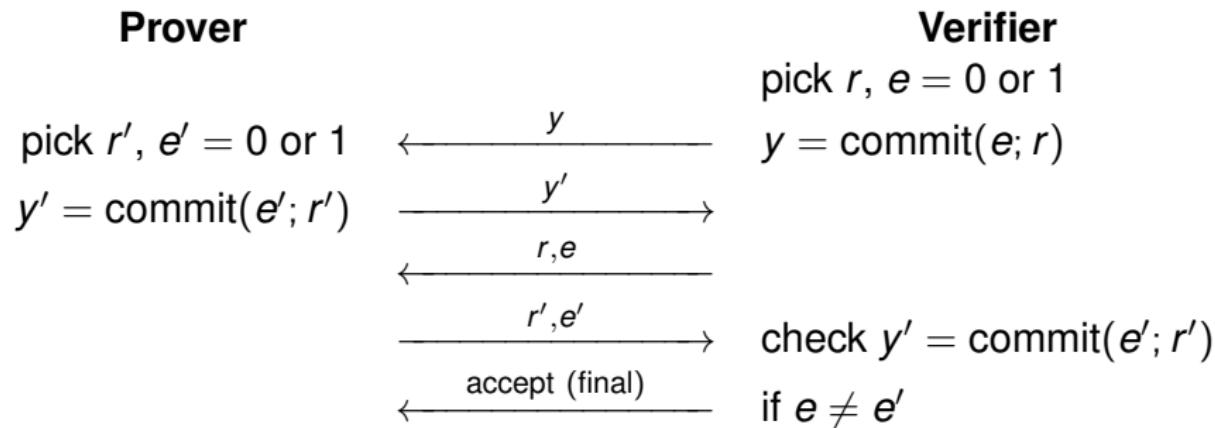


# Tricky Things with Parallel Composition

- OK for parallel composition of *proofs*
- not OK if we consider **computational soundness**
- WARNING: parallel composition does not always work in protocols

# Example: DD Protocol

Bellare-Impagliazzo-Naor 1997



- commitment perfectly hiding and computationally binding
- for any (polynomial) prover, probability of acceptance is negligibly close to  $\frac{1}{2}$

# Parallel Composition of DD

## Prover

pick  $r'_1, r'_2, e'_1, e'_2 = 0 \text{ or } 1$

$y'_i = \text{commit}(e'_i; r'_i)$

$\xleftarrow{y_1, y_2}$

$\xrightarrow{y'_1, y'_2}$

$\xleftarrow{r_1, r_2, e_1, e_2}$

$\xrightarrow{r'_1, r'_2, e'_1, e'_2}$

$\xleftarrow{\text{accept (final)}}$

## Verifier

pick  $r_1, r_2, e_1, e_2 = 0 \text{ or } 1$

$y_i = \text{commit}(e_i; r_i)$

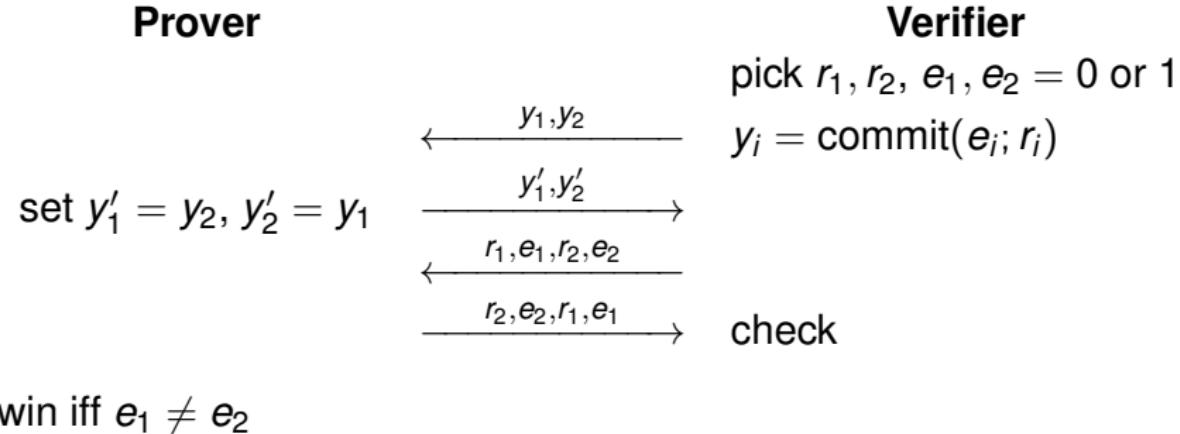
check  $y'_i = \text{commit}(e'_i; r'_i)$

if  $e_1 \neq e'_1$  and  $e_2 \neq e'_2$

- we would expect that for any prover, the probability to pass is  $\leq \frac{1}{4} + \text{negl...}$

## Example: DD Protocol

- a prover can win 2 parallel repetitions with probability  $\frac{1}{2}$  (instead of  $\frac{1}{4}$ )



## Definition ( $\mathcal{IP}$ Language)

A language  $L$  over the alphabet  $Z$  belongs to the class  $\mathcal{IP}$  if there exists an interactive proof system for  $L$ .

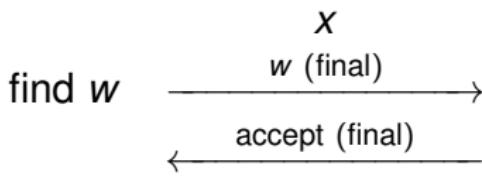
# Example: a Proof System for any NP Language

## Theorem

$$\mathcal{NP} \subseteq \mathcal{IP}$$

**Proof.** Let  $L$  be a language in the  $\mathcal{NP}$  class: define  $R$ .

### Prover



### Verifier

$$\text{if } R(x, w) = 1$$

- we have perfect completeness
- we have perfect soundness ( $\beta = 0$ )

□

# *PSPACE* Class

## **Definition (*PSPACE* Language)**

A language  $L$  over the alphabet  $Z$  belongs to the class  $\text{PSPACE}$  if there exists an algorithm  $\mathcal{A}$  working with a **polynomially bounded memory** such that

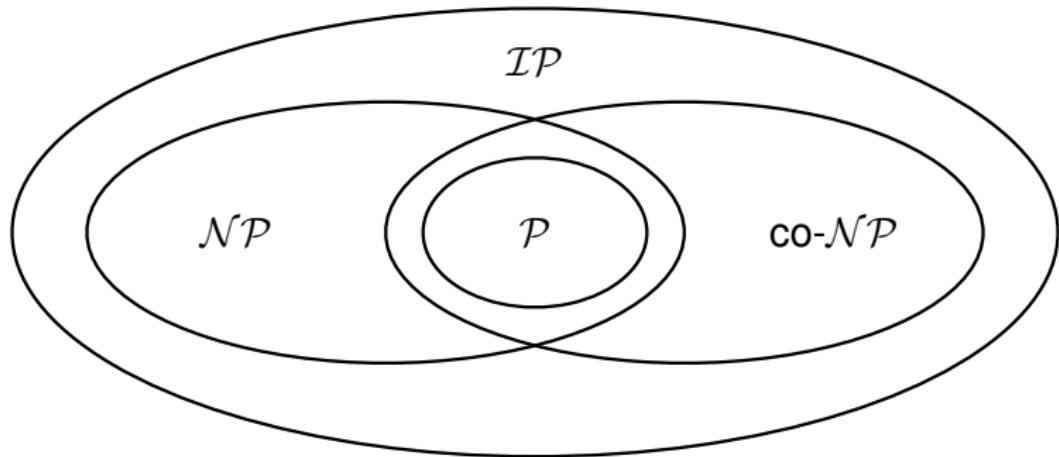
$$L = \{x \in Z^*; \mathcal{A}(x) \rightarrow \text{accept}\}$$

We have  $\mathcal{NP} \subseteq \text{PSPACE}$  because  $\mathcal{A}$  can do an exhaustive search on the witness and check it with limited space.

## **Theorem (Shamir 1992)**

$$\mathcal{IP} = \text{PSPACE}.$$

# Complexity Classes



## 1 The Cryptographic Zoo

## 2 Cryptographic Security Models

## 3 Cryptanalysis (Public-Key)

## 4 The Power of Interaction

- Interactive Proofs
- **Zero-Knowledge**
- Zero-Knowledge Construction from  $\Sigma$  Protocol
- Setup Models
- A Building Block for Making Cryptographic Primitives

## 5 Cryptanalysis (Conventional)

## 6 Proving Security

# Zero-Knowledge Interactive Proof

## Definition (ZKIP)

An interactive proof system  $(\mathcal{P}, \mathcal{V})$  is  $*$ -zero-knowledge if for any ppt interactive machine  $\mathcal{V}^*$  there exists a ppt algorithm  $\mathcal{S}$  such that for any  $x \in L$   $\text{View}_{\mathcal{V}^*}(\mathcal{P}(r_P) \xleftrightarrow{x} \mathcal{V}^*(r_V))$  and  $\mathcal{S}(x; r)$  produce  $*$ -indistinguishable distributions.

$$\forall \mathcal{V}^* \text{ ppt } \exists \mathcal{S} \text{ ppt } \forall x \in L \text{ View}_{\mathcal{V}^*}(x) \stackrel{*}{\sim} \text{Out}_{\mathcal{S}}(x)$$

Several  $*$ -indistinguishability notions:

- **$*$ =perfect:**  $*$ -identical means identical
- **$*$ =statistical:**  $*$ -identical means the statistical distance is negligible in terms of  $|x|$
- **$*$ =computational:**  $*$ -identical means any ppt distinguisher has an advantage which is negligible in terms of  $|x|$

bounded by a polynomial algorithm

# Zero-Knowledge Levels

$$\forall x \quad \text{View}_{\mathcal{V}^*}(x) \stackrel{*}{\sim} \text{Out}_{\mathcal{S}}(x)$$

$\text{View}_{\mathcal{V}^*}(x)$ : what comes from interacting with powerful prover

$\text{Out}_{\mathcal{S}}(x)$ : what comes from powerless simulator

- **Perfect.** No matter the complexity of the distinguisher, the advantage is null.

$$\forall x \forall v \quad \Pr[\text{View}_{\mathcal{V}^*}(x) = v] = \Pr[\text{Out}_{\mathcal{S}}(x) = v]$$

- **Statistical.** No matter the complexity of the distinguisher, the advantage is negligible.

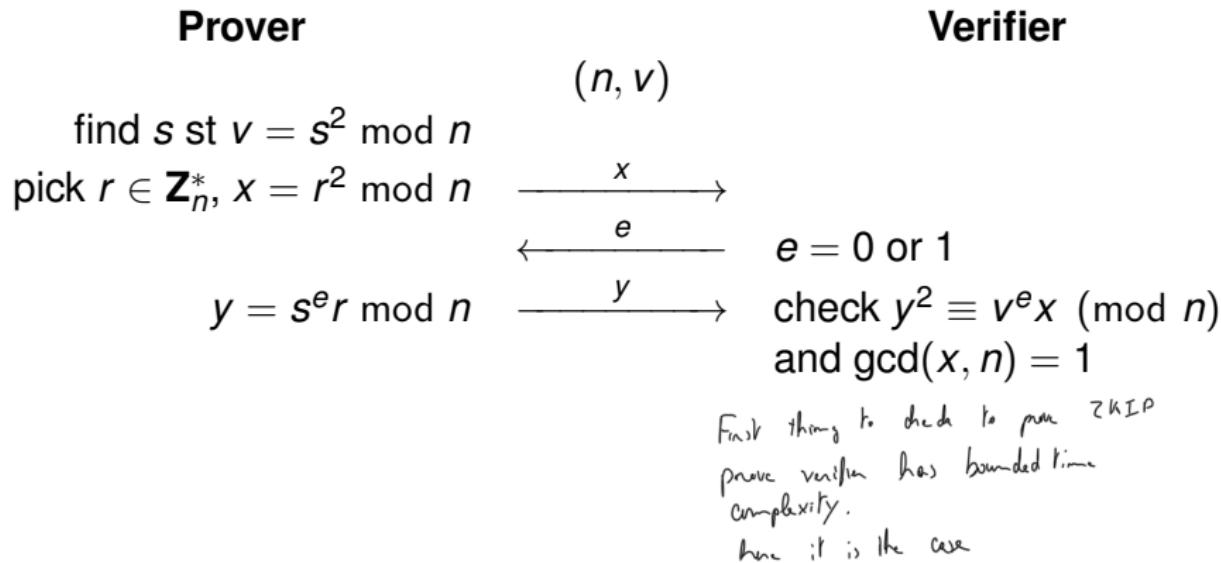
$$\forall x \forall \mathcal{D} \quad |\Pr[\mathcal{D}(\text{View}_{\mathcal{V}^*}(x))] - \Pr[\mathcal{D}(\text{Out}_{\mathcal{S}}(x))]| = \text{negl}(|x|)$$

- **Computational.** With a distinguisher of ppt complexity, the advantage is negligible.

$$\forall x \forall \mathcal{D} \text{ ppt} \quad |\Pr[\mathcal{D}(\text{View}_{\mathcal{V}^*}(x))] - \Pr[\mathcal{D}(\text{Out}_{\mathcal{S}}(x))]| = \text{negl}(|x|)$$

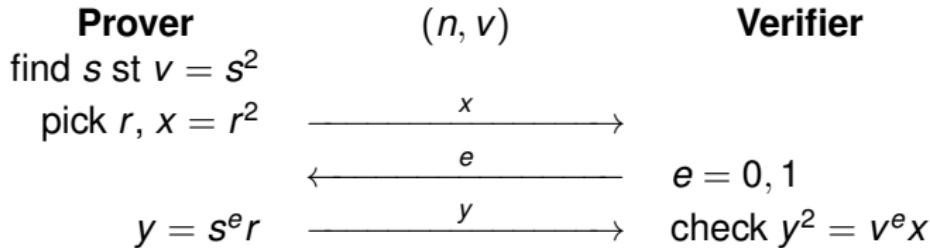
# Example: Goldwasser-Micali-Rackoff 1989

**language:** set of pairs  $(n, v)$  such that  $v$  is a quadratic residue modulo  $n$



# GMR89 - Completeness

completeness:

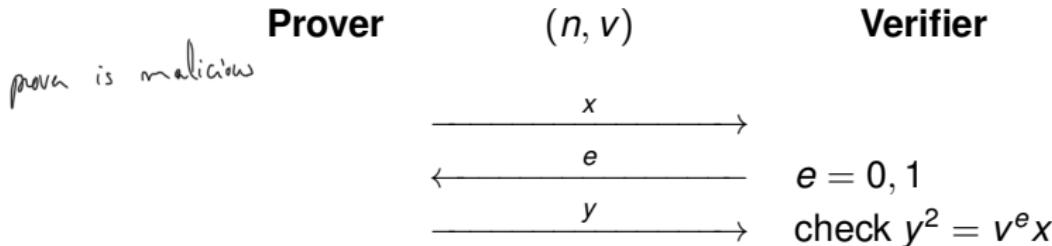


if Prover and Verifier follow the protocol, then it always succeeds

$$\Pr[\mathcal{P} \leftrightarrow \mathcal{V} \text{ accept}] = 1$$

## GMR89 - Soundness

$\beta$ -soundness with  $\beta = \frac{1}{2}$ :

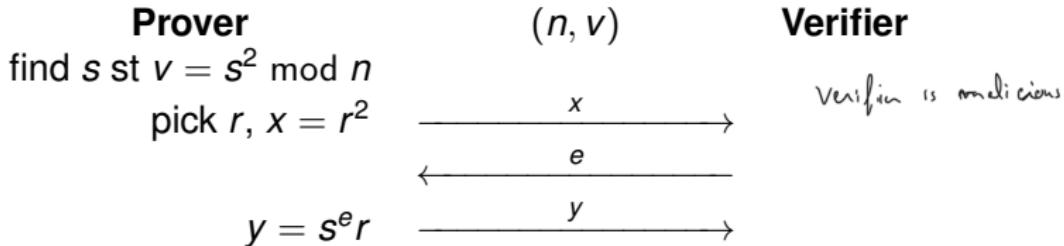


if Verifier follows the protocol and accepts with probability  $> \frac{1}{2}$  then  $v$  is a quadratic residue

$$\Pr[\mathcal{P}^* \leftrightarrow \mathcal{V} \text{ accept}] > \frac{1}{2} \implies v \text{ quadratic residue}$$

- If the protocol succeeds with probability  $> \frac{1}{2}$ , there must be at least one  $r_P$  for which the probability (over  $e$ ) that the verifier accepts is  $> \frac{1}{2}$
- there is one  $x$  for which the verifier accepts  $y_e$  to challenge  $e$
- $y_0^2 = x$  and  $y_1^2 = vx$  so  $v = (y_1/y_0)^2$ :  $v$  is a quadratic residue

# GMR89 - Zero-Knowledge



**Zero-Knowledge:** if Prover follows the protocol, we can make a Simulator which generates random views  $(n, v, x, y; r_V)$  with the same distribution: for all  $(n, v) \in L$ ,

$$\forall \mathcal{V}^* \exists \mathcal{S} \left( \text{View}(\mathcal{P} \xleftrightarrow{n, v} \mathcal{V}^*) \right) = \text{distribution}(\mathcal{S}(n, v))$$

- pick  $e_0 \in \{0, 1\}$  at random, pick  $r_V$
- pick  $x = y^2 v^{-e_0}$  with  $y$  random  
(distribution of  $x$  like in  $P$  and independent from  $e_0$ )
- run  $\mathcal{V}^*(n, v, x; r_V)$
- if  $\mathcal{V}^*$  yields  $e \neq e_0$ , try again, otherwise, output  $(n, v, x, y; r_V)$

to do: the distribution is the same +  $\mathcal{S}$  is polynomially bounded

# Black-Box Zero-Knowledge

## Definition (Black-Box ZK)

An interactive proof system  $(\mathcal{P}, \mathcal{V})$  is  $*$ -**black-box zero-knowledge** if there exists a ppt oracle machine  $\mathcal{S}$  such that any ppt interactive machine  $\mathcal{V}^*$  and any  $x \in L$ ,

$$\text{View}_{\mathcal{V}^*} \left( \mathcal{P}(r_P) \xleftrightarrow{*} \mathcal{V}^*(r_V) \right)$$

and  $\mathcal{S}^{\mathcal{V}^*}(x; r)$  produce  $*$ -identical distributions.

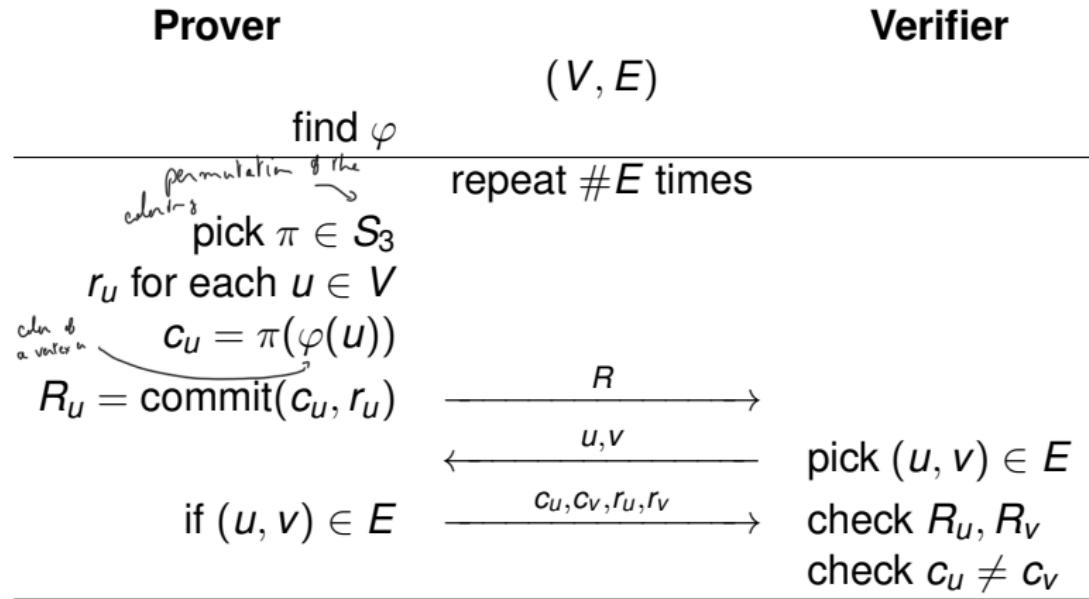
$\mathcal{S}^{\mathcal{V}^*}$  means that algorithm  $\mathcal{S}$  can use the algorithm  $\mathcal{V}^*$  as a subroutine and select all its inputs

# Example

In the GMR89 protocol the simulator was black-box!

# Example: Goldreich-Micali-Wigderson 1986

**language:** set of graphs  $(V, E)$  such that there exists a mapping  $\varphi : V \rightarrow \{1, 2, 3\}$  such that for each  $(u, v) \in E$  then  $\varphi(u) \neq \varphi(v)$



# Graph Coloring

- a **graph** is a pair  $(V, E)$  such that  $E \subseteq V \times V$   
 $V$  is a set of **vertices**  
 $E$  is a set of **edges**  
an edge  $(u, v) \in E$  is said to go from vertex  $u$  to vertex  $v$
- a function  $\varphi : V \rightarrow \{1, 2, \dots, n\}$  is a **coloring** of  $(V, E)$  in  $n$  colors if for any  $(u, v) \in E$  we have  $\varphi(u) \neq \varphi(v)$
- graph 2-colorability can be decided in linear time
- graph 3-colorability is an NP-complete problem

# GMW86 - Completeness

completeness:

| Prover                          | $\varphi$                          | $(V, E)$            |
|---------------------------------|------------------------------------|---------------------|
| pick $\pi \in S_3$ , $r$        |                                    | (#E times)          |
| $c_u = \pi(\varphi(u))$         |                                    |                     |
| $R_u = \text{commit}(c_u, r_u)$ | $\xrightarrow{R}$                  |                     |
|                                 | $\xleftarrow{u, v}$                |                     |
|                                 | $\xrightarrow{c_u, c_v, r_u, r_v}$ | pick $(u, v) \in E$ |
|                                 |                                    | check               |

if Prover and Verifier follow the protocol, then it always succeeds

$$\Pr[\mathcal{P} \leftrightarrow \mathcal{V} \text{ accept}] = 1$$

$$\text{if } \varphi(u) \neq \varphi(v) \text{ then } c_u \neq c_v$$

## GMW86 - Soundness (sketch)

$\frac{2}{3}$ -soundness with  $\beta = \frac{2}{3}$  for  $\#E \geq 5$ :

| Prover                                | $(V, E)$<br>( $\#E$ times)   |
|---------------------------------------|------------------------------|
| $R$<br>$u, v$<br>$c_u, c_v, r_u, r_v$ | pick $(u, v) \in E$<br>check |

$$\Pr[\mathcal{P}^* \leftrightarrow \mathcal{V} \text{ accept}] \leq \frac{2}{3} \Leftarrow \text{graph non-colorable}$$

- if there is no possible  $\varphi$  then at least one edge is incorrect so each iteration may fail with probability at least  $\frac{1}{\#E}$  (assume the commitment to be perfectly binding!)
- all iterations pass with probability at most  $\left(1 - \frac{1}{\#E}\right)^{\#E} \leq e^{-1}$

# GMW86 - Zero-Knowledge (sketch)

Complexity  $O(\#E^2)$

| Prover                          | $\varphi$           | $(V, E)$                           |
|---------------------------------|---------------------|------------------------------------|
| pick $\pi \in S_3$ , $r$        |                     | ( $\#E$ times)                     |
| $c_u = \pi(\varphi(u))$         |                     |                                    |
| $R_u = \text{commit}(c_u, r_u)$ | $\xrightarrow{R}$   |                                    |
|                                 | $\xleftarrow{u, v}$ |                                    |
|                                 |                     | $\xrightarrow{c_u, c_v, r_u, r_v}$ |

**Zero-Knowledge:** if Prover follows the protocol, we can make a Simulator which generate random views with the same distribution

$$\exists \mathcal{S} \quad \forall \mathcal{V}^* \text{ distribution } \left( \text{View}(\mathcal{P} \xleftrightarrow{V, E} \mathcal{V}^*) \right) = \text{distribution } \left( \mathcal{S}^{\mathcal{V}^*}(V, E) \right)$$

- guess  $(u, v)$ , make  $c_u, c_v, r_u, r_v$  such that it works and make random commitments for the other vertices
- run  $\mathcal{V}^*$  on the constructed  $R$
- if  $\mathcal{V}^*$  yields  $e \neq (u, v)$ , rewind it and try again, otherwise, output  $c_{u,v}, r_u, r_v$

## Some Further Technicalities

- assuming that the commitment is computationally hiding and perfectly binding
  - soundness works  
(prover can open commitment on at most one color)
  - zero-knowledge is computational  
(cannot distinguish simulated commitments from real ones)
- assuming that the commitment is perfectly hiding and computationally binding
  - zero-knowledge is perfect (distributions are equal)
  - soundness is only true in a weaker form

# Consequence: NP has Zero-Knowledge Proofs

## Theorem (Goldreich-Micali-Wigderson 1986)

*Assuming that a computationally hiding and perfectly binding commitment exists, for all language  $L \in \text{NP}$  there exists a computational zero-knowledge interactive proof.*

# Proof of Knowledge

## Definition

Given a language  $L \in \mathcal{NP}$  over an alphabet  $Z$  defined by a relation  $R$ , an **interactive proof of knowledge** for  $L$  is a pair  $(\mathcal{P}, \mathcal{V})$  of interactive machines such that there exist a polynomial  $P$  and  $\beta$  such that  $0 \leq \beta < 1$  and

- termination: [as for interactive proof systems]
- **perfect completeness**: [as for interactive proof systems]
- **$\beta$ -soundness**: there exists an oracle algorithm  $\mathcal{E}$  called **extractor** verifying what follows. For any  $\mathcal{P}^*$  we let

$$\varepsilon(x) = \Pr_{r_P, r_V} \left[ \text{Out}_{\mathcal{V}} \left( \mathcal{P}^*(r_P) \xleftrightarrow{x} \mathcal{V}(r_V) \right) = \text{accept} \right]$$

If  $\varepsilon(x) > \beta$  then  $\mathcal{E}^{\mathcal{P}^*}(x)$  outputs  $w$  such that  $R(x, w)$  holds with complexity at most  $P(|x|)/(\varepsilon(x) - \beta)$ .

$\mathcal{E}^{\mathcal{P}^*}$  means that  $\mathcal{E}$  can use  $\mathcal{P}^*$  as a subroutine and select its inputs (note: access to  $\mathcal{P}^*$  counts as 1 in the complexity)

# Typical Prover

the typical provers that we have seen so far:

- ➊  $\mathcal{P}$  finds  $w$  such that  $R(x, w)$  by exhaustive search
- ➋  $\mathcal{P}$  runs an algorithm based on  $x$  and  $w$

typically: second step is polynomial

equivalent definition

- $\mathcal{P}$  uses  $w$  as a private input
- $\mathcal{P}$  is a polynomially bounded algorithm

this is closer to practical use as we want  $\mathcal{P}$  to prove efficiently  
that he knows  $w$

## 1 The Cryptographic Zoo

## 2 Cryptographic Security Models

## 3 Cryptanalysis (Public-Key)

## 4 The Power of Interaction

- Interactive Proofs
- Zero-Knowledge
- **Zero-Knowledge Construction from  $\Sigma$  Protocol**
- Setup Models
- A Building Block for Making Cryptographic Primitives

## 5 Cryptanalysis (Conventional)

## 6 Proving Security

# Motivation

$\Sigma$ -protocols make the design of ZK proofs of knowledge easier

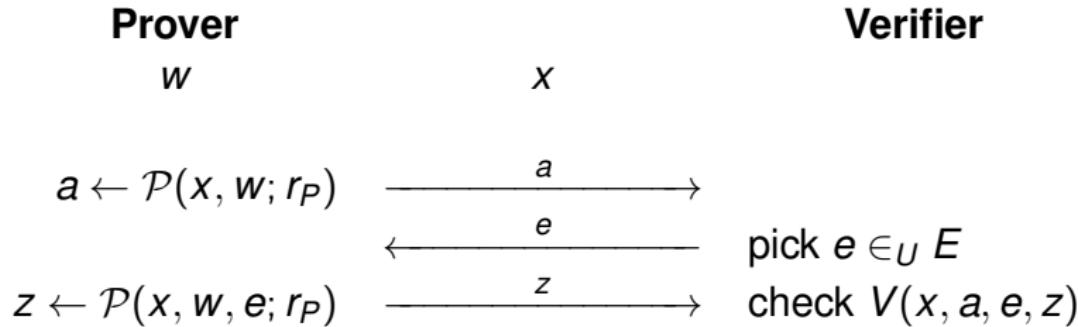
# $\Sigma$ Protocol

## Definition

Given a language  $L \in \mathcal{NP}$  over an alphabet  $Z$  defined by a relation  $R$ , a  $\Sigma$ -protocol for  $L$  is a pair of interactive machines  $\mathcal{P}(x, w)$  and  $\mathcal{V}(x)$  such that

- $\mathcal{V}$  is polynomially bounded
- **3-move:**  $\mathcal{P}$  starts with a message  $a$ ,  $\mathcal{V}$  answers with a challenge  $e \in_U E$ ,  $\mathcal{P}$  terminates with a response  $z$ ,  $\mathcal{V}$  accepts (always for  $x \in L$ ) or reject only depending on  $(x, a, e, z)$
- **special soundness:** there exists a polynomially bounded algorithm  $\mathcal{E}$  called **extractor** such that for any  $x$ , if  $(x, a, z; r)$  and  $(x, a, z'; r')$  are two accepting views for  $\mathcal{V}$  such that  $e \neq e'$  where  $e = \mathcal{V}(x, a; r)$  and  $e' = \mathcal{V}(x, a; r')$  then  $\mathcal{E}(x, a, e, z, e', z')$  yields  $w$  such that  $R(x, w)$
- **special HVZK:** there exists a polynomially bounded algorithm  $\mathcal{S}$  called **simulator** such that for any  $x \in L$  and  $e$ , the transcript  $(a, e, z)$  of the interaction  $\mathcal{P}(r_P) \xleftrightarrow{x} \mathcal{V}(r_V)$  conditioned to  $e$  has same distribution as  $\mathcal{S}(x, e; r)$ .

# $\Sigma$ Protocol



- $w$  s.t.  $R(x, w)$
- $E$  s.t.  $e \neq e'$ ,  $V(x, a, e, z)$ , and  $V(x, a, e', z')$  implies  $R(x, E(x, a, e, z, e', z'))$
- $S$  s.t. a honest  $(a, e, z)$  and  $S(x, e)$  generate the same distribution

# Specifying a $\Sigma$ Protocol

To fully define a  $\Sigma$ -protocol we need

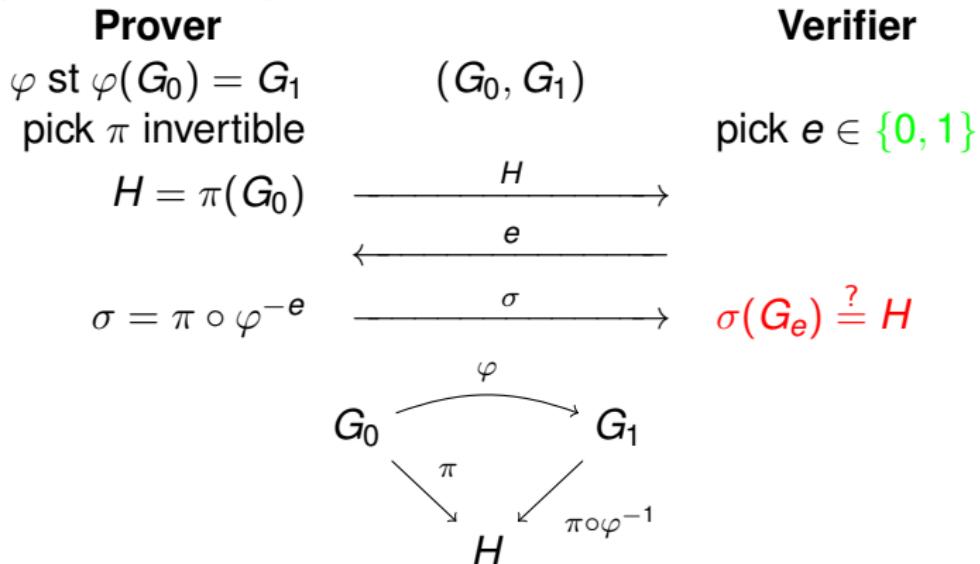
- a relation  $R$  defining the language
- a function for  $a = \mathcal{P}(x, w; r_P)$
- a samplable domain  $E$  for  $e$
- a function for  $z = \mathcal{P}(x, w, e; r_P)$
- a verification relation  $V(x, a, e, z)$
- a function  $\mathcal{E}(x, a, e, z, e', z')$
- a function  $\mathcal{S}(x, e; r)$

Properties to satisfy:

- ①  $R, \mathcal{P}, V, \mathcal{E}, \mathcal{S}$  and sampling are polynomially computable in  $|x|$
- ②  $\forall (x, w) \in R \forall r_P \forall e \in E \quad V(x, a, e, z)$
- ③  $\forall x \forall e, e' \in E \forall a, z, z'$   
 $(e \neq e', V(x, a, e, z), V(x, a, e', z')) \implies$   
 $R(x, \mathcal{E}(x, a, e, z, e', z'))$
- ④  $\forall (x, w) \in R \forall e \in E \quad \text{distrib}_{r_P}(a, e, z) = \text{distrib}_r(\mathcal{S}(x, e; r))$

# Example: Goldreich-Micali-Wigderson 1986

- relation  $R((G_0, G_1), \varphi)$ :  $\varphi$  invertible and  $\varphi(G_0) = G_1$
- $\mathcal{P}(G_0, G_1, \varphi; r_P)$ , **e domain**,  $\mathcal{P}(G_0, G_1, \varphi, e; r_P)$ ,
- $V(G_0, G_1, H, e, \sigma)$



- $\mathcal{E}(G_0, G_1, H, e, \sigma_e, e', \sigma_{e'}) = \sigma_1^{-1} \circ \sigma_0$  ( $e \neq e'$  so  $\{e, e'\} = \{0, 1\}$ )
- $\mathcal{S}(G_0, G_1, e; r)$ : pick  $\sigma$  invertible then set  $H = \sigma(G_e)$

# Graph Isomorphism

- a **graph** is a pair  $(V, E)$  such that  $E \subseteq V \times V$   
 $V$  is a set of **vertices**  
 $E$  is a set of **edges**  
an edge  $(u, v) \in E$  is said to go from vertex  $u$  to vertex  $v$
- given a permutation  $\pi$  of  $V$ ,  $\pi(V, E) = (V, F)$  where  $F$  is the set of all  $(\pi(u), \pi(v))$  for  $(u, v) \in E$
- the graphs  $G$  and  $G'$  on the same vertex set  $V$  are **isomorphic** if there exists a permutation  $\pi$  over  $V$  such that  $\pi(G) = G'$

# Check List

- ➊  $R, \mathcal{P}, V, \mathcal{E}, \mathcal{S}$  and sampling are polynomially computable in  $|G_0, G_1|$
- ➋ perfect completeness: quite clear
- ➌  $\forall(G_0, G_1, \varphi) \in R \forall H, \sigma_0, \sigma_1$   
 $(V(G_0, G_1, H, 0, \sigma_0), V(G_0, G_1, H, 1, \sigma_1)) \implies$   
 $R(G_0, G_1, \mathcal{E}(G_0, G_1, H, 0, \sigma_0, 1, \sigma_1))$
- ➍  $\forall(G_0, G_1, \varphi) \in R \forall e \text{ distrib}_{r_P}(H, e, \sigma) =$   
 $\text{distrib}(\mathcal{S}(G_0, G_1, e))$ 
  - since  $H$  is a function of  $G_0, G_1, e, \sigma$ , it is enough to show that  $\forall(G_0, G_1, \varphi) \in R \forall e, \sigma = \pi \circ \varphi^{-e}$  is uniformly distributed in  $S_V$  if  $\pi \in_U S_V$
  - this is true

## ZK Trick

An easy way to prove special HVZK:

- find a function  $f$  such that

$$\forall x, a, e, z \quad V(x, a, e, z) \implies a = f(x, e, z)$$

(works in most of  $\Sigma$ -protocols):

- show the correctness of  $\mathcal{S}$ :

$$\forall x, a, e, e', z, r \quad \mathcal{S}(x, e; r) = (a, e', z) \implies V(x, a, e, z) \wedge e = e'$$

- show that  $z$  is well distributed:

$$\forall x, e, z \Pr[\text{View}_{\mathcal{V}} \rightarrow z | x, e] = \Pr[\mathcal{S} \rightarrow z | x, e]$$

- conclude:

$$\forall x, a, e, z \quad \Pr[\text{View}_{\mathcal{V}} \rightarrow a, z | x, e] = \Pr[\mathcal{S} \rightarrow a, z | x, e]$$

# A Malicious Prover

$P^*$

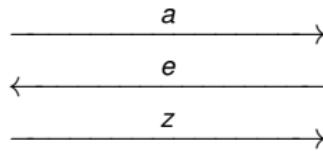
**Verifier**

$x$

pick  $e_{\text{guess}} \in_U E$

$\mathcal{S}(x, e_{\text{guess}}) \rightarrow (a, e_{\text{guess}}, z)$

if  $e \neq e_{\text{guess}}$ : fail!



pick  $e \in_U E$   
check  $V(x, a, e, z)$

$P^*$  succeeds with probability  $1/\#E$

# $\Sigma$ Protocols are Proof of Knowledge — i

## Theorem

A  $\Sigma$  protocol  $(\mathcal{P}, \mathcal{V})$  for  $L$  defined by  $R$  is an interactive proof of knowledge for  $L$ , with  $\beta$ -soundness for  $\beta = 1/\#E$ .

**Proof.** Termination and completeness come from the definition.

Extractor (sketch):

- pick  $r_P, r_V, r'_V$  and run  $\mathcal{P}^*(r_P) \xrightarrow{\text{select } e} \mathcal{V}(r_V)$  and  $\mathcal{P}^*(r_P) \xrightarrow{\text{select } e'} \mathcal{V}(r'_V)$
- we have same  $a$ , if  $e \neq e'$  and the two runs accept then  
 $\mathcal{E}(x, a, e, z, e', z')$  yields  $w$  with  $e = e'$  proba  $1/\#E$
- extraction works iff both runs accept and  $e \neq e'$
- problem: prove that it works within  $\mathcal{O}\left(\frac{1}{\varepsilon(x)-\beta}\right)$  attempts

## $\Sigma$ Protocols are Proof of Knowledge — ii

run 1

run 2

$\mathcal{P}^*(r_P) \xleftrightarrow{X} \mathcal{V}(r_V)$  makes  $(a, e, z)$   
and the acceptance bit  $b$

- $\Pr[b = 1] = \varepsilon(x)$
- $\Pr[b = 1|r_P] = \varepsilon(x, r_P)$
- $E(\varepsilon(x, r_P)) = \varepsilon(x)$

extraction works iff  $bb' = 1$ , and  $e \neq e'$

- if  $r_P$  is fixed,  $b$  resp  $b'$  only depends on  $r_V$  resp  $r'_V$
- if  $r_P$  is fixed,  $b$  and  $b'$  are iid
- $\Pr[bb' = 1|r_P] = \varepsilon(x, r_P)^2$
- $\Pr[bb' = 1, e \neq e'|r_P] = \Pr[bb' = 1|r_P] - \Pr[bb' = 1, e = e'|r_P]$
- $\Pr[bb' = 1, e = e'|r_P] = \Pr[b = 1, e = e'|r_P]$  and

$$\Pr[b = 1, e = e'|r_P] = \sum_{\substack{e \text{ st } b=1 \\ \text{from } r_P}} \Pr[\mathcal{V}(x, \cdot; r_V) = e]^2 = \varepsilon(x, r_P)\beta$$

$\mathcal{P}^*(r_P) \xleftrightarrow{X} \mathcal{V}(r'_V)$  makes  $(a, e', z')$   
and the acceptance bit  $b'$

- $\Pr[b' = 1] = \varepsilon(x)$
- $\Pr[b' = 1|r_P] = \varepsilon(x, r_P)$
- $E(\varepsilon(x, r_P)) = \varepsilon(x)$

## $\Sigma$ Protocols are Proof of Knowledge — iii

$$f(z) = z(z - \beta)$$

- $\Pr[bb' = 1, e = e'|r_P] = \varepsilon(x, r_P)\beta$
- $\Pr[bb' = 1, e \neq e'|r_P] = \varepsilon(x, r_P)(\varepsilon(x, r_P) - \beta) = f(z)$
- by applying the Jensen Inequality on  $Z = \varepsilon(x, r_P)$ , we obtain  $\Pr[bb' = 1, e \neq e'] \geq \varepsilon(x)(\varepsilon(x) - \beta)$   
(Jensen Inequality:  $E(f(Z)) \geq f(E(Z))$  if  $f$  is convex)
- the expected number of iterations until the extractor works is lower than the inverse of this
- for  $\varepsilon(x) > \beta$ , since  $\beta > 0$  is constant then  $1/\varepsilon(x) = \mathcal{O}(1)$   
so the extractor works with complexity  $\mathcal{O}\left(\frac{\text{Poly}(|x|)}{\varepsilon(x) - \beta}\right)$  □

# Parallel Composition

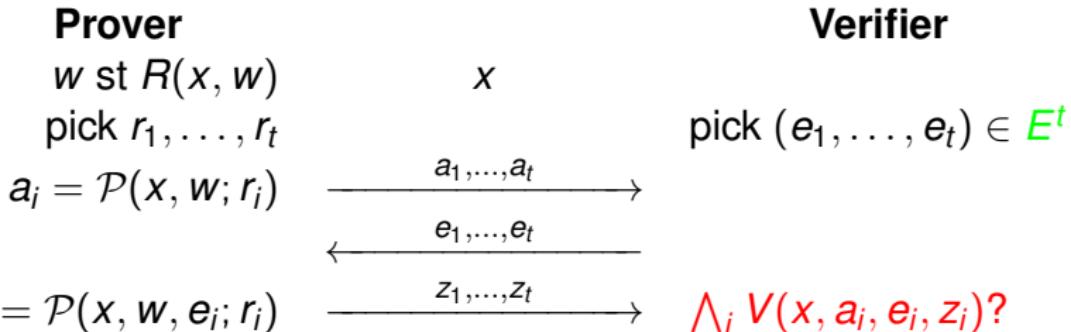
## Theorem

challenge



If  $(\mathcal{P}, \mathcal{V})$  is a  $\Sigma$ -protocol on set  $E$ , what follows is a  $\Sigma$ -protocol on  $E^t$ .

- $\mathcal{P}(n, v, s; r_P)$ , domain for  $e$ ,  $\mathcal{P}(n, v, s, e; r_P)$ ,  $V(n, v, x, e, y)$



- $\mathcal{E}(x, a, e, z, e', z')$ :

find  $i$  s.t.  $e_i \neq e'_i$  then do  $\mathcal{E}_{\text{old}}(x, a_i, e_i, z_i, e'_i, z'_i)$

- $\mathcal{S}(x, e_1, \dots, e_t; r_1, \dots, r_t)$ :

set  $(a_i, e_i, z_i) = \mathcal{S}_{\text{old}}(x, e_i; r_i)$

convenient to improve the soundness threshold!

# Honest Verifier Zero-Knowledge

## Definition (HVZK)

An interactive proof system  $(\mathcal{P}, \mathcal{V})$  is **\*-honest verifier zero-knowledge** if there exists a ppt algorithm  $\mathcal{S}$  such that

$$\text{View}_{\mathcal{V}} \left( \mathcal{P}(r_{\mathcal{P}}) \xleftrightarrow{x} \mathcal{V}(r_{\mathcal{V}}) \right)$$

and  $\mathcal{S}(x, r)$  produce **\*-identical distributions**.

“ZK only when the verifier is honest”

# $\Sigma$ Protocols are HVZK

## Theorem

A  $\Sigma$  protocol  $(\mathcal{P}, \mathcal{V})$  for  $L$  defined by  $R$  is honest verifier zero-knowledge.

Not true for a malicious  $\mathcal{V}^*$ : he could select  $e = H(x, a)$  and extract some unforgeable evidence  $z$

**Proof.** Since  $e$  only depends on  $r_V$  then the simulator can run  $\mathcal{V}(x, a_0; r_V)$  on an arbitrary  $a_0$  to get  $e$ .

- compute  $e = \mathcal{V}(x, a_0; r_V)$   
note:  $e$  has identical distribution as in the correct protocol run
- pick  $r$ , compute  $(a, e, z) = \mathcal{S}(x, e; r)$
- yield  $(x, a, z; r_V)$

Because of special HVZK the view has identical distribution □

# Zero-Knowledge on Small Challenge Set

## Theorem

A  $\Sigma$ -protocol with a challenge set  $E$  with polynomially bounded size is zero-knowledge.

**Proof.** Simulator:

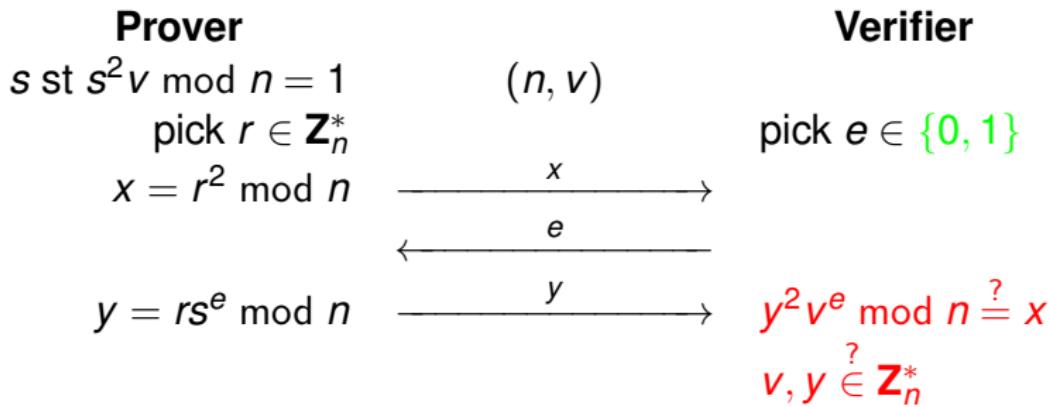
- 1: pick some random coins  $\rho$  to set up the verifier
- 2: pick  $e_{\text{guess}} \in E$  ▷ a guess for  $e$
- 3: run  $S(x, e_{\text{guess}}) \rightarrow (a, e_{\text{guess}}, z)$
- 4: get  $e = \mathcal{V}^*(a; \rho)$
- 5: if  $e \neq e_{\text{guess}}$ : try again ▷ this trial failed
- 6: output  $(a, z; \rho)$

- $(a, e_{\text{guess}}, z)$  has same distribution as  $(a, e, z)$
- so  $a$  is independent from  $e_{\text{guess}}$   $e = \mathcal{V}^*(a; \rho)$
- so  $\Pr[e_{\text{guess}} = \mathcal{V}^*(a; \rho)] = 1/\#E$
- so, it terminates with expected polynomial time
- furthermore,  $(a, z)$  has same distribution as for the honest prover

So, we perfectly simulate the protocol. □

## Example: Fiat-Shamir 1986

- relation  $R((n, v), s)$ :  $s^2 v \bmod n = 1$ ,  $v, s \in \mathbf{Z}_n^*$
- $\mathcal{P}(n, v, s; r_P)$ , domain for  $e$ ,  $\mathcal{P}(n, v, s, e; r_P)$ ,  $V(n, v, x, e, y)$



- $\mathcal{E}(n, v, x, e, y_e, e', y_{e'}) = y_1/y_0 \bmod n$  ( $e \neq e'$  so  $\{e, e'\} = \{0, 1\}$ )

$$\left(\frac{y_1}{y_0}\right)^2 v \equiv \frac{y_1^2 v^1}{y_0^2 v^0} \equiv \frac{x}{x} \equiv 1 \pmod{n}$$

- $\mathcal{S}(n, v, e; r)$ : pick  $y \in \mathbf{Z}_n^*$  then set  $x = y^2 v^e \bmod n$

# Check List

- ➊  $R, \mathcal{P}, V, \mathcal{E}, \mathcal{S}$  and sampling are polynomially computable in |instance|
- ➋ perfect completeness: quite clear
- ➌ special soundness: in previous slide
- ➍  $\forall(\text{instance}, s) \in R \quad \forall e \quad \text{distrib}_{r_P}(x, e, y) = \text{distrib}(\mathcal{S}(\text{instance}, e))$ 
  - since  $x$  is a function of instance,  $e$ ,  $y$ , it is enough to show that  $\forall(\text{instance}, s) \in R \quad \forall e, y = rs^e \bmod n$  is uniformly distributed in  $\mathbf{Z}_n^*$  if  $r \in_U \mathbf{Z}_n^*$
  - this is true since  $(\text{instance}, s) \in R$  implies that  $s \in \mathbf{Z}_n^*$

# Example: Schnorr 1989

- relation  $R((G, q, g, y), x)$ :  $g^x = y$  in  $G$ ,  $q$  prime  $> 2^t$ ,  $g$  of order  $q$
- $\mathcal{P}(G, q, g, y, x; r_P)$ , domain for  $e$ ,  $\mathcal{P}(G, q, g, y, x, e; r_P)$ ,  
 $V(G, q, g, y, r, e, s)$

Prover

$$\begin{array}{l} x \text{ st } g^x = y \\ \text{pick } k \in \mathbf{Z}_q \\ r = g^k \\ s = ex + k \text{ mod } q \end{array} \quad \begin{array}{c} (G, q, g, y) \\ \xrightarrow{r} \\ \xleftarrow{e} \\ \xrightarrow{s} \end{array}$$

Verifier

$$\begin{array}{l} \text{pick } e \in \{1, \dots, 2^t\} \\ q \text{ prime } > 2^t \\ g \text{ of order } q? , y^q \stackrel{?}{=} 1 \\ ry^e \stackrel{?}{=} g^s \end{array}$$

- $\mathcal{E}(G, q, g, y, r, e, s, e', s') = \frac{s-s'}{e-e'} \bmod q$  we have  
 $\gcd(e - e', q) = 1$  and  $g, y$  of order  $q$

$$g^{s-s'} = \frac{ry^e}{ry^{e'}} = y^{e-e'}$$

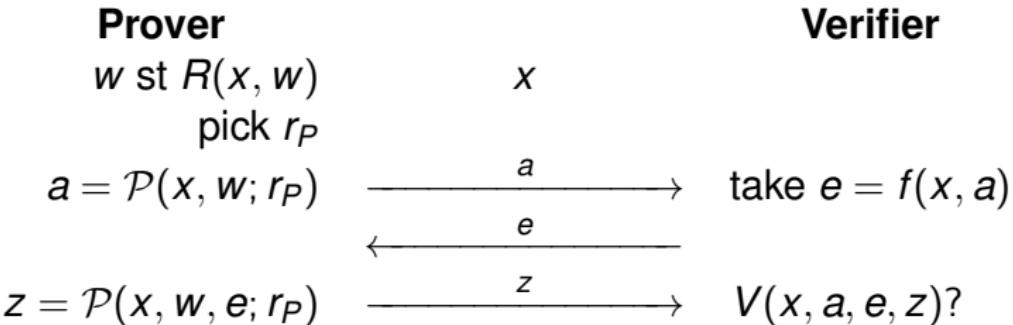
we extract  $(e - e')$ -th roots and get  $g^{\frac{s-s'}{e-e'}} = y$

- $\mathcal{S}(G, q, g, y, e; r)$ : pick  $s \in \mathbf{Z}_q$  then set  $r = g^s y^{-e}$

# Check List

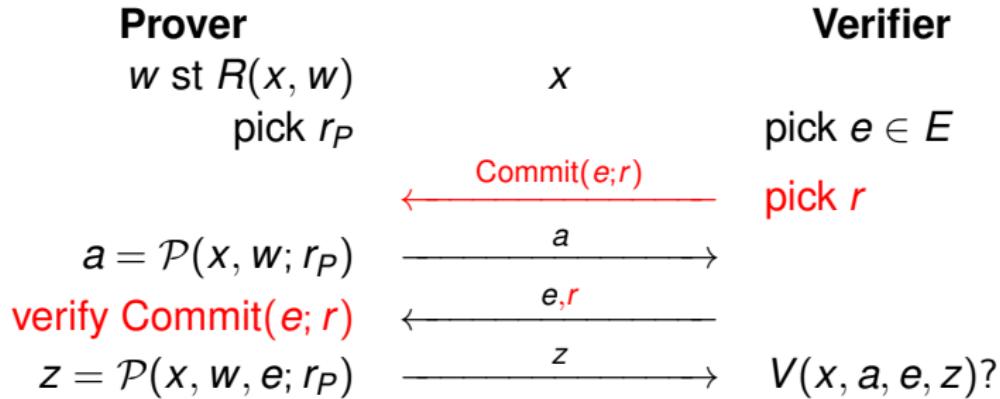
- ➊  $R, \mathcal{P}, V, \mathcal{E}, \mathcal{S}$  and sampling are polynomially computable in |instance|
- ➋ perfect completeness: quite clear
- ➌ special soundness: in previous slide
- ➍  $\forall(\text{instance}, x) \in R \quad \forall e \quad \text{distrib}_{r_P}(r, e, s) = \text{distrib}(\mathcal{S}(\text{instance}, e))$ 
  - since  $r$  is a function of instance and  $e, s$ , it is enough to show that  $\forall(\text{instance}, x) \in R \quad \forall e, s = ex + k \bmod q$  is uniformly distributed in  $\mathbf{Z}_q$  if  $k \in_U \mathbf{Z}_q$
  - this is true

# Problem with a Malicious Verifier



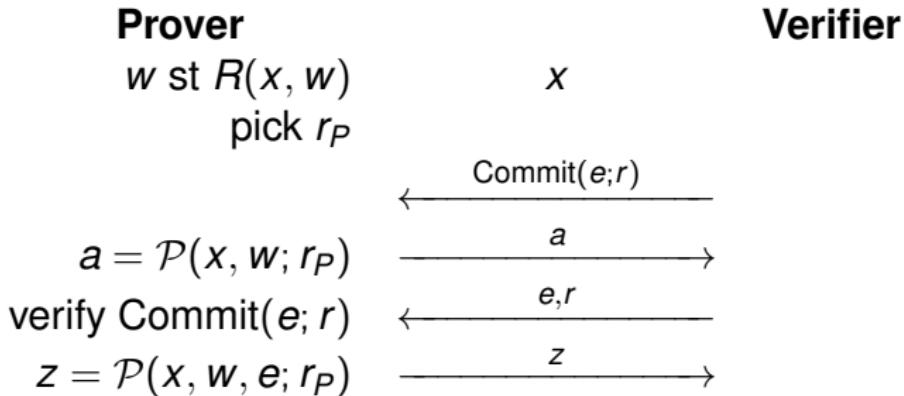
- at the end, get  $(a, z)$  such that  $V(x, a, f(x, a), z)$
- can be easily simulated if  $E$  is small but not when it is large
- not ZK if  $E$  is large

# Strengthening by Commitment



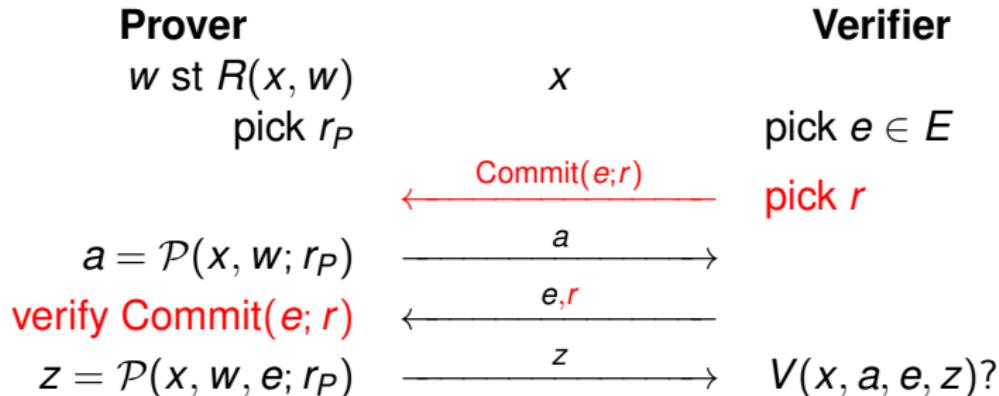
- now  $*$ -ZK if commitment is  $*$ -binding (next slide)
- caveat: problem with soundness...

# Proving the ZK Property



- run  $\mathcal{V}^*$  once on  $r_V$  and send him a dummy  $a_0$  to make him open the commitment
  - given  $e$ , generate a  $(a, e, z)$  triplet
  - rewind  $\mathcal{V}^*$  and run it again on  $r_V$  with  $a$
  - if  $e$  has changed, break the commitment!
  - otherwise, yield the final view  $(x, a, z; r_V)$

# Strengthening by Commitment — Caveat



- \*-ZK
- problem: extractor no longer works  
(our extractor would break the binding property)
- could work using a trapdoor to break the binding property

# Pedersen Commitment 1991

- **setup:** generate two large primes  $p$  and  $q$  s.t.  $q|(p - 1)$ , an element  $g \in \mathbb{Z}_p^*$  of order  $q$ ,  $\tau \in \mathbb{Z}_q^*$ , and  $h = g^\tau \pmod{p}$   
**Domain parameters:**  $\langle p, q, g, h \rangle$
- **commit:**  $\text{Commit}(X; r) = g^X h^r \pmod{p}$
- **unconditionally hiding:** given  $c$  in the subgroup spanned by  $g$ , any  $X$  has a related  $r$  such that  $\text{Commit}(X; r) = c$
- **computationally binding:** committing to  $X$  and opening to  $X' \neq X$  leads to solving  $g^X h^r \equiv g^{X'} h^{r'} \pmod{p}$  hence  
 $\tau = \frac{X' - X}{r - r'} \pmod{q}$  breaking binding property has hard as break it's  $\infty$   
This is equivalent to solving the discrete logarithm problem with the domain parameters
- **trapdoor:** using  $\tau$  we can open a commitment  $\text{Commit}(X_0; r_0)$  on an arbitrary  $X$ :

$$\text{Commit}\left(X; r_0 + \frac{X_0 - X}{\tau}\right) = g^X h^{r_0 + \frac{X_0 - X}{\tau}} = g^{X_0} h^{r_0} = \text{Commit}(X_0; r_0)$$

## Pedersen Commitment — i

$$h = g^r \bmod p \quad \text{Commit}(X; r) = g^X h^r \bmod p$$

- **unconditionally hiding:** given  $c$  in the subgroup spanned by  $g$ , any  $X$  has a related  $r$  such that  $\text{Commit}(X; r) = c$

$$\forall x \quad \Pr[g^X h^r = c | X = x] = \frac{1}{q}$$

so  $c$  and  $X$  are statistically independent (perfect secrecy)

# Pedersen Commitment — ii

$$h = g^r \bmod p \quad \text{Commit}(X; r) = g^X h^r \bmod p$$

- **computationally binding:** committing to  $X$  and opening to  $X' \neq X$  leads to solving  $g^X h^r \equiv g^{X'} h^{r'} \pmod{p}$  hence  
 $\tau = \frac{X' - X}{r - r'} \bmod q$

This is equivalent to solving the discrete logarithm problem with the domain parameters

Game Bind

- 1:  $\text{Setup}(1^s) \xrightarrow{\$} (p, q, g, h)$
- 2:  $\mathcal{A}(p, q, g, h) \xrightarrow{\$} (x, r, x', r')$
- 3: **return**  
 $1_{x \neq x', \text{Commit}(x; r) = \text{Commit}(x'; r')}$

Game DL

- 1:  $\text{Setup}(1^s) \xrightarrow{\$} (p, q, g)$
- 2: pick  $\tau \in \mathbb{Z}_q$ ,  $h \leftarrow g^\tau$
- 3:  $\mathcal{B}(p, q, g, h) \xrightarrow{\$} z$
- 4: **return**  $1_{h=g^z}$

$\mathcal{B}(p, q, g, h)$ :

- 5:  $\mathcal{A}(p, q, g, h) \xrightarrow{\$} (x, r, x', r')$
- 6:  $z \leftarrow \frac{x' - x}{r - r'} \bmod q$
- 7: **return**  $z$

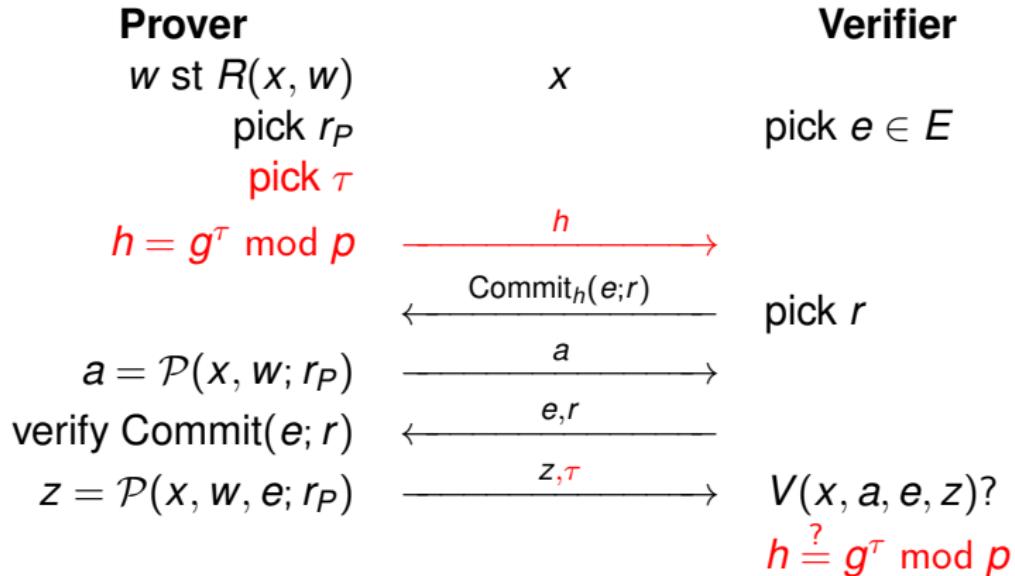
## Pedersen Commitment — iii

$$h = g^r \bmod p \quad \text{Commit}(X; r) = g^X h^r \bmod p$$

- **trapdoor:** using  $\tau$  we can open a commitment  $\text{Commit}(X_0; r_0)$  on an arbitrary  $X$ :

$$\text{Commit}\left(X; r_0 + \frac{X_0 - X}{\tau}\right) = g^X h^{r_0 + \frac{X_0 - X}{\tau}} = g^{X_0} h^{r_0} = \text{Commit}(X_0; r_0)$$

# Strengthened Protocol



- now computational ZK (based on the hardness of discrete logarithm)
- sound

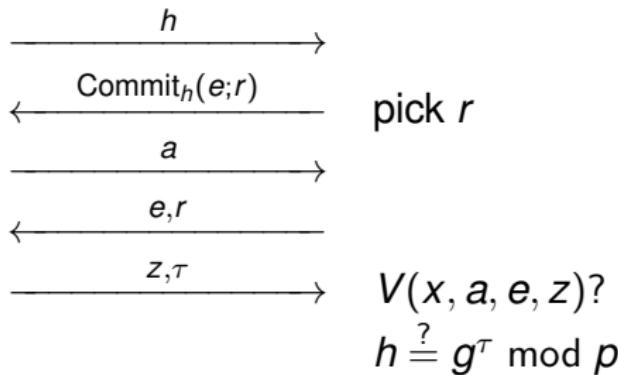
# Proving Soundness

Prover

$x$

Verifier

pick  $e \in E$



- run  $\mathcal{P}^*$  once on  $r_P$  and simulate  $\mathcal{V}$  with commitment  $c$
- get  $\tau$  at the end
- rewind  $\mathcal{P}^*$  and run it again on  $r_P$  with same  $c$
- open  $c$  on  $e'$  (using  $\tau$ ) as in the standard extraction

# Summary about Composition

|                     | parallel composition | sequential composition |
|---------------------|----------------------|------------------------|
| proof systems       | it works             | it works               |
| $\Sigma$ -protocols | it works             | it does not work       |
| ZK proofs           | it may not work      | it works               |

- proof systems: soundness amplifies well
- $\Sigma$ -protocols: a sequential repetition is no longer a  $\Sigma$ -protocol
- ZK proofs: counterexample  
 $\Sigma$ -protocols are fully ZK if  $E$  is small (polynomial size) but not fully ZK if  $E$  is large!

## 1 The Cryptographic Zoo

## 2 Cryptographic Security Models

## 3 Cryptanalysis (Public-Key)

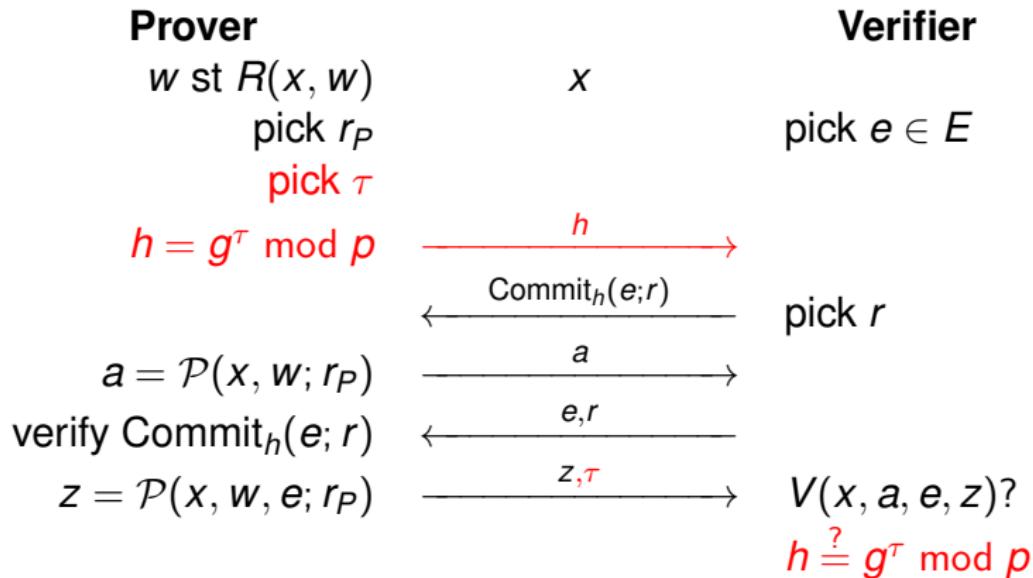
## 4 The Power of Interaction

- Interactive Proofs
- Zero-Knowledge
- Zero-Knowledge Construction from  $\Sigma$  Protocol
- **Setup Models**
- A Building Block for Making Cryptographic Primitives

## 5 Cryptanalysis (Conventional)

## 6 Proving Security

# ZK from $\Sigma$ -Protocol Again



- need for ephemeral key  $h$  looks artificial
- it only helps proving soundness
- it costs an extra round and more computation
- why not using a fixed one?

# Common Reference String Model

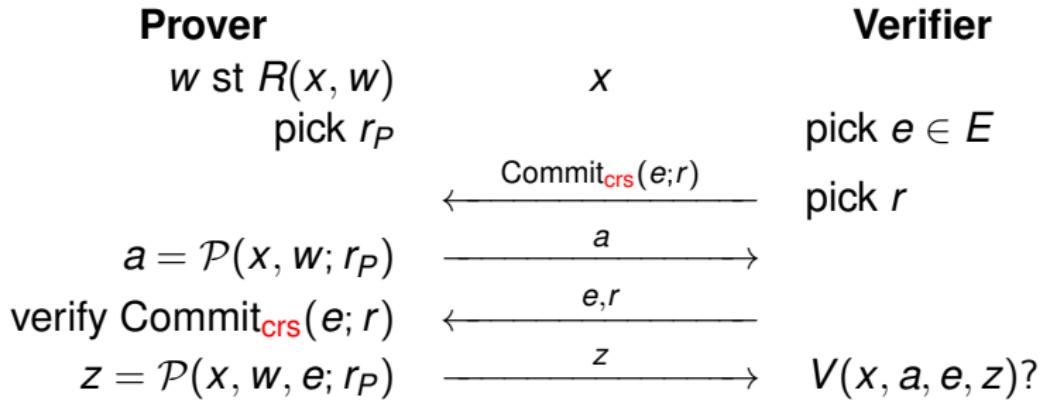
## CRS in Theory

- **setup**: generate a pair  $(\text{crs}, \tau)$  made of a common reference string  $\text{crs}$  and a trapdoor  $\tau$
- add  $\text{crs}$  as an input to  $\mathcal{P}, \mathcal{V}$
- add  $\tau$  as an input to the extractor and the simulator

## CRS in Practice

- use a generator for crs only and a seed to convince that it has been generated without being able to compute the trapdoor

# ZK from $\Sigma$ -Protocol Again

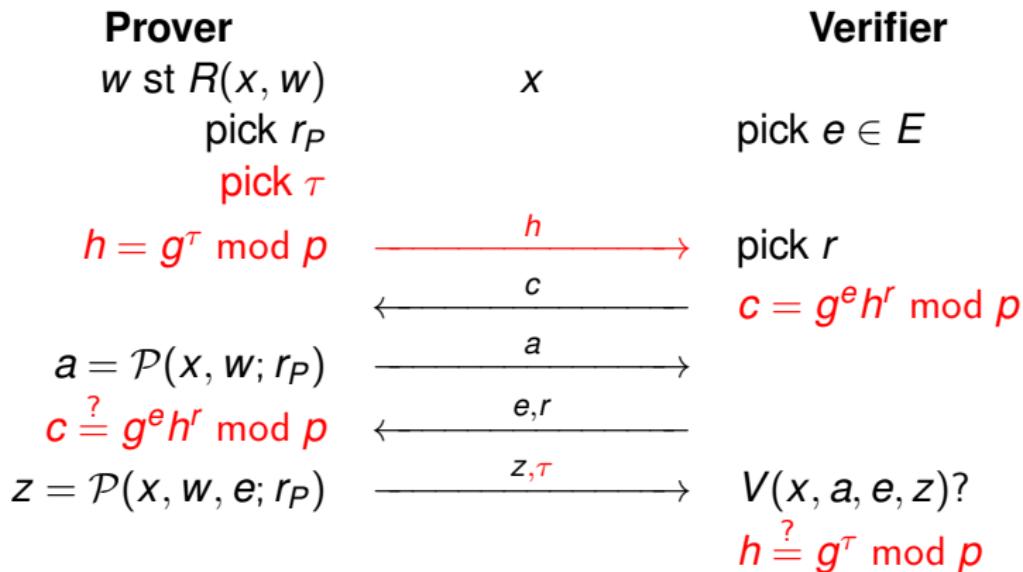


# Deniability Issue

- regular ZK proofs are inherently **deniable**: verifier cannot later prove to a third party that the proof was run  
→ the prover can claim that he did not run the proof
- ZK in the CRS model may lose this property: when the trapdoor is necessary for the simulation, the view from the verifier is unforgeable for the verifier, so he can convince a third party that the proof was run (or that the trapdoor was used)

BUT: still no leakage on the witness

# ZK from $\Sigma$ -Protocol Again



- need for Pedersen commitment looks artificial
- it only helps proving security
- it costs more computation
- why not using  $c = H(e \| r)$ ?

# Random Oracle $H$ (Lazy Sampling)

- setup: erase a table  $\mathcal{T}$
- query  $H(u)$ :
  - look for some  $(u, v)$  pair in  $\mathcal{T}$
  - if there is none, pick  $v$  at random and insert  $(u, v)$  in  $\mathcal{T}$
  - answer by  $v$
- a participant cannot predict the value of  $H(u)$  before a query  $u$  is made

# Random Oracle Model

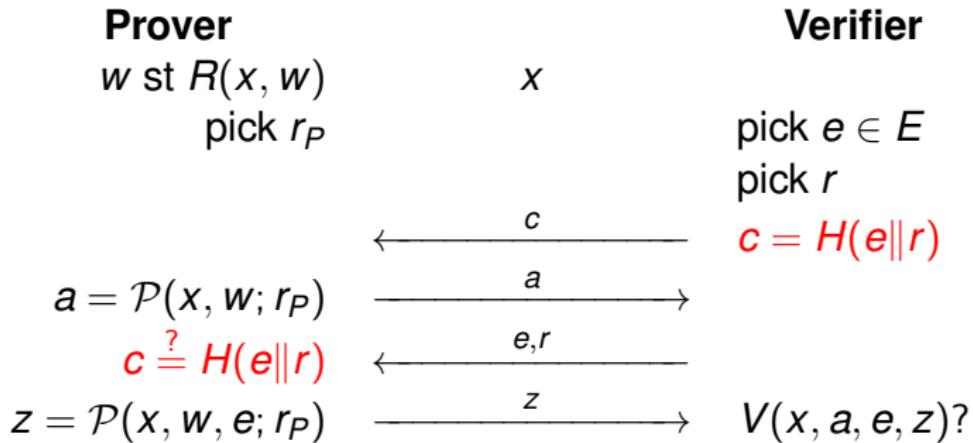
## ROM in Theory

- **setup**: setup the random oracle  $H$
- provide  $\mathcal{P}$  and  $\mathcal{V}$  access to oracle  $H$
- allow the extractor to simulate oracle  $H$  for  $\mathcal{P}^*$
- allow the simulator to simulate oracle  $H$  for  $\mathcal{V}^*$

## ROM in Practice

- use hash function at the place of  $H$

## Example



- soundness: keep same  $c$  in both runs but cheat on the input  $e$
- ZK: from the query by  $\mathcal{V}^*$  to  $H$  see if  $c$  is the commitment of some  $e$  (if not  $\mathcal{V}^*$  is unlikely to be able to open it)

## Other Setup Models

- common reference string
- random oracle
- public key setup for all participants
- key registration to a public directory
- secure token, trusted agent

## 1 The Cryptographic Zoo

## 2 Cryptographic Security Models

## 3 Cryptanalysis (Public-Key)

## 4 The Power of Interaction

- Interactive Proofs
- Zero-Knowledge
- Zero-Knowledge Construction from  $\Sigma$  Protocol
- Setup Models
- A Building Block for Making Cryptographic Primitives

## 5 Cryptanalysis (Conventional)

## 6 Proving Security

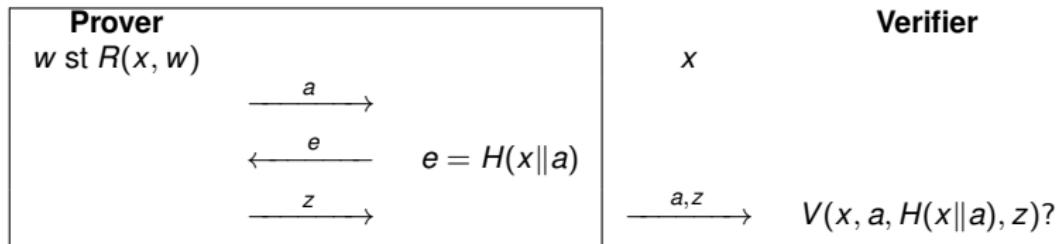
# Constructions

- non-interactive zero-knowledge proofs (NIZK) from  $\Sigma$ -protocol
- signature from  $\Sigma$ -protocol
- trapdoor commitment from  $\Sigma$ -protocol
- hash function from  $\Sigma$ -protocol (in exercise)

# The Fiat-Shamir Paradigm 1986

## NIZK from $\Sigma$ -Protocol

(making the  $\Sigma$  verifier malicious by selecting  $e$  adaptively...)



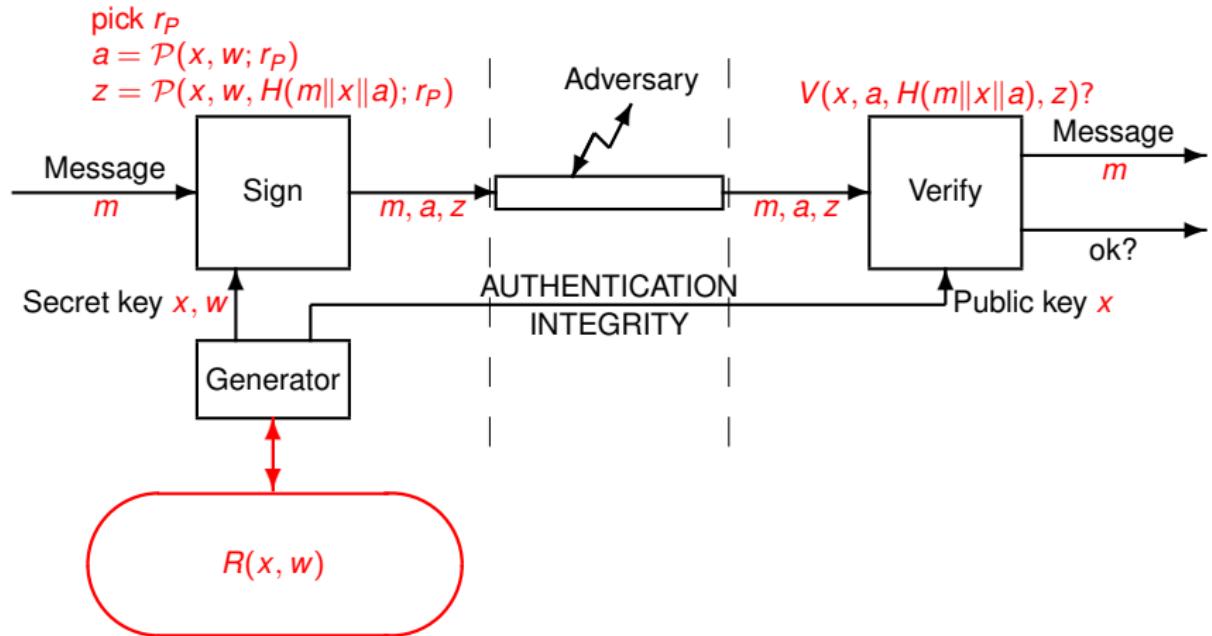
- replace  $\mathcal{V}(x, a; r_V)$  by a random oracle  $H(x\|a)$
- the final view is a non-interactive proof
- a random oracle “looks like” a honest verifier  
→ “kind of” zero-knowledge
- a simulator for the final view becomes a cheating prover  
→ cannot be zero-knowledge and sound at the same time

# Fiat-Shamir Signature

## Signature from $\Sigma$ -Protocol

- public key:  $x$
- secret key:  $(x, w)$
- signature:
  - ➊ pick  $r_P$ , set  $a = \mathcal{P}(x, w; r_p)$
  - ➋ replace  $\mathcal{V}(x, a; r_V)$  by  $e = H(\text{message} \| x \| a)$
  - ➌ set  $z = \mathcal{P}(x, w, e; r_P)$
  - ➍ signature is the pair  $(a, z)$
- verification:  $V(x, a, H(\text{message} \| x \| a), z)$

# Fiat-Shamir Signature



# The Security of a Fiat-Shamir Signature in ROM

NP problem

## Theorem



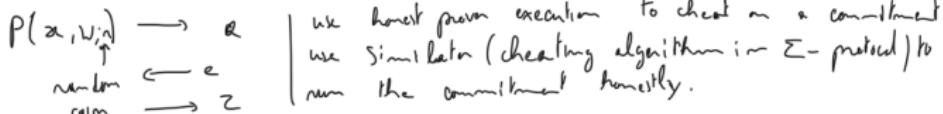
*Given a relation  $R$  s.t. it is hard to find witnesses and a  $\Sigma$ -protocol for its language, the signature scheme obtained by the Fiat-Shamir construction using a random oracle is EF-CMA-secure (existentially unforgeable under chosen message attacks).*

(to be seen in next chapter)

# Trapdoor Commitment

Commitment on  $e \in E$

- take a  $\Sigma$ -protocol  
assume that finding a witness is hard
- take a parameter (crs)  $x \in L$  such that nobody knows a witness  $w$  s.t.  $(x, w) \in R$
- to commit to  $e \in E$ , pick  $r$  and let  $(a, e, z) = \mathcal{S}(x, e; r)$   
set  $a$  to the commit value  
set  $(e, z)$  to the opening value
- for opening: check  $V(x, a, e, z)$  holds
- perfectly hiding (distribution of  $a$  independent from  $e$ )
- computationally binding (otherwise extract  $w$ ) (hard to find a witness)
- trapdoor  $w$ : prover can make  $a$  and open it on any  $e$



# Conclusion

- interaction opens new computational powers (PSPACE instead of P)
- zero-knowledge proof feasible
- nice building block for cryptographic primitives
- theoretical crypto foundation

## References — i

- **Goldwasser-Micali-Rackoff.**  
The Knowledge Complexity of Interactive Proof Systems.  
SIAM Journal of Computing, 18(1), 1989.
- **Goldreich-Micali-Wigderson.**  
Proofs that Yield Nothing But their Validity and a  
Methodology of Cryptographic Protocol Design.  
In *FOCS 1986*, IEEE.
- **Goldreich-Micali-Wigderson.**  
How to Prove all NP Statements in Zero-Knowledge and a  
Methodology of Cryptographic Protocol Design  
In *CRYPTO 1986*, LNCS 218.
- **Fiat-Shamir.**  
How to Prove Yourself: Practical Solutions to Identification  
and Signature Problems.  
In *CRYPTO 1986*, LNCS 218.

## References — ii

- **Schnorr.**

Efficient Identification and Signatures for Smart Cards.  
In *EUROCRYPT 1989*, LNCS 434.  
In *CRYPTO 1989*, LNCS 435.

- **Pass.**

On Deniability in the Common Reference String and  
Random Oracle Model.  
In *CRYPTO 2003*, LNCS 2729.

- **Damgård.**

On the Existence of Bit Commitment Schemes and  
Zero-Knowledge Proofs.  
In *CRYPTO 1989*, LNCS 435.

# Train Yourself

- $\Sigma$ -protocol:
  - final exam 2008–09 ex2 (cubic residues)
  - final exam 2009–10 ex1 (chameleon hash)
  - final exam 2010–11 ex1
  - final exam 2015–16 ex1 (in a group of exponent 2)
- OR proof: final exam 2010–11 ex1
- GQ: final exam 2008–09 ex3
- setup: final exam 2009–10 ex3
- ZKPoK and composition:
  - final exam 2013–14 ex1 (security interference)
  - final exam 2016–17 ex2 (ZKPoK from  $\Sigma$ )
- Fiat-Shamir: final exam 2014–15 ex2
- Unruh transform: final exam 2017–18 ex3







- 1 The Cryptographic Zoo
- 2 Cryptographic Security Models
- 3 Cryptanalysis (Public-Key)
- 4 The Power of Interaction
- 5 Cryptanalysis (Conventional)
- 6 Proving Security

## 1 The Cryptographic Zoo

## 2 Cryptographic Security Models

## 3 Cryptanalysis (Public-Key)

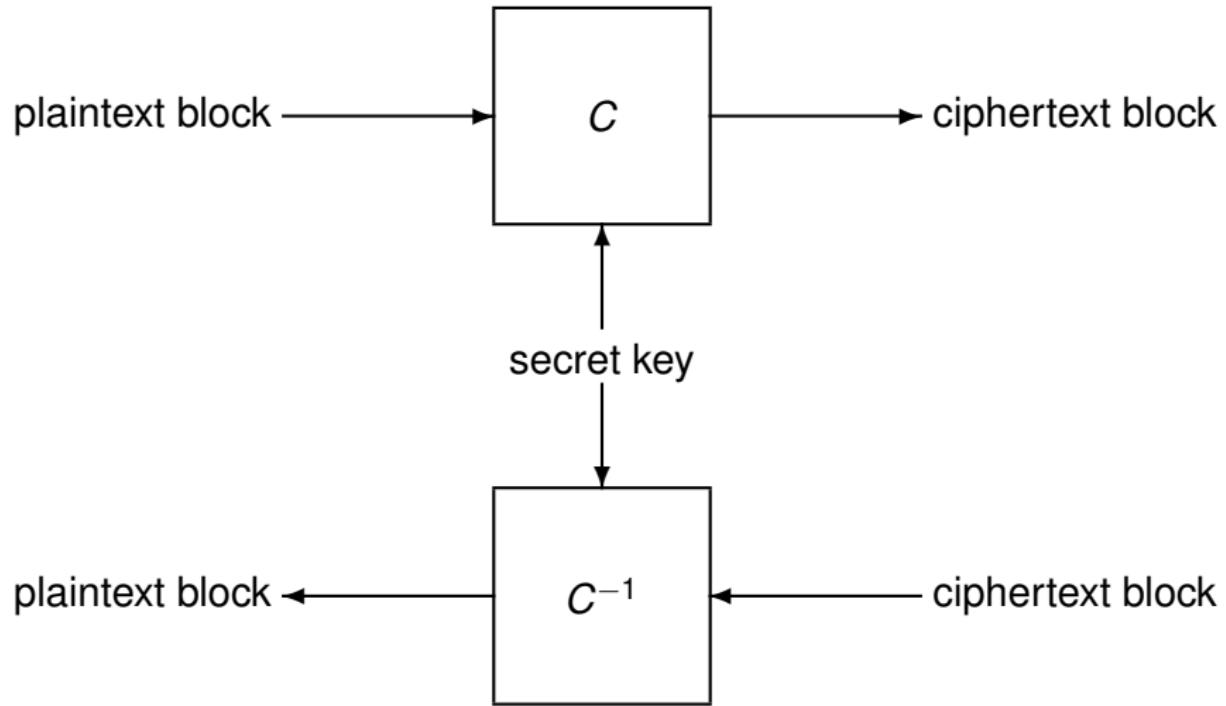
## 4 The Power of Interaction

## 5 Cryptanalysis (Conventional)

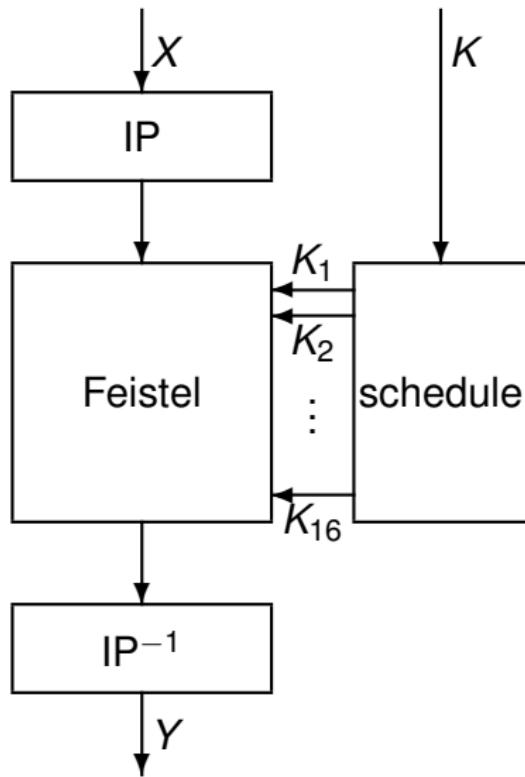
- Block Ciphers
- Differential Cryptanalysis
- Linear Cryptanalysis
- Hypothesis Testing in Cryptography
- Decorrelation

## 6 Proving Security

# Block Cipher

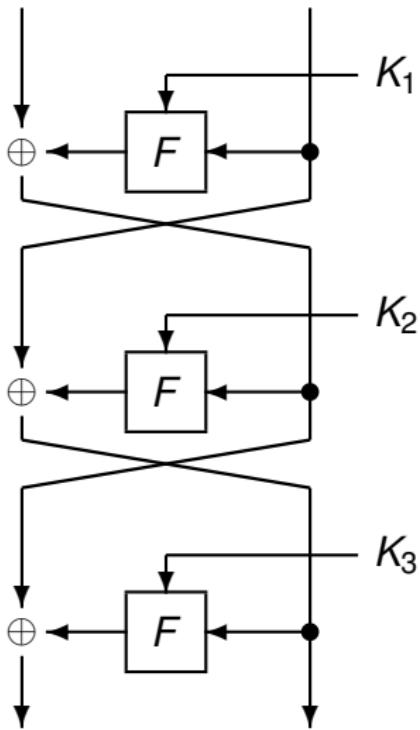


## Example: DES



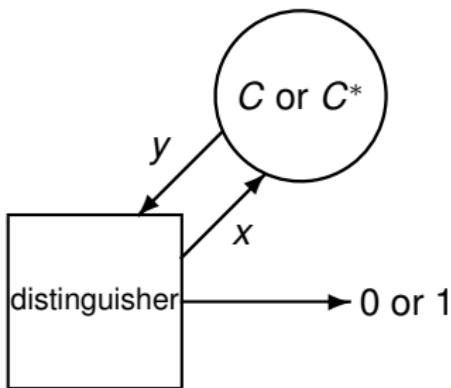
# DES — Feistel Scheme

$$\Psi(F^{K_1}, F^{K_2}, F^{K_3})$$



# Distinguishing Attack on Ciphers

Indistinguishability from an ideal scheme is another security model



- $C$ : permutation (block cipher) defined by a random key
- $C^*$ : uniformly distributed random permutation (ideal scheme)
- Advantage:  $\Pr[\text{output} = 1 | C] - \Pr[\text{output} = 1 | C^*]$

# Perfect Cipher

## Definition

Given a message block space  $\{0, 1\}^\ell$ , a key  $K$  is a uniformly distributed integer between 1 and  $2^\ell!$ , and  $C_K$  is the  $K$ th permutation of  $\{0, 1\}^\ell$ .

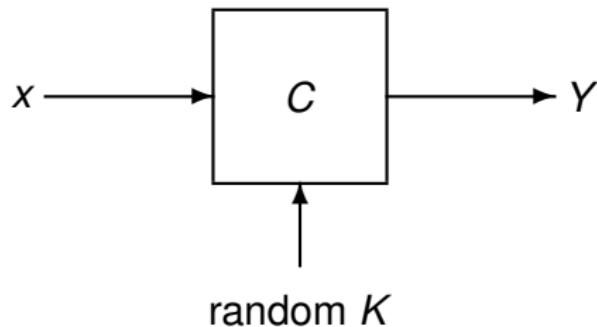
Reminder (Stirling Formula)

$$n! \sim \sqrt{2\pi n} n^n e^{-n}$$

Number of bits to represent  $K$ :  $\log_2(2^\ell!) \approx \ell 2^\ell \dots$

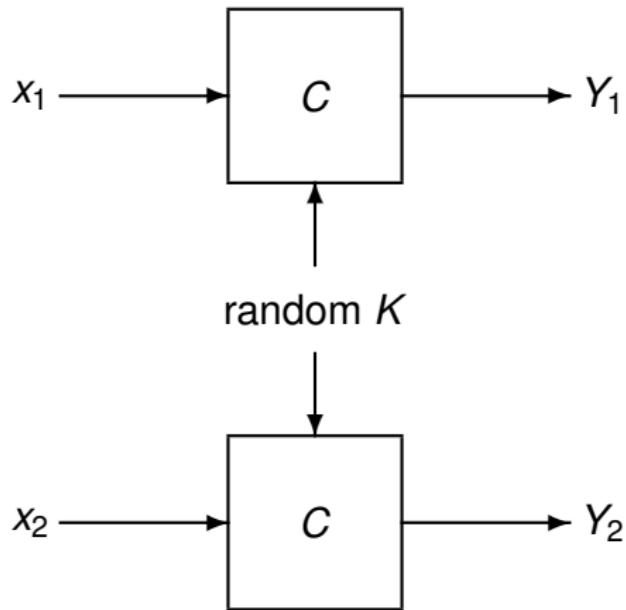
Example for  $\ell = 64$ :  $\log_2(2^\ell!) \approx 1\,180\,591\,620\,717\,411\,303\,424$  bits  
(1 048 576 Petabytes)

# Uniform Distribution



For any  $x$ , the random variables  $Y$  is uniformly distributed.

# Pairwise Independence



If  $x_1 \neq x_2$ ,  $Y_1$  and  $Y_2$  are “nearly” independent.

$$\Pr[Y_1 = y_1 \text{ and } Y_2 = y_2] = \frac{1}{2^\ell(2^\ell - 1)} \text{ for } y_1 \neq y_2$$

# *n*-wise Independence

## Definition

For any pairwise different  $x_1, \dots, x_n$ , the random variables  $Y_i = C_K(x_i)$ ,  $i = 1, \dots, n$ , defined by a random  $K$  are uniform and *nearly* independent if for any  $y_1, \dots, y_n$  of pairwise different values we have

$$\Pr[Y_1 = y_1, \dots, Y_n = y_n] = \frac{1}{2^\ell(2^\ell - 1) \cdots (2^\ell - n + 1)}$$

- The perfect cipher is nearly *n*-wise independent
- The view from the adversary is  $((x_1, y_1), \dots, (x_n, y_n))$
- Real ciphers should look like *n*-wise independent so that adversaries limited to *n* samples get no information

# Goal of Block Cipher Designs

## Goal

A block cipher which is used  $n$  times should have a behavior which is hard to distinguish from the behavior of the perfect cipher

→ uniform distribution and almost  $n$ -wise independence

## 1 The Cryptographic Zoo

## 2 Cryptographic Security Models

## 3 Cryptanalysis (Public-Key)

## 4 The Power of Interaction

## 5 Cryptanalysis (Conventional)

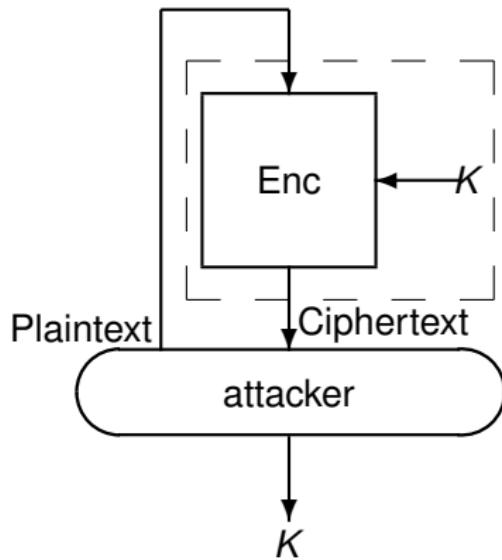
- Block Ciphers
- Differential Cryptanalysis
- Linear Cryptanalysis
- Hypothesis Testing in Cryptography
- Decorrelation

## 6 Proving Security

# History

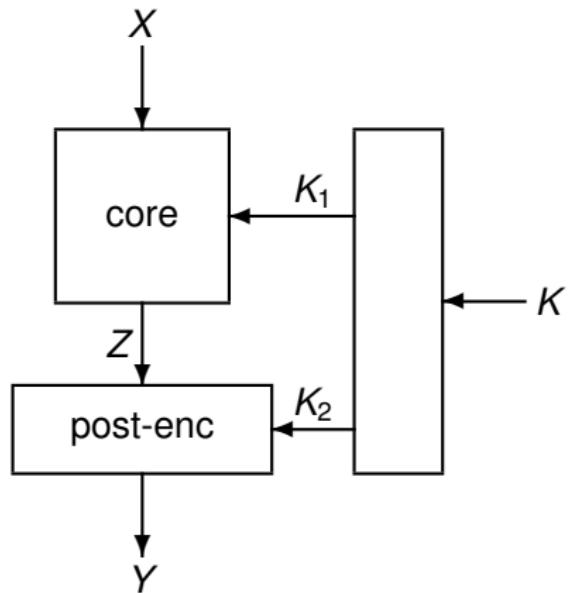
- invented by Eli Biham and Adi Shamir (Biham's PhD Thesis)
- 1990: broke DES-like ciphers
- 1992: theoretical attack against DES requiring  $2^{47}$  chosen plaintexts (not realistic enough to be called an attack)
- 1993: breakdown if the design of DES is slightly modified
- 1994: Coppersmith claimed that DES was designed to optimally resist it

# Chosen Plaintext Key Recovery Attack



## Step 1: Cipher Decomposition

find an appropriate decomposition of following form



## Step 2: Deviant Property

find a deviant property of following form

$$\Pr [\Delta Z = b | \Delta X = a] \text{ large}$$

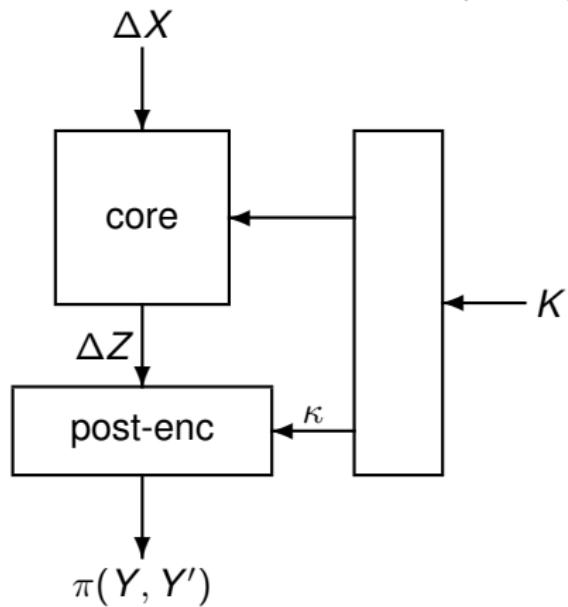
Difference:  $\Delta Z = Z' \oplus Z$

- trick: look at difference propagation
- use heuristic approximations

## Step 3: Differential Computation

isolate (little) information about  $Y, Y', K_2$  to filter out pairs s.t.  
 $\Delta Z \neq b$

we use a predicate  $R(\kappa, \pi(Y, Y'))$  (assuming  $\Delta Z = b$ ):  
“ $\Delta Z$  is consistent with  $\kappa$  and  $\pi(Y, Y')$ ”



## Step 4: Implementation

### Precomputation:

- 1: initialize  $\text{SubCandidate}_u$  to empty set for all  $u$
- 2: for all  $u$  and all  $\kappa$  such that  $R(\kappa, u)$ , insert  $\kappa$  in  $\text{SubCandidate}_u$

### Collection phase:

- 3: collect  $n$  pairs  $((x, y), (x \oplus a, y'))$  of plaintext-ciphertext pairs

### Analysis phase:

- 1: initialize counters  $m_\kappa$  to 0
- 2: **for** each pair **do**
- 3:   compute  $u = \pi(y, y')$
- 4:   for all  $\kappa \in \text{SubCandidate}_u$  increment  $m_\kappa$
- 5: **end for**
- 6: sort all possible  $\kappa$  in decreasing order of  $m_\kappa$

### Search phase:

- 7: for each sorted  $\kappa$ , exhaustively look for  $K$

# Differential Probability

## Definition

Given a function  $f$  from  $\{0, 1\}^p$  to  $\{0, 1\}^q$  and given  $a \in \{0, 1\}^p$  and  $b \in \{0, 1\}^q$ , we define

$$\text{DP}^f(a, b) = \Pr_X[f(X \oplus a) = f(X) \oplus b]$$

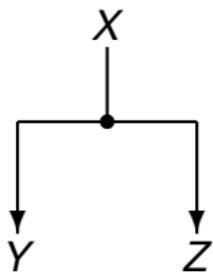
where  $X \in_U \{0, 1\}^p$ .

- Property:  $\text{DP}^f(0, b) = \begin{cases} 1 & \text{if } b = 0 \\ 0 & \text{otherwise} \end{cases}$
- Property: for all  $a$ ,  $\sum_{b \in \{0,1\}^q} \text{DP}^f(a, b) = 1$
- Property:  $2^p \cdot \text{DP}^f(a, b)$  is even

even because if  $f(x + a) = f(x) + b \Rightarrow [ \text{set } x' = x + a ]$   
 $f(x' + a) = f(x + a + a) = f(x) = f(x + a) + b = f(x') + b$

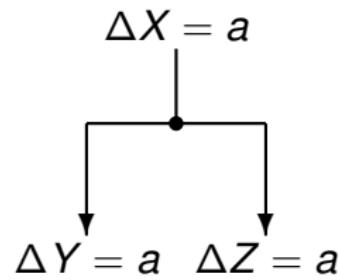
# Differential Circuit — i: Duplicate Gate

computation circuit



$$X = Y = Z$$

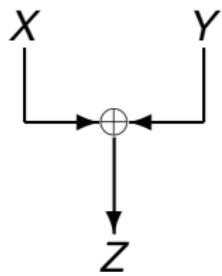
differential circuit



$$\Delta X = a \implies \Delta Y = \Delta Z = a$$

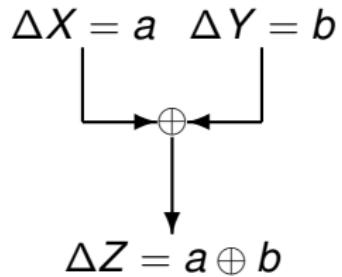
## Differential Circuit — ii: XOR Gate

**computation circuit**



$$Z = X \oplus Y$$

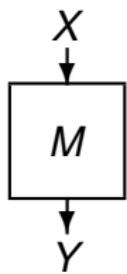
**differential circuit**



$$(\Delta X = a, \Delta Y = b) \implies \Delta Z = a \oplus b$$

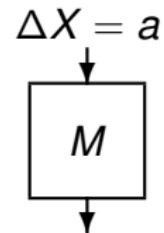
# Differential Circuit — iii: Linear Circuit

computation circuit



$$Y = M \times X$$

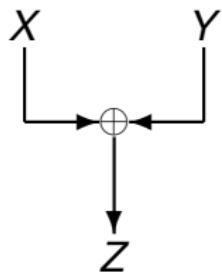
differential circuit



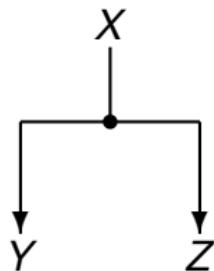
$$\Delta X = a \implies \Delta Y = M \times a$$

# Addition and Duplicate Gates

**XOR Gate**



**Duplicate Gate**

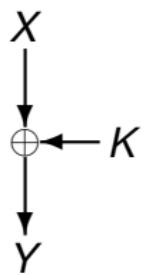


$$(Z) = \begin{pmatrix} 1 & 1 \end{pmatrix} \times \begin{pmatrix} X \\ Y \end{pmatrix}$$

$$\begin{pmatrix} Y \\ Z \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \times (X)$$

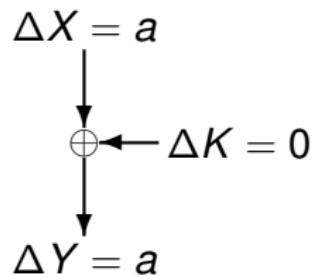
## Differential Circuit — iv: XOR to Constant Gate

computation circuit



$$Y = X \oplus K$$

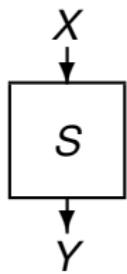
differential circuit



$$\Delta X = a \implies \Delta Y = a$$

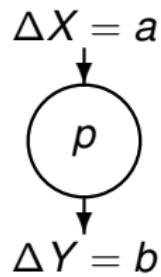
# Differential Circuit — v: Non-Linear Circuit

**computation circuit**



$$Y = S(X)$$

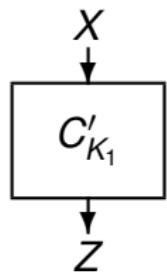
**differential circuit**



$$\Pr[\Delta Y = b | \Delta X = a] = p$$

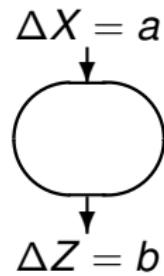
# Differential Characteristic

**computation circuit**



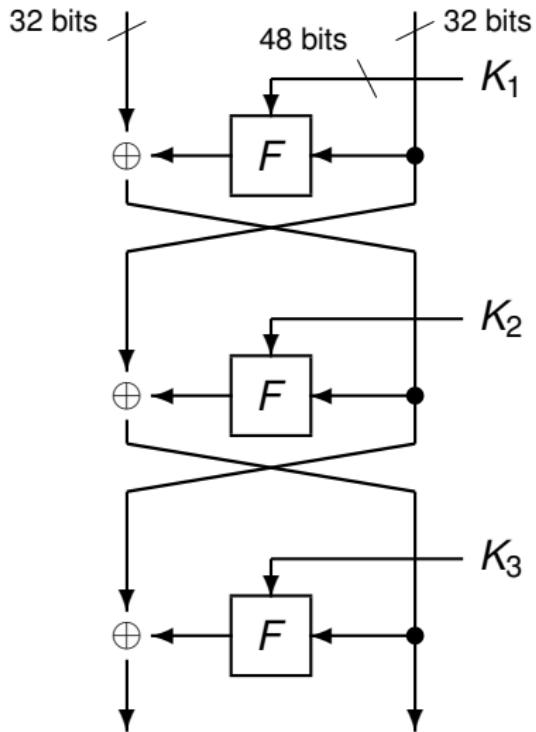
$$Z = C'_{K_1}(X)$$

**differential circuit**

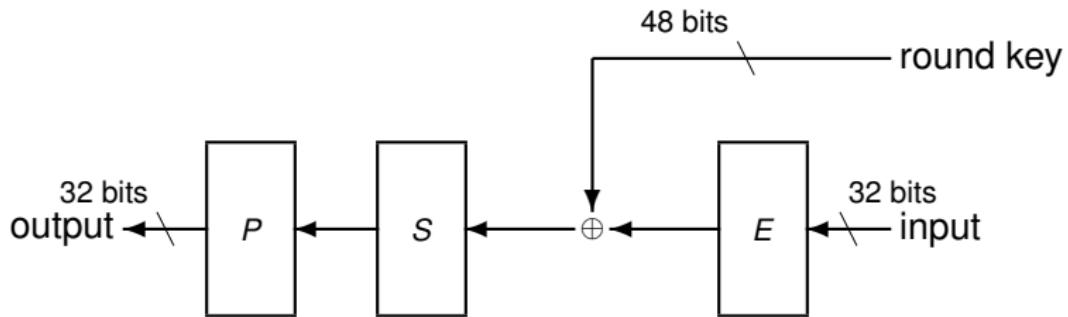


$$\Pr[\Delta Z = b | \Delta X = a] = \text{DP}^{C'_{K_1}}(a, b)$$

## DES (Reminder) — i

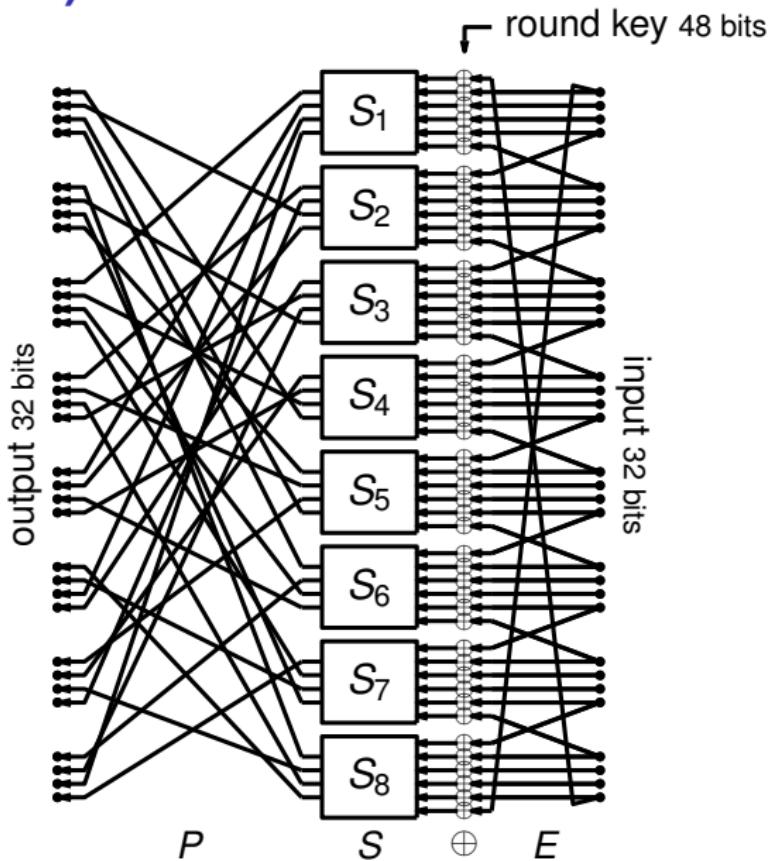


## DES (Reminder) — ii

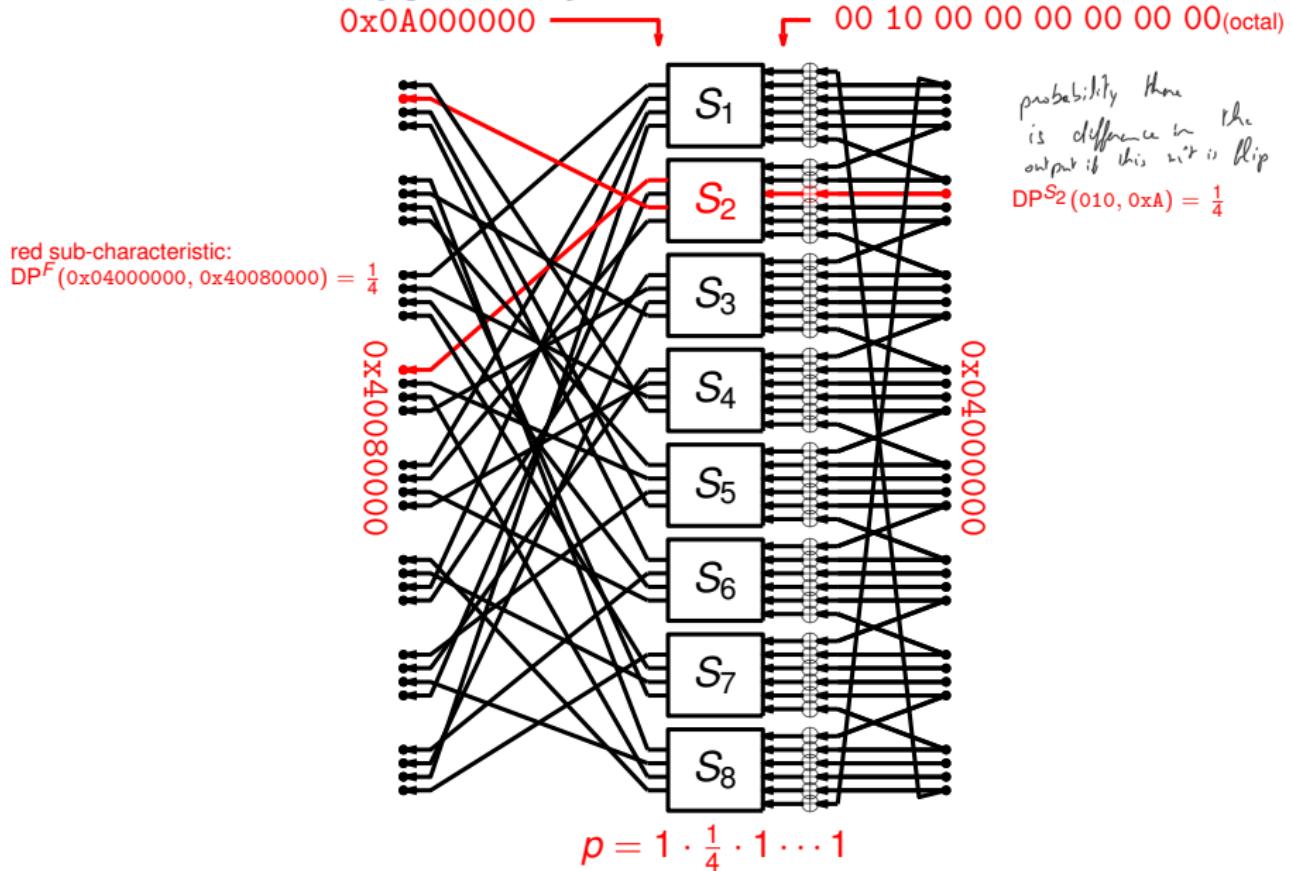


- $E$ : expansion (duplicate one bit out of two)
- $\oplus$ : bitwise XOR with a round key
- $S$ : substitution boxes (eight 6-to-4 bits mappings)
- $P$ : permutation of bits

## DES (Reminder) — iii

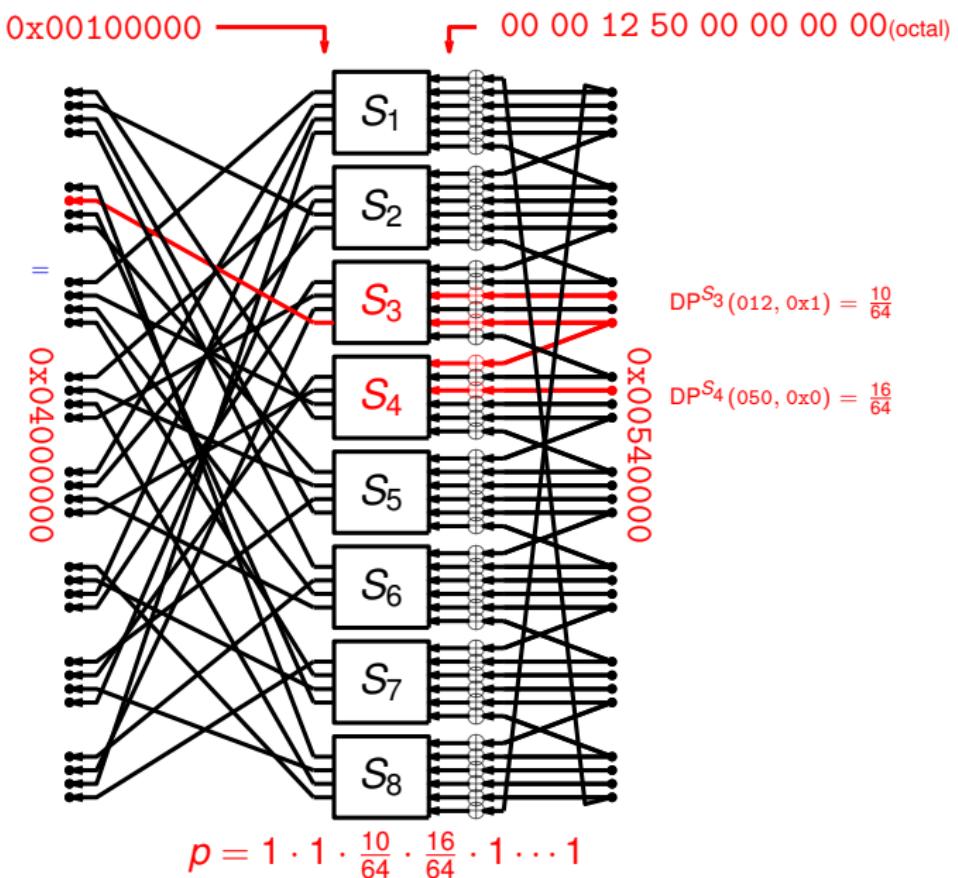


# Differential Cryptanalysis of 8R-DES

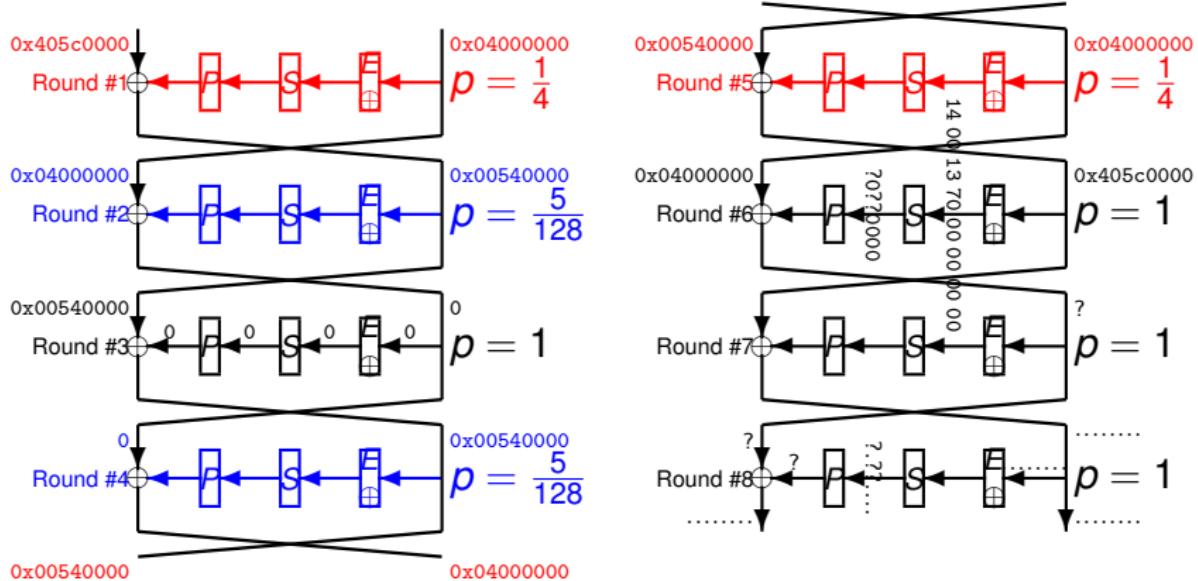


# Other Sub-Characteristic

blue sub-characteristic:  
 $\text{DP}_F^F(0x00540000, 0x04000000)$   
 $\frac{5}{128}$



# Differential Characteristic



$$a = 0x405c0000 \ 0x04000000$$

$$b = 0x04000000 \ 0x405c0000$$

$$\text{DP}^{\text{core}}(a, b) \approx 2^{-13.4}$$

# Ciphertext Pair Analysis (Predicate $R$ )

## Theorem

If the characteristic is satisfied (i.e.,  $\Delta Z = b$ ), then

$$\left( P^{-1}(\Delta y_L \oplus 04000000) \right)_i = S_i(k_i \oplus E(y_R)_i) \oplus S_i(k_i \oplus E(y'_R)_i)$$

for  $i = 2, 5, 6, 7, 8$

and  $k$  is 6 bits subround key

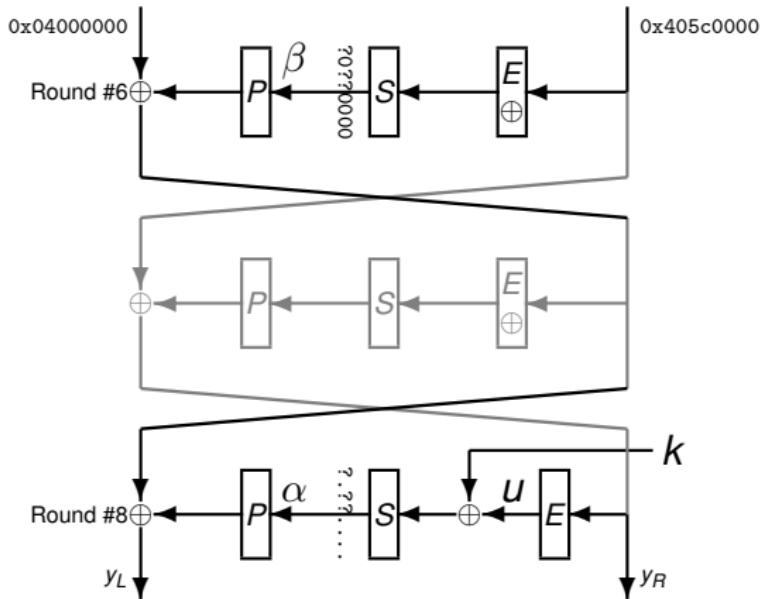
so, we can let



$$\kappa = (k_2, k_5, k_6, k_7, k_8)$$

$\Rightarrow$  30 bits for  $\kappa$  in total of the last round key

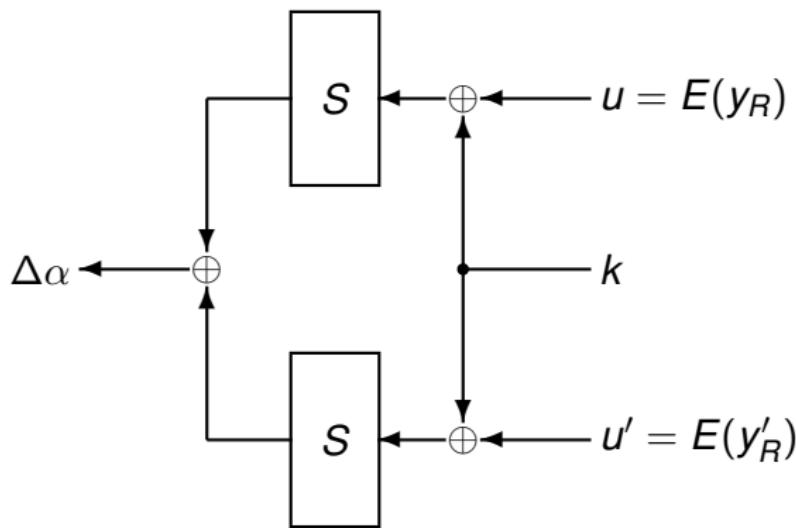
## Proof — i



- $\Delta y_L = 04000000 \oplus P(\Delta\beta \oplus \Delta\alpha)$ ,  $\Delta\beta_i = 0$  for  $i = 2, 5, 6, 7, 8$
- so,  $(P^{-1}(\Delta y_L \oplus 04000000))_i = \Delta\alpha_i$  for  $i = 2, 5, 6, 7, 8$

## Proof — ii

Let  $u = E(y_R)$ ,  $u' = E(y'_R)$ ,  $v = P^{-1}(\Delta y_L \oplus 04000000)$ ,  
 $\Delta\alpha = S(k \oplus u) \oplus S(k \oplus u')$



If  $(x, y)$  and  $(x', y')$  satisfy the characteristic, then  $v_i = \Delta\alpha_i$  for  $i = 2, 5, 6, 7, 8$ . □

## Listing Key Candidates

- If  $(x, y)$  and  $(x', y')$  satisfy the characteristic, then

$$v = P^{-1}(y_L \oplus y'_L \oplus 04000000)$$

tells the output XOR of  $S_i$  in the last round for  $i = 2, 5, 6, 7, 8$ .

- List of 6-bit key values given  $u$ ,  $u'$ , and  $v$   
Define

$$\text{SubCandidate}_{i,\mu,\mu',\nu} = \{r \in \{0, 1\}^6; v = S_i(\mu \oplus r) \oplus S_i(\mu' \oplus r)\}$$

- If  $u = E(y_R)$  and  $u' = E(y'_R)$ , then

$$\prod_{i \in \{2, 5, 6, 7, 8\}} \text{SubCandidate}_{i, u_i, u'_i, v_i}$$

tells the list of all potential  $\kappa = (k_2, k_5, k_6, k_7, k_8)$  possible candidates.

# Implementation

## Precomputation:

- 1: initialize  $\text{SubCandidate}_{i,\mu,\mu',\nu}$  to empty set for  $i = 2, 5, 6, 7, 8$ ,  
 $\mu, \mu' \in \{0, 1\}^6$  and  $\nu \in \{0, 1\}^4$
- 2: for  $i = 2, 5, 6, 7, 8$ , for all  $\mu, \mu', r \in \{0, 1\}^6$ , insert  $r$  in  
 $\text{SubCandidate}_{i,\mu,\mu',S_i(\mu \oplus r) \oplus S_i(\mu' \oplus r)}$

## Collection phase:

- 3: collect  $n$  pairs  $((x, y), (x \oplus a, y'))$  of plaintext-ciphertext pairs

## Analysis phase:

- 4: initialize  $2^{30}$  counters  $m_\kappa$  to 0
- 5: **for** each pair **do**
- 6:   compute  $u = E(y_R)$  and  $u' = E(y'_R)$  and  
 $v = P^{-1}(y_L \oplus y'_L \oplus 04000000)$
- 7:   for all  $\kappa = k_2 k_5 k_6 k_7 k_8$  such that  $k_i \in \text{SubCandidate}_{i,u_i,u'_i,\nu_i}$  for  
 $i = 2, 5, 6, 7, 8$ , increment  $m_\kappa$
- 8: **end for**
- 9: sort all possible 30-bit subkeys  $\kappa$  in decreasing order of  $m_\kappa$

## Search phase:

- 10: for each sorted  $k_2 k_5 k_6 k_7 k_8$ , look for the remaining 26 bits

# Reminder

- $E(X_1 + \dots + X_n) = E(X_1) + \dots + E(X_n)$
- $V(X_1 + \dots + X_n) = V(X_1) + \dots + V(X_n)$  when  $X_1, \dots, X_n$  are independent
- $V(X) = E(X)(1 - E(X))$  when  $X$  is Boolean (support in  $\{0, 1\}$ )
- informally, for  $X_1, \dots, X_n$  iid Boolean of exp. value  $\mu$  we have

$$X_1 + \dots + X_n \approx n\mu \pm \sqrt{n\mu(1 - \mu)}$$

## Complexity Analysis (Heuristic)

- The right 30-bit subkey candidate is suggested once per iteration with probability  $p_1 = \text{DP} = 2^{-13.4}$

$$\text{Signal} = np_1 \pm \sqrt{np_1}$$

- Each iteration suggests  $4^5$  key candidates randomly, so every candidate is suggested once per iteration with probability  $p_2 = 2^{-20}$

$$\text{Noise} = np_2 \pm \sqrt{np_2}$$

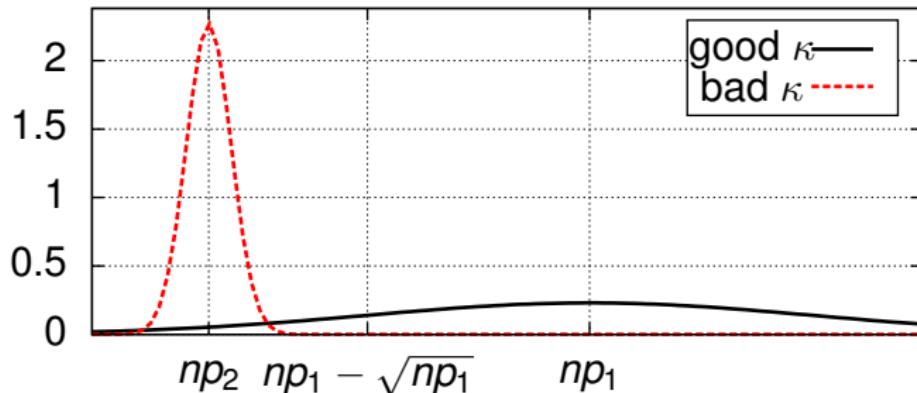
- We need  $n$  such that  $\sqrt{np_1} \ll n(p_1 - p_2) \approx np_1$

$$n \gg \frac{1}{\text{DP}}$$

# Probability Density of Good and Bad Counters

$$y = \frac{d}{dx} \Pr[\text{counter} \leq x]$$

$$n = 3/p_1$$



## 1 The Cryptographic Zoo

## 2 Cryptographic Security Models

## 3 Cryptanalysis (Public-Key)

## 4 The Power of Interaction

## 5 Cryptanalysis (Conventional)

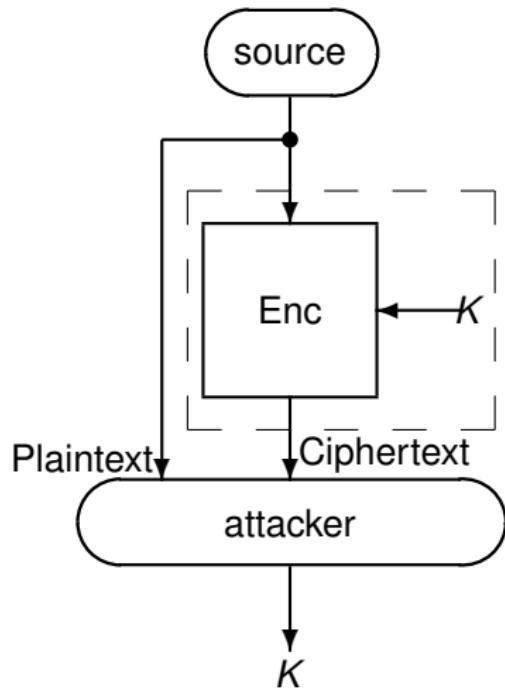
- Block Ciphers
- Differential Cryptanalysis
- **Linear Cryptanalysis**
- Hypothesis Testing in Cryptography
- Decorrelation

## 6 Proving Security

# History

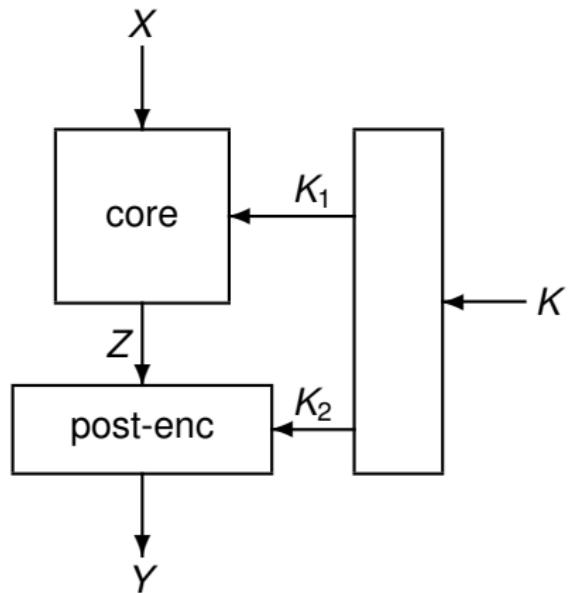
- 1977 DES: an unpopular standard with secret rationales
- 1987 FEAL: a Japanese version
- 1990 Biham-Shamir: differential cryptanalysis
- 1990 Gilbert et al.
- 1993 Matsui: linear cryptanalysis
- 1994 Matsui: application to DES (requires  $2^{43}$  known plaintexts)

# From Chosen to Known Plaintext Key Recovery



## Step 1: Cipher Decomposition

find an appropriate decomposition of following form



(could use pre-encryption as well)

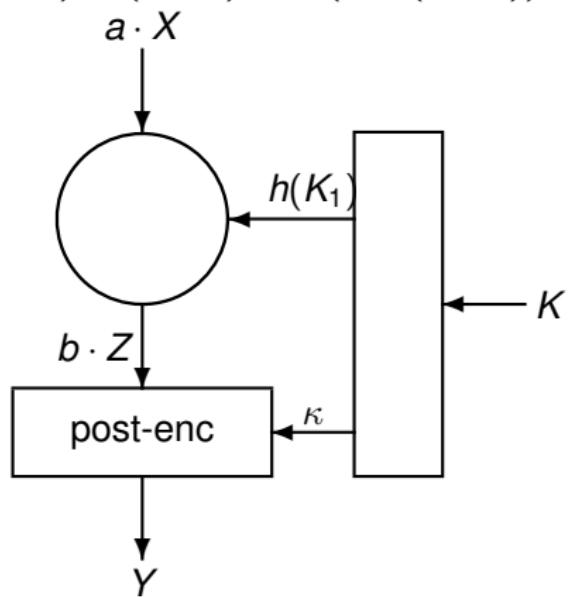
## Step 2: Deviant Property

$$\left| \Pr [X_{i_1} \oplus \cdots \oplus X_{i_r} = Z_{j_1} \oplus \cdots \oplus Z_{j_s}] - \frac{1}{2} \right| \text{ large}$$

$X_{i_1} \oplus \cdots \oplus X_{i_r}$  can be written  $a_1 X_1 \oplus \cdots \oplus a_p X_p = a \cdot X$   
 $Z_{j_1} \oplus \cdots \oplus Z_{j_s}$  can be written  $b_1 Z_1 \oplus \cdots \oplus b_q Z_q = b \cdot Z$

## Step 3: Projection

$$(a \cdot X) \oplus (b \cdot Z) = P(\kappa, \pi(X, Y))$$



(could use pre-encryption as well)

## Step 4: Implementation

### Collection phase:

- 1: **for** all possible  $u = \pi(X, Y)$  **do**
- 2:    initialize a counter  $n_u$  to zero
- 3: **end for**
- 4: collect  $n$  Plaintext-Ciphertext pairs  $(X, Y)$
- 5: **for** each  $(X, Y)$  **do**
- 6:    compute  $u = \pi(X, Y)$
- 7:    increment  $n_u$
- 8: **end for**

### Analysis phase:

- 1: **for** all possible  $\kappa$  **do**
- 2:    compute

$$m_\kappa = \sum_{u \text{ s.t. } P(\kappa, u) = 0} n_u$$

- 3: **end for**

- 4: sort all  $\kappa$  in decreasing order of  $|m_\kappa - \frac{n}{2}|$

### Search phase:

- 5: for each sorted  $\kappa$  exhaustively look for  $K$
- 6: note:  $h(K_2)$  is likely to be  $1_{m_\kappa < n/2}$

# Making Linear Characteristics

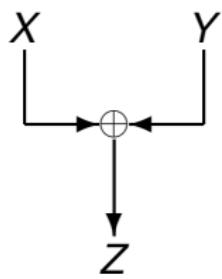
## [Biham, Eurocrypt 94]

Goal: compute  $b \cdot Z = (a \cdot X) \oplus \text{something biased}$

- Put a mask ( $b$ ) at the end and perform the computation in a reverse way.
- Go through substitution boxes as for the differential cryptanalysis.
- Compute the product of all biases.

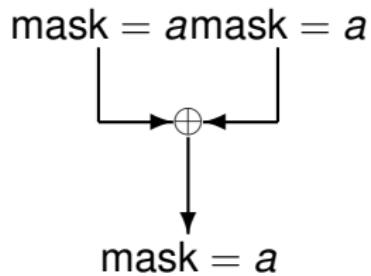
# Dual Circuit — i: XOR Gate

**computation circuit**



$$Z = X \oplus Y$$

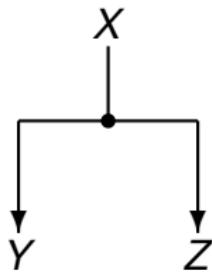
**mask circuit**



$$a \cdot Z = (a \cdot X) \oplus (a \cdot Y)$$

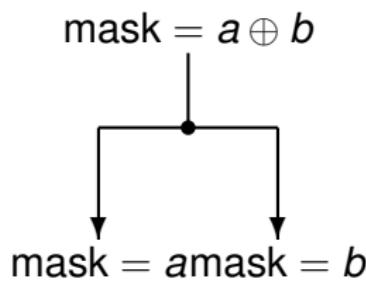
## Dual Circuit — ii: Duplicate Gate

**computation circuit**



$$X = Y = Z$$

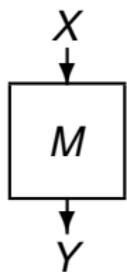
**mask circuit**



$$(a \cdot Y) \oplus (b \cdot Z) = (a \oplus b) \cdot X$$

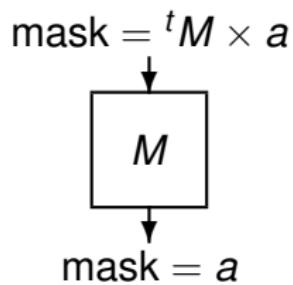
## Dual Circuit — iii: Linear Circuit

**computation circuit**



$$Y = M \times X$$

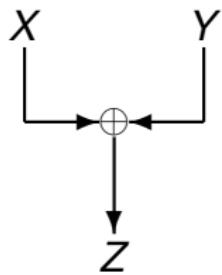
**mask circuit**



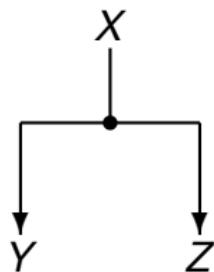
$$a \cdot Y = ({}^t M \times a) \cdot X$$

# Addition and Duplicate Gates

**XOR Gate**



**Duplicate Gate**

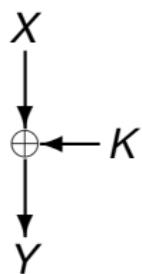


$$(Z) = \begin{pmatrix} 1 & 1 \end{pmatrix} \times \begin{pmatrix} X \\ Y \end{pmatrix}$$

$$\begin{pmatrix} Y \\ Z \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \times (X)$$

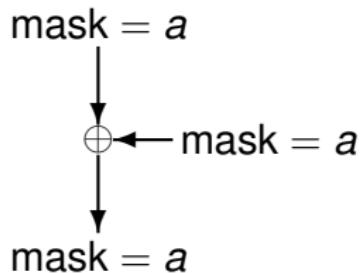
## Dual Circuit — iv: XOR to Constant Gate

**computation circuit**



$$Y = X \oplus K$$

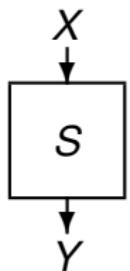
**mask circuit**



$$a \cdot Y = (a \cdot X) \oplus \underbrace{(a \cdot K)}_{\text{constant}}$$

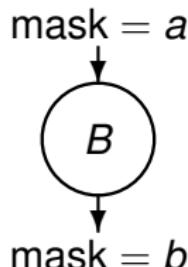
# Dual Circuit — v: Non-Linear Circuit

**computation circuit**



$$Y = S(X)$$

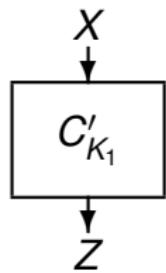
**mask circuit**



$$b \cdot Y = a \cdot X \oplus \underbrace{B}_{\text{biased}}$$

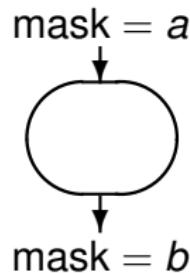
# Linear Characteristic

**computation circuit**



$$Z = C'_{K_1}(X)$$

**mask circuit**



$$(a \cdot X) \oplus (b \cdot Z) = \text{bit}(K_1) \oplus \bigoplus_{i=1}^n B_i$$

# Piling-up Lemma

## Lemma (Piling up Lemma)

For any Boolean variable  $B$  we define

$\text{LP}(B) = (2 \Pr[B = 0] - 1)^2$ . Let  $B_1, \dots, B_n$  be  $n$  independent random variables. We have

$$\text{LP}(B_1 \oplus \dots \oplus B_n) = \text{LP}(B_1) \times \dots \times \text{LP}(B_n).$$

**Proof.** Note that

- $\text{LP}(B) = (\mathbb{E}((-1)^B))^2$ ,
- $(-1)^{A \oplus B} = (-1)^A \times (-1)^B$ ,
- $\mathbb{E}(X \times Y) = \mathbb{E}(X) \times \mathbb{E}(Y)$  when  $X$  and  $Y$  are independent.



# Linear Probability

## Definition

Given a function  $f$  from  $\{0, 1\}^p$  to  $\{0, 1\}^q$  and given  $a \in \{0, 1\}^p$  and  $b \in \{0, 1\}^q$ , we define

$$\text{LP}^f(a, b) = \left( 2 \cdot \Pr_X[a \cdot X = b \cdot f(X)] - 1 \right)^2$$

where  $X \in_U \{0, 1\}^p$ .

$$\begin{aligned} a \cdot X &= a_1 X_1 \oplus \cdots \oplus a_p X_p \\ b \cdot Y &= b_1 Y_1 \oplus \cdots \oplus b_q Y_q \end{aligned}$$

note:  $\text{LP}^f(a, b) = \text{LP}((a \cdot X) \oplus (b \cdot f(X)))$

# Link Between DPs and LPs

## Theorem

Given a function  $f$  from  $\{0, 1\}^p$  to  $\{0, 1\}^q$  we have

$$\begin{aligned}\text{LP}^f(\alpha, \beta) &= (2 \Pr[\alpha \cdot X = \beta \cdot f(X)] - 1)^2 \\ \xrightarrow{\text{discrete Fourier}} \text{DP}^f(a, b) &= \Pr[f(X \oplus a) \oplus f(X) = b] \\ \text{transform of LP} &= 2^{-q} \sum_{\alpha, \beta} (-1)^{a \cdot \alpha \oplus b \cdot \beta} \text{LP}^f(\alpha, \beta)\end{aligned}$$

**Proof.** First observe that  $\text{LP}^f(\alpha, \beta) = (E((-1)^{(\alpha \cdot X) \oplus (\beta \cdot f(X))}))^2 = E((-1)^{(\alpha \cdot (X \oplus Y)) \oplus (\beta \cdot (f(X) \oplus f(Y)))})$  then

$$\begin{aligned}\sum_{\alpha, \beta} (-1)^{a \cdot \alpha \oplus b \cdot \beta} \text{LP}^f(\alpha, \beta) &= E \left( \sum_{\alpha, \beta} (-1)^{(\alpha \cdot (a \oplus X \oplus Y)) \oplus (\beta \cdot (b \oplus f(X) \oplus f(Y)))} \right) \\ &= 2^{p+q} E(1_{X \oplus Y = a, f(X) \oplus f(Y) = b}) \\ &= 2^q \text{DP}^f(a, b)\end{aligned}$$

□

# Link Between DPs and LPs

## Theorem

Given a function  $f$  from  $\{0, 1\}^p$  to  $\{0, 1\}^q$  we have

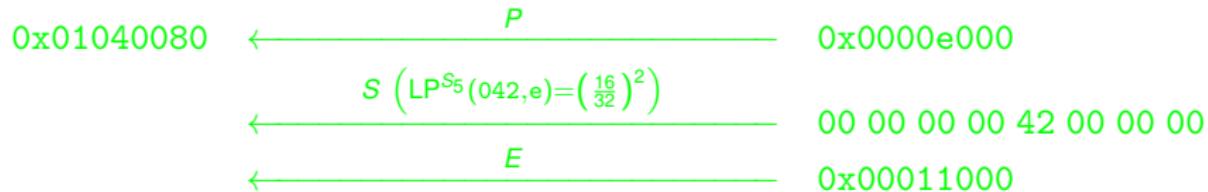
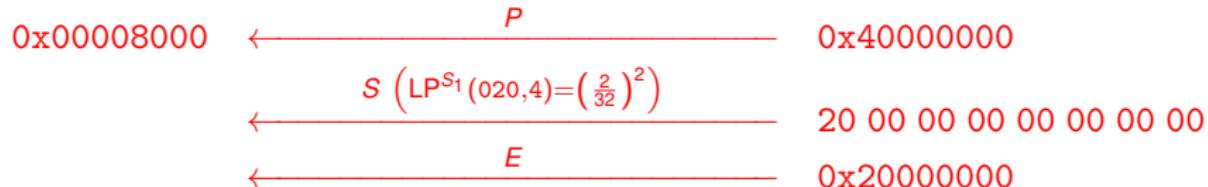
$$\begin{aligned}\text{DP}^f(a, b) &= \Pr[f(X \oplus a) \oplus f(X) = b] \\ \text{LP}^f(\alpha, \beta) &= (2 \Pr[\alpha \cdot X = \beta \cdot f(X)] - 1)^2 \\ &= 2^{-p} \sum_{a,b} (-1)^{a \cdot \alpha \oplus b \cdot \beta} \text{DP}^f(a, b)\end{aligned}$$

## Proof.

$$\begin{aligned}\sum_{a,b} (-1)^{a \cdot \alpha \oplus b \cdot \beta} \text{DP}^f(a, b) &= 2^{-q} \sum_{a,b} \sum_{\alpha', \beta'} (-1)^{a \cdot \alpha \oplus b \cdot \beta} (-1)^{a \cdot \alpha' \oplus b \cdot \beta'} \text{LP}^f(\alpha', \beta') \\ &= 2^{-q} \sum_{\alpha', \beta'} \sum_{a,b} (-1)^{a \cdot (\alpha \oplus \alpha') \oplus b \cdot (\beta \oplus \beta')} \text{LP}^f(\alpha', \beta') \\ &= 2^p \sum_{\alpha', \beta'} \mathbf{1}_{\alpha'=\alpha, \beta'=\beta} \text{LP}^f(\alpha', \beta') \\ &= 2^p \text{LP}^f(\alpha, \beta)\end{aligned}$$



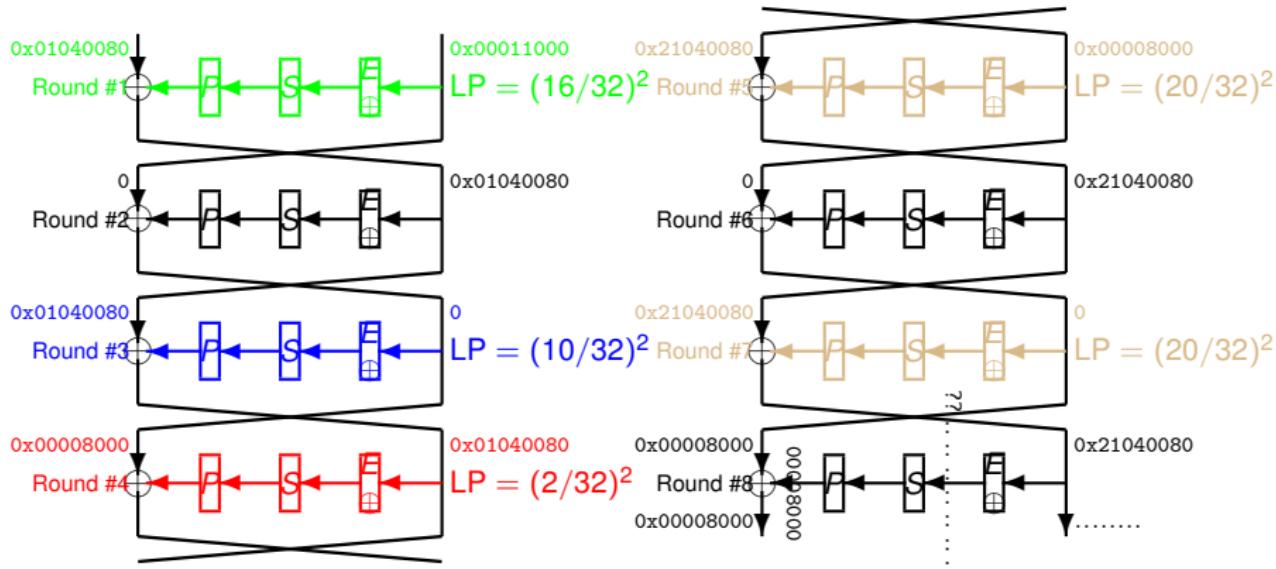
# Application to DES8



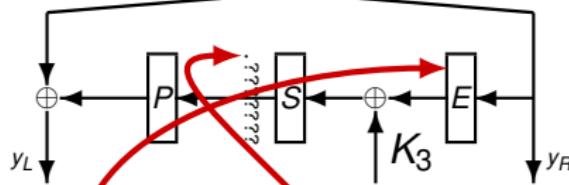
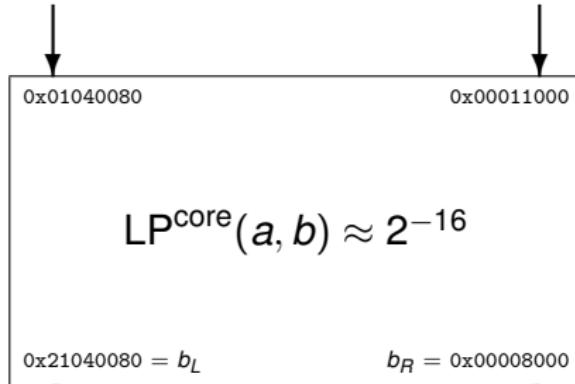
# Other Sub-Characteristics

|            |   |                         |
|------------|---|-------------------------|
| 0x01040080 | $P$   | 0x0000e000              |
|            | $S \left( LP^{S_5}(020,e) = \left(\frac{10}{32}\right)^2 \right)$ | 00 00 00 00 20 00 00 00 |
|            | $E$   | 0x00008000              |
| 0x21040080 | $P$   | 0x0000f000              |
|            | $S \left( LP^{S_5}(020,f) = \left(\frac{20}{32}\right)^2 \right)$ | 00 00 00 00 20 00 00 00 |
|            | $E$   | 0x00008000              |

# Linear Characteristic



# Projection



$$(a \cdot X) \oplus (b_L \cdot Z_L) \oplus (b_R \cdot Z_R)$$

- $a = 0x01040080\ 00011000$ ,  $b = 0x21040080\ 00008000$ ,  $\kappa = (K_3)_1$
- $\pi(x, y) = \begin{pmatrix} u = (E(y_R))_1 \\ v = (a \cdot x) \oplus (b_L \cdot y_R) \oplus (b_R \cdot y_L) \end{pmatrix}$
- Project( $\kappa, (u, v)$ ) =  $v \oplus (b_R \cdot P(S_1(\kappa \oplus u) \parallel 0x00000000))$

# Attack

## Collection phase:

- 1: initialize  $2^7$  counters  $n_{u,v}$  to zero for all possible 6-bit values  $u$  and all possible bits  $v$ .
- 2: collect  $n$  plaintext-ciphertext  $(x, y)$  pairs,
- 3: **for** each  $(x, y)$  pair **do**
- 4:   set  $u$  to the 6 leading bits of the expansion  $E(y_R)$  of  $y_R$  in the round function
- 5:    $v \leftarrow (a \cdot x) \oplus (b_L \cdot y_R) \oplus (b_R \cdot y_L)$
- 6:   increment  $n_{u,v}$
- 7: **end for**

## Analysis phase:

- 1: **for** all possible  $\kappa$  **do**
- 2:    $m_\kappa \leftarrow \sum_u n_{u,b_R \cdot (P(S_1(u \oplus \kappa)) \parallel 0 \dots 0)}$
- 3: **end for**
- 4: sort all  $\kappa$  in decreasing order of  $|m_\kappa - \frac{n}{2}|$ ,

## Search phase:

- 5: do an exhaustive search by using the sorted list for  $\kappa$ .

# Analysis

## Theorem

*For  $n \approx \frac{1}{LP}$ , the correct value  $\kappa$  is first in the sorted list with high probability.*

Example: for 8-round DES,  $LP = 2^{-16}$

## 1 The Cryptographic Zoo

## 2 Cryptographic Security Models

## 3 Cryptanalysis (Public-Key)

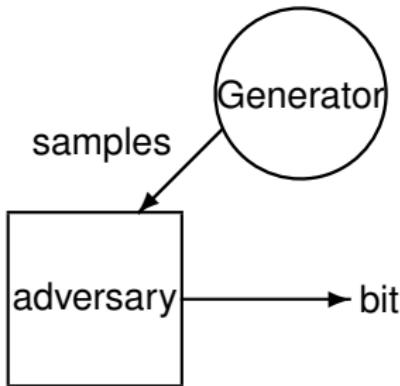
## 4 The Power of Interaction

## 5 Cryptanalysis (Conventional)

- Block Ciphers
- Differential Cryptanalysis
- Linear Cryptanalysis
- Hypothesis Testing in Cryptography
- Decorrelation

## 6 Proving Security

# Indistinguishability



Problem: say whether all samples follow distribution  $P_0$  or they all follow distribution  $P_1$

# Advantage

## Definition

Two samplable distributions  $P_0$  and  $P_1$  are  $(q, \varepsilon)$ -indistinguishable if for any algorithm  $\mathcal{A}$  taking  $q$  iid random variables  $x_1, \dots, x_q$  following  $P$  we have

$$|\text{Adv}_{\mathcal{A}}(P_0, P_1)| \leq \varepsilon$$

where

$$\text{Adv}_{\mathcal{A}}(P_0, P_1) = \Pr[\mathcal{A} \rightarrow 1 | P = P_1] - \Pr[\mathcal{A} \rightarrow 1 | P = P_0]$$

A notion of distance between  $P_0$  and  $P_1$ :

$$\text{distance}_q(P_0, P_1) = \max_{\substack{\text{distinguisher} \\ \text{limited to } q}} |\Pr[\mathcal{A} \rightarrow 1 | P = P_1] - \Pr[\mathcal{A} \rightarrow 1 | P = P_0]|$$

# Applications

- **pseudorandom number generator**  
break a PRNG  $\Rightarrow$  distinguish from an ideal RNG
- **block cipher** and **stream cipher** cryptanalysis  
distinguish biased bits in known plaintext-ciphertexts
- **semantic security** of public-key cryptography  
distinguish between the encryption of two known plaintexts
- **commitment, zero-knowledge**, etc

# Hypothesis Testing

## Problem

Given a source producing random variables, decide upon several hypotheses.

Example:

- iid random variables following either

**Hypothesis  $H_0$ :** variables follow distribution  $P_0$

**Hypothesis  $H_1$ :** variables follow distribution  $P_1$

- iid random variables following either

**Hypothesis  $H_0$ :** variables follow distribution  $P_0$

**Hypothesis  $H_1$ :** variables follow distribution in  
 $\{P_1, \dots, P_n\}$

## Two Approaches

- **Frequentist approach**

Consider two types of errors

**type I error:**  $\alpha = \Pr[\mathcal{A} \rightarrow 1 | P_0]$

**type II error:**  $\beta = \Pr[\mathcal{A} \rightarrow 0 | P_1]$

- **Bayesian approach**

Assign cost to error type (or prior probability to hypotheses)

$$P_e = \Pr[\mathcal{A} \rightarrow 1 | P_0] \pi_0 + \Pr[\mathcal{A} \rightarrow 0 | P_1] \pi_1$$

typical case for crypto:  $\pi_0 = \pi_1 = \frac{1}{2}$

$$\text{Adv}_{\mathcal{A}} = \Pr[\mathcal{A} \rightarrow 1 | P_1] - \Pr[\mathcal{A} \rightarrow 1 | P_0]$$

$$\text{Adv}_{\mathcal{A}} = (1 - \beta) - \alpha = 1 - 2P_e$$

$$1 - \text{Adv}_{\mathcal{A}} = \alpha + \beta = 2P_e$$

# Problems for this Lecture

- What is the best way to distinguish two distributions?
- How many samples do we need to distinguish two distributions with *significant* advantage?

# Best Advantage

Case  $q = 1$

- let  $\mathcal{A}$  be an arbitrary distinguisher
- w.l.o.g. we can assume it is deterministic (we assume no computational bound)
  - let  $\mathcal{A}^{-1}(1)$  be the set of values  $x$  such that  $\mathcal{A} \rightarrow 1$  when  $X = x$
- we have

$$\text{Adv}_{\mathcal{A}} = \sum_{x \in \mathcal{A}^{-1}(1)} (P_1(x) - P_0(x))$$

- clearly

$$\text{Adv}_{\mathcal{A}} \leq \sum_{x; P_0(x) \leq P_1(x)} (P_1(x) - P_0(x))$$

- we have

$$\sum_{x; P_0(x) \leq P_1(x)} (P_1(x) - P_0(x)) = \frac{1}{2} \sum_x |P_1(x) - P_0(x)|$$

# Statistical Distance

## Definition (= $L_1$ distance)

Given two real functions  $f_0$  and  $f_1$  over a discrete set  $\mathcal{Z}$  we define the **statistical distance**  $d(f_0, f_1)$  by

$$d(f_0, f_1) = \frac{1}{2} \sum_{x \in \mathcal{Z}} |f_1(x) - f_0(x)|$$

## Theorem

Given two distributions  $P_0$  and  $P_1$ , all distinguishers using a **single** sample verify

$$\text{Adv}_{\mathcal{A}} \leq d(P_0, P_1)$$

# Best Distinguisher (Single Sample)

**input:**  $x$

1:  $R = \frac{P_0(x)}{P_1(x)}$

2: **if**  $R \leq 1$  **then**

3:      $b \leftarrow 1$

4: **else**

5:      $b \leftarrow 0$

6: **end if**

**output:**  $b$

- $R$  is the *likelihood ratio*

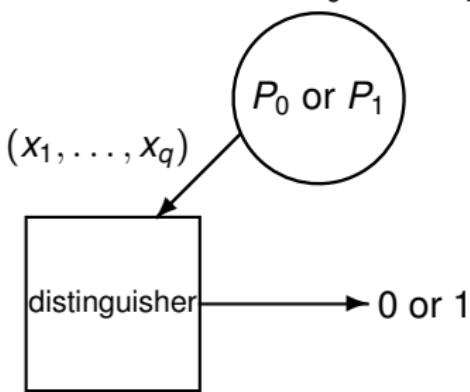
$$\text{Adv}_{\mathcal{A}} = d(P_0, P_1)$$

- caveat:  $\frac{p}{0} = +\infty$

- remark:  $\frac{0}{0}$  never occurs

## General Case

trick: consider  $X = (X_1, \dots, X_q)$  as a random variable with distribution either  $P_0^{\otimes q}$  or  $P_1^{\otimes q}$



The best possible advantage is obtained by the likelihood ratio test:

$$\text{output 1} \iff \frac{\Pr_{P_0^{\otimes q}}[X_1, \dots, X_q]}{\Pr_{P_1^{\otimes q}}[X_1, \dots, X_q]} \leq 1$$

# Best Distinguisher (Multiple Samples)

**input:**  $x_1, \dots, x_q$

1:  $R = \frac{P_0(x_1) \times \dots \times P_0(x_q)}{P_1(x_1) \times \dots \times P_1(x_q)} = \prod_x \frac{P_0(x)^{q_x}}{P_1(x)^{q_x}}$   
with  $q_x = \#\{i; x_i = x\}$

2: **if**  $R \leq 1$  **then**

3:      $b \leftarrow 1$

4: **else**

5:      $b \leftarrow 0$

6: **end if**

**output:**  $b$

## Example i: Biased Coin

$$P_0 = \text{uniform} \quad P_1 = \begin{pmatrix} \text{head} & \text{tail} \\ 1 & 2 \\ \downarrow & \downarrow \\ \frac{1}{2}(1 + \varepsilon) & \frac{1}{2}(1 - \varepsilon) \end{pmatrix}$$

| $x_1$ | $x_2$ | $R$  | outcome |
|-------|-------|--|---------|
| 1     | 1     | $\frac{1}{(1+\varepsilon)^2}$              | 1       |
| 2     | 2     | $\frac{1}{(1-\varepsilon)^2}$              | 0       |
| 1     | 2     | $\frac{1}{(1+\varepsilon)(1-\varepsilon)}$ | 0       |

$$\begin{aligned} \text{output 1} &\iff \frac{1}{(1 + \varepsilon)^{q_1}(1 - \varepsilon)^{q_2}} \leq 1 \\ &\iff q_1 \log(1 + \varepsilon) + q_2 \log(1 - \varepsilon) \geq 0 \iff q_2 < q_1 \end{aligned}$$

## Example ii: Biased Dice

$$P_0 = \text{uniform} \quad P_1 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ \frac{1}{6} + \varepsilon & \frac{1}{6} & \frac{1}{6} + \varepsilon & \frac{1}{6} - \varepsilon & \frac{1}{6} - \varepsilon & \frac{1}{6} \end{pmatrix}$$

| $x_1$ | $x_2$ | $R$   | outcome |
|-------|-------|---|---------|
| 1     | 2     | $\frac{\frac{1}{6} \times \frac{1}{6}}{(\frac{1}{6} + \varepsilon) \times \frac{1}{6}}$                 | 1       |
| 1     | 3     | $\frac{\frac{1}{6} \times \frac{1}{6}}{(\frac{1}{6} + \varepsilon) \times (\frac{1}{6} + \varepsilon)}$ | 1       |
| 2     | 5     | $\frac{\frac{1}{6} \times \frac{1}{6}}{(\frac{1}{6} - \varepsilon) \times \frac{1}{6}}$                 | 0       |

$$\text{output 1} \iff \frac{1}{(1+6\varepsilon)^{q_1+q_3}(1-6\varepsilon)^{q_4+q_5}} \leq 1 \iff q_4 + q_5 \leq q_1 + q_3$$

## Example iii: Uniform over Different Supports

$$P_0 = \text{uniform} \quad P_1 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 \end{pmatrix}$$

| $x_1$ | $x_2$ | $R$  | outcome |
|-------|-------|--|---------|
| 1     | 2     | $\frac{1}{5} \times \frac{1}{5}$<br>$\frac{1}{4} \times \frac{1}{4}$ | 1       |
| 1     | 3     | $\frac{1}{5} \times \frac{1}{5}$<br>$\frac{1}{4} \times \frac{1}{4}$ | 1       |
| 2     | 5     | $\frac{1}{5} \times \frac{1}{5}$<br>$\frac{1}{4} \times 0$           | 0       |

output 1  $\iff q_5 = 0$

## Example iv: Normal of Same Standard Deviation

$$P_0 = \mathcal{N}(\mu, \sigma) \quad P_1 = \mathcal{N}(\mu', \sigma) \quad \mu < \mu'$$

We have

$$\varphi_{\mu, \sigma}(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

and

$$R = \frac{\varphi_{\mu, \sigma}(x)}{\varphi_{\mu', \sigma}(x)}$$

so

$$R \leq 1 \iff x \geq \frac{\mu + \mu'}{2}$$

## Example v: Sum of i.i.d. Bernoulli Variables

$P_b$  : sum of  $n$  instances of  $\text{Ber}(p_b)$

with  $p_0 \approx p_1$  and  $p_0 < p_1$

we approximate  $P_b$  to  $\mathcal{N}(\mu_b, \sigma_b)$  with

$$\mu_b = np_b \quad \sigma_b = \sqrt{np_b(1 - p_b)}$$

and  $\sigma_0 \approx \sigma_1$

so

$$R \leq 1 \iff \frac{x}{n} \geq \frac{p_0 + p_1}{2}$$

# Problem

$$\begin{aligned}\text{Adv}_{\mathcal{A}} &= d(P_0^{\otimes q}, P_1^{\otimes q}) \\ &= \frac{1}{2} \sum_{x_1, \dots, x_q \in \mathcal{Z}} \left| \Pr_{P_1}[x_1] \cdots \Pr_{P_1}[x_q] - \Pr_{P_0}[x_1] \cdots \Pr_{P_0}[x_q] \right|\end{aligned}$$

not very informative about the dependence in terms of  $q$

# Easy Bound

for  $q \ll 1/d(P_0, P_1)$  the advantage must be  $\ll 1$ :

## Theorem

For any  $q$ :

$$d(P_0^{\otimes q}, P_1^{\otimes q}) \leq q \times d(P_0, P_1)$$

## Proof.

$$aa' - bb' = (a - b) \frac{a' + b'}{2} + (a' - b') \frac{a + b}{2}$$

so  $|aa' - bb'| \leq |a - b| \frac{a' + b'}{2} + |a' - b'| \frac{a + b}{2}$  thus (next slide)

$$\frac{1}{2} \sum_{x_1, x_2} |P_1(x_1)Q_1(x_2) - P_0(x_1)Q_0(x_2)| \leq d(P_0, P_1) + d(Q_0, Q_1)$$

and we get  $d(P_0 \otimes Q_0, P_1 \otimes Q_1) \leq d(P_0, P_1) + d(Q_0, Q_1)$   
apply with  $Q_b = P_b^{\otimes(q-1)}$  and iterate

□

## Detail

$$|aa' - bb'| \leq |a - b| \frac{a' + b'}{2} + |a' - b'| \frac{a + b}{2}$$

so

$$\begin{aligned}& \frac{1}{2} \sum_{x_1, x_2} |P_1(x_1)Q_1(x_2) - P_0(x_1)Q_0(x_2)| \\&\leq \frac{1}{2} \sum_{x_1} |P_1(x_1) - P_0(x_1)| \sum_{x_2} \frac{Q_1(x_2) + Q_0(x_2)}{2} + \\&\quad \frac{1}{2} \sum_{x_2} |Q_1(x_2) - Q_0(x_2)| \sum_{x_1} \frac{P_1(x_1) + P_0(x_1)}{2} \\&= \frac{1}{2} \sum_{x_1, x_2} |P_1(x_1) - P_0(x_1)| + \frac{1}{2} \sum_{x_1, x_2} |Q_1(x_2) - Q_0(x_2)| \\&= d(P_0, P_1) + d(Q_0, Q_1)\end{aligned}$$

# Definitions

- Kullback-Leibler divergence

$$D_{\text{KL}}(P_0 \| P_1) = \sum_{x \in \text{Supp}(P_0)} P_0(x) \log \frac{P_0(x)}{P_1(x)}$$

always non-negative, 0 iff  $P_0 = P_1$   
infinite iff  $\text{Supp}(P_0) \not\subseteq \text{Supp}(P_1)$

WARNING: log are in basis 2!

- Neyman divergence

$$D_{\text{N}}(P_0 \| P_1) = \sum_{x \in \text{Supp}(P_0) \cup \text{Supp}(P_1)} \frac{(P_0(x) - P_1(x))^2}{P_1(x)}$$

always non-negative, 0 iff  $P_0 = P_1$   
infinite iff  $\text{Supp}(P_0) \not\subseteq \text{Supp}(P_1)$

# Better Bound

for  $q \ll 1/D_N(P_0\|P_1)$  the advantage must be  $\ll 1$ :

## Theorem

For any  $q$ :

$$d(P_0^{\otimes q}, P_1^{\otimes q}) \leq \sqrt{\frac{q}{2} D_N(P_0\|P_1)}$$

## Proof.

- $d(P_0^{\otimes q}, P_1^{\otimes q}) \leq \sqrt{\frac{1}{2} D_{KL}(P_0^{\otimes q}\|P_1^{\otimes q})}$  (Pinsker Inequality)
- $D_{KL}(P_0^{\otimes q}\|P_1^{\otimes q}) = qD_{KL}(P_0\|P_1)$  (additivity of KL)
- $D_{KL}(P_0\|P_1) \leq D_N(P_0\|P_1)$  [Dai-Hoang-Tessaro  
Crypto 2017]



# Uniform Case

for  $q \ll \frac{1}{N \cdot d(P, U)^2}$  the advantage must be  $\ll 1$ :

## Theorem

If  $U$  is uniform over a support of cardinality  $N$ , for any  $q$ :

$$d(P^{\otimes q}, U^{\otimes q}) \leq \sqrt{2qN} \cdot d(P, U)$$

## Proof.

- $d(P^{\otimes q}, U^{\otimes q}) \leq \sqrt{\frac{q}{2} D_N(P \| U)}$  (previous result)
- $D_N(P \| U) = N \|P - U\|_2^2$  (definition of  $D_N$ )
- $\|P - U\|_2 \leq 2d(P, U)$   
(for  $x_i$  positive,  $\sum x_i^2 \leq (\sum x_i)^2$ )

□

## 1 The Cryptographic Zoo

## 2 Cryptographic Security Models

## 3 Cryptanalysis (Public-Key)

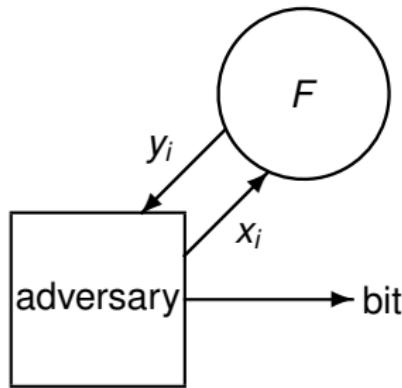
## 4 The Power of Interaction

## 5 Cryptanalysis (Conventional)

- Block Ciphers
- Differential Cryptanalysis
- Linear Cryptanalysis
- Hypothesis Testing in Cryptography
- **Decorrelation**

## 6 Proving Security

# Distinguisher for Random Functions



- the adversary can send chosen queries  $x_i$
- function  $F$  was selected either from one distribution (null hypothesis) or another (alternate hypothesis)
- adversary must guess which distribution was used

# Examples of Distinguishing problem

- $F : \mathcal{A} \rightarrow A$  is a block cipher  $C$  or the perfect cipher  $C^*$

**Hypothesis  $H_0$ :**  $F = C^*$  (uniformly distributes permutation)

**Hypothesis  $H_1$ :**  $F = C_K$  with  $K$  uniformly distributed

- $F : \mathcal{A} \rightarrow B$  is a MAC or a truly random function  $F^*$

**Hypothesis  $H_0$ :**  $F = F^*$  (uniformly distributes function)

**Hypothesis  $H_1$ :**  $F = F_K$  with  $K$  uniformly distributed

- $F : \mathcal{A} \rightarrow A$  is a random function  $F^*$  or a random permutation  $C^*$

**Hypothesis  $H_0$ :**  $F = F^*$  (uniformly distributes function)

**Hypothesis  $H_1$ :**  $F = C^*$  (uniformly distributes permutation)

# Decorrelation

A simple example for  $F : A \rightarrow B$  defined by  $K$ :

- $A = \{0, 1, 2\}$  and  $B = \{0, 1\}$
- $F(x) = (K \cdot x^2 + K \cdot \lfloor \frac{K+x}{2} \rfloor + x + 1) \bmod 2$  for  $K$  uniformly distributed in  $\{1, 2, 3, 4\}$

| $K$ | $F(0)$ | $F(1)$ | $F(2)$ |
|-----|--------|--------|--------|
| 1   | 1      | 0      | 0      |
| 2   | 1      | 0      | 1      |
| 3   | 0      | 1      | 1      |
| 4   | 1      | 0      | 1      |

# First Order of Decorrelation

|         | $y = 0$ | $y = 1$ |
|---------|---------|---------|
| $x = 0$ | 1/4     | 3/4     |
| $x = 1$ | 3/4     | 1/4     |
| $x = 2$ | 1/4     | 3/4     |

$$[F]^1 = \begin{pmatrix} 1/4 & 3/4 \\ 3/4 & 1/4 \\ 1/4 & 3/4 \end{pmatrix}, \quad [F^*]^1 = \begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}$$

1st order of decorrelation of  $F$  = distance between  $[F]^1$  and  $[F^*]^1$

## Second Order of Decorrelation

|                       | $(y_1, y_2) = (0, 0)$ | $(y_1, y_2) = (0, 1)$ | $(y_1, y_2) = (1, 0)$ | $(y_1, y_2) = (1, 1)$ |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| $(x_1, x_2) = (0, 0)$ | 1/4                   | 0                     | 0                     | 3/4                   |
| $(x_1, x_2) = (1, 0)$ | 0                     | 3/4                   | 1/4                   | 0                     |
| $(x_1, x_2) = (2, 0)$ | 0                     | 1/4                   | 1/4                   | 1/2                   |
| $(x_1, x_2) = (0, 1)$ | 0                     | 1/4                   | 3/4                   | 0                     |
| $(x_1, x_2) = (1, 1)$ | 3/4                   | 0                     | 0                     | 1/4                   |
| $(x_1, x_2) = (2, 1)$ | 1/4                   | 0                     | 1/2                   | 1/4                   |
| $(x_1, x_2) = (0, 2)$ | 0                     | 1/4                   | 1/4                   | 1/2                   |
| $(x_1, x_2) = (1, 2)$ | 1/4                   | 1/2                   | 0                     | 1/4                   |
| $(x_1, x_2) = (2, 2)$ | 1/4                   | 0                     | 0                     | 3/4                   |

# Matrices

$$[F]^2 = \begin{pmatrix} 1/4 & 0 & 0 & 3/4 \\ 0 & 3/4 & 1/4 & 0 \\ 0 & 1/4 & 1/4 & 1/2 \\ 0 & 1/4 & 3/4 & 0 \\ 3/4 & 0 & 0 & 1/4 \\ 1/4 & 0 & 1/2 & 1/4 \\ 0 & 1/4 & 1/4 & 1/2 \\ 1/4 & 1/2 & 0 & 1/4 \\ 1/4 & 0 & 0 & 3/4 \end{pmatrix}, \quad [F^*]^2 = \begin{pmatrix} 1/2 & 0 & 0 & 1/2 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/2 & 0 & 0 & 1/2 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/2 & 0 & 0 & 1/2 \end{pmatrix}$$

2nd order of decorrelation of  $F$  = distance between  $[F]^2$  and  $[F^*]^2$

# Definition

## Definition

Given a random function  $F$  from a set  $A$  to a set  $B$  and an integer  $q$ , we define the real matrix  $[F]^q$  as a  $A^q \times B^q$ -type matrix for which the  $((x_1, \dots, x_q), (y_1, \dots, y_q))$ -entry is

$$[F]_{(x_1, \dots, x_q), (y_1, \dots, y_q)}^q = \Pr[F(x_1) = y_1, \dots, F(x_q) = y_q].$$

- A random function  $F$  aimed at being compared to a canonical ideal random function  $F^*$ .
- E.g.  $F$  is a block cipher, and  $F^*$  is a uniformly distributed random permutation.
- Given a distance  $D$  on the vector space of  $A^q \times B^q$ -type real matrices, we define the  $q$ -wise decorrelation bias of  $F$  by

$$\text{Dec}^q(F) = D([F]^q, [F^*]^q).$$

note: various distances will define different decorrelation notions

# Reminder on Matrix Norms

## Definition

Over a vector space  $V$ , a **norm** is a mapping from  $V$  to  $\mathbf{R}_+$  such that

- ①  $\|x\| = 0$  if and only if  $x = 0$
- ②  $\|\lambda x\| = |\lambda| \times \|x\|$  for any  $\lambda \in \mathbf{R}$
- ③  $\|x + y\| \leq \|x\| + \|y\|$

If  $V$  is a matrix space, a **matrix norm** is a norm such that

- ④  $\|xy\| \leq \|x\| \times \|y\|$

# From a Vector Norm to a Matrix Norm

Assume that we have norms over  $\mathbf{R}^p$  and  $\mathbf{R}^q$ . Given a  $p \times q$ -matrix  $M$ , we define

$$|||M||| = \max_{\substack{x \in \mathbf{R}^q \\ \|x\| \leq 1}} \|Mx\| = \max_{\substack{x \in \mathbf{R}^q \\ \|x\|=1}} \|Mx\| = \max_{\substack{x \in \mathbf{R}^q \\ x \neq 0}} \frac{\|Mx\|}{\|x\|}$$

## Theorem

$|||\cdot|||$  is a matrix norm satisfying  $\|Mx\| \leq |||M||| \times \|x\|$ .

# Infinity Norm

## Definition

Given a vector  $V$  and a matrix  $M$ , we define

$$\begin{aligned}\|V\|_\infty &= \max_{\text{row}} |V_{\text{row}}| \\ \||M||\|_\infty &= \max_{V \neq 0} \frac{\|M \times V\|_\infty}{\|V\|_\infty} \\ &= \max_{\text{row}} \sum_{\text{column}} |M_{\text{row},\text{column}}|.\end{aligned}$$

## Example

For

$$[F]^2 - [F^*]^2 = \begin{pmatrix} -0.25 & 0 & 0 & 0.25 \\ -0.25 & 0.5 & 0 & -0.25 \\ -0.25 & 0 & 0 & 0.25 \\ -0.25 & 0 & 0.5 & -0.25 \\ 0.25 & 0 & 0 & -0.25 \\ 0 & -0.25 & 0.25 & 0 \\ -0.25 & 0 & 0 & 0.25 \\ 0 & 0.25 & -0.25 & 0 \\ -0.25 & 0 & 0 & 0.25 \end{pmatrix}$$

we have

$$\begin{aligned}\text{Dec}^2(F) &= |||[F]^2 - [F^*]^2|||_\infty \\ &= \max(0.5, 1, 0.5, 1, 0.5, 0.5, 0.5, 0.5, 0.5) \\ &= 1\end{aligned}$$

# Non-Adaptive Distinguisher

## Theorem

Given two random functions  $F$  and  $G$ , the best **non adaptive** distinguisher between  $F$  and  $G$  which is limited to  $q$  queries is such that

$$\text{Adv}(F, G) = \frac{1}{2} \| [F]^q - [G]^q \|_{\infty}.$$

**Proof.** W.l.o.g. the best non-adaptive distinguisher is deterministic and asks the same question  $x = (x_1, \dots, x_q)$ .

- the response  $Y = (Y_1, \dots, Y_q)$  defines a random variable
- the problem reduces to solving a simple hypothesis testing problem
- best advantage is the statistical distance (half the  $L_1$  norm)
- we maximize over  $x$

□

# Non-Adaptive Distinguisher and Decorrelation

## Corollary

$$\text{BestAdv}_q^{\text{n.a.}}(F, F^*) = \frac{1}{2} \| |[F]^q - [F^*]^q| \|_{\infty} = \frac{1}{2} \text{Dec}_{\| \cdot \|_{\infty}}^q(F)$$

# A-Norm

## Definition

Given a matrix  $M$ , we define

$$\|M\|_a = \max_{x_1} \sum_{y_1} \cdots \max_{x_q} \sum_{y_q} |M_{(x_1, \dots, x_q), (y_1, \dots, y_q)}|.$$

## Theorem

Given two random functions  $F$  and  $G$ , the best **adaptive** distinguisher between  $F$  and  $G$  which is limited to  $q$  queries is such that

$$\text{Adv}(F, G) = \frac{1}{2} \| [F]^q - [G]^q \|_a.$$

## Corollary

$$\text{BestAdv}_q(F, F^*) = \frac{1}{2} \| [F]^q - [F^*]^q \|_a = \frac{1}{2} \text{Dec}_{\|\cdot\|_a}^q(F)$$

# Multiplicativity of Decorrelation

## Theorem

If  $C_1$  and  $C_2$  are two independent random permutations over a set  $A$ ,

$$\text{Dec}^q(C_2 \circ C_1) \leq \text{Dec}^q(C_1) \text{Dec}^q(C_2)$$

when  $\text{Dec}$  is defined from a matrix norm.

so,

$$2\text{BestAdv}_q(C_2 \circ C_1, C^*) \leq (2\text{BestAdv}_q(C_1, C^*)) \times (2\text{BestAdv}_q(C_2, C^*))$$

## Corollary

If  $C_1, \dots, C_r$  are iid random permutations,

$$\text{BestAdv}_q(C_r \circ \dots \circ C_1, C^*) \leq \frac{1}{2} (2\text{BestAdv}_q(C_1, C^*))^r$$

## Proof

- Because of the independence between  $C_1$  and  $C_2$ , we have

$$[C_2 \circ C_1]^q = [C_1]^q \times [C_2]^q$$

- $C^* \circ C_1$ ,  $C_2 \circ C^*$  and  $C^*$  have exactly the same distribution, so

$$\begin{aligned}[C_1]^q \times [C^*]^q &= [C^* \circ C_1]^q = [C^*]^q \\ [C^*]^q \times [C_2]^q &= [C_2 \circ C^*]^q = [C^*]^q\end{aligned}$$

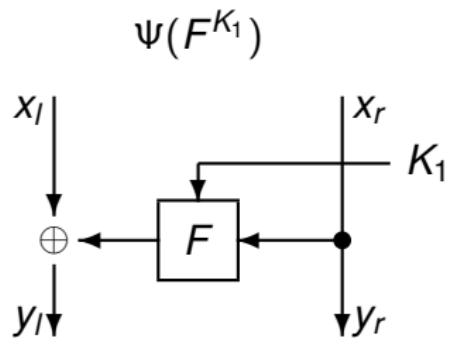
- We obtain

$$([C_1]^q - [C^*]^q) \times ([C_2]^q - [C^*]^q) = [C_2 \circ C_1]^q - [C^*]^q \text{ thus}$$

$$\|[C_2 \circ C_1]^q - [C^*]^q\| \leq \|[C_1]^q - [C^*]^q\| \times \|[C_2]^q - [C^*]^q\|$$



# 1-Round Feistel Scheme is no Good (BestAdv > $\frac{1}{2}$ )

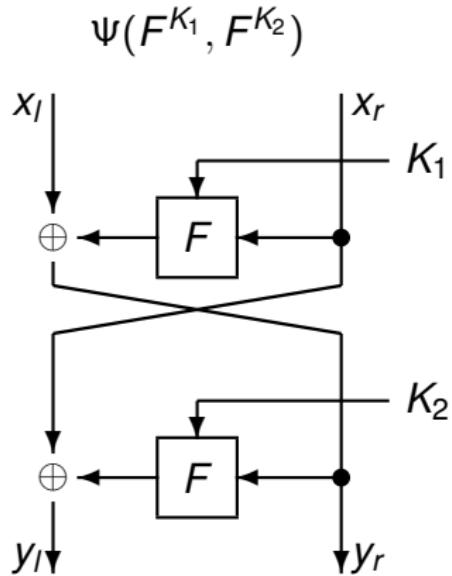


$$x_r = y_r$$

so, we can make a distinguisher with advantage  $1 - 2^{-\frac{\ell}{2}}$ :

- 1: pick  $x$  random
  - 2: get  $y = C(x)$
  - 3: output  $1_{x_r = y_r}$
- size  
of the input

## 2-Round Feistel Scheme is no Good (BestAdv > $\frac{1}{2}$ )

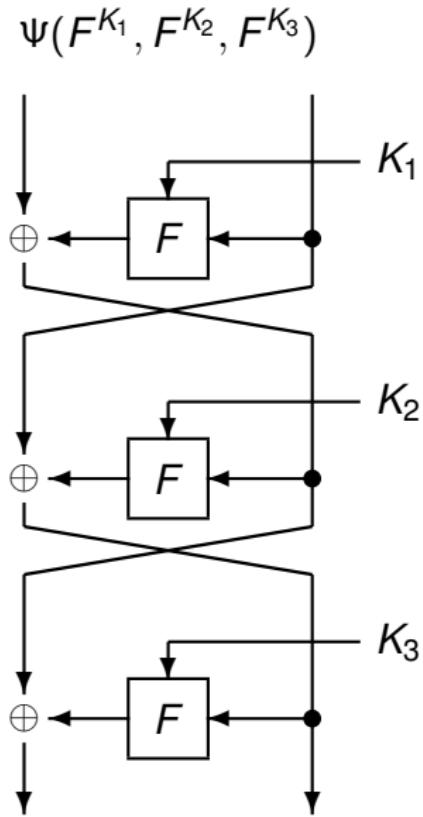


$$x_r = x'_r \implies x_l \oplus y_r = x'_l \oplus y'_r$$

so, we can make a distinguisher with advantage  $1 - 2^{-\frac{\ell}{2}}$ :

- 1: pick  $x, x'$  random s.t.  $x_r = x'_r$
- 2: get  $y = C(x)$  and  $y' = C(x')$
- 3: output  $1_{x_l \oplus y_r = x'_l \oplus y'_r}$

# Feistel Scheme



goal:

$F^{K_1}, F^{K_2}, F^{K_3}$  uniformly distributed

$\Downarrow$   
 $\Psi(F^{K_1}, F^{K_2}, F^{K_3})$  “almost perfect”

*note: not good if the adversary can make chosen plaintext **and** ciphertext queries  
(see midterm exam 2016–17 ex4)*

# Luby-Rackoff Theorem

## Theorem (Luby-Rackoff 1986)

Let  $F_1^*, F_2^*, F_3^*$  be three independent random functions on  $\{0, 1\}^{\frac{\ell}{2}}$  with uniform distribution. We have

$$\text{BestAdv}_q(\Psi(F_1^*, F_2^*, F_3^*), F^*) \leq q^2 \cdot 2^{-\frac{\ell}{2}}$$

$$\text{BestAdv}_q(\Psi(F_1^*, F_2^*, F_3^*), C^*) \leq q^2 \cdot 2^{-\frac{\ell}{2}}$$

where  $F^*$  (resp.  $C^*$ ) is a uniformly distributed random function (resp. permutation).

for the class of distinguishers limited to  $q$  queries to the oracle

# A Convenient Combinatorial Lemma

## Lemma

Let  $q$  be an integer. Let  $F : \mathcal{M}_1 \rightarrow \mathcal{M}_2$  be a random function. We let  $\mathcal{X}$  be the subset of  $\mathcal{M}_1^q$  of all  $(x_1, \dots, x_q)$  with pairwise different entries. We let  $F^* : \mathcal{M}_1 \rightarrow \mathcal{M}_2$  be a uniformly distributed random function. We assume there exists a subset  $\mathcal{Y} \subseteq \mathcal{M}_2^q$  and two positive numbers  $\epsilon_1$  and  $\epsilon_2$  such that

- $\frac{\#\mathcal{Y}}{\#\mathcal{M}_2^q} \geq 1 - \epsilon_1$
- $\forall x \in \mathcal{X} \quad \forall y \in \mathcal{Y} \quad [F]_{x,y}^q \geq \frac{1}{\#\mathcal{M}_2^q} (1 - \epsilon_2).$

this is  $[F^*]_{x,y}^q$

Then we have  $\text{BestAdv}_q(F, F^*) \leq \epsilon_1 + \epsilon_2$ .

“if  $[F]_{x,y}^q \approx [F^*]_{x,y}^q$  for almost all  $y$ 's, then  $\text{BestAdv}_q(F, F^*)$  is small”

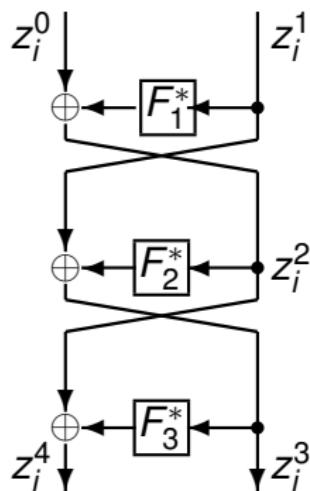
# Proof of the Luby-Rackoff Theorem — i

Following the Feistel scheme, we let

$$x_i = (z_i^0, z_i^1) \quad z_i^2 = z_i^0 \oplus F_1^*(z_i^1) \quad y_i = (z_i^4, z_i^3)$$

- Event  $E$ :  $z_i^3 = z_i^1 \oplus F_2^*(z_i^2)$  and  $z_i^4 = z_i^2 \oplus F_3^*(z_i^3)$  for  $i = 1, \dots, q$ .
- Event  $E^2$ : all  $z_i^2$ 's are pairwise different (depends on  $F_1^*$  only).
- $\mathcal{Y} = \{(y_1, \dots, y_q); \forall i < j \quad z_i^3 \neq z_j^3\}$ .

We have  $[F]_{x,y}^q = \Pr[E]$



## Proof of the Luby-Rackoff Theorem — ii

- We have

$$|\mathcal{Y}| \geq \left(1 - \frac{q(q-1)}{2} 2^{-\frac{\ell}{2}}\right) 2^{\ell q}$$

thus we can let  $\epsilon_1 = \frac{q(q-1)}{2} 2^{-\frac{\ell}{2}}$ .

- For  $y \in \mathcal{Y}$  and any  $x$  (with pairwise different entries), we need to consider  $[F]_{x,y}^q$ . We have

$$[F]_{x,y}^q = \Pr[E] \geq \Pr[E \wedge E^2] = \Pr[E|E^2] \Pr[E^2].$$

For computing  $\Pr[E|E^2]$  we know that  $z_i^3$ 's are pairwise different, as for the  $z_i^2$ 's. Hence  $\Pr[E|E^2] = 2^{-\ell q}$ . It is then straightforward that  $\Pr[E^2] \geq 1 - \frac{q(q-1)}{2} 2^{-\frac{\ell}{2}}$  which is set to  $1 - \epsilon_2$ .

- We thus obtain from the Lemma that

$$\text{BestAdv}_q(F, F^*) \leq q(q-1)2^{-\frac{\ell}{2}}.$$

It remains to show  $\text{BestAdv}_q(F^*, C^*) \leq q2^{-\frac{\ell}{2}}$  (next slide).



# Random Permutation vs Random Function

## Theorem

Let  $F$  (resp.  $C$ ) be a uniformly distributed random function (resp. permutation) over  $\{0, 1\}^\ell$ . We have  $\text{BestAdv}_q(F, C) \leq \frac{q(q-1)}{2} 2^{-\ell}$ .

So,  $\text{BestAdv}_q(F, C) \leq \min(q^2 2^{-\ell}, 1) \leq \min(q 2^{-\frac{\ell}{2}}, 1)$ .

**Proof.** Let  $\mathcal{A}$  be a distinguisher limited to  $q$  queries.

We assume w.l.o.g. that  $\mathcal{A}$  never repeats a query.

Let  $x_i$  be the  $i$ th query.

Conditioned to the event  $E$ : no  $F(x_i)$  collide, the distribution of  $(F(x_1), \dots, F(x_q))|E$  and  $(C(x_1), \dots, C(x_q))$  are identical. So,

$$\Pr[\mathcal{A}^F = 1] - \Pr[\mathcal{A}^C = 1] \leq \Pr[\mathcal{A}^F = 1|E] - \Pr[\mathcal{A}^C = 1] + \Pr[\neg E] = \Pr[\neg E]$$

Then,  $\Pr[\neg E] \leq \sum_{1 \leq i < j \leq q} \Pr[F(x_i) = F(x_j)] = \frac{q(q-1)}{2} 2^{-\ell}$ .

□

$$\Pr[A] = \Pr[A, E] + \Pr[A, \neg E] \leq \Pr[A|E] \Pr[E] + \Pr[\neg E] \leq \Pr[A|E] + \Pr[\neg E]$$

## Proof of Lemma — i

- We use the characterization of  $\text{Dec}_{\|\cdot\|_a}^q$  in term of best adaptive distinguisher. We let  $\mathcal{A}$  be a distinguisher between  $F$  and  $F^*$  limited to  $q$  oracle calls with maximum advantage.
- W.l.o.g. the behavior of  $\mathcal{A}$  is deterministically defined by the oracle responses  $y = (y_1, \dots, y_q)$ . We let  $x_i$  denotes the  $i$ th query defined by  $y$  and  $x = (x_1, \dots, x_q)$  be defined by  $y$ . We let  $A$  be the set of all  $y$  for which  $\mathcal{A}$  accepts. We have

$$\text{Adv}_{\mathcal{A}}(F, F^*) = \Pr[\mathcal{A}(F^*)] - \Pr[\mathcal{A}(F)] = \sum_{y \in A} ([F^*]_{x,y}^q - [F]_{x,y}^q).$$

Since  $[F^*]_{x,y}^q - [F]_{x,y}^q \leq \epsilon_2 [F^*]_{x,y}^q$  in  $\mathcal{Y}$ , we have

$$\text{Adv}_{\mathcal{A}}(F, F^*) \leq \sum_{\substack{y \in A \\ y \in \mathcal{Y}}} \epsilon_2 [F^*]_{x,y}^q + \sum_{\substack{y \in A \\ y \notin \mathcal{Y}}} [F^*]_{x,y}^q.$$

## Proof of Lemma — ii

$$\text{Adv}_{\mathcal{A}}(F, F^*) \leq \sum_{\substack{y \in A \\ y \in \mathcal{Y}}} \epsilon_2 [F^*]_{x,y}^q + \sum_{\substack{y \in A \\ y \notin \mathcal{Y}}} [F^*]_{x,y}^q.$$

- The first sum is upper bounded by  $\epsilon_2$ :

$$\sum_{\substack{y \in A \\ y \in \mathcal{Y}}} [F^*]_{x,y}^q \leq \sum_y [F^*]_{x,y}^q = 1$$

- For the second sum, we recall that all  $x_i$ 's are pairwise different, so

$$\sum_{\substack{y \in A \\ y \notin \mathcal{Y}}} [F^*]_{x,y}^q = \sum_{\substack{y \in A \\ y \notin \mathcal{Y}}} p_0 \leq \sum_{y \notin \mathcal{Y}} p_0 = ((\#\mathcal{M}_2)^q - |\mathcal{Y}|) p_0 \leq \varepsilon_1$$

so  $\text{Adv}_{\mathcal{A}}(F, F^*) \leq \varepsilon_1 + \epsilon_2$ .



# Extension to Feistel Schemes

## Theorem

Let  $F_1, F_2, F_3$  be 3 independent random functions on  $\{0, 1\}^{\frac{\ell}{2}}$  such that  $\text{BestAdv}_q(F_i, F^*) \leq \varepsilon$ . We have

$$\text{BestAdv}_q(\Psi(F_1, F_2, F_3), C^*) \leq q^2 \cdot 2^{-\frac{\ell}{2}} + 3\varepsilon$$

where  $C^*$  is a uniformly distributed random permutation.

## Proof.

$$\begin{aligned} & \text{BestAdv}_q(\Psi(F_1, F_2, F_3), C^*) \\ \leq & \text{BestAdv}_q(\Psi(F_1, F_2, F_3), \Psi(F_1, F_2, F_3^*)) + \text{BestAdv}_q(\Psi(F_1, F_2, F_3^*), \Psi(F_1, F_2^*, F_3^*)) + \\ & \text{BestAdv}_q(\Psi(F_1, F_2^*, F_3^*), \Psi(F_1^*, F_2^*, F_3^*)) + \text{BestAdv}_q(\Psi(F_1^*, F_2^*, F_3^*), C^*) \\ \leq & \underbrace{\text{BestAdv}_q(F_3, F_3^*)}_{\leq \varepsilon} + \underbrace{\text{BestAdv}_q(F_2, F_2^*)}_{\leq \varepsilon} + \underbrace{\text{BestAdv}_q(F_1, F_1^*)}_{\leq \varepsilon} + \underbrace{\text{BestAdv}_q(\Psi(F_1^*, F_2^*, F_3^*), C^*)}_{\leq q^2 \cdot 2^{-\frac{\ell}{2}} \quad [LR]} \end{aligned}$$



# Iterating

## Theorem

Let  $F_1, \dots, F_{3r}$  be  $3r$  independent random functions on  $\{0, 1\}^{\frac{\ell}{2}}$  such that  $\text{BestAdv}_q(F_i, F^*) \leq \varepsilon$ . We have

$$\text{BestAdv}_q(\Psi(F_1, \dots, F_{3r}), C^*) \leq \frac{1}{2} \left( 2q^2 \cdot 2^{-\frac{\ell}{2}} + 6\varepsilon \right)^r$$

where  $C^*$  is a uniformly distributed random permutation.

# Provable Security on Feistel Schemes

$$\text{BestAdv}_q(\Psi(F_1, \dots, F_{3r}), C^*) \leq \frac{1}{2} \left( 2q^2 \cdot 2^{-\frac{\ell}{2}} + 6\varepsilon \right)^r$$

- so, if  $q \ll 2^{\frac{\ell}{4}}$  the best advantage is negligible
- problem: this bound is not so good  
example for  $\ell = 64$  and  $r = 5$  ( $\approx$ DES):

| $q$   | $2^0$      | $2^4$      | $2^8$     | $2^{12}$  | $2^{16}$ |
|---|------------|------------|-----------|-----------|----------|
| $\frac{1}{2} \left( 2q^2 \cdot 2^{-\frac{\ell}{2}} \right)^r$ | $2^{-156}$ | $2^{-116}$ | $2^{-76}$ | $2^{-36}$ | $2^4$    |

- This theory may not be so useful when considering attacks with a large number  $q$  of queries.

# Link with Differential and Linear Probabilities

*advantage of the differential distinguisher with  $q = 2$*

## Theorem

$$E(DP^{C^*}(a, b))$$

Given a random  $C$  on  $\{0, 1\}^\ell$  and  $a, b \neq 0$  we have

$$E(DP^C(a, b)) \leq \frac{1}{2^\ell - 1} + \frac{1}{2} \|[C]^2 - [C^*]^2\|_\infty$$

$$E(LP^C(a, b)) \leq \frac{1}{2^\ell - 1} + 4 \|[C]^2 - [C^*]^2\|_\infty$$

Consequence: making a good decorrelation of order 2 protects against differential and linear cryptanalysis.

## Example: DFCv2

- Team design from ENS, published in 2001
- Patented by CNRS
- family of block ciphers with flexible parameters (nominal choice below)
- block cipher with 128-bit blocks
- dedicated to 64-bit microprocessors
- key length from 0 to 256
- Feistel scheme with 8 rounds with round functions decorrelated to the order 2
- $\| [C]^2 - [C^*]^2 \|_\infty \leq 2^{-115}$  (assuming independent round keys)

# Conclusion

- differential and linear cryptanalysis
- theory on best distinguishers
- decorrelation as a tool to *prove* security

## References

- **Biham-Shamir.**  
Differential Cryptanalysis of the Data Encryption Standard.  
Springer 1993
- **Matsui.**  
The First Experimental Cryptanalysis of the Data  
Encryption Standard.  
In *CRYPTO 1994*, LNCS 839.
- **Baignères.**  
Quantitative Security of Block Ciphers: Designs and  
Cryptanalysis Tools.  
EPFL PhD Thesis  
<http://library.epfl.ch/theses/?nr=4208>
- **Vaudenay.**  
Decorrelation: A Theory for Block Cipher Security.  
Journal of Cryptology vol. 16, 2003.

## Train Yourself

- DP and LP: midterm exam 2008–09 ex2
- distinguishers:
  - midterm exam 2008–09 ex4
  - final exam 2008–09 ex1
  - midterm exam 2013–14 ex3
  - midterm exam 2014–15 ex2 ( $L_1$  norm and KL divergence)
  - midterm exam 2015–16 ex3 (using Hellinger distance)
  - midterm exam 2016–17 ex4 (distinguishing 3-round Feistel)
  - midterm exam 2017–18 ex3 (number of samples)
- non-linearity of functions: midterm exam 2010–11 ex3
- two-time pad: midterm exam 2010–11 ex4
- SQUASH0: final exam 2010–11 ex3
- biases in RC4:
  - midterm exam 2009–10 ex1
  - midterm exam 2012–13 ex1
- multiple encryption: midterm exam 2012–13 ex2
- AES on 4 rounds: final exam 2016–17 ex1











## 1 The Cryptographic Zoo

## 2 Cryptographic Security Models

## 3 Cryptanalysis (Public-Key)

## 4 The Power of Interaction

## 5 Cryptanalysis (Conventional)

## 6 Proving Security

## 1 The Cryptographic Zoo

## 2 Cryptographic Security Models

## 3 Cryptanalysis (Public-Key)

## 4 The Power of Interaction

## 5 Cryptanalysis (Conventional)

## 6 Proving Security

- The Random Oracle Model
- Hybrid ElGamal
- The Fujisaki-Okamoto Transform

# The Random Oracle Model

- participants can query a public oracle which, upon a fresh query  $x$  will answer with a (long enough) bitstring whose bits are i.i.d. and uniformly distributed  
if  $x$  is queried again, the answer will be the same  
→ random oracles implement a **random function  $H$**
- “long enough” means: enough for the use of a polynomial-time Turing machines  
if algorithms only use the first  $\ell$  bits we can assume that  $H$  yields  $\ell$ -bit results
- adversary does not see queries by honest participants
- simulators/extractors may simulate the random oracle, or just look at queries and answers  
for the simulation to work, they should simulate so that the distribution of outputs are indistinguishable from the correct one

# Full-Domain Hash (FDH) Signature

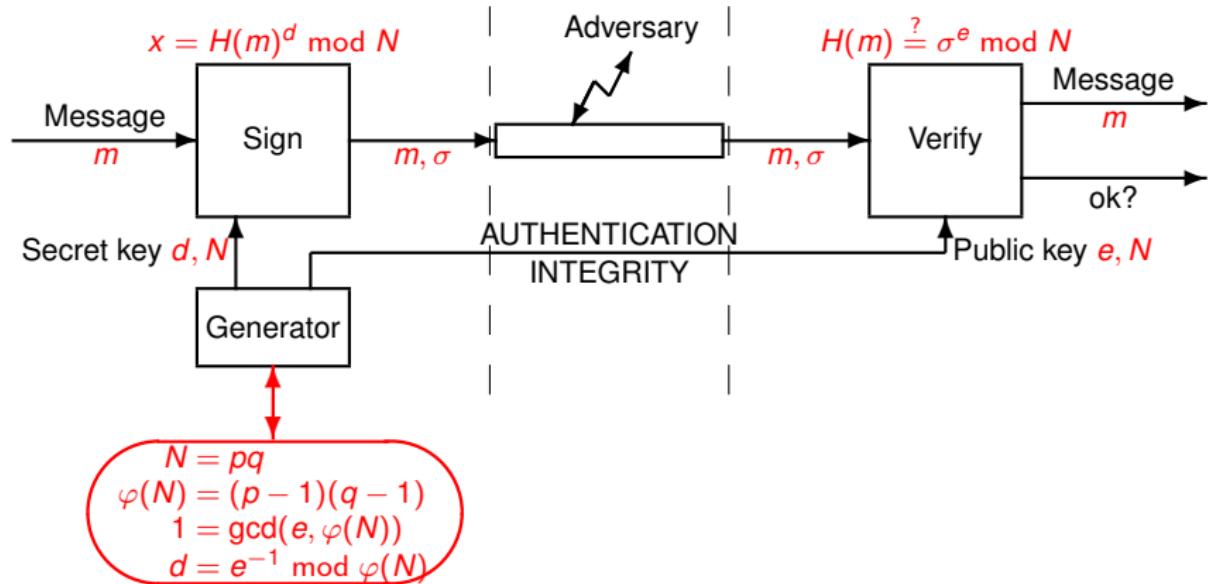
- consider the RSA homomorphic trapdoor permutation over  $\mathbf{Z}_N^*$
- let  $H$  be a random oracle hashing onto  $\mathbf{Z}_N^*$
- $\text{Sign}_{d,N}(m) = (H(m))^d \bmod N$
- $\text{Verify}_{e,N}(m, \sigma) \iff \sigma^e \bmod N = H(m)$

## Theorem

*In the random oracle model, an EF-CMA adversary with time complexity  $t$ ,  $q_S$  chosen messages,  $q_H$  hash queries, and probability of success  $\varepsilon$  can be transformed into an RSA decryption algorithm with complexity  $t + (q_S + q_H)\mathcal{O}(T_e)$  and probability of success  $\varepsilon^{\frac{\exp(-1)}{q_S+1}}$ , where  $T_e$  is the complexity of an RSA encryption.*

$$(\exp(x) = 2.71 \dots^x)$$

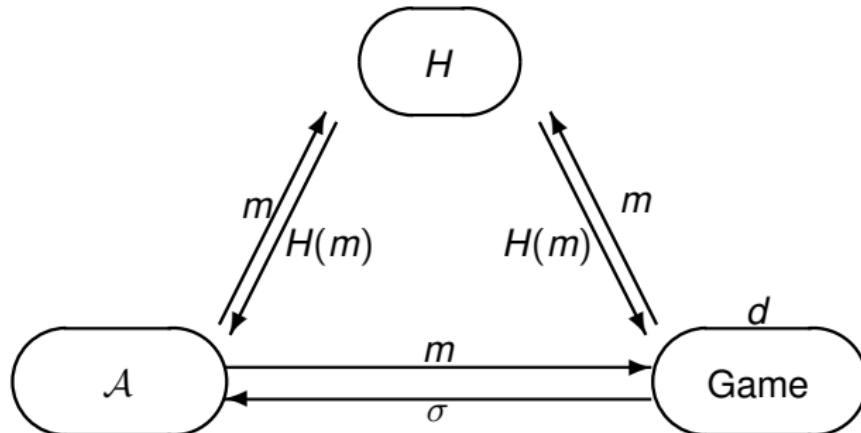
# Full-Domain Hash (FDH) Signature



## Proof — i

Let  $\varepsilon = \Pr[\mathcal{A} \text{ wins}]$ .

- in the EF-CMA game, the challenger selects pk and sk and sends pk to  $\mathcal{A}$ , then answers to any signature query
- example: if  $m$  is a signature query, the challenger queries  $m$  to  $H$ , gets  $h$ , and answers  $h^d \bmod N$  to  $\mathcal{A}$

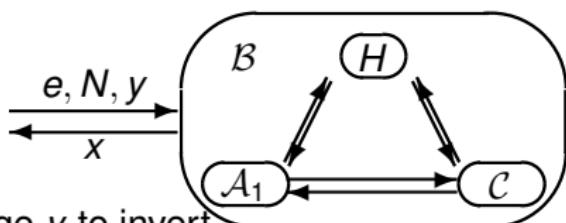


## Proof — ii

- define  $\mathcal{A}_1$  as follows
  - simulate  $\mathcal{A}$  until  $\mathcal{A}$  yields its final  $(m, \sigma)$  forgery
  - if  $m$  was queried to the challenger, abort  
    →  $m$  not queried to  $H$  by challenger
  - if  $m$  was not queried to  $H$  by  $\mathcal{A}$ , query it  
    in any case, set  $h = H(m)$
  - if  $\sigma^e \bmod N \neq h$ , abort
  - yield  $(m, \sigma)$
- we obtain: a new EF-CMA adversary  $\mathcal{A}_1$  with similar complexity and success probability  $\varepsilon$ , who either aborts or yields a final result  $(m, \sigma)$  and wins, and who always queries  $m$  (by  $\mathcal{A}$ ) to  $H$

## Proof — iii

- define  $\mathcal{B}$  as follows
  - get  $e, N$  and the challenge  $y$  to invert
  - set an optimal probability  $p$
  - run  $\mathcal{A}_1$  and simulate  $H$  and the challenger
  - upon a query  $m$  to  $H$ , if  $m$  was queried before, give the same answer; otherwise, pick  $r \in_U \mathbb{Z}_N^*$ , flip a biased coin  $b$  and gives  $y^b r^e \bmod N$  where  $\Pr[b = 1] = p$   
→ perfect simulation since  $y^b r^e \bmod N$  is uniform
  - upon a signature query  $m$ , query  $m$  to  $H$ . If the answer is of type  $r^e \bmod N$ , answer by  $r$ , otherwise, abort
  - when  $\mathcal{A}_1$  finished and output  $(m, \sigma)$ , if query  $m$  to  $H$  produced the answer of type  $y r^e \bmod N$ , yield  $\sigma/r \bmod N$ ; otherwise, abort  
→ if  $H(m) = y r^e \bmod N$ , we have  $(\sigma/r)^e \bmod N = y$
- we obtain: an inverter  $\mathcal{B}$  with similar complexity as  $\mathcal{A}$  who succeeds when the  $m$  query to  $H$  is of type  $y r^e \bmod N$  and sign queries are of type  $r^e \bmod N$



## Proof — iv

- the probability of success is  $p(1 - p)^{q_S} \varepsilon$
- the optimal value for  $p$  is  $p = \frac{1}{q_S + 1}$
- the probability of success is thus

$$\frac{1}{q_S + 1} \left(1 - \frac{1}{q_S + 1}\right)^{q_S} \varepsilon \geq \frac{\exp(-1)}{q_S + 1} \varepsilon$$

□

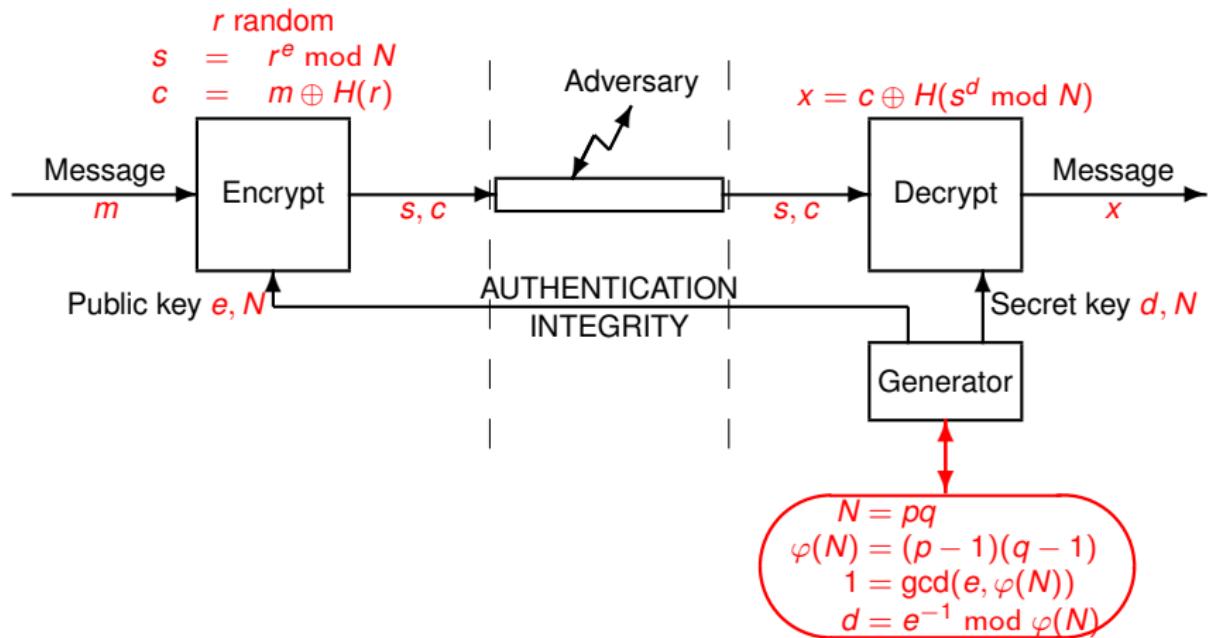
# Semantically Secure Encryption

- consider the RSA trapdoor permutation
- define  $\text{Enc}_{e,N}(m; r) = (r^e \bmod N, m \oplus H(r))$
- $\text{Dec}_{d,N}(s, c) = c \oplus H(s^d \bmod N)$

## Theorem

*In the random oracle model, an IND-CPA adversary with time complexity  $t$ ,  $q$  queries to  $H$ , and advantage  $\varepsilon$  can be transformed into an RSA decryption algorithm with complexity  $t + qO(T_e)$ , and probability of success at least  $\varepsilon$ , where  $T_e$  is the complexity of an RSA encryption.*

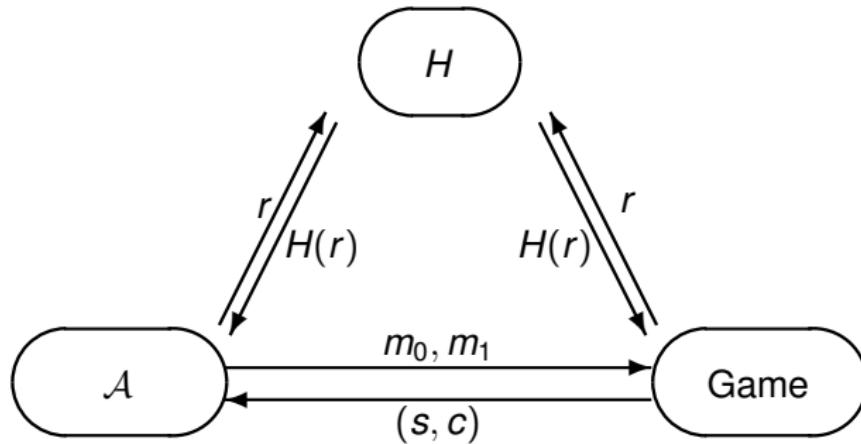
# Hybrid RSA Encryption with a Random Oracle



## Proof — i

Let  $\varepsilon = \text{Adv.}$

- in the IND-CPA game, the challenger selects  $e, N$  and  $d$  and sends  $e, N$  to  $\mathcal{A}$ , then encrypts  $m_b$  into  $(s, c)$
- adversary makes a guess  $b'$  for  $b$



## Proof — ii

- let  $E$  be the event that there is a query  $r$  from  $\mathcal{A}$  to  $H$  is such that  $r^e \bmod N = s$
- define another IND-CPA adversary  $\mathcal{A}_1$  as follows
  - simulate  $\mathcal{A}$  until it terminates
  - answer  $b'$  like  $\mathcal{A}$  if  $\neg E$  holds
  - if  $E$  holds for any query  $r$  by  $\mathcal{A}$ , compute  $c \oplus H(r)$  and answer by  $b'$  such that  $m_{b'} = c \oplus H(r)$
- $\mathcal{A}_1$  always wins if  $E$  holds:  $\text{Adv}(\mathcal{A}_1|E) = 1$
- $\text{Adv}(\mathcal{A}_1|\neg E) = \text{Adv}(\mathcal{A}|\neg E) = 0$   
Indeed, if a random  $H(r)$  is used to release  $c = m_b \oplus H(r)$  but is never used otherwise, then  $b$  is independent from the view of  $\mathcal{A}_1$
- so,  $\text{Adv}(\mathcal{A}_1) = \Pr[E]$
- $\varepsilon = \text{Adv}(\mathcal{A}) \leq \text{Adv}(\mathcal{A}_1) = \Pr[E]$

## Proof — iii

- define  $\mathcal{B}$  as follows
  - receive  $e, N$  and target  $y$  to invert, pick  $r_0 \in \mathbf{Z}_N^*$
  - simulate  $\mathcal{A}_1, H$ , and challenger with key  $(e, N)$
  - upon query  $r$  to  $H$ , if  $(r/r_0)^e \bmod N = y$ , stop and yield  $r/r_0 \bmod N$  as the  $\mathcal{B}$  output, otherwise, simulate  $H$  naturally
  - upon query  $m_0, m_1$  to challenger, pick  $c$  of same length, and answer by  $s, c$  with  $s = yr_0^e \bmod N$
  - if  $\mathcal{A}_1$  terminates, abort ( $\mathcal{B}$  does not succeed)
- simulation is perfect, and  $\mathcal{B}$  succeeds iff  $E$  holds:  
 $\Pr[E] = \Pr[\mathcal{B} \text{ wins}]$
- therefore  $\varepsilon \leq \Pr[E] = \Pr[\mathcal{B} \text{ wins}]$  □

# Fiat-Shamir Signature Paradigm

## Signature from a $\Sigma$ -Protocol

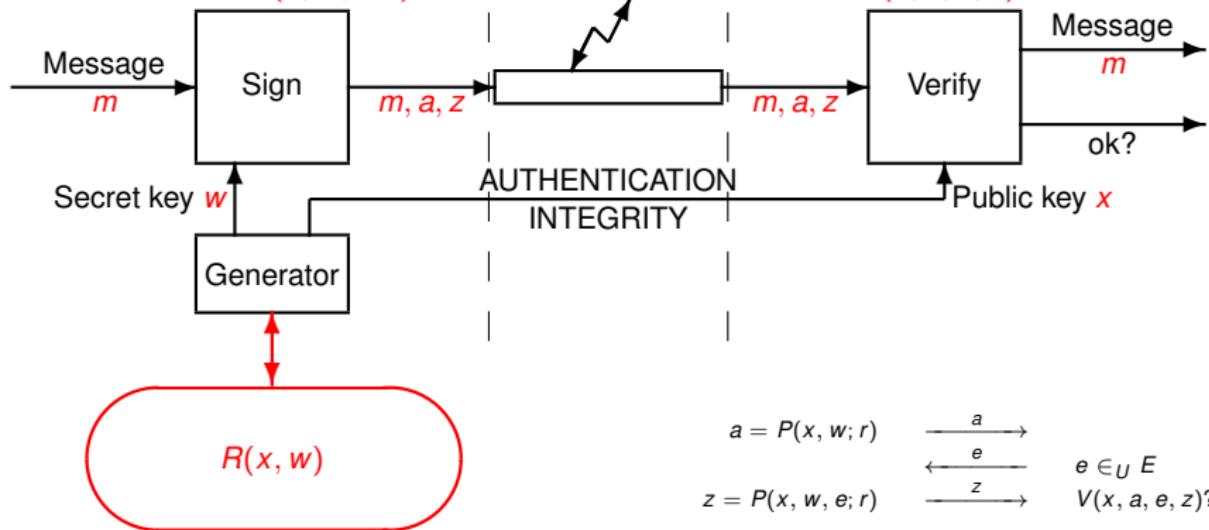
- $\Sigma$ -protocol:  $R, \mathcal{P}, V, \mathcal{E}, \mathcal{S}$ , set of challenges  $E$
- public key:  $x$
- secret key:  $w$  such that  $R(x, w)$
- signature: pick  $r$ , set  $a = \mathcal{P}(x, w; r)$ ,  
 $e = H(\text{message} \| a) \in E$ ,  $z = \mathcal{P}(x, w, e; r)$   
signature is  $(a, z)$
- verification:  $V(x, a, H(\text{message} \| a), z)$

note: the  $x$  missing in  $e = H(\text{message} \| x \| a)$  is an original mistake!

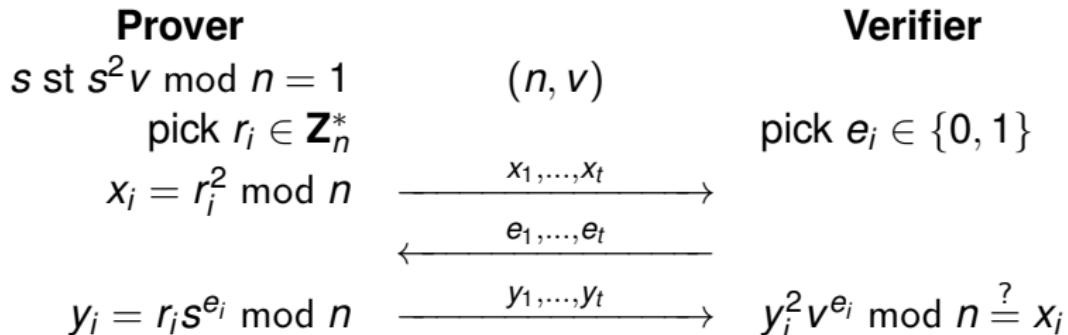
# Fiat-Shamir Signature Paradigm

$$\begin{aligned} a &= P(x, w; r) \\ e &= H(m, a) \\ z &= P(x, w, e; r) \end{aligned}$$

$$\begin{aligned} e &= H(m, a) \\ V(x, a, e, z)? \end{aligned}$$

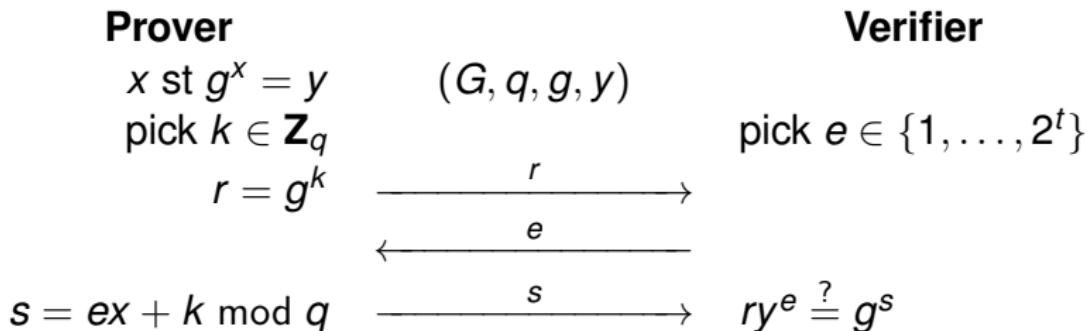


# Fiat-Shamir Signature



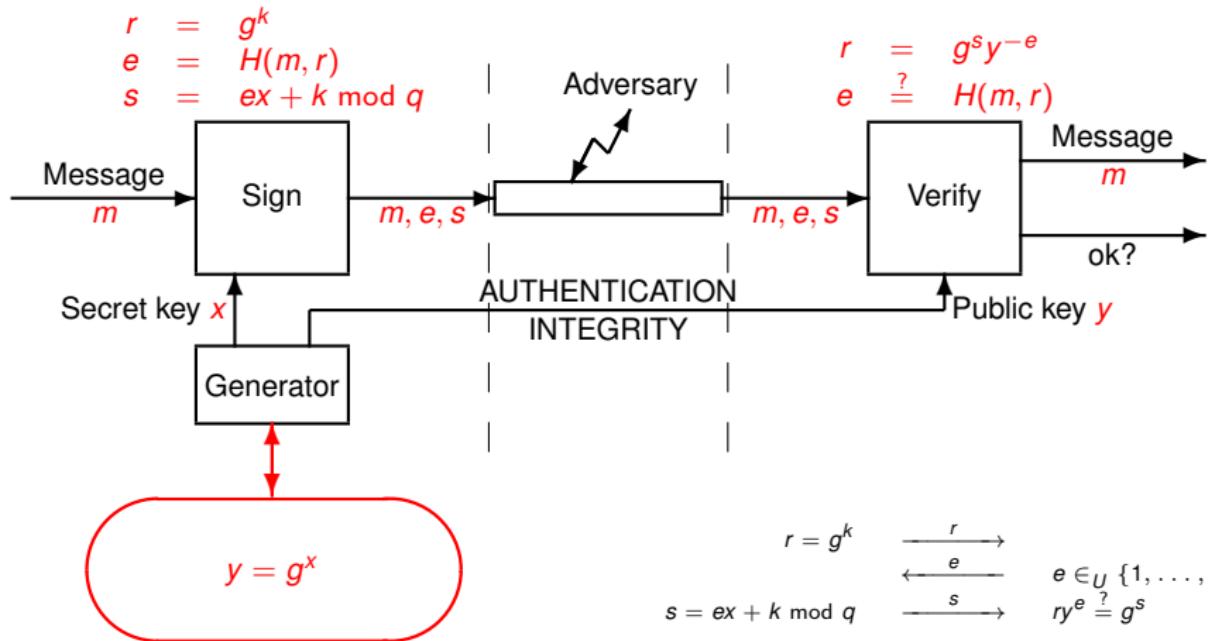
- domain parameter:  $n$
- public key:  $v$
- secret key:  $s$  such that  $s^2 v \bmod n = 1$
- signature: pick  $\vec{r}$ , set  $x_i = r_i^2 \bmod n$ ,  $\vec{e} = H(\text{message} \parallel \vec{x})$ ,  
 $y_i = r_i s^{e_i} \bmod n$   
signature is  $(\vec{x}, \vec{y})$
- verification:  $y_1^2 v^{e_1} \bmod n = x_1, \dots, y_t^2 v^{e_t} \bmod n = x_t$  with  
 $\vec{e} = H(\text{message} \parallel \vec{x})$

# Schnorr Signature



- domain parameter:  $G, q, g$
- public key:  $y$
- secret key:  $x$  such that  $g^x = y$
- signature: pick  $k$ , set  $r = g^k$ ,  $e = H(\text{message} \| r)$ ,  
 $s = ex + k \bmod q$   
signature is  $(e, s)$  (equivalent to  $(r, s)$  by  $r = y^{-e}g^s$ )
- verification:  $e = H(\text{message} \| y^{-e}g^s)$  (equivalent to  
 $ry^{H(\text{message} \| r)} = g^s$ )

# Schnorr Signature



# Fiat-Shamir Signature Paradigm in ROM

## Theorem

*Given a relation  $R$  s.t. it is hard to find witnesses and a  $\Sigma$ -protocol for its language s.t.  $1/\#E = \text{negl}$ , the signature scheme obtained by the Fiat-Shamir construction **using a random oracle** is EF-CMA-secure (existentially unforgeable under chosen message attacks).*

# Removing Chosen Messages

## Lemma

*Given a relation  $R$  s.t. it is hard to find witnesses and a  $\Sigma$ -protocol for its language s.t.  $1/\#E = \text{negl}$ , we consider the signature scheme obtained by the Fiat-Shamir construction using a random oracle.*

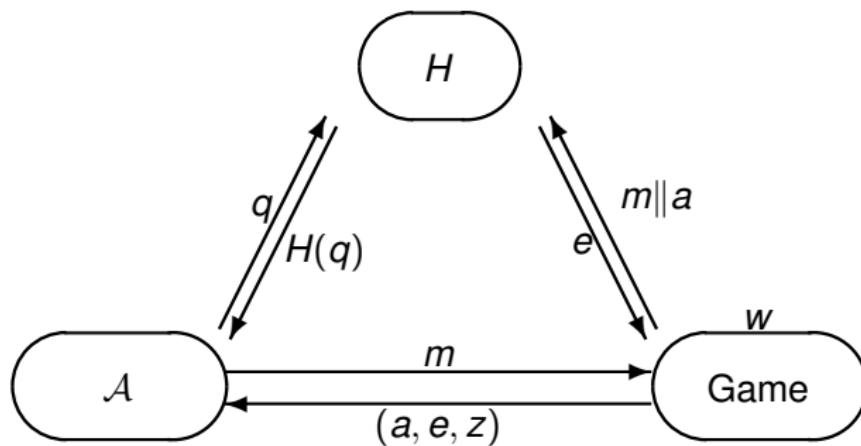
*There is a compiler which can transform an adversary  $\mathcal{A}$  succeeding the EF-CMA game into an adversary  $\mathcal{A}'$  playing the EF-0MA game (no message attack) such that the complexity of  $\mathcal{A}'$  is the one by  $\mathcal{A}$  multiplied by some polynomial and*

$$\Pr[\mathcal{A}' \text{ wins}] = \Pr[\mathcal{A} \text{ wins}] - \text{negl}$$

## Proof of Lemma — i

Let  $\varepsilon = \Pr[\mathcal{A} \text{ wins}]$ .

- change the game s.t. challenger and  $\mathcal{A}$  produce  $e$  as well
- in the EF-CMA game, the challenger selects  $x$  and  $w$  and sends  $x$  to  $\mathcal{A}$ , then answers to any signature query
- example: if  $m$  is a signature query, the challenger picks  $r$ , computes  $a = \mathcal{P}(x, w; r)$ , queries  $m||a$  to  $H$ , gets  $e$ , computes  $z = \mathcal{P}(x, w, e; r)$ , and sends  $(a, e, z)$  to  $\mathcal{A}$



## Proof of Lemma — ii

- define  $\mathcal{A}_1$  as follows
  - simulate  $\mathcal{A}$  until  $\mathcal{A}$  yields its final  $(m, a, e, z)$  forgery
  - if  $m$  was queried to the challenger, abort  
→ no query of form  $m\|a'$  from challenger to  $H$
  - if  $m\|a$  was not queried to  $H$  by  $\mathcal{A}$ , query it
  - if  $\neg V(x, a, e, z)$  or  $e \neq H(m\|a)$ , abort
  - yield  $(m, a, e, z)$
- we obtain: a new EF-CMA adversary  $\mathcal{A}_1$  with similar complexity and success probability  $\varepsilon$ , who either aborts or yields a final result  $(m, a, e, z)$  and wins, and who always queries  $(m\|a)$  to  $H$

## Proof of Lemma — iii

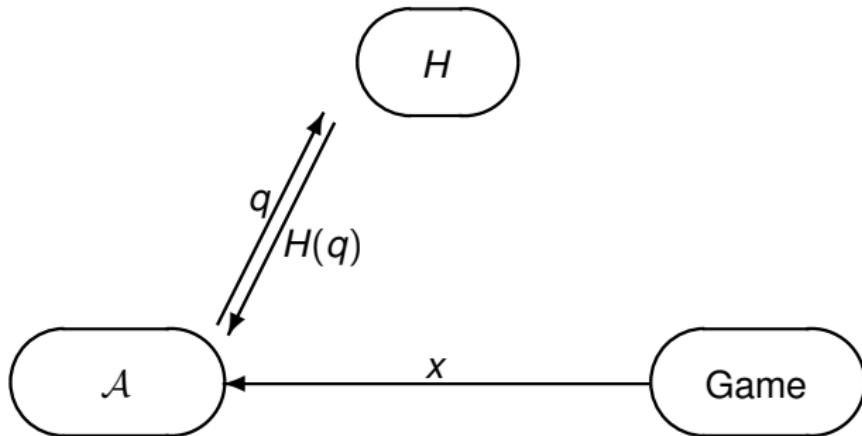
- define  $\mathcal{A}_2$  as follows
  - simulate  $\mathcal{A}_1$  and make a list of all  $(q, H(q))$  queries (queries  $m \parallel a \mapsto e$  by the challenger are deduced from  $x$ ,  $m$ , and  $(a, e, z)$ )
  - if adversary tries to repeat a query which was done before, just take its answer from the list and avoid the query
  - if challenger does a query which was done before (let  $\varepsilon'$  be the probability this happens), abort
- we obtain: a new EF-CMA adversary  $\mathcal{A}_2$  with similar complexity and success probability  $\varepsilon - \varepsilon'$  such that the EF-CMA game never repeat any query
- remark: #queries is polynomial so  $\varepsilon' \leq \text{poly} \times \max_a p_a$  with  $p_a = \Pr[\mathcal{P}(x, w; r) = a]$   
the algorithm running  $\mathcal{S}(x, e; r)$  and  $\mathcal{S}(x, e'; r')$  with random  $e, e', r, r'$  yields  $a$  twice with probability  $p_a^2$  so makes  $\mathcal{E}$  extract  $w$  with probability  $p_a^2(1 - 1/\#E)$   
so,  $p_a$  is negligible  
thus,  $\varepsilon'$  is negligible

## Proof of Lemma — iv

- define  $\mathcal{A}'$  as follows
  - simulate  $\mathcal{A}_2$  until a query  $m$  to the challenger is made
  - upon query  $m$ ,  $\mathcal{A}'$  picks  $r, e$ , and computes  $(a, e, z) = S(x, e; r)$ 
    - if  $m||a$  has been queried before, the simulation fails
    - otherwise, the  $H$  table is augmented with  $m||a \mapsto e$
- we obtain: an EF-0MA adversary  $\mathcal{A}'$  with similar complexity and success probability  $\varepsilon - \varepsilon'$



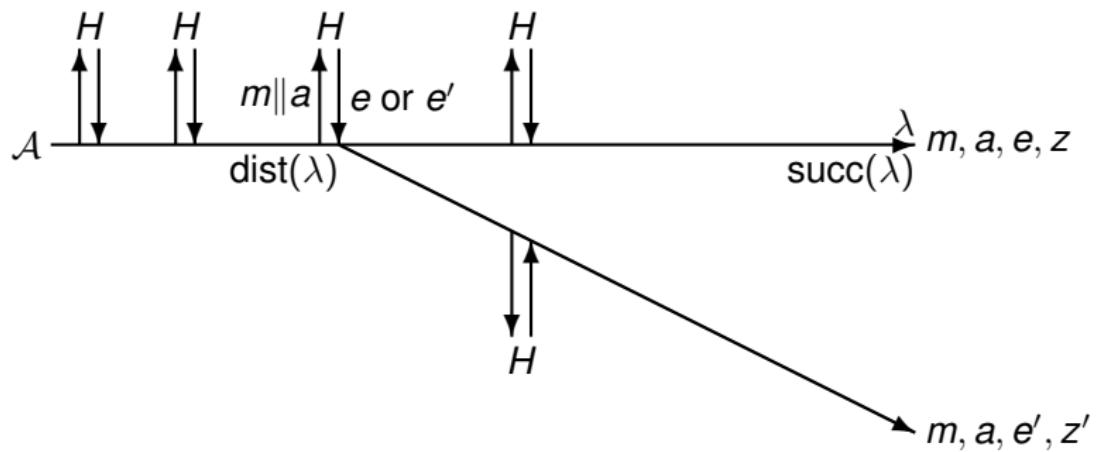
## Proof of Theorem — i



define  $\mathcal{B}$  as follows

- simulate  $\mathcal{A}$  with initial  $x$  from challenger, simulate  $H$  to  $\mathcal{A}$
- if  $\mathcal{A}$  does not output any  $(m, a, e, z)$ , abort; otherwise, run  $\mathcal{A}$  again with same random coins until  $m\|a$  is queried to  $H$
- pick a fresh  $e'$  to be answered to  $\mathcal{A}$  and continue the simulation
- if  $\mathcal{A}$  does not output any  $(m, a, e', z')$ , abort; otherwise, get two forgeries  $(a, e, z)$  and  $(a, e', z')$  with same  $a$  so extract  
 $w = \mathcal{E}(x, a, e, z, e', z')$

## Proof of Theorem — ii



## Proof of Theorem — iii

we build the tree of  $\mathcal{A}$  executions with same random tape based on answers from  $H$  (each node  $\nu$  corresponds to a query, each leaf  $\lambda$  corresponds to a termination)

- we consider the random descent following the  $H$  simulation which leads to a random leaf  $X$
- let  $\text{succ}(\lambda)$  be true if  $\lambda$  yields  $(m, a, e, z)$  and false otherwise
- if  $\lambda$  yields  $(m, a, e, z)$ , let  $\text{dist}(\lambda)$  be the ancestor of  $\lambda$  corresponding to the  $m||a$  query, otherwise, let  $\text{dist}(\lambda) = \lambda$
- we have  $\Pr[\text{succ}(X)] = \varepsilon$  and  $E(\text{depth}(X)) = \text{poly}$
- let  $\text{visit}(\nu)$  the event that  $X$  has  $\nu$  as an ancestor  
let  $f(\nu) = \Pr[\text{succ}(X), \text{dist}(X) = \nu | \text{visit}(\nu)]$

we obtain: a witness extractor  $\mathcal{B}$  with similar complexity who succeeds if  $\text{succ}(X)$ ,  $e \neq e'$ , and the second run succeeds on  $X'$  such that  $\text{dist}(X') = \text{dist}(X)$ .

since  $\Pr[e = e'] = \text{negl}$ , the success probability is greater than  $E(f(\text{dist}(X))) - \text{negl}$

## Proof of Theorem — iv

recap:

- we have a tree with a predicate  $\text{succ}$  on leaves and a function  $\text{dist}$  mapping a leaf  $\lambda$  to one ancestor if  $\text{succ}(\lambda)$  and to itself otherwise
- we have a distribution on leaves such that the depth of a random leaf  $X$  is polynomial and  $\Pr[\text{succ}] = \varepsilon'$
- we define  $f(\nu) = \Pr[\text{succ}(X), \text{dist}(X) = \nu | \text{visit}(\nu)]$  where  $\text{visit}(\nu)$  is the event that  $\nu$  is an ancestor of  $X$
- if  $E(f(\text{dist}(X)))$  is negligible, so is  $\varepsilon'$

we conclude using the Forking Lemma



# Forking Lemma

## Lemma (Forking Lemma)

- We consider a finite tree and a mapping  $\text{dist}$  which maps any leaf  $\lambda$  to one of its ancestors  $\text{dist}(\lambda)$ . We call it a distinguished ancestor.
- We assume we are given a distribution which defines a random leaf  $X$ . We let  $\text{visit}(\nu)$  be the event that the descent goes through  $\nu$ , i.e. that  $\nu$  is an ancestor of  $\lambda$ .
- We let  $\text{succ}(\lambda)$  be true iff  $\text{dist}(\lambda) \neq \lambda$ . When it occurs we say that  $\lambda$  is successful.
- We let  $p = \Pr[\text{succ}(X)]$ ,  $\bar{d} = E(\text{depth}(X))$ , and  $f(\nu) = \Pr[\text{succ}(X) \text{ and } \text{dist}(X) = \nu | \text{visit}(\nu)]$ .

We have

$$\Pr \left[ f(\text{dist}(X)) > \frac{p}{2\bar{d}} \middle| \text{succ}(X) \right] \geq \frac{1}{2}$$

## Consequence

$$\begin{aligned} E(f(\text{dist}(X))) &= \int_0^1 \Pr[f(\text{dist}(X)) \geq t] dt \\ &\geq \int_0^1 \Pr[f(\text{dist}(X)) \geq t, \text{succ}(X)] dt \\ &= p \int_0^1 \Pr[f(\text{dist}(X)) \geq t | \text{succ}(X)] dt \\ &\geq p \int_0^1 \frac{1}{2} \cdot \mathbf{1}_{t \leq \frac{p}{d}} dt \\ &= \frac{p^2}{4d} \end{aligned}$$

$$\text{so, } p \leq \sqrt{4\bar{d}E(f(\text{dist}(X)))}$$

If  $E(f(\text{dist}(X)))$  is negligible,  $p$  is negligible as well, which completes the proof of the Theorem.

# Proof of Forking Lemma

- we have  $\Pr[\text{dist}(X) = \nu | \text{succ}(X)] = f(\nu) \frac{\Pr[\text{visit}(\nu)]}{p}$
- let Bad be the set of  $\nu$ 's s.t.  $f(\nu) \leq \frac{p}{2\bar{d}}$   
we have

$$\begin{aligned}\Pr[\text{dist}(X) \in \text{Bad} | \text{succ}(X)] &= \sum_{\nu \in \text{Bad}} f(\nu) \frac{\Pr[\text{visit}(\nu)]}{p} \\ &\leq \frac{\sum_{\nu} \Pr[\text{visit}(\nu)]}{2\bar{d}} \\ &\leq \frac{1}{2}\end{aligned}$$

- so,  $\Pr[\text{dist}(X) \notin \text{Bad} | \text{succ}(X)] \geq \frac{1}{2}$

□

# Controversy about the Random Oracle Model

- random oracle are idealizations of practical functions
- in practice, no hash function is a random oracle
- we may have scheme secure in the random oracle model but insecure for *any* practical instantiation of the random oracle
- security in this model are still better than nothing
- one should interpret these security results with great care

# Insecure ROM-Secure Signature Scheme

Canetti–Goldreich–Halevi 1998

- consider the FDH ROM-secure digital signature scheme

$$\text{Sign}_{d,N}^0(m) = (H(m))^d \bmod N$$

- construct another ROM-secure digital signature scheme:
  - message semantics: interpret  $m$  as an algorithm implementing a partial function  $h_m$  within a bounded time  $\tau$
  - $\text{Sign}_{d,N}(m)$ : pick  $r$ ; if  $H(r) = h_m(r)$  then output  $d$ ; otherwise output  $\text{Sign}_{d,N}^0(m)$
  - $\text{Verify}_{e,N}(m, \sigma) \iff m^{e\sigma} \bmod N = m$  or  $\sigma^e \bmod N = H(m)$
- this is ROM-secure as well
- if we replace  $H$  by a function which can be implemented by a code  $m$ , the chosen message  $m$  will leak the secret key!

## 1 The Cryptographic Zoo

## 2 Cryptographic Security Models

## 3 Cryptanalysis (Public-Key)

## 4 The Power of Interaction

## 5 Cryptanalysis (Conventional)

## 6 Proving Security

- The Random Oracle Model
- Hybrid ElGamal
- The Fujisaki-Okamoto Transform

# Hybrid ElGamal Cryptosystem using the Leftover Hash Lemma

take a group with a generator  $g$  of order  $q$

**key generation:** pick  $x \in_U \mathbb{Z}_q$ , set  $y = g^x$

**message space:**  $M \in \{0, 1\}^m$

**encryption:**  $\text{Enc}_y(M; r, n) = (g^r, M \oplus h_n(y^r), n)$

**decryption:**  $\text{Dec}_x(u, v, n) = v \oplus h_n(u^x)$

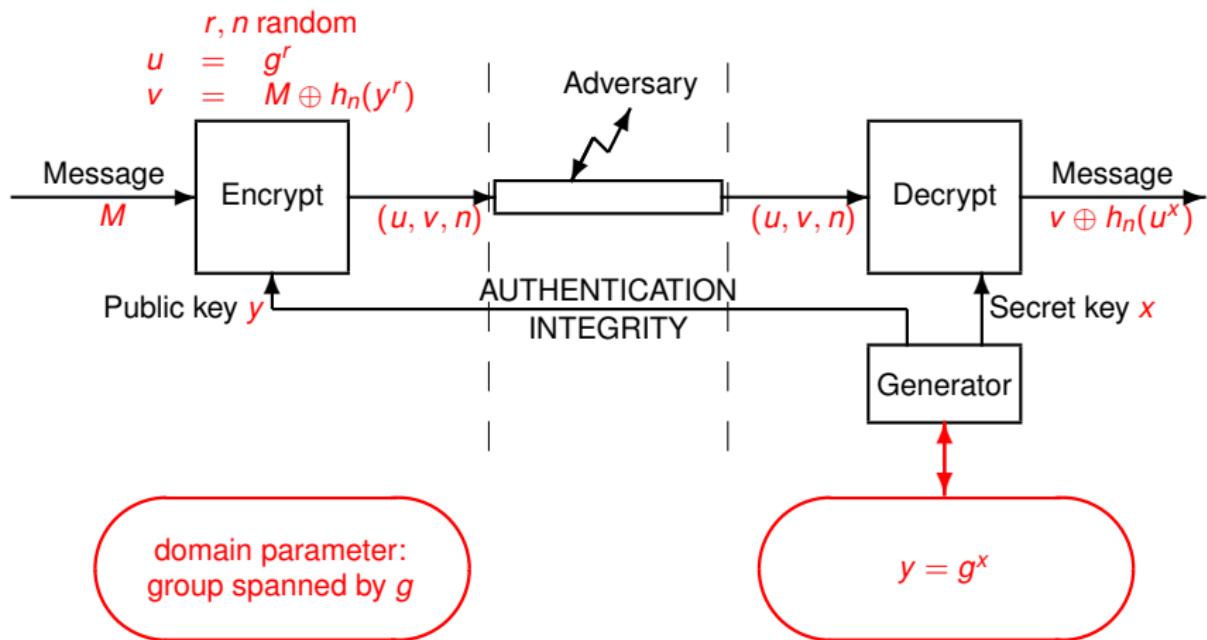
where  $(h_n)_n$  is a family of universal hash functions from  $G \supseteq \langle g \rangle$  to  $\{0, 1\}^m$

leftover hash Lemma: if  $\varepsilon \geq 2^{\frac{m}{2}} / \sqrt{q}$  then  $(h_n(g^s), n)$  is  $\varepsilon$ -indistinguishable from  $(u, n)$  where  $u \in_U \{0, 1\}^m$

## Theorem

If  $2^{\frac{m}{2}} / \sqrt{q} = \text{negl}$  and under the DDH assumption, the above is IND-CPA-secure.

# Hybrid ElGamal Cryptosystem



# Leftover Hash Lemma

- min-entropy:

$$H_{\infty}(X) = - \log_2 \max_x \Pr[X = x]$$

- universal hash function: many probabilistic functions, combinatorial function

$$\forall x \neq x' \quad \Pr_N[h_N(x) = h_N(x')] = \frac{1}{\#\text{range}}$$

where  $\#\text{range} = 2^m$  is the size of the output domain of  $h$  and  $N$  is uniformly distributed

## Lemma (Impagliazzo-Levin-Luby 1989)

If  $m \leq H_{\infty}(X) - 2 \log_2 \frac{1}{\varepsilon}$  and  $h$  is a universal hash function with a range of size  $2^m$  then  $(h_N(X), N)$  and  $(U, N)$  have distributions which are  $\frac{\varepsilon}{2}$ -indistinguishable.

$X, N, U$  are independent.

$N$  and  $U$  are uniformly distributed.

example:  $H_{\infty}(g^s) = \log_2 q$

## Proof

let  $P_0 = (h_N(X), N)$ ,  $P_1 = (U, N)$ , compute the Euclidean distance:

$$\begin{aligned}\|P_1 - P_0\|_2^2 &= \sum_{k,n} \left( \Pr_{X,N}[h_n(X) = k, N = n] - \frac{1}{2^m \#\mathcal{N}} \right)^2 \\ &= \left( \sum_{k,n} \Pr_{X,N}[h_n(X) = k, N = n]^2 \right) - \frac{1}{2^m \#\mathcal{N}} \\ &= \Pr_{X,X',N,N'}[h_N(X) = h_{N'}(X'), N = N'] - \frac{1}{2^m \#\mathcal{N}} \\ &= \frac{1}{\#\mathcal{N}} \sum_{x,x'} \Pr[X = x, X' = x', h_N(x) = h_N(x')] - \frac{1}{2^m \#\mathcal{N}} \\ &= \frac{1 - 2^{-m}}{\#\mathcal{N}} \sum_x \Pr[X = x]^2 \quad (\text{split } x = x' \text{ and } x \neq x') \\ &\leq \frac{1 - 2^{-m}}{\#\mathcal{N}} \max_x \Pr[x] \leq \frac{1 - 2^{-m}}{\#\mathcal{N}} 2^{-H_\infty(X)} \leq \frac{1}{2^m \#\mathcal{N}} \varepsilon^2\end{aligned}$$

we then use

$$d(P_0, P_1) = \frac{1}{2} \|P_0 - P_1\|_1 \leq \frac{1}{2} \|P_0 - P_1\|_2 \sqrt{\#\text{domain}} \leq \frac{\varepsilon}{2}$$

□

# Transitions — i

game  $\Gamma_0^b$ :

- 1: run key generation and get  $y$
- 2: pick  $\rho$  and set view  $= (y; \rho)$
- 3: run  $\mathcal{A}(\text{view}) = (m_0, m_1)$
- 4: pick  $r, u \leftarrow g^r$
- 5: pick  $n, v \leftarrow m_b \oplus h_n(y^r)$
- 6: set view  $= (y, u, v, n; \rho)$
- 7: run  $\mathcal{A}(\text{view}) = b'$
- 8: **return**  $b'$

↑ bridge

game  $\Gamma_1^b$ :

- 1: pick  $x, y \leftarrow g^x$
- 2: pick  $r, u \leftarrow g^r$
- 3:  $X \leftarrow g^{xr}$  ▷ erase  $x, r$
- 4: pick  $\rho$  and set view  $= (y; \rho)$
- 5: run  $\mathcal{A}(\text{view}) = (m_0, m_1)$
- 6: pick  $n, v \leftarrow m_b \oplus h_n(X)$
- 7: set view  $= (y, u, v, n; \rho)$
- 8: run  $\mathcal{A}(\text{view}) = b'$
- 9: **return**  $b'$

DDH assumption in the group

DDH  
≈

game  $\Gamma_2^b$ :

- 1: pick  $x, y \leftarrow g^x$
- 2: pick  $r, u \leftarrow g^r$
- 3: pick  $s, X \leftarrow g^s$  ▷ erase  $x, r, s$
- 4: pick  $\rho$  and set view  $= (y; \rho)$
- 5: run  $\mathcal{A}(\text{view}) = (m_0, m_1)$
- 6: pick  $n, v \leftarrow m_b \oplus h_n(X)$
- 7: set view  $= (y, u, v, n; \rho)$
- 8: run  $\mathcal{A}(\text{view}) = b'$
- 9: **return**  $b'$

# Transitions — ii

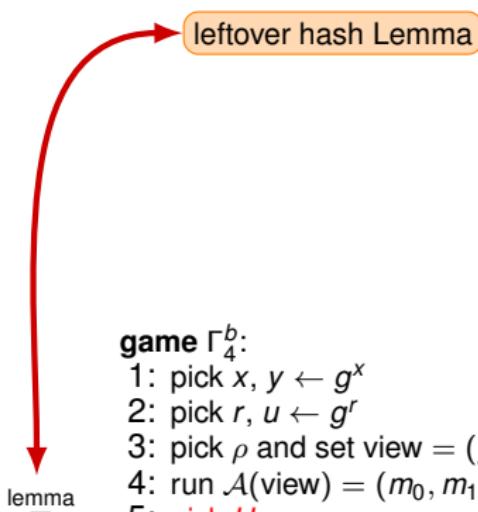
**game**  $\Gamma_2^b$ :

- 1: pick  $x, y \leftarrow g^x$
- 2: pick  $r, u \leftarrow g^r$
- 3: pick  $s, X \leftarrow g^s$
- 4: pick  $\rho$  and set view  $= (y; \rho)$
- 5: run  $\mathcal{A}(\text{view}) = (m_0, m_1)$
- 6: pick  $n, v \leftarrow m_b \oplus h_n(X)$
- 7: set view  $= (y, u, v, n; \rho)$
- 8: run  $\mathcal{A}(\text{view}) = b'$
- 9: **return**  $b'$

↑ bridge

**game**  $\Gamma_3^b$ :

- 1: pick  $x, y \leftarrow g^x$
- 2: pick  $r, u \leftarrow g^r$
- 3: pick  $\rho$  and set view  $= (y; \rho)$
- 4: run  $\mathcal{A}(\text{view}) = (m_0, m_1)$
- 5: **pick**  $s, X \leftarrow g^s$
- 6: **pick**  $n, v_0 \leftarrow h_n(X)$  ▷ erase  $s, X$
- 7:  $v \leftarrow m_b \oplus v_0$
- 8: set view  $= (y, u, v, n; \rho)$
- 9: run  $\mathcal{A}(\text{view}) = b'$
- 10: **return**  $b'$



**game**  $\Gamma_4^b$ :

- 1: pick  $x, y \leftarrow g^x$
- 2: pick  $r, u \leftarrow g^r$
- 3: pick  $\rho$  and set view  $= (y; \rho)$
- 4: run  $\mathcal{A}(\text{view}) = (m_0, m_1)$
- 5: **pick**  $U$
- 6: **pick**  $n, v_0 \leftarrow U$  ▷ erase  $U$
- 7:  $v \leftarrow m_b \oplus v_0$
- 8: set view  $= (y, u, v, n; \rho)$
- 9: run  $\mathcal{A}(\text{view}) = b'$
- 10: **return**  $b'$

# Transitions — iii

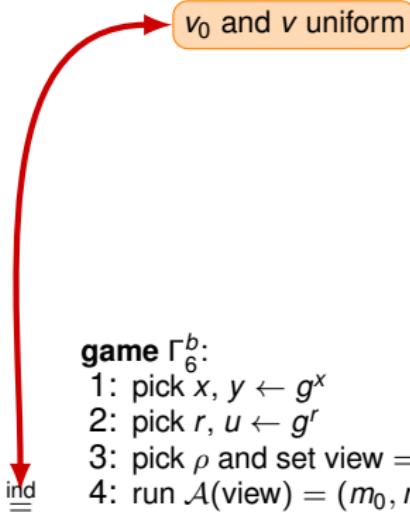
**game**  $\Gamma_4^b$ :

- 1: pick  $x, y \leftarrow g^x$
- 2: pick  $r, u \leftarrow g^r$
- 3: pick  $\rho$  and set view  $= (y; \rho)$
- 4: run  $\mathcal{A}(\text{view}) = (m_0, m_1)$
- 5: pick  $U$
- 6: pick  $n, v_0 \leftarrow U$
- 7:  $v \leftarrow m_b \oplus v_0$
- 8: set view  $= (y, u, v, n; \rho)$
- 9: run  $\mathcal{A}(\text{view}) = b'$
- 10: **return**  $b'$

↑ bridge

**game**  $\Gamma_5^b$ :

- 1: pick  $x, y \leftarrow g^x$
- 2: pick  $r, u \leftarrow g^r$
- 3: pick  $\rho$  and set view  $= (y; \rho)$
- 4: run  $\mathcal{A}(\text{view}) = (m_0, m_1)$
- 5: pick  $n$
- 6: **pick**  $v_0$
- 7:  $v \leftarrow m_b \oplus v_0$     ▷ erase  $v_0$
- 8: set view  $= (y, u, v, n; \rho)$
- 9: run  $\mathcal{A}(\text{view}) = b'$
- 10: **return**  $b'$



**game**  $\Gamma_6^b$ :

- 1: pick  $x, y \leftarrow g^x$
- 2: pick  $r, u \leftarrow g^r$
- 3: pick  $\rho$  and set view  $= (y; \rho)$
- 4: run  $\mathcal{A}(\text{view}) = (m_0, m_1)$
- 5: pick  $n$
- 6: **pick**  $v$
- 7: set view  $= (y, u, v, n; \rho)$
- 8: run  $\mathcal{A}(\text{view}) = b'$
- 9: **return**  $b'$

# Transitions — iv

final step:  $\Gamma_6^0$  and  $\Gamma_6^1$  are identical!

$$\begin{array}{cccccccccccc} \Gamma_0^0 & \xrightarrow{\text{bridge}} & \Gamma_1^0 & \xrightarrow{\text{DDH}} & \Gamma_2^0 & \xrightarrow{\text{bridge}} & \Gamma_3^0 & \xrightarrow{\text{lemma}} & \Gamma_4^0 & \xrightarrow{\text{bridge}} & \Gamma_5^0 & \xrightarrow{\text{domain}} & \Gamma_6^0 \\ \Gamma_0^1 & \xrightarrow{\text{bridge}} & \Gamma_1^1 & \xrightarrow{\text{DDH}} & \Gamma_2^1 & \xrightarrow{\text{bridge}} & \Gamma_3^1 & \xrightarrow{\text{lemma}} & \Gamma_4^1 & \xrightarrow{\text{bridge}} & \Gamma_5^1 & \xrightarrow{\text{domain}} & \Gamma_6^1 \end{array}$$

so,  $\Pr[\Gamma_0^0 = 0] - \Pr[\Gamma_0^1 = 0] \leq 2\text{Adv}_{\text{DDH}} + 2\varepsilon = \text{negl}$

# Hybrid ElGamal Cryptosystem using Random Oracles

take a group with a generator  $g$  of order  $q$

**key generation:** pick  $x \in \mathbf{Z}_q$ , set  $y = g^x$

**message space:**  $M \in \{0, 1\}^m$

**encryption:**  $\text{Enc}_y(M; r) = (g^r, M \oplus H(y^r))$

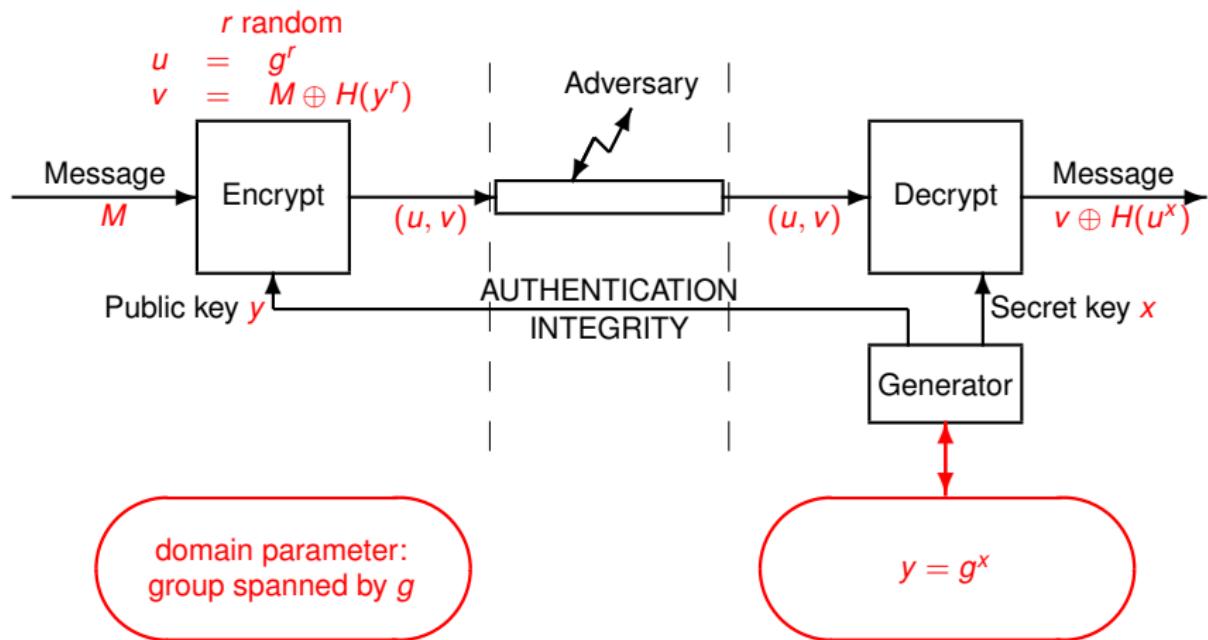
**decryption:**  $\text{Dec}_x(u, v) = v \oplus H(u^x)$

where  $H$  is a random oracle onto  $\{0, 1\}^m$

## Theorem

*Under the DDH assumption, the above is IND-CPA-secure in the random oracle model.*

# Hybrid ElGamal Cryptosystem



# Transitions — i

game  $\Gamma_0^b$ :

- 1: pick  $H$
- 2: run key generation and get  $y$
- 3: pick  $\rho$  and set view  $= (y; \rho)$
- 4: run  $\mathcal{A}^H(\text{view}) = (m_0, m_1)$
- 5: pick  $r, u \leftarrow g^r$
- 6:  $v \leftarrow m_b \oplus H(y^r)$
- 7: set view  $= (y, u, v; \rho)$
- 8: run  $\mathcal{A}^H(\text{view}) = b'$
- 9: return  $b'$

$\Updownarrow$  bridge

game  $\Gamma_1^b$ :

- 1: pick  $x, y \leftarrow g^x$
- 2: pick  $r, u \leftarrow g^r$
- 3:  $X \leftarrow g^{xr}$
- ▷ erase  $x, r$
- 4: pick  $H, \rho$ , set view  $= (y; \rho)$
- 5: run  $\mathcal{A}^H(\text{view}) = (m_0, m_1)$
- 6:  $v \leftarrow m_b \oplus H(X)$
- 7: set view  $= (y, u, v; \rho)$
- 8: run  $\mathcal{A}^H(\text{view}) = b'$
- 9: return  $b'$

DDH assumption in the group



game  $\Gamma_2^b$ :

- 1: pick  $x, y \leftarrow g^x$
- 2: pick  $r, u \leftarrow g^r$
- 3: pick  $s, X \leftarrow g^s$
- ▷ erase  $x, r, s$
- 4: pick  $H, \rho$ , set view  $= (y; \rho)$
- 5: run  $\mathcal{A}^H(\text{view}) = (m_0, m_1)$
- 6:  $v \leftarrow m_b \oplus H(X)$
- 7: set view  $= (y, u, v; \rho)$
- 8: run  $\mathcal{A}^H(\text{view}) = b'$
- 9: return  $b'$

# Transitions — ii

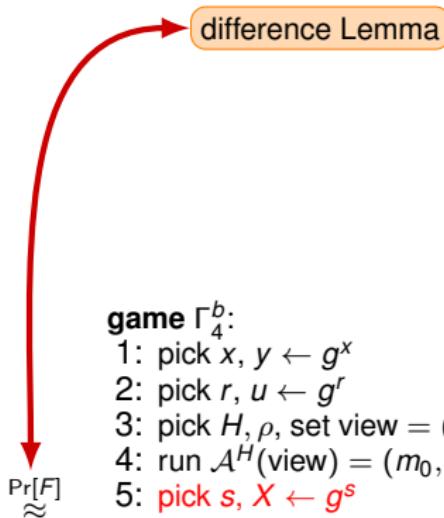
game  $\Gamma_2^b$ :

- 1: pick  $x, y \leftarrow g^x$
- 2: pick  $r, u \leftarrow g^r$
- 3: pick  $s, X \leftarrow g^s$
- 4: pick  $H, \rho$ , set view  $= (y; \rho)$
- 5: run  $\mathcal{A}^H(\text{view}) = (m_0, m_1)$
- 6:  $v \leftarrow m_b \oplus H(X)$
- 7: set view  $= (y, u, v; \rho)$
- 8: run  $\mathcal{A}^H(\text{view}) = b'$
- 9: return  $b'$

↑ bridge

game  $\Gamma_3^b$ :

- 1: pick  $x, y \leftarrow g^x$
- 2: pick  $r, u \leftarrow g^r$
- 3: pick  $H, \rho$ , set view  $= (y; \rho)$
- 4: run  $\mathcal{A}^H(\text{view}) = (m_0, m_1)$
- 5: pick  $s, X \leftarrow g^s$
- 6:  $v_0 \leftarrow H(X)$
- 7:  $v \leftarrow m_b \oplus v_0$
- 8: set view  $= (y, u, v; \rho)$
- 9: run  $\mathcal{A}^H(\text{view}) = b'$
- 10: return  $b'$



game  $\Gamma_4^b$ :

- 1: pick  $x, y \leftarrow g^x$
- 2: pick  $r, u \leftarrow g^r$
- 3: pick  $H, \rho$ , set view  $= (y; \rho)$
- 4: run  $\mathcal{A}^H(\text{view}) = (m_0, m_1)$
- 5: pick  $s, X \leftarrow g^s$
- 6: pick  $v_0$
- 7:  $v \leftarrow m_b \oplus v_0$
- 8: set view  $= (y, u, v; \rho)$
- 9: run  $\mathcal{A}^H(\text{view}) = b'$
- 10: **return 0 if  $\mathcal{A}$  queried  $H(X)$**
- 11: return  $b'$

# Transitions — iii

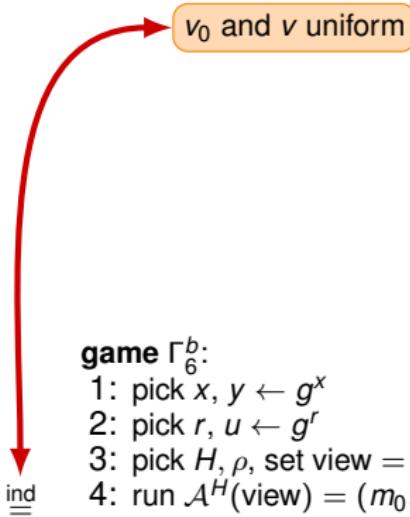
game  $\Gamma_4^b$ :

- 1: pick  $x, y \leftarrow g^x$
- 2: pick  $r, u \leftarrow g^r$
- 3: pick  $H, \rho$ , set view  $= (y; \rho)$
- 4: run  $\mathcal{A}^H(\text{view}) = (m_0, m_1)$
- 5: **pick  $s, X \leftarrow g^s$**
- 6: pick  $v_0$
- 7:  $v \leftarrow m_b \oplus v_0$
- 8: set view  $= (y, u, v; \rho)$
- 9: run  $\mathcal{A}^H(\text{view}) = b'$
- 10: **return 0 if  $\mathcal{A}$  queried  $H(X)$**
- 11: return  $b'$

↑ bridge

game  $\Gamma_5^b$ :

- 1: pick  $x, y \leftarrow g^x$
- 2: pick  $r, u \leftarrow g^r$
- 3: pick  $H, \rho$ , set view  $= (y; \rho)$
- 4: run  $\mathcal{A}^H(\text{view}) = (m_0, m_1)$
- 5: **pick  $v_0, v \leftarrow m_b \oplus v_0$**
- 6: set view  $= (y, u, v; \rho)$
- 7: run  $\mathcal{A}^H(\text{view}) = b'$
- 8: **return 0 if  $\mathcal{A}$  queried  $H(X)$**
- 9: return  $b'$



game  $\Gamma_6^b$ :

- 1: pick  $x, y \leftarrow g^x$
- 2: pick  $r, u \leftarrow g^r$
- 3: pick  $H, \rho$ , set view  $= (y; \rho)$
- 4: run  $\mathcal{A}^H(\text{view}) = (m_0, m_1)$
- 5: **pick  $v$**
- 6: set view  $= (y, u, v; \rho)$
- 7: run  $\mathcal{A}^H(\text{view}) = b'$
- 8: **return 0 if  $\mathcal{A}$  queried  $H(X)$**
- 9: return  $b'$

## Transitions — iv

final step:  $\Gamma_6^0$  and  $\Gamma_6^1$  are identical!

$$\begin{array}{ccccccccccccc} \Gamma_0^0 & \xrightarrow{\text{bridge}} & \Gamma_1^0 & \xrightarrow{\text{DDH}} & \Gamma_2^0 & \xrightarrow{\text{bridge}} & \Gamma_3^0 & \xrightarrow{\text{difference}} & \Gamma_4^0 & \xrightarrow{\text{bridge}} & \Gamma_5^0 & \xrightarrow{\text{domain}} & \Gamma_6^0 \\ \Gamma_0^1 & \xrightarrow{\text{bridge}} & \Gamma_1^1 & \xrightarrow{\text{DDH}} & \Gamma_2^1 & \xrightarrow{\text{bridge}} & \Gamma_3^1 & \xrightarrow{\text{difference}} & \Gamma_4^1 & \xrightarrow{\text{bridge}} & \Gamma_5^1 & \xrightarrow{\text{domain}} & \Gamma_6^1 \end{array}$$

so,  $\Pr[\Gamma_0^0 = 1] - \Pr[\Gamma_0^1 = 1] \leq 2\text{Adv}_{\text{DDH}} + 2\Pr[F] = \text{negl}$   
since  $\Pr[F] \leq \frac{\#\text{queries}}{q}$

## 1 The Cryptographic Zoo

## 2 Cryptographic Security Models

## 3 Cryptanalysis (Public-Key)

## 4 The Power of Interaction

## 5 Cryptanalysis (Conventional)

## 6 Proving Security

- The Random Oracle Model
- Hybrid ElGamal
- The Fujisaki-Okamoto Transform

# Fujisaki-Okamoto

Secure Integration of Asymmetric and Symmetric Encryption Schemes,  
CRYPTO 1999, JoC 2013

- **$\gamma$ -spread** and **OWCPA-secure** PKC ( $\text{Gen}_0, \text{Enc}_0, \text{Dec}_0$ )
  - **one-time secure** cipher (e.g. one-time pad)
  - **random oracles**  $G$  and  $H$
- construct a PKC which is INDCCA-secure

$$\begin{aligned}\text{Gen} &= \text{Gen}_0 \\ \text{Enc}_{\text{pk}}(\text{pt}; \text{coins}) &= \left( \text{Enc}_{0,\text{pk}}(\text{coins}; \underbrace{H(\text{coins}, \text{ct}_2)}_{\text{new coins}}), \overbrace{\text{pt} \oplus G(\text{coins})}^{\text{ct}_2} \right)\end{aligned}$$

$\text{Dec}_{\text{sk}}(\text{ct}_1, \text{ct}_2)$ :

- 1:  $\text{Dec}_{0,\text{sk}}(\text{ct}_1) \rightarrow \text{coins}$
- 2: **if**  $\text{ct}_1 \neq \text{Enc}_{0,\text{pk}}(\text{coins}; H(\text{coins}, \text{ct}_2))$  **then return**  $\perp$
- 3: **return**  $\text{ct}_2 \oplus G(\text{coins})$

# Security Notions

- **$\gamma$ -spread:**

$$\forall \text{pk}, \text{pt}, \text{ct} \quad \Pr[\text{Enc}_{\text{pk}}(\text{pt}) = \text{ct}] \leq 2^{-\gamma}$$

- **OWCPA-secure:**

secure against decryption under chosen plaintext attacks

# Proof Sketch

OWCPA  $\xrightarrow{\text{FO}}$  INDCCA

(need  $\text{PKC}_0$  to be  $\gamma$ -spread as well)

$\text{Enc}_{\text{pk}}(\text{pt})$ :

- 1: pick  $\sigma$
- 2:  $\text{ct}_2 \leftarrow \text{pt} \oplus G(\sigma)$
- 3:  $\text{ct}_1 \leftarrow \text{Enc}_{0,\text{pk}}(\sigma; H(\sigma, \text{ct}_2))$
- 4: **return**  $(\text{ct}_1, \text{ct}_2)$

$\text{Dec}_{\text{sk}}(\text{ct}_1, \text{ct}_2)$ :

- 1:  $\sigma \leftarrow \text{Dec}_{0,\text{sk}}(\text{ct}_1)$
- 2: **if**  $\sigma = \perp$  **then return**  $\perp$
- 3: **if**  $\text{ct}_1 \neq \text{Enc}_{0,\text{pk}}(\sigma; H(\sigma, \text{ct}_2))$   
**then return**  $\perp$
- 4:  $\text{pt} \leftarrow \text{ct}_2 \oplus G(\sigma)$
- 5: **return**  $\text{pt}$

- modify the decryption oracle so that it does not use  $\text{sk}$  but only the oracle tables:  
if there is no  $(\sigma, \text{ct}_2, h) \in H$  such that  $\text{ct}_1 = \text{Enc}_{0,\text{pk}}(\sigma; h)$ ,  
then return  $\perp$ , otherwise, decrypt by using  $G$
- modify  $F$  and  $G$  on the challenge  $\sigma$  point

# Transform Adapted to Quantum Random Oracles...

Targhi–Unruh, Quantum Security of the Fijisaki-Okamoto and OAEP Transforms,  
TCC 2016

$$\text{OWCPA} \xrightarrow{\text{TU}} \text{INDCCA}$$

(need  $\text{PKC}_0$  to be  $\gamma$ -spread as well)

$\text{Enc}_{\text{pk}}(\text{pt})$ :

- 1: pick  $\sigma$
- 2:  $\text{ct}_2 \leftarrow \text{pt} \oplus G(\sigma)$
- 3:  $\text{ct}_1 \leftarrow \text{Enc}_{0,\text{pk}}(\sigma; H(\sigma, \text{ct}_2))$
- 4:  $\text{ct}_3 \leftarrow H'(\sigma)$
- 5: **return**  $(\text{ct}_1, \text{ct}_2, \text{ct}_3)$

$\text{Dec}_{\text{sk}}(\text{ct}_1, \text{ct}_2, \text{ct}_3)$ :

- 1:  $\sigma \leftarrow \text{Dec}_{0,\text{sk}}(\text{ct}_1)$
- 2: **if**  $\sigma = \perp$  **then return**  $\perp$
- 3: **if**  $\text{ct}_1 \neq \text{Enc}_{0,\text{pk}}(\sigma; H(\sigma, \text{ct}_2))$  **then return**  $\perp$
- 4: **if**  $\text{ct}_3 \neq H'(\sigma)$  **then return**  $\perp$
- 5:  $\text{pt} \leftarrow \text{ct}_2 \oplus G(\sigma)$
- 6: **return**  $\text{pt}$

secure, even with quantum access to the random oracle...

# A Modular Analysis of the FO Transformation

Hofheinz-Hövelmanns-Kiltz TCC 2017

adversary has access to:

**Pco**(ct, pt):  
1: **return**  $1_{pt = \text{Dec}_{\text{sk}}(\text{ct})}$

**Cvo**(ct):  
1: **return**  $1_{\text{Dec}_{\text{sk}}(\text{ct}) \neq \perp}$



$S^\ell$ :

**Enc**<sub>pk</sub>(pt):  
1: pick  $x_1, \dots, x_\ell$   
2:  $\text{ct}_0 \leftarrow pt \oplus F(x_1, \dots, x_\ell)$   
3:  $\text{ct}_i \xleftarrow{\$} \text{Enc}_{0,\text{pk}}(x_i), i = 1, \dots, \ell$   
4: **return**  $(\text{ct}_0, \dots, \text{ct}_\ell)$

**Dec**<sub>sk</sub>(ct<sub>0</sub>, ..., ct<sub>ℓ</sub>):  
5:  $x_i \leftarrow \text{Dec}_{0,\text{sk}}(\text{ct}_i), i = 1, \dots, \ell$   
6:  $pt \leftarrow \text{ct}_0 \oplus F(x_1, \dots, x_\ell)$   
7: **return** pt

$T$ :

**Enc**<sub>pk</sub>(pt):  
1:  $\text{ct} \leftarrow \text{Enc}_{0,\text{pk}}(\text{pt}; G(\text{pt}))$   
2: **return** ct

**Dec**<sub>sk</sub>(ct):  
3:  $\text{pt} \leftarrow \text{Dec}_{0,\text{sk}}(\text{ct})$   
4: **if**  $\text{pt} = \perp$  **then return**  $\perp$   
5: **if**  $\text{ct} \neq \text{Enc}_{0,\text{pk}}(\text{pt}; G(\text{pt}))$  **then return**  $\perp$   
6: **return** pt

$U$ :

**Enc**<sub>pk</sub>(pt):  
1: pick pt at random  
2:  $\text{ct} \xleftarrow{\$} \text{Enc}_{0,\text{pk}}(\text{pt})$   
3:  $K \leftarrow H(\text{pt}, \text{ct})$   
4: **return**  $(K, \text{ct})$

**Dec**<sub>sk</sub>(ct):  
5:  $\text{pt} \leftarrow \text{Dec}_{0,\text{sk}}(\text{ct})$   
6: **if**  $\text{pt} = \perp$  **then return**  $\perp$   
7:  $K \leftarrow H(\text{pt}, \text{ct})$   
8: **return** K

# S Transform

$$\text{OWCPA} \xrightarrow{S^\ell} \text{INDCPA}$$

$\text{Enc}_{\text{pk}}(\text{pt})$ :

- 1: pick  $x_1, \dots, x_\ell$
- 2:  $\text{ct}_0 \leftarrow \text{pt} \oplus F(x_1, \dots, x_\ell)$
- 3:  $\text{ct}_i \xleftarrow{\$} \text{Enc}_{0,\text{pk}}(x_i), i = 1, \dots, \ell$
- 4: **return**  $(\text{ct}_0, \dots, \text{ct}_\ell)$

$\text{Dec}_{\text{sk}}(\text{ct}_0, \dots, \text{ct}_\ell)$ :

- 1:  $x_i \leftarrow \text{Dec}_{0,\text{sk}}(\text{ct}_i), i = 1, \dots, \ell$
- 2:  $\text{pt} \leftarrow \text{ct}_0 \oplus F(x_1, \dots, x_\ell)$
- 3: **return**  $\text{pt}$

The  $\text{OWCPA} \rightarrow \text{INDCPA}$  reduction is loosing a factor  $q^{1/\ell}$  in the advantage, where  $q$  is the number of random oracle queries the adversary can make.

This factor can be huge.

Increase  $\ell$  to make it smaller (but make encryption more costly).

## S Transform (Proof Sketch)

$$\text{OWCPA} \xrightarrow{S^\ell} \text{INDCPA}$$

$\text{Enc}_{\text{pk}}(\text{pt})$ :

- 1: pick  $x_1, \dots, x_\ell$
- 2:  $\text{ct}_0 \leftarrow \text{pt} \oplus F(x_1, \dots, x_\ell)$
- 3:  $\text{ct}_i \xleftarrow{\$} \text{Enc}_{0,\text{pk}}(x_i), i = 1, \dots, \ell$
- 4: **return**  $(\text{ct}_0, \dots, \text{ct}_\ell)$

$\text{Dec}_{\text{sk}}(\text{ct}_0, \dots, \text{ct}_\ell)$ :

- 1:  $x_i \leftarrow \text{Dec}_{0,\text{sk}}(\text{ct}_i), i = 1, \dots, \ell$
- 2:  $\text{pt} \leftarrow \text{ct}_0 \oplus F(x_1, \dots, x_\ell)$
- 3: **return**  $\text{pt}$

- add a failure event that the adversary queries  $F(x_1, \dots, x_\ell)$
- construct an OWCPA game in which the challenge ct is put at a random place and completed
- define  $p_i$  as the probability that one  $F$  query out of the  $q$  ones (taken at random) finds the right  $x_i$  given that  $(x_1, \dots, x_{i-1})$  are found and the failure event occurs
- use  $\frac{1}{\ell} \sum p_i \geq (\prod p_i)^{\frac{1}{\ell}}$  and  $\prod p_i = \frac{1}{q}$

## $T$ Transform (Proof Sketch)

$$\text{INDCPA} \xrightarrow{T} \text{OWPCVA}$$

(need  $\text{PKC}_0$  to be  $\gamma$ -spread as well)

$\text{Enc}_{\text{pk}}(\text{pt})$ :

- 1:  $\text{ct} \leftarrow \text{Enc}_{0,\text{pk}}(\text{pt}; G(\text{pt}))$
- 2: **return**  $\text{ct}$

$\text{Dec}_{\text{sk}}(\text{ct})$ :

- 1:  $\text{pt} \leftarrow \text{Dec}_{0,\text{sk}}(\text{ct})$
- 2: **if**  $\text{pt} = \perp$  **then return**  $\perp$
- 3: **if**  $\text{ct} \neq \text{Enc}_{0,\text{pk}}(\text{pt}; G(\text{pt}))$  **then return**  $\perp$
- 4: **return**  $\text{pt}$

- remove the  $\text{Pco}$  oracles which can be simulated (use correctness)
- get rid of  $\text{Cvo}$  oracles queries which can be simulated from the  $G$  table
- other queries answer 0, but with probability  $2^{-\gamma}$

## $\mathcal{U}$ Transform (Proof Sketch)

$$\text{OWPCVA} \xrightarrow{\mathcal{U}} \text{INDCCA}_{(\text{KEM})}$$

$\text{Enc}_{\text{pk}}$ :

- 1: pick pt at random
- 2:  $\text{ct} \xleftarrow{\$} \text{Enc}_{0,\text{pk}}(\text{pt})$
- 3:  $K \leftarrow H(\text{pt}, \text{ct})$
- 4: **return**  $(K, \text{ct})$

$\text{Dec}_{\text{sk}}(\text{ct})$ :

- 1:  $\text{pt} \leftarrow \text{Dec}_{0,\text{sk}}(\text{ct})$
- 2: **if**  $\text{pt} = \perp$  **then return**  $\perp$
- 3:  $K \leftarrow H(\text{pt}, \text{ct})$
- 4: **return**  $K$

- define a failure event that the right  $(\text{pt}, \text{ct})$  is queried to  $H$
- simulate the decryption query by

$\text{Dec}(\text{ct})$ :

- 1: **if**  $\text{Cvo}(\text{ct}) = 0$  **then return**  $\perp$
- 2: **for all**  $(\text{pt}, \text{ct}, K) \in H$  **do**
- 3:     **if**  $\text{Pco}(\text{pt}, \text{ct}) = 1$  **then return**  $K$
- 4: **end for**
- 5: **return** a random  $K$  and get ready to update  $H$   
(in  $H$ , check if pt would decrypt...)

# Conclusion

- a battery of formal security definitions
- the random oracle model: a tool to idealize hash functions
- games: a methodology to prove security

# References

- **Coron.**  
On the Exact Security of Full Domain Hash.  
In *CRYPTO 2000*, LNCS 1880.
- **Fiat-Shamir.**  
How to Prove Yourself: Practical Solutions to Identification  
and Signature Problems.  
In *CRYPTO 1986*, LNCS 263.
- **Bellare-Rogaway.**  
Random Oracles are Practical: A Paradigm for Designing  
Efficient Protocols.  
In *CCS 1993*, ACM.

# References

- **Pointcheval-Stern.**  
Security Proofs for Signature Schemes.  
In *EUROCRYPT 1996*, LNCS 1070.
- **Shoup.**  
Sequences of Games: a Tool for Taming Complexity in  
Security Proofs.  
Cryptology ePrint Archive 2004/332, IACR.
- **Boneh.**  
The Decision Diffie-Hellman Problem  
In *ANTS 1998*, LNCS 1423.

# Train Yourself

- BLS signature: final exam 2012–13 ex2
- PRF: final exam 2012–13 ex3 (PRF programming)
- distance bounding: final exam 2013–14 ex2
- forking Lemma and Fiat-Shamir: final exam 2013–14 ex3